System Administration Guide: Security Services

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–4557–14
April 2008

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, SunOS, JumpStart, Trusted Solaris, OpenSolaris, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Xylogics product is protected by copyright and licensed to Sun by Xylogics. Xylogics and Annex are trademarks of Xylogics, Inc., Portions of the software copyright 1996 by the Massachusetts Institute of Technology. All rights reserved.

The OPEN LOOK and Sun^{TM} Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED “AS IS” AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2008 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plusieurs brevets américains ou des applications de brevet en attente aux Etats-Unis et dans d'autres pays.

Cette distribution peut comprendre des composants développés par des tierces personnes.

Certaines composants de ce produit peuvent être dérivées du logiciel Berkeley BSD, licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays; elle est licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, le logo Solaris, le logo Java Coffee Cup, docs.sun.com, SunOS, JumpStart, Trusted Solaris, OpenSolaris, Java et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Le produit de Xylogics est protégé par le copyright et autorisé au Sun par Xylogics. Xylogics et Annex sont des marques déposées de Xylogics, Inc.; Copyright 1996 des portions du logiciel par Massachusetts Institute of Technology. Tous droits réservés.

L'interface d'utilisation graphique OPEN LOOK et Sun a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui, en outre, se conforment aux licences écrites de Sun.

Les produits qui font l'objet de cette publication et les informations qu'il contient sont régis par la legislation américaine en matière de contrôle des exportations et peuvent être soumis au droit d'autres pays dans le domaine des exportations et importations. Les utilisations finales, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes chimiques ou biologiques ou pour le nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers des pays sous embargo des Etats-Unis, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exclusive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la legislation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement designés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFACON.

080401@19860

Contents

Preface 23

Part I **Security Overview** 27

1 **Security Services (Overview)** 29

System Security 30

Solaris Cryptographic Services 30

Authentication Services 32

Authentication With Encryption 32

Solaris Auditing 33

Security Policy 33

Part II **System, File, and Device Security** 35

2 **Managing Machine Security (Overview)** 37

Enhancements to Machine Security in the Solaris 10 Release 37

Controlling Access to a Computer System 38

- Maintaining Physical Security 38
- Maintaining Login Control 39

Controlling Access to Devices 44

- Device Policy (Overview) 45
- Device Allocation (Overview) 45

Controlling Access to Machine Resources 46

- Limiting and Monitoring Superuser 46
- Configuring Role-Based Access Control to Replace Superuser 46
- Preventing Unintentional Misuse of Machine Resources 47
- Restricting `setuid` Executable Files 48

Using the Automated Security Enhancement Tool 48
Using the Solaris Security Toolkit 49
Using Solaris Resource Management Features 49
Using Solaris Zones 49
Monitoring Use of Machine Resources 50
Monitoring File Integrity 50
Controlling Access to Files 50
Protecting Files With Encryption 50
Using Access Control Lists 51
Sharing Files Across Machines 51
Restricting `root` Access to Shared Files 51
Controlling Network Access 52
Network Security Mechanisms 52
Authentication and Authorization for Remote Access 53
Firewall Systems 55
Encryption and Firewall Systems 56
Reporting Security Problems 56

3 Controlling Access to Systems (Tasks) 57
Controlling System Access (Task Map) 57
Securing Logins and Passwords (Task Map) 58
Securing Logins and Passwords 58
▼ How to Display a User's Login Status 58
▼ How to Display Users Without Passwords 59
▼ How to Temporarily Disable User Logins 60
▼ How to Monitor Failed Login Attempts 61
▼ How to Monitor All Failed Login Attempts 62
▼ How to Create a Dial-Up Password 63
▼ How to Temporarily Disable Dial-Up Logins 65
Changing the Password Algorithm (Task Map) 65
Changing the Default Algorithm for Password Encryption 66
▼ How to Specify an Algorithm for Password Encryption 66
▼ How to Specify a New Password Algorithm for an NIS Domain 67
▼ How to Specify a New Password Algorithm for an NIS+ Domain 67
▼ How to Specify a New Password Algorithm for an LDAP Domain 68

▼ How to Install a Password Encryption Module From a Third Party 68
Monitoring and Restricting Superuser (Task Map) 69
Monitoring and Restricting Superuser 70
▼ How to Monitor Who Is Using the `su` Command 70
▼ How to Restrict and Monitor Superuser Logins 71
SPARC: Controlling Access to System Hardware (Task Map) 72
Controlling Access to System Hardware 72
▼ How to Require a Password for Hardware Access 72
▼ How to Disable a System's Abort Sequence 73

4 Controlling Access to Devices (Tasks) 75
Configuring Devices (Task Map) 75
Configuring Device Policy (Task Map) 76
Configuring Device Policy 76
▼ How to View Device Policy 76
▼ How to Change the Device Policy on an Existing Device 77
▼ How to Audit Changes in Device Policy 78
▼ How to Retrieve IP MIB-II Information From a `/dev/*` Device 79
Managing Device Allocation (Task Map) 79
Managing Device Allocation 80
▼ How to Make a Device Allocatable 80
▼ How to Authorize Users to Allocate a Device 81
▼ How to View Allocation Information About a Device 82
▼ Forcibly Allocating a Device 82
▼ Forcibly Deallocating a Device 83
▼ How to Change Which Devices Can Be Allocated 83
▼ How to Audit Device Allocation 84
Allocating Devices (Task Map) 85
Allocating Devices 85
▼ How to Allocate a Device 85
▼ How to Mount an Allocated Device 86
▼ How to Deallocate a Device 88
Device Protection (Reference) 89
Device Policy Commands 89
Device Allocation 90

5 Using the Basic Audit Reporting Tool (Tasks) 97
Basic Audit Reporting Tool (Overview) 97
  BART Features 98
  BART Components 98
Using BART (Task Map) 100
Using BART (Tasks) 101
  BART Security Considerations 101
  ▼ How to Create a Manifest 101
  ▼ How to Customize a Manifest 104
  ▼ How to Compare Manifests for the Same System Over Time 107
  ▼ How to Compare Manifests From a Different System With the Manifest of a Control System 110
  ▼ How to Customize a BART Report by Specifying File Attributes 112
  ▼ How to Customize a BART Report by Using a Rules File 113
BART Manifest, Rules File, and Reporting (Reference) 115
  BART Manifest File Format 115
  BART Rules File Format 116
  BART Reporting 117

6 Controlling Access to Files (Tasks) 121
Using UNIX Permissions to Protect Files 121
  Commands for Viewing and Securing Files 121
  File and Directory Ownership 122
  UNIX File Permissions 122
  Special File Permissions (`setuid`, `setgid` and Sticky Bit) 123
  Default `umask` Value 125
  File Permission Modes 125
Using Access Control Lists to Protect Files 127
  ACL Entries for Files 128
  ACL Entries for Directories 129
  Commands for Administering ACLs 129
Preventing Executable Files From Compromising Security 130
Protecting Files (Task Map) 131
Protecting Files With UNIX Permissions (Task Map) 131
  ▼ How to Display File Information 131

- ▼ How to Change the Owner of a File 133
- ▼ How to Change Group Ownership of a File 134
- ▼ How to Change File Permissions in Symbolic Mode 134
- ▼ How to Change File Permissions in Absolute Mode 135
- ▼ How to Change Special File Permissions in Absolute Mode 136

Protecting Files With ACLs (Task Map) 137
- ▼ How to Check if a File Has an ACL 138
- ▼ How to Add ACL Entries to a File 138
- ▼ How to Copy an ACL 140
- ▼ How to Change ACL Entries on a File 140
- ▼ How to Delete ACL Entries From a File 141
- ▼ How to Display ACL Entries for a File 142

Protecting Against Programs With Security Risk (Task Map) 143
- ▼ How to Find Files With Special File Permissions 143
- ▼ How to Disable Programs From Using Executable Stacks 144

7 Using the Automated Security Enhancement Tool (Tasks) 147

Automated Security Enhancement Tool (ASET) 147
- ASET Security Levels 148
- ASET Task List 149
- ASET Execution Log 151
- ASET Reports 152
- ASET Master Files 155
- ASET Environment File (`asetenv`) 156
- Configuring ASET 156
- Restoring System Files Modified by ASET 158
- Network Operation With the NFS System 159
- ASET Environment Variables 160
- ASET File Examples 163

Running ASET (Task Map) 164
- ▼ How to Run ASET Interactively 165
- ▼ How to Run ASET Periodically 166
- ▼ How to Stop Running ASET Periodically 167
- ▼ How to Collect ASET Reports on a Server 167

Troubleshooting ASET Problems 168

ASET Error Messages 168

Part III **Roles, Rights Profiles, and Privileges** 173

8 **Using Roles and Privileges (Overview)** 175
What's New in RBAC? 175
Role-Based Access Control (Overview) 175
  RBAC: An Alternative to the Superuser Model 176
  Solaris RBAC Elements and Basic Concepts 178
  RBAC Authorizations 181
  Authorizations and Privileges 181
  Privileged Applications and RBAC 181
  RBAC Rights Profiles 183
  RBAC Roles 183
  Profile Shell in RBAC 184
  Name Service Scope and RBAC 184
  Security Considerations When Directly Assigning Security Attributes 184
Privileges (Overview) 185
  Privileges Protect Kernel Processes 185
  Privilege Descriptions 186
  Administrative Differences on a System With Privileges 187
  Privileges and System Resources 188
  How Privileges Are Implemented 189
  How Processes Get Privileges 190
  Assigning Privileges 191
  Privileges and Devices 192
  Privileges and Debugging 193

9 **Using Role-Based Access Control (Tasks)** 195
Using RBAC (Task Map) 195
Configuring RBAC (Task Map) 196
Configuring RBAC 197
  ▼ How to Plan Your RBAC Implementation 197
  ▼ How to Create and Assign a Role By Using the GUI 199

▼ How to Create a Role From the Command Line 202
▼ How to Assign a Role to a Local User 205
▼ How to Audit Roles 206
▼ How to Make `root` User Into a Role 207
Using Roles (Task Map) 209
Using Roles 209
▼ How to Assume a Role in a Terminal Window 209
▼ How to Assume a Role in the Solaris Management Console 211
Managing RBAC (Task Map) 213
Managing RBAC 213
▼ How to Change the Properties of a Role 213
▼ How to Create or Change a Rights Profile 215
▼ How to Change the RBAC Properties of a User 218
▼ How to Add RBAC Properties to Legacy Applications 219

10 Role-Based Access Control (Reference) 223
Contents of Rights Profiles 223
Primary Administrator Rights Profile 224
System Administrator Rights Profile 224
Operator Rights Profile 225
Printer Management Rights Profile 225
Basic Solaris User Rights Profile 226
All Rights Profile 227
Order of Rights Profiles 227
Viewing the Contents of Rights Profiles 227
Authorization Naming and Delegation 228
Authorization Naming Conventions 228
Example of Authorization Granularity 228
Delegation Authority in Authorizations 228
Databases That Support RBAC 229
RBAC Database Relationships 229
RBAC Databases and the Name Service 230
`user_attr` Database 231
`auth_attr` Database 232
`prof_attr` Database 233

`exec_attr` Database 234
`policy.conf` File 235
RBAC Commands 236
Commands That Manage RBAC 236
Commands That Require Authorizations 237

11 Privileges (Tasks) 239
Managing and Using Privileges (Task Map) 239
Managing Privileges (Task Map) 239
Managing Privileges 240
▼ How to Determine the Privileges on a Process 240
▼ How to Determine Which Privileges a Program Requires 242
▼ How to Add Privileges to a Command 243
▼ How to Assign Privileges to a User or Role 244
▼ How to Limit a User's or Role's Privileges 245
▼ How to Run a Shell Script With Privileged Commands 247
Determining Your Privileges (Task Map) 247
Determining Your Assigned Privileges 248
▼ How to Determine the Privileges That You Have Been Directly Assigned 248
▼ How to Determine the Privileged Commands That You Can Run 249
▼ How to Determine the Privileged Commands That a Role Can Run 251

12 Privileges (Reference) 255
Administrative Commands for Handling Privileges 255
Files With Privilege Information 256
Privileges and Auditing 257
Prevention of Privilege Escalation 257
Legacy Applications and the Privilege Model 258

Part IV Solaris Cryptographic Services 259

13 Solaris Cryptographic Framework (Overview) 261
What's New in the Solaris Cryptographic Framework? 261
Solaris Cryptographic Framework 262

Terminology in the Solaris Cryptographic Framework 262
Scope of the Solaris Cryptographic Framework 264
Administrative Commands in the Solaris Cryptographic Framework 264
User-Level Commands in the Solaris Cryptographic Framework 265
    Binary Signatures for Third-Party Software 265
Plugins to the Solaris Cryptographic Framework 266
Cryptographic Services and Zones 266

14 Solaris Cryptographic Framework (Tasks) 267
Using the Cryptographic Framework (Task Map) 267
Protecting Files With the Solaris Cryptographic Framework (Task Map) 268
Protecting Files With the Solaris Cryptographic Framework 268
▼ How to Generate a Symmetric Key 268
▼ How to Compute a Digest of a File 270
▼ How to Compute a MAC of a File 271
▼ How to Encrypt and Decrypt a File 273
Administering the Cryptographic Framework (Task Map) 275
Administering the Cryptographic Framework 276
▼ How to List Available Providers 276
▼ How to Add a Software Provider 279
▼ How to Prevent the Use of a User-Level Mechanism 280
▼ How to Prevent the Use of a Kernel Software Provider 282
▼ How to List Hardware Providers 285
▼ How to Disable Hardware Provider Mechanisms and Features 285
▼ How to Refresh or Restart All Cryptographic Services 287

15 Solaris Key Management Framework 289
Managing Public Key Technologies 289
Key Management Framework Utilities 290
KMF Policy Management 290
KMF Keystore Management 290
Using the Key Management Framework (Task Map) 291
Using the Key Management Framework (Tasks) 291
▼ How to Create a Certificate by Using the `pktool gencert` Command 291
▼ How to Import a Certificate Into Your Keystore 293

▼ How to Export a Certificate and Private Key in PKCS #12 Format 294
▼ How to Generate a Passphrase by Using the `pktool setpin` Command 295

Part V **Authentication Services and Secure Communication** 297

16 **Using Authentication Services (Tasks)** 299
Overview of Secure RPC 299
  NFS Services and Secure RPC 299
  DES Encryption With Secure NFS 300
  Kerberos Authentication 300
  Diffie-Hellman Authentication and Secure RPC 300
Administering Secure RPC (Task Map) 304
Administering Authentication With Secure RPC 304
▼ How to Restart the Secure RPC Keyserver 305
▼ How to Set Up a Diffie-Hellman Key for an NIS+ Host 305
▼ How to Set Up a Diffie-Hellman Key for an NIS+ User 306
▼ How to Set Up a Diffie-Hellman Key for an NIS Host 307
▼ How to Set Up a Diffie-Hellman Key for an NIS User 308
▼ How to Share NFS Files With Diffie-Hellman Authentication 309

17 **Using PAM** 311
PAM (Overview) 311
  Benefits of Using PAM 311
  Introduction to the PAM Framework 312
  Changes to PAM for the Solaris 10 Release 313
PAM (Tasks) 314
  PAM (Task Map) 314
  Planning for Your PAM Implementation 315
▼ How to Add a PAM Module 316
▼ How to Prevent Rhost-Style Access From Remote Systems With PAM 316
▼ How to Log PAM Error Reports 317
PAM Configuration (Reference) 317
  PAM Configuration File Syntax 317
  How PAM Stacking Works 318

PAM Stacking Example 321

18 **Using SASL** 323
SASL (Overview) 323
SASL (Reference) 324
SASL Plug-ins 324
SASL Environment Variable 324
SASL Options 325

19 **Using Solaris Secure Shell (Tasks)** 327
Solaris Secure Shell (Overview) 327
Solaris Secure Shell Authentication 328
Solaris Secure Shell in the Enterprise 330
Solaris Secure Shell Enhancements in the Solaris 10 Release 330
Solaris Secure Shell (Task Map) 331
Configuring Solaris Secure Shell (Task Map) 331
Configuring Solaris Secure Shell 331
▼ How to Set Up Host-Based Authentication for Solaris Secure Shell 331
▼ How to Enable Solaris Secure Shell v1 334
▼ How to Configure Port Forwarding in Solaris Secure Shell 334
Using Solaris Secure Shell (Task Map) 335
Using Solaris Secure Shell 336
▼ How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell 336
▼ How to Change the Passphrase for a Solaris Secure Shell Private Key 339
▼ How to Log In to a Remote Host With Solaris Secure Shell 339
▼ How to Reduce Password Prompts in Solaris Secure Shell 340
▼ How to Set Up the `ssh-agent` Command to Run Automatically in CDE 342
▼ How to Use Port Forwarding in Solaris Secure Shell 342
▼ How to Copy Files With Solaris Secure Shell 344
▼ How to Set Up Default Connections to Hosts Outside a Firewall 345

20 **Solaris Secure Shell (Reference)** 347
A Typical Solaris Secure Shell Session 347
Session Characteristics in Solaris Secure Shell 348

- Authentication and Key Exchange in Solaris Secure Shell 348
- Command Execution and Data Forwarding in Solaris Secure Shell 349

Client and Server Configuration in Solaris Secure Shell 350

- Client Configuration in Solaris Secure Shell 350
- Server Configuration in Solaris Secure Shell 350

Keywords in Solaris Secure Shell 350

- Host-Specific Parameters in Solaris Secure Shell 354
- Solaris Secure Shell and Login Environment Variables 354

Maintaining Known Hosts in Solaris Secure Shell 355

Solaris Secure Shell Packages and Initialization 356

Solaris Secure Shell Files 356

Solaris Secure Shell Commands 358

Part VI Kerberos Service 361

21 Introduction to the Kerberos Service 363

What Is the Kerberos Service? 363

How the Kerberos Service Works 364

- Initial Authentication: the Ticket-Granting Ticket 365
- Subsequent Kerberos Authentications 366
- The Kerberos Remote Applications 368
- Kerberos Principals 368
- Kerberos Realms 369
- Kerberos Servers 370

Kerberos Security Services 371

The Components of Various Kerberos Releases 371

- Kerberos Components 372
- Kerberos Additions for the Solaris 10 5/08 Release 373
- Kerberos Additions for the Solaris 10 8/07 Release 373
- Kerberos Additions for the Solaris 10 6/06 Release 374
- Kerberos Enhancements in the Solaris 10 3/05 Release 374
- Kerberos Components in the Solaris 9 Release 376
- SEAM 1.0.2 Components 376
- Kerberos Components in the Solaris 8 Release 377
- SEAM 1.0.1 Components 377

SEAM 1.0 Components ... 378

22 Planning for the Kerberos Service ... 379

Why Plan for Kerberos Deployments? ... 379
Planning Kerberos Realms ... 380
Realm Names ... 380
Number of Realms ... 380
Realm Hierarchy ... 381
Mapping Host Names Onto Realms ... 381
Client and Service Principal Names ... 381
Ports for the KDC and Admin Services ... 382
The Number of Slave KDCs ... 382
Mapping GSS Credentials to UNIX Credentials ... 383
Automatic User Migration to a Kerberos Realm ... 383
Which Database Propagation System to Use ... 383
Clock Synchronization Within a Realm ... 384
Client Configuration Options ... 384
KDC Configuration Options ... 385
Kerberos Encryption Types ... 385
Online Help URL in the Graphical Kerberos Administration Tool ... 386

23 Configuring the Kerberos Service (Tasks) ... 387

Configuring the Kerberos Service (Task Map) ... 387
Configuring Additional Kerberos Services (Task Map) ... 388
Configuring KDC Servers ... 389
▼ How to Automatically Configure a Master KDC ... 390
▼ How to Interactively Configure a Master KDC ... 390
▼ How to Configure a Master KDC ... 392
▼ How to Configure a KDC to Use an LDAP Data Server ... 397
▼ How to Automatically Configure a Slave KDC ... 404
▼ How to Interactively Configure a Slave KDC ... 405
▼ How to Configure a Slave KDC ... 406
Configuring Cross-Realm Authentication ... 410
▼ How to Establish Hierarchical Cross-Realm Authentication ... 410
▼ How to Establish Direct Cross-Realm Authentication ... 411

Configuring Kerberos Network Application Servers 413
- ▼ How to Configure a Kerberos Network Application Server 413

Configuring Kerberos NFS Servers 414
- ▼ How to Configure Kerberos NFS Servers 415
- ▼ How to Create a Credential Table 417
- ▼ How to Add a Single Entry to the Credential Table 417
- ▼ How to Provide Credential Mapping Between Realms 418
- ▼ How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes 419

Configuring Kerberos Clients 421
- Configuring Kerberos Clients (Task Map) 421
- ▼ How to Create a Kerberos Client Installation Profile 422
- ▼ How to Automatically Configure a Kerberos Client 422
- ▼ How to Interactively Configure a Kerberos Client 423
- ▼ How to Manually Configure a Kerberos Client 424
- ▼ How to Access a Kerberos Protected NFS File System as the `root` User 431
- ▼ How to Configure Automatic Migration of Users in a Kerberos Realm 432

Synchronizing Clocks Between KDCs and Kerberos Clients 434

Swapping a Master KDC and a Slave KDC 436
- ▼ How to Configure a Swappable Slave KDC 436
- ▼ How to Swap a Master KDC and a Slave KDC 437

Administering the Kerberos Database 441
- Backing Up and Propagating the Kerberos Database 442
- ▼ How to Back Up the Kerberos Database 443
- ▼ How to Restore the Kerberos Database 444
- ▼ How to Convert a Kerberos Database After a Server Upgrade 445
- ▼ How to Reconfigure a Master KDC to Use Incremental Propagation 446
- ▼ How to Reconfigure a Slave KDC to Use Incremental Propagation 448
- ▼ How to Configure a Slave KDC to Use Full Propagation 449
- ▼ How to Verify That the KDC Servers Are Synchronized 452
- ▼ How to Manually Propagate the Kerberos Database to the Slave KDCs 454
- Setting Up Parallel Propagation 454
- Configuration Steps for Setting Up Parallel Propagation 455
- Administering the Stash File 456
- ▼ How to Remove a Stash File 456

Increasing Security on Kerberos Servers 457
- ▼ How to Enable Only Kerberized Applications 457

▼ How to Restrict Access to KDC Servers 457

24 **Kerberos Error Messages and Troubleshooting** 459
Kerberos Error Messages 459
SEAM Administration Tool Error Messages 459
Common Kerberos Error Messages (A-M) 460
Common Kerberos Error Messages (N-Z) 468
Kerberos Troubleshooting 471
Problems With the Format of the `krb5.conf` File 471
Problems Propagating the Kerberos Database 471
Problems Mounting a Kerberized NFS File System 472
Problems Authenticating as `root` 472
Observing Mapping from GSS Credentials to UNIX Credentials 473

25 **Administering Kerberos Principals and Policies (Tasks)** 475
Ways to Administer Kerberos Principals and Policies 475
SEAM Administration Tool 476
Command-Line Equivalents of the SEAM Tool 476
The Only File Modified by the SEAM Tool 477
Print and Online Help Features of the SEAM Tool 477
Working With Large Lists in the SEAM Tool 478
▼ How to Start the SEAM Tool 478
Administering Kerberos Principals 480
Administering Kerberos Principals (Task Map) 481
Automating the Creation of New Kerberos Principals 481
▼ How to View the List of Kerberos Principals 482
▼ How to View a Kerberos Principal's Attributes 484
▼ How to Create a New Kerberos Principal 486
▼ How to Duplicate a Kerberos Principal 488
▼ How to Modify a Kerberos Principal 488
▼ How to Delete a Kerberos Principal 489
▼ How to Set Up Defaults for Creating New Kerberos Principals 490
▼ How to Modify the Kerberos Administration Privileges 491
Administering Kerberos Policies 493
Administering Kerberos Policies (Task Map) 493

▼ How to View the List of Kerberos Policies 493
▼ How to View a Kerberos Policy's Attributes 495
▼ How to Create a New Kerberos Policy 497
▼ How to Duplicate a Kerberos Policy 499
▼ How to Modify a Kerberos Policy 499
▼ How to Delete a Kerberos Policy 500
SEAM Tool Reference 501
SEAM Tool Panel Descriptions 501
Using the SEAM Tool With Limited Kerberos Administration Privileges 504
Administering Keytab Files 505
Administering Keytab Files (Task Map) 506
▼ How to Add a Kerberos Service Principal to a Keytab File 507
▼ How to Remove a Service Principal From a Keytab File 509
▼ How to Display the Keylist (Principals) in a Keytab File 510
▼ How to Temporarily Disable Authentication for a Service on a Host 510

26 Using Kerberos Applications (Tasks) 513
Kerberos Ticket Management 513
Do You Need to Worry About Tickets? 513
Creating a Kerberos Ticket 514
Viewing Kerberos Tickets 515
Destroying Kerberos Tickets 516
Kerberos Password Management 517
Advice on Choosing a Password 517
Changing Your Password 518
Granting Access to Your Account 520
Kerberos User Commands 522
Overview of Kerberized Commands 522
Forwarding Kerberos Tickets 525
Using Kerberized Commands (Examples) 526

27 The Kerberos Service (Reference) 529
Kerberos Files 529
Kerberos Commands 531
Kerberos Daemons 532

Kerberos Terminology 532

  Kerberos-Specific Terminology 532

  Authentication-Specific Terminology 533

  Types of Tickets 534

How the Kerberos Authentication System Works 537

Gaining Access to a Service Using Kerberos 538

  Obtaining a Credential for the Ticket-Granting Service 538

  Obtaining a Credential for a Server 539

  Obtaining Access to a Specific Service 540

Using Kerberos Encryption Types 541

Using the `gsscred` Table 543

Notable Differences Between Solaris Kerberos and MIT Kerberos 543

Part VII **Solaris Auditing** 545

28 **Solaris Auditing (Overview)** 547

What Is Auditing? 547

How Does Auditing Work? 548

How Is Auditing Related to Security? 549

Audit Terminology and Concepts 550

  Audit Events 551

  Audit Classes and Preselection 551

  Audit Records and Audit Tokens 552

  Audit Files 553

  Audit Storage 554

  Examining the Audit Trail 555

Auditing on a System With Zones 555

Solaris Auditing Enhancements in the Solaris 10 Release 556

29 **Planning for Solaris Auditing** 559

Planning Solaris Auditing (Task Map) 559

Planning Solaris Auditing (Tasks) 560

  ▼ How to Plan Auditing in Zones 560

  ▼ How to Plan Storage for Audit Records 561

▼ How to Plan Who and What to Audit 562
Determining Audit Policy 564
Controlling Auditing Costs 566
Cost of Increased Processing Time of Audit Data 566
Cost of Analysis of Audit Data 567
Cost of Storage of Audit Data 567
Auditing Efficiently 568

30 Managing Solaris Auditing (Tasks) 569
Solaris Auditing (Task Map) 569
Configuring Audit Files (Task Map) 570
Configuring Audit Files (Tasks) 570
▼ How to Modify the `audit_control` File 571
▼ How to Configure `syslog` Audit Logs 573
▼ How to Change a User's Audit Characteristics 575
▼ How to Add an Audit Class 577
▼ How to Change an Audit Event's Class Membership 578
Configuring and Enabling the Auditing Service (Task Map) 579
Configuring and Enabling the Auditing Service (Tasks) 580
▼ How to Create Partitions for Audit Files 580
▼ How to Configure the `audit_warn` Email Alias 583
▼ How to Configure Audit Policy 584
▼ How to Enable the Auditing Service 586
▼ How to Disable the Auditing Service 587
▼ How to Update the Auditing Service 589
Configuring the Auditing Service in Zones (Tasks) 590
▼ How to Configure All Zones Identically for Auditing 590
▼ How to Configure Per-Zone Auditing 592
Managing Audit Records (Task Map) 594
Managing Audit Records 594
▼ How to Display Audit Record Formats 594
▼ How to Merge Audit Files From the Audit Trail 596
▼ How to Select Audit Events From the Audit Trail 598
▼ How to View the Contents of Binary Audit Files 600
▼ How to Clean Up a `not_terminated` Audit File 602

▼ How to Prevent Audit Trail Overflow ... 603

31 Solaris Auditing (Reference) ... 605

Audit Commands ... 605

`auditd` Daemon ... 606

`audit` Command ... 606

`bsmrecord` Command ... 607

`auditreduce` Command ... 607

`praudit` Command ... 609

`auditconfig` Command ... 610

Files Used in the Auditing Service ... 611

`system` File ... 611

`syslog.conf` File ... 611

`audit_class` File ... 611

`audit_control` File ... 612

`audit_event` File ... 613

`audit_startup` Script ... 613

`audit_user` Database ... 613

`audit_warn` Script ... 614

`bsmconv` Script ... 616

Rights Profiles for Administering Auditing ... 616

Auditing and Solaris Zones ... 616

Audit Classes ... 617

Definitions of Audit Classes ... 617

Audit Class Syntax ... 618

Audit Policy ... 620

Process Audit Characteristics ... 620

Audit Trail ... 620

Conventions for Binary Audit File Names ... 621

Binary Audit File Names ... 621

Binary Audit File Timestamps ... 622

Audit Record Structure ... 622

Audit Record Analysis ... 622

Audit Token Formats ... 623

`acl` Token ... 625

`arbitrary` Token (Obsolete) 625
`arg` Token 626
`attribute` Token 626
`cmd` Token 627
`exec_args` Token 627
`exec_env` Token 628
`exit` Token (Obsolete) 628
`file` Token 628
`group` Token (Obsolete) 629
`groups` Token 629
`header` Token 629
`in_addr` Token 630
`ip` Token (Obsolete) 630
`ipc` Token 631
`ipc_perm` Token 631
`iport` Token 632
`opaque` Token (Obsolete) 632
`path` Token 632
`path_attr` Token 633
`privilege` Token 633
`process` Token 634
`return` Token 635
`sequence` Token 636
`socket` Token 636
`subject` Token 637
`text` Token 638
`trailer` Token 639
`uauth` Token 639
`zonename` Token 639

Glossary 641

Index 651

Preface

System Administration Guide: Security Services is part of a multivolume set that covers a significant part of the Solaris™ Operating System (Solaris OS) administration information. This book assumes that you have already installed the Solaris 10 5/08 release, and you have set up any networking software that you plan to use. The Solaris OS is part of the Solaris product family, which includes many features, such as the Solaris Common Desktop Environment (CDE).

Note – This Solaris release supports systems that use the SPARC® and x86 families of processor architectures: UltraSPARC®, SPARC64, AMD64, Pentium, and Xeon EM64T. The supported systems appear in the *Solaris 10 Hardware Compatibility List* at `http://www.sun.com/bigadmin/hcl`. This document cites any implementation differences between the platform types.

In this document these x86 related terms mean the following:

- "x86" refers to the larger family of 64-bit and 32-bit x86 compatible products.
- "x64" points out specific 64-bit information about AMD64 or EM64T systems.
- "32-bit x86" points out specific 32-bit information about x86 based systems.

For supported systems, see the *Solaris 10 Hardware Compatibility List*.

Who Should Use This Book

This book is intended for anyone who is responsible for administering one or more systems that run the Solaris 10 5/08 release. To use this book, you should have more than two years of UNIX® system administration experience. Attending training courses in UNIX system administration might be helpful.

How the System Administration Volumes Are Organized

Here is a list of the topics that are covered by the volumes of the System Administration Guides.

Book Title	Topics
System Administration Guide: Basic Administration	User accounts and groups, server and client support, shutting down and booting a system, managing services, and managing software (packages and patches)
System Administration Guide: Advanced Administration	Terminals and modems, system resources (disk quotas, accounting, and crontabs), system processes, and troubleshooting Solaris software problems
System Administration Guide: Devices and File Systems	Removable media, disks and devices, file systems, and backing up and restoring data
System Administration Guide: IP Services	TCP/IP network administration, IPv4 and IPv6 address administration, DHCP, IPsec, IKE, Solaris IP filter, Mobile IP, IP network multipathing (IPMP), and IPQoS
System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)	DNS, NIS, and LDAP naming and directory services, including transitioning from NIS to LDAP and transitioning from NIS+ to LDAP
System Administration Guide: Naming and Directory Services (NIS+)	NIS+ naming and directory services
System Administration Guide: Network Services	Web cache servers, time-related services, network file systems (NFS and Autofs), mail, SLP, and PPP
System Administration Guide: Security Services	Auditing, device management, file security, BART, Kerberos services, PAM, Solaris Cryptographic Framework, privileges, RBAC, SASL, and Solaris Secure Shell
System Administration Guide: Virtualization Using the Solaris Operating System	Resource management topics projects and tasks, extended accounting, resource controls, fair share scheduler (FSS), physical memory control using the resource capping daemon (rcapd), and resource pools; virtualization using Solaris Zones software partitioning technology and `lx` branded zones
ZFS Administration Guide	ZFS storage pool and file system creation and management, snapshots, clones, backups, using access control lists (ACLs) to protect ZFS files, using ZFS on a Solaris system with zones installed, emulated volumes, and troubleshooting and data recovery
Solaris Trusted Extensions Administrator's Procedures	System administration that is specific to a Solaris Trusted Extensions system

Book Title	Topics
Solaris Trusted Extensions Configuration Guide	Starting with the Solaris 10 5/08 release, describes how to plan for, enable, and initially configure Solaris Trusted Extensions
System Administration Guide: Solaris Printing	Solaris printing topics and tasks, using services, tools, protocols, and technologies to set up and administer printing services and printers

Related Third-Party Web Site References

Third party URLs are referenced in this document and provide additional, related information.

Sun is not responsible for the availability of third-party web sites mentioned in this document. Sun does not endorse and is not responsible or liable for any content, advertising, products, or other materials that are available on or through such sites or resources. Sun will not be responsible or liable for any actual or alleged damage or loss caused by or in connection with the use of or reliance on any such content, goods, or services that are available on or through such sites or resources.

Documentation, Support, and Training

The Sun web site provides information about the following additional resources:

- Documentation (`http://www.sun.com/documentation/`)
- Support (`http://www.sun.com/support/`)
- Training (`http://www.sun.com/training/`)

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Meaning	Example
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*. A *cache* is a copy that is stored locally. Do *not* save the file. **Note:** Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows the default UNIX system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell	`machine_name%`
C shell for superuser	`machine_name#`
Bourne shell and Korn shell	`$`
Bourne shell and Korn shell for superuser	`#`

PART I

Security Overview

This book focuses on the features that enhance security in the Solaris Operating System. This book is intended for system administrators and users of these security features. The overview chapter introduces the topics in the book.

◆ ◆ ◆

CHAPTER 1

Security Services (Overview)

To maintain the security of the Solaris Operating System (Solaris OS), Solaris software provides the following features:

- "System Security" on page 30 – The ability to prevent intrusion, to protect machine resources and devices from misuse, and to protect files from malicious modification or unintentional modification by users or intruders

  For a general discussion of system security, see Chapter 2, "Managing Machine Security (Overview)."
- "Solaris Cryptographic Services" on page 30 – The ability to scramble data so that only the sender and the designated receiver can read the contents, and to manage cryptographic providers and public key objects
- "Authentication Services" on page 32 – The ability to securely identify a user, which requires the user's name and some form of proof, typically a password
- "Authentication With Encryption" on page 32 – The ability to ensure that authenticated parties can communicate without interception, modification, or spoofing
- "Solaris Auditing" on page 33 – The ability to identify the source of security changes to the system, including file access, security-related system calls, and authentication failures
- "Security Policy" on page 33 – The design and implementation of security guidelines for a computer or network of computers

System Security

System security ensures that the system's resources are used properly. Access controls can restrict who is permitted access to resources on the system. The Solaris OS features for system security and access control include the following:

- **Login administration tools –** Commands for monitoring and controlling a user's ability to log in. See "Securing Logins and Passwords (Task Map)" on page 58.
- **Hardware access –** Commands for limiting access to the PROM, and for restricting who can boot the system. See "SPARC: Controlling Access to System Hardware (Task Map)" on page 72.
- **Resource access –** Tools and strategies for maximizing the appropriate use of machine resources while minimizing the misuse of those resources. See "Controlling Access to Machine Resources" on page 46.
- **Role-based access control (RBAC) –** An architecture for creating special, restricted user accounts that are permitted to perform specific administrative tasks. See "Role-Based Access Control (Overview)" on page 175.
- **Privileges –** Discrete rights on processes to perform operations. These process rights are enforced in the kernel. See "Privileges (Overview)" on page 185.
- **Device management –** Device *policy* additionally protects devices that are already protected by UNIX permissions. Device *allocation* controls access to peripheral devices, such as a microphone or CD-ROM drive. Upon deallocation, device-clean scripts can then erase any data from the device. See "Controlling Access to Devices" on page 44.
- **Basic Audit Reporting Tool (BART) –** A snapshot, called a *manifest*, of the file attributes of files on a system. By comparing the manifests across systems or on one system over time, changes to files can be monitored to reduce security risks. See Chapter 5, "Using the Basic Audit Reporting Tool (Tasks)."
- **File permissions –** Attributes of a file or directory. Permissions restrict the users and groups that are permitted to read, write, or execute a file, or search a directory. See Chapter 6, "Controlling Access to Files (Tasks)."
- **Security enhancement scripts –** Through the use of scripts, many system files and parameters can be adjusted to reduce security risks. See Chapter 7, "Using the Automated Security Enhancement Tool (Tasks)."

Solaris Cryptographic Services

Cryptography is the science of encrypting and decrypting data. Cryptography is used to insure integrity, privacy, and authenticity. Integrity means that the data has not been altered. Privacy means that the data is not readable by others. Authenticity for data means that what was delivered is what was sent. User authentication means that the user has supplied one or more proofs of identity. Authentication mechanisms mathematically verify the source of the data or the proof of identity. Encryption mechanisms scramble data so that the data is not readable by a casual observer. Cryptographic services provide authentication and encryption mechanisms to applications and users.

Cryptographic algorithms use hashing, chaining, and other mathematical techniques to create ciphers that are difficult to break. Authentication mechanisms require that the sender and the receiver compute an identical number from the data. Encryption mechanisms rely on the sender and the receiver sharing information about the method of encryption. This information enables only the receiver and the sender to decrypt the message. The Solaris OS provides a centralized cryptographic framework, and provides encryption mechanisms that are tied to particular applications.

- **Solaris™ Cryptographic Framework –** A central framework of cryptographic services for kernel-level and user-level consumers. Uses include passwords, IPsec, and third-party applications. The cryptographic framework includes a number of software encryption modules. The framework enables you to specify which software encryption modules or hardware encryption sources an application can use. The framework is built on the PKCS #11 v2 library. This library is implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki). The library provides an API for third-party developers to plug in the cryptographic requirements for their applications. See Chapter 13, "Solaris Cryptographic Framework (Overview)."
- **Encryption mechanisms per application –**
  - For the use of DES in Secure RPC, see "Overview of Secure RPC" on page 299.
  - For the use of DES, 3DES, AES, and ARCFOUR in the Kerberos service, see Chapter 21, "Introduction to the Kerberos Service."
  - For the use of RSA, DSA, and ciphers such as Blowfish in Solaris Secure Shell, see Chapter 19, "Using Solaris Secure Shell (Tasks)."
  - For the use of cryptographic algorithms in passwords, see "Changing the Password Algorithm (Task Map)" on page 65.

Starting in the Solaris 10 8/07 release, the Key Management Framework (KMF) provides a central utility for managing public key objects, including policy, keys, and certificates. KMF manages these objects for OpenSSL, NSS, and PKCS #11 public key technologies. See Chapter 15, "Solaris Key Management Framework."

Authentication Services

Authentication is a mechanism that identifies a user or service based on predefined criteria. Authentication services range from simple name-password pairs to more elaborate challenge-response systems, such as smart cards and biometrics. Strong authentication mechanisms rely on a user supplying information that only that person knows, and a personal item that can be verified. A user name is an example of information that the person knows. A smart card or a fingerprint, for example, can be verified. The Solaris features for authentication include the following:

- **Secure RPC** – An authentication mechanism that uses the Diffie-Hellman protocol to protect NFS mounts and a name service, such as NIS or NIS+. See "Overview of Secure RPC" on page 299.
- **Pluggable Authentication Module (PAM)** – A framework that enables various authentication technologies to be plugged into a system entry service without recompiling the service. Some of the system entry services include `login` and `ftp`. See Chapter 17, "Using PAM."
- **Simple Authentication and Security Layer (SASL)** – A framework that provides authentication and security services to network protocols. See Chapter 18, "Using SASL."
- **Solaris Secure Shell** – A secure remote login and transfer protocol that encrypts communications over an insecure network. See Chapter 19, "Using Solaris Secure Shell (Tasks)."
- **Kerberos service** – A client-server architecture that provides encryption with authentication. See Chapter 21, "Introduction to the Kerberos Service."
- **Solaris smart card** – A plastic card with a microprocessor and memory that can be used with a card reader to access systems. See *Solaris Smartcard Administration Guide*.

Authentication With Encryption

Authentication with encryption is the basis of secure communication. Authentication helps ensure that the source and the destination are the intended parties. Encryption codes the communication at the source, and decodes the communication at the destination. Encryption prevents intruders from reading any transmissions that the intruders might manage to intercept. The Solaris features for secure communication include the following:

- **Solaris Secure Shell** – A protocol for protecting data transfers and interactive user network sessions from eavesdropping, session hijacking, and "man-in-the-middle" attacks. Strong authentication is provided through public key cryptography. X windows services and other network services can be tunneled safely over Secure Shell connections for additional protection. See Chapter 19, "Using Solaris Secure Shell (Tasks)."
- **Kerberos service** – A client-server architecture that provides authentication with encryption. See Chapter 21, "Introduction to the Kerberos Service."

- **Internet Protocol Security Architecture (IPsec)** – An architecture that provides IP datagram protection. Protections include confidentiality, strong integrity of the data, data authentication, and partial sequence integrity. See Chapter 19, "IP Security Architecture (Overview)," in *System Administration Guide: IP Services*.

Solaris Auditing

Auditing is a fundamental concept of system security and maintainability. Auditing is the process of examining the history of actions and events on a system to determine what happened. The history is kept in a log of what was done, when it was done, by whom, and what was affected. See Chapter 28, "Solaris Auditing (Overview)."

Security Policy

The phrase security policy, or policy, is used throughout this book to refer to an organization's security guidelines. Your site's security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. Security technologies such as Solaris Secure Shell, authentication, RBAC, authorization, privileges, and resource control provide measures to protect information.

Some security technologies also use the word policy when describing specific aspects of their implementation. For example, Solaris auditing uses audit policy options to configure some aspects of auditing policy. The following table points to glossary, man page, and information on features that use the word policy to describe specific aspects of their implementation.

TABLE 1–1 Use of Policy in the Solaris OS

Glossary Definition	Selected Man Pages	Further Information
audit policy	`audit_control`(4), `audit_user`(4), `auditconfig`(1M)	Chapter 28, "Solaris Auditing (Overview),"
policy in the cryptographic framework	`cryptoadm`(1M)	Chapter 13, "Solaris Cryptographic Framework (Overview),"
device policy	`getdevpolicy`(1M)	"Controlling Access to Devices" on page 44
Kerberos policy	`krb5.conf`(4)	Chapter 25, "Administering Kerberos Principals and Policies (Tasks),"

TABLE 1–1 Use of Policy in the Solaris OS *(Continued)*

Glossary Definition	Selected Man Pages	Further Information
network policies	`ipfilter`(5), `ifconfig`(1M), `ike.config`(4), `ipsecconf`(1M), `routeadm`(1M)	Part III, "IP Security," in *System Administration Guide: IP Services*
password policy	`passwd`(1), `nsswitch.conf`(4), `crypt.conf`(4), `policy.conf`(4)	"Maintaining Login Control" on page 39
policy for public key technologies	`kmfcfg`(1)	Chapter 15, "Solaris Key Management Framework,"
RBAC policy	`rbac`(5)	"`exec_attr` Database" on page 234

PART II

System, File, and Device Security

This section covers security that can be configured on a non-networked system. The chapters discuss planning, monitoring, and controlling access to the disk, to files, and to peripheral devices.

◆ ◆ ◆

CHAPTER 2

Managing Machine Security (Overview)

Keeping a machine's information secure is an important system administration responsibility. This chapter provides overview information about managing machine security.

The following is a list of the overview information in this chapter.

- "Enhancements to Machine Security in the Solaris 10 Release" on page 37
- "Controlling Access to a Computer System" on page 38
- "Controlling Access to Devices" on page 44
- "Controlling Access to Machine Resources" on page 46
- "Controlling Access to Files" on page 50
- "Controlling Network Access" on page 52
- "Reporting Security Problems" on page 56

Enhancements to Machine Security in the Solaris 10 Release

Since the Solaris 9 release, the following features have been introduced to enhance system security:

- Strong password encryption is available and configurable. For more information, see "Password Encryption" on page 40.
- Device policy is enforced with privileges. For more information, see "Device Policy (Overview)" on page 45.

  For device allocation, the `/etc/security/dev` directory might not be supported in future releases of the Solaris OS.
- The Basic Audit Reporting Tool (BART) can monitor the authenticity of the files on your system. For more information, see Chapter 5, "Using the Basic Audit Reporting Tool (Tasks)."
- Files can be protected with strong encryption. For more information, see "Protecting Files With Encryption" on page 50.

- Privileges enforce process rights at the kernel level. For more information, see "Privileges (Overview)" on page 185.
- The Solaris Cryptographic Framework centralizes cryptographic services for providers and for consumers. For more information, see Chapter 13, "Solaris Cryptographic Framework (Overview)."
- The PAM framework provides functionality for many programs, such as Solaris Secure Shell. For more information, see "Changes to PAM for the Solaris 10 Release" on page 313.
- Solaris zones and resource management control access to machine resources. For more information, see *System Administration Guide: Virtualization Using the Solaris Operating System*.

Controlling Access to a Computer System

In the workplace, all machines that are connected to a server can be thought of as one large multifaceted system. You are responsible for the security of this larger system. You need to defend the network from outsiders who are trying to gain access to the network. You also need to ensure the integrity of the data on the machines within the network.

At the file level, the Solaris OS provides standard security features that you can use to protect files, directories, and devices. At the system and network levels, the security issues are mostly the same. The first line of security defense is to control access to your system.

You can control and monitor system access by doing the following:

- "Maintaining Physical Security" on page 38
- "Maintaining Login Control" on page 39
- "Controlling Access to Devices" on page 44
- "Controlling Access to Machine Resources" on page 46
- "Controlling Access to Files" on page 50
- "Controlling Network Access" on page 52
- "Reporting Security Problems" on page 56

Maintaining Physical Security

To control access to your system, you must maintain the physical security of your computing environment. For instance, a system that is logged in and left unattended is vulnerable to unauthorized access. An intruder can gain access to the operating system and to the network. The computer's surroundings and the computer hardware should be physically protected from unauthorized access.

You can protect a SPARC system from unauthorized access to the hardware settings. Use the `eeprom` command to require a password to access the PROM. For more information, see "How to Require a Password for Hardware Access" on page 72.

Maintaining Login Control

You also must prevent unauthorized logins to a system or the network, which you can do through password assignment and login control. All accounts on a system should have a password. A password is a simple authentication mechanism. An account without a password makes your entire network accessible to an intruder who guesses a user name. A strong password algorithm protects against brute force attacks.

When a user logs in to a system, the `login` command checks the appropriate name service or directory service database according to the information that is listed in the `/etc/nsswitch.conf` file. This file can include the following entries:

- `files` – Designates the `/etc` files on the local system
- `ldap` – Designates the LDAP directory service on the LDAP server
- `nis` – Designates the NIS database on the NIS master server
- `nisplus` – Designates the NIS+ database on the NIS+ root server

For a description of the `nsswitch.conf` file, see the `nsswitch.conf`(4) man page. For information about naming services and directory services, see the *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* or the *System Administration Guide: Naming and Directory Services (NIS+)*.

The `login` command verifies the user name and password that were supplied by the user. If the user name is not in the password file, the `login` command denies access to the system. If the password is not correct for the user name that was specified, the `login` command denies access to the system. When the user supplies a valid user name and its corresponding password, the system grants the user access to the system.

PAM modules can streamline login to applications after a successful system login. For more information, see Chapter 17, "Using PAM."

Sophisticated authentication and authorization mechanisms are available on Solaris systems. For a discussion of authentication and authorization mechanisms at the network level, see "Authentication and Authorization for Remote Access" on page 53.

Managing Password Information

When users log in to a system, they must supply both a user name and a password. Although logins are publicly known, passwords must be kept secret. Passwords should be known only to each user. You should ask your users to choose their passwords carefully. Users should change their passwords often.

Passwords are initially created when you set up a user account. To maintain security on user accounts, you can set up password aging to force users to routinely change their passwords. You can also disable a user account by locking the password. For detailed information about administering passwords, see Chapter 4, "Managing User Accounts and Groups (Overview)," in *System Administration Guide: Basic Administration* and the `passwd`(1) man page.

Local Passwords

If your network uses local files to authenticate users, the password information is kept in the system's `/etc/passwd` and `/etc/shadow` files. The user name and other information are kept in the password file `/etc/passwd`. The encrypted password itself is kept in a separate *shadow* file, `/etc/shadow`. This security measure prevents a user from gaining access to the encrypted passwords. While the `/etc/passwd` file is available to anyone who can log in to a system, only superuser or an equivalent role can read the `/etc/shadow` file. You can use the `passwd` command to change a user's password on a local system.

NIS and NIS+ Passwords

If your network uses NIS to authenticate users, password information is kept in the NIS password map. NIS does not support password aging. You can use the command `passwd -r nis` to change a user's password that is stored in an NIS password map.

If your network uses NIS+ to authenticate users, password information is kept in the NIS+ database. Information in the NIS+ database can be protected by restricting access to authorized users only. You can use the `passwd -r nisplus` command to change a user's password that is stored in an NIS+ database.

LDAP Passwords

The Solaris LDAP naming service stores password information and shadow information in the `ou=people` container of the LDAP directory tree. On the Solaris LDAP naming service client, you can use the `passwd -r ldap` command to change a user's password. The LDAP naming service stores the password in the LDAP repository.

In the Solaris 10 release, password policy is enforced on the Sun Java™ System Directory Server. Specifically, the client's `pam_ldap` module follows the password policy controls that are enforced on the Sun Java System Directory Server. For more information, see "LDAP Naming Services Security Model" in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

Password Encryption

Strong password encryption provides an early barrier against attack. Solaris software provides four password encryption algorithms. The two MD5 algorithms and the Blowfish algorithm provide more robust password encryption than the UNIX algorithm.

Password Algorithm Identifiers

You specify the algorithms configuration for your site in the `/etc/security/policy.conf` file. In the `policy.conf` file, the algorithms are named by their identifier, as shown in the following table.

TABLE 2–1 Password Encryption Algorithms

Identifier	Description	Algorithm Man Page
`1`	The MD5 algorithm that is compatible with MD5 algorithms on BSD and Linux systems.	`crypt_bsdmd5`(5)
`2a`	The Blowfish algorithm that is compatible with the Blowfish algorithm on BSD systems.	`crypt_bsdbf`(5)
`md5`	The Sun MD5 algorithm, which is considered stronger than the BSD and Linux version of MD5.	`crypt_sunmd5`(5)
`__unix__`	The traditional UNIX encryption algorithm. This algorithm is the default module in the `policy.conf` file.	`crypt_unix`(5)

Algorithms Configuration in the `policy.conf` File

The following shows the default algorithms configuration in the `policy.conf` file:

```
#
...
# crypt(3c) Algorithms Configuration
#
# CRYPT_ALGORITHMS_ALLOW specifies the algorithms that are allowed to
# be used for new passwords.  This is enforced only in crypt_gensalt(3c).
#
CRYPT_ALGORITHMS_ALLOW=1,2a,md5

# To deprecate use of the traditional unix algorithm, uncomment below
# and change CRYPT_DEFAULT= to another algorithm.  For example,
# CRYPT_DEFAULT=1 for BSD/Linux MD5.
#
#CRYPT_ALGORITHMS_DEPRECATE=__unix__

# The Solaris default is the traditional UNIX algorithm.  This is not
# listed in crypt.conf(4) since it is internal to libc.  The reserved
# name __unix__ is used to refer to it.
#
CRYPT_DEFAULT=__unix__
...
```

When you change the value for `CRYPT_DEFAULT`, the passwords of new users are encrypted with the algorithm that is associated with the new value. When current users change their passwords, how their old password was encrypted affects which algorithm is used to encrypt the new password.

For example, assume that `CRYPT_ALGORITHMS_ALLOW=1,2a,md5` and `CRYPT_DEFAULT=1`. The following table shows which algorithm would be used to generate the encrypted password.

Identifier = Password Algorithm		
Initial Password	**Changed Password**	**Explanation**
1 = `crypt_bsdmd5`	Uses same algorithm	The `1` identifier is also the value of `CRYPT_DEFAULT`. The user's password continues to be encrypted with the `crypt_bsdmd5` algorithm.
`2a` = `crypt_bsdbf`	Uses same algorithm	The `2a` identifier is in the `CRYPT_ALGORITHMS_ALLOW` list. Therefore, the new password is encrypted with the `crypt_bsbdf` algorithm.
`md5` = `crypt_md5`	Uses same algorithm	The `md5` identifier is in the `CRYPT_ALGORITHMS_ALLOW` list. Therefore, the new password is encrypted with the `crypt_md5` algorithm.
`__unix__` = `crypt_unix`	Uses `crypt_bsdmd5` algorithm	The `__unix__` identifier is not in the `CRYPT_ALGORITHMS_ALLOW` list. Therefore, the `crypt_unix` algorithm cannot be used. The new password is encrypted with the `CRYPT_DEFAULT` algorithm.

For more information on configuring the algorithm choices, see the `policy.conf`(4) man page. To specify password encryption algorithms, see "Changing the Password Algorithm (Task Map)" on page 65.

Special System Logins

Two common ways to access a system are by using a conventional user login, or by using the `root` login. In addition, a number of special *system* logins enable a user to run administrative commands without using the `root` account. As system administrator, you assign passwords to these login accounts.

The following table lists some system login accounts and their uses. The system logins perform special functions. Each login has its own group identification number (GID). Each login should have its own password, which should be divulged on a need-to-know basis.

TABLE 2–2 System Login Accounts and Their Uses

Login Account	GID	Use
`root`	`0`	Has almost no restrictions. Overrides all other logins, protections, and permissions. The `root` account has access to the entire system. The password for the `root` login should be very carefully protected. The `root` account, superuser, owns most of the Solaris commands.
`daemon`	`1`	Controls background processing.
`bin`	`2`	Owns some Solaris commands.
`sys`	`3`	Owns many system files.

TABLE 2–2 System Login Accounts and Their Uses *(Continued)*

Login Account	GID	Use
`adm`	4	Owns certain administrative files.
`lp`	71	Owns the object data files and spooled data files for the printer.
`uucp`	5	Owns the object data files and spooled data files for UUCP, the UNIX-to-UNIX copy program.
`nuucp`	9	Is used by remote systems to log in to the system and start file transfers.

Remote Logins

Remote logins offer a tempting avenue for intruders. The Solaris OS provides several commands to monitor, limit, and disable remote logins. For procedures, see "Securing Logins and Passwords (Task Map)" on page 58.

By default, remote logins cannot gain control or read certain system devices, such as the system mouse, keyboard, frame buffer, or audio device. For more information, see the `logindevperm`(4) man page.

Dial-Up Logins

When a computer can be accessed through a modem or a dial-up port, you can add an extra layer of security. You can require a *dial-up password* for users who access a system through a modem or dial-up port. The dial-up password is an additional password that a user must supply before being granted access to the system.

Only superuser or a role of equivalent capabilities can create or change a dial-up password. To ensure the integrity of the system, the password should be changed about once a month. The most effective use of this feature is to require a dial-up password to gain access to a gateway system. To set up dial-up passwords, see "How to Create a Dial-Up Password" on page 63.

Two files are involved in creating a dial-up password, `/etc/dialups` and `/etc/d_passwd`. The `dialups` file contains a list of ports that require a dial-up password. The `d_passwd` file contains a list of shell programs that require an encrypted password as the additional dial-up password. The information in these two files is processed as follows:

- If the user's login shell in `/etc/passwd` matches an entry in `/etc/d_passwd`, the user must supply a dial-up password.
- If the user's login shell in `/etc/passwd` is not found in `/etc/d_passwd`, the user must supply the default password. The default password is the entry for `/usr/bin/sh`.
- If the login shell field in `/etc/passwd` is empty, the user must supply the default password. The default password is the entry for `/usr/bin/sh`.
- If `/etc/d_passwd` has no entry for `/usr/bin/sh`, then those users whose login shell field in `/etc/passwd` is empty or does not match any entry in `/etc/d_passwd` are not prompted for a dial-up password.

- Dial-up logins are disabled if `/etc/d_passwd` has the `/usr/bin/sh:*:` entry only.

Controlling Access to Devices

Peripheral devices that are attached to a computer system pose a security risk. Microphones can pick up conversations and transmit them to remote systems. CD-ROMs can leave their information behind for reading by the next user of the CD-ROM device. Printers can be accessed remotely. Devices that are integral to the system can also present security issues. For example, network interfaces such as `hme0` are considered integral devices.

Solaris software provides two methods of controlling access to devices. *Device policy* restricts or prevents access to devices that are integral to the system. Device policy is enforced in the kernel. *Device allocation* restricts or prevents access to peripheral devices. Device allocation is enforced at user allocation time.

Device policy uses privileges to protect selected devices in the kernel. For example, the device policy on network interfaces such as `hme` requires all privileges for reading or writing.

Device allocation uses authorizations to protect peripheral devices, such as printers or microphones. By default, device allocation is not enabled. Once enabled, device allocation can be configured to prevent the use of a device or to require authorization for access to the device. When a device is allocated for use, no other user can access the device until the current user deallocates it.

A Solaris system can be configured in several areas to control access to devices:

- **Set device policy –** In the Solaris 10 release, you can require that the process that is accessing a particular device be running with a set of privileges. Processes without those privileges cannot use the device. At boot time, Solaris software configures device policy. Third-party drivers can be configured with device policy during installation. After installation, you, as the system administrator can add device policy to a device.
- **Make devices allocatable –** When you enable device allocation, you can restrict the use of a device to one user at a time. You can further require that the user fulfill some security requirements. For example, you can require that the user be authorized to use the device.
- **Prevent devices from being used –** You can prevent the use of a device, such as a microphone, by any user on a computer system. A computer kiosk might be a good candidate for making certain devices unavailable for use.
- **Confine a device to a particular zone –** You can assign the use of a device to a non-global zone. For more information, see “Device Use in Non-Global Zones” in *System Administration Guide: Virtualization Using the Solaris Operating System*. For a more general discussion of devices and zones, see “Configured Devices in Zones” in *System Administration Guide: Virtualization Using the Solaris Operating System*.

Device Policy (Overview)

The device policy mechanism enables you to specify that processes that open a device require certain privileges. Devices that are protected by device policy can only be accessed by processes that are running with the privileges that the device policy specifies. The Solaris OS provides default device policy. For example, network interfaces such as `hme0` require that the processes that access the interface be running with the `net_rawaccess` privilege. The requirement is enforced in the kernel. For more information about privileges, see "Privileges (Overview)" on page 185.

In earlier Solaris OS releases, device nodes were protected by file permissions alone. For example, devices owned by group `sys` could be opened only by members of group `sys`. In the Solaris 10 release, file permissions do not predict who can open a device. Instead, devices are protected with file permissions *and* with device policy. For example, the `/dev/ip` file has `666` permissions. However, the device can only be opened by a process with the appropriate privileges.

The configuration of device policy can be audited. The `AUE_MODDEVPLCY` audit event records changes in device policy.

For more information about device policy, see the following:

- "Configuring Device Policy (Task Map)" on page 76
- "Device Policy Commands" on page 89
- "Privileges and Devices" on page 192

Device Allocation (Overview)

The device allocation mechanism enables you to restrict access to a peripheral device, such as a CD-ROM. You manage the mechanism locally. If device allocation is not enabled, peripheral devices are protected only by file permissions. For example, by default, peripheral devices are available for the following uses:

- Any user can read and write to a diskette or CD-ROM.
- Any user can attach a microphone.
- Any user can access an attached printer.

Device allocation can restrict a device to authorized users. Device allocation can also prevent a device from being accessed at all. A user who allocates a device has exclusive use of that device until the user deallocates the device. When a device is deallocated, device-clean scripts erase any leftover data. You can write a device-clean script to purge information from devices that do not have a script. For an example, see "Writing New Device-Clean Scripts" on page 95.

Attempts to allocate a device, deallocate a device, and list allocatable devices can be audited. The audit events are part of the `ot` audit class.

For more information on device allocation, see the following:

- "Managing Device Allocation (Task Map)" on page 79
- "Device Allocation" on page 90
- "Device Allocation Commands" on page 91

Controlling Access to Machine Resources

As system administrator, you can control and monitor system activity. You can set limits on who can use what resources. You can log resource use, and you can monitor who is using the resources. You can also set up your machines to minimize improper use of resources.

Limiting and Monitoring Superuser

Your system requires a `root` password for superuser access. In the default configuration, a user cannot remotely log in to a system as `root`. When logging in remotely, a user must log in with the user's user name and then use the `su` command to become `root`. You can monitor who has been using the `su` command, especially those users who are trying to gain superuser access. For procedures that monitor superuser and limit access to superuser, see "Monitoring and Restricting Superuser (Task Map)" on page 69.

Configuring Role-Based Access Control to Replace Superuser

Role-based access control, or RBAC, is designed to limit the capabilities of superuser. Superuser, the `root` user, has access to every resource in the system. With RBAC, you can replace `root` with a set of roles with discrete powers. For example, you can set up one role to handle user account creation, and another role to handle system file modification. When you have established a role to handle a function or set of functions, you can remove those functions from `root`'s capabilities.

Each role requires that a known user log in with their user name and password. After logging in, the user then assumes the role with a specific role password. As a consequence, someone who learns the `root` password has limited ability to damage your system. For more on RBAC, see "Role-Based Access Control (Overview)" on page 175.

Preventing Unintentional Misuse of Machine Resources

You can prevent you and your users from making unintentional errors in the following ways:

- You can keep from running a Trojan horse by correctly setting the `PATH` variable.
- You can assign a restricted shell to users. A restricted shell prevents user error by steering users to those parts of the system that the users need for their jobs. In fact, through careful setup, you can ensure that users access only those parts of the system that help the users work efficiently.
- You can set restrictive permissions on files that users do not need to access.

Setting the `PATH` Variable

You should take care to correctly set the `PATH` variable. Otherwise, you can accidentally run a program that was introduced by someone else. The intruding program can corrupt your data or harm your system. This kind of program, which creates a security hazard, is referred to as a *Trojan horse*. For example, a substitute `su` program could be placed in a public directory where you, as system administrator, might run the substitute program. Such a script would look just like the regular `su` command. Because the script removes itself after execution, you would have little evidence to show that you have actually run a Trojan horse.

The `PATH` variable is automatically set at login time. The path is set through the startup files: `.login`, `.profile`, and `.cshrc`. When you set up the user search path so that the current directory (`.`) comes last, you are protected from running this type of Trojan horse. The `PATH` variable for superuser should not include the current directory at all.

The Automated Security Enhancement Tool (ASET) examines the startup files to ensure that the `PATH` variable is set up correctly. ASET also ensures that the `PATH` variable does not contain a dot (`.`) entry.

Assigning a Restricted Shell to Users

The standard shell allows a user to open files, execute commands, and so on. The restricted shell limits the ability of a user to change directories and to execute commands. The restricted shell is invoked with the `/usr/lib/rsh` command. Note that the restricted shell is not the remote shell, which is `/usr/sbin/rsh`.

The restricted shell differs from the standard shell in the following ways:

- The user is limited to the user's home directory, so the user cannot use the `cd` command to change directories. Therefore, the user cannot browse system files.
- The user cannot change the `PATH` variable, so the user can use only commands in the path that is set by the system administrator. The user also cannot execute commands or scripts by using a complete path name.

- The user cannot redirect output with > or >>.

The restricted shell enables you to limit a user's ability to stray into system files. The shell creates a limited environment for a user who needs to perform specific tasks. The restricted shell is not completely secure, however, and is only intended to keep unskilled users from inadvertently doing damage.

For information about the restricted shell, use the `man -s1m rsh` command to see the `rsh`(1M) man page.

Restricting Access to Data in Files

Because the Solaris OS is a multiuser environment, file system security is the most basic security risk on a system. You can use traditional UNIX file protections to protect your files. You can also use the more secure access control lists (ACLs).

You might want to allow some users to read some files, and give other users permission to change or delete some files. You might have some data that you do not want anyone else to see. Chapter 6, "Controlling Access to Files (Tasks)," discusses how to set file permissions.

Restricting `setuid` Executable Files

Executable files can be security risks. Many executable programs have to be run as `root`, that is, as superuser, to work properly. These `setuid` programs run with the user ID set to `0`. Anyone who is running these programs runs the programs with the `root` ID. A program that runs with the `root` ID creates a potential security problem if the program was not written with security in mind.

Except for the executables that Sun ships with the `setuid` bit set to `root`, you should disallow the use of `setuid` programs. If you cannot disallow the use of `setuid` programs, then you should at least restrict their use. Secure administration requires few `setuid` programs.

For more information, see "Preventing Executable Files From Compromising Security" on page 130. For procedures, see "Protecting Against Programs With Security Risk (Task Map)" on page 143.

Using the Automated Security Enhancement Tool

The ASET security package provides automated administration tools that enable you to control and monitor your system's security. ASET provides three security levels: low, medium, and high. You specify an ASET security level. At each higher level, ASET's file-control functions increase to reduce file access and tighten your system's security. For more information, see Chapter 7, "Using the Automated Security Enhancement Tool (Tasks)."

Using the Solaris Security Toolkit

While ASET can be used to make a small number of security changes to a system, the Solaris Security Toolkit provides a flexible and extensible mechanism to minimize, harden, and secure a Solaris system. The Solaris Security Toolkit, informally known as the JASS toolkit, is a tool that enables the user to perform security modifications to a system. The tool can provide a report on the security status of a system. The tool also has the ability to undo previous runs of the tool. The JASS toolkit can be downloaded from the Sun web site, `http://wwws.sun.com/security/jass`. The web site contains pointers to online documentation.

The toolkit is described in detail in *Securing Systems with the Solaris Security Toolkit*, by Alex Noordergraaf and Glenn Brunette, ISBN 0-13-141071-7, June 2003. The book is part of the Sun BluePrints Series, which is published by Sun Microsystems Press.

Using Solaris Resource Management Features

Solaris software provides sophisticated resource management features. Using these features, you can allocate, schedule, monitor, and cap resource use by applications in a server consolidation environment. The resource controls framework enables you to set constraints on system resources that are consumed by processes. Such constraints help to prevent denial-of-service attacks by a script that attempts to flood a system's resources.

With Solaris resource management features, you can designate resources for particular projects. You can also dynamically adjust the resources that are available. For more information, see Part I, “Resource Management,” in *System Administration Guide: Virtualization Using the Solaris Operating System*.

Using Solaris Zones

Solaris zones provide an application execution environment in which processes are isolated from the rest of the system within a single instance of the Solaris OS. This isolation prevents processes that are running in one zone from monitoring or affecting processes that are running in other zones. Even a process running with superuser capabilities cannot view or affect activity in other zones.

Solaris zones are ideal for environments that place several applications on a single server. For more information, see Part II, “Zones,” in *System Administration Guide: Virtualization Using the Solaris Operating System*.

Monitoring Use of Machine Resources

As a system administrator, you need to monitor system activity. You need to be aware of all aspects of your machines, including the following:

- What is the normal load?
- Who has access to the system?
- When do individuals access the system?
- What programs normally run on the system?

With this kind of knowledge, you can use the available tools to audit system use and monitor the activities of individual users. Monitoring is very useful when a breach in security is suspected. For more information on the auditing service, see Chapter 28, "Solaris Auditing (Overview)."

Monitoring File Integrity

As a system administrator, you need assurance that the files that were installed on the systems that you administer have not changed in unexpected ways. In large installations, a comparison and reporting tool about the software stack on each of your systems enables you to track your systems. The Basic Audit Reporting Tool (BART) enables you to comprehensively validate systems by performing file-level checks of one or more systems over time. Changes in a BART *manifest* across systems, or for one system over time, can validate the integrity of your systems. BART provides manifest creation, manifest comparison, and rules for scripting reports. For more information, see Chapter 5, "Using the Basic Audit Reporting Tool (Tasks)."

Controlling Access to Files

The Solaris OS is a multiuser environment. In a multiuser environment, all the users who are logged in to a system can read files that belong to other users. With the appropriate file permissions, users can also use files that belong to other users. For more discussion, see Chapter 6, "Controlling Access to Files (Tasks)." For step-by-step instructions on setting appropriate permissions on files, see "Protecting Files (Task Map)" on page 131.

Protecting Files With Encryption

You can keep a file secure by making the file inaccessible to other users. For example, a file with permissions of `600` cannot be read except by its owner and by superuser. A directory with permissions of `700` is similarly inaccessible. However, someone who guesses your password or who discovers the `root` password can access that file. Also, the otherwise inaccessible file is preserved on a backup tape every time that the system files are backed up to offline media.

The Solaris Cryptographic Framework provides `digest`, `mac`, and `encrypt` commands to protect files. For more information, see Chapter 13, "Solaris Cryptographic Framework (Overview)."

Using Access Control Lists

ACLs, pronounced "ackkls," can provide greater control over file permissions. You add ACLs when traditional UNIX file protections are not sufficient. Traditional UNIX file protections provide read, write, and execute permissions for the three user classes: owner, group, and other. An ACL provides finer-grained file security.

ACLs enable you to define the following file permissions:

- Owner file permissions
- File permissions for the owner's group
- File permissions for other users who are outside the owner's group
- File permissions for specific users
- File permissions for specific groups
- Default permissions for each of the previous categories

For more information about using ACLs, see "Using Access Control Lists to Protect Files" on page 127.

Sharing Files Across Machines

A network file server can control which files are available for sharing. A network file server can also control which clients have access to the files, and what type of access is permitted for those clients. In general, the file server can grant read-write access or read-only access either to all clients or to specific clients. Access control is specified when resources are made available with the `share` command.

The `/etc/dfs/dfstab` file on the file server lists the file systems that the server makes available to clients on the network. For more information about sharing file systems, see "Automatic File-System Sharing" in *System Administration Guide: Network Services*.

Restricting `root` Access to Shared Files

In general, superuser is not allowed `root` access to file systems that are shared across the network. The NFS system prevents `root` access to mounted file systems by changing the user of the requester to the user `nobody` with the user ID `60001`. The access rights of user `nobody` are the same as those access rights that are given to the public. The user `nobody` has the access rights of a user without credentials. For example, if the public has only execute permission for a file, then user `nobody` can only execute that file.

An NFS server can grant superuser capabilities on a shared file system on a per-host basis. To grant these privileges, use the `root=`*hostname* option to the `share` command. You should use this option with care. For a discussion of security options with NFS, see Chapter 6, "Accessing Network File Systems (Reference)," in *System Administration Guide: Network Services*.

Controlling Network Access

Computers are often part of a configuration of computers. This configuration is called a *network*. A network allows connected computers to exchange information. Networked computers can access data and other resources from other computers on the network. Computer networks create a powerful and sophisticated computing environment. However, networks also complicate computer security.

For example, within a network of computers, individual machines allow the sharing of information. Unauthorized access is a security risk. Because many people have access to a network, unauthorized access is more likely, especially through user error. A poor use of passwords can also allow unauthorized access.

Network Security Mechanisms

Network security is usually based on limiting or blocking operations from remote systems. The following figure describes the security restrictions that you can impose on remote operations.

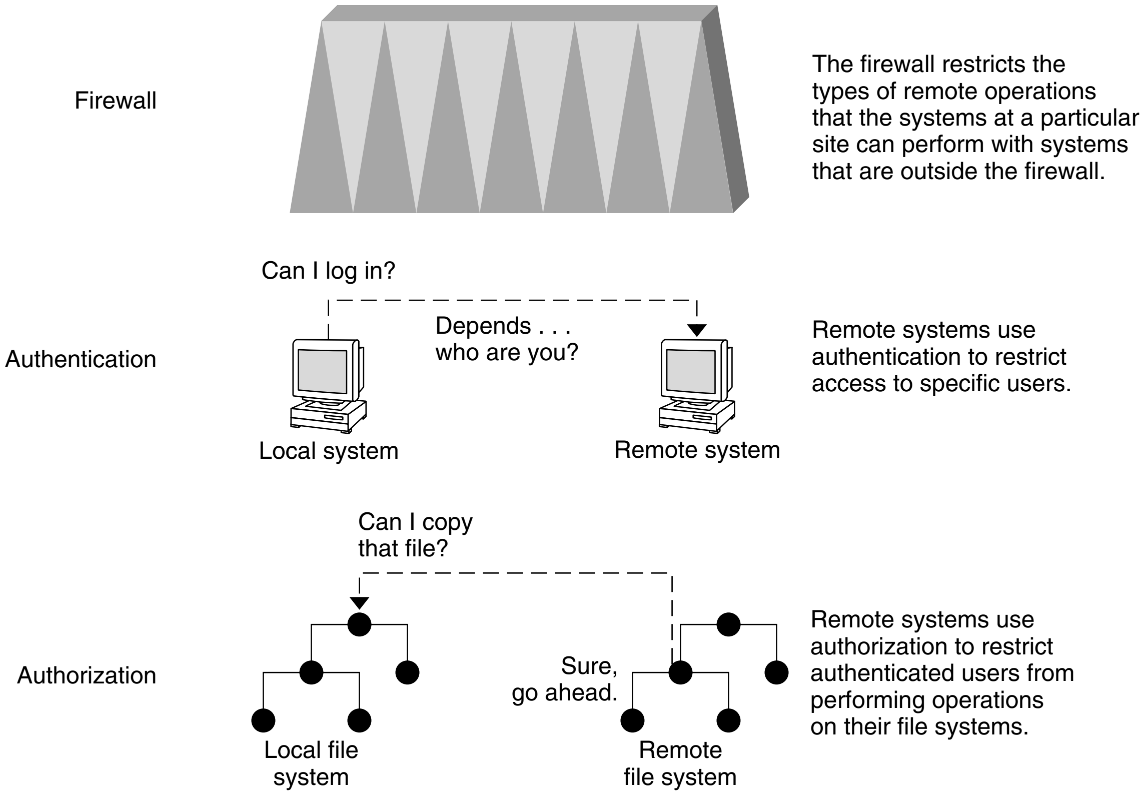

Firewall
The firewall restricts the types of remote operations that the systems at a particular site can perform with systems that are outside the firewall.
Can I log in?
Depends . . . who are you?
Authentication
Local system
Remote system
Remote systems use authentication to restrict access to specific users.
Can I copy that file?
Authorization
Sure, go ahead.
Local file system
Remote file system
Remote systems use authorization to restrict authenticated users from performing operations on their file systems.

FIGURE 2–1 Security Restrictions for Remote Operations

Authentication and Authorization for Remote Access

Authentication is a way to restrict access to specific users when these users access a remote system. Authentication can be set up at both the system level and the network level. After a user has gained access to a remote system, *authorization* is a way to restrict operations that the user can perform. The following table lists the services that provide authentication and authorization.

TABLE 2–3 Authentication and Authorization Services for Remote Access

Service	Description	For More Information
IPsec	IPsec provides host-based and certificate-based authentication and network traffic encryption.	Chapter 19, "IP Security Architecture (Overview)," in *System Administration Guide: IP Services*

TABLE 2–3 Authentication and Authorization Services for Remote Access *(Continued)*

Service	Description	For More Information
Kerberos	Kerberos uses encryption to authenticate and authorize a user who is logging in to the system.	For an example, see "How the Kerberos Service Works" on page 364.
LDAP and NIS+	The LDAP directory service and the NIS+ name service can provide both authentication and authorization at the network level.	*System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)* and *System Administration Guide: Naming and Directory Services (NIS+)*
Remote login commands	The remote login commands enable users to log in to a remote system over the network and use its resources. Some of the remote login commands are `rlogin`, `rcp`, and `ftp`. If you are a "trusted host," authentication is automatic. Otherwise, you are asked to authenticate yourself.	Chapter 29, "Accessing Remote Systems (Tasks)," in *System Administration Guide: Network Services*
SASL	The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. Plugins enable you to choose an appropriate authentication protocol.	"SASL (Overview)" on page 323
Secure RPC	Secure RPC improves the security of network environments by authenticating users who make requests on remote machines. You can use either the UNIX, DES, or Kerberos authentication system for Secure RPC.	"Overview of Secure RPC" on page 299
	Secure RPC can also be used to provide additional security in an NFS environment. An NFS environment with secure RPC is called Secure NFS. Secure NFS uses Diffie-Hellman authentication for public keys.	"NFS Services and Secure RPC" on page 299
Solaris Secure Shell	Solaris Secure Shell encrypts network traffic over an unsecured network. Solaris Secure Shell provides authentication by the use of passwords, public keys, or both. Solaris Secure Shell uses RSA and DSA authentication for public keys.	"Solaris Secure Shell (Overview)" on page 327

A possible substitute for Secure RPC is the Solaris *privileged port* mechanism. A privileged port is assigned a port number less than `1024`. After a client system has authenticated the client's credential, the client builds a connection to the server by using the privileged port. The server then verifies the client credential by examining the connection's port number.

Clients that are not running Solaris software might be unable to communicate by using the privileged port. If the clients cannot communicate over the port, you see an error message that is similar to the following:

```
“Weak Authentication
NFS request from unprivileged port”
```

Firewall Systems

You can set up a firewall system to protect the resources in your network from outside access. A *firewall system* is a secure host that acts as a barrier between your internal network and outside networks. The internal network treats every other network as untrusted. You should consider this setup as mandatory between your internal network and any external networks, such as the Internet, with which you communicate.

A firewall acts as a gateway and as a barrier. A firewall acts as a gateway that passes data between the networks. A firewall acts as a barrier that blocks the free passage of data to and from the network. The firewall requires a user on the internal network to log in to the firewall system to access hosts on remote networks. Similarly, a user on an outside network must first log in to the firewall system before being granted access to a host on the internal network.

A firewall can also be useful between some internal networks. For example, you can set up a firewall or a secure gateway computer to restrict the transfer of packets. The gateway can forbid packet exchange between two networks, unless the gateway computer is the source address or the destination address of the packet. A firewall should also be set up to forward packets for particular protocols only. For example, you can allow packets for transferring mail, but not allow packets for the `telnet` or the `rlogin` command. ASET, when run at high security, disables the forwarding of Internet Protocol (IP) packets.

In addition, all electronic mail that is sent from the internal network is first sent to the firewall system. The firewall then transfers the mail to a host on an external network. The firewall system also receives all incoming electronic mail, and distributes the mail to the hosts on the internal network.

Caution – A firewall prevents unauthorized users from accessing the hosts on your network. You should maintain strict and rigidly enforced security on the firewall, but security on other hosts on the network can be more relaxed. However, an intruder who can break into your firewall system can then gain access to all the other hosts on the internal network.

A firewall system should not have any trusted hosts. A *trusted host* is a host from which a user can log in without being required to supply a password. A firewall system should not share any of its file systems, or mount any file systems from other servers.

The following technologies can be used to harden a system into a firewall:

- ASET enforces high security on a firewall system, as described in Chapter 7, “Using the Automated Security Enhancement Tool (Tasks).”
- The Solaris Security Toolkit, informally known as the JASS toolkit, can harden a Solaris system into a firewall. The toolkit can be downloaded from the Sun web site, `http://wwws.sun.com/security/jass`.

- IPsec and Solaris IP filter can provide firewall protection. For more information on protecting network traffic, see Part III, “IP Security,” in *System Administration Guide: IP Services*.

Encryption and Firewall Systems

Most local area networks transmit data between computers in blocks that are called *packets*. Through a procedure that is called *packet smashing*, unauthorized users from outside the network can corrupt or destroy data.

Packet smashing involves capturing the packets before the packets reach their destination. The intruder then injects arbitrary data into the contents, and sends the packets back on their original course. On a local area network, packet smashing is impossible because packets reach all systems, including the server, at the same time. Packet smashing is possible on a gateway, however, so make sure that all gateways on the network are protected.

The most dangerous attacks affect the integrity of the data. Such attacks involve changing the contents of the packets or impersonating a user. Attacks that involve eavesdropping do not compromise data integrity. An eavesdropper records conversations for later replay. An eavesdropper does not impersonate a user. Although eavesdropping attacks do not attack data integrity, the attacks do affect privacy. You can protect the privacy of sensitive information by encrypting data that goes over the network.

- To encrypt remote operations over an insecure network, see Chapter 19, “Using Solaris Secure Shell (Tasks).”
- To encrypt and authenticate data across a network, see Chapter 21, “Introduction to the Kerberos Service.”
- To encrypt IP datagrams, see Chapter 19, “IP Security Architecture (Overview),” in *System Administration Guide: IP Services*.

Reporting Security Problems

If you experience a suspected security breach, you can contact the Computer Emergency Response Team/Coordination Center (CERT/CC). CERT/CC is a Defense Advanced Research Projects Agency (DARPA) funded project that is located at the Software Engineering Institute at Carnegie Mellon University. This agency can assist you with any security problems you are having. This agency can also direct you to other Computer Emergency Response Teams that might be more appropriate for your particular needs. You can call CERT/CC at its 24-hour hotline: (412) 268-7090. Or, contact the team by email at `cert@cert.sei.cmu.edu`.

◆ ◆ ◆ CHAPTER 3

Controlling Access to Systems (Tasks)

This chapter describes the procedures for controlling who can access Solaris systems. The following is a list of the information in this chapter.

- "Controlling System Access (Task Map)" on page 57
- "Securing Logins and Passwords (Task Map)" on page 58
- "Changing the Password Algorithm (Task Map)" on page 65
- "Monitoring and Restricting Superuser (Task Map)" on page 69
- "SPARC: Controlling Access to System Hardware (Task Map)" on page 72

For overview information about system security, see Chapter 2, "Managing Machine Security (Overview)."

Controlling System Access (Task Map)

A computer is as secure as its weakest point of entry. The following task map shows the areas that you should monitor and secure.

Task	Description	For Instructions
Monitor, permit, and deny user login	Monitors unusual login activity. Prevents logins temporarily. Manages dial-up logins.	"Securing Logins and Passwords (Task Map)" on page 58
Provide strong password encryption	Specifies algorithms to encrypt user passwords. Installs additional algorithms.	"Changing the Password Algorithm (Task Map)" on page 65
Monitor and restrict superuser activities	Regularly monitors superuser activity. Prevents remote login by a `root` user.	"Monitoring and Restricting Superuser (Task Map)" on page 69
Prevent access to hardware settings	Keeps ordinary users away from the PROM.	"SPARC: Controlling Access to System Hardware (Task Map)" on page 72

Securing Logins and Passwords (Task Map)

The following task map points to procedures that monitor user logins and that disable user logins.

Task	Description	For Instructions
Display a user's login status	Lists extensive information about a user's login account, such as full name and password aging information.	"How to Display a User's Login Status" on page 58
Find users who do not have passwords	Finds only those users whose accounts do not require a password.	"How to Display Users Without Passwords" on page 59
Disable logins temporarily	Denies user logins to a machine as part of system shutdown or routine maintenance.	"How to Temporarily Disable User Logins" on page 60
Save failed login attempts	Creates a log of users who failed to provide the correct password after five attempts.	"How to Monitor Failed Login Attempts" on page 61
Save all failed login attempts	Creates a log of failed attempts to log in.	"How to Monitor All Failed Login Attempts" on page 62
Create a dial-up password	Requires an additional password for users who log in remotely through a modem or dial-up port.	"How to Create a Dial-Up Password" on page 63
Disable dial-up logins temporarily	Prevents users from dialing in remotely through a modem or port.	"How to Temporarily Disable Dial-Up Logins" on page 65

Securing Logins and Passwords

You can limit remote logins and require users to have passwords. You can also monitor failed access attempts and disable logins temporarily.

▼ How to Display a User's Login Status

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Display a user's login status by using the `logins` command.**

   ```
   # logins -x -l username
   ```

`-x`	Displays an extended set of login status information.

`-l` *username*	Displays the login status for the specified user. The variable *username* is a user's login name. Multiple login names must be specified in a comma-separated list.

The `logins` command uses the appropriate password database to obtain a user's login status. The database can be the local `/etc/passwd` file, or a password database for the name service. For more information, see the `logins`(1M) man page.

Example 3–1 Displaying a User's Login Status

In the following example, the login status for the user `rimmer` is displayed.

```
# logins -x -l rimmer
rimmer       500     staff           10   Annalee J. Rimmer
                     /export/home/rimmer
                     /bin/sh
                     PS 010103 10 7 -1
```

`rimmer`	Identifies the user's login name.
`500`	Identifies the user ID (UID).
`staff`	Identifies the user's primary group.
`10`	Identifies the group ID (GID).
`Annalee J. Rimmer`	Identifies the comment.
`/export/home/rimmer`	Identifies the user's home directory.
`/bin/sh`	Identifies the login shell.
`PS 010170 10 7 -1`	Specifies the password aging information: ■ Last date that the password was changed ■ Number of days that are required between changes ■ Number of days before a change is required ■ Warning period

▼ How to Display Users Without Passwords

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Display all users who have no passwords by using the `logins` command.**

```
# logins -p
```

The `-p` option displays a list of users with no passwords. The `logins` command uses the password database from the local system unless a name service is enabled.

Example 3–2 Displaying Users Without Passwords

In the following example, the user `pmorph` does not have a password.

```
# logins -p
pmorph          501     other           1       Polly Morph
#
```

▼ How to Temporarily Disable User Logins

Temporarily disable user logins during system shutdown or routine maintenance. Superuser logins are not affected. For more information, see the `nologin`(4) man page.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Create the `/etc/nologin` file in a text editor.**

```
# vi /etc/nologin
```

3 **Include a message about system availability.**

4 **Close and save the file.**

Example 3–3 Disabling User Logins

In this example, users are notified of system unavailability.

```
# vi /etc/nologin
(Add system message here)

# cat /etc/nologin
***No logins permitted.***

***The system will be unavailable until 12 noon.***
```

You can also bring the system to run level 0, single-user mode, to disable logins. For information on bringing the system to single-user mode, see Chapter 9, "Shutting Down a System (Tasks)," in *System Administration Guide: Basic Administration*.

▼ How to Monitor Failed Login Attempts

This procedure captures failed login attempts from terminal windows. This procedure does not capture failed logins from a CDE or GNOME login attempt.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Create the `loginlog` file in the `/var/adm` directory.**

   ```
   # touch /var/adm/loginlog
   ```

3. **Set read-and-write permissions for `root` user on the `loginlog` file.**

   ```
   # chmod 600 /var/adm/loginlog
   ```

4. **Change group membership to `sys` on the `loginlog` file.**

   ```
   # chgrp sys /var/adm/loginlog
   ```

5. **Verify that the log works.**

   For example, log in to the system five times with the wrong password. Then, display the `/var/adm/loginlog` file.

   ```
   # more /var/adm/loginlog
   jdoe:/dev/pts/2:Tue Nov  4 10:21:10 2003
   jdoe:/dev/pts/2:Tue Nov  4 10:21:21 2003
   jdoe:/dev/pts/2:Tue Nov  4 10:21:30 2003
   jdoe:/dev/pts/2:Tue Nov  4 10:21:40 2003
   jdoe:/dev/pts/2:Tue Nov  4 10:21:49 2003
   #
   ```

   The `loginlog` file contains one entry for each failed attempt. Each entry contains the user's login name, tty device, and time of the failed attempt. If a person makes fewer than five unsuccessful attempts, no failed attempts are logged.

   A growing `loginlog` file can indicate an attempt to break into the computer system. Therefore, check and clear the contents of this file regularly. For more information, see the `loginlog`(4) man page.

▼ How to Monitor All Failed Login Attempts

This procedure captures in a `syslog` file all failed login attempts.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Set up the `/etc/default/login` file with the desired values for `SYSLOG` and `SYSLOG_FAILED_LOGINS`**

   Edit the `/etc/default/login` file to change the entry. Make sure that **`SYSLOG=YES`** is uncommented.

   ```
   # grep SYSLOG /etc/default/login
   # SYSLOG determines whether the syslog(3) LOG_AUTH facility
   # should be used
   SYSLOG=YES
   ...
   SYSLOG_FAILED_LOGINS=0
   #
   ```

3. **Create a file with the correct permissions to hold the logging information.**

   a. **Create the `authlog` file in the `/var/adm` directory.**

   ```
   # touch /var/adm/authlog
   ```

   b. **Set read-and-write permissions for `root` user on the `authlog` file.**

   ```
   # chmod 600 /var/adm/authlog
   ```

   c. **Change group membership to `sys` on the `authlog` file.**

   ```
   # chgrp sys /var/adm/authlog
   ```

4. **Edit the `syslog.conf` file to log failed password attempts.**

   The failures should be sent to the `authlog` file.

   a. **Type the following entry into the `syslog.conf` file.**

   Fields on the same line in `syslog.conf` are separated by tabs.

   ```
   auth.notice      <Press Tab>   /var/adm/authlog
   ```

   b. **Refresh the configuration information for the `syslog` daemon.**

   ```
   # svcadm refresh system/system-log
   ```

5 **Verify that the log works.**

For example, as an ordinary user, log in to the system with the wrong password. Then, in the Primary Administrator role or as superuser, display the `/var/adm/authlog` file.

```
# more /var/adm/authlog
Nov  4 14:46:11 example1 login: [ID 143248 auth.notice]
 Login failure on /dev/pts/8 from example2, stacey
#
```

6 **Monitor the `/var/adm/authlog` file on a regular basis.**

Example 3–4 Logging Access Attempts After Three Login Failures

Follow the preceding procedure, except set the value of `SYSLOG_FAILED_LOGINS` to 3 in the `/etc/default/login` file.

Example 3–5 Closing Connection After Three Login Failures

Uncomment the `RETRIES` entry in the `/etc/default/login` file, then set the value of `RETRIES` to 3. Your edits take effect immediately. After three login retries in one session, the system closes the connection.

▼ How to Create a Dial-Up Password

Caution – When you first establish a dial-up password, be sure to remain logged in to at least one port. Test the password on a different port. If you log off to test the new password, you might not be able to log back in. If you are still logged in to another port, you can go back and fix your mistake.

---

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Create an `/etc/dialups` file that contains a list of serial devices.**

Include all the ports that are being protected with dial-up passwords. The `/etc/dialups` file should appear similar to the following:

```
/dev/term/a
/dev/term/b
/dev/term/c
```

3 **Create an `/etc/d_passwd` file that contains the login programs that you are requiring to have a dial-up password.**

Include shell programs that a user could be running at login, for example, `uucico`, `sh`, `ksh`, and `csh`. The `/etc/d_passwd` file should appear similar to the following:

```
/usr/lib/uucp/uucico:encrypted-password:
/usr/bin/csh:encrypted-password:
/usr/bin/ksh:encrypted-password:
/usr/bin/sh:encrypted-password:
```

Later in the procedure, you are going to add the encrypted password for each login program.

4 **Set ownership to `root` on the two files.**

```
# chown root /etc/dialups /etc/d_passwd
```

5 **Set group ownership to `root` on the two files.**

```
# chgrp root /etc/dialups /etc/d_passwd
```

6 **Set read-and-write permissions for `root` on the two files.**

```
# chmod 600 /etc/dialups /etc/d_passwd
```

7 **Create the encrypted passwords.**

a. **Create a temporary user.**

```
# useradd username
```

b. **Create a password for the temporary user.**

```
# passwd username
New Password:       <Type password>
Re-enter new Password:       <Retype password>
passwd: password successfully changed for username
```

c. **Capture the encrypted password.**

```
# grep username /etc/shadow > username.temp
```

d. **Edit the *username*.temp file.**

Delete all fields except the encrypted password. The second field holds the encrypted password.

For example, in the following line, the encrypted password is `U9gp9SyA/JlSk`.

```
temp:U9gp9SyA/JlSk:7967:::::7988:
```

e. **Delete the temporary user.**

```
# userdel username
```

8 **Copy the encrypted password from** *username*`.temp` **file into the** `/etc/d_passwd` **file.**

You can create a different password for each login shell. Alternatively, use the same password for each login shell.

9 **Inform your dial-up users of the password.**

You should ensure that your means of informing the users cannot be tampered with.

▼ How to Temporarily Disable Dial-Up Logins

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Put the following single-line entry into the** `/etc/d_passwd` **file:**

```
/usr/bin/sh:*:
```

Changing the Password Algorithm (Task Map)

The following task map points to procedures to administer password algorithms.

Task	For Instructions
Provide strong password encryption	“How to Specify an Algorithm for Password Encryption” on page 66
Provide strong password encryption with a name service	“How to Specify a New Password Algorithm for an NIS Domain” on page 67
	“How to Specify a New Password Algorithm for an NIS+ Domain” on page 67
	“How to Specify a New Password Algorithm for an LDAP Domain” on page 68
Add new password encryption module	“How to Install a Password Encryption Module From a Third Party” on page 68

Changing the Default Algorithm for Password Encryption

By default, user passwords are encrypted with the `crypt_unix` algorithm. You can use a stronger encryption algorithm, such as MD5 or Blowfish, by changing the default password encryption algorithm.

▼ How to Specify an Algorithm for Password Encryption

In this procedure, the BSD-Linux version of the MD5 algorithm is the default encryption algorithm that is used when users change their passwords. This algorithm is suitable for a mixed network of machines that run the Solaris, BSD, and Linux versions of UNIX. For a list of password encryption algorithms and algorithm identifiers, see Table 2–1.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Specify the identifier for your chosen encryption algorithm.**

   Type the identifier as the value for the `CRYPT_DEFAULT` variable in the `/etc/security/policy.conf` file.

   You might want to comment the file to explain your choice.

```
# cat  /etc/security/policy.conf
...
CRYPT_ALGORITHMS_ALLOW=1,2a,md5
#
# Use the version of MD5 that works with Linux and BSD systems.
# Passwords previously encrypted with __unix__ will be encrypted with MD5
# when users change their passwords.
#
#
CRYPT_DEFAULT=__unix__
CRYPT_DEFAULT=1
```

In this example, the algorithms configuration ensures that the weakest algorithm, `crypt_unix`, is never used to encrypt a password. Users whose passwords were encrypted with the `crypt_unix` module get a `crypt_bsdmd5`-encrypted password when they change their passwords.

For more information on configuring the algorithm choices, see the `policy.conf`(4) man page.

Example 3–6 Using the Blowfish Algorithm for Password Encryption

In this example, the identifier for the Blowfish algorithm, `2a`, is specified as the value for the `CRYPT_DEFAULT` variable in the `policy.conf` file:

```
CRYPT_ALGORITHMS_ALLOW=1,2a,md5
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=2a
```

This configuration is compatible with BSD systems that use the Blowfish algorithm.

▼ How to Specify a New Password Algorithm for an NIS Domain

When users in an NIS domain change their passwords, the NIS client consults its local algorithms configuration in the `/etc/security/policy.conf` file. The NIS client machine encrypts the password.

1. **Specify the password encryption algorithm in the `/etc/security/policy.conf` file on the NIS client.**

2. **Copy the modified `/etc/security/policy.conf` file to every client machine in the NIS domain.**

3. **To minimize confusion, copy the modified `/etc/security/policy.conf` file to the NIS root server and to the slave servers.**

▼ How to Specify a New Password Algorithm for an NIS+ Domain

When users in an NIS+ domain change their passwords, the NIS+ name service consults the algorithms configuration in the `/etc/security/policy.conf` file on the NIS+ master. The NIS+ master, which is running the `rpc.nispasswd` daemon, creates the encrypted password.

1. **Specify the password encryption algorithm in the `/etc/security/policy.conf` file on the NIS+ master.**

2. **To minimize confusion, copy the NIS+ master's `/etc/security/policy.conf` file to every host in the NIS+ domain.**

▼ How to Specify a New Password Algorithm for an LDAP Domain

When the LDAP client is properly configured, the LDAP client can use the new password algorithms. The LDAP client behaves just as an NIS client behaves.

1. **Specify a password encryption algorithm in the `/etc/security/policy.conf` file on the LDAP client.**

2. **Copy the modified `policy.conf` file to every client machine in the LDAP domain.**

3. **Ensure that the client's `/etc/pam.conf` file does not use a `pam_ldap` module.**

   Ensure that a comment sign (#) precedes entries that include `pam_ldap.so.1`. Also, do not use the new `server_policy` option with the `pam_authtok_store.so.1` module.

   The PAM entries in the client's `pam.conf` file enable the password to be encrypted according to the local algorithms configuration. The PAM entries also enable the password to be authenticated.

   When users in the LDAP domain change their passwords, the LDAP client consults its local algorithms configuration in the `/etc/security/policy.conf` file. The LDAP client machine encrypts the password. Then, the client sends the encrypted password, with a `{crypt}` tag, to the server. The tag tells the server that the password is already encrypted. The password is then stored, as is, on the server. For authentication, the client retrieves the stored password from the server. The client then compares the stored password with the encrypted version that the client has just generated from the user's typed password.

Note – To take advantage of password policy controls on the LDAP server, use the `server_policy` option with the `pam_authtok_store` entries in the `pam.conf` file. Passwords are then encrypted on the server by using the Sun Java™ System Directory Server's cryptographic mechanism. For the procedure, see Chapter 11, "Setting Up Sun Java System Directory Server With LDAP Clients (Tasks)," in *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP)*.

▼ How to Install a Password Encryption Module From a Third Party

A third-party password encryption algorithm is typically delivered as a module in a software package. When you run the `pkgadd` command, scripts from the vendor should modify the `/etc/security/crypt.conf` file. You then modify the `/etc/security/policy.conf` file to include the new module and its identifier.

1. **Add the software by using the `pkgadd` command.**

   For detailed instructions on how to add software, see "Adding or Removing a Software Package (pkgadd)" in *System Administration Guide: Basic Administration*.

2. **Confirm that the new module and module identifier have been added.**

   Read the list of encryption algorithms in the `/etc/security/crypt.conf` file.

   For example, the following lines show that a module that implements the `crypt_rot13` algorithm has been installed.

```
# crypt.conf
#
md5 /usr/lib/security/$ISA/crypt_md5.so
rot13 /usr/lib/security/$ISA/crypt_rot13.so

# For *BSD - Linux compatibility
# 1 is MD5,  2a is Blowfish
1 /usr/lib/security/$ISA/crypt_bsdmd5.so
2a /usr/lib/security/$ISA/crypt_bsdbf.so
```

3. **Add the identifier of the newly installed algorithm to the `/etc/security/policy.conf` file.**

   The following lines show excerpts from the `policy.conf` file that would need to be modified to add the `rot13` identifier.

```
# Copyright 1999-2002 Sun Microsystems, Inc.  All rights reserved.
# ...
#ident  "@(#)policy.conf        1.6     02/06/07 SMI"
# ...
# crypt(3c) Algorithms Configuration
CRYPT_ALGORITHMS_ALLOW=1,2a,md5,rot13
#CRYPT_ALGORITHMS_DEPRECATE=__unix__
CRYPT_DEFAULT=md5
```

   In this example, the `rot13` algorithm is used if the current password was encrypted with the `crypt_rot13` algorithm. New user passwords are encrypted with the `crypt_sunmd5` algorithm. This algorithms configuration works on Solaris-only networks.

Monitoring and Restricting Superuser (Task Map)

The following task map describes how to monitor and restrict the `root` user login.

Task	Description	For Instructions
Monitor who is using the `su` command	Scans the `sulog` file on a regular basis.	"How to Monitor Who Is Using the `su` Command" on page 70

Task	Description	For Instructions
Display superuser activity on the console	Monitors superuser access attempts as the attempts occur.	"How to Restrict and Monitor Superuser Logins" on page 71

Monitoring and Restricting Superuser

An alternative to using the superuser account is to set up role-based access control. Role-based access control is called RBAC. For overview information on RBAC, see "Role-Based Access Control (Overview)" on page 175. To set up RBAC, see Chapter 9, "Using Role-Based Access Control (Tasks)."

▼ How to Monitor Who Is Using the `su` Command

The `sulog` file lists every use of the `su` command, not only the `su` attempts that are used to switch from user to superuser.

● **Monitor the contents of the `/var/adm/sulog` file on a regular basis.**

```
# more /var/adm/sulog
SU 12/20 16:26 + pts/0 stacey-root
SU 12/21 10:59 + pts/0 stacey-root
SU 01/12 11:11 + pts/0 root-rimmer
SU 01/12 14:56 + pts/0 pmorph-root
SU 01/12 14:57 + pts/0 pmorph-root
```

The entries display the following information:

- The date and time that the command was entered.
- If the attempt was successful. A plus sign (+) indicates a successful attempt. A minus sign (`-`) indicates an unsuccessful attempt.
- The port from which the command was issued.
- The name of the user and the name of the switched identity.

The `su` logging in this file is enabled by default through the following entry in the `/etc/default/su` file:

```
SULOG=/var/adm/sulog
```

Troubleshooting Entries that include `???` indicate that the controlling terminal for the `su` command cannot be identified. Typically, system invocations of the `su` command before the desktop appears include `???`, as in `SU 10/10 08:08 + ??? root-root`. After the user starts a desktop session, the `ttynam` command returns the value of the controlling terminal to the `sulog`: `SU 10/10 10:10 + pts/3 jdoe-root`.

Entries similar to the following can indicate that the `su` command was not invoked on the command line: `SU 10/10 10:20 + ??? root-oracle`. The user might have switched to the oracle role by using a GUI.

▼ How to Restrict and Monitor Superuser Logins

This method immediately detects superuser attempts to access the local system.

1. **View the `CONSOLE` entry in the `/etc/default/login` file.**

   ```
   CONSOLE=/dev/console
   ```

   By default, the console device is set to `/dev/console`. With this setting, `root` can log in to the console. `root` cannot log in remotely.

2. **Verify that `root` cannot log in remotely.**

   From a remote system, try to log in as superuser.

   ```
   mach2 % rlogin -l root mach1
   Password:       <Type root password of mach1>
   Not on system console
   Connection closed.
   ```

3. **Monitor attempts to become superuser.**

   By default, attempts to become superuser are printed to the console by the `SYSLOG` utility.

   a. **Open a terminal console on your desktop.**

   b. **In another window, use the `su` command to become superuser.**

   ```
   % su -
   Password:       <Type root password>
   #
   ```

   A message is printed on the terminal console.

   ```
   Sep 7 13:22:57 mach1 su: 'su root' succeeded for jdoe on /dev/pts/6
   ```

Example 3–7 Logging Superuser Access Attempts

In this example, superuser attempts are not being logged by `SYSLOG`. Therefore, the administrator is logging those attempts by removing the comment from the `#CONSOLE=/dev/console` entry in the `/etc/default/su` file.

```
# CONSOLE determines whether attempts to su to root should be logged
# to the named device
#
CONSOLE=/dev/console
```

When a user attempts to become superuser, the attempt is printed on the terminal console.

```
SU 09/07 16:38 + pts/8 jdoe-root
```

Troubleshooting To become superuser from a remote system when the `/etc/default/login` file contains the default `CONSOLE` entry, users must first log in with their user name. After logging in with their user name, users then can use the `su` command to become superuser.

If the console displays an entry similar to `Mar 16 16:20:36 mach1 login: ROOT LOGIN /dev/pts/14 FROM mach2.Example.COM`, then the system is permitting remote `root` logins. To prevent remote superuser access, change the `#CONSOLE=/dev/console` entry to `CONSOLE=/dev/console` in the `/etc/default/login` file.

SPARC: Controlling Access to System Hardware (Task Map)

The following task map describes how to protect the PROM from unwanted access.

Task	Description	For Instructions
Prevent users from changing system hardware settings	Requires a password to modify PROM settings.	"How to Require a Password for Hardware Access" on page 72
Disable the abort sequence	Prevents users from accessing the PROM.	"How to Disable a System's Abort Sequence" on page 73

Controlling Access to System Hardware

You can protect the physical machine by requiring a password to gain access to the hardware settings. You can also protect the machine by preventing a user from using the abort sequence to leave the windowing system.

▼ How to Require a Password for Hardware Access

On an x86 system, the equivalent to protecting the PROM is to protect the BIOS. Refer to your machine's manuals for how to protect the BIOS.

1. **Become superuser or assume a role that includes the Device Security profile, the Maintenance and Repair profile, or the System Administrator profile.**

   The System Administrator profile includes the Maintenance and Repair profile. To create a role that includes the System Administrator profile and to assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2 **In a terminal window, type the PROM security mode.**

```
# eeprom security-mode=command

Changing PROM password:
New password:      <Type password>
Retype new password:      <Retype password>
```

Choose the value `command` or `full`. For more details, see the `eeprom`(1M) man page.

If, when you type the preceding command, you are not prompted for a PROM password, the system already has a PROM password.

3 **(Optional) To change the PROM password, type the following command:**

```
# eeprom security-password=       Press Return
Changing PROM password:
New password:      <Type password>
Retype new password:      <Retype password>
```

The new PROM security mode and password are in effect immediately. However, they are most likely to be noticed at the next boot.

Caution – Do not forget the PROM password. The hardware is unusable without this password.

How to Disable a System's Abort Sequence

Some server systems have a key switch. When the key switch is set in the secure position, the switch overrides the software keyboard abort settings. So, any changes that you make with the following procedure might not be implemented.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Change the value of `KEYBOARD_ABORT` to `disable`.**

Comment out the `enable` line in the `/etc/default/kbd` file. Then, add a `disable` line:

```
# cat /etc/default/kbd
...
# KEYBOARD_ABORT affects the default behavior of the keyboard abort
# sequence, see kbd(1) for details.  The default value is "enable".
# The optional value is "disable".  Any other value is ignored.
...
```

```
#KEYBOARD_ABORT=enable
KEYBOARD_ABORT=disable
```

3. **Update the keyboard defaults.**

```
# kbd -i
```

◆ ◆ ◆ CHAPTER 4

Controlling Access to Devices (Tasks)

This chapter provides step-by-step instructions for protecting devices, in addition to a reference section. The following is a list of the information in this chapter.

- "Configuring Devices (Task Map)" on page 75
- "Configuring Device Policy (Task Map)" on page 76
- "Managing Device Allocation (Task Map)" on page 79
- "Allocating Devices (Task Map)" on page 85
- "Device Protection (Reference)" on page 89

For overview information about device protection, see "Controlling Access to Devices" on page 44.

Configuring Devices (Task Map)

The following task map points to tasks for managing access to devices.

Task	For Instructions
Manage device policy	"Configuring Device Policy (Task Map)" on page 76
Manage device allocation	"Managing Device Allocation (Task Map)" on page 79
Use device allocation	"Allocating Devices (Task Map)" on page 85

Configuring Device Policy (Task Map)

The following task map points to device configuration procedures that are related to device policy.

Task	Description	For Instructions
View the device policy for the devices on your system	Lists the devices and their device policy.	"How to View Device Policy" on page 76
Require privilege for device use	Uses privileges to protect a device.	"How to Change the Device Policy on an Existing Device" on page 77
Remove privilege requirements from a device	Removes or lessens the privileges that are required to access a device.	Example 4–3
Audit changes in device policy	Records changes in device policy in the audit trail	"How to Audit Changes in Device Policy" on page 78
Access `/dev/arp`	Gets Solaris IP MIB-II information.	"How to Retrieve IP MIB-II Information From a `/dev/*` Device" on page 79

Configuring Device Policy

Device policy restricts or prevents access to devices that are integral to the system. The policy is enforced in the kernel.

▼ How to View Device Policy

- **Display the device policy for all devices on your system.**

```
% getdevpolicy | more
DEFAULT
        read_priv_set=none
        write_priv_set=none
ip:*
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
...
```

Example 4–1 Viewing the Device Policy for a Specific Device

In this example, the device policy for three devices is displayed.

```
% getdevpolicy /dev/allkmem /dev/ipsecesp /dev/hme
/dev/allkmem
```

```
        read_priv_set=all
        write_priv_set=all
/dev/ipsecesp
        read_priv_set=sys_net_config
        write_priv_set=sys_net_config
/dev/hme
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
```

▼ How to Change the Device Policy on an Existing Device

1. **Assume a role that includes the Device Security rights profile, or become superuser.**

   The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2. **Add policy to a device.**

   `# update_drv -a -p` *policy device-driver*

`-a`	Specifies a *policy* for *device-driver*.
`-p` *policy*	Is the device policy for *device-driver*. Device policy specifies two sets of privileges. One set is required to read the device. The other set is required to write to the device.
device-driver	Is the device driver.

   For more information, see the `update_drv`(1M) man page.

Example 4–2 Adding Policy to an Existing Device

In the following example, device policy is added to the `ipnat` device.

```
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=none
        write_priv_set=none
# update_drv -a \
-p 'read_priv_set=net_rawaccess write_priv_set=net_rawaccess' ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
```

Example 4–3 Removing Policy From a Device

In the following example, the read set of privileges is removed from the device policy for the `ipnat` device.

```
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
# update_drv -a -p write_priv_set=net_rawaccess ipnat
# getdevpolicy /dev/ipnat
/dev/ipnat
        read_priv_set=none
        write_priv_set=net_rawaccess
```

▼ How to Audit Changes in Device Policy

By default, the `as` audit class includes the `AUE_MODDEVPLCY` audit event.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Preselect the audit class that includes `AUE_MODDEVPLCY` audit event.**

   Add the `as` class to the `flags` line of the `audit_control` file. The file would appear similar to the following:

   ```
   # audit_control file
   dir:/var/audit
   flags:lo,as
   minfree:20
   naflags:lo
   ```

   For detailed instructions, see "How to Modify the `audit_control` File" on page 571.

▼ How to Retrieve IP MIB-II Information From a `/dev/*` Device

Applications that retrieve Solaris IP MIB-II information should open `/dev/arp`, not `/dev/ip`.

1. **Determine the device policy on `/dev/ip` and `/dev/arp`.**

```
% getdevpolicy /dev/ip /dev/arp
/dev/ip
        read_priv_set=net_rawaccess
        write_priv_set=net_rawaccess
/dev/arp
        read_priv_set=none
        write_priv_set=none
```

Note that the `net_rawaccess` privilege is required for reading and writing to `/dev/ip`. No privileges are required for `/dev/arp`.

2. **Open `/dev/arp` and push the `tcp` and `udp` modules.**

No privileges are required. This method is equivalent to opening `/dev/ip` and pushing the `arp`, `tcp` and `udp` modules. Because opening `/dev/ip` now requires a privilege, the `/dev/arp` method is preferred.

Managing Device Allocation (Task Map)

The following task map points to procedures that enable and configure device allocation. Device allocation is not enabled by default. After device allocation is enabled, see "Allocating Devices (Task Map)" on page 85.

Task	Description	For Instructions
Make a device allocatable	Enables a device to be allocated to one user at a time.	"How to Make a Device Allocatable" on page 80
Authorize users to allocate a device	Assigns device allocation authorizations to users.	"How to Authorize Users to Allocate a Device" on page 81
View the allocatable devices on your system	Lists the devices that are allocatable, and the state of the device.	"How to View Allocation Information About a Device" on page 82
Forcibly allocate a device	Allocates a device to a user who has an immediate need	"Forcibly Allocating a Device" on page 82
Forcibly deallocate a device	Deallocates a device that is currently allocated to a user	"Forcibly Deallocating a Device" on page 83

Task	Description	For Instructions
Change the allocation properties of a device	Changes the requirements for allocating a device	"How to Change Which Devices Can Be Allocated" on page 83
Create a device-clean script	Purges data from a physical device.	"Writing New Device-Clean Scripts" on page 95
Disable device allocation	Removes allocation restrictions from all devices.	"How to Disable the Auditing Service" on page 587
Audit device allocation	Records device allocation in the audit trail	"How to Audit Device Allocation" on page 84

Managing Device Allocation

Device allocation restricts or prevents access to peripheral devices. Restrictions are enforced at user allocation time. By default, users must have authorization to access allocatable devices.

▼ How to Make a Device Allocatable

If you have already run the `bsmconv` command to enable auditing, then device allocation is already enabled on your system. For more information, see the `bsmconv`(1M) man page.

1 **Assume a role that includes the Audit Control rights profile, or become superuser.**

The Primary Administrator role includes the Audit Control rights profile. You can also assign the Audit Control rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2 **Enable device allocation.**

```
# bsmconv
This script is used to enable the Basic Security Module (BSM).
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: checking startup file.
bsmconv: INFO: move aside /etc/rc3.d/S81volmgt.
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation files.

The Basic Security Module is ready.
If there were any errors, please fix them now.
Configure BSM by editing files located in /etc/security.
Reboot this system now to come up with BSM enabled.
```

Note – The Volume Management daemon (`/etc/rc3.d/S81volmgt`) is disabled by this command.

▼ How to Authorize Users to Allocate a Device

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Create a rights profile that contains the appropriate authorization and commands.**

   Typically, you would create a rights profile that includes the `solaris.device.allocate` authorization. Follow the instructions in “How to Create or Change a Rights Profile” on page 215. Give the rights profile appropriate properties, such as the following:

   - Rights profile name: `Device Allocation`
   - Granted authorizations: `solaris.device.allocate`
   - Commands with security attributes: `mount` with the `sys_mount` privilege, and `umount` with the `sys_mount` privilege

3. **Create a role for the rights profile.**

   Follow the instructions in “How to Create and Assign a Role By Using the GUI” on page 199. Use the following role properties as a guide:

   - Role name: `devicealloc`
   - Role full name: `Device Allocator`
   - Role description: `Allocates and mounts allocated devices`
   - Rights profile: `Device Allocation`

     This rights profile must be at the top of the list of profiles that are included in the role.

4. **Assign the role to every user who is permitted to allocate a device.**

5. **Teach the users how to use device allocation.**

   For examples of allocating removable media, see “How to Allocate a Device” on page 85.

   Because the Volume Management daemon (`vold`) is not running, removable media are not automatically mounted. For examples of mounting a device that has been allocated, see “How to Mount an Allocated Device” on page 86.

▼ How to View Allocation Information About a Device

Before You Begin Device allocation must be enabled for this procedure to succeed. To enable device allocation, see "How to Make a Device Allocatable" on page 80.

1. **Assume a role that includes the Device Security rights profile, or become superuser.**

   The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2. **Display information about allocatable devices on your system.**

   ```
   # list_devices device-name
   ```

   where *device-name* is one of the following:

   - `audio`[*n*] – Is a microphone and speaker.
   - `fd`[*n*] – Is a diskette drive.
   - `sr`[*n*] – Is a CD-ROM drive.
   - `st`[*n*] – Is a tape drive.

Troubleshooting If the `list_devices` command returns an error message similar to the following, then either device allocation is not enabled, or you do not have sufficient permissions to retrieve the information.

```
list_devices: No device maps file entry for specified device.
```

For the command to succeed, enable device allocation and assume a role with the `solaris.device.revoke` authorization.

▼ Forcibly Allocating a Device

Forcible allocation is used when someone has forgotten to deallocate a device. Forcible allocation can also be used when a user has an immediate need for a device.

Before You Begin The user or role must have the `solaris.device.revoke` authorization.

1. **Determine if you have the appropriate authorizations in your role.**

   ```
   $ auths
   solaris.device.allocate solaris.device.revoke
   ```

2. **Forcibly allocate the device to the user who needs the device.**

   In this example, the tape drive is forcibly allocated to the user `jdoe`.

   ```
   $ allocate -U jdoe
   ```

▼ Forcibly Deallocating a Device

Devices that a user has allocated are not automatically deallocated when the process terminates or when the user logs out. Forcible deallocation is used when a user has forgotten to deallocate a device.

Before You Begin The user or role must have the `solaris.device.revoke` authorization.

1. **Determine if you have the appropriate authorizations in your role.**

```
$ auths
solaris.device.allocate solaris.device.revoke
```

2. **Forcibly deallocate the device.**

In this example, the printer is forcibly deallocated. The printer is now available for allocation by another user.

```
$ deallocate -f /dev/lp/printer-1
```

▼ How to Change Which Devices Can Be Allocated

1. **Assume a role that includes the Device Security rights profile, or become superuser.**

The Primary Administrator role includes the Device Security rights profile. You can also assign the Device Security rights profile to a role that you create. To create the role and assign the role to a user, see Example 9–3.

2. **Specify if authorization is required, or specify the `solaris.device.allocate` authorization.**

Change the fifth field in the device entry in the `device_allocate` file.

```
audio;audio;reserved;reserved;solaris.device.allocate;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;solaris.device.allocate;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean
```

where `solaris.device.allocate` indicates that a user must have the `solaris.device.allocate` authorization to use the device.

Example 4–4 Permitting Any User to Allocate a Device

In the following example, any user on the system can allocate any device. The fifth field in every device entry in the `device_allocate` file has been changed to an at sign (`@`).

```
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;@;/etc/security/lib/audio_clean
```

```
fd0;fd;reserved;reserved;@;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;@;/etc/security/lib/sr_clean
...
```

Example 4–5 Preventing Some Peripheral Devices From Being Used

In the following example, the audio device cannot be used. The fifth field in the audio device entry in the `device_allocate` file has been changed to an asterisk (*).

```
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;solaris device.allocate;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;solaris device.allocate;/etc/security/lib/sr_clean
...
```

Example 4–6 Preventing All Peripheral Devices From Being Used

In the following example, no peripheral device can be used. The fifth field in every device entry in the `device_allocate` file has been changed to an asterisk (*).

```
$ whoami
devicesec
$ vi /etc/security/device_allocate
audio;audio;reserved;reserved;*;/etc/security/lib/audio_clean
fd0;fd;reserved;reserved;*;/etc/security/lib/fd_clean
sr0;sr;reserved;reserved;*;/etc/security/lib/sr_clean
...
```

▼ How to Audit Device Allocation

By default, the device allocation commands are in the `other` audit class.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **Preselect the `ot` class for auditing.**

   Add the `ot` class to the `flags` line of the `audit_control` file. The file would appear similar to the following:

```
# audit_control file
dir:/var/audit
```

```
flags:lo,ot
minfree:20
naflags:lo
```

For detailed instructions, see "How to Modify the `audit_control` File" on page 571.

Allocating Devices (Task Map)

The following task map points to procedures that show users how to allocate devices.

Task	Description	For Instructions
Allocate a device	Enables a user to use a device, while preventing any other user from using the device.	"How to Allocate a Device" on page 85
Mount an allocated device	Enables a user to view a device that requires mounting, such as a CD-ROM or a diskette.	"How to Mount an Allocated Device" on page 86
Deallocate a device	Makes an allocatable device available for use by another user.	"How to Deallocate a Device" on page 88

Allocating Devices

Device allocation reserves the use of a device to one user at a time. Devices that require a mount point must be mounted.

▼ How to Allocate a Device

Before You Begin Device allocation must be enabled, as described in "How to Make a Device Allocatable" on page 80. If authorization is required, the user must have the authorization.

1. **Allocate the device.**

   Specify the device by device name.

   ```
   % allocate device-name
   ```

2. **Verify that the device is allocated.**

   Run the identical command.

   ```
   % allocate device-name
   allocate. Device already allocated.
   ```

Example 4–7 Allocating a Microphone

In this example, the user `jdoe` allocates a microphone, `audio`.

```
% whoami
jdoe
% allocate audio
```

Example 4–8 Allocating a Printer

In this example, a user allocates a printer. No one else can print to `printer-1` until the user deallocates it, or until the printer is forcibly allocated to another user.

```
% allocate /dev/lp/printer-1
```

For an example of forcible deallocation, see "Forcibly Deallocating a Device" on page 83.

Example 4–9 Allocating a Tape Drive

In this example, the user `jdoe` allocates a tape drive, `st0`.

```
% whoami
jdoe
% allocate st0
```

Troubleshooting If the `allocate` command cannot allocate the device, an error message is displayed in the console window. For a list of allocation error messages, see the `allocate`(1) man page.

▼ How to Mount an Allocated Device

Before You Begin The user or role has allocated the device. To mount a device, the user or role must have the privileges that are required for mounting the device. To give the required privileges, see "How to Authorize Users to Allocate a Device" on page 81.

1. **Assume a role that can allocate and mount a device.**

```
% su - role-name
Password:       <Type role-name password>
$
```

2. **Create and protect a mount point in the role's home directory.**

   You only need to do this step the first time you need a mount point.

```
$ mkdir mount-point ; chmod 700 mount-point
```

3. **List the allocatable devices.**

```
$ list_devices -l
List of allocatable devices
```

4 **Allocate the device.**

Specify the device by device name.

```
$ allocate device-name
```

5 **Mount the device.**

```
$ mount -o ro -F filesystem-type device-path mount-point
```

where

`-o ro`	Indicates that the device is to be mounted read-only. Use `-o rw` to indicate that you should be able to write to the device.
`-F` *filesystem-type*	Indicates the file system format of the device. Typically, a CD-ROM is formatted with an HSFS file system. A diskette is typically formatted with a PCFS file system.
device-path	Indicates the path to the device. The output of the `list_devices -l` command includes the *device-path*.
mount-point	Indicates the mount point that you created in Step 2.

Example 4–10 Allocating a Diskette Drive

In this example, a user assumes a role that can allocate and mount a diskette drive, `fd0`. The diskette is formatted with a PCFS file system.

```
% roles
devicealloc
% su - devicealloc
Password:       <Type devicealloc password>
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
...
device: fd0 type: fd files: /dev/diskette /dev/rdiskette /dev/fd0a
...
$ allocate fd0
$ mount -o ro -F pcfs /dev/diskette /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
List of the contents of diskette
```

Example 4–11 Allocating a CD-ROM Drive

In this example, a user assumes a role that can allocate and mount a CD-ROM drive, `sr0`. The drive is formatted as an HSFS file system.

```
% roles
devicealloc
% su - devicealloc
Password:       <Type devicealloc password>
$ mkdir /home/devicealloc/mymnt
$ chmod 700 /home/devicealloc/mymnt
$ list_devices -l
...
device: sr0 type: sr files: /dev/sr0 /dev/rsr0 /dev/dsk/c0t2d0s0 ...
...
$ allocate sr0
$ mount -o ro -F hsfs /dev/sr0 /home/devicealloc/mymnt
$ cd /home/devicealloc/mymnt ; ls
List of the contents of CD-ROM
```

Troubleshooting If the `mount` command cannot mount the device, an error message is displayed: `mount: insufficient privileges`. Check the following:

- Make sure that you are executing the `mount` command in a profile shell. If you have assumed a role, the role has a profile shell. If you are a user who has been assigned a profile with the `mount` command, you must create a profile shell. The commands `pfsh`, `pfksh`, and `pfcsh` create a profile shell.
- Make sure that you own the specified mount point. You should have read, write, and execute access to the mount point.

Contact your administrator if you still cannot mount the allocated device.

▼ How to Deallocate a Device

Deallocation enables other users to allocate and use the device when you are finished.

Before You Begin You must have allocated the device.

1 If the device is mounted, unmount the device.

```
$ cd $HOME
$ umount mount-point
```

2 Deallocate the device.

```
$ deallocate device-name
```

Example 4–12 Deallocating a Microphone

In this example, the user `jdoe` deallocates the microphone, `audio`.

```
% whoami
jdoe
% deallocate audio
```

Example 4–13 Deallocating a CD-ROM Drive

In this example, the Device Allocator role deallocates a CD-ROM drive. After the message is printed, the CD-ROM is ejected.

```
$ whoami
devicealloc
$ cd /home/devicealloc
$ umount /home/devicealloc/mymnt
$ ls /home/devicealloc/mymnt
$
$ deallocate sr0
/dev/sr0:      326o
/dev/rsr0:     326o
...
sr_clean: Media in sr0 is ready.  Please, label and store safely.
```

Device Protection (Reference)

Devices in the Solaris OS are protected by device policy. Peripheral devices can be protected by device allocation. Device policy is enforced by the kernel. Device allocation is optionally enabled, and is enforced at the user level.

Device Policy Commands

Device management commands administer the device policy on local files. Device policy can include privilege requirements. Only superuser or a role of equivalent capabilities can manage devices.

The following table lists the device management commands.

TABLE 4–1 Device Management Commands

Command	Purpose	Man Page
`devfsadm`	Administers devices and device drivers on a running system. Also loads device policy. The `devfsadm` command enables the cleanup of dangling `/dev` links to disk, tape, port, audio, and pseudo devices. Devices for a named driver can also be reconfigured.	`devfsadm`(1M)
`getdevpolicy`	Displays the policy associated with one or more devices. This command can be run by any user.	`getdevpolicy`(1M)
`add_drv`	Adds a new device driver to a running system. Contains options to add device policy to the new device. Typically, this command is called in a script when a device driver is being installed.	`add_drv`(1M)
`update_drv`	Updates the attributes of an existing device driver. Contains options to update the device policy for the device. Typically, this command is called in a script when a device driver is being installed.	`update_drv`(1M)
`rem_drv`	Removes a device or device driver.	`rem_drv`(1M)

Device Allocation

Device allocation can protect your site from loss of data, computer viruses, and other security breaches. Unlike device policy, device allocation is optional. Devices are not allocatable until the `bsmconv` script is run. Device allocation uses authorizations to limit access to allocatable devices.

Components of Device Allocation

The components of the device allocation mechanism are as follows:

- The `allocate`, `deallocate`, `dminfo`, and `list_devices` commands. For more information, see "Device Allocation Commands" on page 91.
- Device-clean scripts for each allocatable device.

These commands and scripts use the following local files to implement device allocation:

- The `/etc/security/device_allocate` file. For more information, see the `device_allocate`(4) man page.
- The `/etc/security/device_maps` file. For more information, see the `device_maps`(4) man page.
- A lock file, in the `/etc/security/dev` directory, for each allocatable device.
- The changed attributes of the lock files that are associated with each allocatable device.

Note – The `/etc/security/dev` directory might not be supported in future releases of the Solaris OS.

Device Allocation Commands

With uppercase options, the `allocate`, `deallocate`, and `list_devices` commands are administrative commands. Otherwise, these commands are user commands. The following table lists the device allocation commands.

TABLE 4–2 Device Allocation Commands

Command	Purpose	Man Page
`bsmconv`	Creates databases to handle device allocation. Also enables the auditing service. You must be superuser or in the Primary Administrator role.	`bsmconv`(1M)
`dminfo`	Searches for an allocatable device by device type, by device name, and by full path name.	`dminfo`(1M)
`list_devices`	Lists the status of allocatable devices. Lists all the device-special files that are associated with any device that is listed in the `device_maps` file.	`list_devices`(1)
`list_devices -U`	Lists the devices that are allocatable or allocated to the specified user ID. This option allows you to check which devices are allocatable or allocated to another user. You must have the `solaris.device.revoke` authorization.	
`allocate`	Reserves an allocatable device for use by one user. By default, a user must have the `solaris.device.allocate` authorization to allocate a device. You can modify the `device_allocate` file to not require user authorization. Then, any user on the system can request the device to be allocated for use.	`allocate`(1)
`deallocate`	Removes the allocation reservation from a device.	`deallocate`(1)

Authorizations for the Allocation Commands

By default, users must have the `solaris.device.allocate` authorization to reserve an allocatable device. To create a rights profile to include the `solaris.device.allocate` authorization, see "How to Authorize Users to Allocate a Device" on page 81.

Administrators must have the `solaris.device.revoke` authorization to change the allocation state of any device. For example, the `-U` option to the `allocate` and `list_devices` commands, and the `-F` option to the `deallocate` command require the `solaris.device.revoke` authorization.

For more information, see "Commands That Require Authorizations" on page 237.

Allocate Error State

A device is put in an *allocate error state* when the `deallocate` command fails to deallocate, or when the `allocate` command fails to allocate. When an allocatable device is in an allocate error state, then the device must be forcibly deallocated. Only superuser or a role with the Device Management rights profile or the Device Security rights profile can handle an allocate error state.

The `deallocate` command with the `-F` option forces deallocation. Or, you can use `allocate -U` to assign the device to a user. Once the device is allocated, you can investigate any error messages that appear. After any problems with the device are corrected, you can forcibly deallocate it.

`device_maps` File

Device maps are created when you set up device allocation. A default `/etc/security/device_maps` file is created by the `bsmconv` command when the auditing service is enabled. This initial `device_maps` file can be customized for your site. The `device_maps` file includes the device names, device types, and device-special files that are associated with each allocatable device.

The `device_maps` file defines the device-special file mappings for each device, which in many cases is not intuitive. This file allows programs to discover which device-special files map to which devices. You can use the `dminfo` command, for example, to retrieve the device name, the device type, and the device-special files to specify when you set up an allocatable device. The `dminfo` command uses the `device_maps` file to report this information.

Each device is represented by a one-line entry of the form:

device-name:*device-type*:*device-list*

EXAMPLE 4–14 Sample `device_maps` Entry

The following is an example of an entry in a `device_maps` file for a diskette drive, `fd0`:

```
fd0:\
        fd:\
        /dev/diskette /dev/rdiskette /dev/fd0a /dev/rfd0a \
/dev/fd0b /dev/rfd0b /dev/fd0c /dev/fd0 /dev/rfd0c /dev/rfd0:\
```

Lines in the `device_maps` file can end with a backslash (`\`) to continue an entry on the next line. Comments can also be included. A pound sign (`#`) comments all subsequent text until the next newline that is not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field. The fields are defined as follows:

device-name	Specifies the name of the device. For a list of current device names, see "How to View Allocation Information About a Device" on page 82.
device-type	Specifies the generic device type. The generic name is the name for the class of devices, such as `st`, `fd`, or `audio`. The *device-type* field logically groups related devices.
device-list	Lists the device-special files that are associated with the physical device. The *device-list* must contain all of the special files that allow access to a particular device. If the list is incomplete, a malevolent user can still obtain or modify private information. Valid entries for the *device-list* field reflect the device files that are located in the `/dev` directory.

`device_allocate` **File**

An initial `/etc/security/device_allocate` file is created by the `bsmconv` command when the auditing service is enabled. This initial `device_allocate` file can be used as a starting point. You can modify the `device_allocate` file to change devices from allocatable to nonallocatable, or to add new devices. A sample `device_allocate` file follows.

```
st0;st;;;;/etc/security/lib/st_clean
fd0;fd;;;;/etc/security/lib/fd_clean
sr0;sr;;;;/etc/security/lib/sr_clean
audio;audio;;;*;/etc/security/lib/audio_clean
```

An entry in the `device_allocate` file does not mean that the device is allocatable, unless the entry specifically states that the device is allocatable. In the sample `device_allocate` file, note the asterisk (`*`) in the fifth field of the audio device entry. An asterisk in the fifth field indicates to the system that the device is not allocatable. Therefore, the device cannot be used. Other values or no value in this field indicates that the device can be used.

In the `device_allocate` file, each device is represented by a one-line entry of the form:

device-name`;`*device-type*`;reserved;reserved;`*auths*`;`*device-exec*

Lines in the `device_allocate` file can end with a backslash (`\`) to continue an entry on the next line. Comments can also be included. A pound sign (`#`) comments all subsequent text until the next newline that is not immediately preceded by a backslash. Leading and trailing blanks are allowed in any field. The fields are defined as follows:

device-name	Specifies the name of the device. For a list of current device names, see "How to View Allocation Information About a Device" on page 82.
device-type	Specifies the generic device type. The generic name is the name for the class of devices, such as `st`, `fd`, and `sr`. The *device-type* field logically groups related devices. When you make a device allocatable, retrieve the device name from the *device-type* field in the `device_maps` file.

`reserved`	Sun reserves the two fields that are marked `reserved` for future use.
auths	Specifies whether the device is allocatable. An asterisk (`*`) in this field indicates that the device is not allocatable. An authorization string, or an empty field, indicates that the device is allocatable. For example, the string `solaris.device.allocate` in the *auths* field indicates that the `solaris.device.allocate` authorization is required to allocate the device. An at sign (`@`) in this file indicates that the device is allocatable by any user.
device-exec	Supplies the path name of a script to be invoked for special handling, such as cleanup and object-reuse protection during the allocation process. The *device-exec* script is run any time that the device is acted on by the `deallocate` command.

For example, the following entry for the `sr0` device indicates that the CD-ROM drive is allocatable by a user with the `solaris.device.allocate` authorization:

```
sr0;sr;reserved;reserved;solaris.device.allocate;/etc/security/lib/sr_clean
```

You can decide to accept the default devices and their defined characteristics. After you install a new device, you can modify the entries. Any device that needs to be allocated before use must be defined in the `device_allocate` and `device_maps` files for that device's system. Currently, cartridge tape drives, diskette drives, CD-ROM drives, and audio chips are considered allocatable. These device types have device-clean scripts.

Note – Xylogics™ tape drives or Archive tape drives also use the `st_clean` script that is supplied for SCSI devices. You need to create your own device-clean scripts for other devices, such as modems, terminals, graphics tablets, and other allocatable devices. The script must fulfill object-reuse requirements for that type of device.

Device-Clean Scripts

Device allocation satisfies part of what is called the object reuse requirement. The *device-clean* scripts address the security requirement that all usable data be purged from a physical device before reuse. The data is cleared before the device is allocatable by another user. By default, cartridge tape drives, diskette drives, CD-ROM drives, and audio devices require device-clean scripts. The Solaris OS provides the scripts. This section describes what device-clean scripts do.

Device-Clean Script for Tapes

The `st_clean` device-clean script supports three tape devices:

- SCSI ¼-inch tape
- Archive ¼-inch tape
- Open-reel ½-inch tape

The `st_clean` script uses the `rewoffl` option to the `mt` command to clean up the device. For more information, see the `mt`(1) man page. If the script runs during system boot, the script queries the device to determine if the device is online. If the device is online, the script determines if the device has media in it. The ¼-inch tape devices that have media in them are placed in the allocate error state. The allocate error state forces the administrator to manually clean up the device.

During normal system operation, when the `deallocate` command is executed in interactive mode, the user is prompted to remove the media. Deallocation is delayed until the media is removed from the device.

Device-Clean Scripts for Diskettes and CD-ROM Drives

The following device-clean scripts are provided for diskettes and CD-ROM drives:

- **`fd_clean` script –** Is a device-clean script for diskettes.
- **`sr_clean` script –** Is a device-clean script for CD-ROM drives.

The scripts use the `eject` command to remove the media from the drive. If the `eject` command fails, the device is placed in the allocate error state. For more information, see the `eject`(1) man page.

Device-Clean Script for Audio

Audio devices are cleaned up with an `audio_clean` script. The script performs an `AUDIO_GETINFO` ioctl system call to read the device. The script then performs an `AUDIO_SETINFO` ioctl system call to reset the device configuration to the default.

Writing New Device-Clean Scripts

If you add more allocatable devices to the system, you might need to create your own device-clean scripts. The `deallocate` command passes a parameter to the device-clean scripts. The parameter, which is shown here, is a string that contains the device name. For more information, see the `device_allocate`(4) man page.

clean-script `-[I|i|f|S]` *device-name*

Device-clean scripts must return “`0`” for success and greater than “`0`” for failure. The options `-I`, `-f`, and `-S` determine the running mode of the script:

`-I` Is needed during system boot only. All output must go to the system console. Failure or inability to forcibly eject the media must put the device in the allocate error state.

`-i` Similar to the `-I` option, except that output is suppressed.

`-f`	Is for forced cleanup. The option is interactive and assumes that the user is available to respond to prompts. A script with this option must attempt to complete the cleanup if one part of the cleanup fails.
`-S`	Is for standard cleanup. The option is interactive and assumes that the user is available to respond to prompts.

CHAPTER 5

Using the Basic Audit Reporting Tool (Tasks)

This chapter describes how to create a manifest of the files on a system and how to use that manifest to check the integrity of the system. The Basic Audit Reporting Tool (BART) enables you to comprehensively validate systems by performing file-level checks of a system over time.

The following is a list of the information in this chapter:

- "Using BART (Task Map)" on page 100
- "Basic Audit Reporting Tool (Overview)" on page 97
- "Using BART (Tasks)" on page 101
- "BART Manifest, Rules File, and Reporting (Reference)" on page 115

Basic Audit Reporting Tool (Overview)

BART is a file tracking tool that operates entirely at the file system level. Using BART gives you the ability to quickly, easily, and reliably gather information about the components of the software stack that is installed on deployed systems. Using BART can greatly reduce the costs of administering a network of systems by simplifying time-consuming administrative tasks.

BART enables you to determine what file-level changes have occurred on a system, relative to a known baseline. You use BART to create a baseline or *control* manifest from a fully installed and configured system. You can then compare this baseline with a snapshot of the system at a later time, generating a report that lists file-level changes that have occurred on the system since it was installed.

The `bart` command is a standard UNIX command. You can redirect the output of the `bart` command to a file for later processing.

BART Features

BART has been designed with an emphasis on a simple syntax that is both powerful and flexible. The tool enables you to generate manifests of a given system over time. Then, when the system's files need to be validated, you can generate a report by comparing the old and new manifests. Another way to use BART is to generate manifests of several similar systems and run system-to-system comparisons. The main difference between BART and existing auditing tools is that BART is flexible, both in terms of what information is tracked and what information is reported.

Additional benefits and uses of BART include the following:

- Provides an efficient and easy method for cataloging a system that is running the Solaris software at the file level.
- Enables you to define which files to monitor and gives you the ability to modify profiles when necessary. This flexibility allows you to monitor local customizations and enables you to reconfigure software easily and efficiently.
- Ensures that systems are running reliable software.
- Allows you to monitor file-level changes of a system over time, which can help you locate corrupted or unusual files.
- Helps you troubleshoot system performance issues.

BART Components

BART has two main components and one optional component:

- BART Manifest
- BART Report
- BART Rules File

BART Manifest

You use the `bart create` command to take a file-level snapshot of a system at a particular time. The output is a catalog of files and file attributes called a *manifest*. The manifest lists information about all the files or specific files on a system. It contains information about attributes of files, which can include some uniquely identifying information, such as an MD5 checksum. For more information about the MD5 checksum, see the `md5`(3EXT) man page. A manifest can be stored and transferred between client and server systems.

Note – BART does *not* cross file system boundaries, with the exception of file systems of the same type. This constraint makes the output of the `bart create` command more predictable. For example, without arguments, the `bart create` command catalogs all UFS file systems under the root (`/`) directory. However, no NFS or TMPFS file systems or mounted CD-ROMs would be cataloged. When creating a manifest, do not attempt to audit file systems on a network. Note that using BART to monitor networked file systems can consume large resources to generate manifests that will have little value.

For more information about BART manifests, see "BART Manifest File Format" on page 115.

BART Report

The report tool has three inputs: the two manifests to be compared and an optional user-provided rules file that indicates which discrepancies are to be flagged.

You use the `bart compare` command to compare two manifests, a *control manifest* and a *test manifest*. These manifests must be prepared with the same file systems, options, and rules file that you use with the `bart create` command.

The output of the `bart compare` command is a report that lists per-file discrepancies between the two manifests. A *discrepancy* is a change to any attribute for a given file that is cataloged for both manifests. Additions or deletions of file entries between the two manifests are also considered discrepancies.

There are two levels of control when reporting discrepancies:

- When generating a manifest
- When producing reports

These levels of control are intentional, since generating a manifest is more costly than reporting discrepancies between two manifests. Once you have created manifests, you have the ability to compare manifests from different perspectives by running the `bart compare` command with different rules files.

For more information about BART reports, see "BART Reporting" on page 117.

BART Rules File

The *rules file* is a text file that you can optionally use as input to the `bart` command. This file uses inclusion and exclusion rules. A rules file is used to create custom manifests and reports. A rules file enables you to express in a concise syntax which sets of files you want to catalog, as well as which attributes to monitor for any given set of files. When you compare manifests, the rules file aids in flagging discrepancies between the manifests. Using a rules file is an effective way to gather specific information about files on a system.

You create a rules file by using a text editor. With a rules file, you can perform the following tasks:

- Use the `bart create` command to create a manifest that lists information about all or specific files on a system.
- Use the `bart compare` command to generate a report that monitors specific attributes of a file system.

Note – You can create several rules files for different purposes. However, if you create a manifest by using a rules file, you must use the same rules file when you compare the manifests. If you do not use the same rules file when comparing manifests that were created with a rules file, the output of the `bart compare` command will list many invalid discrepancies.

A rules file can also contain syntax errors and other ambiguous information as a result of user error. If a rules file does contain misinformation, these errors will also be reported.

Using a rules file to monitor specific files and file attributes on a system requires planning. Before you create a rules file, decide which files and file attributes on the system you want to monitor. Depending on what you are trying to accomplish, you might use a rules file to create manifests, compare manifests, or for purposes.

For more information about the BART rules file, see "BART Rules File Format" on page 116 and the `bart_rules`(4) man page.

Using BART (Task Map)

Task	Description	For Instructions
Create a manifest.	Obtain a manifest that lists information about every file that is installed on a system.	"How to Create a Manifest" on page 101
Create a custom manifest.	Obtain a manifest that lists information about specific files that are installed on a system in one of the following ways: ■ By specifying a subtree ■ By specifying a file name ■ By using a rules file	"How to Customize a Manifest" on page 104

Task	Description	For Instructions
Compare manifests for the same system over time. Or, compare manifests for different systems with a control system manifest.	Obtain a report that compares changes to a system over time. Or, obtain a report that compares one or several systems to control system.	"How to Compare Manifests for the Same System Over Time" on page 107 "How to Compare Manifests From a Different System With the Manifest of a Control System" on page 110
(Optional) Customize a BART report.	Obtain a custom BART report in one of the following ways: ■ By specifying attributes. ■ By using a rules file.	"How to Customize a BART Report by Specifying File Attributes" on page 112 "How to Customize a BART Report by Using a Rules File" on page 113

Using BART (Tasks)

You can run the `bart` command as a regular user, superuser, or a user who has assumed the Primary Administrator role. If you run the `bart` command as a regular user, you will only be able to catalog and monitor files that you have permission to access, for example, information about files in your home directory. The advantage of becoming superuser when you run the `bart` command is that the manifests you create will contain information about hidden and private files that you might want to monitor. If you need to catalog and monitor information about files that have restricted permissions, for example, the `/etc/passwd` or `/etc/shadow` file, run the `bart` command as superuser or assume an equivalent role. For more information about using role-based access control, see "Configuring RBAC (Task Map)" on page 196.

BART Security Considerations

Running the `bart` command as superuser makes the output readable by anyone. This output might contain file names that are intended to be private. If you become superuser when you run the `bart` command, take appropriate measures to protect the output. For example, use options that generate output files with restrictive permissions.

Note – The procedures and examples in this chapter show the `bart` command run by superuser. Unless otherwise specified, running the `bart` command as superuser is optional.

▼ How to Create a Manifest

You can create a manifest of a system immediately after an initial Solaris software installation. This type of manifest will provide you with a baseline for comparing changes to the same system over time. Or, you can use this manifest to compare with the manifests for different systems. For example, if you take a snapshot of each system on your network, and then compare each test manifest with the control manifest, you can quickly determine what you need to do to synchronize the test system with the baseline configuration.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **After installing the Solaris software, create a control manifest and redirect the output to a file.**

`# bart create` *options* > *control-manifest*

`-R`	Specifies the root directory for the manifest. All paths specified by the rules will be interpreted relative to this directory. All paths reported in the manifest will be relative to this directory.
`-I`	Accepts a list of individual files to be cataloged, either on the command line or read from standard input.
`-r`	Is the name of the rules file for this manifest. Note that –, when used with the `-r` option, will be read the rules file from standard input.
`-n`	Turns off content signatures for all regular files in the file list. This option can be used to improve performance. Or, you can use this option if the contents of the file list are expected to change, as in the case of system log files.

3 **Examine the contents of the manifest.**

4 **Save the manifest for future use.**

Choose a meaningful name for the manifest. For example, use the system name and date that the manifest was created.

Example 5–1 Creating a Manifest That Lists Information About Every File on a System

If you run the `bart create` command without any options, information about every file that is installed on the system will be cataloged. Use this type of manifest as a baseline when you are installing many systems from a central image. Or, use this type of manifest to run comparisons when you want to ensure that the installations are identical.

For example:

```
# bart create
! Version 1.0
! Thursday, December 04, 2003 (16:17:39)
# Format:
```

```
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 1024 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9ea47 0 0
/.java D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f8dc04d 0 10
/.java/.userPrefs D 512 40700 user::rwx,group::---,mask:---
other:--- 3f8dc06b 010
/.java/.userPrefs/.user.lock.root F 0 100600 user::rw-
group::---,mask:---,other:--- 3f8dc06b 0 10 -
/.java/.userPrefs/.userRootModFile.root F 0 100600 user::rw-,
group::---,mask:---,other:--- 3f8dc0a1 0 10 -
/.smc.properties F 1389 100644 user::rw-,group::r--,mask:r--
other:r-- 3f8dca0c0 10
.
.
.
/var/sadm/pkg/SUNWdtmad/install/depend F 932 100644 user::rw-,
group::r--,mask:r--,other:r-- 3c23a19e 0 0 -
/var/sadm/pkg/SUNWdtmad/pkginfo F 594 100644 user::rw-
group::r--,mask:r--,other:r-- 3f81e416 0 0 -
/var/sadm/pkg/SUNWdtmad/save D 512 40755 user::rwx,group::r-x
mask:r-x,other:r-x 3f81e416 0 0
/var/sadm/pkg/SUNWdtmaz D 512 40755 user::rwx,group::r-x
mask:r-x,other:r-x 3f81e41b 0 0
/var/sadm/pkg/TSIpgxw/save D 512 40755 user::rwx
group::r-x,mask:r-x,other:r-x 3f81e892 0 0
.
.
.
```

Each manifest consists of a header and entries. Each manifest file entry is a single line, depending on the file type. For example, for each manifest entry in the preceding output, type `F` specifies a file and type `D` specifies a directory. Also listed is information about size, content, user ID, group ID, and permissions. File entries in the output are sorted by the encoded versions of the file names to correctly handle special characters. All entries are sorted in ascending order by file name. All nonstandard file names, such as those that contain embedded newline or tab characters, have the nonstandard characters quoted before being sorted.

Lines that begin with `!` supply metadata about the manifest. The manifest version line indicates the manifest specification version. The date line shows the date on which the manifest was created, in date form. See the `date`(1) man page. Some lines are ignored by the manifest comparison tool. Ignored lines include blank lines, lines that consist only of white space, and comments that begin with `#`.

▼ How to Customize a Manifest

You can customize a manifest in one of the following ways:

- By specifying a subtree

  Creating a manifest for an individual subtree on a system is an efficient way to monitor changes to specific files, rather than the entire contents of a large directory. You can create a baseline manifest of a specific subtree on your system, then periodically create test manifests of the same subtree. Use the `bart compare` command to compare the control manifest with the test manifest. By using this option, you are able to efficiently monitor important file systems to determine whether any files have been compromised by an intruder.
- By specifying a file name

  Since creating a manifest that catalogs the entire system is more time-consuming, takes up more space, and is more costly, you might choose to use this option of the `bart` command when you want to only list information about a specific file or files on a system.
- By using a rules file

  You use a rules file to create custom manifests that list information about specific files and specific subtrees on a given system. You can also use a rules file to monitor specific file attributes. Using a rules file to create and compare manifests gives you the flexibility to specify multiple attributes for more than one file or subtree. Whereas, from the command line, you can only specify a global attribute definition that applies to all files for each manifest you create or report you generate.

1. **Determine which files you want to catalog and monitor.**

2. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

3. **After installing the Solaris software, create a custom manifest by using one of the following options:**

   - By specifying a subtree:

     ```
     # bart create -R root-directory
     ```
   - By specifying a file name or file names:

     ```
     # bart create -I filename...
     ```

     For example:

     ```
     # bart create -I /etc/system /etc/passwd /etc/shadow
     ```

- By using a rules file:

```
# bart create -r rules-file
```

4 **Examine the contents of the manifest.**

5 **Save the manifest for future use.**

Example 5–2 Creating a Manifest by Specifying a Subtree

This example shows how to create a manifest that contains information about the files in the `/etc/ssh` subtree only.

```
# bart create -R /etc/ssh
! Version 1.0
! Saturday, November 29, 2003 (14:05:36)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f81eab9 0 3
/ssh_config F 861 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81e504 0 3 422453ca0e2348cd9981820935600395
/ssh_host_dsa_key F 668 100600 user::rw-,group::---,mask:---,
other:--- 3f81eab9 0 0 5cc28cdc97e833069fd41ef89e4d9834
/ssh_host_dsa_key.pub F 602 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81eab9 0 0 16118c736995a4e4754f5ab4f28cf917
/ssh_host_rsa_key F 883 100600 user::rw-,group::---,mask:---,
other:--- 3f81eaa2 0 0 6ff17aa968ecb20321c448c89a8840a9
/ssh_host_rsa_key.pub F 222 100644 user::rw-,group::r--,mask:r--,
other:r-- 3f81eaa2 0 0 9ea27617efc76058cb97aa2caa6dd65a
.
.
.
```

Example 5–3 Customizing a Manifest by Specifying a File Name

This example shows how to create a manifest that lists only information about the `/etc/passwd` and `/etc/shadow` files on a system.

```
# bart create -I /etc/passwd /etc/shadow
! Version 1.0
! Monday, December 15, 2003 (16:28:55)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/etc/passwd F 542 100444 user::r--,group::r--,mask:r--,
other:r-- 3fcfd45b 0 3 d6
84554f85d1de06219d80543174ad1a
/etc/shadow F 294 100400 user::r--,group::---,mask:---,
other:--- 3f8dc5a0 0 3 fd
c3931c1ae5ee40341f3567b7cf15e2
```

By comparison, the following is the standard output of the `ls -al` command for the `/etc/passwd` and the `/etc/shadow` files on the same system.

```
# ls -al /etc/passwd
-r--r--r--   1 root     sys          542 Dec  4 17:42 /etc/passwd

# ls -al /etc/shadow
-r--------   1 root     sys          294 Oct 15 16:09 /etc/shadow
```

Example 5–4 Customizing a Manifest by Using a Rules File

This example shows how to create a manifest by using a rules file to catalog only those files in the `/etc` directory. The same rules file includes directives to be used by the `bart compare` command for monitoring changes to the `acl` attribute of the `/etc/system` file.

- Use a text editor to create a rules file that catalogs only those files in the `/etc` directory.

  ```
  # List information about all the files in the /etc directory.

  CHECK all
  /etc

  # Check only acl changes in the /etc/system file

  IGNORE all
  CHECK acl
  /etc/system
  ```

  For more information about creating a rules file, see "BART Rules File" on page 99.

- Create a control manifest by using the rules file you created.

  ```
  # bart create -r etc.rules-file > etc.system.control-manifest
  ! Version 1.0
  ! Thursday, December 11, 2003 (21:51:32)
  # Format:
  #fname D size mode acl dirmtime uid gid
  #fname P size mode acl mtime uid gid
  #fname S size mode acl mtime uid gid
  #fname F size mode acl mtime uid gid contents
  #fname L size mode acl lnmtime uid gid dest
  #fname B size mode acl mtime uid gid devnode
  #fname C size mode acl mtime uid gid devnode
  /etc/system F 1883 100644 user::rw-,group::r--,mask:r--,
  other:r-- 3f81db61 0 3
  ```

- Create a test manifest whenever you want to monitor changes to the system. Prepare the test manifest identically to the control manifest by using the same `bart` options and the same rules file.
- Compare manifests by using the same rules file.

▼ How to Compare Manifests for the Same System Over Time

Use this procedure when you want to monitor file-level changes to the same system over time. This type of manifest can assist you in locating corrupted or unusual files, detecting security breaches, or in troubleshooting performance issues on a system.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **After installing the Solaris software, create a control manifest of the files that you want to monitor on the system.**

   ```
   # bart create -R /etc > control-manifest
   ```

3. **Create a test manifest that is prepared identically to the control manifest whenever you want monitor changes to the system.**

   ```
   # bart create -R /etc > test-manifest
   ```

4 **Compare the control manifest with the test manifest.**

`# bart compare` *options control-manifest test-manifest* > *bart-report*

`-r`	Is the name of the rules file for this comparison. Using the `-r` option with the – means that the directives will be read from standard input.
`-i`	Allows the user to set global `IGNORE` directives from the command line.
`-p`	Is the programmatic mode that generates standard non-localized output for programmatic parsing.
control-manifest	Is the output from the `bart create` command for the control system.
test-manifest	Is the output from the `bart create` command of the test system.

5 **Examine the BART report for oddities.**

Example 5–5 Comparing Manifests for the Same System Over Time

This example shows how to monitor changes that have occurred in the `/etc` directory between two points in time. This type of comparison enables you to quickly determine whether important files on the system have been compromised.

- Create a control manifest.

```
# bart create -R /etc > system1.control.121203
! Version 1.0
! Friday, December 12, 2003 (08:34:51)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 4096 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9dfb4 0 3
/.cpr_config F 2236 100644 user::rw-,group::r--,mask:r--,other:r--
3fd9991f 0 0
67cfa2c830b4ce3e112f38c5e33c56a2
/.group.lock F 0 100600 user::rw-,group::---,mask:---,other:--- 3f81f14d
0 1 d41
d8cd98f00b204e9800998ecf8427e
/.java D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f81dcb5 0 2
/.java/.systemPrefs D 512 40755 user::rwx,group::r-x,mask:r-x,
other:r-x 3f81dcb7
.
```

```
.
.
```

- Create a test manifest when you want to monitor changes to the `/etc` directory.

```
# bart create -R /etc > system1.test.121503
Version 1.0
! Monday, December 15, 2003 (08:35:28)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 4096 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9dfb4 0 3
/.cpr_config F 2236 100644 user::rw-,group::r--,mask:r--,other:r--
3fd9991f 0 0
67cfa2c830b4ce3e112f38c5e33c56a2
/.group.lock F 0 100600 user::rw-,group::---,mask:---,other:---
3f81f14d 0 1 d41d8cd98f00b204e9800998ecf8427e
/.java D 512 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3f81dcb5 0 2
/.java/.systemPrefs D 512 40755 user::rwx,group::r-x,mask:r-x,
other:r-x 3f81dcb70 2
/.java/.systemPrefs/.system.lock F 0 100644 user::rw-,group::r--
,mask:r--,other:
r-- 3f81dcb5 0 2 d41d8cd98f00b204e9800998ecf8427e
/.java/.systemPrefs/.systemRootModFile F 0 100644 user::rw-,
group::r--,mask:r--,
other:r-- 3f81dd0b 0 2 d41d8cd98f00b204e9800998ecf8427e
.
.
.
```

- Compare the control manifest with the test manifest.

```
# bart compare system1.control.121203 system1.test.121503
/vfstab:
  mode  control:100644  test:100777
  acl  control:user::rw-,group::r--,mask:r--,other:r--  test:user::rwx,
group::rwx,mask:rwx,other:rwx
```

The preceding output indicates permissions on the `vfstab` file have changed since the control manifest was created. This report can be used to investigate whether ownership, date, content, or any other file attributes have changed. Having this type of information readily available can assist you in tracking down who might have tampered with the file and when the change might have occurred.

▼ How to Compare Manifests From a Different System With the Manifest of a Control System

You can run system to system comparisons, thereby enabling you to quickly determine whether there are any file-level differences between a baseline system and the other systems. For example, if you have installed a particular version of the Solaris software on a baseline system, and you want to know whether other systems have identical packages installed, you can create manifests for those systems and then compare the test manifests with the control manifest. This type of comparison will list any discrepancies in the file contents for each test system that you compare with the control system.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **After installing the Solaris software, create a control manifest.**

   ```
   # bart create options > control-manifest
   ```

3. **Save the control manifest.**

4. **On the test system, use the same `bart` options to create a manifest, and redirect the output to a file.**

   ```
   # bart create options > test1-manifest
   ```

   Choose a distinct and meaningful name for the test manifest.

5. **Save the test manifest to a central location on the system until you are ready to compare manifests.**

6. **When you want to compare manifests, copy the control manifest to the location of the test manifest. Or, copy the test manifest to the control system.**

   For example:

   ```
   # cp control-manifest /net/test-server/bart/manifests
   ```

   If the test system is not an NFS-mounted system, use `FTP` or some other reliable means to copy the control manifest to the test system.

7. **Compare the control manifest with the test manifest and redirect the output to a file.**

   ```
   # bart compare control-manifest test1-manifest > test1.report
   ```

8. **Examine the BART report for oddities.**

9 **Repeat Step 4 through Step 9 for each test manifest that you want to compare with the control manifest.**

Use the same `bart` options for each test system.

Example 5–6 Comparing Manifests From Different Systems With the Manifest of a Control System

This example describes how to monitor changes to the contents of the `/usr/bin` directory by comparing a control manifest with a test manifest from a different system.

- Create a control manifest.

```
# bart create -R /usr/bin > control-manifest.121203
!Version 1.0
! Friday, December 12, 2003 (09:19:00)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9e925 0 2
/.s F 14200 104711 user::rwx,group::--x,mask:--x,other:--x
3f8dbfd6 0 1 8ec7e52d8a35ba3b054a6394cbf71cf6
/ControlPanel L 28 120777 - 3f81dc71 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcdc 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5750 0 2 d6e99b19c847ab4ec084d9088c7c7608
/activation-client F 9172 100755 user::rwx,group::r-x,mask:r-x,
other:r-x 3f5cb907 0 1 b3836ad1a656324a6e1bd01edcba28f0
/adb F 9712 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5736 0 2 5e026413175f65fb239ee628a8870eda
/addbib F 11080 100555 user::r-x,group::r-x,mask:r-x,other:r-x
3f6b5803 0 2 a350836c36049febf185f78350f27510
.
.
.
```

- Create a test manifest for each system that you want to compare with the control system.

```
# bart create -R /usr/bin > system2-manifest.121503
! Version 1.0
! Friday, December 15, 2003 (13:30:58)
# Format:
#fname D size mode acl dirmtime uid gid
#fname P size mode acl mtime uid gid
```

```
#fname S size mode acl mtime uid gid
#fname F size mode acl mtime uid gid contents
#fname L size mode acl lnmtime uid gid dest
#fname B size mode acl mtime uid gid devnode
#fname C size mode acl mtime uid gid devnode
/ D 13312 40755 user::rwx,group::r-x,mask:r-x,other:r-x 3fd9ea9c 0 2
/.s F 14200 104711 user::rwx,group::--x,mask:--x,other:--x
3f8dbfd6 0 1 8ec7e52d8a35ba3b054a6394cbf71cf6
/ControlPanel L 28 120777 - 3f81dc71 0 1 jre/bin/ControlPanel
/HtmlConverter L 25 120777 - 3f81dcdc 0 1 bin/HtmlConverter
/acctcom F 28300 100555 user::r-x,group::r-x,mask:r-x,other:
r-x 3f6b5750 0 2 d6e99b19c847ab4ec084d9088c7c7608
.
.
.
```

- When you want to compare manifests, copy the manifests to the same location.

```
# cp control-manifest /net/system2.central/bart/manifests
```

- Compare the control manifest with the test manifest.

```
# bart compare control-manifest  system2.test > system2.report
/su:
  gid  control:3  test:1
/ypcat:
  mtime  control:3fd72511  test:3fd9eb23
```

The previous output indicates that the group ID of the `su` file in the `/usr/bin` directory is not the same as that of the control system. This information can be helpful in determining whether a different version of the software was installed on the test system or if possibly someone has tampered with the file.

▼ How to Customize a BART Report by Specifying File Attributes

This procedure is optional and explains how to customize a BART report by specifying file attributes from the command line. If you create a baseline manifest that lists information about all the files or specific on your system, you can run the `bart compare` command, specifying different attributes, whenever you need to monitor changes to a particular directory, subdirectory, file or files. You can run different types of comparisons for the same manifests by specifying different file attributes from the command line.

1 **Determine which file attributes you want to monitor.**

2 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

3 **After installing the Solaris software, create a control manifest.**

4 **Create a test manifest when you want to monitor changes.**

Prepare the test manifest identically to the control manifest.

5 **Compare the manifests.**

For example:

```
# bart compare -i dirmtime,lnmtime,mtime control-manifest.121503 \
test-manifest.010504 > bart.report.010504
```

Note that a comma separates each attribute you specify in the command-line syntax.

6 **Examine the BART report for oddities.**

▼ How to Customize a BART Report by Using a Rules File

This procedure is also optional and explains how to customize a BART report by using a rules file as input to the `bart compare` command. By using a rules file, you can customize a BART report, which allows you the flexibility of specifying multiple attributes for more than one file or subtree. You can run different comparisons for the same manifests by using different rules files.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Determine which files and file attributes you want to monitor.**

3 **Use a text editor to create a rules file with the appropriate directives.**

4 **After installing the Solaris software, create a control manifest by using the rules file you created.**

```
# bart create -r rules-file > control-manifest
```

5 **Create a test manifest that is prepared identically to the control manifest.**

```
# bart create -r rules-file > test-manifest
```

6 **Compare the control manifest with the test manifest by using the same rules file.**

`# bart compare -r` *rules-file control-manifest test-manifest > bart.report*

7 **Examine the BART report for oddities.**

Example 5–7 Customizing a BART Report by Using a Rules File

The following rules file includes directives for both the `bart create` and the `bart compare` commands. The rules file directs the `bart create` command to list information about the contents of the `/usr/bin` directory. In addition, the rules file directs the `bart compare` command to track only size and content changes in the same directory.

```
# Check size and content changes in the /usr/bin directory.
# This rules file only checks size and content changes.
# See rules file example.

IGNORE all
CHECK size contents
/usr/bin
```

- Create a control manifest by using the rules file you created.

  ```
  # bart create -r bartrules.txt > usr_bin.control-manifest.121003
  ```

- Create a test manifest whenever you want to monitor changes to the `/usr/bin` directory.

  ```
  # bart create -r bartrules.txt > usr_bin.test-manifest.121103
  ```

- Compare the manifests by using the same rules file.

  ```
  # bart compare -r bartrules.txt usr_bin.control-manifest \
  usr_bin.test-manifest
  ```

- Examine the output of the `bart compare` command.

  ```
   /usr/bin/gunzip:  add
  /usr/bin/ypcat:
    delete
  ```

In the preceding output, the `bart compare` command reported a discrepancy in the `/usr/bin` directory. This output indicates that `/usr/bin/ypcat` file was deleted, and the `/usr/bin/gunzip` file was added.

BART Manifest, Rules File, and Reporting (Reference)

This section includes the following reference information:

- "BART Manifest File Format" on page 115
- "BART Rules File Format" on page 116
- "BART Reporting" on page 117

BART Manifest File Format

Each manifest file entry is a single line, depending on the file type. Each entry begins with *fname*, which is the name of the file. To prevent parsing problems that are caused by special characters embedded in file names, the file names are encoded. For more information, see "BART Rules File Format" on page 116.

Subsequent fields represent the following file attributes:

type	Type of file with the following possible values: ■ B for a block device node ■ C for a character device node ■ D for a directory ■ F for a file ■ L for a symbolic link ■ P for a pipe ■ S for a socket
size	File size in bytes.
mode	Octal number that represents the permissions of the file.
acl	ACL attributes for the file. For a file with ACL attributes, this contains the output from `acltotext()`.
uid	Numerical user ID of the owner of this entry.
gid	Numerical group ID of the owner of this entry.
dirmtime	Last modification time, in seconds, since 00:00:00 UTC, January 1, 1970, for directories.
lnmtime	Last modification time, in seconds, since 00:00:00 UTC, January 1, 1970, for links.
mtime	Last modification time, in seconds, since 00:00:00 UTC January 1, 1970, for files.

contents	Checksum value of the file. This attribute is only specified for regular files. If you turn off context checking, or if checksums cannot be computed, the value of this field is –.
dest	Destination of a symbolic link.
devnode	Value of the device node. This attribute is for character device files and block device files only.

For more information about BART manifests, see the `bart_manifest`(4) man page.

BART Rules File Format

The input files to the `bart` command are text files. These files consist of lines that specify which files are to be included in the manifest and which file attributes are to be included the report. The same input file can be used across both pieces of BART functionality. Lines that begin with `#`, blank lines, and lines that contain white space are ignored by the tool.

The input files have three types of directives:

- Subtree directive, with optional pattern matching modifiers
- `CHECK` directive
- `IGNORE` directive

EXAMPLE 5–8 Rules File Format

```
<Global CHECK/IGNORE Directives>
<subtree1> [pattern1..]
<IGNORE/CHECK Directives for subtree1>

<subtree2> [pattern2..]
<subtree3> [pattern3..]
<subtree4> [pattern4..]
<IGNORE/CHECK Directives for subtree2, subtree3, subtree4>
```

Note – All directives are read in order, with later directives possibly overriding earlier directives.

There is one subtree directive per line. The directive *must* begin with an absolute pathname, followed by zero or more pattern matching statements.

Rules File Attributes

The `bart` command uses `CHECK` and `IGNORE` statements to define which attributes to track or ignore. Each attribute has an associated keyword.

The attribute *keywords* are as follows:

- `acl`
- `all`
- `contents`
- `dest`
- `devnode`
- `dirmtime`
- `gid`
- `lnmtime`
- `mode`
- `mtime`
- `size`
- `type`
- `uid`

The `all` keyword refers to all file attributes.

Quoting Syntax

The rules file specification language that BART uses is the standard UNIX quoting syntax for representing nonstandard file names. Embedded tab, space, newline, or special characters are encoded in their octal forms to enable the tool to read file names. This nonuniform quoting syntax prevents certain file names, such as those containing an embedded carriage return, from being processed correctly in a command pipeline. The rules specification language allows the expression of complex file name filtering criteria that would be difficult and inefficient to describe by using shell syntax alone.

For more information about the BART rules file or the quoting syntax used by BART, see the `bart_rules`(4) man page.

BART Reporting

In default mode, the `bart compare` command, as shown in the following example, will check all the files installed on the system, with the exception of modified directory timestamps (`dirmtime`):

```
CHECK all
IGNORE    dirmtime
```

If you supply a rules file, then the global directives of `CHECK all` and `IGNORE dirmtime`, in that order, are automatically prepended to the rules file.

BART Output

The following exit values are returned:

0	Success
1	Nonfatal error when processing files, such as permission problems
>1	Fatal error, such as an invalid command-line option

The reporting mechanism provides two types of output: verbose and programmatic:

- Verbose output is the default output and is localized and presented on multiple lines. Verbose output is internationalized and is human-readable. When the `bart compare` command compares two system manifests, a list of file differences is generated.

  For example:

  filename attribute control:xxxx test:yyyy

filename	Name of the file that differs between the control manifest and the test manifest.
attribute	Name of the file attribute that differs between the manifests that are compared. *xxxx* is the attribute value from the control manifest, and *yyyy* is the attribute value from the test manifest. When discrepancies for multiple attributes occur in the same file, each difference is noted on a separate line.

  Following is an example of the default output for the `bart compare` command. The attribute differences are for the `/etc/passwd` file. The output indicates that the `size`, `mtime`, and `contents` attributes have changed.

  ```
  /etc/passwd:
  size    control:74    test:81
  mtime control:3c165879    test:3c165979
  contents    control:daca28ae0de97afd7a6b91fde8d57afa
  test:84b2b32c4165887355317207b48a6ec7
  ```

- Programmatic output is generated if you use the `-p` option when you run the `bart compare` command. This output is generated in a form that is suitable for programmatic manipulation. Programmatic output can be easily parsed by other programs and is designed to be used as input for other tools.

  For example:

  filename attribute control-val test-val [*attribute control-val test-val*]*

filename	Same as the *filename* attribute in the default format
attribute control-val test-val	A description of the file attributes that differ between the control and test manifests for each file

For a list of attributes that are supported by the `bart` command, see "Rules File Attributes" on page 117.

For more information about BART, see the `bart`(1M) man page.

◆ ◆ ◆ CHAPTER 6

Controlling Access to Files (Tasks)

This chapter describes how to protect files in the Solaris Operating System (Solaris OS). The chapter also describes how to protect against files whose permissions could compromise the system.

The following is a list of the information in this chapter.

- "Using UNIX Permissions to Protect Files" on page 121
- "Using Access Control Lists to Protect Files" on page 127
- "Preventing Executable Files From Compromising Security" on page 130
- "Protecting Files (Task Map)" on page 131
- "Protecting Files With UNIX Permissions (Task Map)" on page 131
- "Protecting Files With ACLs (Task Map)" on page 137
- "Protecting Against Programs With Security Risk (Task Map)" on page 143

Using UNIX Permissions to Protect Files

Files can be secured through UNIX file permissions and through ACLs. Files with sticky bits, and files that are executable, require special security measures.

Commands for Viewing and Securing Files

This table describes the commands for monitoring and securing files and directories.

TABLE 6–1 Commands for Securing Files and Directories

Command	Description	Man Page
`ls`	Lists the files in a directory and information about the files.	`ls(1)`

TABLE 6–1 Commands for Securing Files and Directories *(Continued)*

Command	Description	Man Page
`chown`	Changes the ownership of a file.	`chown`(1)
`chgrp`	Changes the group ownership of a file.	`chgrp`(1)
`chmod`	Changes permissions on a file. You can use either symbolic mode, which uses letters and symbols, or absolute mode, which uses octal numbers, to change permissions on a file.	`chmod`(1)

File and Directory Ownership

Traditional UNIX file permissions can assign ownership to three classes of users:

- **user –** The file or directory owner, which is usually the user who created the file. The owner of a file can decide who has the right to read the file, to write to the file (make changes to it), or, if the file is a command, to execute the file.
- **group –** Members of a group of users.
- **others –** All other users who are not the file owner and are not members of the group.

The owner of the file can usually assign or modify file permissions. Additionally, users or roles with administrative capabilities, such as superuser or the Primary Administrator role, can change a file's ownership. To override system policy, see Example 6–2.

A file can be one of seven types. Each type is displayed by a symbol:

- (Minus symbol)	Text or program
b	Block special file
c	Character special file
d	Directory
l	Symbolic link
s	Socket
D	Door
P	Named pipe (FIFO)

UNIX File Permissions

The following table lists and describes the permissions that you can give to each class of user for a file or directory.

TABLE 6–2 File and Directory Permissions

Symbol	Permission	Object	Description
`r`	Read	File	Designated users can open and read the contents of a file.
		Directory	Designated users can list files in the directory.
`w`	Write	File	Designated users can modify the contents of the file or delete the file.
		Directory	Designated users can add files or add links in the directory. They can also remove files or remove links in the directory.
`x`	Execute	File	Designated users can execute the file, if it is a program or shell script. They also can run the program with one of the `exec(2)` system calls.
		Directory	Designated users can open files or execute files in the directory. They also can make the directory and the directories beneath it current.
`-`	Denied	File and Directory	Designated users cannot read, write, or execute the file.

These file permissions apply to regular files, and to special files such as devices, sockets, and named pipes (FIFOs).

For a symbolic link, the permissions that apply are the permissions of the file that the link points to.

You can protect the files in a directory and its subdirectories by setting restrictive file permissions on that directory. Note, however, that superuser has access to all files and directories on the system.

Special File Permissions (`setuid`, `setgid` and Sticky Bit)

Three special types of permissions are available for executable files and public directories: `setuid`, `setgid`, and sticky bit. When these permissions are set, any user who runs that executable file assumes the ID of the owner (or group) of the executable file.

You must be extremely careful when you set special permissions, because special permissions constitute a security risk. For example, a user can gain superuser capabilities by executing a program that sets the user ID (UID) to `0`, which is the UID of `root`. Also, all users can set special permissions for files that they own, which constitutes another security concern.

You should monitor your system for any unauthorized use of the `setuid` permission and the `setgid` permission to gain superuser capabilities. A suspicious permission grants ownership of an administrative program to a user rather than to `root` or `bin`. To search for and list all files that use this special permission, see "How to Find Files With Special File Permissions" on page 143.

`setuid` Permission

When `setuid` permission is set on an executable file, a process that runs this file is granted access on the basis of the owner of the file. The access is *not* based on the user who is running the executable file. This special permission allows a user to access files and directories that are normally available only to the owner.

For example, the `setuid` permission on the `passwd` command makes it possible for users to change passwords. A `passwd` command with `setuid` permission would resemble the following:

```
-r-sr-sr-x   3 root     sys       28144 Jun 17 12:02 /usr/bin/passwd
```

This special permission presents a security risk. Some determined users can find a way to maintain the permissions that are granted to them by the `setuid` process even after the process has finished executing.

Note – The use of `setuid` permissions with the reserved UIDs (0–100) from a program might not set the effective UID correctly. Use a shell script, or avoid using the reserved UIDs with `setuid` permissions.

`setgid` Permission

The `setgid` permission is similar to the `setuid` permission. The process's effective group ID (GID) is changed to the group that owns the file, and a user is granted access based on the permissions that are granted to that group. The `/usr/bin/mail` command has `setgid` permissions:

```
-r-x--s--x   1 root   mail     67504 Jun 17 12:01 /usr/bin/mail
```

When the `setgid` permission is applied to a directory, files that were created in this directory belong to the group to which the directory belongs. The files do not belong to the group to which the creating process belongs. Any user who has write and execute permissions in the directory can create a file there. However, the file belongs to the group that owns the directory, not to the group that the user belongs to.

You should monitor your system for any unauthorized use of the `setgid` permission to gain superuser capabilities. A suspicious permission grants group access to such a program to an unusual group rather than to `root` or `bin`. To search for and list all files that use this permission, see "How to Find Files With Special File Permissions" on page 143.

Sticky Bit

The *sticky bit* is a permission bit that protects the files within a directory. If the directory has the sticky bit set, a file can be deleted only by the file owner, the directory owner, or by a privileged user. The `root` user and the Primary Administrator role are examples of privileged users. The sticky bit prevents a user from deleting other users' files from public directories such as `/tmp`:

```
drwxrwxrwt 7  root  sys   400 Sep  3 13:37 tmp
```

Be sure to set the sticky bit manually when you set up a public directory on a TMPFS file system. For instructions, see Example 6–5.

Default `umask` Value

When you create a file or directory, you create it with a default set of permissions. The system defaults are open. A text file has `666` permissions, which grants read and write permission to everyone. A directory and an executable file have `777` permissions, which grants read, write, and execute permission to everyone. Typically, users override the system defaults in their `/etc/profile` file, `.cshrc` file, or `.login` file.

The value assigned by the `umask` command is subtracted from the default. This process has the effect of denying permissions in the same way that the `chmod` command grants them. For example, the `chmod 022` command grants write permission to group and others. The `umask 022` command denies write permission to group and others.

The following table shows some typical `umask` settings and their effect on an executable file.

TABLE 6–3 `umask` Settings for Different Security Levels

Level of Security	`umask` Setting	Permissions Disallowed
Permissive (`744`)	`022`	`w` for group and others
Moderate (`740`)	`027`	`w` for group, `rwx` for others
Moderate (`741`)	`026`	`w` for group, `rw` for others
Severe (`700`)	`077`	`rwx` for group and others

For more information on setting the `umask` value, see the `umask`(1) man page.

File Permission Modes

The `chmod` command enables you to change the permissions on a file. You must be superuser or the owner of a file or directory to change its permissions.

You can use the `chmod` command to set permissions in either of two modes:

- **Absolute Mode –** Use numbers to represent file permissions. When you change permissions by using the absolute mode, you represent permissions for each triplet by an octal mode number. Absolute mode is the method most commonly used to set permissions.
- **Symbolic Mode –** Use combinations of letters and symbols to add permissions or remove permissions.

The following table lists the octal values for setting file permissions in absolute mode. You use these numbers in sets of three to set permissions for owner, group, and other, in that order. For example, the value `644` sets read and write permissions for owner, and read-only permissions for group and other.

TABLE 6–4 Setting File Permissions in Absolute Mode

Octal Value	File Permissions Set	Permissions Description
`0`	`---`	No permissions
`1`	`--x`	Execute permission only
`2`	`-w-`	Write permission only
`3`	`-wx`	Write and execute permissions
`4`	`r--`	Read permission only
`5`	`r-x`	Read and execute permissions
`6`	`rw-`	Read and write permissions
`7`	`rwx`	Read, write, and execute permissions

The following table lists the symbols for setting file permissions in symbolic mode. Symbols can specify whose permissions are to be set or changed, the operation to be performed, and the permissions that are being assigned or changed.

TABLE 6–5 Setting File Permissions in Symbolic Mode

Symbol	Function	Description
`u`	*who*	User (owner)
`g`	*who*	Group
`o`	*who*	Others
`a`	*who*	All
`=`	*operator*	Assign

TABLE 6–5 Setting File Permissions in Symbolic Mode *(Continued)*

Symbol	Function	Description
`+`	*operator*	Add
`-`	*operator*	Remove
`r`	*permissions*	Read
`w`	*permissions*	Write
`x`	*permissions*	Execute
`l`	*permissions*	Mandatory locking, `setgid` bit is on, group execution bit is off
`s`	*permissions*	`setuid` or `setgid` bit is on
`t`	*permissions*	Sticky bit is on, execution bit for others is on

The *who operator permissions* designations in the function column specify the symbols that change the permissions on the file or directory.

who	Specifies whose permissions are to be changed.
operator	Specifies the operation to be performed.
permissions	Specifies what permissions are to be changed.

You can set special permissions on a file in absolute mode or symbolic mode. However, you must use symbolic mode to set or remove `setuid` permissions on a directory. In absolute mode, you set special permissions by adding a new octal value to the left of the permission triplet. The following table lists the octal values for setting special permissions on a file.

TABLE 6–6 Setting Special File Permissions in Absolute Mode

Octal Value	Special File Permissions
`1`	Sticky bit
`2`	`setgid`
`4`	`setuid`

Using Access Control Lists to Protect Files

Traditional UNIX file protection provides read, write, and execute permissions for the three user classes: file owner, file group, and other. An access control list (ACL) provides better file security by enabling you to do the following:

- Define file permissions for the file owner, the group, other, specific users and groups
- Define default permissions for each of the preceding categories

For example, if you want everyone in a group to be able to read a file, you can simply grant group read permissions on that file. Now, assume that you want only one person in the group to be able to write to that file. Standard UNIX does not provide that level of file security. However, an ACL provides this level of file security.

ACL entries define an ACL on a file. The entries are set through the `setfacl` command. ACL entries consist of the following fields separated by colons:

entry-type:[*uid*|*gid*]:*perms*

entry-type	Is the type of ACL entry on which to set file permissions. For example, *entry-type* can be `user` (the owner of a file) or `mask` (the ACL mask). For a listing of ACL entries, see Table 6–7 and Table 6–8.
uid	Is the user name or user ID (UID).
gid	Is the group name or group ID (GID).
perms	Represents the permissions that are set on *entry-type*. *perms* can be indicated by the symbolic characters `rwx` or an octal number. These are the same numbers that are used with the `chmod` command.

In the following example, an ACL entry sets read and write permissions for the user `stacey`.

```
user:stacey:rw-
```

Caution – UFS file system attributes such as ACLs are supported in UFS file systems only. Thus, if you restore or copy files with ACL entries into the `/tmp` directory, which is usually mounted as a TMPFS file system, the ACL entries will be lost. Use the `/var/tmp` directory for temporary storage of UFS files.

ACL Entries for Files

The following table lists the valid ACL entries that you might use when setting ACLs on files. The first three ACL entries provide the basic UNIX file protection.

TABLE 6–7 ACL Entries for Files

ACL Entry	Description
`u[ser]::`*perms*	File owner permissions.
`g[roup]::`*perms*	File group permissions.
`o[ther]:`*perms*	Permissions for users other than the file owner or members of the file group.

TABLE 6–7 ACL Entries for Files *(Continued)*

ACL Entry	Description
`m[ask]:`*perms*	The ACL mask. The mask entry indicates the maximum permissions that are allowed for users (other than the owner) and for groups. The mask is a quick way to change permissions on all the users and groups. For example, the `mask:r--` mask entry indicates that users and groups cannot have more than read permissions, even though they might have write and execute permissions.
`u[ser]:`*uid:perms*	Permissions for a specific user. For *uid*, you can specify either a user name or a numeric UID.
`g[roup]:`*gid:perms*	Permissions for a specific group. For *gid*, you can specify either a group name or a numeric GID.

ACL Entries for Directories

In addition to the ACL entries that are described in Table 6–7, you can set default ACL entries on a directory. Files or directories created in a directory that has default ACL entries will have the same ACL entries as the default ACL entries. Table 6–8 lists the default ACL entries for directories.

When you set default ACL entries for specific users and groups on a directory for the first time, you must also set default ACL entries for the file owner, file group, others, and the ACL mask. These entries are required. They are the first four default ACL entries in the following table.

TABLE 6–8 Default ACL Entries for Directories

Default ACL Entry	Description
`d[efault]:u[ser]::`*perms*	Default file owner permissions.
`d[efault]:g[roup]::`*perms*	Default file group permissions.
`d[efault]:o[ther]:`*perms*	Default permissions for users other than the file owner or members of the file group.
`d[efault]:m[ask]:`*perms*	Default ACL mask.
`d[efault]:u[ser]:`*uid:perms*	Default permissions for a specific user. For *uid*, you can specify either a user name or a numeric UID.
`d[efault]:g[roup]:`*gid:perms*	Default permissions for a specific group. For *gid*, you can specify either a group name or a numeric GID.

Commands for Administering ACLs

The following commands administer ACLs on files or directories.

`setfacl` command	Sets, adds, modifies, and deletes ACL entries. For more information, see the `setfacl`(1) man page.
`getfacl` command	Displays ACL entries. For more information, see the `getfacl`(1) man page.

Preventing Executable Files From Compromising Security

A number of security bugs are related to default executable stacks when their permissions are set to read, write, and execute. While stacks with execute permissions are allowed, most programs can function correctly without using executable stacks.

The `noexec_user_stack` variable enables you to specify whether stack mappings are executable. The variable is available as of the Solaris 2.6 release. By default, the variable is set to zero, except on 64-bit applications, which provides ABI-compliant behavior. If the variable is set to a non-zero value, the system marks the stack of every process in the system as readable and writable, but not executable.

Once this variable is set, programs that attempt to execute code on their stack are sent a `SIGSEGV` signal. This signal usually results in the program terminating with a core dump. Such programs also generate a warning message that includes the name of the offending program, the process ID, and the real UID of the user who ran the program. For example:

```
a.out[347] attempt to execute code on stack by uid 555
```

The message is logged by the `syslog` daemon when the `syslog` `kern` facility is set to `notice` level. This logging is set by default in the `syslog.conf` file, which means that the message is sent to both the console and the `/var/adm/messages` file. For more information, see the `syslogd`(1M) and `syslog.conf`(4) man pages.

The `syslog` message is useful for observing potential security problems. The message also identifies valid programs that depend upon executable stacks that have been prevented from correct operation by setting this variable. If you do not want any messages logged, then set the `noexec_user_stack_log` variable to zero in the `/etc/system` file. Even though messages are not being logged, the SIGSEGV signal can continue to cause the executing program to terminate with a core dump.

You can use the `mprotect()` function if you want programs to explicitly mark their stack as executable. For more information, see the `mprotect`(2) man page.

Because of hardware limitations, the capability of catching and reporting executable stack problems is not available on most x86 based systems. Systems in the AMD64 product family can catch and report executable stack problems.

Protecting Files (Task Map)

The following task map points to sets of procedures for protecting files.

Task	Description	For Instructions
Use UNIX permissions to protect files	Views UNIX permissions on files. Protects files with UNIX permissions.	"Protecting Files With UNIX Permissions (Task Map)" on page 131
Use ACLs to protect files	Adds ACLs to protect files at a more granular level than UNIX permissions can.	"Protecting Files With ACLs (Task Map)" on page 137
Protect system from files that pose a security risk	Finds executable files that have suspicious ownership. Disables files that can damage the system.	"Protecting Against Programs With Security Risk (Task Map)" on page 143

Protecting Files With UNIX Permissions (Task Map)

The following task map points to procedures that list file permissions, change file permissions, and protect files with special file permissions.

Task	For Instructions
Display file information	"How to Display File Information" on page 131
Change file ownership	"How to Change the Owner of a File" on page 133 "How to Change Group Ownership of a File" on page 134
Change file permissions	"How to Change File Permissions in Symbolic Mode" on page 134 "How to Change File Permissions in Absolute Mode" on page 135 "How to Change Special File Permissions in Absolute Mode" on page 136

▼ How to Display File Information

Display information about all the files in a directory by using the `ls` command.

- **Type the following command to display a long listing of all files in the current directory.**

```
% ls -la
```

`-l` Displays the long format that includes user ownership, group ownership, and file permissions.

`-a` Displays all files, including hidden files that begin with a dot (`.`).

Example 6–1 Displaying File Information

In the following example, a partial list of the files in the `/sbin` directory is displayed.

```
% cd /sbin
% ls -la
total 13456
drwxr-xr-x   2 root     sys         512 Sep  1 14:11 .
drwxr-xr-x  29 root     root       1024 Sep  1 15:40 ..
-r-xr-xr-x   1 root     bin      218188 Aug 18 15:17 autopush
lrwxrwxrwx   1 root     root         21 Sep  1 14:11 bpgetfile -> ...
-r-xr-xr-x   1 root     bin      505556 Aug 20 13:24 dhcpagent
-r-xr-xr-x   1 root     bin      456064 Aug 20 13:25 dhcpinfo
-r-xr-xr-x   1 root     bin      272360 Aug 18 15:19 fdisk
-r-xr-xr-x   1 root     bin      824728 Aug 20 13:29 hostconfig
-r-xr-xr-x   1 root     bin      603528 Aug 20 13:21 ifconfig
-r-xr-xr-x   1 root     sys      556008 Aug 20 13:21 init
-r-xr-xr-x   2 root     root     274020 Aug 18 15:28 jsh
-r-xr-xr-x   1 root     bin      238736 Aug 21 19:46 mount
-r-xr-xr-x   1 root     sys        7696 Aug 18 15:20 mountall
   .
   .
   .
```

Each line displays information about a file in the following order:

- Type of file – For example, `d`. For list of file types, see "File and Directory Ownership" on page 122.
- Permissions – For example, `r-xr-xr-x`. For description, see "File and Directory Ownership" on page 122.
- Number of hard links – For example, `2`.
- Owner of the file – For example, `root`.
- Group of the file – For example, `bin`.
- Size of the file, in bytes – For example, `7696`.
- Date the file was created or the last date that the file was changed – For example, `Aug 18 15:20`.
- Name of the file – For example, `mountall`.

▼ How to Change the Owner of a File

The file owner, the Primary Administrator role, or superuser can change any file's ownership.

1. **Display the permissions on a file.**

```
% ls -l example-file
-rw-r--r--   1 janedoe   staff   112640 May 24 10:49 example-file
```

2. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

3. **Change the owner of the file.**

```
# chown stacey example-file
```

4. **Verify that the owner of the file has changed.**

```
# ls -l example-file
-rw-r--r--   1 stacey   staff   112640 May 26 08:50 example-file
```

Example 6–2 Enabling Users to Change the Ownership of Files That Others Own

Security Consideration – You should have good reason to override system security policy by changing the setting of the `rstchown` variable to zero. Any user who accesses the system can change the ownership of any file on the system.

In this example, the value of the `rstchown` variable is set to zero in the `/etc/system` file. This setting enables the owner of a file to use the `chown` command to change the file's ownership to another user. This setting also enables the owner to use the `chgrp` command to set the group ownership of a file to a group that the owner does not belong to. The change goes into effect when the system is rebooted.

```
set rstchown = 0
```

For more information, see the `chown`(1) and `chgrp`(1) man pages.

Also, be aware that NFS-mounted file systems have further restrictions on changing ownership and groups. For more information on restricting access to NFS-mounted systems, see Chapter 6, "Accessing Network File Systems (Reference)," in *System Administration Guide: Network Services*.

▼ How to Change Group Ownership of a File

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Change the group ownership of a file.**

   ```
   $ chgrp scifi example-file
   ```

   For information on setting up groups, see Chapter 4, "Managing User Accounts and Groups (Overview)," in *System Administration Guide: Basic Administration*.

3. **Verify that the group ownership of the file has changed.**

   ```
   $ ls -l example-file
    -rw-r--r--   1 stacey   scifi   112640 June 20 08:55  example-file
   ```

   Also see Example 6–2.

▼ How to Change File Permissions in Symbolic Mode

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**

   Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. **Change permissions in symbolic mode.**

   ```
   % chmod who operator permissions filename
   ```

who	Specifies whose permissions are to be changed.
operator	Specifies the operation to be performed.
permissions	Specifies what permissions are to be changed. For the list of valid symbols, see Table 6–5.
filename	Specifies the file or directory.

3. **Verify that the permissions of the file have changed.**

   ```
   % ls -l filename
   ```

Example 6–3 Changing Permissions in Symbolic Mode

In the following example, read permission is taken away from others.

```
% chmod o-r example-file1
```

In the following example, read and execute permissions are added for user, group, and others.

```
$ chmod a+rx example-file2
```

In the following example, read, write, and execute permissions are assigned to group.

```
$ chmod g=rwx example-file3
```

▼ How to Change File Permissions in Absolute Mode

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**

   Only the current owner or superuser can use the `chmod` command to change file permissions on a file or directory.

2. **Change permissions in absolute mode.**

   ```
   % chmod nnn filename
   ```

nnn	Specifies the octal values that represent the permissions for the file owner, file group, and others, in that order. For the list of valid octal values, see Table 6–4.
filename	Specifies the file or directory.

   Note – When you use the `chmod` command to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the permissions for other users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl`(1) man page.

   ---

3. **Verify that the permissions of the file have changed.**

   ```
   % ls -l filename
   ```

Example 6–4 Changing Permissions in Absolute Mode

In the following example, the permissions of a public directory are changed from `744` (read, write, execute; read-only; and read-only) to `755` (read, write, execute; read and execute; and read and execute).

```
# ls -ld public_dir
drwxr--r--  1 ignatz   staff    6023 Aug  5 12:06 public_dir
# chmod 755 public_dir
# ls -ld public_dir
drwxr-xr-x  1 ignatz   staff    6023 Aug  5 12:06 public_dir
```

In the following example, the permissions of an executable shell script are changed from read and write to read, write, and execute.

```
% ls -l my_script
-rw------- 1 ignatz   staff    6023 Aug  5 12:06 my_script
% chmod 700 my_script
% ls -l my_script
-rwx------ 1 ignatz   staff    6023 Aug  5 12:06 my_script
```

▼ How to Change Special File Permissions in Absolute Mode

1. **If you are not the owner of the file or directory, become superuser or assume an equivalent role.**

   Only the current owner or a user with superuser capabilities can use the `chmod` command to change the special permissions on a file or directory.

2. **Change special permissions in absolute mode.**

   `% chmod` *nnnn filename*

nnnn	Specifies the octal values that change the permissions on the file or directory. The leftmost octal value sets the special permissions on the file. For the list of valid octal values for special permissions, see Table 6–6.
filename	Specifies the file or directory.

Note – When you use the `chmod` command to change the file group permissions on a file with ACL entries, both the file group permissions and the ACL mask are changed to the new permissions. Be aware that the new ACL mask permissions can change the permissions for additional users and groups who have ACL entries on the file. Use the `getfacl` command to make sure that the appropriate permissions are set for all ACL entries. For more information, see the `getfacl`(1) man page.

---

3. **Verify that the permissions of the file have changed.**

   `% ls -l` *filename*

Example 6–5 Setting Special File Permissions in Absolute Mode

In the following example, the `setuid` permission is set on the `dbprog` file.

```
# chmod 4555 dbprog
# ls -l dbprog
-r-sr-xr-x   1 db     staff        12095 May  6 09:29 dbprog
```

In the following example, the `setgid` permission is set on the `dbprog2` file.

```
# chmod 2551 dbprog2
# ls -l dbprog2
-r-xr-s--x   1 db     staff       24576 May  6 09:30 dbprog2
```

In the following example, the sticky bit permission is set on the `public_dir` directory.

```
# chmod 1777 public_dir
# ls -ld public_dir
drwxrwxrwt   2 ignatz   staff          512 May 15 15:27 public_dir
```

Protecting Files With ACLs (Task Map)

The following task map points to procedures that list the ACLs on a file, change the ACLs, and copy the ACLs to another file.

Task	For Instructions
Determine if a file has an ACL	"How to Check if a File Has an ACL" on page 138
Add an ACL to a file	"How to Add ACL Entries to a File" on page 138
Copy an ACL	"How to Copy an ACL" on page 140

Task	For Instructions
Modify an ACL	"How to Change ACL Entries on a File" on page 140
Remove ACLs from a file	"How to Delete ACL Entries From a File" on page 141
Display the ACLs on a file	"How to Display ACL Entries for a File" on page 142

▼ How to Check if a File Has an ACL

- **Check if a file has an ACL.**

  `% ls -l` *filename*

  where *filename* specifies the file or directory.

  In the output, a plus sign (+) to the right of the mode field indicates that the file has an ACL.

Note – Unless you have added ACL entries that extend UNIX file permissions, a file is considered to have a "trivial" ACL and the plus sign (+) does not display.

Example 6–6 Checking if a File Has an ACL

In the following example, the `ch1.sgm` file has an ACL. The ACL is indicated by the plus sign (+) to the right of the mode field.

```
% ls -l ch1.sgm
-rwxr-----+  1 stacey   techpubs      167 Nov 11 11:13 ch1.sgm
```

▼ How to Add ACL Entries to a File

1. **Set an ACL on a file by using the `setfacl` command.**

   `% setfacl -s user::`*perms*`,group::`*perms*`,other:`*perms*`,mask:`*perms*`,`*acl-entry-list filename ...*

`-s`	Sets an ACL on the file. If a file already has an ACL, it is replaced. This option requires at least the `user::`, `group::`, and `other::` entries.
`user::`*perms*	Specifies the file owner permissions.
`group::`*perms*	Specifies the group ownership permissions.
`other:`*perms*	Specifies the permissions for users other than the file owner or members of the group.
`mask:`*perms*	Specifies the permissions for the ACL mask. The mask indicates the maximum permissions that are allowed for users (other than the owner) and for groups.

acl-entry-list	Specifies the list of one or more ACL entries to set for specific users and groups on the file or directory. You can also set default ACL entries on a directory. Table 6–7 and Table 6–8 show the valid ACL entries.
filename ...	Specifies one or more files or directories on which to set the ACL. Multiple *filenames* are separated by spaces.

Caution – If an ACL already exists on the file, the `-s` option replaces the entire ACL with the new ACL.

For more information, see the `setfacl`(1) man page.

2 **Verify that the ACL entries were set on the file.**

```
% getfacl filename
```

For more information, see "How to Check if a File Has an ACL" on page 138.

Example 6–7

Setting an ACL on a File

In the following example, the file owner permissions are set to read and write, file group permissions are set to read only, and other permissions are set to none on the `ch1.sgm` file. In addition, the user `anusha` is given read and write permissions on the file. The ACL mask permissions are set to read and write, which means that no user or group can have execute permissions.

```
% setfacl -s user::rw-,group::r--,other:---,mask:rw-,user:anusha:rw- ch1.sgm
% ls -l
total 124
-rw-r-----+  1 stacey  techpubs   34816 Nov 11 14:16 ch1.sgm
-rw-r--r--   1 stacey  techpubs   20167 Nov 11 14:16 ch2.sgm
-rw-r--r--   1 stacey  techpubs    8192 Nov 11 14:16 notes
% getfacl ch1.sgm
# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:rw-     #effective:rw-
group::r--          #effective:r--
mask:rw-
other:---
```

In the following example, the file owner permissions are set to read, write, and execute, file group permissions are set to read only, other permissions are set to none. In addition, the ACL mask permissions are set to read on the `ch2.sgm` file. Finally, the user `anusha` is given read and write permissions. However, due to the ACL mask, the permissions for `anusha` are read only.

```
% setfacl -s u::7,g::4,o:0,m:4,u:anusha:7 ch2.sgm
% getfacl ch2.sgm
# file: ch2.sgm
# owner: stacey
# group: techpubs
user::rwx
user:anusha:rwx         #effective:r--
group::r--              #effective:r--
mask:r--
other:---
```

How to Copy an ACL

- **Copy a file's ACL to another file by redirecting the `getfacl` output.**

  `% getfacl` *filename1* `| setfacl -f -` *filename2*

filename1	Specifies the file from which to copy the ACL.
filename2	Specifies the file on which to set the copied ACL.

Example 6–8 Copying an ACL

In the following example, the ACL on `ch2.sgm` is copied to `ch3.sgm`.

```
% getfacl ch2.sgm | setfacl -f - ch3.sgm
```

How to Change ACL Entries on a File

1. **Modify ACL entries on a file by using the `setfacl` command.**

   `% setfacl -m` *acl-entry-list filename ...*

`-m`	Modifies the existing ACL entry.
acl-entry-list	Specifies the list of one or more ACL entries to modify on the file or directory. You can also modify default ACL entries on a directory. Table 6–7 and Table 6–8 show the valid ACL entries.
filename ...	Specifies one or more files or directories, separated by a space.

2. **Verify that the ACL entries were modified on the file.**

   `% getfacl` *filename*

Example 6–9 Modifying ACL Entries on a File

In the following example, the permissions for the user `anusha` are modified to read and write.

```
% setfacl -m user:anusha:6 ch3.sgm
% getfacl ch3.sgm
# file: ch3.sgm
# owner: stacey
# group: techpubs
user::rw-
user::anusha:rw-        #effective:r--
group::r-               #effective:r--
mask:r--
other:r-
```

In the following example, the default permissions for the group `staff` are modified to read on the `book` directory. In addition, the default ACL mask permissions are modified to read and write.

```
% setfacl -m default:group:staff:4,default:mask:6 book
```

▼ How to Delete ACL Entries From a File

1. **Delete ACL entries from a file.**

   `% setfacl -d` *acl-entry-list filename ...*

`-d`	Deletes the specified ACL entries.
acl-entry-list	Specifies the list of ACL entries (without specifying the permissions) to delete from the file or directory. You can only delete ACL entries and default ACL entries for specific users and groups. Table 6–7 and Table 6–8 show the valid ACL entries.
filename ...	Specifies one or more files or directories, separated by a space.

   Alternatively, you can use the `setfacl -s` command to delete all the ACL entries on a file and replace them with the new ACL entries that are specified.

2. **Verify that the ACL entries were deleted from the file.**

   `% getfacl` *filename*

Example 6–10 Deleting ACL Entries on a File

In the following example, the user `anusha` is deleted from the `ch4.sgm` file.

```
% setfacl -d user:anusha ch4.sgm
```

▼ How to Display ACL Entries for a File

- **Display ACL entries for a file by using the `getfacl` command.**

  `% getfacl [-a | -d]` *filename ...*

`-a`	Displays the file name, file owner, file group, and ACL entries for the specified file or directory.
`-d`	Displays the file name, file owner, file group, and the default ACL entries, if they exist, for the specified directory.
filename ...	Specifies one or more files or directories, separated by a space.

  If you specify multiple file names on the command line, the ACL entries are displayed with a blank line between each entry.

Example 6–11 Displaying ACL Entries for a File

In the following example, all the ACL entries for the `ch1.sgm` file are displayed. The `#effective:` note beside the user and group entries indicates what the permissions are after being modified by the ACL mask.

```
% getfacl ch1.sgm

# file: ch1.sgm
# owner: stacey
# group: techpubs
user::rw-
user:anusha:r-        #effective:r--
group::rw-            #effective:rw-
mask:rw-
other:---
```

In the following example, the default ACL entries for the `book` directory are displayed.

```
% getfacl -d book

# file: book
# owner: stacey
# group: techpubs
user::rwx
user:anusha:r-x       #effective:r-x
group::rwx            #effective:rwx
mask:rwx
other:---
default:user::rw-
default:user:anusha:r--
default:group::rw-
default:mask:rw-
default:other:---
```

Protecting Against Programs With Security Risk (Task Map)

The following task map points to procedures that find risky executables on the system, and that prevent programs from exploiting an executable stack.

Task	Description	For Instructions
Find files with special permissions	Locates files with the `setuid` bit set, but that are not owned by the `root` user.	"How to Find Files With Special File Permissions" on page 143
Prevent executable stack from overflowing	Prevents programs from exploiting an executable stack.	"How to Disable Programs From Using Executable Stacks" on page 144
Prevent logging of executable stack messages	Turns off logging of executable stack messages.	Example 6–13

▼ How to Find Files With Special File Permissions

You should monitor your system for any unauthorized use of the `setuid` and `setgid` permissions on programs. The `setuid` and `setgid` permissions enable ordinary users to gain superuser capabilities. A suspicious executable file grants ownership to a user rather than to `root` or `bin`.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Find files with `setuid` permissions by using the `find` command.**

   ```
   # find directory -user root -perm -4000 -exec ls -ldb {} \; >/tmp/filename
   ```

`find` *directory*	Checks all mounted paths starting at the specified *directory*, which can be root (`/`), `sys`, `bin`, or `mail`.
`-user root`	Displays files owned only by `root`.
`-perm -4000`	Displays files only with permissions set to `4000`.
`-exec ls -ldb`	Displays the output of the `find` command in `ls -ldb` format.
`>/tmp/`*filename*	Is the file that contains the results of the `find` command.

3. **Display the results in `/tmp/`*filename*.**

   ```
   # more /tmp/filename
   ```

   For background information on `setuid` permissions, see "`setuid` Permission" on page 124.

Example 6–12 Finding Files With `setuid` Permissions

The output from the following example shows that a user named `rar` has made a personal copy of `/usr/bin/sh`, and has set the permissions as `setuid` to `root`. As a result, the `/usr/rar/bin/sh` program runs with `root` permissions.

This output was saved for future reference by moving the file out of the `/tmp` directory.

```
# find / -user root -perm -4000 -exec ls -ldb {} \; > /var/tmp/ckprm
# cat /var/tmp/ckprm
-r-sr-xr-x 1 root bin 38836 Aug 10 16:16 /usr/bin/at
-r-sr-xr-x 1 root bin 19812 Aug 10 16:16 /usr/bin/crontab
---s--x--x 1 root sys 46040 Aug 10 15:18 /usr/bin/ct
-r-sr-xr-x 1 root sys 12092 Aug 11 01:29 /usr/lib/mv_dir
-r-sr-sr-x 1 root bin 33208 Aug 10 15:55 /usr/lib/lpadmin
-r-sr-sr-x 1 root bin 38696 Aug 10 15:55 /usr/lib/lpsched
---s--x--- 1 root rar 45376 Aug 18 15:11 /usr/rar/bin/sh
-r-sr-xr-x 1 root bin 12524 Aug 11 01:27 /usr/bin/df
-rwsr-xr-x 1 root sys 21780 Aug 11 01:27 /usr/bin/newgrp
-r-sr-sr-x 1 root sys 23000 Aug 11 01:27 /usr/bin/passwd
-r-sr-xr-x 1 root sys 23824 Aug 11 01:27 /usr/bin/su
# mv /var/tmp/ckprm /export/sysreports/ckprm
```

▼ How to Disable Programs From Using Executable Stacks

For a description of the security risks of executable stacks, see "Preventing Executable Files From Compromising Security" on page 130.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Edit the `/etc/system` file, and add the following line:**

```
set noexec_user_stack=1
```

3 **Reboot the system.**

```
# init 6
```

Example 6–13 Disabling the Logging of Executable Stack Messages

In this example, the logging of executable stack messages is disabled, and then the system is rebooted.

```
# cat /etc/system
set noexec_user_stack=1
set noexec_user_stack_log=0
# init 6
```

◆ ◆ ◆

CHAPTER 7

Using the Automated Security Enhancement Tool (Tasks)

This chapter describes how to use the Automated Security Enhancement Tool (ASET) to monitor or restrict access to system files and directories.

The following is a list of the step-by-step instructions in this chapter.

- "Automated Security Enhancement Tool (ASET)" on page 147
- "Running ASET (Task Map)" on page 164
- "Troubleshooting ASET Problems" on page 168

For a more comprehensive tool than ASET, use the Solaris Security Toolkit. The Solaris Security Toolkit provides a framework for hardening and minimizing a Solaris system. The kit includes a profiling tool, a reporting tool, and an undo capability. The toolkit is free, and can be downloaded from the Sun web site, `http://wwws.sun.com/security/jass`. The web site contains pointers to online documentation.

The toolkit is described in detail in *Securing Systems with the Solaris Security Toolkit*, by Alex Noordergraaf and Glenn Brunette, ISBN 0-13-141071-7, June 2003. The book is part of the Sun BluePrints Series, which is published by Sun Microsystems Press.

Automated Security Enhancement Tool (ASET)

The Solaris Operating System includes the Automated Security Enhancement Tool (ASET). ASET helps you to monitor and to control system security by automatically performing tasks that you would otherwise do manually.

The ASET security package provides automated administration tools that enable you to control and monitor your system's security. You specify a security level at which to run ASET. The security levels are low, medium, and high. At each higher level, ASET's file-control functions increase to reduce file access and tighten your system security.

There are seven tasks that ASET runs. Each task performs specific checks and adjustments to system files. The ASET tasks tighten file permissions, check the contents of critical system files for security weaknesses, and monitor crucial areas. ASET can also safeguard a network by applying the basic requirements of a firewall system to a system that serves as a gateway system. See "Firewall Setup" on page 151.

ASET uses master files for configuration. Master files, reports, and other ASET files are in the `/usr/aset` directory. These files can be changed to suit the particular requirements of your site.

Each task generates a report. The report notes detected security weaknesses and any changes that the task has made to the system files. When run at the highest security level, ASET attempts to modify all system security weaknesses. If ASET cannot correct a potential security problem, ASET reports the existence of the problem.

You can initiate an ASET session by using the `/usr/aset/aset` command interactively. Or, you can set up ASET to run periodically by putting an entry into the `crontab` file.

ASET tasks are disk-intensive. The tasks can interfere with regular activities. To minimize the impact on system performance, schedule ASET to run when system activity level is lowest. For example, run ASET once every 24 or 48 hours at midnight.

ASET Security Levels

ASET can be set to operate at one of three security levels: low, medium, or high. At each higher level, ASET's file-control functions increase to reduce file access and heighten system security. These functions range from monitoring system security without limiting users' file access, to increasingly tightening access permissions until the system is fully secured.

The following table outlines these three levels of security.

Security Level	Description
Low	Ensures that attributes of system files are set to standard release values. ASET performs several checks, then reports potential security weaknesses. At this level, ASET takes no action, so ASET does not affect system services.
Medium	Provides adequate security control for most environments. ASET modifies some settings of system files and parameters. ASET restricts system access to reduce the risks from security attacks. ASET reports security weaknesses and any modifications that ASET has made to restrict access. At this level, ASET does not affect system services.
High	Renders a highly secure system. ASET adjusts many system files and parameter settings to minimum access permissions. Most system applications and commands continue to function normally. However, at this level, security considerations take precedence over other system behavior.

Note – ASET does not change the permissions of a file to make the file less secure, unless you downgrade the security level. You could also intentionally revert the system to the settings that existed prior to running ASET.

ASET Task List

This section discusses what ASET does. You should understand each ASET task. By understanding the objectives of ASET, the operations that ASET performs, and the system components that ASET affects, you can interpret and use the reports effectively.

ASET report files contain messages that describe as specifically as possible any problems that were discovered by each ASET task. These messages can help you diagnose and correct these problems. However, successful use of ASET assumes that you possess a general understanding of system administration and system components. If you are a novice administrator, you can refer to other Solaris system administration documentation. You can read related manual pages to prepare yourself for ASET administration.

The `taskstat` utility identifies the tasks that have been completed. The utility also identifies the tasks that are still running. Each completed task produces a report file. For a complete description of the `taskstat` utility, refer to `taskstat`(1M).

System Files Permissions Tuning

This task sets the permissions on system files to the security level that you designate. This task is run when the system is installed. If you decide later to alter the previously established levels, then run this task again. At low security, permissions are set to values that are appropriate for an open information-sharing environment. At medium security, permissions are tightened to produce adequate security for most environments. At high security, permissions are tightened to severely restrict access.

Any modifications that this task makes to system files permissions or parameter settings are reported in the `tune.rpt` file. For an example of the files that ASET consults when ASET sets permissions, see “Tune File Examples” on page 163.

System Files Checks

This task examines system files and compares each file with a description of that file in a master file. The master file is created the first time ASET runs this task. The master file contains the system file settings that are enforced by `checklist` for the specified security level.

A list of directories whose files are to be checked is defined for each security level. You can use the default list, or you can modify the list, specifying different directories for each level.

For each file, the following criteria are checked:

- Owner and group
- Permission bits
- Size and checksum
- Number of links
- Last modification time

Any discrepancies that ASET finds are reported in the `cklist.rpt` file. This file contains the results of comparing system file size, permission, and checksum values to the master file.

User and Group Checks

This task checks the consistency and integrity of user accounts and groups. The task uses the definitions in the `passwd` and `group` files. This task checks the local, and NIS or NIS+ password files. Password file problems for NIS+ are reported but not corrected. This task checks for the following violations:

- Duplicate names or IDs
- Entries in incorrect format
- Accounts without a password
- Invalid login directories
- The `nobody` account
- Null group password
- A plus sign (+) in the `/etc/passwd` file on an NIS server or an NIS+ server

Discrepancies are reported in the `usrgrp.rpt` file.

System Configuration Files Check

During this task, ASET checks various system tables, most of which are in the `/etc` directory. These files are the following:

- `/etc/default/login`
- `/etc/hosts.equiv`
- `/etc/inetd.conf`
- `/etc/aliases`
- `/var/adm/utmpx`
- `/.rhosts`
- `/etc/vfstab`
- `/etc/dfs/dfstab`
- `/etc/ftpd/ftpusers`

ASET performs various checks and various modifications on these files. ASET reports problems in the `sysconf.rpt` file.

Environment Variables Check

This task checks how the `PATH` and `UMASK` environment variables are set for root, and for other users. The task checks the `/.profile`, `/.login`, and `/.cshrc` files.

The results of checking the environment for security are reported in the `env.rpt` file.

`eeprom` Check

This task checks the value of the `eeprom` security parameter to ensure that the parameter is set to the appropriate security level. You can set the `eeprom` security parameter to `none`, `command`, or `full`.

ASET does not change this setting, but reports its recommendations in the `eeprom.rpt` file.

Firewall Setup

This task ensures that the system can be safely used as a network relay. This task protects an internal network from external public networks by setting up a dedicated system as a firewall, which is described in "Firewall Systems" on page 55. The firewall system separates two networks. In this situation, each network approaches the other network as untrusted. The firewall setup task disables the forwarding of Internet Protocol (IP) packets. The firewall also hides routing information from the external network.

The firewall task runs at all security levels, but takes action only at the highest level. If you want to run ASET at high security, but find that your system does not require firewall protection, you can eliminate the firewall task. You eliminate the task by editing the `asetenv` file.

Any changes that are made are reported in the `firewall.rpt` file.

ASET Execution Log

ASET generates an execution log whether ASET runs interactively or in the background. By default, ASET generates the log file on standard output. The execution log confirms that ASET ran at the designated time, and also contains any execution error messages. The `aset -n` command directs the log to be delivered by electronic mail to a designated user. For a complete list of ASET options, see the `aset`(1M) man page.

Example of an ASET Execution Log File

```
ASET running at security level low

Machine=example; Current time = 0325_08:00


aset: Using /usr/aset as working directory
```

```
Executing task list...
        firewall
        env
        sysconfig
        usrgrp
        tune
        cklist
        eeprom
All tasks executed. Some background tasks may still be running.

Run /usr/aset/util/taskstat to check their status:
     $/usr/aset/util/taskstat     aset_dir
Where aset_dir is ASET’s operating directory, currently=/usr/aset

When the tasks complete, the reports can be found in:
     /usr/aset/reports/latest/*.rpt
You can view them by:
more /usr/aset/reports/latest/*.rpt
```

The execution log first shows the system and time that ASET was run. Then, the execution log lists each task as the task was started.

ASET invokes a background process for each of these tasks, which are described in “ASET Task List” on page 149. The task is listed in the execution log when the task starts. This listing does not indicate that the task completed. To check the status of the background tasks, use the `taskstat` command.

ASET Reports

All report files that are generated from ASET tasks are stored in subdirectories under the `/usr/aset/reports` directory. This section describes the structure of the `/usr/aset/reports` directory, and provides guidelines on managing the report files.

ASET places the report files in subdirectories that are named to reflect the time and date when the reports are generated. This convention enables you to keep an orderly trail of records that document the system status as the status varies between ASET executions. You can monitor and compare these reports to determine the soundness of your system's security.

The following figure shows an example of the `reports` directory structure.

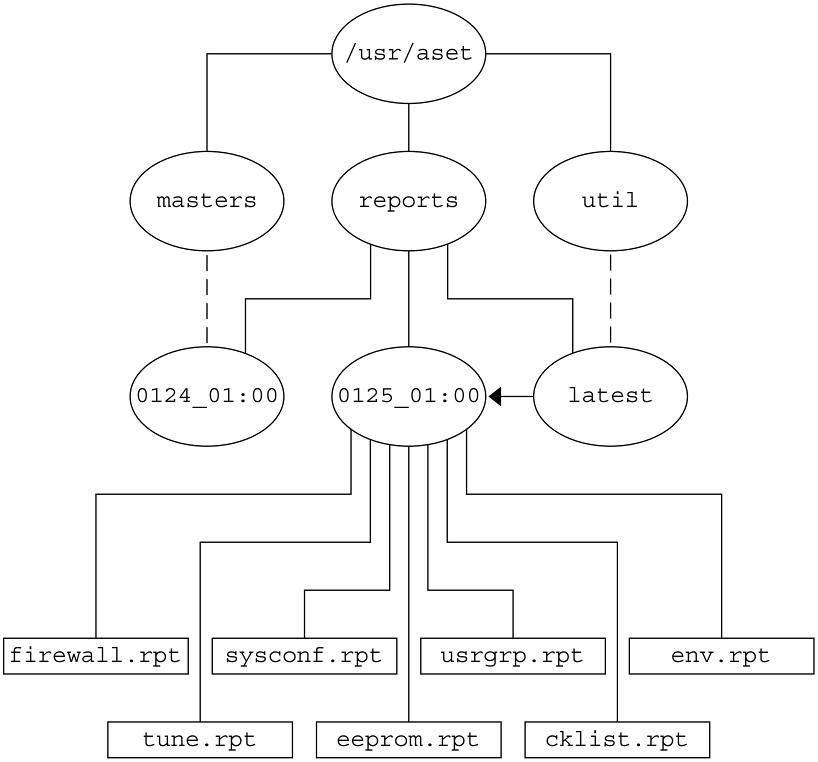

/usr/aset
masters
reports
util
0124_01:00
0125_01:00
latest
firewall.rpt
sysconf.rpt
usrgrp.rpt
env.rpt
tune.rpt
eeprom.rpt
cklist.rpt

FIGURE 7–1 Structure of the ASET `reports` Directory

This example shows two report subdirectories.

- `0124_01:00`
- `0125_01:00`

The subdirectory names indicate the date and time that the reports were generated. Each report subdirectory name has the following format:

monthdate_hour:minute

month, *date*, *hour,* and *minute* are all two-digit numbers. For example, `0125_01:00` represents January 25, at 1 a.m.

Each of the two report subdirectories contains a collection of reports that are generated from one execution of ASET.

The `latest` directory is a symbolic link that always points to the subdirectory that contains the latest reports. Therefore, to look at the latest reports that ASET has generated, you can go to the `/usr/aset/reports/latest` directory. There is a report file in this directory for each task that ASET performed during its most recent execution.

Format of ASET Report Files

Each report file is named after the task that generates the report. The following table lists tasks and their reports.

TABLE 7–1 ASET Tasks and Resulting Reports

Tasks	Report
System files permissions tuning (`tune`)	`tune.rpt`
System files checks (`cklist`)	`cklist.rpt`
User and group checks (`usrgrp`)	`usrgrp.rpt`
System configuration files check (`sysconf`)	`sysconf.rpt`
Environment variables check (`env`)	`env.rpt`
`eeprom` check (`eeprom`)	`eeprom.rpt`
Firewall setup (`firewall`)	`firewall.rpt`

Within each report file, messages are bracketed by a beginning and an ending banner line. Sometimes, a task ends prematurely. For example, a task can end prematurely when a component of ASET is accidentally removed or damaged. In such cases, the report file usually contains a message near the end that indicates the reason for the premature termination.

The following is a sample report file, `usrgrp.rpt`.

```
*** Begin User and Group Checking ***

Checking /etc/passwd ...
Warning! Password file, line 10, no passwd
:sync::1:1::/:/bin/sync
..end user check; starting group check ...
Checking /etc/group...
*** End User And group Checking ***
```

Examining ASET Report Files

After you initially run or reconfigure ASET, you should examine the report files closely. Reconfiguration includes modifying the `asetenv` file or the master files in the `masters` subdirectory, or changing the security level at which ASET operates.

The reports record any errors that were introduced when you reconfigured ASET. By watching the reports closely, you can react to, and solve, problems as the problems arise.

Comparing ASET Report Files

After you monitor the report files for a period during which there are no configuration changes or system updates, you might find that the content of the reports begins to stabilize. When the reports contain little unexpected information, you can use the `diff` utility to compare reports.

ASET Master Files

ASET's master files, `tune.high`, `tune.low`, `tune.med`, and `uid_aliases`, are located in the `/usr/aset/masters` directory. ASET uses the master files to define security levels. For more detail, see the `asetmasters`(4) man page.

Tune Files

The `tune.low`, `tune.med`, and `tune.high` master files define the available ASET security levels. The files specify the attributes of system files at each level and are used for comparison and reference purposes.

The `uid_aliases` File

The `uid_aliases` file contains a list of multiple user accounts that share the same user ID (UID). Normally, ASET warns about such multiple user accounts because this practice lessens accountability. You can allow for exceptions to this rule by listing the exceptions in the `uid_aliases` file. ASET does not report entries in the `passwd` file with duplicate UIDs if these entries are specified in the `uid_aliases` file.

Avoid having multiple user accounts share the same UID. You should consider other methods of achieving your objective. For example, if you intend for several users to share a set of permissions, you could create a group account. You could also create a role. The sharing of UIDs should be your last resort, used only when other methods cannot accomplish your objectives.

You can use the `UID_ALIASES` environment variable to specify an alternate aliases file. The default file is `/usr/aset/masters/uid_aliases`.

The Checklist Files

The master files that are used by the system files checks are generated when you first execute ASET. The master files are also generated when you run ASET after changing the security level.

The following environment variables define the files that are checked by this task:

- `CKLISTPATH_LOW`
- `CKLISTPATH_MED`
- `CKLISTPATH_HIGH`

ASET Environment File (`asetenv`)

The environment file, `asetenv`, contains a list of environment variables that affect ASET tasks. Some of these variables can be changed to modify ASET operation. For details about the `asetenv` file, see `asetenv`(4).

Configuring ASET

This section discusses how ASET is configured. This section also discusses the environment in which ASET operates.

ASET requires minimum administration and minimum configuration. In most cases, you can run ASET with the default values. You can, however, fine-tune some of the parameters that affect the operation and behavior of ASET to maximize its benefit. Before you change the default values, you should understand how ASET works, and how ASET affects the components of your system.

ASET relies on four configuration files to control the behavior of its tasks:

- `/usr/aset/asetenv`
- `/usr/aset/masters/tune.low`
- `/usr/aset/masters/tune.med`
- `/usr/aset/masters/tune.high`

Modifying the Environment File (`asetenv`)

The `/usr/aset/asetenv` file has two main sections:

- A user-configurable environment variables section
- An internal environment variables section

You can alter the user-configurable parameters section. However, the settings in the internal environment variables section are for internal use only. These settings should not be modified.

You can edit the entries in the user-configurable section to do the following:

- Choose which tasks to run
- Specify the directories for the system files checks task
- Schedule ASET execution
- Specify a UID aliases file
- Extend checks to NIS+ tables

Choosing Which Tasks to Run: `TASKS`

Each task that ASET performs monitors a particular area of system security. In most system environments, all the tasks are necessary to provide balanced security coverage. However, you might decide to eliminate one or more tasks.

For example, the firewall task runs at all security levels, but takes action only at the high security level. You might want to run ASET at the high security level, but you do not require firewall protection.

You can set up ASET to run at the high security level without the firewall feature. To do so, edit the `TASKS` list of environment variables in the `asetenv` file. By default, the `TASKS` list contains all of the ASET tasks. To delete a task, remove the task-related environment variable from the file. In this case, you would delete the `firewall` environment variable from the list. The next time ASET runs, the excluded task is not performed.

In the following example, the `TASKS` list with all of the ASET tasks is displayed.

```
TASKS=”env sysconfig usrgrp tune cklist eeprom firewall”
```

Specifying Directories for System Files Checks Task: `CKLISTPATH`

The system files check checks the attributes of files in selected system directories. You define which directories to check by using the following environment variables.

The `CKLISTPATH_LOW` variable defines the directories to be checked at the low security level. `CKLISTPATH_MED` and `CKLISTPATH_HIGH` environment variables function similarly for the medium and high security levels.

The directory list that is defined by an environment variable at a lower security level should be a subset of the directory list that is defined at the next higher level. For example, all directories that are specified for `CKLISTPATH_LOW` should be included in `CKLISTPATH_MED`. Similarly, all the directories that are specified for `CKLISTPATH_MED` should be included in `CKLISTPATH_HIGH`.

Checks that are performed on these directories are not recursive. ASET only checks those directories that are explicitly listed in the environment variable. ASET does not check their subdirectories.

You can edit these environment variable definitions to add or delete directories that you want ASET to check. Note that these checklists are useful only for system files that do not normally change from day to day. A user's home directory, for example, is generally too dynamic to be a candidate for a checklist.

Scheduling ASET Execution: `PERIODIC_SCHEDULE`

You can start ASET interactively, or you can use the `-p` option to request that the ASET tasks run at a scheduled time. You can run ASET periodically, at a time when system demand is light. For example, ASET consults `PERIODIC_SCHEDULE` to determine how frequently to execute the ASET tasks, and at what time to run the tasks. For detailed instructions about setting up ASET to run periodically, see “How to Run ASET Periodically” on page 166.

The format of `PERIODIC_SCHEDULE` follows the format of `crontab` entries. For complete information, see `crontab`(1).

Specifying an Aliases File: `UID_ALIASES`

The `UID_ALIASES` variable specifies an aliases file that lists shared UIDs. The default file is `/usr/aset/masters/uid_aliases`.

Extending Checks to NIS+ Tables: `YPCHECK`

The `YPCHECK` environment variable specifies whether ASET should also check system configuration file tables. `YPCHECK` is a Boolean variable. You can specify only true or false for `YPCHECK`. The default value is false, which disables NIS+ table checking.

To understand how this environment variable works, consider its effect on the `passwd` file. When set to false, ASET checks the local `passwd` file. When set to true, the task also checks the NIS+ `passwd` table for the domain of the system.

Note – Although ASET automatically repairs the local files, ASET only reports potential problems in the NIS+ tables. ASET does not change the tables.

Modifying the Tune Files

ASET uses the three master tune files, `tune.low`, `tune.med`, and `tune.high`, to ease or tighten access to critical system files. These master files are located in the `/usr/aset/masters` directory. You can modify the files to suit your environment. For examples, see "Tune File Examples" on page 163.

The `tune.low` file sets permissions to values that are appropriate for default system settings. The `tune.med` file further restricts these permissions. The `tune.med` file also includes entries that are not present in `tune.low`. The `tune.high` file restricts permissions even further.

Note – Modify settings in the tune files by adding or deleting file entries. You cannot effectively set a permission to a less restrictive value than the current setting. The ASET tasks do not relax permissions unless you downgrade your system security to a lower level.

Restoring System Files Modified by ASET

When ASET is executed for the first time, ASET saves and archives the original system files. The `aset.restore` utility reinstates these files. This utility also deschedules ASET, if ASET is currently scheduled for periodic execution. The `aset.restore` command is located in `/usr/aset`, the ASET operating directory.

Changes that are made to system files are lost when you run the `aset.restore` command.

You should use the `aset.restore` command in the following instances:

- When you want to remove ASET changes and to restore the original system.

If you want to deactivate ASET permanently, you can remove ASET from `cron` scheduling if the `aset` command had previously been added to root's `crontab`. For instructions on how to use `cron` to remove automatic execution, see "How to Stop Running ASET Periodically" on page 167.

- After a brief period of experimenting with ASET, to restore the original system state.
- When some major system feature is not working properly, and you suspect that ASET is causing the problem.

Network Operation With the NFS System

Generally, ASET is used in standalone mode, even on a system that is part of a network. As system administrator for your standalone system, you are responsible for the security of your system. Therefore, you are responsible for running and managing ASET to protect your system.

You can also use ASET in the NFS distributed environment. As a network administrator, you are responsible for installing, running, and managing various administrative tasks for all your clients. To facilitate ASET management across several client systems, you can make configuration changes that are applied globally to all clients. By globally applying changes, you eliminate the need to log in to each system to repeat the configuration changes.

When you are deciding how to set up ASET on your networked systems, you should consider who you want to control security. You might want users to control some security on their own systems. You might want to centralize responsibility for security control.

Providing a Global Configuration for Each Security Level

A situation might arise where you want to set up more than one network configuration. For example, you might want to set up one configuration for clients that are designated with low security level. You might want to set up another configuration for medium level clients, and yet another configuration with high level.

If you need to create a separate ASET network configuration for each security level, you can create three ASET configurations on the server. You create one configuration for each level. You would export each configuration to the clients with the appropriate security level. Some ASET components that are common to all three configurations could be shared by using links.

Collecting ASET Reports

Not only can you centralize the ASET components on a server, but you can also set up a central directory on a server to collect all ASET reports. The server can be accessed by clients with or without superuser privileges. For instructions on setting up a collection mechanism, see "How to Collect ASET Reports on a Server" on page 167.

By setting up the collection of reports on a server, you can review reports for all clients from one location. You can use this method whether or not a client has superuser privileges. Alternatively, you can leave the reports directory on the local system when you want users to monitor their own ASET reports.

ASET Environment Variables

The following is a list of the ASET environment variables and the values that the variables specify.

`ASETDIR`	Specifies the ASET working directory
`ASETSECLEVEL`	Specifies the security level
`PERIODIC_SCHEDULE`	Specifies the periodic schedule
`TASKS`	Specifies which ASET tasks to run
`UID_ALIASES`	Specifies an aliases file
`YPCHECK`	Determines whether to extend checks to NIS maps and NIS+ tables
`CKLISTPATH_LOW`	Is the directory list for low security
`CKLISTPATH_MED`	Is the directory for medium security
`CKLISTPATH_HIGH`	Is the directory list for high security

The environment variables that are listed in the following sections are found in the `/usr/aset/asetenv` file. The `ASETDIR` and `ASETSECLEVEL` variables are optional. The variables can be set only through the shell by using the `/usr/aset/aset` command. The other environment variables can be set by editing the file.

`ASETDIR` Environment Variable

`ASETDIR` specifies an ASET working directory.

From the C shell, type:

```
% setenv ASETDIR pathname
```

From the Bourne shell or the Korn shell, type:

```
$ ASETDIR=pathname
$ export ASETDIR
```

Set *pathname* to the full path name of the ASET working directory.

`ASETSECLEVEL` Environment Variable

The `ASETSECLEVEL` variable specifies a security level at which ASET tasks are executed.

From the C shell, type:

```
% setenv ASETSECLEVEL level
```

From the Bourne shell or the Korn shell, type:

```
$ ASETSECLEVEL=level
$ export ASETSECLEVEL
```

In these commands, *level* can be set to one of the following:

`low`	Low security level
`med`	Medium security level
`high`	High security level

`PERIODIC_SCHEDULE` Environment Variable

The value of `PERIODIC_SCHEDULE` follows the same format as the `crontab` file. Specify the variable value as a string of five fields enclosed in double quotation marks, with each field separated by a space:

"*minutes hours day-of-month month day-of-week*"

minutes hours	Specifies start time in number of minutes (0-59) after the hour and the hour (0-23).
day-of-month	Specifies the day of the month when ASET should be run, with values from 1-31.
month	Specifies the month of the year when ASET should be run, with values from 1-12.
day-of-week	Specifies the day of the week when ASET should be run, with values from 0-6. Sunday is day 0.

The following rules apply when creating a periodic schedule for ASET:

- You can specify a list of values, each delimited by a comma, for any field.
- You can specify a value as a number, or you can specify the value as a range. A range is a pair of numbers that are joined by a hyphen. A range states that the ASET tasks should be executed for every time that is included in the range.
- You can specify an asterisk (`*`) as the value of any field. An asterisk inclusively specifies all possible values of the field.

The default entry for the `PERIODIC_SCHEDULE` variable causes ASET to execute at 12:00 midnight every day:

```
PERIODIC_SCHEDULE=”0 0 * * *”
```

`TASKS` **Environment Variable**

The `TASKS` variable lists the tasks that ASET performs. The default is to list all seven tasks:

```
TASKS=”env sysconfig usrgrp tune cklist eeprom firewall”
```

`UID_ALIASES` **Environment Variable**

The `UID_ALIASES` variable specifies an aliases file. If present, ASET consults this file for a list of permitted multiple aliases. The format is `UID_ALIASES=`*pathname*, where *pathname* is the full path name of the aliases file.

The default is as follows:

```
UID_ALIASES=${ASETDIR}/masters/uid_aliases
```

`YPCHECK` **Environment Variable**

The `YPCHECK` variable extends the task of checking system tables to include NIS or NIS+ tables. The `YPCHECK` variable is a Boolean variable, which can be set to either true or false.

The default is false, which confines the checking to local system tables:

```
YPCHECK=false
```

`CKLISTPATH_`*level* **Environment Variables**

The three checklist path variables list the directories to be checked by the system files checks task. The following definitions of the variables are set by default. The definitions illustrate the relationship between the variables at different levels:

```
CKLISTPATH_LOW=${ASETDIR}/tasks:${ASETDIR}/util:${ASETDIR}/masters:/etc
CKLISTPATH_MED=${CKLISTPATH_LOW}:/usr/bin:/usr/ucb
CKLISTPATH_HIGH=${CKLISTPATH_MED}:/usr/lib:/sbin:/usr/sbin:/usr/ucblib
```

The values for the checklist path environment variables are similar to the values of the shell path variables. Like the shell path variables, the checklist path environment variables are lists of directory names. The directory names are separated by colons. You use an equal sign (=) to connect the variable name to its value.

ASET File Examples

This section has examples of some ASET files, including the tune files and the aliases file.

Tune File Examples

ASET maintains three tune files. Each entry in a tune file occupies one line. The fields in an entry are in the following order:

pathname mode owner group type

pathname	The full path name of the file
mode	A five-digit number that represents the permission setting
owner	The owner of the file
group	The group owner of the file
type	The type of file

The following rules apply when you edit the tune files:

- You can use regular shell wildcard characters, such as an asterisk (`*`) and a question mark (`?`), in the path name for multiple references. For more information, see `sh`(1).
- *mode* represents the least restrictive value. If the current setting is already more restrictive than the specified value, ASET does not loosen the permission settings. For example, if the specified value is `00777`, the permission remains unchanged, because `00777` is always less restrictive than whatever the current setting is.

  This process is how ASET handles mode setting. The process is different if the security level is being downgraded, or if you are removing ASET. When you decrease the security level from the level in the previous execution, or when you want to restore the system files to the state they were in before ASET was first executed, ASET recognizes what you are doing and decreases the protection level.
- You must use names for *owner* and *group* instead of numeric IDs.
- You can use a question mark (`?`) in place of *owner*, *group*, and *type* to prevent ASET from changing the existing values of these parameters.
- *type* can be `symlink`, directory, or file. A `symlink` is a symbolic link.
- Higher security level tune files reset file permissions to be at least as restrictive as file permissions at lower levels. Also, at higher security levels, additional files are added to the list.
- A file can match more than one tune file entry. For example, `etc/passwd` matches the `etc/pass*` and `/etc/*` entries.

- Where two entries have different permissions, the file permission is set to the most restrictive value. In the following example, the permission of the `/etc/passwd` file is set to `00755`, which is the more restrictive of `00755` and `00770`.

  ```
  /etc/pass* 00755 ? ? file
  /etc/* 00770 ? ? file
  ```

- If two entries have different *owner* designations or *group* designations, the last entry takes precedence. In the following example, the owner of `/usr/sbin/chroot` is set to `root`.

  ```
  /usr/sbin/chroot 00555 bin bin file
  /usr/sbin/chroot 00555 root bin file
  ```

Aliases File Examples

The aliases file contains a list of aliases that share the same user ID.

Each entry is in this form:

uid=*alias1*=*alias2*=*alias3*=`...`

uid	Shared UID.
aliasn	User accounts that share a UID.

For example, the following entry lists the UID `0`. The UID is being shared by the `sysadm` and `root` accounts:

```
0=root=sysadm
```

Running ASET (Task Map)

Task	Description	For Instructions
Run ASET from the command line	Protects the system at the ASET level that you specify. Views the execution log to see the changes.	"How to Run ASET Interactively" on page 165
Run ASET in batch mode at regular intervals	Sets up a cron job to ensure that ASET protects the system.	"How to Run ASET Periodically" on page 166
Stop running ASET in batch mode	Removes the ASET cron job.	"How to Stop Running ASET Periodically" on page 167
Store ASET reports on a server	Collects ASET reports from clients for monitoring in a central location.	"How to Collect ASET Reports on a Server" on page 167

To set the variables in ASET, see "ASET Environment Variables" on page 160. To configure ASET, see "Configuring ASET" on page 156.

▼ How to Run ASET Interactively

1 **Become superuser or assume an equivalent role.**

Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 196.

2 **Run ASET interactively by using the `aset` command.**

`# /usr/aset/aset -l` *level* `-d` *pathname*

level	Specifies the level of security. Valid values are `low`, `medium`, or `high`. The default setting is `low`. For detailed information about security levels, see "ASET Security Levels" on page 148.
pathname	Specifies the working directory for ASET. The default is `/usr/aset`.

3 **Verify that ASET is running by viewing the ASET execution log that is displayed on the screen.**

The execution log message identifies which tasks are being run.

Example 7–1 Running ASET Interactively

In the following example, ASET is run at low security with the default working directory.

```
# /usr/aset/aset -l low
======= ASET Execution Log =======

ASET running at security level low

Machine = jupiter; Current time = 0111_09:26

aset: Using /usr/aset as working directory

Executing task list ...
    firewall
    env
    sysconf
    usrgrp
    tune
    cklist
    eeprom

All tasks executed. Some background tasks may still be running.
```

```
Run /usr/aset/util/taskstat to check their status:
 /usr/aset/util/taskstat [aset_dir]

where aset_dir is ASET's operating
directory,currently=/usr/aset.

When the tasks complete, the reports can be found in:
 /usr/aset/reports/latest/*.rpt

You can view them by:
 more /usr/aset/reports/latest/*.rpt
```

▼ How to Run ASET Periodically

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 196.

2. **If necessary, set up the time when you want ASET to run periodically.**

   You should have ASET run when system demand is light. The `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file is used to set up the time for ASET to run periodically. By default, the time is set for every day at midnight.

   If you want to set up a different time, edit the `PERIODIC_SCHEDULE` variable in the `/usr/aset/asetenv` file. For detailed information about setting the `PERIODIC_SCHEDULE` variable, see "`PERIODIC_SCHEDULE` Environment Variable" on page 161.

3. **Add an entry to the `crontab` file by using the `aset` command.**

   ```
   # /usr/aset/aset -p
   ```

   The `-p` option inserts a line in the `crontab` file that starts ASET running at the time determined by the `PERIODIC_SCHEDULE` environment variable in the `/usr/aset/asetenv` file.

4. **Display the `crontab` entry to verify when ASET is scheduled to run.**

   ```
   # crontab -l root
   ```

▼ How to Stop Running ASET Periodically

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Edit the `crontab` file.**

```
# crontab -e root
```

3 **Delete the ASET entry.**

4 **Save the changes and exit.**

5 **Display the `crontab` entry to verify that the ASET entry is deleted.**

```
# crontab -l root
```

▼ How to Collect ASET Reports on a Server

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **Set up a directory on the server:**

a. **Change to the `/usr/aset` directory.**

```
mars# cd /usr/aset
```

b. **Create a *rptdir* directory.**

```
mars# mkdir rptdir
```

c. **Change to the *rptdir* directory, and create a *client_rpt* directory.**

This step creates a *client_rpt* subdirectory for a client. Repeat this step for each client whose reports you need to collect.

```
mars# cd rptdir
mars# mkdir client_rpt
```

In the following example, the directory `all_reports`, and the subdirectories `pluto_rpt` and `neptune_rpt` are created.

```
mars# cd /usr/aset
mars# mkdir all_reports
mars# cd all_reports
mars# mkdir pluto_rpt
mars# mkdir neptune_rpt
```

3 **Add the *client_rpt* directories to the `/etc/dfs/dfstab` file.**

The directories should have read and write options.

For example, the following entries in the `dfstab` file are shared with read and write permissions.

```
share -F nfs -o rw=pluto /usr/aset/all_reports/pluto_rpt
share -F nfs -o rw=neptune /usr/aset/all_reports/neptune_rpt
```

4 **Make the resources in the `dfstab` file available to the clients.**

```
# shareall
```

5 **On each client, mount the client subdirectory from the server at the mount point, `/usr/aset/masters/reports`.**

```
# mount server:/usr/aset/client_rpt  /usr/aset/masters/reports
```

6 **Edit the `/etc/vfstab` file to mount the directory automatically at boot time.**

The following sample entry in `/etc/vfstab` on `neptune` lists the directory to be mounted from `mars`, `/usr/aset/all_reports/neptune_rpt`, and the mount point on `neptune`, `/usr/aset/reports`. At boot time, the directories that are listed in `vfstab` are automatically mounted.

```
mars:/usr/aset/all_reports/neptune.rpt /usr/aset/reports nfs - yes hard
```

Troubleshooting ASET Problems

This section describes the error messages that are generated by ASET.

ASET Error Messages

`ASET failed: no mail program found.`

Cause: ASET is directed to send the execution log to a user, but no mail program can be found.

Solution: Install a mail program.

```
Usage: aset [-n user[@host]] in /bin/mail or /usr/ucb/mail.
```

```
Cannot decide current and previous security levels.
```

Cause: ASET cannot determine what the security levels are for the current and previous invocations.

Solution: Ensure the current security level is set either through the command-line option or the `ASETSECLEVEL` environment variable. Also, ensure that the last line of `ASETDIR/archives/asetseclevel.arch` correctly reflects the previous security level. If these values are not set, or if these values are incorrect, enter the correct values.

```
ASET working directory undefined.
To specify, set ASETDIR environment variable or use command line option -d.
ASET startup unsuccessful.
```

Cause: The ASET working directory is not defined, or the directory is defined incorrectly. The working directory is the operating directory.

Solution: Use the `ASETDIR` environment variable or the `-d` command-line option to correct the error, and restart ASET.

```
ASET working directory $ASETDIR missing.
ASET startup unsuccessful.
```

Cause: The ASET working directory is not defined, or the directory is defined incorrectly. The working directory is the operating directory. This problem might be because the `ASETDIR` variable refers to a nonexistent directory. Or the `-d` command-line option might refer to a nonexistent directory.

Solution: Ensure that the correct directory, that is, the directory that contains the ASET directory hierarchy, is referred to correctly.

```
Cannot expand $ASETDIR to full pathname.
```

Cause: ASET cannot expand the directory name that is given by the `ASETDIR` variable or the `-d` command-line option to a full path name.

Solution: Ensure that the directory name is correct. Ensure that the directory refers to an existing directory to which the user has access.

```
aset: invalid/undefined security level.
To specify, set ASETSECLEVEL environment variable or use command line option -l,
with argument= low/med/high.
```

Cause: The security level is not defined, or the level is defined incorrectly. Only the values `low`, `med`, or `high` are acceptable.

Solution: Use the `ASETSECLEVEL` variable or the `-l` command-line option to specify one of the three values.

`ASET environment file asetenv not found in $ASETDIR.`

`ASET startup unsuccessful.`

Cause: ASET cannot locate an `asetenv` file in its working directory.

Solution: Ensure there is an `asetenv` file in ASET's working directory. For the details about this file, see the `asetenv`(4) man page.

`filename doesn’t exist or is not readable.`

Cause: The file that is referred to by *filename* either does not exist or is not readable. This problem can occur when you are using the `-u` option. The option permits you to specify a file that contains a list of users whom you want to check.

Solution: Ensure that the argument to the `-u` option exists and that the argument is readable.

`ASET task list TASKLIST undefined.`

Cause: The ASET task list, which should be defined in the `asetenv` file, is not defined. This message can mean that your `asetenv` file is bad.

Solution: Examine your `asetenv` file. Ensure that the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure that the file is intact. For the content of a valid `asetenv` file, see the `asetenv`(4) man page.

`ASET task list $TASKLIST missing.`

`ASET startup unsuccessful.`

Cause: The ASET task list, which should be defined in the `asetenv` file, is not defined. This message can mean that your `asetenv` file is bad.

Solution: Examine your `asetenv` file. Ensure that the task list is defined in the `User Configurable` section. Also check other parts of the file to ensure that the file is intact. For the content of a valid `asetenv` file, see the `asetenv`(4) man page.

`Schedule undefined for periodic invocation.`

`No tasks executed or scheduled. Check asetenv file.`

Cause: ASET scheduling is requested by using the `-p` option, but the environment variable `PERIODIC_SCHEDULE` is undefined in the `asetenv` file.

Solution: Check the `User Configurable` section of the `asetenv` file to ensure that the variable is defined. Ensure that the variable is in proper format.

```
Warning! Duplicate ASET execution scheduled.
```

```
Check crontab file.
```

Cause: ASET is scheduled to run more than once. In other words, ASET scheduling is requested while a schedule is already in effect. This message does not necessarily indicate an error if more than one schedule is indeed desired. In this instance, the messages servers only as a warning. If you want more than one schedule, you should use the proper scheduling format with the `crontab` command. For more information, see the `crontab`(1) man page.

Solution: Verify, through the `crontab` command, that the correct schedule is in effect. Ensure that no unnecessary `crontab` entries for ASET are in place.

PART III

Roles, Rights Profiles, and Privileges

This section covers role-based access control (RBAC) and process rights management. RBAC components include roles, rights profiles, and authorizations. Process rights management is implemented through privileges. Privileges work with RBAC to provide a more secure administration alternative than administration of a system by a superuser.

◆ ◆ ◆

CHAPTER 8

Using Roles and Privileges (Overview)

Solaris role-based access control (RBAC) and privileges provide a more secure alternative to superuser. This chapter provides overview information about RBAC and about privileges.

The following is a list of the overview information in this chapter.

- "Role-Based Access Control (Overview)" on page 175
- "Privileges (Overview)" on page 185

What's New in RBAC?

Solaris 10 8/07: Starting in this release, the `project.max-locked-memory` and `zone.max-locked-memory` resource controls were introduced. If the `PRIV_PROC_LOCK_MEMORY` privilege is assigned to a user or non-global zone, these resource controls can be set to prevent the user or zone from locking all memory. For more discussion, see "Privileges and System Resources" on page 188.

Role-Based Access Control (Overview)

Role-based access control (RBAC) is a security feature for controlling user access to tasks that would normally be restricted to superuser. By applying security attributes to processes and to users, RBAC can divide up superuser capabilities among several administrators. Process rights management is implemented through *privileges*. User rights management is implemented through RBAC.

- For a discussion of process rights management, see "Privileges (Overview)" on page 185.
- For information on RBAC tasks, see Chapter 9, "Using Role-Based Access Control (Tasks)."
- For reference information, see Chapter 10, "Role-Based Access Control (Reference)."

RBAC: An Alternative to the Superuser Model

In conventional UNIX systems, the `root` user, also referred to as superuser, is all-powerful. Programs that run as `root`, or `setuid` programs, are all-powerful. The `root` user has the ability to read and write to any file, run all programs, and send kill signals to any process. Effectively, anyone who can become superuser can modify a site's firewall, alter the audit trail, read confidential records, and shut down the entire network. A `setuid` program that is hijacked can do anything on the system.

Role-based access control (RBAC) provides a more secure alternative to the all-or-nothing superuser model. With RBAC, you can enforce security policy at a more fine-grained level. RBAC uses the security principle of *least privilege*. Least privilege means that a user has precisely the amount of privilege that is necessary to perform a job. Ordinary users have enough privilege to use their applications, check the status of their jobs, print files, create new files, and so on. Capabilities beyond ordinary user capabilities are grouped into rights profiles. Users who are expected to do jobs that require some of the capabilities of superuser assume a role that includes the appropriate rights profile.

RBAC collects superuser capabilities into *rights profiles*. These rights profiles are assigned to special user accounts that are called *roles*. A user can then assume a role to do a job that requires some of superuser's capabilities. Predefined rights profiles are supplied with Solaris software. You create the roles and assign the profiles.

Rights profiles can provide broad capabilities. For example, the Primary Administrator rights profile is equivalent to superuser. Rights profiles can also be narrowly defined. For example, the Cron Management rights profile manages `at` and `cron` jobs. When you create roles, you can decide to create roles with broad capabilities, or roles with narrow capabilities, or both.

In the RBAC model, superuser creates one or more roles. The roles are based on rights profiles. Superuser then assigns the roles to users who are trusted to perform the tasks of the role. Users log in with their user name. After login, users assume roles that can run restricted administrative commands and graphical user interface (GUI) tools.

The flexibility in setting up roles enables a variety of security policies. Although no roles are shipped with the Solaris Operating System (Solaris OS), three recommended roles can easily be configured. The roles are based on rights profiles of the same name:

- **Primary Administrator –** A powerful role that is equivalent to the `root` user, or superuser.
- **System Administrator –** A less powerful role for administration that is not related to security. This role can manage file systems, mail, and software installation. However, this role cannot set passwords.
- **Operator –** A junior administrator role for operations such as backups and printer management.

These three roles do not have to be implemented. Roles are a function of an organization's security needs. Roles can be set up for special-purpose administrators in areas such as security, networking, or firewall administration. Another strategy is to create a single powerful administrator role along with an advanced user role. The advanced user role would be for users who are permitted to fix portions of their own systems.

The superuser model and the RBAC model can co-exist. The following table summarizes the gradations from superuser to restricted ordinary user that are possible in the RBAC model. The table includes the administrative actions that can be tracked in both models. For a summary of the effect of privileges alone on a system, see Table 8–2.

TABLE 8–1 Superuser Model Versus RBAC With Privileges Model

User Capabilities on a System	Superuser Model	RBAC Model
Can become superuser with full superuser capability	Yes	Yes
Can log in as a user with full user capabilities	Yes	Yes
Can become superuser with limited capabilities	No	Yes
Can log in as a user, and have superuser capabilities, sporadically	Yes, with `setuid` programs only	Yes, with `setuid` programs and with RBAC
Can log in as a user with administrative capabilities, but without full superuser capability	No	Yes, with RBAC and with directly-assigned privileges and authorizations
Can log in as a user with fewer capabilities than an ordinary user	No	Yes, with RBAC and with removed privileges
Can track superuser actions	Yes, by auditing the `su` command	Yes, by auditing profile shell commands Also, if `root` user is disabled, the name of the user who has assumed the `root` role is in the audit trail

Solaris RBAC Elements and Basic Concepts

The RBAC model in the Solaris OS introduces the following elements:

- **Authorization** – A permission that enables a user or role to perform a class of actions that could affect security. For example, security policy at installation gives ordinary users the `solaris.device.cdrw` authorization. This authorization enables users to read and write to a CD-ROM device. For a list of authorizations, see the `/etc/security/auth_attr` file.
- **Privilege** – A discrete right that can be granted to a command, a user, a role, or a system. Privileges enable a process to succeed. For example, the `proc_exec` privilege allows a process to call `execve()`. Ordinary users have basic privileges. To see your basic privileges, run the `ppriv -vl basic` command.
- **Security attributes** – An attribute that enables a process to perform an operation. In a typical UNIX environment, a security attribute enables a process to perform an operation that is otherwise forbidden to ordinary users. For example, `setuid` and `setgid` programs have security attributes. In the RBAC model, operations that ordinary users perform might require security attributes. In addition to `setuid` and `setgid` programs, authorizations and privileges are also security attributes in the RBAC model. For example, a user with the `solaris.device.allocate` authorization can allocate a device for exclusive use. A process with the `sys_time` privilege can manipulate system time.
- **Privileged application** – An application or command that can override system controls by checking for *security attributes*. In a typical UNIX environment and in the RBAC model, programs that use `setuid` and `setgid` are privileged applications. In the RBAC model, programs that require privileges or authorizations to succeed are also privileged applications. For more information, see "Privileged Applications and RBAC" on page 181.
- **Rights profile** – A collection of administrative capabilities that can be assigned to a role or to a user. A rights profile can consist of authorizations, of commands with security attributes, and of other rights profiles. Rights profiles offer a convenient way to group security attributes.
- **Role** – A special identity for running privileged applications. The special identity can be assumed by assigned users only. In a system that is run by roles, superuser is unnecessary. Superuser capabilities are distributed to different roles. For example, in a two-role system, security tasks would be handled by a security role. The second role would handle system administration tasks that are not security-related. Roles can be more fine-grained. For example, a system could include separate administrative roles for handling the cryptographic framework, printers, system time, file systems, and auditing.

The following figure shows how the RBAC elements work together.

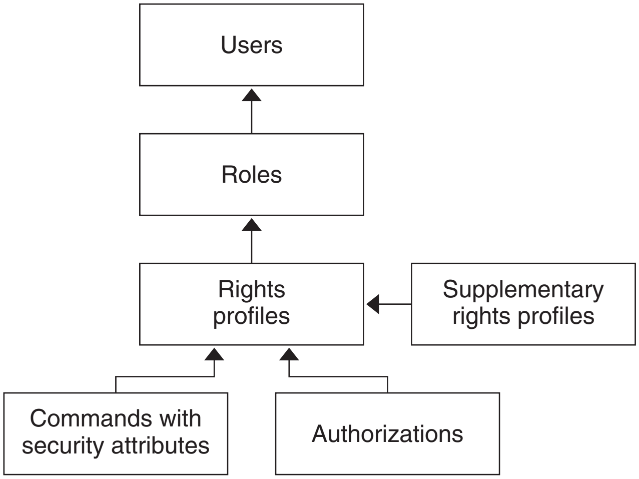

Users
Roles
Rights profiles
Supplementary rights profiles
Commands with security attributes
Authorizations

FIGURE 8–1 Solaris RBAC Element Relationships

In RBAC, roles are assigned to users. When a user assumes a role, the capabilities of the role are available. Roles get their capabilities from rights profiles. Rights profiles can contain authorizations, privileged commands, and other supplementary rights profiles. Privileged commands are commands that execute with security attributes.

The following figure uses the Operator role, the Operator rights profile, and the Printer Management rights profile to demonstrate RBAC relationships.

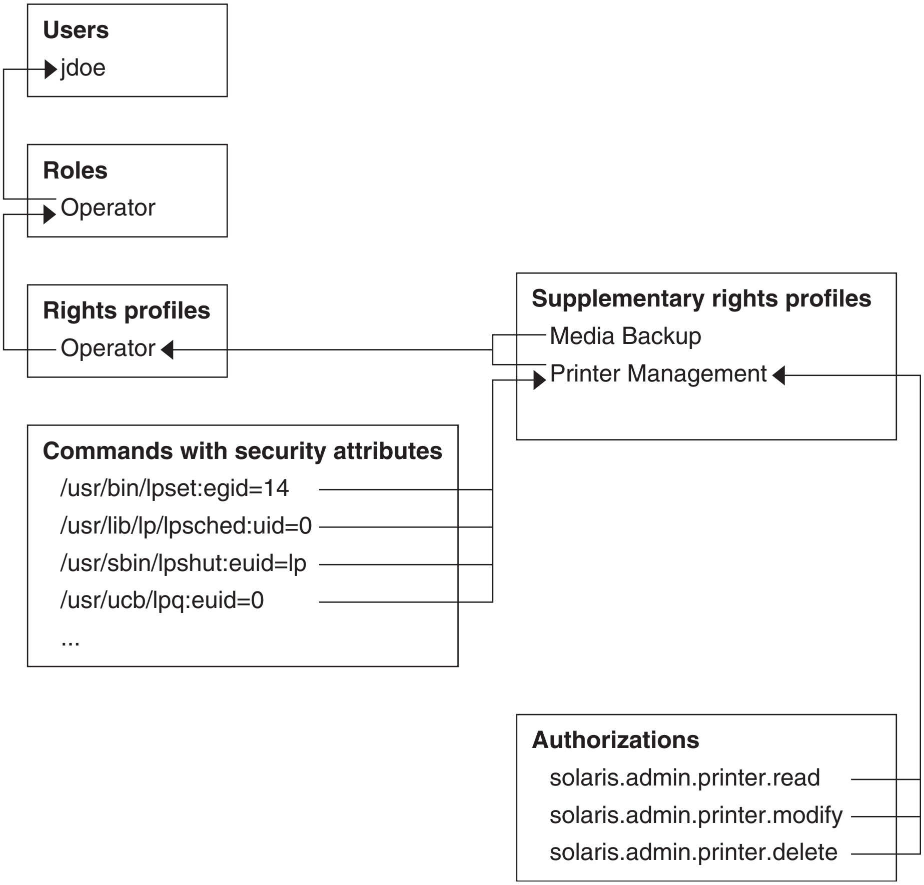

Users
jdoe
Roles
Operator
Rights profiles
Operator
Supplementary rights profiles
Media Backup
Printer Management
Commands with security attributes
/usr/bin/lpset:egid=14
/usr/lib/lp/lpsched:uid=0
/usr/sbin/lpshut:euid=lp
/usr/ucb/lpq:euid=0
...
Authorizations
solaris.admin.printer.read
solaris.admin.printer.modify
solaris.admin.printer.delete

FIGURE 8–2 Example of Solaris RBAC Element Relationships

The Operator role is used to maintain printers and to perform media backup. The role is assigned to the user `jdoe`. `jdoe` can assume the role by switching to the role, and then supplying the role password.

The Operator rights profile has been assigned to the Operator role. The Operator rights profile contains two supplementary profiles, Printer Management and Media Backup. The supplementary profiles reflect the role's primary tasks.

The Printer Management rights profile is for managing printers, print daemons, and spoolers. Three authorizations are included in the Printer Management rights profile: `solaris.admin.printer.read`, `solaris.admin.printer.delete`, and `solaris.admin.printer.modify`. These authorizations enable roles and users to manipulate information in the printer queue. The Printer Management rights profile also includes a number of commands with security attributes, such as `/usr/sbin/lpshut` with `euid=lp` and `/usr/ucb/lpq` with `euid=0`.

RBAC Authorizations

An *authorization* is a discrete right that can be granted to a role or to a user. Authorizations enforce policy at the user application level. Authorizations can be assigned directly to a role or to a user. Typically, authorizations are included in a rights profile. The rights profile is then included in a role, and the role is assigned to a user. For an example, see Figure 8–2.

RBAC-compliant applications can check a user's authorizations prior to granting access to the application or specific operations within the application. This check replaces the check in conventional UNIX applications for `UID=0`. For more information on authorizations, see the following sections:

- "Authorization Naming and Delegation" on page 228
- "`auth_attr` Database" on page 232
- "Commands That Require Authorizations" on page 237

Authorizations and Privileges

Privileges enforce security policy in the kernel. The difference between authorizations and privileges concerns the level at which the security policy is enforced. Without the proper privilege, a process can be prevented from performing privileged operations by the kernel. Without the proper authorizations, a user might be prevented from using a privileged application or from performing security-sensitive operations within a privileged application. For a fuller discussion of privileges, see "Privileges (Overview)" on page 185.

Privileged Applications and RBAC

Applications and commands that can override system controls are considered privileged applications. Security attributes such as `UID=0`, privileges, and authorizations make an application privileged.

Applications That Check UIDs and GIDs

Privileged applications that check for `root` (`UID=0`) or some other special UID or GID have long existed in the UNIX environment. The rights profile mechanism enables you to isolate commands that require a specific ID. Instead of changing the ID on a command that anyone can access, you can place the command with execution security attributes in a rights profile. A user or role with that rights profile can then run the program without having to become superuser.

IDs can be specified as real or effective. Assigning effective IDs is preferred over assigning real IDs. Effective IDs are equivalent to the `setuid` feature in the file permission bits. Effective IDs also identify the UID for auditing. However, because some shell scripts and programs require a real UID of `root`, real UIDs can be set as well. For example, the `pkgadd` command requires a real rather than an effective UID. If an effective ID is not sufficient to run a command, you need to change the ID to a real ID. For the procedure, see "How to Create or Change a Rights Profile" on page 215.

Applications That Check for Privileges

Privileged applications can check for the use of privileges. The RBAC rights profile mechanism enables you to specify the privileges for specific commands. Instead of requiring superuser capabilities to use an application or command, you can isolate the command with execution security attributes in a rights profile. A user or role with that rights profile can then run the command with just the privileges that the command requires to succeed.

Commands that check for privileges include the following:

- Kerberos commands, such as `kadmin`, `kprop`, and `kdb5_util`
- Network commands, such as `ifconfig`, `routeadm`, and `snoop`
- File and file system commands, such as `chmod`, `chgrp`, and `mount`
- Commands that control processes, such as `kill`, `pcred`, and `rcapadm`

To add commands with privileges to a rights profile, see "How to Create or Change a Rights Profile" on page 215. To determine what commands check for privileges in a particular profile, see "Determining Your Assigned Privileges" on page 248.

Applications That Check Authorizations

The Solaris OS additionally provides commands that check authorizations. By definition, the `root` user has all authorizations. Therefore, the `root` user can run any application. Applications that check for authorizations include the following:

- The entire Solaris Management Console suite of tools
- Audit administration commands, such as `auditconfig` and `auditreduce`
- Printer administration commands, such as `lpadmin` and `lpfilter`
- The batch job-related commands, such as `at`, `atq`, `batch`, and `crontab`
- Device-oriented commands, such as `allocate`, `deallocate`, `list_devices`, and `cdrw`.

To test a script or program for authorizations, see Example 9–19. To write a program that requires authorizations, see "About Authorizations" in *Solaris Security for Developers Guide*.

RBAC Rights Profiles

A *rights profile* is a collection of system overrides that can be assigned to a role or user. A rights profile can include authorizations, commands with assigned security attributes, and other rights profiles. Rights profile information is split between the `prof_attr` and `exec_attr` databases. The rights profile name and authorizations are in the `prof_attr` database. The rights profile name and the commands with assigned security attributes are in the `exec_attr` database.

For more information on rights profiles, see the following sections:

- "Contents of Rights Profiles" on page 223
- "`prof_attr` Database" on page 233
- "`exec_attr` Database" on page 234

RBAC Roles

A *role* is a special type of user account from which you can run privileged applications. Roles are created in the same general manner as user accounts. Roles have a home directory, a group assignment, a password, and so on. Rights profiles and authorizations give the role administrative capabilities. Roles cannot inherit capabilities from other roles or other users. Discrete roles parcel out superuser capabilities, and thus enable more secure administrative practices.

When a user assumes a role, the role's attributes replace all user attributes. Role information is stored in the `passwd`, `shadow`, and `user_attr` databases. Role information can be added to the `audit_user` database. For detailed information on setting up roles, see the following sections:

- "How to Plan Your RBAC Implementation" on page 197
- "How to Create a Role From the Command Line" on page 202
- "How to Change the Properties of a Role" on page 213

A role can be assigned to more than one user. All users who can assume the same role have the same role home directory, operate in the same environment, and have access to the same files. Users can assume roles from the command line by running the `su` command and supplying the role name and password. Users can also assume a role in the Solaris Management Console tool.

A role cannot log in directly. A user logs in, and then assumes a role. Having assumed a role, the user cannot assume another role without first exiting their current role. Having exited the role, the user can then assume another role.

You can prevent anonymous `root` login by changing the `root` user into a role, as shown in "How to Make `root` User Into a Role" on page 207. If the profile shell command, `pfexec`, is being audited, the audit trail contains the login user's real UID, the roles that the user has assumed, and the actions that the role performed. To audit the system or a particular user for role operations, see "How to Audit Roles" on page 206.

No predefined roles are shipped with Solaris software.

- To configure the Primary Administrator role, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.
- To configure other roles, see "How to Create and Assign a Role By Using the GUI" on page 199.
- To create roles on the command line, see "Managing RBAC (Task Map)" on page 213.

Profile Shell in RBAC

Roles can run privileged applications from the Solaris Management Console launcher or from a *profile shell*. A profile shell is a special shell that recognizes the security attributes that are included in a rights profile. Profile shells are launched when the user runs the `su` command to assume a role. The profile shells are `pfsh`, `pfcsh`, and `pfksh`. The shells correspond to Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`), respectively.

Users who have been directly assigned a rights profile must invoke a profile shell to run the commands with security attributes. For usability and security considerations, see "Security Considerations When Directly Assigning Security Attributes" on page 184.

All commands that are executed in a profile shell can be audited. For more information, see "How to Audit Roles" on page 206.

Name Service Scope and RBAC

Name service scope is an important concept for understanding RBAC. The scope of a role might be limited to an individual host. Alternatively, the scope might include all hosts that are served by a name service such as NIS, NIS+, or LDAP. The name service scope for a system is specified in the file `/etc/nsswitch.conf`. A lookup stops at the first match. For example, if a rights profile exists in two name service scopes, only the entries in the first name service scope are used. If `files` is the first match, then the scope of the role is limited to the local host.

Security Considerations When Directly Assigning Security Attributes

Typically, a user obtains administrative capabilities through a role. Authorizations and privileged commands are grouped into a rights profile. The rights profile is included in a role, and the role is assigned to a user.

Direct assignment of rights profiles and security attributes is also possible:

- Rights profiles, privileges, and authorizations can be assigned directly to users.
- Privileges and authorizations can be assigned directly to roles.

However, direct assignment is not a secure practice. Users and roles with a directly assigned privilege could override security policy wherever this privilege is required by the kernel. When a privilege is a security attribute of a command in a rights profile, that privilege is available only for that command by someone who has that rights profile. The privilege is not available for other commands that the user or role might run.

Since authorizations act at the user level, direct assignment of authorizations can be less dangerous than direct assignment of privileges. However, authorizations can enable a user to perform highly secure tasks, such as delegate device administration.

A rights profile that is assigned directly to a user presents usability problems more than security problems. The commands with security attributes in the rights profile can only succeed in a profile shell. The user must open a profile shell, then type the commands. A role that is assigned a rights profile gets a profile shell automatically. Therefore, the commands succeed in the role's shell.

Rights profiles provide an extensible, clean way to group security characteristics for particular administrative tasks.

Privileges (Overview)

Process rights management enables processes to be restricted at the command, user, role, or system level. The Solaris OS implements process rights management through *privileges*. Privileges decrease the security risk that is associated with one user or one process having full superuser capabilities on a system. Privileges and RBAC provide a compelling alternative model to the traditional superuser model.

- For information on RBAC, see “Role-Based Access Control (Overview)” on page 175.
- For information on how to administer privileges, see Chapter 11, “Privileges (Tasks).”
- For reference information on privileges, see Chapter 12, “Privileges (Reference).”

Privileges Protect Kernel Processes

A privilege is a discrete right that a process requires to perform an operation. The right is enforced in the kernel. A program that operates within the bounds of the Solaris *basic set* of privileges operates within the bounds of the system security policy. `setuid` programs are examples of programs that operate outside the bounds of the system security policy. By using privileges, programs eliminate the need for calls to `setuid`.

Privileges discretely enumerate the kinds of operations that are possible on a system. Programs can be run with the exact privileges that enable the program to succeed. For example, a program that sets the date and writes the date to an administrative file might require the `file_dac_write` and `sys_time` privileges. This capability eliminates the need to run any program as `root`.

Historically, systems have not followed the privilege model. Rather, systems used the superuser model. In the superuser model, processes run as `root` or as a user. User processes were limited to acting on the user's directories and files. `root` processes could create directories and files anywhere on the system. A process that required creation of a directory outside the user's directory would run with a `UID=0`, that is, as `root`. Security policy relied on DAC, discretionary access control, to protect system files. Device nodes were protected by DAC. For example, devices owned by group `sys` could be opened only by members of group `sys`.

However, `setuid` programs, file permissions, and administrative accounts are vulnerable to misuse. The actions that a `setuid` process is permitted are more numerous than the process requires to complete its operation. A `setuid` program can be compromised by an intruder who then runs as the all-powerful `root` user. Similarly, any user with access to the `root` password can compromise the entire system.

In contrast, a system that enforces policy with privileges allows a gradation between user capabilities and `root` capabilities. A user can be granted privileges to perform activities that are beyond the capabilities of ordinary users, and `root` can be limited to fewer privileges than `root` currently possesses. With RBAC, a command that runs with privileges can be isolated in a rights profile and assigned to one user or role. Table 8–1 summarizes the gradation between user capabilities and root capabilities that the RBAC plus privileges model provides.

The privilege model provides greater security than the superuser model. Privileges that have been removed from a process cannot be exploited. Process privileges prevent a program or administrative account from gaining access to all capabilities. Process privileges can provide an additional safeguard for sensitive files, where DAC protections alone can be exploited to gain access.

Privileges, then, can restrict programs and processes to just the capabilities that the program requires. This capability is called the *principle of least privilege*. On a system that implements least privilege, an intruder who captures a process has access to only those privileges that the process has. The rest of the system cannot be compromised.

Privilege Descriptions

Privileges are logically grouped on the basis of the area of the privilege.

- `FILE` **privileges** – Privileges that begin with the string `file` operate on file system objects. For example, the `file_dac_write` privilege overrides discretionary access control when writing to files.

- `IPC` **privileges –** Privileges that begin with the string `ipc` override IPC object access controls. For example, the `ipc_dac_read` privilege enables a process to read remote shared memory that is protected by DAC.
- `NET` **privileges –** Privileges that begin with the string `net` give access to specific network functionality. For example, the `net_rawaccess` privilege enables a device to connect to the network.
- `PROC` **privileges –** Privileges that begin with the string `proc` allow processes to modify restricted properties of the process itself. `PROC` privileges include privileges that have a very limited effect. For example, the `proc_clock_highres` privilege enables a process to use high resolution timers.
- `SYS` **privileges –** Privileges that begin with the string `sys` give processes unrestricted access to various system properties. For example, the `sys_linkdir` privilege enables a process to make and break hard links to directories.

Some privileges have a limited effect on the system, and some have a broad effect. The definition of the `proc_taskid` privilege indicates its limited effect:

```
proc_taskid
        Allows a process to assign a new task ID to the calling process.
```

The definition of the `file_setid` privilege indicates its broad effect:

```
net_rawaccess
        Allow a process to have direct access to  the  network layer.
```

The `privileges`(5) man page provides descriptions of every privilege. The command `ppriv -lv` prints a description of every privilege to standard out.

Administrative Differences on a System With Privileges

A system that has privileges has several visible differences from a system that does not have privileges. The following table lists some of the differences.

TABLE 8–2 Visible Differences Between a System With Privileges and a System Without Privileges

Feature	No Privileges	Privileges
Daemons	Daemons run as `root`.	Daemons run as the user `daemon`. For example, the following daemons have been assigned appropriate privileges and run as `daemon`: `lockd`, `mountd`, `nfsd`, and `rpcbind`.

TABLE 8–2 Visible Differences Between a System With Privileges and a System Without Privileges *(Continued)*

Feature	No Privileges	Privileges
Log File Ownership	Log files are owned by `root`.	Log files are now owned by `daemon`, who created the log file. The `root` user does not own the file.
Error Messages	Error messages refer to superuser. For example, `chroot: not superuser`.	Error messages reflect the use of privileges. For example, the equivalent error message for `chroot` failure is `chroot: exec failed`.
`setuid` Programs	Programs use `setuid` to complete tasks that ordinary users are not allowed to perform.	Many `setuid` programs have been changed to run with privileges. For example, the following utilities use privileges: `ufsdump`, `ufsrestore`, `rsh`, `rlogin`, `rcp`, `rdist`, `ping`, `traceroute`, and `newtask`.
File Permissions	Device permissions are controlled by DAC. For example, members of the group `sys` can open `/dev/ip`.	File permissions (DAC) do not predict who can open a device. Devices are protected with DAC *and* device policy. For example, the `/dev/ip` file has `666` permissions, but the device can only be opened by a process with the appropriate privileges. Raw sockets are still protected by DAC.
Audit Events	Auditing the use of the `su` command covers many administrative functions.	Auditing the use of privileges covers most administrative functions. The `pm` and `as` audit classes include audit events that configure device policy and audit events that set privileges.
Processes	Processes are protected by who owns the process.	Processes are protected by privileges. Process privileges and process flags are visible as a new entry in the `/proc/<pid>` directory, `priv`.
Debugging	No reference to privileges in core dumps.	The ELF note section of core dumps includes information about process privileges and flags in the `NT_PRPRIV` and `NT_PRPRIVINFO` notes. The `ppriv` utility and other utilities show the proper number of properly sized sets. The utilities correctly map the bits in the bit sets to privilege names.

Privileges and System Resources

Starting in the Solaris 10 8/07 release, the `project.max-locked-memory` and `zone.max-locked-memory` resource controls can be used to limit the memory consumption of processes that are assigned the `PRIV_PROC_LOCK_MEMORY` privilege. This privilege allows a process to lock pages in physical memory.

If you assign the `PRIV_PROC_LOCK_MEMORY` privilege to a rights profile, you can give the processes that have this privilege the ability to lock all memory. As a safeguard, set a resource control to prevent the user of the privilege from locking all memory. For privileged processes that run in a non-global zone, set the `zone.max-locked-memory` resource control. For privileged processes that run on a system, create a project and set the `project.max-locked-memory` resource control. For information about these resource controls, see Chapter 6, "Resource Controls (Overview)," in *System Administration Guide: Virtualization Using the Solaris Operating System* and Chapter 17, "Non-Global Zone Configuration (Overview)," in *System Administration Guide: Virtualization Using the Solaris Operating System*.

How Privileges Are Implemented

Every process has four sets of privileges that determine whether a process can use a particular privilege. The kernel automatically calculates the *effective set* of privileges. You can modify the initial *inheritable set* of privileges. A program that is coded to use privileges can reduce the program's *permitted set* of privileges. You can shrink the *limit set* of privileges.

- **Effective privilege set, or E –** Is the set of privileges that is currently in effect. A process can add privileges that are in the permitted set to the effective set. A process can also remove privileges from E.
- **Permitted privilege set, or P –** Is the set of privileges that is available for use. Privileges can be available to a program from inheritance or through assignment. An execution profile is one way to assign privileges to a program. The `setuid` command assigns all privileges that `root` has to a program. Privileges can be removed from the permitted set, but privileges cannot be added to the set. Privileges that are removed from P are automatically removed from E.

  A *privilege-aware* program removes the privileges that a program never uses from the program's permitted set. In this way, unnecessary privileges cannot be exploited by the program or a malicious process. For more information on privilege-aware programs, see Chapter 2, "Developing Privileged Applications," in *Solaris Security for Developers Guide*.
- **Inheritable privilege set, or I –** Is the set of privileges that a process can inherit across a call to `exec`. After the call to `exec`, the permitted and the effective sets are equal, except in the special case of a `setuid` program.

  For a `setuid` program, after the call to `exec`, the inheritable set is first restricted by the limit set. Then, the set of privileges that were inherited (I), minus any privileges that were in the limit set (L), are assigned to P and E for that process.
- **Limit privilege set, or L –** Is the outside limit of what privileges are available to a process and its children. By default, the limit set is all privileges. Processes can shrink the limit set but can never extend the limit set. L is used to restrict I. Consequently, L restricts P and E at the time of `exec`.

  If a user has been assigned a profile that includes a program that has been assigned privileges, the user can usually run that program. On an unmodified system, the program's assigned privileges are within the user's limit set. The privileges that have been assigned to the program become part of the user's permitted set. To run the program that has been assigned privileges, the user must run the program from a profile shell.

The kernel recognizes a *basic privilege set*. On an unmodified system, each user's initial inheritable set equals the basic set at login. You can modify the user's initial inheritable set. You cannot modify the basic set.

On an unmodified system, a user's privilege sets at login would appear similar to the following:

```
E (Effective): basic
I (Inheritable): basic
P (Permitted): basic
L (Limit): all
```

Therefore, at login, all users have the basic set in their inheritable set, their permitted set, and their effective set. A user's limit set contains all privileges. To put more privileges in the user's effective set, you must assign a rights profile to the user. The rights profile would include commands to which you have added privileges. You can also assign privileges directly to the user or role, though such privilege assignment can be risky. For a discussion of the risks, see "Security Considerations When Directly Assigning Security Attributes" on page 184.

How Processes Get Privileges

Processes can inherit privileges. Or, processes can be assigned privileges. A process inherits privileges from its parent process. At login, the user's initial inheritable set of privileges determines what privileges are available to the user's processes. All child processes of the user's initial login inherit that set.

You can also directly assign privileges to programs, users, and roles. When a program requires privileges, you assign the privileges to the program's executable in a rights profile. Users or roles that are permitted to run the program are assigned the profile that includes the program. At login or when a profile shell is entered, the program runs with privilege when the program's executable is typed in the profile shell. For example, a role that includes the Object Access Management profile is able to run the `chmod` command with the `file_chown` privilege.

When a role or user runs a program that has been directly assigned an additional privilege, the assigned privilege is added to the role or user's inheritable set. Child processes of the program that was assigned privileges inherit the privileges of the parent. If the child process requires more privileges than the parent process, the child process must be directly assigned those privileges.

Programs that are coded to use privileges are called privilege-aware programs. A privilege-aware program turns on the use of privilege and turns off the use of privilege during program execution. To succeed in a production environment, the program must be assigned the privileges that the program turns on and off.

For examples of privilege-aware code, see Chapter 2, "Developing Privileged Applications," in *Solaris Security for Developers Guide*. To assign privileges to a program that requires privileges, see "How to Add Privileges to a Command" on page 243.

Assigning Privileges

You, in your capacity as system administrator, are responsible for assigning privileges. Typically, you assign the privilege to a command in a rights profile. The rights profile is then assigned to a role or to a user. The Solaris Management Console provides the graphical user interface (GUI) to assign privileges. Privileges can also be assigned by using commands such as `smuser` and `smrole`. For more information on how to use the GUI to assign privileges, see Chapter 9, "Using Role-Based Access Control (Tasks)."

Privileges can also be assigned directly to a user. If you trust a subset of users to use a privilege responsibly throughout their sessions, you can assign the privilege directly. Good candidates for direct assignment are privileges that have a limited effect, such as `proc_clock_highres`. Poor candidates for direct assignment are privileges that have far-reaching effects, such as `file_dac_write`.

Privileges can also be denied to a user or to a system. Care must be taken when removing privileges from the initial inheritable set or the limit set of a user or a system.

Expanding a User or Role's Privileges

Users and roles have an inheritable set of privileges, and a limit set of privileges. The limit set cannot be expanded, since the limit set is initially all privileges. The initial inheritable set can be expanded for users, roles, and systems. A privilege that is not in the inheritable set can also be assigned to a process.

The assignment of privileges per process is the most precise way to add privileges. You can expand the number of privileged operations that a user can perform by enabling the user to assume a role. The role would be assigned profiles that include commands with added privileges. When the user assumes the role, the user gets the role's profile shell. By typing in the role's shell, the commands in the role's profiles execute with the added privileges.

You can also assign a profile to the user rather than to a role that the user assumes. The profile would include commands with added privileges. When the user opens a profile shell, such as `pfksh`, the user can execute the commands in the user's profile with privilege. In a regular shell, the commands do not execute with privilege. The privileged process can only execute in a privileged shell.

To expand the initial inheritable set of privileges for users, roles, or systems is a riskier way to assign privileges. All privileges in the inheritable set are in the permitted and effective sets. All commands that the user or role types in a shell can use the directly assigned privileges. Directly assigned privileges enable a user or role to easily perform operations that can be outside the bounds of their administrative responsiblities.

When you add to the initial inheritable set of privileges on a system, all users who log on to the system have a larger set of basic privileges. Such direct assignment enables all users of the system to easily perform operations that are probably outside the bounds of ordinary users.

Restricting a User or Role's Privileges

By removing privileges, you can prevent users and roles from performing particular tasks. You can remove privileges from the initial inheritable set, and from the limit set. You should carefully test removal of privileges before you distribute an initial inheritable set or a limit set that is smaller than the default set. By removing privileges from the initial inheritable set, you might prevent users from logging in. When privileges are removed from the limit set, a legacy `setuid` program might fail because the program requires a privilege that was removed.

Assigning Privileges to a Script

Scripts are executables, like commands. Therefore, in a rights profile, you can add privileges to a script just as you can add privileges to a command. The script runs with the added privileges when a user or role who has been assigned the profile executes the script in a profile shell. If the script contains commands that require privileges, the commands with added privileges should also be in the profile.

Privilege-aware programs can restrict privileges per process. Your job with a privilege-aware program is to assign the executable just the privileges that the program needs. You then test the program to see that the program succeeds in performing its tasks. You also check that the program does not abuse its use of privileges.

Privileges and Devices

The privilege model uses privileges to protect system interfaces that are protected by file permissions alone in the superuser model. In a system with privileges, file permissions are too weak to protect the interfaces. A privilege such as `proc_owner` could override file permissions and then give full access to all of the system.

Therefore, ownership of the device directory is not sufficient to open a device. For example, members of the group `sys` are no longer automatically allowed to open the `/dev/ip` device. The file permissions on `/dev/ip` are `0666`, but the `net_rawaccess` privilege is required to open the device.

Device policy is controlled by privileges. The `getdevpolicy` command displays the device policy for every device. The device configuration command, `devfsadm`, installs the device policy. The `devfsadm` command binds privilege sets with `open` for reading or writing of devices. For more information, see the `getdevpolicy`(1M) and `devfsadm`(1M) man pages.

Device policy allows you more flexibility in granting permission to open devices. You can require different privileges or more privileges than the default device policy. The privilege requirements can be modified for the device policy and for the driver proper. You can modify the privileges when installing, adding, or updating a device driver.

The `add_drv` and `update_drv` commands can modify device policy entries and driver-specific privileges. You must be running a process with the full set of privileges to change the device policy. For more information, see the `add_drv`(1M) and `update_drv`(1M) man pages.

Privileges and Debugging

The Solaris OS provides tools to debug privilege failure. The `ppriv` command and the `truss` command provide debugging output. For examples, see the `ppriv(1)` man page. For a procedure, see "How to Determine Which Privileges a Program Requires" on page 242.

◆ ◆ ◆

CHAPTER 9

Using Role-Based Access Control (Tasks)

This chapter covers tasks for distributing the capabilities of superuser by using discrete roles. The mechanisms that roles can use include rights profiles, authorizations, and privileges. The following is a list of the task maps in this chapter.

- "Using RBAC (Task Map)" on page 195
- "Configuring RBAC (Task Map)" on page 196
- "Using Roles (Task Map)" on page 209
- "Managing RBAC (Task Map)" on page 213

For an overview of RBAC, see "Role-Based Access Control (Overview)" on page 175. For reference information, see Chapter 10, "Role-Based Access Control (Reference)." To use privileges with RBAC or without RBAC, see Chapter 11, "Privileges (Tasks)."

Using RBAC (Task Map)

To use RBAC requires planning, configuring RBAC, and knowing how to assume a role. Once roles become familiar, you might further customize RBAC to handle new operations. The following task map points to these major tasks.

Task	Description	For Instructions
Plan and configure RBAC	Configure RBAC at your site.	"Configuring RBAC (Task Map)" on page 196
Use roles	Assume roles from the command line and in the Solaris Management Console GUI.	"Using Roles (Task Map)" on page 209
Customize RBAC	Customize RBAC for your site.	"Managing RBAC (Task Map)" on page 213

Configuring RBAC (Task Map)

To use RBAC effectively requires planning. Use the following task map to plan and initially implement RBAC at your site.

Task	Description	For Instructions
1. Plan for RBAC	Involves examining your site's security needs, and deciding how to use RBAC at your site.	"How to Plan Your RBAC Implementation" on page 197
2. Learn to use the Solaris Management Console	Involves becoming familiar with the Solaris Management Console.	Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*
3. Configure the first user and role	Uses the RBAC configuration tools in the Solaris Management Console to create a user and a role, and to assign the role to the user.	"Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*
4. (Optional) Create other users who can assume roles	Ensures that users who can assume an administrative role exist.	"Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*
5. (Recommended) Create other roles and assign them to users	Uses the RBAC tools to create roles for particular administrative areas, and to assign the roles to users.	"How to Create and Assign a Role By Using the GUI" on page 199
		Example 9–5
	Uses the command line to create roles, and to assign the roles to users	"How to Create a Role From the Command Line" on page 202
		"How to Assign a Role to a Local User" on page 205
6. (Recommended) Audit role actions	Preselect an audit class that includes the audit event that records role actions.	"How to Audit Roles" on page 206
7. (Optional) Make `root` user a role	Prevents anonymous `root` login, which is a security hole.	"How to Make `root` User Into a Role" on page 207

Configuring RBAC

RBAC can be configured with the following utilities:

- **Solaris Management Console GUI –** The preferred method for performing RBAC-related tasks is through the GUI. The console tools for managing the RBAC elements are contained in the Users Tool collection.
- **Solaris Management Console commands –** With the Solaris Management Console command-line interfaces, such as `smrole`, you can operate on any name service. The Solaris Management Console commands require authentication to connect to the server. As a result, these commands are not practical for use in scripts.
- **Local commands –** With the `user*` and `role*` set of command-line interfaces, such as `useradd`, you can operate on local files only. The commands that operate on local files must be run by superuser or by a role with the appropriate privileges.

▼ How to Plan Your RBAC Implementation

RBAC can be an integral part of how an organization manages its information resources. Planning requires a thorough knowledge of the RBAC capabilities as well as the security requirements of your organization.

1 **Learn the basic RBAC concepts.**

Read "Role-Based Access Control (Overview)" on page 175. Using RBAC to administer a system is very different from using conventional UNIX administrative practices. You should be familiar with the RBAC concepts before you start your implementation. For greater detail, see Chapter 10, "Role-Based Access Control (Reference)."

2 **Examine your security policy.**

Your organization's security policy should detail the potential threats to your system, measure the risk of each threat, and have a strategy to counter these threats. Isolating the security-relevant tasks through RBAC can be a part of the strategy. Although you can install the recommended roles and their configurations as is, you might need to customize your RBAC configuration to adhere to your security policy.

3 **Decide how much RBAC your organization needs.**

Depending on your security needs, you can use varying degrees of RBAC, as follows:

- **No RBAC –** You can perform all tasks as `root` user. In this configuration, you log in as yourself. Then, you type `root` as the user when you select a Solaris Management Console tool.
- **Single Role Only –** This method adds one role. The one role is assigned the Primary Administrator rights profile. This method is similar to the superuser model, in that the role has superuser capabilities. However, this method enables you to track the user who has assumed the role.
- **Recommended Roles –** This method creates three roles that are based on the following rights profiles: Primary Administrator, System Administrator, and Operator. The roles are suitable for organizations with administrators at different levels of responsibility.
- **Custom Roles –** You can create your own roles to meet the security requirements of your organization. The new roles can be based on existing or customized rights profiles.
- **Root User as a Role –** This method prevents any user from logging in as `root`. Instead, users must log in as ordinary users prior to assuming the `root` role. For details, see "How to Make `root` User Into a Role" on page 207.

4. **Decide which recommended roles are appropriate for your organization.**

Review the capabilities of the recommended roles and default rights profiles. Default rights profiles enable administrators to configure a recommended role by using a single profile.

Three default rights profiles are available for configuring the recommended roles:

- **Primary Administrator rights profile –** For configuring a role that can perform all administrative tasks, can grant rights to others, and can edit rights that are associated with administrative roles. A user in this role can assign this role to other users, and can grant rights to other users.
- **System Administrator rights profile –** For configuring a role that can perform most administrative tasks that are not related to security. For example, the System Administrator can add new user accounts, but cannot set passwords or grant rights to other users.
- **Operator rights profile –** For configuring a role that can perform simple administrative tasks, such as media backup and printer maintenance.

To further examine rights profiles, read one of the following:

- In the `/etc/security` directory, read the contents of the `prof_attr` database and the `exec_attr` database.
- In the Solaris Management Console, use the Rights tool to display the contents of a rights profile.
- In this book, refer to "Contents of Rights Profiles" on page 223 for summaries of some typical rights profiles.

5 **Decide if any additional roles or rights profiles are appropriate for your organization.**

Look for other applications or families of applications at your site that might benefit from restricted access. Applications that affect security, that can cause denial-of-service problems, or that require special administrator training are good candidates for RBAC. You can customize roles and rights profiles to handle the security requirements of your organization.

a. **Determine which commands are needed for the new task.**

b. **Decide which rights profile is appropriate for this task.**

Check if an existing rights profile can handle this task or if a separate rights profile needs to be created.

c. **Determine which role is appropriate for this rights profile.**

Decide if the rights profile for this task should be assigned to an existing role or if a new role should be created. If you use an existing role, check that the other rights profiles are appropriate for users who are assigned to this role.

6 **Decide which users should be assigned to the available roles.**

According to the principle of least privilege, you should assign users to roles that are appropriate to their level of trust. When you prevent users from access to tasks that the users do not need to perform, you reduce potential problems.

▼ How to Create and Assign a Role By Using the GUI

To create a new role, you can be superuser, or you can use the Primary Administrator role. In this procedure, the creator of the new role has assumed the role of Primary Administrator.

Before You Begin

- You have already created users who can assume a role at your site. If the users are not yet created, create them by following the instructions in “Using the Solaris Management Tools With RBAC (Task Map)” in *System Administration Guide: Basic Administration*.
- You have been assigned the Primary Administrator role by following the procedures in “Using the Solaris Management Tools With RBAC (Task Map)” in *System Administration Guide: Basic Administration*.

1 **Start the Solaris Management Console.**

```
# /usr/sbin/smc &
```

For login instructions, see “How to Assume a Role in the Solaris Management Console” on page 211.

2 **Click the Administrative Roles icon.**

3 **Select Add Administrative Role from the Action menu.**

4. **Create a new role by filling in the fields in the series of dialog boxes.**

   For possible roles, see Example 9–1 to Example 9–4.

Tip – All tools in the Solaris Management Console display information in the bottom section of the page or at the left side of a wizard panel. Choose Help at any time to find additional information about performing tasks in this interface.

---

5. **Assign the role to a user.**

Tip – After filling in the properties of the role, the last dialog box prompts you for a user for the role.

---

6. **In a terminal window, restart the name service cache daemon.**

```
# svcadm restart system/name-service-cache
```

For more information, see the `svcadm`(1M) and `nscd`(1M) man pages.

Example 9–1 Creating a Role for the System Administrator Rights Profile

In this example, the new role can do system administration tasks that are not connected to security. The role is created by performing the preceding procedure with the following parameters:

- Role name: `sysadmin`
- Role full name: `System Administrator`
- Role description: `Performs non-security admin tasks`
- Rights profile: `System Administrator`

  This rights profile is at the top of the list of profiles that are included in the role.

Example 9–2 Creating a Role for the Operator Rights Profile

The Operator rights profile can manage printers and back up the system to offline media. You might want to assign the role to one user on each shift. To do so, you would select the role mailing list option in the Step 1: Enter a Role Name dialog box. The role is created by performing the preceding procedure with the following parameters:

- Role name: `operadm`
- Role full name: `Operator`
- Role description: `Backup operator`
- Rights profile: `Operator`

  This rights profile must be at the top of the list of profiles that are included in the role.

Example 9–3 Creating a Role for a Security-Related Rights Profile

By default, the only rights profile that contains security-related commands and rights is the Primary Administrator profile. If you want to create a role that is not as powerful as Primary Administrator, but can handle some security-related tasks, you must create the role.

In the following example, the role protects devices. The role is created by performing the preceding procedure with the following parameters:

- Role name: `devicesec`
- Role full name: `Device Security`
- Role description: `Configures Devices`
- Rights profile: `Device Security`

In the following example, the role secures systems and hosts on the network. The role is created by performing the preceding procedure with the following parameters:

- Role name: `netsec`
- Role full name: `Network Security`
- Role description: `Handles IPsec, IKE, and SSH`
- Rights profile: `Network Security`

Example 9–4 Creating a Role for a Rights Profile With Limited Scope

A number of rights profiles are of limited scope. In this example, the sole task of the role is to manage DHCP. The role is created by performing the preceding procedure with the following parameters:

- Role name: `dhcpmgt`
- Role full name: `DHCP Management`
- Role description: `Manages Dynamic Host Config Protocol`
- Rights profile: `DHCP Management`

Example 9–5 Modifying a User's Role Assignment

In this example, a role is added to an existing user. The user's role assignment is modified by clicking the User Accounts icon in the Users tool in the Solaris Management Console, double-clicking the user, and following the online help to add a role to the user's capabilities.

Troubleshooting Check the following if the role does not have the capabilities that it should:

- Are the role's rights profiles listed in the GUI from most to least powerful?

  For example, if the `All` rights profile is at the top of the list, then no commands are run with security attributes. A profile that contains commands with security attributes must precede the `All` rights profile in the list.
- Do the commands in the role's rights profiles have the appropriate security attributes?

  For example, when the policy is `suser`, some commands require `uid=0` rather than `euid=0`.
- Is the rights profile defined in the appropriate name service scope? Is the role operating in the name service scope where the rights profile is defined?
- Has the name service cache, `svc:/system/name-service-cache`, been restarted?

  The `nscd` daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the name service with current data.

▼ How to Create a Role From the Command Line

The Solaris Management Console GUI is the preferred method for managing RBAC. To use the GUI, see "How to Create and Assign a Role By Using the GUI" on page 199. You can also use the command-line interfaces, as described in this procedure.

Note – Do not attempt to administer RBAC with the command line and the graphical user interface at the same time. Conflicting changes could be made to the configuration, and the behavior would be unpredictable. You can use both tools to administer RBAC, but you cannot use both concurrently.

Before You Begin To create a role, you must either assume a role that includes the Primary Administrator rights profile, or switch to the user `root`.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Choose one of the following commands to create a role on the command line.**

- **For roles in the local name service scope, use the `roleadd` command.**

Note – The `roleadd` command is more limited than the Solaris Management Console GUI or command-line interfaces. After running the `roleadd` command, you must run the `usermod` command to assign the role to a user. And, the user then must set the password for the role, as shown in "How to Assign a Role to a Local User" on page 205.

```
# roleadd -c comment \
-g group -m homedir -u UID -s shell \
-P profile rolename
```

`-c` *comment*	Is a comment that describes *rolename*.
`-g` *group*	Is the group assignment for *rolename*.
`-m` *homedir*	Is the path to the home directory for *rolename*.
`-u` *UID*	Is the UID for *rolename*.
`-s` *shell*	Is the login shell for *rolename*. This shell must be a profile shell.
`-P` *profile*	Is one or more rights profiles for *rolename*.
rolename	Is the name of the new local role.

- **Use the `smrole add` command.**

  This command creates a role in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

```
$ /usr/sadm/bin/smrole -D domain-name \
-r admin-role -l <Type admin-role password> \
add -- -n rolename -a rolename -d directory\
-F full-description -p profile
```

`-D` *domain-name*	Is the name of the domain that you want to manage.
`-r` *admin-role*	Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
`-l`	Is the prompt for the password of *admin-role*.
`--`	Is the required separator between authentication options and subcommand options.
`-n` *rolename*	Is the name of the new role.
`-c` *comment*	Is the comment that describes the capabilities of the role.
`-a` *username*	Is the name of the user who can assume *rolename*.

`-d` *directory*	Is the home directory for *rolename*.
`-F` *full-description*	Is the full description for *rolename*. This description is displayed in the Solaris Management Console GUI.
`-p` *profile*	Is a rights profile that is included in the capabilities of *rolename*. This option gives commands with administrative capabilities to the role. You can specify multiple `-p` *profile* options.

3 **To put the changes into effect, see "How to Assign a Role to a Local User" on page 205.**

Example 9–6 Creating a Custom Operator Role by Using the `smrole` Command

The `smrole` command specifies a new role and its attributes in a name service. In the following example, the Primary Administrator creates a new version of the Operator role. The role includes the standard Operator rights profile as well as the Media Restore rights profile. Note that the command prompts you for a password for the new role.

```
% su - primaryadm
Password:       <Type primaryadm password>
$ /usr/sadm/bin/smrole add -H myHost -- -c "Backup and Restore Operator" \
-n operadm2 -a janedoe -d /export/home/operadm \
-F "Backup/Restore Operator" -p "Operator" -p "Media Restore"
Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::    <Type operadm2 password>

$ svcadm restart system/name-service-cache
```

The `smrole` command with the `list` subcommand is used to display the new role:

```
$ /usr/sadm/bin/smrole list --
Authenticating as user: primaryadm

Type /? for help, pressing <enter> accepts the default denoted by [ ]
Please enter a string value for: password ::     <Type primaryadm password>

Loading Tool: com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost
Login to myHost as user primaryadm was successful.
```

```
Download of com.sun.admin.usermgr.cli.role.UserMgrRoleCli from myHost was successful.
root                      0            Superuser
primaryadm              100            Most powerful role
sysadmin                101            Performs non-security admin tasks
operadm                 102            Backup Operator
operadm2                103            Backup/Restore Operator
```

▼ How to Assign a Role to a Local User

This procedure assigns a local role to a local user, restarts the name cache daemon, and then shows how the user can assume the role.

To assign a role to a user in a distributed name service, see "How to Create a Role From the Command Line" on page 202 and "How to Change the Properties of a Role" on page 213.

Before You Begin You have added a local role, as described in "How to Create a Role From the Command Line" on page 202. You must have assumed the role of Primary Administrator or have switched to superuser.

1. **Assign the role to a local user.**

   If you added a local role with the `roleadd` command, this step is required. This step is optional when you use the `smrole` command and the Solaris Management Console to create a role.

   ```
   # usermod -u UID -R rolename
   ```

`-u` *UID*	Is the UID of the user.
`-R` *rolename*	Is the role that is being assigned to the user.

2. **To put the changes into effect, restart the name service cache daemon.**

   ```
   # svcadm restart system/name-service-cache
   ```

   If you added a role with a Solaris Management Console interface, go to "Using Roles (Task Map)" on page 209. Otherwise, continue with the next step.

3. **(Optional) To unlock the role account, the user must create a password.**

   If you added a local role with the `roleadd` command, this step is required.

   ```
   % su - rolename
   Password:       <Type rolename password>
   Confirm Password:      <Retype rolename password>
   $
   ```

Example 9–7 Creating and Assigning a Local Role From the Command Line

In this example, a role is created to administer the Solaris Cryptographic Framework. The Crypto Management rights profile contains the `cryptoadm` command for administering hardware and software cryptographic services on a local system.

```
# roleadd -c "Cryptographic Services manager" \
-g 14 -m /export/home/cryptoadm -u 104 -s pfksh \
-P "Crypto Management" cryptomgt
# usermod -u 1111 -R cryptomgt
# svcadm restart system/name-service-cache
% su - cryptomgt
Password:       <Type cryptomgt password>
Confirm Password:       <Retype cryptomgt password>
$ /usr/ucb/whoami
cryptomgt
$
```

For information about the Solaris Cryptographic Framework, see Chapter 13, "Solaris Cryptographic Framework (Overview)." To administer the framework, see "Administering the Cryptographic Framework (Task Map)" on page 275.

▼ How to Audit Roles

The actions that a role performs can be audited. Included in the audit record is the login name of the user who assumed the role, the role name, and the action that the role performed. The `6180:AUE_prof_cmd:profile command:ua,as` audit event collects the information. By preselecting the `as` class or the `ua` class, you can audit role actions.

1 **Plan for auditing and edit the audit configuration files.**

For more information, see "Solaris Auditing (Task Map)" on page 569.

2 **Include the `ua` class or the `as` class in the `flags` line of the `audit_control` file.**

```
# audit_control file
dir:/var/audit
flags:lo,as
minfree:20
naflags:lo
```

The `ua` class and the `as` class include other audit events. To see the audit events that are included in a class, read the `audit_event` file. You can also use the `bsmrecord` command, as shown in Example 30–25.

3 **Finish configuring the auditing service, then enable auditing.**

For more information, see "Configuring and Enabling the Auditing Service (Tasks)" on page 580.

▼ How to Make `root` User Into a Role

This procedure shows how to change `root` from a login user to a role. When you complete this procedure, you can no longer log in to the system as `root`, except in single-user mode. You can `su` to `root` if the `root` role has been assigned to you.

By changing the `root` user into a role, you prevent anonymous `root` login. Because a user must log in and *then* assume the `root` role, the user's login ID is provided to the auditing service and is in the `sulog` file.

Before You Begin If you change the `root` user into a role without assigning the role to a valid user or without a currently existing role that is equivalent to the `root` user, no one can become superuser.

- For safety, at least one local user should be assigned the `root` role.
- You cannot perform this procedure when you are logged in as `root`. You must log in as yourself, then `su` to `root`.

1 **As an ordinary user, log in to the target host.**

2 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

3 **Create a local user who can assume the `root` role.**

```
$ useradd -c comment -d homedir username
```

`-c` *comment*	Is the comment that describes the user.
`-d` *homedir*	Is the home directory of the user. This directory should be on the local system.
username	Is the name of the new local user.

```
# useradd -c "Local administrative user" -d /export/home1 admuser
```

4 **Give the user a password.**

```
# passwd -r files admuser
New Password:        <Type password>
Re-enter new Password:     <Retype password>
```

```
passwd: password successfully changed for admuser
#
```

5. **Make sure that you are not logged in as `root`.**

   ```
   # who
   jdoe    console      May 24 13:51    (:0)
   jdoe    pts/5        May 24 13:51    (:0.0)
   jdoe    pts/4        May 24 13:51    (:0.0)
   jdoe    pts/10       May 24 13:51    (:0.0)
   ```

6. **Change `root` user into a role.**

   ```
   # usermod -K type=role root
   ```

7. **Verify that `root` is a role.**

   The `root` entry in the `user_attr` file should appear similar to the following:

   ```
   root::::type=role;auths=solaris.*,solaris.grant;profiles=Web Console
   Management,All;lock_after_retries=no
   ```

8. **Assign the `root` role to the local administrative user.**

   ```
   # usermod -R root admuser
   ```

9. **Configure the name service to return in case of failure.**

   a. **Open a new terminal window and assume the `root` role.**

   ```
   % whoami
   jdoe
   % su - admuser
   Enter password:        <Type admuser password>
   % roles
   root
   % su - root
   Enter password:        <Type root password>
   #
   ```

   b. **Edit the `nsswitch.conf` file.**

   For example, the following entries in the `nsswitch.conf` file would enable the name service to return.

   ```
   passwd:  files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
   group:  files nis [TRYAGAIN=0 UNAVAIL=return NOTFOUND=return]
   ```

10. **Assign the `root` role to selected user accounts in the name service.**

    For the procedure, see "How to Change the RBAC Properties of a User" on page 218.

Using Roles (Task Map)

The following task map points to procedures for using your role after roles have been assigned.

Task	Description	For Instructions
Use the Solaris Management Console	Authenticate yourself as a role to perform administrative tasks in the Solaris Management Console.	"How to Assume a Role in the Solaris Management Console" on page 211
Assume a role in a terminal window	Perform command-line administrative tasks in a profile shell.	"How to Assume a Role in a Terminal Window" on page 209

Using Roles

After you have set up roles with default Solaris rights profiles, and assigned the roles to users, the roles can be used. A role can be assumed on the command line. In the Solaris Management Console, a role can also be used for administering the system locally and over the network.

▼ How to Assume a Role in a Terminal Window

Before You Begin The role must already be assigned to you. The name service must be updated with that information.

1. **In a terminal window, determine which roles you can assume.**

   ```
   % roles
   Comma-separated list of role names is displayed
   ```

2. **Use the `su` command to assume a role.**

   ```
   % su - rolename
   Password:       <Type rolename password>
   $
   ```

   The `su -` *rolename* command changes the shell to a profile shell for the role. A profile shell recognizes security attributes (authorizations, privileges, and set ID bits).

3. **Verify that you are now in a role.**

   ```
   $ /usr/ucb/whoami
   rolename
   ```

   You can now perform role tasks in this terminal window.

4 **(Optional) View the capabilities of your role.**

For the procedure, see "How to Determine the Privileged Commands That a Role Can Run" on page 251.

Example 9–8 Assuming the Primary Administrator Role

In the following example, the user assumes the role of Primary Administrator. In the default configuration, this role is equivalent to superuser. The role then checks to see which privileges are available to any command that is typed in the profile shell for the role.

```
% roles
sysadmin,oper,primaryadm
% su - primaryadm
Password:       <Type primaryadm password>
$ /usr/ucb/whoami       Prompt has changed to role prompt
primaryadm
$ ppriv $$
1200:   pfksh
flags = <none>
        E (Effective): all
        I (Inheritable): basic
        P (Permitted): all
        L (Limit): all
```

For information about privileges, see "Privileges (Overview)" on page 185.

Example 9–9 Assuming the `root` Role

In the following example, the user assumes the `root` role. The role was created in "How to Make `root` User Into a Role" on page 207

```
% roles
root
% su - root
Password:       <Type root password>
# /usr/ucb/whoami       Prompt has changed to role prompt
root
$ ppriv $$
1200:   pfksh
flags = <none>
        E: all
        I: basic
        P: all
        L: all
```

For information about privileges, see "Privileges (Overview)" on page 185.

Example 9–10 Assuming the System Administrator Role

In the following example, the user assumes the role of System Administrator. In contrast to the Primary Administrator role, the System Administrator has the basic set of privileges in its effective set.

```
% roles
sysadmin,oper,primaryadm
% su - sysadmin
Password:       <Type sysadmin password>
$ /usr/ucb/whoami       Prompt has changed to role prompt
sysadmin
$ ppriv $$
1200:   pfksh
flags = <none>
        E: basic
        I: basic
        P: basic
        L: all
```

For information about privileges, see "Privileges (Overview)" on page 185. For a short description of the capabilities of the role, see "System Administrator Rights Profile" on page 224.

▼ How to Assume a Role in the Solaris Management Console

To change information in the Solaris Management Console GUI requires administrative capabilities. A role gives you administrative capabilities. If you want to view information, you must have the `solaris.admin.usermgr.read` authorization. The Basic Solaris User rights profile includes this authorization.

Before You Begin An administrative role that can change the properties of users or roles must have already been assigned to you. For example, the Primary Administrator role can change the properties of users or roles.

1. **Start the Solaris Management Console.**

```
% /usr/sbin/smc &
```

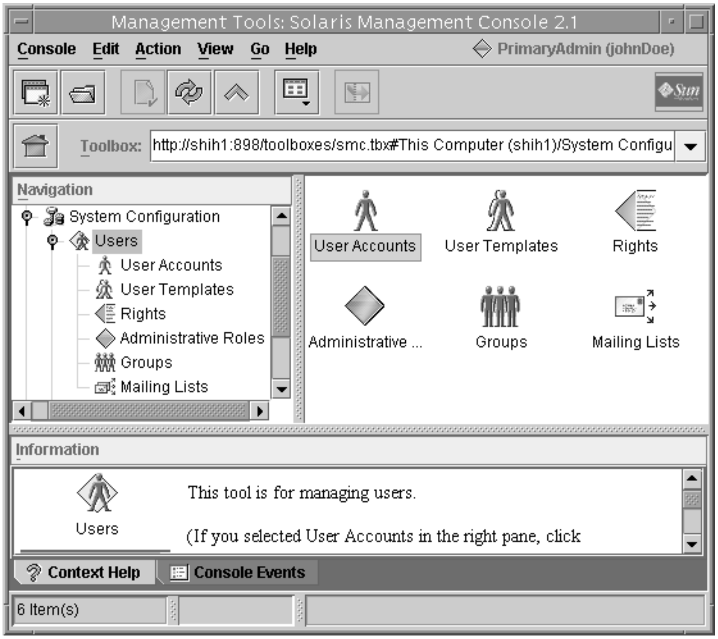
For detailed instructions, see "Using the Solaris Management Tools With RBAC (Task Map)" in *System Administration Guide: Basic Administration*.

2 **Select the toolbox for your task.**

Navigate to the toolbox that contains the tool or collection in the appropriate name service scope and click the icon. The scopes are files (local), NIS, NIS+, and LDAP. If the appropriate toolbox is not displayed in the navigation pane, choose Open Toolbox from the Console menu and load the relevant toolbox.

3 **Select the tool that you want to use.**

Navigate to the tool or collection and click the icon. The tools for managing the RBAC elements are in the Users tool, as shown in the following figure.


Management Tools: Solaris Management Console 2.1
Console
Edit
Action
View
Go
Help
PrimaryAdmin (johnDoe)
Sun
Navigation
System Configuration
Users
User Accounts
User Templates
Rights
Administrative Roles
Groups
Mailing Lists
User Accounts
User Templates
Rights
Administrative ...
Groups
Mailing Lists
Information
Users
Context Help
Console Events
6 Item(s)

Toolbox: http://shih1:898/toolboxes/smc.tbx#This Computer (shih1)/System Configu

This tool is for managing users.

(If you selected User Accounts in the right pane, click

4 **Type your user name and password in the Login: User Name dialog box.**

5 **Authenticate yourself in the Login: Role dialog box.**

The Role option menu in the dialog box displays the roles that are assigned to you. Choose a role and type the role password.

Managing RBAC (Task Map)

The following task map points to procedures for customizing role-based access control (RBAC) after RBAC has been initially implemented.

Task	Description	For Instructions
Modify the properties of a role	Modifies the capabilities (privileges, privileged commands, profiles, or authorizations) of a role.	"How to Change the Properties of a Role" on page 213
Create or change rights profiles	Creates a rights profile. Or modifies the authorizations, privileged commands, or supplementary rights profiles in a rights profile.	"How to Create or Change a Rights Profile" on page 215
Change a user's administrative capabilities	Adds a role, a rights profile, an authorization, or privileges to an ordinary user.	"How to Change the RBAC Properties of a User" on page 218
Secure legacy applications	Turns on the set ID permissions for legacy applications. Scripts can contain commands with set IDs. Legacy applications can check for authorizations, if appropriate.	"How to Add RBAC Properties to Legacy Applications" on page 219

These procedures manage the elements that are used in RBAC. For user management procedures, refer to Chapter 5, "Managing User Accounts and Groups (Tasks)," in *System Administration Guide: Basic Administration*.

Managing RBAC

The Solaris Management Console GUI is the preferred method for managing RBAC.

Note – Do not attempt to administer RBAC with the command line and the graphical user interface at the same time. Conflicting changes could be made to the configuration, and the behavior would be unpredictable. Both tools can administer RBAC, but you cannot use both tools concurrently.

▼ How to Change the Properties of a Role

Before You Begin You must have assumed the role of Primary Administrator or have switched to superuser to change the properties of a role. Role properties include password, rights profiles, and authorizations.

- **Use one of the following methods to change the properties of a role.**
  - **Use the Users tool in the Solaris Management Console.**

    To start the console, see "How to Assume a Role in the Solaris Management Console" on page 211. Follow the instructions in the left-hand pane to modify a role in Administrative Roles. For more extensive information, see the online help.

  - **Use the `rolemod` command.**

    This command modifies the attributes of a role that is defined in the local name service.

    ```
    $ rolemod -c comment -P profile-list rolename
    ```

`-c` *comment*	Is the new comment that describes the capabilities of the role.
`-P` *profile-list*	Is the list of the profiles that are included in the role. This list replaces the current list of profiles.
rolename	Is the name of an existing, local role that you want to modify.

    For more command options, see the `rolemod`(1M) man page.

  - **Use the `smrole` command with the `modify` subcommand.**

    This command modifies the attributes of a role in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

    ```
    $ /usr/sadm/bin/smrole -D domain-name \
    -r admin-role -l <Type admin-role password> \
    modify -- -n rolename  -r username -u username
    ```

`-D` *domain-name*	Is the name of the domain that you want to manage.
`-r` *admin-role*	Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
`-l`	Is the prompt for the password of *admin-role*.
`--`	Is the required separator between authentication options and subcommand options.
`-n` *rolename*	Is the name of the new role.
`-r` *username*	Is the name of the user who can no longer assume *rolename*.
`-u` *username*	Is the name of the user who can now assume *rolename*.

    For more command options, see the `smrole`(1M) man page.

Example 9–11 Changing a Local Role's Properties With the `rolemod` Command

In this example, the `operadm` role is modified to include the Media Restore rights profile.

```
$ rolemod -c "Handles printers, backup, AND restore" \
-P "Printer Management,Media Backup,Media Restore,All" operadm
```

Example 9–12 Changing a Local Role's Properties With the `smrole modify` Command

In the following example, the `operadm` role is modified to add the Media Restore rights profile.

```
$ /usr/sadm/bin/smrole -r primaryadm -l <Type primaryadm password> \
modify -- -n operadm -c "Handles printers, backup, AND restore" \
-p "Media Restore"
```

Example 9–13 Changing a Role in a Domain With the `smrole modify` Command

In the following example, the `clockmgr` role is changed. The NIS user whose ID is `108` can no longer assume the role. The NIS user whose ID is `110` can assume the role `clockmgr`.

```
$ /usr/sadm/bin/smrole -D nis:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n clockmgr -r 108 -u 110
```

▼ How to Create or Change a Rights Profile

A rights profile is a property of a role. You should create or change a rights profile when the `prof_attr` database does not contain a rights profile that fulfills your needs. To learn more about rights profiles, see "RBAC Rights Profiles" on page 183.

Before You Begin To create or change a rights profile, you must have assumed the role of Primary Administrator or have switched to superuser.

● **Use one of the following methods to change the properties of a role.**

- **Use the Users tool in the Solaris Management Console.**

  To start the console, see "How to Assume a Role in the Solaris Management Console" on page 211. Follow the instructions in the left-hand pane to create or change a rights profile in Rights. For more extensive information, see the online help.

- **Use the `smprofile` command.**

  This command enables you to add, modify, list, or delete a rights profile. The command works on files, and in a distributed name service, such as NIS, NIS+, or LDAP. The `smprofile` command runs as a client of the Solaris Management Console server.

  ```
  $ /usr/sadm/bin/smprofile -D domain-name \
  -r admin-role -l <Type admin-role password> \
  add | modify -- -n profile-name \
  -d description  -m help-file -p supplementary-profile
  ```

`-D` *domain-name*	Is the name of the domain that you want to manage.
`-r` *admin-role*	Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
`-l`	Is the prompt for the password of *admin-role*.
`--`	Is the required separator between authentication options and subcommand options.
`-n` *profile-name*	Is the name of the new profile.
`-d` *description*	Is a short description of the profile.
`-m` *help-file*	Is the name of the HTML help file that you have created and placed in the `/usr/lib/help/profiles/locale/C` directory.
`-p` *supplementary-profile*	Is the name of an existing rights profile that is included in this rights profile. You can specify multiple `-p` *supplementary-profile* options.

  For more command options, see the `smprofile`(1M) man page.

Example 9–14 Modifying a Rights Profile From the Command Line

In the following example, the Network Management rights profile is made a supplementary profile of the Network Security rights profile. The role that contains the Network Security profile can now configure the network and hosts, as well has run security-relevant commands.

```
$ /usr/sadm/bin/smprofile -D nisplus:/example.host/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n "Network Security" \
-d "Manage network and host configuration and security" \
-m RtNetConfSec.html -p "Network Management"
```

The administrator created a new help file, `RtNetConfSec.html`, and placed it in the `/usr/lib/help/profiles/locale/C` directory, before running this command.

Example 9–15 Creating a New Rights Profile With the Rights Tool

The following table shows sample data for a hypothetical rights profile that is called "Build Administrator". This rights profile includes the commands in the subdirectory `/usr/local/swctrl/bin`. These commands have an effective UID of `0`. The Build Administrator rights profile would be useful for administrators who manage the builds and versioning for software development.

Tab	Field	Example
General	Name	Build Administrator
	Description	For managing software builds and versioning.
	Help File Name	`BuildAdmin.html`
Commands	Add Directory	Click Add Directory, type `/usr/local/swctrl/bin` in the dialog box, and click OK.
	Commands Denied / Commands Permitted	Move `/usr/local/swctrl/bin` to the Commands Permitted column.
	Set Security Attributes	Select `/usr/local/swctrl/bin`, click Set Security Attributes, and set Effective UID = `root`.
Authorizations	Authorizations Excluded / Authorizations Included	*No authorizations.*
Supplementary Rights	Rights Excluded / Rights Included	*No supplementary rights profiles.*

Troubleshooting Check the following if the rights profile does not provide the role with the capabilities that you expect:

- Are the rights profiles for the role listed in the GUI from most to least powerful?

  For example, if the `All` rights profile is at the top of the list, then no commands are run with security attributes. A profile that contains commands with security attributes must precede the `All` rights profile in the list.

- Is a command listed more than once in the role's rights profiles? If so, does the first instance of the command have all the security attributes that are required?

  For example, a command can require privileges for particular options to the command. For the options that require privileges to succeed, the first instance of the command in the highest rights profile in the list must have the assigned privileges.

- Do the commands in the role's rights profiles have the appropriate security attributes?

For example, when the policy is `suser`, some commands require `uid=0` rather than `euid=0` to succeed.

- Has the name service cache, `svc:/system/name-service-cache`, been restarted?

  The `nscd` daemon can have a lengthy time-to-live interval. By restarting the daemon, you update the name service with current data.

▼ How to Change the RBAC Properties of a User

User properties include password, rights profiles, and authorizations. The most secure method of giving a user administrative capabilities is to assign a role to the user. For a discussion, see "Security Considerations When Directly Assigning Security Attributes" on page 184.

Before You Begin You must have assumed the role of Primary Administrator or have switched to superuser to change the properties of a user.

● **Use one of the following methods to change the RBAC properties of a user.**

- **Use the Users tool in the Solaris Management Console.**

  To start the console, see "How to Assume a Role in the Solaris Management Console" on page 211. Follow the instructions in the left-hand pane to modify a user in User Accounts. For more extensive information, see the online help.

  Tip – It is not good practice to assign authorizations, privileges, or rights profiles directly to users. The preferred approach is to assign a role to users. Users then assume a role to perform privileged operations.

- **Use the `usermod` command.**

  This command modifies the attributes of a user that is defined in the local name service.

  ```
  $ usermod -R rolename username
  ```

`-R` *rolename*	Is the name of an existing local role.
username	Is the name of an existing, local user that you want to modify.

  For more command options, see the `usermod`(1M) man page.

- **Use the `smuser` command with the `modify` subcommand.**

  This command modifies the attributes of a user in a distributed name service, such as NIS, NIS+, or LDAP. This command runs as a client of the Solaris Management Console server.

  ```
  $ /usr/sadm/bin/smuser -D domain-name \
  -r admin-role -l <Type admin-role password> \
  modify -- -n username -a rolename
  ```

`-D` *domain-name*	Is the name of the domain that you want to manage.
`-r` *admin-role*	Is the name of the administrative role that can modify the role. The administrative role must have the `solaris.role.assign` authorization. If you are modifying a role that you have assumed, the role must have the `solaris.role.delegate` authorization.
`-l`	Is the prompt for the password of *admin-role*.
`--`	Is the required separator between authentication options and subcommand options.
`-n` *username*	Is the name of the user who is being assigned *rolename*.
`-a` *rolename*	Is the name of the role that you are assigning to *username*. You can specify multiple `-a` *rolename*options.

For more command options, see the `smuser`(1M) man page.

Example 9–16 Modifying a Local User's RBAC Properties From the Command Line

In this example, the user `jdoe` can now assume the role of System Administrator.

```
$ usermod -R sysadmin jdoe
```

Example 9–17 Modifying a User's RBAC Properties With the `smuser` Command

In this example, the user `jdoe` is assigned two roles, System Administrator and Operator. Because the user and the roles are defined locally, the `-D` option is not necessary.

```
$ /usr/sadm/bin/smuser -r primaryadm -l <Type primaryadm password> \
modify -- -n jdoe -a sysadmin -a operadm
```

In the following example, the user is defined in the NIS name service. Therefore, the `-D` option is required. Two roles are defined in the name service. One role, `root`, is defined locally.

```
$ /usr/sadm/bin/smuser -D nis:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
modify -- -n jdoe -a sysadmin -a operadm -a root
```

▼ How to Add RBAC Properties to Legacy Applications

A legacy application is a command or set of commands. The security attributes are set for each command in a rights profile. The rights profile is then included in a role. A user who assumes the role can run the legacy application with the security attributes.

To add legacy applications to the Solaris Management Console, see “Adding Tools to the Solaris Management Console” in *System Administration Guide: Basic Administration*.

Before You Begin You must have assumed the role of Primary Administrator or have switched to superuser to change the security attributes of a command in a rights profile.

1. **Use the Users tool in the Solaris Management Console.**

   To start the console, see “How to Assume a Role in the Solaris Management Console” on page 211. Follow the instructions in the left-hand pane to modify a rights profile in Rights. For more extensive information, see the online help.

2. **Add security attributes to the commands that implement the legacy application.**

   You add security attributes to a legacy application in the same way that you would for any command. You must add the command with security attributes to a rights profile. For a legacy command, give the command `euid=0` or `uid=0` security attributes. For details of the procedure, see “How to Create or Change a Rights Profile” on page 215.

3. **After adding the legacy application to a rights profile, include the rights profile in a role's list of profiles.**

   To add a rights profile to a role, see “How to Change the Properties of a Role” on page 213.

Example 9–18 Adding Security Attributes to Commands in a Script

If a command in a script needs to have the `setuid` bit or `setgid` bit set to succeed, the script executable *and* the command must have the security attributes added in a rights profile. Then, the rights profile is included in a role, and the role is assigned to a user. When the user assumes the role and executes the script, the command runs with the security attributes.

To add security attributes to a command or shell script, see “How to Create or Change a Rights Profile” on page 215.

Example 9–19 Checking for Authorizations in a Script or Program

To have a script for authorizations, you need to add a test that is based on the `auths` command. For detailed information about this command, see the `auths`(1) man page.

For example, the following line tests if the user has the authorization that is supplied as the `$1` argument:

```
if [ ‘/usr/bin/auths|/usr/xpg4/bin/grep $1‘ ]; then
        echo Auth granted
else
```

```
        echo Auth denied
fi
```

To be more complete, the test should include logic that checks for other authorizations that use wildcards. For example, to test if the user has the `solaris.admin.usermgr.write` authorization, you would need to check for the following strings:

- `solaris.admin.usermgr.write`
- `solaris.admin.usermgr.*`
- `solaris.admin.*`
- `solaris.*`

If you are writing a program, use the function `getauthattr()` to test for the authorization.

◆ ◆ ◆

CHAPTER 10

Role-Based Access Control (Reference)

This chapter provides reference material about RBAC. The following is a list of the reference information in this chapter:

- "Contents of Rights Profiles" on page 223
- "Authorization Naming and Delegation" on page 228
- "Databases That Support RBAC" on page 229
- "RBAC Commands" on page 236

For information on using RBAC, see Chapter 9, "Using Role-Based Access Control (Tasks)." For overview information, see "Role-Based Access Control (Overview)" on page 175.

Contents of Rights Profiles

This section describes some typical rights profiles. Rights profiles can include authorizations, commands with security attributes, and supplementary rights profiles. The rights profiles are listed from most to least powerful. For suggestions on how to distribute rights profiles to roles at your site, see "How to Plan Your RBAC Implementation" on page 197.

- **Primary Administrator rights profile –** Provides the capabilities of superuser in one profile.
- **System Administrator rights profile –** Provides a profile that can do most tasks that are not connected with security. This profile includes several other profiles to create a powerful role.
- **Operator rights profile –** Provides limited capabilities to manage files and offline media. This profile includes supplementary rights profiles to create a simple role.
- **Printer Management rights profile –** Provides a limited number of commands and authorizations to handle printing. This profile is one of several profiles that cover a single area of administration.
- **Basic Solaris User rights profile –** Enables users to use the system within the bounds of security policy. This profile is listed by default in the `policy.conf` file.

- **All rights profile –** For roles, provides access to commands that do not have security attributes.

Each rights profile has an associated help file. The help files are in HTML and are customizable. The files reside in the `/usr/lib/help/auths/locale/C` directory.

Primary Administrator Rights Profile

The Primary Administrator rights profile is assigned to the most powerful role on the system. The role that includes the Primary Administrator rights profile has superuser capabilities.

- The `solaris.*` authorization effectively assigns all of the authorizations that are provided by the Solaris software.
- The `solaris.grant` authorization lets a role assign any authorization to any rights profile, role, or user.
- The command assignment `*:uid=0;gid=0` provides the ability to run any command with `UID=0` and `GID=0`.

You can customize the help file `RtPriAdmin.html` for your site, if necessary. Help files are stored in the `/usr/lib/help/auths/locale/C` directory.

Note also that if the Primary Administrator rights profile is not consistent with a site's security policy, the profile can be modified or not assigned at all. However, the security capabilities in the Primary Administrator rights profile would need to be handled in one or more other rights profiles. Those other rights profiles would then be assigned to roles.

TABLE 10–1 Contents of Primary Administrator Rights Profile

Purpose	Contents
To perform all administrative tasks	**Commands**: `*:uid=0;gid=0` **Authorizations**: `solaris.*`, `solaris.grant` **Help File**: `RtPriAdmin.html`

System Administrator Rights Profile

The System Administrator rights profile is intended for the System Administrator role. Because the System Administrator does not have the broad capabilities of the Primary Administrator, no wildcards are used. Instead, this profile is a set of discrete, supplementary administrative rights profiles that do not deal with security. The commands with security attributes from one of the supplementary rights profiles are shown.

Note that the All rights profile is assigned at the end of the list of supplementary rights profiles.

TABLE 10–2 Contents of System Administrator Rights Profile

Purpose	Contents
To perform most nonsecurity administrative tasks	**Supplementary rights profiles**: Audit Review, Printer Management, Cron Management, Device Management, File System Management, Mail Management, Maintenance and Repair, Media Backup, Media Restore, Name Service Management, Network Management, Object Access Management, Process Management, Software Installation, User Management, All **Help File**: `RtSysAdmin.html`
Commands from one of the supplementary profiles	**Object Access Management rights profile**, `solaris` policy: `/usr/bin/chgrp:privs=file_chown`, `/usr/bin/chmod:privs=file_chown`, `/usr/bin/chown:privs=file_chown`, `/usr/bin/setfacl:privs=file_chown` `suser` policy: `/usr/bin/chgrp:euid=0`, `/usr/bin/chmod:euid=0`, `/usr/bin/chown:euid=0`, `/usr/bin/getfacl:euid=0`, `/usr/bin/setfacl:euid=0`

Operator Rights Profile

The Operator rights profile is a less powerful profile that provides the ability to do backups and printer maintenance. The ability to restore files has more security consequences. Therefore, in this profile, the default is to not include the ability to restore files.

TABLE 10–3 Contents of Operator Rights Profile

Purpose	Contents
To perform simple administrative tasks	**Supplementary rights profiles**: Printer Management, Media Backup, All **Help File**: `RtOperator.html`

Printer Management Rights Profile

Printer Management is a typical rights profile that is intended for a specific task area. This profile includes authorizations and commands. The following table shows a partial list of commands.

TABLE 10–4 Contents of Printer Management Rights Profile

Purpose	Contents
To manage printers, daemons, and spooling	**Authorizations**: `solaris.admin.printer.delete`, `solaris.admin.printer.modify`, `solaris.admin.printer.read` **Commands**: `/usr/bin/cancel:euid=lp;uid=lp`, `/usr/bin/lpset:egid=14`, `/usr/bin/lpstat:euid=0`, `/usr/lib/lp/local/lpadmin:uid=lp;gid=8`, `/usr/lib/lp/lpsched:uid=0`, `/usr/sbin/lpadmin:egid=14;uid=lp;gid=8`, `/usr/sbin/lpfilter:euid=lp;uid=lp`, `/usr/ucb/lprm:euid=0` **Help File**: `RtPrntMngmnt.html`

Basic Solaris User Rights Profile

By default, the Basic Solaris User rights profile is assigned automatically to all users through the `policy.conf` file. This profile provides basic authorizations that are useful in normal operations. Note that the convenience that is offered by the Basic Solaris User rights profile must be balanced against site security requirements. Sites that need stricter security might prefer to remove this profile from the `policy.conf` file.

TABLE 10–5 Contents of Basic Solaris User Rights Profile

Purpose	Contents
To automatically assign rights to all users	**Authorizations**: `solaris.profmgr.read`, `solaris.jobs.users`, `solaris.mail.mailq`, `solaris.admin.usermgr.read`, `solaris.admin.logsvc.read`, `solaris.admin.fsmgr.read`, `solaris.admin.serialmgr.read`, `solaris.admin.diskmgr.read`, `solaris.admin.procmgr.user`, `solaris.compsys.read`, `solaris.admin.printer.read`, `solaris.admin.prodreg.read`, `solaris.admin.dcmgr.read`, `solaris.snmp.read`, `solaris.project.read`, `solaris.admin.patchmg.read`, `solaris.network.hosts.read`, `solaris.compsys.read`, `solaris.admin.volmgr.read` **Supplementary rights profiles**: All **Help File**: `RtDefault.html`

All Rights Profile

The All rights profile uses the wildcard to include all commands. This profile provides a role with access to all commands that are not explicitly assigned in other rights profiles. Without the All rights profile or other rights profiles that use wildcards, a role has access to explicitly assigned commands only. Such a limited a set of commands is not very practical.

The All rights profile, if used, should be the final rights profile that is assigned. This last position ensures that explicit security attribute assignments in other rights profiles are not inadvertently overridden.

TABLE 10–6 Contents of All Rights Profile

Purpose	Contents
To execute any command as the user or role	**Commands**: * **Help File**: `RtAll.html`

Order of Rights Profiles

The commands in rights profiles are interpreted in order. The first occurrence of a command is the only version of the command that is used for that role or user. Different rights profiles can include the same command. Therefore, the order of rights profiles in a list of profiles is important. The rights profile with the most capabilities should be listed first.

Rights profiles are listed in the Solaris Management Console GUI and in the `prof_attr` file. In the Solaris Management Console GUI, the rights profile with the most capabilities should be the top profile in a list of assigned rights profiles. In the `prof_attr` file, the rights profile with the most capabilities should be the first in a list of supplementary profiles. This placement ensures that a command with security attributes is listed before that same command without security attributes.

Viewing the Contents of Rights Profiles

The Solaris Management Console Rights tool provides one way of inspecting the contents of the rights profiles.

The `prof_attr` and `exec_attr` files offer a more fragmented view. The `prof_attr` file contains the name of every rights profile that is defined on the system. The file also includes the authorizations and the supplementary rights profiles for each profile. The `exec_attr` file contains the names of rights profiles and their commands with security attributes.

Authorization Naming and Delegation

An RBAC *authorization* is a discrete right that can be granted to a role or a user. Authorizations are checked by RBAC-compliant applications before a user gets access to the application or specific operations within the application. This check replaces the tests in conventional UNIX applications for `UID=0`.

Authorization Naming Conventions

An authorization has a name that is used internally and in files. For example, `solaris.admin.usermgr.pswd` is the name of an authorization. An authorization has a short description, which appears in the graphical user interfaces (GUIs). For example, `Change Passwords` is the description of the `solaris.admin.usermgr.pswd` authorization.

By convention, authorization names consist of the reverse order of the Internet name of the supplier, the subject area, any subareas, and the function. The parts of the authorization name are separated by dots. An example would be `com.xyzcorp.device.access`. Exceptions to this convention are the authorizations from Sun Microsystems, Inc., which use the prefix `solaris` instead of an Internet name. The naming convention enables administrators to apply authorizations in a hierarchical fashion. A wildcard (*) can represent any strings to the right of a dot.

Example of Authorization Granularity

As an example of how authorizations are used, consider the following: A user in the Operator role might be limited to the `solaris.admin.usermgr.read` authorization, which provides read but not write access to user configuration files. The System Administrator role naturally has the `solaris.admin.usermgr.read` and the `solaris.admin.usermgr.write` authorizations for making changes to user files. However, without the `solaris.admin.usermgr.pswd` authorization, the System Administrator cannot change passwords. The Primary Administrator has all three of these authorizations.

The `solaris.admin.usermgr.pswd` authorization is required to make password changes in the Solaris Management Console User tool. This authorization is also required for using the password modification options in the `smuser`, `smmultiuser`, and `smrole` commands.

Delegation Authority in Authorizations

An authorization that ends with the suffix `grant` enables a user or a role to delegate to other users any assigned authorizations that begin with the same prefix.

For example, a role with the authorizations `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.read` can delegate the `solaris.admin.usermgr.read` authorization to another user. A role with the `solaris.admin.usermgr.grant` and `solaris.admin.usermgr.*` authorizations can delegate any of the authorizations with the `solaris.admin.usermgr` prefix to other users.

Databases That Support RBAC

The following four databases store the data for the RBAC elements:

- **Extended user attributes database** (`user_attr`) – Associates users and roles with authorizations and rights
- **Rights profile attributes database** (`prof_attr`) – Defines rights profiles, lists the profiles' assigned authorizations, and identifies the associated help file
- **Authorization attributes database** (`auth_attr`) – Defines authorizations and their attributes, and identifies the associated help file
- **Execution attributes database** (`exec_attr`) – Identifies the commands with security attributes that are assigned to specific rights profiles

The `policy.conf` database contains authorizations,privileges, and rights profiles that are applied to all users. For more information, see "`policy.conf` File" on page 235.

RBAC Database Relationships

Each RBAC database uses a *key*=*value* syntax for storing attributes. This method accommodates future expansion of the databases. The method also enables a system to continue to operate if the system encounters a keyword that is unknown to its policy. The *key*=*value* contents link the files. The following linked entries from the four databases illustrate how the RBAC databases work together.

EXAMPLE 10–1 Showing RBAC Database Connections

In the following example, the user `jdoe` gets the capabilities of the File System Management profile through being assigned the role `filemgr`.

1. The user `jdoe` is assigned the role `filemgr` in the `jdoe` user entry in the `user_attr` database.

   ```
   # user_attr - user definition
   jdoe::::type=normal;roles=filemgr
   ```

2. The role `filemgr` is assigned the rights profile File System Management in the role's entry in the `user_attr` database.

   ```
   # user_attr - role definition
   filemgr::::profiles=File System Management;type=role
   ```

EXAMPLE 10–1 Showing RBAC Database Connections *(Continued)*

The user and the role are uniquely defined in the `passwd` and `shadow` files on the local system, or in equivalent databases in a distributed name service.

3. The File System Management rights profile is defined in the `prof_attr` database. This database also assigns three sets of authorizations to the File System Management entry.

```
# prof_attr - rights profile definitions and assigned authorizations
File System Management:::Manage, mount, share file systems:
help=RtFileSysMngmnt.html;
auths=solaris.admin.fsmgr.*,solaris.admin.diskmgr.*,solaris.admin.volmgr.*
```

4. The authorizations are defined in the `auth_attr` database.

```
# auth_attr - authorization definitions
solaris.admin.fsmgr.:::Mounts and Shares::help=AuthFsmgrHeader.html
solaris.admin.fsmgr.read:::View Mounts and Shares::help=AuthFsmgrRead.html
solaris.admin.fsmgr.write:::Mount and Share Files::help=AuthFsmgrWrite.html
```

5. The File System Management rights profile is assigned commands with security attributes in the `exec_attr` database.

```
# exec_attr - rights profile names with secured commands
File System Management:suser:cmd:::/usr/sbin/mount:uid=0
File System Management:suser:cmd:::/usr/sbin/dfshares:euid=0
...
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
...
```

RBAC Databases and the Name Service

The name service scope of the RBAC databases can apply to the local host only. The scope can also include all hosts that are served by a name service such as NIS, NIS+, or LDAP. Which name service has precedence is set for each of the databases in the `/etc/nsswitch.conf` file.

- `auth_attr` **entry** – Sets the name service precedence for the `auth_attr` database.
- `passwd` **entry** – Sets the name service precedence for the `user_attr` database.
- `prof_attr` **entry** – Sets the name service precedence for the `prof_attr` database. Also sets the name service precedence for the `exec_attr` database.

  For example, if a command with security attributes is assigned to a rights profile that exists in two name service scopes, only the entry in the first name service scope is used.

`user_attr` Database

The `user_attr` database contains user and role information that supplements the `passwd` and `shadow` databases. The `user_attr` database contains extended user attributes such as authorizations, rights profiles, and assigned roles. The fields in the `user_attr` database are separated by colons, as follows:

```
user:qualifier:res1:res2:attr
```

The fields have the following meanings:

`user`
The name of the user or role as specified in the `passwd` database.

`qualifier:res1:res2`
These fields are reserved for future use.

`attr`
An optional list of semicolon-separated (`;`) key-value pairs that describes the security attributes to be applied when the user runs commands. The four valid keys are `type`, `auths`, `profiles`, and `roles`.

- The `type` keyword can be set to `normal`, if this account is for a normal user. The `type` is `role` if this account is for a role.
- The `auths` keyword specifies a comma-separated list of authorization names that are chosen from names that are defined in the `auth_attr` database. Authorization names can include the asterisk (*) character as a wildcard. For example, `solaris.device.*` means all of the Solaris device authorizations.
- The `profiles` keyword specifies an ordered, comma-separated list of rights profile names from the `prof_attr` database. The order of rights profiles works similarly to UNIX search paths. The first profile in the list that contains the command to be executed defines which (if any) security attributes are to be applied to the command.
- The `roles` keyword can be assigned to the user through a comma-separated list of role names. Note that roles are defined in the same `user_attr` database. Roles are indicated by setting the type value to `role`. Roles cannot be assigned to other roles.

The following example demonstrates how the Operator role is defined in a typical `user_attr` database. The example shows how the role is assigned to user `jdoe`. Roles and users are differentiated by the `type` keyword.

```
% grep operator /etc/user_attr
jdoe::::type=normal;roles=operator
operator::::profiles=Operator;type=role
```

`auth_attr` Database

All authorizations are stored in the `auth_attr` database. Authorizations can be assigned to users, to roles, or to rights profiles. The preferred method is to place authorizations in a rights profile, to include the profile in a role's list of profiles, and then to assign the role to a user.

The fields in the `auth_attr` database are separated by colons, as follows:

```
authname:res1:res2:short_desc:long_desc:attr
```

The fields have the following meanings:

`authname`	A unique character string that is used to identify the authorization in the format *prefix*.[*suffix*]. Authorizations for the Solaris OS use `solaris` as a prefix. All other authorizations should use a prefix that begins with the reverse-order Internet domain name of the organization that creates the authorization (for example, `com.xyzcompany`). The suffix indicates what is being authorized, which is typically the functional area and operation. When the `authname` consists of a prefix and functional area and ends with a period, the `authname` serves as a heading to be used by applications in their GUIs. A two-part `authname` is not an actual authorization. The `authname` of `solaris.printmgr.` is an example of a heading. When `authname` ends with the word “grant,” the `authname` serves as a grant authorization. A grant authorization enables the user to delegate to other users authorizations with the same prefix and functional area. The `authname` of `solaris.printmgr.grant` is an example of a grant authorization. `solaris.printmgr.grant` gives the user the right to delegate to other users such authorizations as `solaris.printmgr.admin` and `solaris.printmgr.nobanner`.
`res1:res2`	Reserved for future use.
`short_desc`	A short name for the authorization. This short name is suitable for display in user interfaces, such as in a scrolling list in a GUI.
`long_desc`	A long description. This field identifies the purpose of the authorization, the applications in which the authorization is used, and the type of user who might use the authorization. The long description can be displayed in the help text of an application.
`attr`	An optional list of semicolon-separated (;) key-value pairs that describe the attributes of an authorization. Zero or more keys can be specified. The keyword `help` identifies a help file in HTML. Help files can be accessed from the `index.html` file in the `/usr/lib/help/auths/locale/C` directory.

The following example shows an `auth_attr` database with some typical values:

```
% grep printer /etc/security/auth_attr
solaris.admin.printer.:::Printer Information::help=AuthPrinterHeader.html
solaris.admin.printer.delete:::Delete Printer Information::help=AuthPrinterDelete.html
solaris.admin.printer.modify:::Update Printer Information::help=AuthPrinterModify.html
solaris.admin.printer.read:::View Printer Information::help=AuthPrinterRead.html
```

Note that `solaris.admin.printer.` is defined as a heading, because the authorization name ends in a dot (.). Headings are used by the GUIs to organize families of authorizations.

`prof_attr` Database

The `prof_attr` database stores the name, description, help file location, and authorizations that are assigned to rights profiles. The commands and security attributes that are assigned to rights profiles are stored in the `exec_attr` database. For more information, see "`exec_attr` Database" on page 234. The fields in the `prof_attr` database are separated by colons, as follows:

```
profname:res1:res2:desc:attr
```

The fields have the following meanings:

`profname`	The name of the rights profile. Rights profile names are case-sensitive. This name is also used by the `user_attr` database to indicate the profiles that are assigned to roles and users.
`res1:res2`	Reserved for future use.
`desc`	A long description. This field should explain the purpose of the rights profile, including what type of user would be interested in using the profile. The long description should be suitable for display in the help text of an application.
`attr`	An optional list of key-value pairs that are separated by semicolons (;) that describes the security attributes to apply to the object on execution. Zero or more keys can be specified. The two valid keys are `help` and `auths`. The keyword `help` identifies a help file in HTML. Help files can be accessed from the `index.html` file in the `/usr/lib/help/auths/locale/C` directory. The keyword `auths` specifies a comma-separated list of authorization names that are chosen from those names that are defined in the `auth_attr` database. Authorization names can be specified with the asterisk (*) character as a wildcard.

The following example shows two typical `prof_attr` database entries. Note that the Printer Management rights profile is a supplementary rights profile of the Operator rights profile. The example is wrapped for display purposes.

```
% grep 'Printer Management' /etc/security/prof_attr
Printer Management:::                  Name of rights profile
Manage printers, daemons, spooling:       Description
help=RtPrntAdmin.html;                    Help file
auths=solaris.admin.printer.read,      Authorizations
solaris.admin.printer.modify,solaris.admin.printer.delete
...
Operator:::                               Name of rights profile
Can perform simple administrative tasks:      Description
profiles=Printer Management,           Supplementary rights profiles
Media Backup,All;
help=RtOperator.html                      Help file
```

`exec_attr` Database

The `exec_attr` database defines commands that require security attributes to succeed. The commands are part of a rights profile. A command with its security attributes can be run by roles to whom the profile is assigned.

The fields in the `exec_attr` database are separated by colons, as follows:

```
name:policy:type:res1:res2:id:attr
```

The fields have the following meanings.

`profname`	The name of the rights profile. Rights profile names are case-sensitive. The name refers to a profile in the `prof_attr` database.
`policy`	The security policy that is associated with this entry. Currently, `suser` and `solaris` are the valid entries. The `solaris` policy recognizes privileges. The `suser` policy does not.
`type`	The type of entity that is specified. Currently, the only valid entity type is `cmd` (command).
`res1:res2`	Reserved for future use.
`id`	A string that identifies the entity. Commands should have the full path or a path with a wildcard (`*`). To specify arguments, write a script with the arguments and point the `id` to the script.

`attr` An optional list of semicolon (`;`) separated key-value pairs that describes the security attributes to apply to the entity on execution. Zero or more keys can be specified. The list of valid keywords depends on the policy that is enforced.

For the `suser` policy, the four valid keys are `euid`, `uid`, `egid`, and `gid`.

- The `euid` and `uid` keywords contain a single user name or a numeric user ID (UID). Commands that are designated with `euid` run with the supplied UID, which is similar to setting the `setuid` bit on an executable file. Commands that are designated with `uid` run with both the real UID and the effective UID.
- The `egid` and `gid` keywords contain a single group name or numeric group ID (GID). Commands that are designated with `egid` run with the supplied GID, which is similar to setting the `setgid` bit on an executable file. Commands that are designated with `gid` run with both the real GID and the effective GID.

For the `solaris` policy, the valid keyword is `privs`. The value consists of a list of privileges that are separated by commas.

The following example shows some typical values from an `exec_attr` database:

```
% grep 'File System Management' /etc/security/exec_attr
File System Management:suser:cmd:::/usr/sbin/ff:euid=0
File System Management:solaris:cmd:::/usr/sbin/mount:privs=sys_mount
...
```

`policy.conf` File

The `policy.conf` file provides a way of granting specific rights profiles, specific authorizations, and specific privileges to all users. The relevant entries in the file consist of *key=value* pairs:

- `AUTHS_GRANTED=`*authorizations* – Refers to one or more authorizations.
- `PROFS_GRANTED=`*rights profiles* – Refers to one or more rights profiles.
- `PRIV_DEFAULT=`*privileges* – Refers to one or more privileges.
- `PRIV_LIMIT=`*privileges* – Refers to all privileges.

The following example shows some typical values from a `policy.conf` database:

```
# grep AUTHS /etc/security/policy
AUTHS_GRANTED=solaris.device.cdrw

# grep PROFS /etc/security/policy
PROFS_GRANTED=Basic Solaris User
```

```
# grep PRIV /etc/security/policy

#PRIV_DEFAULT=basic
#PRIV_LIMIT=all
```

For more information about privileges, see “Privileges (Overview)” on page 185.

RBAC Commands

This section lists commands that are used to administer RBAC. Also provided is a table of commands whose access can be controlled by authorizations.

Commands That Manage RBAC

While you can edit the local RBAC databases manually, such editing is strongly discouraged. The following commands are available for managing access to tasks with RBAC.

TABLE 10–7 RBAC Administration Commands

Man Page for Command	Description
`auths`(1)	Displays authorizations for a user.
`makedbm`(1M)	Makes a `dbm` file.
`nscd`(1M)	Name service cache daemon, useful for caching the `user_attr`, `prof_attr`, and `exec_attr` databases. Use the `svcadm` command to restart the daemon.
`pam_roles`(5)	Role account management module for PAM. Checks for the authorization to assume role.
`pfexec`(1)	Used by profile shells to execute commands with security attributes that are specified in the `exec_attr` database.
`policy.conf`(4)	Configuration file for system security policy. Lists granted authorizations, granted privileges, and other security information.
`profiles`(1)	Displays rights profiles for a specified user.
`roles`(1)	Displays roles that a specified user can assume.
`roleadd`(1M)	Adds a role to a local system.
`roledel`(1M)	Deletes a role from a local system.
`rolemod`(1M)	Modifies a role's properties on a local system.

TABLE 10–7 RBAC Administration Commands *(Continued)*

Man Page for Command	Description
`smattrpop`(1M)	Merges the source security attribute database into the target database. For use in situations where local databases need to be merged into a name service. Also for use in upgrades where conversion scripts are not supplied.
`smexec`(1M)	Manages entries in the `exec_attr` database. Requires authentication.
`smmultiuser`(1M)	Manages bulk operations on user accounts. Requires authentication.
`smprofile`(1M)	Manages rights profiles in the `prof_attr` and `exec_attr` databases. Requires authentication.
`smrole`(1M)	Manages roles and users in role accounts. Requires authentication.
`smuser`(1M)	Manages user entries. Requires authentication.
`useradd`(1M)	Adds a user account to the system. The `-P` option assigns a role to a user's account.
`userdel`(1M)	Deletes a user's login from the system.
`usermod`(1M)	Modifies a user's account properties on the system.

Commands That Require Authorizations

The following table provides examples of how authorizations are used to limit command options on a Solaris system. For more discussion of authorizations, see "Authorization Naming and Delegation" on page 228.

TABLE 10–8 Commands and Associated Authorizations

Man Page for Command	Authorization Requirements
`at`(1)	`solaris.jobs.user` required for all options (when neither `at.allow` nor `at.deny` files exist)
`atq`(1)	`solaris.jobs.admin` required for all options
`cdrw`(1)	`solaris.device.cdrw` required for all options, and is granted by default in the `policy.conf` file
`crontab`(1)	`solaris.jobs.user` required for the option to submit a job (when neither `crontab.allow` nor `crontab.deny` files exist) `solaris.jobs.admin` required for the options to list or modify other users' `crontab` files

TABLE 10–8 Commands and Associated Authorizations *(Continued)*

Man Page for Command	Authorization Requirements
`allocate`(1)	`solaris.device.allocate` (or other authorization as specified in `device_allocate` file) required to allocate a device `solaris.device.revoke` (or other authorization as specified in `device_allocate` file) required to allocate a device to another user (`-F` option)
`deallocate`(1)	`solaris.device.allocate` (or other authorization as specified in `device_allocate` file) required to deallocate another user's device `solaris.device.revoke` (or other authorization as specified in `device_allocate`) required to force deallocation of the specified device (`-F` option) or all devices (`-I` option)
`list_devices`(1)	`solaris.device.revoke` required to list another user's devices (`-U` option)
`sendmail`(1M)	`solaris.mail` required to access mail subsystem functions; `solaris.mail.mailq` required to view mail queue

◆ ◆ ◆

CHAPTER 11

Privileges (Tasks)

This chapter provides step-by-step instructions for managing privileges and using privileges on your system. The following is a list of the information in this chapter.

- "Managing and Using Privileges (Task Map)" on page 239
- "Managing Privileges (Task Map)" on page 239
- "Determining Your Privileges (Task Map)" on page 247

For an overview of privileges, see "Privileges (Overview)" on page 185. For reference information, see Chapter 12, "Privileges (Reference)."

Managing and Using Privileges (Task Map)

The following task map points to task maps for managing privileges and for using privileges.

Task	Description	For Instructions
Use privileges at your site	Involves assigning, removing, adding, and debugging the use of privileges.	"Managing Privileges (Task Map)" on page 239
Use privileges when you run a command	Involves using the privileges that have been assigned to you.	"Determining Your Privileges (Task Map)" on page 247

Managing Privileges (Task Map)

The following task map points to procedures for viewing privileges, assigning privileges, and running a script that contains privileged commands.

Task	Description	For Instructions
Determine what privileges are in a process	Lists the effective, inheritable, permitted, and limit privilege sets for a process.	"How to Determine the Privileges on a Process" on page 240
Determine what privileges are missing from a process	Lists the privileges that a failed process requires to succeed.	"How to Determine Which Privileges a Program Requires" on page 242
Add privileges to a command	Adds privileges to a command in a rights profile. Users or roles can be assigned the rights profile. The users can then run the command with the assigned privileges in a profile shell.	"How to Add Privileges to a Command" on page 243
Assign privileges to a user	Expands a user's or role's inheritable set of privileges. Use this procedure with caution.	"How to Assign Privileges to a User or Role" on page 244
Restrict a user's privileges	Limits the user's basic set of privileges. Use this procedure with caution.	"How to Limit a User's or Role's Privileges" on page 245
Run a privileged shell script	Adds privilege to a shell script and to the commands in the shell script. Then, runs the script in a profile shell.	"How to Run a Shell Script With Privileged Commands" on page 247

Managing Privileges

The most secure way to manage privileges for users and roles is to confine use of privilege to commands in a rights profile. The rights profile is then included in a role. The role is assigned to a user. When the user assumes the assigned role, the privileged commands are available to be run in a profile shell. The following procedures show how to assign privileges, remove privileges, and debug privilege use.

▼ How to Determine the Privileges on a Process

This procedure shows how to determine which privileges are available to your processes. The listing does not include privileges that have been assigned to particular commands.

- **List the privileges that are available to your shell's process.**

```
% ppriv pid
$ ppriv -v pid
```

pid	Is the process number. Use a double dollar sign (`$$`) to pass the process number of the parent shell to the command.
`-v`	Provides a verbose listing of the privilege names.

Example 11–1 Determining the Privileges in Your Current Shell

In the following example, the privileges in the parent process of the user's shell process are listed. In the second example, the full names of the privileges are listed. The single letters in the output refer to the following privilege sets:

`E`	Is the effective privilege set.
`I`	Is the inheritable privilege set.
`P`	Is the permitted privilege set.
`L`	Is the limit privilege set.

```
% ppriv $$
1200:   -csh
flags = <none>
        E: basic
        I: basic
        P: basic
        L: all
% ppriv -v $$
1200:   -csh
flags = <none>
        E: file_link_any,proc_exec,proc_fork,proc_info,proc_session
        I: file_link_any,proc_exec,proc_fork,proc_info,proc_session
        P: file_link_any,proc_exec,proc_fork,proc_info,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

Example 11–2 Determining the Privileges of a Role That You Can Assume

Roles use an administrative shell, or profile shell. You must assume a role and use the role's shell to list the privileges that have been directly assigned to the role. In the following example, the role `sysadmin` has no directly assigned privileges.

```
% su - sysadmin
Password:       <Type sysadmin password>
$ /usr/ucb/whoami
sysadmin
$ ppriv -v $$
1400:   pfksh
flags = <none>
        E: file_link_any,proc_exec,proc_fork,proc_info,proc_session
        I: file_link_any,proc_exec,proc_fork,proc_info,proc_session
        P: file_link_any,proc_exec,proc_fork,proc_info,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

▼ How to Determine Which Privileges a Program Requires

This procedure determines which privileges a command or process requires to succeed.

Before You Begin The command or process must have failed for this procedure to work.

1. **Type the command that is failing as an argument to the `ppriv` debugging command.**

```
% ppriv -eD touch /etc/acct/yearly
touch[11365]: missing privilege "file_dac_write"
     (euid = 130, syscall = 224) needed at ufs_direnter_cm+0x27c
touch: /etc/acct/yearly cannot create
```

2. **Determine which system call is failing by finding the `syscall` number in the `/etc/name_to_sysnum` file.**

```
% grep 224 /etc/name_to_sysnum
creat64                 224
```

Example 11–3 Using the `truss` Command to Examine Privilege Use

The `truss` command can debug privilege use in a regular shell. For example, the following command debugs the failing `touch` process:

```
% truss -t creat touch /etc/acct/yearly
creat64("/etc/acct/yearly", 0666)
                       Err#13 EACCES [file_dac_write]
touch: /etc/acct/yearly cannot create
```

The extended `/proc` interfaces report the missing privilege after the error code in `truss` output.

Example 11–4 Using the `ppriv` Command to Examine Privilege Use in a Profile Shell

The `ppriv` command can debug privilege use in a profile shell. If you assign a rights profile to a user, and the rights profile includes commands with privileges, the commands must be typed in a profile shell. When the privileged commands are typed in a regular shell, the commands do not execute with privilege.

In this example, the `jdoe` user can assume the role `objadmin`. The `objadmin` role includes the Object Access Management rights profile. This rights profile allows the `objadmin` role to change permissions on files that `objadmin` does not own.

In the following excerpt, `jdoe` fails to change the permissions on the `useful.script` file:

```
jdoe% ls -l useful.script
-rw-r--r--  1 aloe  staff  2303 Mar 11 05:29 useful.script
jdoe% chown objadmin useful.script
chown: useful.script: Not owner
jdoe% ppriv -eD chown objadmin useful.script
chown[11444]: missing privilege "file_chown"
            (euid = 130, syscall = 16) needed at ufs_setattr+0x258
chown: useful.script: Not owner
```

When `jdoe` assumes the `objadmin` role, the permissions on the file are changed:

```
jdoe% su - objadmin
Password:       <Type objadmin password>
$ ls -l useful.script
-rw-r--r--  1 aloe  staff  2303 Mar 11 05:29 useful.script
$ chown objadmin useful.script
$ ls -l useful.script
-rw-r--r--  1 objadmin  staff  2303 Mar 11 05:29 useful.script
$ chgrp admin useful.script
$ ls -l objadmin.script
-rw-r--r--  1 objadmin  admin  2303 Mar 11 05:31 useful.script
```

Example 11–5 Changing a File Owned by the `root` User

This example illustrates the protections against privilege escalation. For a discussion, see "Prevention of Privilege Escalation" on page 257. The file is owned by the `root` user. The less powerful role, `objadmin` role needs all privileges to change the file's ownership, so the operation fails.

```
jdoe% su - objadmin
Password:       <Type objadmin password>
$ cd /etc; ls -l system
-rw-r--r--  1 root  sys   1883 Mar 20 14:04 system
$ chown objadmin system
chown: system: Not owner
$ ppriv -eD chown objadmin system
chown[11481]: missing privilege "ALL"
     (euid = 101, syscall = 16) needed at ufs_setattr+0x258
chown: system: Not owner
```

▼ How to Add Privileges to a Command

You add privileges to a command when you are adding the command to a rights profile. The privileges enable the role that includes the rights profile to run the administrative command, while not gaining any other superuser capabilities.

Before You Begin The command or program must be privilege-aware. For a fuller discussion, see "How Processes Get Privileges" on page 190.

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 196.

2. **Open the Solaris Management Console GUI.**

   For instructions, see "How to Assume a Role in the Solaris Management Console" on page 211.

3. **Use the Rights tool to update an appropriate profile.**

   Select the command to include. For each included command, add the privileges that the command requires.

Caution – When you include commands in a rights profile and add privileges to the commands, the commands execute with those privileges when the commands are run in a profile shell.

The order of profiles is important. The profile shell executes a command or action with the security attributes that are specified in the earliest profile in the account's list of profiles. For example, if the `chgrp` command is in the Object Access Management rights profile with privileges, and Object Access Management is the first profile in which the `chgrp` command is found, then the `chgrp` command executes with the privileges specified in the Object Access Management profile.

How to Assign Privileges to a User or Role

You might trust some users with a particular privilege all the time. Very specific privileges that affect a small part of the system are good candidates for assigning to a user. For a discussion of the implications of directly assigned privileges, see "Security Considerations When Directly Assigning Security Attributes" on page 184.

The following procedure enables user `jdoe` to use high resolution timers.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Add the privilege that affects high resolution times to the user's initial inheritable set of privileges.**

   ```
   $ usermod -K defaultpriv=basic,proc_clock_highres jdoe
   ```

3 **Read the resulting `user_attr` entry.**

```
$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic,proc_clock_highres
```

Example 11–6 Creating a Role With Privileges to Configure System Time

In this example, a role is created whose only task is to handle time on the system.

```
$ /usr/sadm/bin/smrole -D nisplus:/examplehost/example.domain \
-r primaryadm -l <Type primaryadm password> \
add -- -n clockmgr \
-c "Role that sets system time" \
-F "Clock Manager" \
-s /bin/pfksh \
-u 108 \
-P <Type clockmgr password> \
-K defaultpriv=basic,proc_priocntl,sys_cpu_config,
proc_clock_highres,sys_time
```

The `-K` line is wrapped for display purposes.

If the role was created locally, the `user_attr` entry for the role would appear similar to the following:

```
clockmgr:::Role that sets system time:
type=role;defaultpriv=basic,proc_priocntl,sys_cpu_config,
proc_clock_highres,sys_time
```

How to Limit a User's or Role's Privileges

You can limit the privileges that are available to a user or role by reducing the basic set, or by reducing the limit set. You should have good reason to limit the user's privileges in this way, because such limitations can have unintended side effects.

Caution – You should thoroughly test any user's capabilities where the basic set or the limit set has been modified for a user.

- When the basic set is less than the default, users can be prevented from using the system.
- When the limit set is less than all privileges, processes that need to run with an effective `UID=0` might fail.

---

1 **Determine the privileges in a user's basic set and limit set.**

For the procedure, see "How to Determine the Privileges on a Process" on page 240.

2. **(Optional) Remove one of the privileges from the basic set.**

   ```
   $ usermod -K defaultpriv=basic,!priv-name username
   ```

   By removing the `proc_session` privilege, you prevent the user from examining any processes outside the user's current session. By removing the `file_link_any` privilege, you prevent the user from making hard links to files that are not owned by the user.

   Caution – Do not remove the `proc_fork` or the `proc_exec` privilege. Without these privileges, the user would not be able to use the system. In fact, these two privileges are only reasonably removed from daemons that should not `fork()` or `exec()` other processes.

   ---

3. **(Optional) Remove one of the privileges from the limit set.**

   ```
   $ usermod -K limitpriv=all,!priv-name username
   ```

4. **Test the capabilities of *username*.**

   Log in as *username* and try to perform the tasks that *username* must perform on the system.

Example 11–7 Removing Privileges From a User's Limit Set

In the following example, all sessions that originate from `jdoe`'s initial login are prevented from using the `sys_linkdir` privilege. That is, the user cannot make hard links to directories, nor can the user unlink directories, even after the user runs the `su` command.

```
$ usermod -K limitpriv=all,!sys_linkdir jdoe
$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic;limitpriv=all,!sys_linkdir
```

Example 11–8 Removing Privileges From a User's Basic Set

In the following example, all sessions that originate from `jdoe`'s initial login are prevented from using the `proc_session` privilege. That is, the user cannot examine any processes outside the user's session, even after the user runs the `su` command.

```
$ usermod -K defaultpriv=basic,!proc_session jdoe

$ grep jdoe /etc/user_attr
jdoe::::type=normal;defaultpriv=basic,!proc_session;limitpriv=all
```

▼ How to Run a Shell Script With Privileged Commands

Note – When you create a shell script that runs commands with inherited privileges, the appropriate rights profile must contain the commands with privileges assigned to them.

---

1. **Start the script with `/bin/pfsh`, or any other profile shell, on the first line.**

```
#!/bin/pfsh
# Copyright (c) 2003 by Sun Microsystems, Inc.
```

2. **Determine the privileges that the commands in the script need.**

```
% ppriv -eD script-full-path
```

3. **Open the Solaris Management Console GUI.**

   For instructions, see "How to Assume a Role in the Solaris Management Console" on page 211. Choose a role, such as Primary Administrator, that can create a rights profile.

4. **Use the Rights tool to create or update an appropriate profile.**

   Select the script, and include in the rights profile each of the commands in the shell script that need privileges to run. For each included command, add the privileges that the command requires.

Caution – The order of rights profiles is important. The profile shell executes the earliest instance of a command in the list of profiles. For example, if the `chgrp` command is in the Object Access Management rights profile, and Object Access Management is the first profile in which the `chgrp` command is found, then the `chgrp` command executes with the privileges that are specified in the Object Access Management profile.

---

5. **Add the rights profile to a role and assign the role to a user.**

   To execute the profile, the user assumes the role and runs the script in the role's profile shell.

Determining Your Privileges (Task Map)

The following task map points to procedures for using the privileges that have been assigned to you.

Task	Description	For Instructions
View your privileges as a user in any shell	Shows the privileges that have been directly assigned to you. All of your processes run with these privileges.	"How to Determine the Privileges That You Have Been Directly Assigned" on page 248
Determine which commands you can run with privilege	When privileges are assigned to executables in a rights profile, the executable must be typed in a profile shell.	"How to Determine the Privileged Commands That You Can Run" on page 249
Determine which commands a role can run with privileges	Assumes the role to determine which commands the role can run with privileges.	"How to Determine the Privileged Commands That a Role Can Run" on page 251

Determining Your Assigned Privileges

When a user is directly assigned privileges, the privileges are in effect in every shell. When a user is not directly assigned privileges, then the user must open a profile shell. For example, when commands with assigned privileges are in a rights profile that is in the user's list of rights profiles, then the user must execute the command in a profile shell.

▼ How to Determine the Privileges That You Have Been Directly Assigned

The following procedure shows how to determine if you have been directly assigned privileges.

Caution – Inappropriate use of directly assigned privileges can result in unintentional breaches of security. For a discussion, see "Security Considerations When Directly Assigning Security Attributes" on page 184.

---

1 **List the privileges that your processes can use.**

See "How to Determine the Privileges on a Process" on page 240 for the procedure.

2 **Invoke actions and run commands in any shell.**

The privileges that are listed in the effective set are in effect throughout your session. If you have been directly assigned privileges in addition to the basic set, the privileges are listed in the effective set.

Example 11–9 Determining Your Directly-Assigned Privileges

If you have been directly assigned privileges, then your basic set contains more than the default basic set. In this example, the user always has access to the `proc_clock_highres` privilege.

```
% /usr/ucb/whoami
jdoe
% ppriv -v $$
1800:   pfksh
flags = <none>
        E: file_link_any,...,proc_clock_highres,proc_session
        I: file_link_any,...,proc_clock_highres,proc_session
        P: file_link_any,...,proc_clock_highres,proc_session
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
% ppriv -vl proc_clock_highres
        Allows a process to use high resolution timers.
```

Example 11–10 Determining a Role's Directly-Assigned Privileges

Roles use an administrative shell, or profile shell. Users who assume a role can use the role's shell to list the privileges that have been directly assigned to the role. In the following example, the role `realtime` has been directly assigned privileges to handle date and time programs.

```
% su - realtime
Password:       <Type realtime password>
$ /usr/ucb/whoami
realtime
$ ppriv -v $$
1600:   pfksh
flags = <none>
        E: file_link_any,...,proc_clock_highres,proc_session,sys_time
        I: file_link_any,...,proc_clock_highres,proc_session,sys_time
        P: file_link_any,...,proc_clock_highres,proc_session,sys_time
        L: cpc_cpu,dtrace_kernel,dtrace_proc,dtrace_user,...,sys_time
```

▼ How to Determine the Privileged Commands That You Can Run

When a user is not directly assigned privileges, then the user gets access to privileged commands through a rights profile. Commands in a rights profile must be executed in a profile shell.

Before You Begin The user or role who authenticates to the Solaris Management Console must have the `solaris.admin.usermgr.read` authorization. The Basic Solaris User rights profile includes this authorization.

1. **Determine the rights profiles that you have been assigned.**

```
$ /usr/sadm/bin/smuser list -- -n username -l
Authenticating as user: admin
... Please enter a string value for: password ::
```

```
...
User name:      username
User ID (UID):  130
Primary group:  staff
Secondary groups:
Comment: object mgt jobs
Login Shell: /bin/sh
Home dir server: system
Home directory: /export/home/username
AutoHome setup: True
Mail server: system
Rights: Object Access Management
Assigned Roles:
```

2 **Locate the line that begins with "Rights:".**

The "Rights" line lists the names of the rights profiles that have been directly assigned to you.

3 **Find the names of the rights profiles in the `exec_attr` database.**

```
$ cd /etc/security
$ grep "Object Access Management" exec_attr
Object Access Management:solaris:cmd:::/usr/bin/chgrp:privs=file_chown
Object Access Management:solaris:cmd:::/usr/bin/chown:privs=file_chown
Object Access Management:suser:cmd:::/usr/bin/chgrp:euid=0
Object Access Management:suser:cmd:::/usr/bin/chmod:euid=0
...
```

The commands with added privileges are listed at the end of `solaris` policy entries.

4 **Type the commands that require privileges in a profile shell.**

When the commands are typed in a regular shell, the commands do not run with privilege, and do not succeed.

```
% pfsh
$
```

Example 11–11 Running Privileged Commands in a Profile Shell

In the following example, the user `jdoe` cannot change the group permissions on a file from his regular shell. However, `jdoe` can change the permissions when typing the command in a profile shell.

```
% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 1 nodoe eng 262 Apr 2 10:52 useful.script
chgrp staff useful.script
chgrp: useful.script: Not owner
```

```
% pfksh
$ /usr/ucb/whoami
jdoe
$ chgrp staff useful.script
$ chown jdoe useful.script
$ ls -l useful.script
-rwxr-xr-- 1 jdoe staff 262 Apr 2 10:53 useful.script
```

▼ How to Determine the Privileged Commands That a Role Can Run

A role gets access to privileged commands through a rights profile that contains commands with assigned privileges. The most secure way to provide a user with access to privileged commands is to assign a role to them. After assuming the role, the user can execute all the privileged commands that are included in the rights profiles for that role.

Before You Begin The user or role who authenticates to the Solaris Management Console must have the `solaris.admin.usermgr.read` authorization. The Basic Solaris User rights profile includes this authorization.

1 **Determine the roles that you can assume.**

```
$ /usr/sadm/bin/smuser list -- -n username -l
Authenticating as user: primadmin
...
User name:      username
User ID (UID):  110
Primary group:  staff
Secondary groups:
Comment: Has admin roles
Login Shell: /bin/sh
...
Rights:
Assigned Roles: primadmin, admin
```

2 **Locate the line that begins with "Assigned Roles:".**

The "Assigned Roles" line lists the roles that you can assume.

3 **Determine the rights profiles that are included in one of your roles.**

```
$ /usr/sadm/bin/smuser list -- -n admin -l
Authenticating as user: primadmin
...
User name:      admin
User ID (UID):  101
```

```
Primary group:  sysadmin
Secondary groups:
Comment: system administrator
Login Shell: /bin/pfksh
...
Rights: System Administrator
Assigned Roles:
```

4 **Locate the names of the rights profiles for the role in the "Rights:" line.**

5 **Find the rights profiles in the `prof_attr` database.**

Because the System Administrator profile is a collection of profiles, you need to list the profiles in the System Administrator profile.

```
$ cd /etc/security
$ grep "System Administrator" prof_attr
System Administrator:::Can perform most non-security administrative
tasks:profiles=Audit Review,Printer Management,Cron Management,
Device Management,File System Management,Mail Management,Maintenance
and Repair,Media Backup,Media Restore,Name Service Management,Network
Management,Object Access Management,Process Management,Software
Installation,User Management,All;help=RtSysAdmin.html
```

6 **For each rights profile, find the rights profiles in the `exec_attr` database.**

For example, the Network Management profile is a supplementary profile of the System Administrator profile. The Network Management profile includes a number of privileged commands.

```
$ cd /etc/security
$ grep "Network Management" exec_attr
Network Management:solaris:cmd:::/usr/sbin/ifconfig:privs=sys_net_config
Network Management:solaris:cmd:::/usr/sbin/route:privs=sys_net_config
...
```

The commands and their assigned privileges are the final two fields of `solaris` policy entries. You can run these commands in the profile shell of your role.

Example 11–12 Running the Privileged Commands in Your Role

When a user assumes a role, the shell becomes a profile shell. Therefore, the commands are executed with the privileges that were assigned to the commands. In the following example, the `admin` role can change the permissions on the `useful.script` file.

```
% whoami
jdoe
% ls -l useful.script
-rwxr-xr-- 1 elsee eng 262 Apr 2 10:52 useful.script
chgrp admin useful.script
```

```
chgrp: useful.script: Not owner
% su - admin
Password:      <Type admin password>
$ /usr/ucb/whoami
admin
$ chgrp admin useful.script
$ chown admin useful.script
$ ls -l useful.script
-rwxr-xr-- 1 admin admin 262 Apr 2 10:53 useful.script
```

◆ ◆ ◆

CHAPTER 12

Privileges (Reference)

The following is a list of the reference information in this chapter:

- "Administrative Commands for Handling Privileges" on page 255
- "Files With Privilege Information" on page 256
- "Privileges and Auditing" on page 257
- "Prevention of Privilege Escalation" on page 257
- "Legacy Applications and the Privilege Model" on page 258

To use privileges, see Chapter 11, "Privileges (Tasks)." For overview information, see "Privileges (Overview)" on page 185.

Administrative Commands for Handling Privileges

The following table lists the commands that are available to handle privileges.

TABLE 12–1 Commands for Handling Privilege

Purpose	Command	Man Page
Examine process privileges	`ppriv -v` *pid*	`ppriv`(1)
Set process privileges	`ppriv -s` *spec*	
List the privileges on the system	`ppriv -l`	
List a privilege and its description	`ppriv -lv` *priv*	
Debug privilege failure	`ppriv -eD` *failed-operation*	
Assign privileges to a new local user	`useradd`	`useradd`(1M)
Add privileges to an existing local user	`usermod`	`usermod`(1M)

TABLE 12–1 Commands for Handling Privilege *(Continued)*

Purpose	Command	Man Page
Assign privileges to a user in a name service	`smuser`	`smuser`(1M)
Assign privileges to a new local role	`roleadd`	`roleadd`(1M)
Add privileges to an existing local role	`rolemod`	`rolemod`(1M)
Assign privileges to a role in a name service	`smrole`	`smrole`(1M)
View device policy	`getdevpolicy`	`getdevpolicy`(1M)
Set device policy	`devfsadm`	`devfsadm`(1M)
Update device policy on open devices	`update_drv -p` *policy driver*	`update_drv`(1M)
Add device policy to a device	`add_drv -p` *policy driver*	`add_drv`(1M)

The Solaris Management Console GUI is the preferred tool for assigning privileges to commands, users, and roles. For more information, see "How to Assume a Role in the Solaris Management Console" on page 211.

Files With Privilege Information

The following files contain information about privileges.

TABLE 12–2 Files That Contain Privilege Information

File and Man Page	Keyword	Description
`/etc/security/policy.conf` `policy.conf`(4)	`PRIV_DEFAULT`	Inheritable set of privileges for the system
	`PRIV_LIMIT`	Limit set of privileges for the system
`/etc/user_attr` `user_attr`(4)	`defaultpriv` keyword in user or role's entry Value is usually set in the Solaris Management Console GUI	Inheritable set of privileges for a user or role
	`limitpriv` keyword in user or role's entry Value is usually set in the Solaris Management Console GUI	Limit set of privileges for a user or role
`/etc/security/exec_attr` `exec_attr`(4)	`privs` keyword in the profile's entry for the command Policy for the command must be `solaris`	List of privileges that are assigned to a command in a rights profile

TABLE 12–2 Files That Contain Privilege Information *(Continued)*

File and Man Page	Keyword	Description
`syslog.conf` `syslog.conf`(4)	System log file for debug messages Path set in `priv.debug` entry	Privilege debugging log

---

Note – Do not edit the `exec_attr` and `user_attr` databases directly. To administer privileges, use the Solaris Management Console, or commands such as `smuser`. For more information, see the `smc`(1M) and the `smuser`(1M) man pages. For procedures, see "Managing Privileges (Task Map)" on page 239.

Privileges and Auditing

Privilege use can be audited. When a process uses a privilege, the use of privilege is recorded in the audit trail. The privileges are recorded in their textual representation. The following audit events record use of privilege:

- `AUE_SETPPRIV` **audit event –** The event generates an audit record when a privilege set is changed. The `AUE_SETPPRIV` audit event is in the `pm` class.
- `AUE_MODALLOCPRIV` **audit event –** The audit event generates an audit record when a privilege is added from outside the kernel. The `AUE_MODALLOCPRIV` audit event is in the `ad` class.
- `AUE_MODDEVPLCY` **audit event –** The audit event generates an audit record when the device policy is changed. The `AUE_MODDEVPLCY` audit event is in the `ad` class.
- `AUE_prof_cmd` **audit event –** The audit event generates an audit record when a command is executed in a profile shell. The `AUE_prof_cmd` audit event is in the `as` and `ua` audit classes.

The successful use of privileges that are in the basic set is not audited. The attempt to use a basic privilege that has been removed from a user's basic set is audited.

Prevention of Privilege Escalation

The Solaris kernel prevents *privilege escalation*. Privilege escalation is when a privilege enables a process to do more than the process should be able to do. To prevent a process from gaining more privileges than the process should have, certain system modifications require the full set of privileges. For example, a file or process that is owned by `root` (`UID=0`) can only be changed by a process with the full set of privileges. The `root` user does not require privileges to change a file that `root` owns. However, a non-root user must have all privileges in order to change a file that is owned by `root`.

Similarly, operations that provide access to devices require all privileges in the effective set.

The `file_chown_self` and `proc_owner` privileges are subject to privilege escalation. The `file_chown_self` privilege allows a process to give away its files. The `proc_owner` privilege allows a process to inspect processes that the process does not own.

The `file_chown_self` privilege is limited by the `rstchown` system variable. When the `rstchown` variable is set to zero, the `file_chown_self` privilege is removed from the initial inheritable set of the system and of all users. For more information on the `rstchown` system variable, see the `chown`(1) man page.

The `file_chown_self` privilege is most safely assigned to a particular command, placed in a profile, and assigned to a role for use in a profile shell.

The `proc_owner` privilege is not sufficient to switch a process UID to `0`. To switch a process from any UID to `UID=0` requires all privileges. Because the `proc_owner` privilege gives unrestricted read access to all files on the system, the privilege is most safely assigned to a particular command, placed in a profile, and assigned to a role for use in a profile shell.

Caution – A user's account can be modified to include the `file_chown_self` privilege or the `proc_owner` privilege in the user's initial inheritable set. You should have overriding security reasons for placing such powerful privileges in the inheritable set of privileges for any user, role, or system.

For details of how privilege escalation is prevented for devices, see "Privileges and Devices" on page 192.

Legacy Applications and the Privilege Model

To accommodate legacy applications, the implementation of privileges works with both the superuser and the privilege models. The kernel automatically tracks the `PRIV_AWARE` flag, which indicates that a program has been designed to work with privileges. Consider a child process that is not aware of privileges. Any privileges that were inherited from the parent process are available in the child's permitted and effective sets. If the child process sets a UID to 0, the child process might not have full superuser capabilities. The process's effective and permitted sets are restricted to those privileges in the child's limit set. Thus, the limit set of a privilege-aware process restricts the root privileges of child processes that are not aware of privileges.

PART IV

Solaris Cryptographic Services

This section describes the centralized cryptographic and public key technology services that the Solaris OS provides.

◆ ◆ ◆ CHAPTER 13

Solaris Cryptographic Framework (Overview)

This chapter describes the Solaris™ Cryptographic Framework. The following is a list of the information in this chapter.

- "What's New in the Solaris Cryptographic Framework?" on page 261
- "Solaris Cryptographic Framework" on page 262
- "Terminology in the Solaris Cryptographic Framework" on page 262
- "Scope of the Solaris Cryptographic Framework" on page 264
- "Administrative Commands in the Solaris Cryptographic Framework" on page 264
- "User-Level Commands in the Solaris Cryptographic Framework" on page 265
- "Plugins to the Solaris Cryptographic Framework" on page 266
- "Cryptographic Services and Zones" on page 266

To administer and use the Solaris Cryptographic Framework, see Chapter 14, "Solaris Cryptographic Framework (Tasks)."

What's New in the Solaris Cryptographic Framework?

Solaris 10 1/06: The framework library, `libpkcs11.so`, contains a new component, the *metaslot*. The metaslot serves as a single virtual slot with the combined capabilities of all tokens and slots that have been installed in the framework. Effectively, the metaslot enables an application to transparently connect with any available cryptographic service through a single slot.

- For more information, see the definitions of slot, metaslot, and token in "Terminology in the Solaris Cryptographic Framework" on page 262.
- To administer the metaslot, see the `cryptoadm`(1M) man page.
- For a complete listing of new Solaris features and a description of Solaris releases, see *Solaris 10 What's New*.

Solaris Cryptographic Framework

The Solaris Cryptographic Framework provides a common store of algorithms and PKCS #11 libraries to handle cryptographic requirements. The PKCS #11 libraries are implemented according to the following standard: RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki).

At the kernel level, the framework currently handles cryptographic requirements for Kerberos and IPsec. User-level consumers include `libsasl` and IKE.

Export law in the United States requires that the use of open cryptographic interfaces be restricted. The Solaris Cryptographic Framework satisfies the current law by requiring that kernel cryptographic providers and PKCS #11 cryptographic providers be signed. For further discussion, see "Binary Signatures for Third-Party Software" on page 265.

The framework enables *providers* of cryptographic services to have their services used by many *consumers* in the Solaris Operating System. Another name for providers is *plugins*. The framework allows three types of plugins:

- **User-level plugins** – Shared objects that provide services by using PKCS #11 libraries, such as `pkcs11_softtoken.so.1`.
- **Kernel-level plugins** – Kernel modules that provide implementations of cryptographic algorithms in software, such as AES.

  Many of the algorithms in the framework are optimized for x86 with the SSE2 instruction set and for SPARC hardware.
- **Hardware plugins** – Device drivers and their associated hardware accelerators. A hardware accelerator offloads expensive cryptographic functions from the operating system. The Sun Crypto Accelerator 1000 board is one example.

The framework implements a standard interface, the PKCS #11, v2.11 library, for user-level providers. The library can be used by third-party applications to reach providers. Third parties can also add signed libraries, signed kernel algorithm modules, and signed device drivers to the framework. These plugins are added when the `pkgadd` utility installs the third-party software. For a diagram of the major components of the framework, see Chapter 8, "Introduction to the Solaris Cryptographic Framework," in *Solaris Security for Developers Guide*.

Terminology in the Solaris Cryptographic Framework

The following list of definitions and examples is useful when working with the cryptographic framework.

- **Algorithms –** Cryptographic algorithms. These are established, recursive computational procedures that encrypt or hash input. Encryption algorithms can be symmetric or asymmetric. Symmetric algorithms use the same key for encryption and decryption. Asymmetric algorithms, which are used in public-key cryptography, require two keys. Hashing functions are also algorithms.

  Examples of algorithms include:

  - Symmetric algorithms, such as AES and ARCFOUR
  - Asymmetric algorithms, such as Diffie-Hellman and RSA
  - Hashing functions, such as MD5
- **Consumers –** Are users of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations.

  Examples of consumers include:

  - Applications, such as IKE
  - End users, such as an ordinary user who runs the `encrypt` command
  - Kernel operations, such as IPsec
- **Mechanism –** Is the application of a mode of an algorithm for a particular purpose.

  For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.
- **Metaslot –** Is a single slot that presents a union of the capabilities of other slots which are loaded in the framework. The metaslot eases the work of dealing with all of the capabilities of the providers that are available through the framework. When an application that uses the metaslot requests an operation, the metaslot figures out which actual slot should perform the operation. Metaslot capabilities are configurable, but configuration is not required. The metaslot is on by default. To configure the metaslot, see the `cryptoadm`(1M) man page.
- **Mode –** Is a version of a cryptographic algorithm. For example, CBC (Cipher Block Chaining) is a different mode from ECB (Electronic Code Book). The AES algorithm has two modes, CKM_AES_ECB and CKM_AES_CBC.
- **Policy –** Is the choice, by an administrator, of which mechanisms to make available for use. By default, all providers and all mechanisms are available for use. The disabling of any mechanism would be an application of policy. The enabling of a disabled mechanism would also be an application of policy.
- **Providers –** Are cryptographic services that consumers use. Providers plug in to the framework, so are also called *plugins*.

  Examples of providers include:

  - PKCS #11 libraries, such as `pkcs11_softtoken.so`
  - Modules of cryptographic algorithms, such as `aes` and `arcfour`
  - Device drivers and their associated hardware accelerators, such as the `dca/0` accelerator

- **Slot –** Is an interface to one or more cryptographic devices. Each slot, which corresponds to a physical reader or other device interface, might contain a token. A token provides a logical view of a cryptographic device in the framework.
- **Token –** In a slot, a token provides a logical view of a cryptographic device in the framework.

Scope of the Solaris Cryptographic Framework

The framework provides commands for administrators, for users, and for developers who supply providers:

- **Administrative commands –** The `cryptoadm` command provides a `list` subcommand to list the available providers and their capabilities. Ordinary users can run the `cryptoadm list` and the `cryptoadm --help` commands.

  All other `cryptoadm` subcommands require you to assume a role that includes the Crypto Management rights profile, or to become superuser. Subcommands such as `disable`, `install`, and `uninstall` are available for administering the framework. For more information, see the `cryptoadm`(1M) man page.

  The `svcadm` command is used to manage the `kcfd` daemon, and to refresh cryptographic policy in the kernel. For more information, see the `svcadm`(1M) man page.
- **User-level commands –** The `digest` and `mac` commands provide file integrity services. The `encrypt` and `decrypt` commands protect files from eavesdropping. To use these commands, see "Protecting Files With the Solaris Cryptographic Framework (Task Map)" on page 268.
- **Binary signatures for third-party providers –** The `elfsign` command enables third parties to sign binaries for use within the framework. Binaries that can be added to the framework are PKCS #11 libraries, kernel algorithm modules, and hardware device drivers. To use the `elfsign` command, see Appendix F, "Packaging and Signing Cryptographic Providers," in *Solaris Security for Developers Guide*.

Administrative Commands in the Solaris Cryptographic Framework

The `cryptoadm` command administers a running cryptographic framework. The command is part of the Crypto Management rights profile. This profile can be assigned to a role for secure administration of the cryptographic framework. The `cryptoadm` command manages the following:

- Displaying cryptographic provider information
- Disabling or enabling provider mechanisms
- Solaris 10 1/06: Disabling or enabling the metaslot

The `svcadm` command is used to enable, refresh, and disable the cryptographic services daemon, `kcfd`. This command is part of the Solaris service management facility, `smf`. `svc:/system/cryptosvcs` is the service instance for the cryptographic framework. For more information, see the `smf`(5) and `svcadm`(1M) man pages.

User-Level Commands in the Solaris Cryptographic Framework

The Solaris Cryptographic Framework provides user-level commands to check the integrity of files, to encrypt files, and to decrypt files. A separate command, `elfsign`, enables providers to sign binaries for use with the framework.

- `digest` **command** – Computes a message digest for one or more files or for stdin. A digest is useful for verifying the integrity of a file. SHA1 and MD5 are examples of digest functions.
- `mac` **command** – Computes a message authentication code (MAC) for one or more files or for stdin. A MAC associates data with an authenticated message. A MAC enables a receiver to verify that the message came from the sender and that the message has not been tampered with. The `sha1_mac` and `md5_hmac` mechanisms can compute a MAC.
- `encrypt` **command** – Encrypts files or stdin with a symmetric cipher. The `encrypt -l` command lists the algorithms that are available. Mechanisms that are listed under a user-level library are available to the `encrypt` command. The framework provides AES, DES, 3DES (Triple-DES), and ARCFOUR mechanisms for user encryption.
- `decrypt` **command** – Decrypts files or stdin that were encrypted with the `encrypt` command. The `decrypt` command uses the identical key and mechanism that were used to encrypt the original file.

Binary Signatures for Third-Party Software

The `elfsign` command provides a means to sign providers to be used with the Solaris Cryptographic Framework. Typically, this command is run by the developer of a provider.

The `elfsign` command has subcommands to request a certificate from Sun and to sign binaries. Another subcommand verifies the signature. Unsigned binaries cannot be used by the Solaris Cryptographic Framework. To sign one or more providers requires the certificate from Sun and the private key that was used to request the certificate. For more information, see Appendix F, "Packaging and Signing Cryptographic Providers," in *Solaris Security for Developers Guide*.

Plugins to the Solaris Cryptographic Framework

Third parties can plug their providers into the Solaris Cryptographic Framework. A third-party provider can be one of the following objects:

- PKCS #11 shared library
- Loadable kernel software module, such as an encryption algorithm, MAC function, or digest function
- Kernel device driver for a hardware accelerator

The objects from a provider must be signed with a certificate from Sun. The certificate request is based on a private key that the third party selects, and a certificate that Sun provides. The certificate request is sent to Sun, which registers the third party and then issues the certificate. The third party then signs its provider object with the certificate from Sun.

The loadable kernel software modules and the kernel device drivers for hardware accelerators must also register with the kernel. Registration is through the Solaris Cryptographic Framework SPI (service provider interface).

To install the provider, the third party provides a package that installs the signed object and the certificate from Sun. The package must include the certificate, and enable the administrator to place the certificate in a secure directory. For more information, see the Appendix F, “Packaging and Signing Cryptographic Providers,” in *Solaris Security for Developers Guide*.

Cryptographic Services and Zones

The global zone and each non-global zone has its own `/system/cryptosvc` service. When the cryptographic service is enabled or refreshed in the global zone, the `kcfd` daemon starts in the global zone, user-level policy for the global zone is set, and kernel policy for the system is set. When the service is enabled or refreshed in a non-global zone, the `kcfd` daemon starts in the zone, and user-level policy for the zone is set. Kernel policy was set by the global zone.

For more information on zones, see Part II, “Zones,” in *System Administration Guide: Virtualization Using the Solaris Operating System*. For more information on the service management facility that manages persistent applications, see Chapter 15, “Managing Services (Overview),” in *System Administration Guide: Basic Administration* and the `smf`(5) man page.

◆ ◆ ◆

CHAPTER 14

Solaris Cryptographic Framework (Tasks)

This chapter describes how to use the Solaris Cryptographic Framework. The following is a list of information in this chapter.

- "Using the Cryptographic Framework (Task Map)" on page 267
- "Protecting Files With the Solaris Cryptographic Framework (Task Map)" on page 268
- "Administering the Cryptographic Framework (Task Map)" on page 275

Using the Cryptographic Framework (Task Map)

The following task map points to tasks for using the cryptographic framework.

Task	Description	For Instructions
Protect individual files or sets of files	Ensures that file content has not been tampered with. Prevents files from being read by intruders. These procedures can be done by ordinary users.	"Protecting Files With the Solaris Cryptographic Framework (Task Map)" on page 268
Administer the framework	Adds, configures, and removes software providers. Disables and enables hardware provider mechanisms. These procedures are administrative procedures.	"Administering the Cryptographic Framework (Task Map)" on page 275
Sign a provider	Enables a provider to be added to the Solaris Cryptographic Framework. These procedures are developer procedures.	Appendix F, "Packaging and Signing Cryptographic Providers," in *Solaris Security for Developers Guide*.

Protecting Files With the Solaris Cryptographic Framework (Task Map)

The Solaris Cryptographic Framework can help you protect your files. The following task map points to procedures for listing the available algorithms, and for protecting your files cryptographically.

Task	Description	For Instructions
Generate a symmetric key	Generates a random key for use with algorithms that the user specifies.	"How to Generate a Symmetric Key" on page 268
Provide a checksum that ensures the integrity of a file	Verifies that the receiver's copy of a file is identical to the file that was sent.	"How to Compute a Digest of a File" on page 270
Protect a file with a message authentication code (MAC)	Verifies to the receiver of your message that you were the sender.	"How to Compute a MAC of a File" on page 271
Encrypt a file, and then decrypt the encrypted file	Protects the content of files by encrypting the file. Provides the encryption parameters to decrypt the file.	"How to Encrypt and Decrypt a File" on page 273

Protecting Files With the Solaris Cryptographic Framework

This section describes how to generate symmetric keys, how to create checksums for file integrity, and how to protect files from eavesdropping. The commands in this section can be run by regular users. Developers can write scripts that use these commands.

▼ How to Generate a Symmetric Key

A key is needed to encrypt files and to generate the MAC of a file. The key should be derived from a random pool of numbers.

If your site has a random number generator, use the generator. Otherwise, you can use the `dd` command with the Solaris `/dev/urandom` device as input. For more information, see the `dd`(1M) man page.

1. **Determine the key length that your algorithm requires.**

   a. **List the available algorithms.**

```
% encrypt -l
Algorithm       Keysize:  Min   Max (bits)
------------------------------------------
aes                       128   128
```

```
arcfour                     8   128
des                        64    64
3des                      192   192

% mac -l
Algorithm       Keysize:  Min   Max (bits)
------------------------------------------
des_mac                    64    64
sha1_hmac                   8   512
md5_hmac                    8   512
```

b. Determine the key length in bytes to pass to the `dd` command.

Divide the minimum and maximum key sizes by 8. When the minimum and maximum key sizes are different, intermediate key sizes are possible. For example, the value 8, 16, or 64 can be passed to the `dd` command for the `sha1_hmac` and `md5_hmac` functions.

2 Generate the symmetric key.

```
% dd if=/dev/urandom of=keyfile bs=n count=n
```

`if=`*file*	Is the input file. For a random key, use the `/dev/urandom` file.
`of=`*keyfile*	Is the output file that holds the generated key.
`bs=`*n*	Is the key size in bytes. For the length in bytes, divide the key length in bits by 8.
`count=`*n*	Is the count of the input blocks. The number for *n* should be `1`.

3 Store your key in a protected directory.

The key file should not be readable by anyone but the user.

```
% chmod 400 keyfile
```

Example 14–1 Creating a Key for the AES Algorithm

In the following example, a secret key for the AES algorithm is created. The key is also stored for later decryption. AES mechanisms use a 128-bit key. The key is expressed as 16 bytes in the `dd` command.

```
% ls -al ~/keyf
drwx------   2 jdoe  staff        512 May 3 11:32 ./
% dd if=/dev/urandom of=$HOME/keyf/05.07.aes16 bs=16  count=1
% chmod 400 ~/keyf/05.07.aes16
```

Example 14–2 Creating a Key for the DES Algorithm

In the following example, a secret key for the DES algorithm is created. The key is also stored for later decryption. DES mechanisms use a 64-bit key. The key is expressed as 8 bytes in the `dd` command.

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.des8 bs=8  count=1
% chmod 400 ~/keyf/05.07.des8
```

Example 14–3 Creating a Key for the 3DES Algorithm

In the following example, a secret key for the 3DES algorithm is created. The key is also stored for later decryption. 3DES mechanisms use a 192-bit key. The key is expressed as 24 bytes in the `dd` command.

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.3des.24 bs=24 count=1
% chmod 400 ~/keyf/05.07.3des.24
```

Example 14–4 Creating a Key for the MD5 Algorithm

In the following example, a secret key for the MD5 algorithm is created. The key is also stored for later decryption. The key is expressed as 64 bytes in the `dd` command.

```
% dd if=/dev/urandom of=$HOME/keyf/05.07.mack64 bs=64 count=1
% chmod 400 ~/keyf/05.07.mack64
```

▼ How to Compute a Digest of a File

When you compute a digest of a file, you can check to see that the file has not been tampered with by comparing digest outputs. A digest does not alter the original file.

1. **List the available digest algorithms.**

```
% digest -l
md5
sha1
```

2. **Compute the digest of the file and save the digest listing.**

Provide an algorithm with the `digest` command.

`% digest -v -a` *algorithm input-file* `>` *digest-listing*

`-v` Displays the output in the following format:

algorithm `(`*input-file*`) =` *digest*

`-a` *algorithm*	Is the algorithm to use to compute a digest of the file. Type the algorithm as the algorithm appears in the output of Step 1.
input-file	Is the input file for the `digest` command.
digest-listing	Is the output file for the `digest` command.

Example 14–5 Computing a Digest With the MD5 Mechanism

In the following example, the `digest` command uses the MD5 mechanism to compute a digest for an email attachment.

```
% digest -v -a md5 email.attach >>  $HOME/digest.emails.05.07
% cat  ~/digest.emails.05.07
md5 (email.attach) = 85c0a53d1a5cc71ea34d9ee7b1b28b01
```

When the `-v` option is not used, the digest is saved with no accompanying information:

```
% digest -a md5 email.attach >>  $HOME/digest.emails.05.07
% cat  ~/digest.emails.05.07
85c0a53d1a5cc71ea34d9ee7b1b28b01
```

Example 14–6 Computing a Digest With the SHA1 Mechanism

In the following example, the `digest` command uses the SHA1 mechanism to provide a directory listing. The results are placed in a file.

```
% digest -v -a sha1 docs/* > $HOME/digest.docs.legal.05.07
% more ~/digest.docs.legal.05.07
sha1 (docs/legal1) = 1df50e8ad219e34f0b911e097b7b588e31f9b435
sha1 (docs/legal2) = 68efa5a636291bde8f33e046eb33508c94842c38
sha1 (docs/legal3) = 085d991238d61bd0cfa2946c183be8e32cccf6c9
sha1 (docs/legal4) = f3085eae7e2c8d008816564fdf28027d10e1d983
```

▼ How to Compute a MAC of a File

A message authentication code, or MAC, computes a digest for the file and uses a secret key to further protect the digest. A MAC does not alter the original file.

1 **List the available mechanisms.**

```
% mac -l
Algorithm       Keysize:  Min   Max
-----------------------------------
des_mac                    64    64
```

```
sha1_hmac                      8    512
md5_hmac                       8    512
```

2. **Generate a symmetric key of the appropriate length.**

   You have two options. You can provide a passphrase from which a key will be generated. Or you can provide a key.

   - If you provide a passphrase, you must store or remember the passphrase. If you store the passphrase online, the passphrase file should be readable only by you.
   - If you provide a key, it must be the correct size for the mechanism. For the procedure, see "How to Generate a Symmetric Key" on page 268.

3. **Create a MAC for a file.**

   Provide a key and use a symmetric key algorithm with the `mac` command.

   ```
   % mac -v -a algorithm [ -k keyfile ] input-file
   ```

`-v`	Displays the output in the following format: *algorithm* (*input-file*) = *mac*
`-a` *algorithm*	Is the algorithm to use to compute the MAC. Type the algorithm as the algorithm appears in the output of the `mac -l` command.
`-k` *keyfile*	Is the file that contains a key of algorithm-specified length.
input-file	Is the input file for the MAC.

Example 14–7 Computing a MAC With DES_MAC and a Passphrase

In the following example, the email attachment is authenticated with the DES_MAC mechanism and a key that is derived from a passphrase. The MAC listing is saved to a file. If the passphrase is stored in a file, the file should not be readable by anyone but the user.

```
% mac -v -a des_mac email.attach
Enter key:     <Type passphrase>
des_mac (email.attach) = dd27870a
% echo "des_mac (email.attach) = dd27870a" >> ~/desmac.daily.05.07
```

Example 14–8 Computing a MAC With MD5_HMAC and a Key File

In the following example, the email attachment is authenticated with the MD5_HMAC mechanism and a secret key. The MAC listing is saved to a file.

```
% mac -v -a md5_hmac -k $HOME/keyf/05.07.mack64 email.attach
md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c
```

```
% echo "md5_hmac (email.attach) = 02df6eb6c123ff25d78877eb1d55710c" \
>> ~/mac.daily.05.07
```

Example 14–9 Computing a MAC With SHA1_HMAC and a Key File

In the following example, the directory manifest is authenticated with the SHA1_HMAC mechanism and a secret key. The results are placed in a file.

```
% mac -v -a sha1_hmac \
-k $HOME/keyf/05.07.mack64 docs/* > $HOME/mac.docs.legal.05.07
% more ~/mac.docs.legal.05.07
sha1_hmac (docs/legal1) = 9b31536d3b3c0c6b25d653418db8e765e17fe07a
sha1_hmac (docs/legal2) = 865af61a3002f8a457462a428cdb1a88c1b51ff5
sha1_hmac (docs/legal3) = 076c944cb2528536c9aebd3b9fbe367e07b61dc7
sha1_hmac (docs/legal4) = 7aede27602ef6e4454748cbd3821e0152e45beb4
```

▼ How to Encrypt and Decrypt a File

When you encrypt a file, the original file is not removed or changed. The output file is encrypted.

For solutions to common errors from the `encrypt` command, see the section that follows the examples.

1. **Create a symmetric key of the appropriate length.**

   You have two options. You can provide a passphrase from which a key will be generated. Or you can provide a key.

   - If you provide a passphrase, you must store or remember the passphrase. If you store the passphrase online, the passphrase file should be readable only by you.
   - If you provide a key, it must be the correct size for the mechanism. For the procedure, see "How to Generate a Symmetric Key" on page 268.

2. **Encrypt a file.**

   Provide a key and use a symmetric key algorithm with the `encrypt` command.

```
% encrypt -a algorithm  [ -k keyfile ] -i input-file -o output-file
```

`-a` *algorithm*	Is the algorithm to use to encrypt the file. Type the algorithm as the algorithm appears in the output of the `encrypt -l` command.
`-k` *keyfile*	Is the file that contains a key of algorithm-specified length. The key length for each algorithm is listed, in bits, in the output of the `encrypt -l` command.

`-i` *input-file*	Is the input file that you want to encrypt. This file is left unchanged by the command.
`-o` *output-file*	Is the output file that is the encrypted form of the input file.

Example 14–10 Encrypting and Decrypting With AES and a Passphrase

In the following example, a file is encrypted with the AES algorithm. The key is generated from the passphrase. If the passphrase is stored in a file, the file should not be readable by anyone but the user.

```
% encrypt -a aes -i ticket.to.ride -o ~/enc/e.ticket.to.ride
Enter key:     <Type passphrase>
```

The input file, `ticket.to.ride`, still exists in its original form.

To decrypt the output file, the user uses the same passphrase and encryption mechanism that encrypted the file.

```
% decrypt -a aes -i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
Enter key:     <Type passphrase>
```

Example 14–11 Encrypting and Decrypting With AES and a Key File

In the following example, a file is encrypted with the AES algorithm. AES mechanisms use a key of 128 bits, or 16 bytes.

```
% encrypt -a aes -k ~/keyf/05.07.aes16 \
-i ticket.to.ride -o ~/enc/e.ticket.to.ride
```

The input file, `ticket.to.ride`, still exists in its original form.

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a aes -k ~/keyf/05.07.aes16  \
-i ~/enc/e.ticket.to.ride -o ~/d.ticket.to.ride
```

Example 14–12 Encrypting and Decrypting With ARCFOUR and a Key File

In the following example, a file is encrypted with the ARCFOUR algorithm. The ARCFOUR algorithm accepts a key of 8 bits (1 byte), 64 bits (8 bytes), or 128 bits (16 bytes).

```
% encrypt -a arcfour -i personal.txt \
-k ~/keyf/05.07.rc4.8 -o ~/enc/e.personal.txt
```

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a arcfour -i ~/enc/e.personal.txt \
-k ~/keyf/05.07.rc4.8 -o ~/personal.txt
```

Example 14–13 Encrypting and Decrypting With 3DES and a Key File

In the following example, a file is encrypted with the 3DES algorithm. The 3DES algorithm requires a key of 192 bits, or 24 bytes.

```
% encrypt -a 3des -k ~/keyf/05.07.des24 \
-i ~/personal2.txt -o ~/enc/e.personal2.txt
```

To decrypt the output file, the user uses the same key and encryption mechanism that encrypted the file.

```
% decrypt -a 3des -k ~/keyf/05.07.des24 \
-i ~/enc/e.personal2.txt -o ~/personal2.txt
```

Troubleshooting The following messages indicate that the key that you provided to the `encrypt` command is not permitted by the algorithm that you are using.

- `encrypt: unable to create key for crypto operation: CKR_ATTRIBUTE_VALUE_INVALID`
- `encrypt: failed to initialize crypto operation: CKR_KEY_SIZE_RANGE`

If you pass a key that does not meet the requirements of the algorithm, you must supply a better key.

- One option is to use a passphrase. The framework then provides a key that meets the requirements.
- The second option is to pass a key size that the algorithm accepts. For example, the DES algorithm requires a key of 64 bits. The 3DES algorithm requires a key of 192 bits.

Administering the Cryptographic Framework (Task Map)

The following task map points to procedures for administering software and hardware providers in the Solaris Cryptographic Framework.

Task	Description	For Instructions
List the providers in the Solaris Cryptographic Framework	Lists the algorithms, libraries, and hardware devices that are available for use in the Solaris Cryptographic Framework.	"How to List Available Providers" on page 276
Add a software provider	Adds a PKCS #11 library or a kernel module to the Solaris Cryptographic Framework. The provider must be signed.	"How to Add a Software Provider" on page 279
Prevent the use of a user-level mechanism	Removes a software mechanism from use. The mechanism can be enabled again.	"How to Prevent the Use of a User-Level Mechanism" on page 280
Temporarily disable mechanisms from a kernel module	Temporarily removes a mechanism from use. Usually used for testing.	"How to Prevent the Use of a Kernel Software Provider" on page 282
Uninstall a provider	Removes a kernel software provider from use.	Example 14–22
List available hardware providers	Shows the attached hardware, shows the mechanisms that the hardware provides, and shows which mechanisms are enabled for use.	"How to List Hardware Providers" on page 285
Disable mechanisms from a hardware provider	Ensures that selected mechanisms on a hardware accelerator are not used.	"How to Disable Hardware Provider Mechanisms and Features" on page 285
Restart or refresh cryptographic services	Ensures that cryptographic services are available.	"How to Refresh or Restart All Cryptographic Services" on page 287

Administering the Cryptographic Framework

This section describes how to administer the software providers and the hardware providers in the Solaris Cryptographic Framework. Software providers and hardware providers can be removed from use when desirable. For example, you can disable the implementation of an algorithm from one software provider. You can then force the system to use the algorithm from a different software provider.

▼ How to List Available Providers

The Solaris Cryptographic Framework provides algorithms for several types of consumers:

- User-level providers provide a PKCS #11 cryptographic interface to applications that are linked with the `libpkcs11` library
- Kernel software providers provide algorithms for IPsec, Kerberos, and other Solaris kernel components
- Kernel hardware providers provide algorithms that are available to kernel consumers and to applications through the `pkcs11_kernel` library

1 **List the providers in a brief format.**

Only those mechanisms at the user level are available for use by regular users.

```
% cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa

kernel hardware providers:
    dca/0
```

2 **List the providers and their mechanisms in the Solaris Cryptographic Framework.**

All mechanisms are listed in the following output. However, some of the listed mechanisms might be unavailable for use. To list only the mechanisms that the administrator has approved for use, see Example 14–15.

The output is reformatted for display purposes.

```
% cryptoadm list -m
user-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: CKM_MD5,CKM_MD5_HMAC,
CKM_MD5_HMAC_GENERAL,CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL,
...
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
...
kernel software providers:
==========================
des: CKM_DES_ECB,CKM_DES_CBC,CKM_DES3_ECB,CKM_DES3_CBC
aes: CKM_AES_ECB,CKM_AES_CBC
blowfish: CKM_BF_ECB,CKM_BF_CBC
arcfour: CKM_RC4
sha1: CKM_SHA_1,CKM_SHA_1_HMAC,CKM_SHA_1_HMAC_GENERAL
md5: CKM_MD5,CKM_MD5_HMAC,CKM_MD5_HMAC_GENERAL
rsa: CKM_RSA_PKCS,CKM_RSA_X_509,CKM_MD5_RSA_PKCS,CKM_SHA1_RSA_PKCS
swrand: No mechanisms presented.
```

```
kernel hardware providers:
==========================
dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM_DES_CBC, CKM_DES3_CBC
```

Example 14–14 Finding the Existing Cryptographic Mechanisms

In the following example, all mechanisms that the user-level library, `pkcs11_softtoken`, offers are listed.

```
% cryptoadm list -m provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
...
CKM_SSL3_KEY_AND_MAC_DERIVE,CKM_TLS_KEY_AND_MAC_DERIVE
```

Example 14–15 Finding the Available Cryptographic Mechanisms

Policy determines which mechanisms are available for use. The administrator sets the policy. An administrator can choose to disable mechanisms from a particular provider. The `-p` option displays the list of mechanisms that are permitted by the policy that the administrator has set.

```
% cryptoadm list -p
user-level providers:
=====================
/usr/lib/security/$ISA/pkcs11_kernel.so: all mechanisms are enabled.
random is enabled.
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.

kernel software providers:
==========================
des: all mechanisms are enabled.
aes: all mechanisms are enabled.
blowfish: all mechanisms are enabled.
arcfour: all mechanisms are enabled.
sha1: all mechanisms are enabled.
md5: all mechanisms are enabled.
rsa: all mechanisms are enabled.
swrand: random is enabled.

kernel hardware providers:
==========================
dca/0: all mechanisms are enabled. random is enabled.
```

▼ How to Add a Software Provider

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **List the software providers that are available to the system.**

```
# cryptoadm list
user-level providers:
        /usr/lib/security/$ISA/pkcs11_kernel.so
        /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand

kernel hardware providers:
     dca/0
```

3. **Add the provider's package by using the `pkgadd` command.**

```
# pkgadd -d /path/to/package  pkginst
```

   The package must include software that has been signed by a certificate from Sun. To request a certificate from Sun and to sign a provider, see Appendix F, "Packaging and Signing Cryptographic Providers," in *Solaris Security for Developers Guide*.

   The package should have scripts that notify the cryptographic framework that another provider with a set of mechanisms is available. For information about the packaging requirements, see Appendix F, "Packaging and Signing Cryptographic Providers," in *Solaris Security for Developers Guide*.

4. **Refresh the providers.**

   You need to refresh providers if you added a software provider, or if you added hardware and specified policy for the hardware.

```
# svcadm refresh svc:/system/cryptosvc
```

5 **Locate the new provider on the list.**

In this case, a new kernel software provider was installed.

```
# cryptoadm list
...
kernel software providers:
    des
    aes
    blowfish
    arcfour
    sha1
    md5
    rsa
    swrand
    ecc     <-- added provider
...
```

Example 14–16 Adding a User-Level Software Provider

In the following example, a signed PKCS #11 library is installed.

```
# pkgadd -d /cdrom/cdrom0/SolarisNew
    Answer the prompts
# svcadm refresh system/cryptosvc
# cryptoadm list
user-level providers:
==========================
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so
    /opt/SUNWconn/lib/$ISA/libpkcs11.so.1     <-- added provider
```

Developers who are testing a library with the cryptographic framework can install the library manually.

```
# cryptoadm install provider=/opt/SUNWconn/lib/'$ISA'/libpkcs11.so.1
```

▼ How to Prevent the Use of a User-Level Mechanism

If some of the cryptographic mechanisms from a library provider should not be used, you can remove selected mechanisms. This procedure uses the DES mechanisms in the `pkcs11_softtoken` library as an example.

1 **Become superuser or assume a role that includes the Crypto Management rights profile.**

To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.

2. **List the mechanisms that are offered by a particular user-level software provider.**

```
% cryptoadm list -m provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
CKM_AES_CBC,CKM_AES_CBC_PAD,CKM_AES_ECB,CKM_AES_KEY_GEN,
...
```

3. **List the mechanisms that are available for use.**

```
$ cryptoadm list -p
user-level providers:
=====================
...
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
...
```

4. **Disable the mechanisms that should not be used.**

```
$ cryptoadm disable provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so \
> mechanism=CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB
```

5. **List the mechanisms that are available for use.**

```
$ cryptoadm list -p provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

Example 14–17 Enabling a User-Level Software Provider Mechanism

In the following example, a disabled DES mechanism is again made available for use.

```
$ cryptoadm list -m provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so:
CKM_DES_CBC,CKM_DES_CBC_PAD,CKM_DES_ECB,CKM_DES_KEY_GEN,
CKM_DES3_CBC,CKM_DES3_CBC_PAD,CKM_DES3_ECB,CKM_DES3_KEY_GEN,
...
$ cryptoadm list -p provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_ECB,CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
$ cryptoadm enable provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so \
> mechanism=CKM_DES_ECB
$ cryptoadm list -p provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled,
except CKM_DES_CBC_PAD,CKM_DES_CBC. random is enabled.
```

Example 14–18 Enabling All User-Level Software Provider Mechanisms

In the following example, all mechanisms from the user-level library are enabled.

```
$ cryptoadm enable provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so all
$ cryptoadm list -p provider=/usr/lib/security/'$ISA'/pkcs11_softtoken.so
/usr/lib/security/$ISA/pkcs11_softtoken.so: all mechanisms are enabled.
random is enabled.
```

Example 14–19 Permanently Removing User-Level Software Provider Availability

In the following example, the `libpkcs11.so.1` library is removed.

```
$ cryptoadm uninstall provider=/opt/SUNWconn/lib/'$ISA'/libpkcs11.so.1
$ cryptoadm list
user-level providers:
    /usr/lib/security/$ISA/pkcs11_kernel.so
    /usr/lib/security/$ISA/pkcs11_softtoken.so

kernel software providers:
...
```

▼ How to Prevent the Use of a Kernel Software Provider

If the cryptographic framework provides multiple modes of a provider such as AES, you might remove a slow mechanism from use, or a corrupted mechanism. This procedure uses the AES algorithm as an example.

1. **Become superuser or assume a role that includes the Crypto Management rights profile.**

   To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.

2. **List the mechanisms that are offered by a particular kernel software provider.**

   ```
   $ cryptoadm list -m provider=aes
   aes: CKM_AES_ECB,CKM_AES_CBC
   ```

3. **List the mechanisms that are available for use.**

   ```
   $ cryptoadm list -p provider=aes
   aes: all mechanisms are enabled.
   ```

4. **Disable the mechanism that should not be used.**

   ```
   $ cryptoadm disable provider=aes mechanism=CKM_AES_ECB
   ```

5 **List the mechanisms that are available for use.**

```
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
```

Example 14–20 Enabling a Kernel Software Provider Mechanism

In the following example, a disabled AES mechanism is again made available for use.

```
cryptoadm list -m provider=aes
aes: CKM_AES_ECB,CKM_AES_CBC
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled, except CKM_AES_ECB.
$ cryptoadm enable provider=aes mechanism=CKM_AES_ECB
$ cryptoadm list -p provider=aes
aes: all mechanisms are enabled.
```

Example 14–21 Temporarily Removing Kernel Software Provider Availability

In the following example, the AES provider is temporarily removed from use. The `unload` subcommand is useful to prevent a provider from being loaded automatically while the provider is being uninstalled. For example, the `unload` subcommand would be used when installing a patch that affects the provider.

```
$ cryptoadm unload provider=aes
$ cryptoadm list
...
kernel software providers:
         des
         aes (inactive)
         blowfish
         arcfour
         sha1
         md5
         rsa
         swrand
```

The AES provider is unavailable until the cryptographic framework is refreshed.

```
$ svcadm refresh system/cryptosvc
$ cryptoadm list
...
kernel software providers:
         des
         aes
         blowfish
         arcfour
         sha1
         md5
         rsa
         swrand
```

If a kernel consumer is using the kernel software provider, the software is not unloaded. An error message is displayed and the provider continues to be available for use.

Example 14–22 Permanently Removing Software Provider Availability

In the following example, the AES provider is removed from use. Once removed, the AES provider does not appear in the policy listing of kernel software providers.

```
$ cryptoadm uninstall provider=aes
$ cryptoadm list
...
kernel software providers:
         des
         blowfish
         arcfour
         sha1
         md5
         rsa
         swrand
```

If a kernel consumer is using the kernel software provider, an error message is displayed and the provider continues to be available for use.

Example 14–23 Reinstalling a Removed Kernel Software Provider

In the following example, the AES kernel software provider is reinstalled.

```
$ cryptoadm install provider=aes mechanism=CKM_AES_ECB,CKM_AES_CBC
$ cryptoadm list
...
kernel software providers:
         des
         aes
         blowfish
         arcfour
         sha1
         md5
         rsa
         swrand
```

▼ How to List Hardware Providers

Hardware providers are automatically located and loaded. For more information, see `driver.conf`(4) man page.

Before You Begin When you add hardware that expects to be used within the Solaris Cryptographic Framework, the hardware registers with the SPI in the kernel. The framework checks that the hardware driver is signed. Specifically, the framework checks that the object file of the driver is signed with a certificate that Sun issues.

1. **List the hardware providers that are available on the system.**

```
% cryptoadm list
...
kernel hardware providers:
   dca/0
```

2. **List the mechanisms that the board provides.**

```
% cryptoadm list -m provider=dca/0
dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM_DES_CBC, CKM_DES3_CBC
```

3. **List the mechanisms that are available for use on the board.**

```
% cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

▼ How to Disable Hardware Provider Mechanisms and Features

You can selectively disable mechanisms and the random number feature from a hardware provider. To enable them again, see Example 14–24.

1. **List the mechanisms and features that are available from the board.**

```
% cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

2. **Become superuser or assume a role that includes the Crypto Management rights profile.**

   To create a role that includes the Crypto Management rights profile and assign the role to a user, see Example 9–7.

3 **Choose the mechanisms or feature to disable:**

- **Disable selected mechanisms.**

  ```
  # cryptoadm list -m provider=dca/0
  dca/0: CKM_RSA_PKCS, CKM_RSA_X_509, CKM_DSA, CKM_DES_CBC, CKM_DES3_CBC
  random is enabled.
  # cryptoadm disable provider=dca/0 mechanism=CKM_DES_CBC,CKM_DES3_CBC
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled except CKM_DES_CBC,CKM_DES3_CBC.
  random is enabled.
  ```

- **Disable the random number generator.**

  ```
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is enabled.
  # cryptoadm disable provider=dca/0 random
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is disabled.
  ```

- **Disable all mechanisms. Do not disable the random number generator.**

  ```
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is enabled.
  # cryptoadm disable provider=dca/0 mechanism=all
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are disabled. random is enabled.
  ```

- **Disable every feature and mechanism on the hardware.**

  ```
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are enabled. random is enabled.
  # cryptoadm disable provider=dca/0 all
  # cryptoadm list -p provider=dca/0
  dca/0: all mechanisms are disabled. random is disabled.
  ```

Example 14–24 Enabling Mechanisms and Features on a Hardware Provider

In the following examples, disabled mechanisms on a piece of hardware are selectively enabled.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB,CKM_DES3_ECB.
random is enabled.
# cryptoadm enable provider=dca/0 mechanism=CKM_DES3_ECB
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled except CKM_DES_ECB. random is enabled.
```

In the following example, only the random generator is enabled.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.
# cryptoadm enable provider=dca/0 random
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is enabled.
```

In the following example, only the mechanisms are enabled. The random generator continues to be disabled.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_MD5,CKM_MD5_HMAC,....
random is disabled.
# cryptoadm enable provider=dca/0 mechanism=all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is disabled.
```

In the following example, every feature and mechanism on the board is enabled.

```
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled, except CKM_DES_ECB,CKM_DES3_ECB.
random is disabled.
# cryptoadm enable provider=dca/0 all
# cryptoadm list -p provider=dca/0
dca/0: all mechanisms are enabled. random is enabled.
```

▼ How to Refresh or Restart All Cryptographic Services

By default, the Solaris Cryptographic Framework is enabled. When the `kcfd` daemon fails for any reason, the service management facility can be used to restart cryptographic services. For more information, see the `smf`(5) and `svcadm`(1M) man pages. For the effect on zones of restarting cryptographic services, see “Cryptographic Services and Zones” on page 266.

1. **Check the status of cryptographic services.**

```
% svcs \*cryptosvc\*
 STATE          STIME    FMRI
offline         Dec_09   svc:/system/cryptosvc:default
```

2. **Become superuser or assume an equivalent role to enable cryptographic services.**

   Roles contain authorizations and privileged commands. For more information about roles, see “Configuring RBAC (Task Map)” on page 196.

```
# svcadm enable svc:/system/cryptosvc
```

Example 14–25 Refreshing Cryptographic Services

In the following example, cryptographic services are refreshed in the global zone. Therefore, kernel-level cryptographic policy in every non-global zone is also refreshed.

```
# svcadm refresh system/cryptosvc
```

◆ ◆ ◆ CHAPTER 15

Solaris Key Management Framework

Starting in the Solaris 10 8/07 release, the Key Management Framework (KMF) provides tools and programming interfaces for managing public key objects. Public key objects include X.509 certificates and public/private key pairs. The formats for storing these objects can vary. KMF also provides a tool for managing policies that define the use of X.509 certificates by applications.

- "Managing Public Key Technologies" on page 289
- "Key Management Framework Utilities" on page 290
- "Using the Key Management Framework (Tasks)" on page 291

Managing Public Key Technologies

The Key Management Framework (KMF) provides a unified approach to managing public key technologies (PKI). The Solaris OS has several different applications that make use of PKI technologies. Each application provides its own programming interfaces, key storage mechanisms, and administrative utilities. If an application provides a policy enforcement mechanism, the mechanism applies to that application only. With KMF, applications use a unified set of administrative tools, a single set of programming interfaces, and a single policy enforcement mechanism. These features manage the PKI needs of all applications that adopt these interfaces.

KMF unifies the management of public key technologies with the following interfaces:

- **pktool command** – This command manages PKI objects, such as certificates, in a variety of keystores.
- **kmfcfg command** – This command manages the PKI policy database.

  PKI policy decisions include operations such as the validation method for an operation. Also, PKI policy can limit the scope of a certificate. For example, PKI policy might assert that a certificate can be used only for specific purposes. Such a policy would prevent that certificate from being used for other requests.

- **KMF library –** This library contains programming interfaces that abstract the underlying keystore mechanism.

  Applications do not have to choose one particular keystore mechanism, but can migrate from one mechanism to another mechanism. The supported keystores are PKCS #11, NSS, and OpenSSL. The library includes a pluggable framework so that new keystore mechanisms can be added. Therefore, applications that use the new mechanisms would require only minor modifications to use a new keystore.

Key Management Framework Utilities

KMF provides methods for managing the storage of keys and provides the overall policy for the use of those keys. KMF manages the policy, keys, and certificates for three public key technologies:

- Tokens from PKCS #11 providers, that is, from the Solaris Cryptographic Framework
- NSS, that is, Network Security Services
- OpenSSL, a file-based keystore

The `kmfcfg` tool can create, modify, or delete KMF policy entries. KMF manages keystores through the `pktool` command. For more information, see the `kmfcfg`(1) and `pktool`(1) man pages, and the following sections.

KMF Policy Management

KMF policy is stored in a database. This policy database is accessed internally by all applications that use the KMF programming interfaces. The database can constrain the use of the keys and certificates that are managed by the KMF library. When an application attempts to verify a certificate, the application checks the policy database. The `kmfcfg` command modifies the policy database.

KMF Keystore Management

KMF manages the keystores for three public key technologies, PKCS #11 tokens, NSS, and OpenSSL. For all of these technologies, the `pktool` command enables you to do the following:

- Designate a keystore for each certificate and PIN.
- Generate a self-signed certificate.
- Generate a certificate request.
- Import objects into the keystore.
- List the objects in the keystore.
- Delete objects from the keystore.

- Download a CRL.

For the PKCS #11 and NSS technologies, the `pktool` command also enables you to set a PIN by generating a passphrase:

- Generate a passphrase for the keystore.
- Generate a passphrase for an object in the keystore.

For examples of using the `pktool` utility, see the `pktool(1)` man page and "Using the Key Management Framework (Task Map)" on page 291.

Using the Key Management Framework (Task Map)

The Key Management Framework (KMF) enables you to centrally manage public key technologies.

Task	Description	For Instructions
Create a certificate.	Creates a certificate for use by PKCS #11, NSS, or SSL.	"How to Create a Certificate by Using the `pktool gencert` Command" on page 291
Export a certificate.	Creates a file with the certificate and its supporting keys. The file can be protected with a password.	"How to Export a Certificate and Private Key in PKCS #12 Format" on page 294
Import a certificate.	Imports a certificate from another system.	"How to Import a Certificate Into Your Keystore" on page 293
	Imports a certificate in PKCS #12 format from another system.	Example 15–2
Generate a passphrase.	Generates a passphrase for access to a PKCS #11 keystore or an NSS keystore.	"How to Generate a Passphrase by Using the `pktool setpin` Command" on page 295

Using the Key Management Framework (Tasks)

This section describes how to use the `pktool` command to manage your public key objects, such as passwords, passphrases, files, keystores, certificates, and CRLs.

▼ How to Create a Certificate by Using the `pktool gencert` Command

This procedure creates a self-signed certificate and stores the certificate in the PKCS #11 keystore. As a part of this operation, an RSA public/private key pair is also created. The private key is stored in the keystore with the certificate.

1 Generate a self-signed certificate.

```
% pktool gencert [keystore=keystore] label=label-name \
subject=subject-DN serial=hex-serial-number
```

`keystore=`*keystore*	Specifies the keystore by type of public key object. The value can be `nss`, `pkcs11`, or `ssl`. This keyword is optional.
`label=`*label-name*	Is a unique name that the issuer gives to the certificate.
`subject=`*subject-DN*	Is the distinguished name for the certificate.
`serial=`*hex-serial-number*	Is the serial number in hexadecimal format. The issuer of the certificate chooses the number, such as `0x0102030405`.

2 Verify the contents of the keystore.

```
% pktool list
Found number certificates.
1. (X.509 certificate)
      Label:   label-name
      ID: Fingerprint that binds certificate to private key
      Subject: subject-DN
      Issuer:   distinguished-name
      Serial:   hex-serial-number
n. ...
```

This command lists all certificates in the keystore. In the following example, the keystore contains one certificate only.

Example 15–1 Creating a Self-Signed Certificate by Using `pktool`

In the following example, a user at `My Company` creates a self-signed certificate and stores the certificate in a keystore for PKCS #11 objects. The keystore is initially empty. If the keystore has not been initialized, the PIN for the softtoken is `changeme`.

```
% pktool gencert keystore=pkcs11 label="My Cert" \
subject="C=US, O=My Company, OU=Security Engineering Group, CN=MyCA" \
serial=0x000000001
Enter pin for Sun Software PKCS#11 softtoken:     Type PIN for token

% pktool list
Found 1 certificates.
1. (X.509 certificate)
      Label: My Cert
      ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
      Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Serial: 0x01
```

▼ How to Import a Certificate Into Your Keystore

This procedure describes how to import a file with PKI information that is encoded with PEM or with raw DER into your keystore. For an export procedure, see Example 15–4.

1. **Import the certificate.**

   ```
   % pktool import keystore=keystore infile=infile-name label=label-name
   ```

2. **If you are importing private PKI objects, provide passwords when prompted.**

   a. **At the prompt, provide the password for the file.**

   If you are importing PKI information that is private, such as an export file in PKCS #12 format, the file requires a password. The creator of the file that you are importing provides you with the PKCS #12 password.

   ```
   Enter password to use for accessing the PKCS12 file:     Type PKCS #12 password
   ```

   b. **At the prompt, type the password for your keystore.**

   ```
   Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
   ```

3. **Verify the contents of the keystore.**

   ```
   % pktool list
   Found number certificates.
   1. (X.509 certificate)
         Label:   label-name
         ID:  Fingerprint that binds certificate to private key
         Subject: subject-DN
         Issuer:   distinguished-name
         Serial:   hex-serial-number
   2. ...
   ```

Example 15–2 Importing a PKCS #12 File Into Your Keystore

In the following example, the user imports a PKCS #12 file from a third party. The `pktool import` command extracts the private key and the certificate from the `gracedata.p12` file, and stores them in the user's preferred keystore.

```
% pktool import keystore=pkcs11 infile=gracedata.p12 label=GraceCert
Enter password to use for accessing the PKCS12 file:     Type PKCS #12 password
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Found 1 certificate(s) and 1 key(s) in gracedata.p12
% pktool list
Found 1 certificates.
1. (X.509 certificate)
        Label: GraceCert
```

```
        ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
        Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
        Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
        Serial: 0x01
```

Example 15–3 Importing an X.509 Certificate Into Your Keystore

In the following example, the user imports an X.509 certificate in PEM format into the user's preferred keystore. This public certificate is not protected with a password. The user's public keystore is also not protected by a password.

```
% pktool import keystore=pkcs11 infile=somecert.pem label="TheirCompany Root Cert"
% pktool list
Found 1 certificates.
1. (X.509 certificate)
        Label: TheirCompany Root Cert
        ID: 21:ae:83:98:24:d1:1f:cb:65:5b:48:75:7d:02:47:cf:98:1f:ec:a0
        Subject: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
        Issuer: C=US, O=TheirCompany, OU=Security, CN=TheirCompany Root CA
        Serial: 0x01
```

▼ How to Export a Certificate and Private Key in PKCS #12 Format

You can create a file in PKCS #12 format to export private keys and their associated X.509 certificate to other systems. Access to the file is protected by a password.

1. **Find the certificate to export.**

```
% pktool list
Found number certificates.
1. (X.509 certificate)
      Label:   label-name
      ID: Fingerprint that binds certificate to private key
      Subject: subject-DN
      Issuer:   distinguished-name
      Serial:   hex-serial-number
2. ...
```

2. **Export the keys and certificate.**

   Use the keystore and label from the `pktool list` command. Provide a file name for the export file. When the name contains a space, surround the name with double quotes.

```
% pktool export keystore=keystore outfile=outfile-name label=label-name
```

3 **Protect the export file with a password.**

At the prompt, type the current password for the keystore. At this point, you create a password for the export file. The receiver must provide this password when importing the file.

```
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:     Create PKCS #12 password
```

Tip – Send the password separately from the export file. Best practice suggests that you provide the password out of band, such as during a telephone call.

Example 15–4 Exporting a Certificate and Private Key in PKCS #12 Format

In the following example, a user exports the private keys with their associated X.509 certificate into a standard PKCS #12 file. This file can be imported into other keystores. The PKCS #11 password protects the source keystore. The PKCS #12 password is used to protect private data in the PKCS #12 file. This password is required to import the file.

```
% pktool list
Found 1 certificates.
1. (X.509 certificate)
      Label: My Cert
      ID: 12:82:17:5f:80:78:eb:44:8b:98:e3:3c:11:c0:32:5e:b6:4c:ea:eb
      Subject: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Issuer: C=US, O=My Company, OU=Security Engineering Group, CN=MyCA
      Serial: 0x01

% pktool export keystore=pkcs11 outfile=mydata.p12 label="My Cert"
Enter pin for Sun Software PKCS#11 softtoken:      Type PIN for token
Enter password to use for accessing the PKCS12 file:     Create PKCS #12 password
```

The user then telephones the recipient and provides the PKCS #12 password.

▼ How to Generate a Passphrase by Using the `pktool setpin` Command

You can generate a passphrase for an object in a keystore, and for the keystore itself. The passphrase is required to access the object or keystore. For an example of generating a passphrase for an object in a keystore, see Example 15–4.

1 **Generate a passphrase for access to a keystore.**

```
% pktool setpin keystore=nss|pkcs11 dir=directory
```

2 **Answer the prompts.**

If the keystore does not have a password already set, press the Return key to create the password.

```
Enter current token passphrase:     Press the Return key
Create new passphrase:     Type the passphrase that you want to use
Re-enter new passphrase:     Retype the passphrase
Passphrase changed.
```

The keystore is now protected by *passphrase*. If you lose the passphrase, you lose access to the objects in the keystore.

Example 15–5 Protecting a Keystore With a Passphrase

The following example shows how to set the passphrase for an NSS database. Because no passphrase has been created, the user presses the Return key at the first prompt.

```
% pktool setpin keystore=nss dir=/var/nss
Enter current token passphrase:     Press the Return key
Create new passphrase:    has8n0NdaH
Re-enter new passphrase:  has8n0NdaH
Passphrase changed.
```

PART V

Authentication Services and Secure Communication

This section discusses authentication services that can be configured on a non-networked system, or between two systems. To configure a network of authenticated users and systems, see Part VI.

◆ ◆ ◆ CHAPTER 16

Using Authentication Services (Tasks)

This chapter provides information about how to use Secure RPC to authenticate a host and a user across an NFS mount. The following is a list of the topics in this chapter.

- "Overview of Secure RPC" on page 299
- "Administering Secure RPC (Task Map)" on page 304

Overview of Secure RPC

Secure RPC (Remote Procedure Call) protects remote procedures with an authentication mechanism. The Diffie-Hellman authentication mechanism authenticates both the host and the user who is making a request for a service. The authentication mechanism uses Data Encryption Standard (DES) encryption. Applications that use Secure RPC include NFS and the name services, NIS and NIS+.

NFS Services and Secure RPC

NFS enables several hosts to share files over the network. Under the NFS service, a server holds the data and resources for several clients. The clients have access to the file systems that the server shares with the clients. Users who are logged in to the client systems can access the file systems by mounting the file systems from the server. To the user on the client system, it appears as if the files are local to the client. One of the most common uses of NFS allows systems to be installed in offices, while storing all user files in a central location. Some features of the NFS service, such as the `-nosuid` option to the `mount` command, can be used to prohibit the opening of devices and file systems by unauthorized users.

The NFS service uses Secure RPC to authenticate users who make requests over the network. This process is known as *Secure NFS*. The Diffie-Hellman authentication mechanism, `AUTH_DH`, uses DES encryption to ensure authorized access. The `AUTH_DH` mechanism has also been called `AUTH_DES`. For more information, see the following:

- To set up and administer Secure NFS, see "Administering the Secure NFS System" in *System Administration Guide: Network Services*.
- To set up the NIS+ tables and enter names in the `cred` table, see *System Administration Guide: Naming and Directory Services (NIS+)*.
- For an outline of the transactions that are involved in RPC authentication, see "Implementation of Diffie-Hellman Authentication" on page 301.

DES Encryption With Secure NFS

The Data Encryption Standard (DES) encryption functions use a 56-bit key to encrypt data. If two credential users or principals know the same DES key, they can communicate in private by using the key to encipher and decipher text. DES is a relatively fast encryption mechanism. A DES chip makes the encryption even faster. However, if the chip is not present, a software implementation is substituted.

The risk of using just the DES key is that an intruder can collect enough cipher-text messages that were encrypted with the same key to be able to discover the key and decipher the messages. For this reason, security systems such as Secure NFS need to change the keys frequently.

Kerberos Authentication

Kerberos is an authentication system that was developed at MIT. Some encryption in Kerberos is based on DES. Kerberos V4 support is no longer supplied as part of Secure RPC. However, a client-side and server-side implementation of Kerberos V5, which uses RPCSEC_GSS, is included with this release. For more information, see Chapter 21, "Introduction to the Kerberos Service."

Diffie-Hellman Authentication and Secure RPC

The Diffie-Hellman (DH) method of authenticating a user is nontrivial for an intruder to crack. The client and the server have their own private key, which they use with the public key to devise a common key. The private key is also known as the *secret key*. The client and the server use the common key to communicate with each other. The common key is encrypted with an agreed-upon encryption function, such as DES.

Authentication is based on the ability of the sending system to use the common key to encrypt the current time. Then, the receiving system can decrypt and check against its current time. The time on the client and the server must be synchronized. For more information, see “Managing Network Time Protocol (Tasks)” in *System Administration Guide: Network Services*.

The public keys and private keys are stored in an NIS or NIS+ database. NIS stores the keys in the `publickey` map. NIS+ stores the keys in the `cred` table. These files contain the public key and the private key for all potential users.

The system administrator is responsible for setting up NIS maps or NIS+ tables, and for generating a public key and a private key for each user. The private key is stored in encrypted form with the user's password. This process makes the private key known only to the user.

Implementation of Diffie-Hellman Authentication

This section describes the series of transactions in a client-server session that use Diffie-Hellman authentication (`AUTH_DH`).

Generating the Public Keys and Secret Keys for Secure RPC

Sometime prior to a transaction, the administrator runs either the `newkey` or the `nisaddcred` command to generate a public key and a secret key. Each user has a unique public key and secret key. The public key is stored in a public database. The secret key is stored in encrypted form in the same database. The `chkey` command changes the key pair.

Running the `keylogin` Command for Secure RPC

Normally, the login password is identical to the Secure RPC password. In this case, the `keylogin` command is not required. However, if the passwords are different, the users have to log in and then run the `keylogin` command.

The `keylogin` command prompts the user for a Secure RPC password. The command then uses the password to decrypt the secret key. The `keylogin` command then passes the decrypted secret key to the *keyserver* program. The keyserver is an RPC service with a local instance on every computer. The keyserver saves the decrypted secret key and waits for the user to initiate a Secure RPC transaction with a server.

If both the login password and the RPC password are the same, the login process passes the secret key to the keyserver. If the passwords are required to be different, then the user must always run the `keylogin` command. When the `keylogin` command is included in the user's environment configuration file, such as the `~/.login`, `~/.cshrc`, or `~/.profile` file, the `keylogin` command runs automatically whenever the user logs in.

Generating the Conversation Key for Secure RPC

When the user initiates a transaction with a server, the following occurs:

1. The keyserver randomly generates a conversation key.
2. The kernel uses the conversation key, plus other material, to encrypt the client's timestamp.
3. The keyserver looks up the server's public key in the public key database. For more information, see the `publickey(4)` man page.
4. The keyserver uses the client's secret key and the server's public key to create a common key.
5. The keyserver encrypts the conversation key with the common key.

Initially Contacting the Server in Secure RPC

The transmission, which includes the encrypted timestamp and the encrypted conversation key, is then sent to the server. The transmission includes a credential and a verifier. The credential contains three components:

- The client's network name
- The conversation key, which is encrypted with the common key
- A “window,” which is encrypted with the conversation key

The window is the difference in time that the client says should be allowed between the server's clock and the client's timestamp. If the difference between the server's clock and the timestamp is greater than the window, the server rejects the client's request. Under normal circumstances, this rejection does not happen, because the client first synchronizes with the server before starting the RPC session.

The client's verifier contains the following:

- The encrypted timestamp
- An encrypted verifier of the specified window, which is decremented by 1

The window verifier is needed in case somebody wants to impersonate a user. The impersonator can write a program that, instead of filling in the encrypted fields of the credential and verifier, just inserts random bits. The server decrypts the conversation key into some random key. The server then uses the key to try to decrypt the window and the timestamp. The result is random numbers. After a few thousand trials, however, the random window/timestamp pair is likely to pass the authentication system. The window verifier lessens the chance that a fake credential could be authenticated.

Decrypting the Conversation Key in Secure RPC

When the server receives the transmission from the client, the following occurs:

1. The keyserver that is local to the server looks up the client's public key in the public key database.
2. The keyserver uses the client's public key and the server's secret key to deduce the common key. The common key is the same common key that is computed by the client. Only the server and the client can calculate the common key because the calculation requires knowing one of the secret keys.
3. The kernel uses the common key to decrypt the conversation key.
4. The kernel calls the keyserver to decrypt the client's timestamp with the decrypted conversation key.

Storing Information on the Server in Secure RPC

After the server decrypts the client's timestamp, the server stores four items of information in a credential table:

- The client's computer name
- The conversation key
- The window
- The client's timestamp

The server stores the first three items for future use. The server stores the client's timestamp to protect against replays. The server accepts only timestamps that are chronologically greater than the last timestamp seen. As a result, any replayed transactions are guaranteed to be rejected.

Note – Implicit in these transactions is the name of the caller, who must be authenticated in some manner. The keyserver cannot use DES authentication to authenticate the caller because the use of DES by the keyserver would create a deadlock. To avoid a deadlock, the keyserver stores the secret keys by user ID (UID) and grants requests only to local `root` processes.

Returning the Verifier to the Client in Secure RPC

The server returns a verifier to the client, which includes the following:

- The index ID, which the server records in its credential cache
- The client's timestamp minus 1, which is encrypted by the conversation key

The reason for subtracting 1 from the client's timestamp is to ensure that the timestamp is out of date. An out-of-date timestamp cannot be reused as a client verifier.

Authenticating the Server in Secure RPC

The client receives the verifier and authenticates the server. The client knows that only the server could have sent the verifier because only the server knows what timestamp the client sent.

Handling Transactions in Secure RPC

With every transaction after the first transaction, the client returns the index ID to the server in its next transaction. The client also sends another encrypted timestamp. The server sends back the client's timestamp minus 1, which is encrypted by the conversation key.

Administering Secure RPC (Task Map)

The following task map points to procedures that configure Secure RPC for NIS, NIS+, and NFS.

Task	Description	For Instructions
1. Start the keyserver.	Ensures that keys can be created so that users can be authenticated.	"How to Restart the Secure RPC Keyserver" on page 305
2. Set up credentials on an NIS+ host.	Ensures that the `root` user on a host can be authenticated in an NIS+ environment.	"How to Set Up a Diffie-Hellman Key for an NIS+ Host" on page 305
3. Give an NIS+ user a key.	Enables a user to be authenticated in an NIS+ environment.	"How to Set Up a Diffie-Hellman Key for an NIS+ User" on page 306
4. Set up credentials on an NIS host.	Ensures that the `root` user on a host can be authenticated in an NIS environment.	"How to Set Up a Diffie-Hellman Key for an NIS Host" on page 307
5. Give an NIS user a key.	Enables a user to be authenticated in an NIS environment.	"How to Set Up a Diffie-Hellman Key for an NIS User" on page 308
6. Share NFS files with authentication.	Enables an NFS server to securely protect shared file systems using authentication.	"How to Share NFS Files With Diffie-Hellman Authentication" on page 309

Administering Authentication With Secure RPC

By requiring authentication for use of mounted NFS file systems, you increase the security of your network.

▼ How to Restart the Secure RPC Keyserver

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Verify that the `keyserv` daemon is running.**

   ```
   # svcs \*keyserv\*
   STATE    STIME   FMRI
   disabled Dec_14  svc:/network/rpc/keyserv
   ```

3. **Enable the keyserver service if the service is not online.**

   ```
   # svcadm enable network/rpc/keyserv
   ```

▼ How to Set Up a Diffie-Hellman Key for an NIS+ Host

This procedure should be done on every host in the NIS+ domain. After `root` has run the `keylogin` command, the server has GSS-API acceptor credentials for `mech_dh` and the client has GSS-API initiator credentials.

For a detailed description of NIS+ security, see *System Administration Guide: Naming and Directory Services (NIS+)*.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Enable the `publickey` table in the name service.**

   Add the following line to the `/etc/nsswitch.conf` file:

   ```
   publickey: nisplus
   ```

3. **Initialize the NIS+ client.**

   ```
   # nisinit -cH hostname
   ```

   where *hostname* is the name of a trusted NIS+ server that contains an entry in its tables for the client system.

4 **Add the client to the `cred` table.**

Type the following commands:

```
# nisaddcred local
# nisaddcred des
```

5 **Verify the setup by using the `keylogin` command.**

If you are prompted for a password, the procedure has succeeded.

```
# keylogin
Password:
```

Example 16–1 Setting Up a New Key for `root` on an NIS+ Client

The following example uses the host `pluto` to set up `earth` as an NIS+ client. You can ignore the warnings. The `keylogin` command is accepted, verifying that `earth` is correctly set up as a secure NIS+ client.

```
# nisinit -cH pluto
NIS Server/Client setup utility.
This system is in the example.com. directory.
Setting up NIS+ client ...
All done.
# nisaddcred local
# nisaddcred des
DES principal name : unix.earth@example.com
Adding new key for unix.earth@example.com (earth.example.com.)
Network password:    <Type password>
Warning, password differs from login password.
Retype password:     <Retype password>
# keylogin
Password:            <Type password>
#
```

▼ How to Set Up a Diffie-Hellman Key for an NIS+ User

This procedure should be done on every user in the NIS+ domain.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Add the user to the `cred` table on the root master server.**

Type the following command:

```
# nisaddcred -p unix.UID@domain-name -P username.domain-name. des
```

Note that, in this case, the *username.domain-name* must end with a dot (`.`).

3 **Verify the setup by logging in as the client and typing the `keylogin` command.**

Example 16–2 Setting Up a New Key for an NIS+ User

In the following example, a key for Diffie-Hellman authentication is given to the user `jdoe`.

```
# nisaddcred -p unix.1234@example.com -P jdoe.example.com. des
DES principal name : unix.1234@example.com
Adding new key for unix.1234@example.com (jdoe.example.com.)
Password:            <Type password>
Retype password:     <Retype password>
# rlogin rootmaster -l jdoe
% keylogin
Password:            <Type password>
%
```

▼ How to Set Up a Diffie-Hellman Key for an NIS Host

This procedure should be done on every host in the NIS domain.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Enable the `publickey` map in the name service.**

Add the following line to the `/etc/nsswitch.conf` file:

```
publickey: nis
```

3 **Create a new key pair by using the `newkey` command.**

```
# newkey -h hostname
```

where *hostname* is the name of the client.

Example 16–3 Setting Up a New Key for `root` on an NIS Client

In the following example, `earth` is set up as a secure NIS client.

```
# newkey -h earth
Adding new key for unix.earth@example.com
New Password:        <Type password>
Retype password:     <Retype password>
```

```
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

▼ How to Set Up a Diffie-Hellman Key for an NIS User

This procedure should be done for every user in the NIS domain.

Before You Begin Only system administrators, when logged in to the NIS master server, can generate a new key for a user.

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Create a new key for a user.

```
# newkey -u username
```

where *username* is the name of the user. The system prompts for a password. You can type a generic password. The private key is stored in an encrypted form by using the generic password.

3 Tell the user to log in and type the `chkey -p` command.

This command allows users to re-encrypt their private keys with a password known only to the user.

Note – The `chkey` command can be used to create a new key pair for a user.

Example 16–4 Setting Up and Encrypting a New User Key in NIS

In this example, superuser sets up the key.

```
# newkey -u jdoe
Adding new key for unix.12345@example.com
New Password:       <Type password>
Retype password:    <Retype password>
Please wait for the database to get updated...
Your new key has been successfully stored away.
#
```

Then the user `jdoe` re-encrypts the key with a private password.

```
% chkey -p
Updating nis publickey database.
Reencrypting key for unix.12345@example.com
Please enter the Secure-RPC password for jdoe:    <Type password>
Please enter the login password for jdoe:         <Type password>
Sending key change request to centralexample...
```

▼ How to Share NFS Files With Diffie-Hellman Authentication

This procedure protects shared file systems on an NFS server by requiring authentication for access.

Before You Begin Diffie-Hellman public key authentication must be enabled on the network. To enable authentication on the network, do one of the following:

- "How to Set Up a Diffie-Hellman Key for an NIS+ Host" on page 305
- "How to Set Up a Diffie-Hellman Key for an NIS Host" on page 307

1. **Become superuser or assume a role that includes the System Management profile.**

   The System Administrator role includes the System Management profile. To create the role and assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2. **On the NFS server, share a file system with Diffie-Hellman authentication.**

   ```
   # share -F nfs -o sec=dh /filesystem
   ```

   where *filesystem* is the file system that is being shared.

   The `-o sec=dh` option means that `AUTH_DH` authentication is now required to access the file system.

3. **On an NFS client, mount a file system with Diffie-Hellman authentication.**

   ```
   # mount -F nfs -o sec=dh server:filesystem mount-point
   ```

server	Is the name of the system that is sharing *filesystem*
filesystem	Is the name of the file system that is being shared, such as `opt`
mount-point	Is the name of the mount point, such as `/opt`

The `-o sec=dh` option mounts the file system with `AUTH_DH` authentication.

◆ ◆ ◆ CHAPTER 17

Using PAM

This chapter covers the Pluggable Authentication Module (PAM) framework. PAM provides a method to “plug in” authentication services into the Solaris Operating System (Solaris OS). PAM provides support for multiple authentication services when accessing a system.

- “PAM (Overview)” on page 311
- “PAM (Tasks)” on page 314
- “PAM Configuration (Reference)” on page 317

PAM (Overview)

The Pluggable Authentication Module (PAM) framework lets you “plug in” new authentication services without changing system entry services, such as `login`, `ftp`, and `telnet`. You can also use PAM to integrate UNIX login with other security mechanisms such as Kerberos. Mechanisms for account, credential, session, and password management can also be “plugged in” by using this framework.

Benefits of Using PAM

The PAM framework enables you to configure the use of system entry services (such as, `ftp`, `login`, `telnet`, or `rsh`) for user authentication. Some benefits that PAM provides are as follows:

- Flexible configuration policy
  - Per-application authentication policy
  - The ability to choose a default authentication mechanism
  - The ability to require multiple authorizations on high-security systems
- Ease of use for the end user
  - No retyping of passwords if the passwords are the same for different authentication services

  - The ability to prompt the user for passwords for multiple authentication services without requiring the user to type multiple commands
- The ability to pass optional options to the user authentication services
- The ability to implement a site-specific security policy without having to change the system entry services

Introduction to the PAM Framework

The PAM framework consists of four parts:

- PAM consumers
- PAM library
- The `pam.conf`(4) configuration file
- PAM service modules, also referred to as providers

The framework provides a uniform way for authentication-related activities to take place. This approach enables application developers to use PAM services without having to know the semantics of the policy. Algorithms are centrally supplied. The algorithms can be modified independently of the individual applications. With PAM, administrators can tailor the authentication process to the needs of a particular system without having to change any applications. Adjustments are made through `pam.conf`, the PAM configuration file.

The following figure illustrates the PAM architecture. Applications communicate with the PAM library through the PAM application programming interface (API). PAM modules communicate with the PAM library through the PAM service provider interface (SPI). Thus, the PAM library enables applications and modules to communicate with each other.

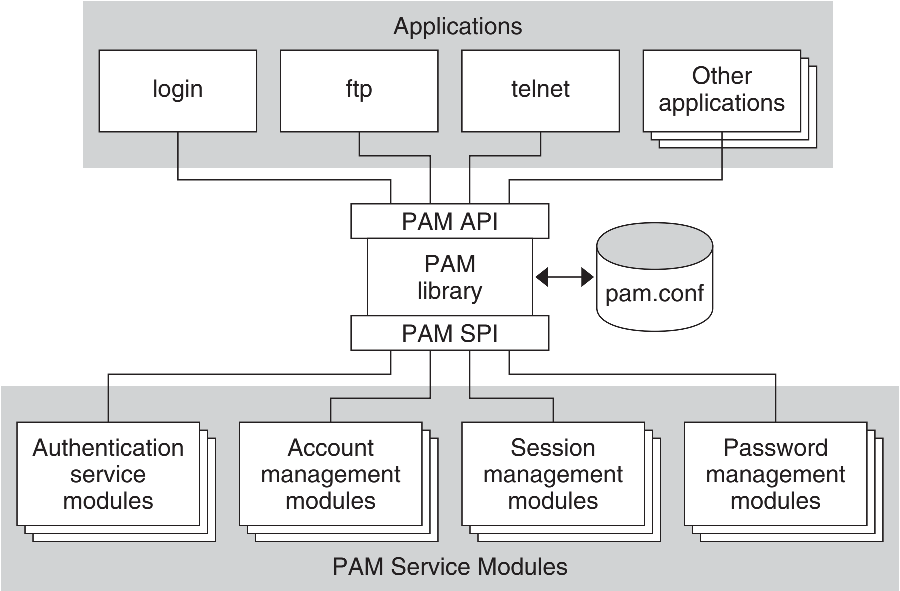

Applications
login
ftp
telnet
Other applications
PAM API
PAM library
PAM SPI
pam.conf
Authentication service modules
Account management modules
Session management modules
Password management modules
PAM Service Modules

FIGURE 17–1 PAM Architecture

Changes to PAM for the Solaris 10 Release

The Solaris 10 release includes the following changes to the Pluggable Authentication Module (PAM) framework:

- The `pam_authtok_check` module now allows for strict password checking using new tunable parameters in the `/etc/default/passwd` file. The new parameters define:
  - A list of comma separated dictionary files used for checking common dictionary words in a password
  - The minimum differences required between a new password and an old password
  - The minimum number of alphabetic or nonalphabetic characters that must be used in a new password
  - The minimum number of uppercase or lowercase letters that must be used in a new password
  - The number of allowable consecutive repeating characters
- The `pam_unix_auth` module implements account locking for local users. Account locking is enabled by the `LOCK_AFTER_RETRIES` parameter in `/etc/security/policy.conf` and the `lock_after-retries` key in `/etc/user_attr`. See the `policy.conf`(4) and the `user_attr`(4) man pages for more information.
- A new `binding` control flag has been defined. This control flag is documented in the `pam.conf`(4) man page and in "How PAM Stacking Works" on page 318.

- The `pam_unix` module has been removed and replaced by a set of service modules of equivalent or greater functionality. Many of these modules were introduced in the Solaris 9 release. Here is a list of the replacement modules:
  - `pam_authtok_check`
  - `pam_authtok_get`
  - `pam_authtok_store`
  - `pam_dhkeys`
  - `pam_passwd_auth`
  - `pam_unix_account`
  - `pam_unix_auth`
  - `pam_unix_cred`
  - `pam_unix_session`
- The functionality of the `pam_unix_auth` module has been split into two modules. The `pam_unix_auth` module now verifies that the password is correct for the user. The new `pam_unix_cred` module provides functions that establish user credential information.
- Additions to the `pam_krb5` module have been made to manage the Kerberos credentials cache using the PAM framework.
- A new `pam_deny` module has been added. The module can be used to deny access to services. By default, the `pam_deny` module is not used. For more information, see the `pam_deny`(5) man page.

PAM (Tasks)

This section discusses some tasks that might be required to make the PAM framework use a particular security policy. You should be aware of some security issues that are associated with the PAM configuration file. For information about the security issues, see "Planning for Your PAM Implementation" on page 315.

PAM (Task Map)

Task	Description	For Instructions
Plan for your PAM installation.	Consider configuration issues and make decisions about them before you start the software configuration process.	"Planning for Your PAM Implementation" on page 315
Add new PAM modules.	Sometimes, site-specific modules must be written and installed to cover requirements that are not part of the generic software. This procedure explains how to install these new PAM modules.	"How to Add a PAM Module" on page 316

Task	Description	For Instructions
Block access through `~/.rhosts`.	Further increase security by preventing access through `~/.rhosts`.	“How to Prevent Rhost-Style Access From Remote Systems With PAM” on page 316
Initiate error logging.	Start the logging of PAM error messages through `syslog`.	“How to Log PAM Error Reports” on page 317

Planning for Your PAM Implementation

As delivered, the `pam.conf` configuration file implements the standard Solaris security policy. This policy should work in many situations. If you need to implement a different security policy, here are the issues that you should focus on:

- Determine what your needs are, especially which PAM service modules you should select.
- Identify the services that need special configuration options. Use `other` if appropriate.
- Decide the order in which the modules should be run.
- Select the control flag for each module. See “How PAM Stacking Works” on page 318 for more information about all of the control flags.
- Choose any options that are necessary for each module. The man page for each module should list any special options.

Here are some suggestions to consider before you change the PAM configuration file:

- Use `other` entries for each module type so that every application does not have to be included in `/etc/pam.conf`.
- Make sure to consider the security implications of the `binding`, `sufficient`, and `optional` control flags.
- Review the man pages that are associated with the modules. These man pages can help you understand how each module functions, what options are available, and the interactions between stacked modules.

Caution – If the PAM configuration file is misconfigured or the file becomes corrupted, no user might be able to log in. Because the `sulogin` command does not use PAM, the root password would then be required to boot the machine into single-user mode and fix the problem.

After you change the `/etc/pam.conf` file, review the file as much as possible while you still have system access to correct problems. Test all the commands that might have been affected by your changes. An example is adding a new module to the `telnet` service. In this example, you would use the `telnet` command and verify that your changes make the service behave as expected.

▼ How to Add a PAM Module

This procedure shows how to add a new PAM module. New modules can be created to cover site-specific security policies or to support third party applications.

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 196.

2. **Determine which control flags and which other options should be used.**

   Refer to "How PAM Stacking Works" on page 318 for information on the control flags.

3. **Ensure that the ownership and permissions are set so that the module file is owned by `root` and the permissions are `555`.**

4. **Edit the PAM configuration file, `/etc/pam.conf`, and add this module to the appropriate services.**

5. **Verify that the module has been added properly.**

   You must test *before* the system is rebooted in case the configuration file is misconfigured. Login using a direct service, such as `ssh`, and run the `su` command, before you reboot the system. The service might be a daemon that is spawned only once when the system is booted. Then, you must reboot the system before you can verify that the module has been added.

▼ How to Prevent Rhost-Style Access From Remote Systems With PAM

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 196.

2. **Remove all of the lines that include `rhosts_auth.so.1` from the PAM configuration file.**

   This step prevents the reading of the `~/.rhosts` files during an `rlogin` session. Therefore, this step prevents unauthenticated access to the local system from remote systems. All `rlogin` access requires a password, regardless of the presence or contents of any `~/.rhosts` or `/etc/hosts.equiv` files.

3. **Disable the `rsh` service.**

   To prevent other unauthenticated access to the `~/.rhosts` files, remember to disable the `rsh` service.

   ```
   # svcadm disable network/shell
   ```

▼ How to Log PAM Error Reports

1. **Become superuser or assume an equivalent role.**

   Roles contain authorizations and privileged commands. For more information about roles, see "Configuring RBAC (Task Map)" on page 196.

2. **Configure the `/etc/syslog.conf` file for the level of logging that you need.**

   See the `syslog.conf`(4) for more information about the logging levels.

3. **Refresh the configuration information for the `syslog` daemon.**

   ```
   # svcadm refresh system/system-log
   ```

PAM Configuration (Reference)

The PAM configuration file, `pam.conf`(4), is used to configure PAM service modules for system services, such as `login`, `rlogin`, `su`, and `cron`. The system administrator manages this file. An incorrect order of entries in `pam.conf` can cause unforeseen side effects. For example, a badly configured `pam.conf` can lock out users so that single-user mode becomes necessary for repair. For a description of setting the order, see "How PAM Stacking Works" on page 318.

PAM Configuration File Syntax

The entries in the configuration file are in the format:

service-name module-type control-flag module-path module-options

service-name	Name of the service, for example, `ftp`, `login`, or `passwd`. An application can use different service names for the services that the application provides. For example, the Solaris secure shell daemon uses these service names: `sshd-none`, `sshd-password`, `sshd-kbdint`, `sshd-pubkey`, and `sshd-hostbased`. The service-name *other* is a predefined name that is used as a wildcard service-name. If a particular service-name is not found in the configuration file, the configuration for *other* is used.
module-type	The type of service, that is, `auth`, `account`, `session`, or `password`.
control-flag	Indicates the role of the module in determining the integrated success or failure value for the service. Valid control flags are `binding`, `include`, `optional`, `required`, `requisite`, and `sufficient`. See "How PAM Stacking Works" on page 318 for information on the use of these flags.
module-path	The path to the library object that implements the service. If the pathname is not absolute, the pathname is assumed to be relative to

	`/usr/lib/security/$ISA/`. Use the architecture-dependent macro $ISA to cause `libpam` to look in the directory for the particular architecture of the application.
module-options	Options that are passed to the service modules. A module's man page describes the options that are accepted by that module. Typical module options include `nowarn` and `debug`.

How PAM Stacking Works

When an application calls on the following functions, `libpam` reads the configuration file `/etc/pam.conf` to determine which modules participate in the operation for this service:

- `pam_authenticate`(3PAM)
- `pam_acct_mgmt`(3PAM)
- `pam_setcred`(3PAM)
- `pam_open_session`(3PAM)
- `pam_close_session`(3PAM)
- `pam_chauthtok`(3PAM)

If `/etc/pam.conf` contains only one module for an operation for this service such as authentication or account management, the result of that module determines the outcome of the operation. For example, the default authentication operation for the `passwd` application contains one module, `pam_passwd_auth.so.1`:

```
passwd  auth required           pam_passwd_auth.so.1
```

If, on the other hand, there are multiple modules defined for the service's operation, those modules are said to be *stacked* and that a *PAM stack* exists for that service. For example, consider the case where `pam.conf` contains the following entries:

```
login   auth requisite          pam_authtok_get.so.1
login   auth required           pam_dhkeys.so.1
login   auth required           pam_unix_cred.so.1
login   auth required           pam_unix_auth.so.1
login   auth required           pam_dial_auth.so.1
```

These entries represent a sample `auth` stack for the `login` service. To determine the outcome of this stack, the result codes of the individual modules require an *integration process*. In the integration process, the modules are executed in order as specified in `/etc/pam.conf`. Each success or failure code is integrated in the overall result depending on the module's control flag. The control flag can cause early termination of the stack. For example, a `requisite` module might fail, or a `sufficient` or `binding` module might succeed. After the stack has been processed, the individual results are combined into a single, overall result that is delivered to the application.

The control flag indicates the role that a PAM module plays in determining access to the service. The control flags and their effects are:

- **Binding** – Success in meeting a binding module's requirements returns success immediately to the application if no previous required modules have failed. If these conditions are met, then no further execution of modules occurs. Failure causes a required failure to be recorded and the processing of modules to be continued.
- **Include** – Adds lines from a separate PAM configuration file to be used at this point in the PAM stack. This flag does not control success or failure behaviors. When a new file is read, the PAM include stack is incremented. When the stack check in the new file finishes, the include stack value is decremented. When the end of a file is reached and the PAM include stack is 0, then the stack processing ends. The maximum number for the PAM include stack is 32.
- **Optional** – Success in meeting an optional module's requirements is not necessary for using the service. Failure causes an optional failure to be recorded.
- **Required** – Success in meeting a required module's requirements is necessary for using the service. Failure results in an error return after the remaining modules for this service have been executed. Final success for the service is returned only if no binding or required modules have reported failures.
- **Requisite** – Success in meeting a requisite module's requirements is necessary for using the service. Failure results in an immediate error return with no further execution of modules. All requisite modules for a service must return success for the function to be able to return success to the application.
- **Sufficient** – If no previous required failures have occurred, success in a sufficient module returns success to the application immediately with no further execution of modules. Failure causes an optional failure to be recorded.

The following two diagrams shows how access is determined in the integration process. The first diagram indicates how success or failure is recorded for each type of control flag. The second diagram shows how the integrated value is determined.

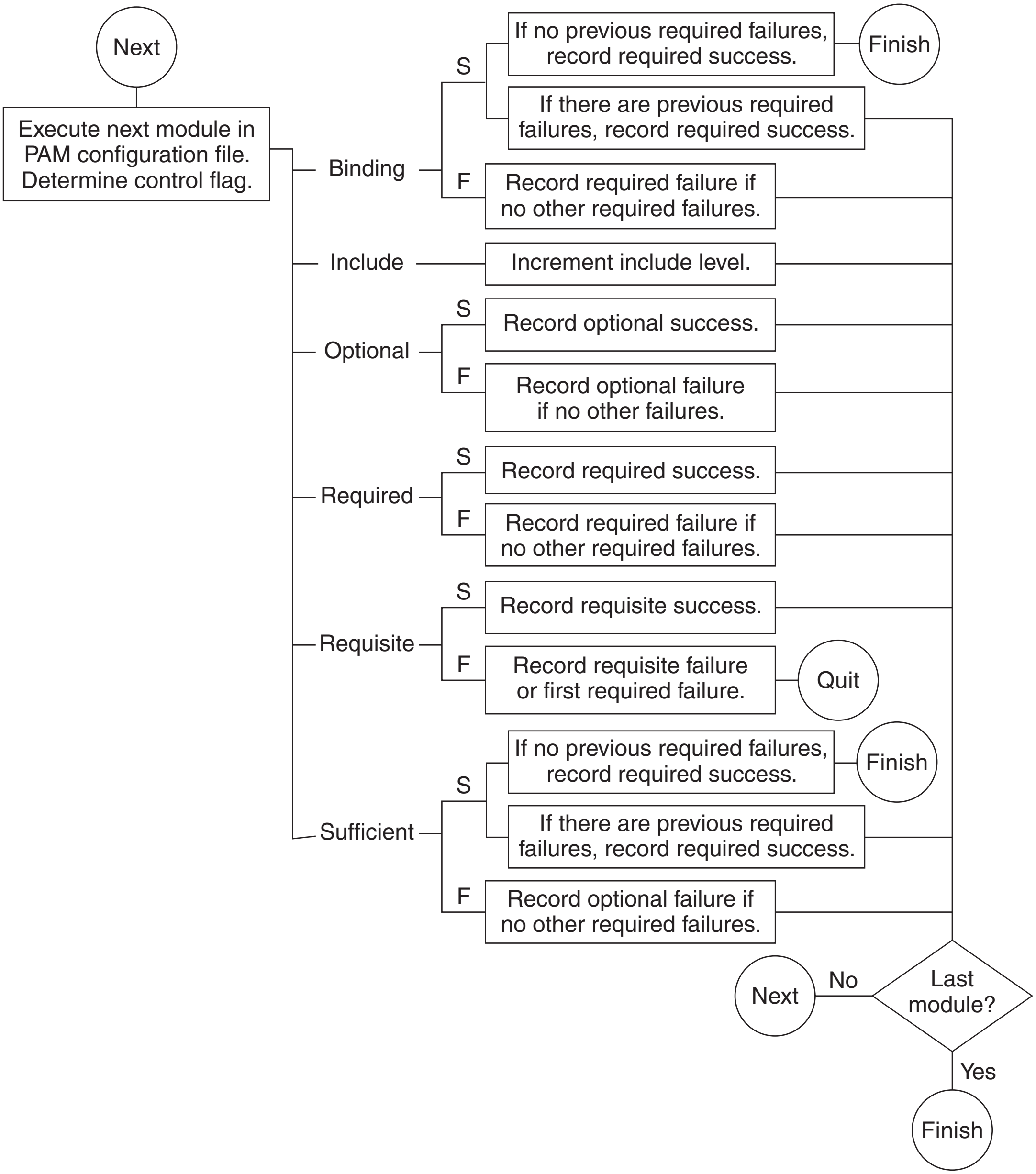

Next
Execute next module in PAM configuration file. Determine control flag.
Binding
S
If no previous required failures, record required success.
Finish
If there are previous required failures, record required success.
F
Record required failure if no other required failures.
Include
Increment include level.
Optional
S
Record optional success.
F
Record optional failure if no other failures.
Required
S
Record required success.
F
Record required failure if no other required failures.
Requisite
S
Record requisite success.
F
Record requisite failure or first required failure.
Quit
Sufficient
S
If no previous required failures, record required success.
Finish
If there are previous required failures, record required success.
F
Record optional failure if no other required failures.
Next
No
Last module?
Yes
Finish

FIGURE 17–2 PAM Stacking: Effect of Control Flags

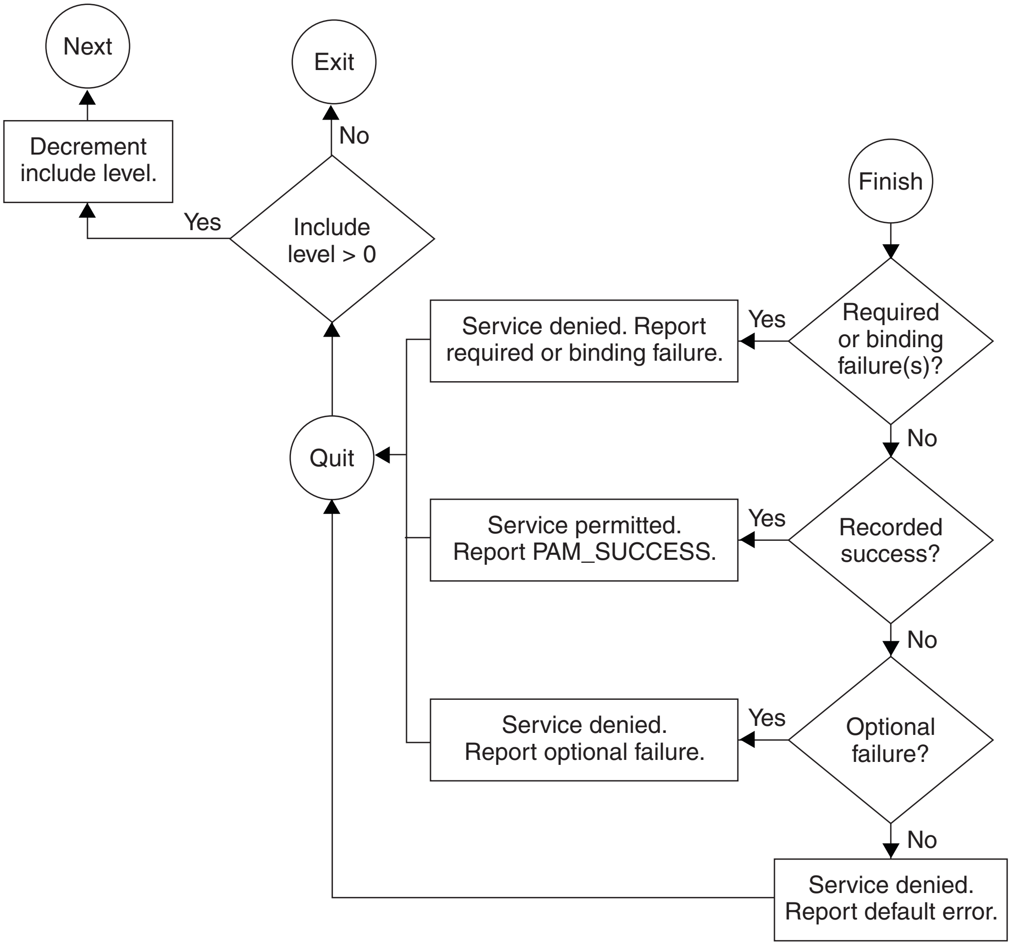

Next
Exit
No
Decrement include level.
Finish
Yes
Include level > 0
Service denied. Report required or binding failure.
Yes
Required or binding failure(s)?
Quit
No
Service permitted. Report PAM_SUCCESS.
Yes
Recorded success?
No
Service denied. Report optional failure.
Yes
Optional failure?
No
Service denied. Report default error.

FIGURE 17–3 PAM Stacking: How Integrated Value Is Determined

PAM Stacking Example

Consider the following example of an `rlogin` service that requests authentication.

EXAMPLE 17–1 Partial Contents of a Typical PAM Configuration File

The `pam.conf` file in this example has the following contents for `rlogin` services:

```
# Authentication management
...
# rlogin service
rlogin  auth sufficient        pam_rhosts_auth.so.1
```

EXAMPLE 17–1 Partial Contents of a Typical PAM Configuration File *(Continued)*

```
rlogin  auth requisite          pam_authtok_get.so.1
rlogin  auth required           pam_dhkeys.so.1
rlogin  auth required           pam_unix_auth.so.1
...
```

When the `rlogin` service requests authentication, `libpam` first executes the `pam_rhosts_auth`(5) module. The control flag is set to `sufficient` for the `pam_rhosts_auth` module. If the `pam_rhosts_auth` module is able to authenticate the user, then processing stops and success is returned to the application.

If the `pam_rhosts_auth` module fails to authenticate the user, then the next PAM module, `pam_authtok_get`(5) is executed. The control flag for this module is set to `requisite`. If `pam_authtok_get` fails, then the authentication process ends and the failure is returned to `rlogin`.

If `pam_authtok_get` succeeds, then the next two modules, `pam_dhkeys`(5) and `pam_unix_auth`(5), are executed. Both modules have the associated control flags that are set to `required` so that the process continues regardless of whether an individual failure is returned. After `pam_unix_auth` is executed, no modules for `rlogin` authentication remain. At this point, if either `pam_dhkeys` or `pam_unix_auth` has returned a failure, the user is denied access through `rlogin`.

◆ ◆ ◆

CHAPTER 18

Using SASL

This chapter includes information about the Simple Authentication and Security Layer (SASL).

- "SASL (Overview)" on page 323
- "SASL (Reference)" on page 324

SASL (Overview)

The Simple Authentication and Security Layer (SASL) is a framework that provides authentication and optional security services to network protocols. An application calls the SASL library, `/usr/lib/libsasl.so`, which provides a glue layer between the application and the various SASL mechanisms. The mechanisms are used in the authentication process and in providing optional security services. The version of SASL delivered with the Solaris 10 release is derived from the Cyrus SASL with a few changes.

SASL provides the following services:

- Loading of any plug-ins
- Determining the necessary security options from the application to aid in the choice of a security mechanism
- Listing of plug-ins that are available to the application
- Choosing the best mechanism from a list of available mechanisms for a particular authentication attempt
- Routing the authentication data between the application and the chosen mechanism
- Providing information about the SASL negotiation back to the application

SASL (Reference)

The following section provides information about the implementation of SASL for the Solaris 10 release.

SASL Plug-ins

SASL plug-ins provide support for security mechanisms, user-canonicalization, and auxiliary property retrieval. By default, the dynamically loaded 32-bit plug-ins are installed in `/usr/lib/sasl`, and the 64-bit plug-ins are installed in `/usr/lib/sasl/`*$ISA*. The following security mechanism plug-ins are provided in the Solaris 10 release:

`crammd5.so.1`	CRAM-MD5, which supports authentication only, no authorization
`digestmd5.so.1`	DIGEST-MD5, which supports authentication, integrity, and privacy, as well as authorization
`gssapi.so.1`	GSSAPI, which supports authentication, integrity, and privacy, as well as authorization. The GSSAPI security mechanism requires a functioning Kerberos infrastructure.
`plain.so.1`	PLAIN, which supports authentication and authorization.

In addition, the EXTERNAL security mechanism plug-in and the INTERNAL user canonicalization plug-ins are built into `libsasl.so.1`. The EXTERNAL mechanism supports authentication and authorization. The mechanism supports integrity and privacy if the external security source provides it. The INTERNAL plug-in adds the realm name if necessary to the username.

The Solaris 10 release is not supplying any `auxprop` plug-ins at this time. For the CRAM-MD5 and DIGEST-MD5 mechanism plug-ins to be fully operational on the server side, the user must provide an `auxprop` plug-in to retrieve clear text passwords. The PLAIN plug-in requires additional support to verify the password. The support for password verification can be one of the following: a callback to the server application, an `auxprop` plug-in, `saslauthd`, or `pwcheck`. The `salauthd` and `pwcheck` daemons are not provided in the Solaris releases. For better interoperability, restrict server applications to those mechanisms that are fully operational by using the `mech_list` SASL option.

SASL Environment Variable

By default, the client authentication name is set to `getenv("LOGNAME")`. This variable can be reset by the client or by the plug-in.

SASL Options

The behavior of `libsasl` and the plug-ins can be modified on the server side by using options that can be set in the `/etc/sasl/`*app*`.conf` file. The variable *app* is the server-defined name for the application. The documentation for the server *app* should specify the application name.

The following options are supported in the Solaris 10 release:

`auto_transition`	Automatically transitions the user to other mechanisms when the user does a successful plain text authentication.
`auxprop_login`	Lists the name of auxiliary property plug-ins to use.
`canon_user_plugin`	Selects the canon_user plug-in to use.
`mech_list`	Lists the mechanisms that are allowed to be used by the server application.
`pwcheck_method`	Lists the mechanisms used to verify passwords. Currently, `auxprop` is the only allowed value.
`reauth_timeout`	Sets the length of time, in minutes, that authentication information is cached for a fast reauthentication. This option is used by the DIGEST-MD5 plug-in. Setting this option to 0 disables reauthentication.

The following options are not supported in the Solaris 10 release:

`plugin_list`	Lists available mechanisms. Not used because the option changes the behavior of the dynamic loading of plugins.
`saslauthd_path`	Defines the location of the saslauthd door, which is used for communicating with the `saslauthd` daemon. The `saslauthd` daemon is not included in the Solaris 10 release. So, this option is also not included.
`keytab`	Defines the location of the `keytab` file used by the GSSAPI plug-in. Use the `KRB5_KTNAME` environment variable instead to set the default `keytab` location.

The following options are options not found in Cyrus SASL. However, they have been added for the Solaris 10 release:

`use_authid`	Acquire the client credentials rather than use the default credentials when creating the GSS client security context. By default, the default client Kerberos identity is used.
`log_level`	Sets the desired level of logging for a server.

◆ ◆ ◆ CHAPTER 19

Using Solaris Secure Shell (Tasks)

Solaris Secure Shell enables a user to securely access a remote host over an unsecured network. The shell provides commands for remote login and remote file transfer. The following is a list of topics in this chapter.

- "Solaris Secure Shell (Overview)" on page 327
- "Solaris Secure Shell Enhancements in the Solaris 10 Release" on page 330
- "Configuring Solaris Secure Shell (Task Map)" on page 331
- "Using Solaris Secure Shell (Task Map)" on page 335

For reference information, see Chapter 20, "Solaris Secure Shell (Reference)."

Solaris Secure Shell (Overview)

In Solaris Secure Shell, authentication is provided by the use of passwords, public keys, or both. All network traffic is encrypted. Thus, Solaris Secure Shell prevents a would-be intruder from being able to read an intercepted communication. Solaris Secure Shell also prevents an adversary from spoofing the system.

Solaris Secure Shell can also be used as an on-demand virtual private network (VPN). A VPN can forward X Window system traffic or can connect individual port numbers between the local machines and remote machines over an encrypted network link.

With Solaris Secure Shell, you can perform these actions:

- Log in to another host securely over an unsecured network.
- Copy files securely between the two hosts.
- Run commands securely on the remote host.

Solaris Secure Shell supports two versions of the Secure Shell protocol. Version 1 is the original version of the protocol. Version 2 is more secure, and it amends some of the basic security design flaws of version 1. Version 1 is provided only to assist users who are migrating to version 2. Users are strongly discouraged from using version 1.

Note – Hereafter in this text, v1 is used to represent version 1, and v2 is used to represent version 2.

Solaris Secure Shell Authentication

Solaris Secure Shell provides public key and password methods for authenticating the connection to the remote host. Public key authentication is a stronger authentication mechanism than password authentication, because the private key never travels over the network.

The authentication methods are tried in the following order. When the configuration does not satisfy an authentication method, the next method is tried.

- **GSS-API –** Uses credentials for GSS-API mechanisms such as `mech_krb5` (Kerberos V) and `mech_dh` (AUTH_DH) to authenticate clients and servers. For more information on GSS-API, see "Introduction to GSS-API" in *Solaris Security for Developers Guide*.
- **Host-based authentication –** Uses host keys and rhosts files. Uses the client's RSA and DSA public/private host keys to authenticate the client. Uses the rhosts files to authorize clients to users.
- **Public key authentication –** Authenticates users with their RSA and DSA public/private keys.
- **Password authentication –** Uses PAM to authenticate users. Keyboard authentication method in v2 allows for arbitrary prompting by PAM. For more information, see the `SECURITY` section in the `sshd`(1M) man page.

The following table shows the requirements for authenticating a user who is trying to log into a remote host. The user is on the local host, the client. The remote host, the server, is running the `sshd` daemon. The table shows the Solaris Secure Shell authentication methods, the compatible protocol versions, and the host requirements.

TABLE 19–1 Authentication Methods for Solaris Secure Shell

Authentication Method (Protocol Version)	Local Host (Client) Requirements	Remote Host (Server) Requirements
GSS-API (v2)	Initiator credentials for the GSS mechanism.	Acceptor credentials for the GSS mechanism. For more information, see "Acquiring GSS Credentials in Solaris Secure Shell" on page 348.

TABLE 19–1 Authentication Methods for Solaris Secure Shell *(Continued)*

Authentication Method (Protocol Version)	Local Host (Client) Requirements	Remote Host (Server) Requirements

Host-based (v2)	User account
	Local host private key in `/etc/ssh/ssh_host_rsa_key` or `/etc/ssh/ssh_host_dsa_key`
	`HostbasedAuthentication yes` in `/etc/ssh/ssh_config`
		User account
		Local host public key in `/etc/ssh/known_hosts` or `~/.ssh/known_hosts`
		`HostbasedAuthentication yes` in `/etc/ssh/sshd_config`
		`IgnoreRhosts no` in `/etc/ssh/sshd_config`
		Local host entry in `/etc/shosts.equiv`, `/etc/hosts.equiv`, `~/.rhosts`, or `~/.shosts`

RSA or DSA public key (v2)	User account Private key in `~/.ssh/id_rsa` or `~/.ssh/id_dsa` User's public key in `~/.ssh/id_rsa.pub` or `~/.ssh/id_dsa.pub`	User account User's public key in `~/.ssh/authorized_keys`
RSA public key (v1)	User account Private key in `~/.ssh/identity` User's public key in `~/.ssh/identity.pub`	User account User's public key in `~/.ssh/authorized_keys`
Keyboard-interactive (v2)	User account	User account Supports PAM, including arbitrary prompting and password changing when password aging is triggered.
Password-based (v1 or v2)	User account	User account Supports PAM.
`.rhosts` only (v1)	User account	User account `IgnoreRhosts no` in `/etc/ssh/sshd_config` Local host entry in `/etc/shosts.equiv`, `/etc/hosts.equiv`, `~/.shosts`, or `~/.rhosts`
`.rhosts` with RSA (v1) on server only	User account Local host public key in `/etc/ssh/ssh_host_rsa1_key`	User account Local host public key in `/etc/ssh/ssh_known_hosts` or `~/.ssh/known_hosts` `IgnoreRhosts no` in `/etc/ssh/sshd_config` Local host entry in `/etc/shosts.equiv`, `/etc/hosts.equiv`, `~/.shosts`, or `~/.rhosts`

Solaris Secure Shell in the Enterprise

For a comprehensive discussion of Secure Shell on a Solaris system, see *Secure Shell in the Enterprise*, by Jason Reid, ISBN 0-13-142900-0, June 2003. The book is part of the Sun BluePrints Series, which is published by Sun Microsystems Press.

For online information, navigate to Sun's BigAdmin System Administration Portal web site, `http://www.sun.com/bigadmin`. Click Docs, then Sun BluePrints under Misc./Comprehensive. Click Sun BluePrints OnLine, then Archives by Subject, then Security. The archives include the following articles:

- *Role Based Access Control and Secure Shell – A Closer Look At Two Solaris Operating Environment Security Features*
- *Integrating the Secure Shell Software*
- *Configuring the Secure Shell Software*

Solaris Secure Shell Enhancements in the Solaris 10 Release

Since the Solaris 9 release, the following changes have been introduced to Solaris Secure Shell:

- Solaris Secure Shell is based on OpenSSH 3.5p1. The Solaris implementation also includes features and bug fixes from versions up to OpenSSH 3.8p1.
- The default value of `X11Forwarding` is `yes` in the `/etc/ssh/sshd_config` file.
- The following keywords have been introduced:
  - `GSSAPIAuthentication`
  - `GSSAPIKeyExchange`
  - `GSSAPIDelegateCredentials`
  - `GSSAPIStoreDelegatedCredentials`
  - `KbdInteractiveAuthentication`

  The `GSSAPI` keywords enable Solaris Secure Shell to use GSS credentials for authentication. The `KbdInteractiveAuthentication` keyword supports arbitrary prompting and password changing in PAM. For a complete list of keywords and their default values, see "Keywords in Solaris Secure Shell" on page 350.
- The ARCFOUR and AES128-CTR ciphers are now available. ARCFOUR is also known as RC4. The AES cipher is AES in counter mode.
- The `sshd` daemon uses the variables in `/etc/default/login` and the `login` command. The `/etc/default/login` variables can be overridden by values in the `sshd_config` file. For more information, see "Solaris Secure Shell and Login Environment Variables" on page 354 and the `sshd_config`(4) man page.

Solaris Secure Shell (Task Map)

The following task map points to task maps for configuring Solaris Secure Shell and for using Solaris Secure Shell.

Task	Description	For Instructions
Configure Solaris Secure Shell	Guides administrators in configuring Solaris Secure Shell for users.	"Configuring Solaris Secure Shell (Task Map)" on page 331
Use Solaris Secure Shell	Guides users in using Solaris Secure Shell.	"Using Solaris Secure Shell (Task Map)" on page 335

Configuring Solaris Secure Shell (Task Map)

The following task map points to procedures for configuring Solaris Secure Shell.

Task	Description	For Instructions
Configure host-based authentication	Configures host-based authentication on the client and server.	"How to Set Up Host-Based Authentication for Solaris Secure Shell" on page 331
Configure a host to use v1 and v2	Creates public key files for hosts that use v1 and v2 protocols.	"How to Enable Solaris Secure Shell v1" on page 334
Configure port forwarding	Enables users to use port forwarding.	"How to Configure Port Forwarding in Solaris Secure Shell" on page 334

Configuring Solaris Secure Shell

By default, host-based authentication and the use of both protocols are not enabled in Solaris Secure Shell. Changing these defaults requires administrative intervention. Also, for port forwarding to work requires administrative intervention.

▼ How to Set Up Host-Based Authentication for Solaris Secure Shell

The following procedure sets up a public key system where the client's public key is used for authentication on the server. The user must also create a public/private key pair.

In the procedure, the terms *client* and *local host* refer to the machine where a user types the `ssh` command. The terms *server* and *remote host* refer to the machine that the client is trying to reach.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **On the client, enable host-based authentication.**

In the client configuration file, `/etc/ssh/ssh_config`, type the following entry:

```
HostbasedAuthentication yes
```

3 **On the server, enable host-based authentication.**

In the server configuration file, `/etc/ssh/sshd_config`, type the same entry:

```
HostbasedAuthentication yes
```

4 **On the server, configure a file that enables the client to be recognized as a trusted host.**

For more information, see the `FILES` section of the `sshd`(1M) man page.

- **Add the client as an entry to the server's `/etc/shosts.equiv` file.**

  client-host

- **Or, you can instruct users to add an entry for the client to their `~/.shosts` file on the server.**

  client-host

5 **On the server, ensure that the `sshd` daemon can access the list of trusted hosts.**

Set `IgnoreRhosts` to no in the `/etc/ssh/sshd_config` file.

```
# sshd_config
IgnoreRhosts no
```

6 **Ensure that users of Solaris Secure Shell at your site have accounts on both hosts.**

7 **Do one of the following to put the client's public key on the server.**

- **Modify the `sshd_config` file on the server, then instruct your users to add the client's public host keys to their `~/.ssh/known_hosts` file.**

  ```
  # sshd_config
  IgnoreUserKnownHosts no
  ```

  For user instructions, see "How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell" on page 336.

- **Copy the client's public key to the server.**

  The host keys are stored in the `/etc/ssh` directory. The keys are typically generated by the `sshd` daemon on first boot.

  a. **Add the key to the `/etc/ssh/ssh_known_hosts` file on the server.**

     On the client, type the command on one line with no backslash.

     ```
     # cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \
     'cat >> /etc/ssh/ssh_known_hosts && echo "Host key copied"'
     ```

  b. **When you are prompted, supply your login password.**

     When the file is copied, the message "Host key copied" is displayed.

Example 19–1

Setting Up Host-based Authentication

In the following example, each host is configured as a server and as a client. A user on either host can initiate an `ssh` connection to the other host. The following configuration makes each host a server and a client:

- On each host, the Solaris Secure Shell configuration files contain the following entries:

  ```
  # /etc/ssh/ssh_config
  HostBasedAuthentication yes
  #
  # /etc/ssh/sshd_config
  HostBasedAuthentication yes
  IgnoreRhosts no
  ```

- On each host, the `shosts.equiv` file contains an entry for the other host:

  ```
  # /etc/hosts.equiv on machine2
  machine1

  # /etc/hosts.equiv on machine1
  machine2
  ```

- The public key for each host is in the `/etc/ssh/ssh_known_hosts` file on the other host:

  ```
  # /etc/ssh/ssh_known_hosts on machine2
  ... machine1

  # /etc/ssh/ssh_known_hosts on machine1
  ... machine2
  ```

- Users have an account on both hosts:

  ```
  # /etc/passwd on machine1
  jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
  ```

```
# /etc/passwd on machine2
jdoe:x:3111:10:J Doe:/home/jdoe:/bin/sh
```

▼ How to Enable Solaris Secure Shell v1

This procedure is useful when a host interoperates with hosts that run v1 and v2.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Configure the host to use both Solaris Secure Shell protocols.**

   Edit the `/etc/ssh/sshd_config` file.

   ```
   # Protocol 2
   Protocol 2,1
   ```

3. **Provide a separate file for the host key for v1.**

   Add a `HostKey` entry to the `/etc/ssh/sshd_config` file.

   ```
   HostKey /etc/ssh/ssh_host_rsa_key
   HostKey /etc/ssh/ssh_host_dsa_key
   HostKey /etc/ssh/ssh_host_rsa1_key
   ```

4. **Generate a host key for v1.**

   ```
   # ssh-keygen -t rsa1 -f /etc/ssh/ssh_host_rsa1_key -N ''
   ```

`-t rsa1`	Indicates the RSA algorithm for v1.
`-f`	Indicates the file that holds the host key.
`-N ''`	Indicates that no passphrase is required.

5. **Restart the `sshd` daemon.**

   ```
   # svcadm restart network/ssh:default
   ```

   You can also reboot the system.

▼ How to Configure Port Forwarding in Solaris Secure Shell

Port forwarding enables a local port be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. Similarly, a port can be specified on the remote side.

Note – Solaris Secure Shell port forwarding must use TCP connections. Solaris Secure Shell does not support UDP connections for port forwarding.

---

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 **Configure a Solaris Secure Shell setting on the remote server to allow port forwarding.**

Change the value of `AllowTcpForwarding` to `yes` in the `/etc/ssh/sshd_config` file.

```
# Port forwarding
AllowTcpForwarding yes
```

3 **Restart the Solaris Secure Shell service.**

```
remoteHost# svcadm restart network/ssh:default
```

For information on managing persistent services, see Chapter 15, "Managing Services (Overview)," in *System Administration Guide: Basic Administration* and the `svcadm`(1M) man page.

4 **Verify that port forwarding can be used.**

```
remoteHost# /usr/bin/pgrep -lf sshd
 1296 ssh -L 2001:remoteHost:23 remoteHost
```

Using Solaris Secure Shell (Task Map)

The following task map points to user procedures for using Solaris Secure Shell.

Task	Description	For Instructions
Create a public/private key pair	Enables access to Solaris Secure Shell for sites that require public-key authentication.	"How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell" on page 336
Change your passphrase	Changes the phrase that authenticates your private key.	"How to Change the Passphrase for a Solaris Secure Shell Private Key" on page 339
Log in with Solaris Secure Shell	Provides encrypted Solaris Secure Shell communication when logging in remotely. The process is similar to using the `rsh` command.	"How to Log In to a Remote Host With Solaris Secure Shell" on page 339

Task	Description	For Instructions
Log in to Solaris Secure Shell without being prompted for a password	Enables login by using an agent which provides your password to Solaris Secure Shell.	"How to Reduce Password Prompts in Solaris Secure Shell" on page 340 "How to Set Up the `ssh-agent` Command to Run Automatically in CDE" on page 342
Use port forwarding in Solaris Secure Shell	Specifies a local port or a remote port to be used in a Solaris Secure Shell connection over TCP.	"How to Use Port Forwarding in Solaris Secure Shell" on page 342
Copy files with Solaris Secure Shell	Securely copies files between hosts.	"How to Copy Files With Solaris Secure Shell" on page 344
Securely connect from a host inside a firewall to a host outside the firewall	Uses Solaris Secure Shell commands that are compatible with HTTP or SOCKS5 to connect hosts that are separated by a firewall.	"How to Set Up Default Connections to Hosts Outside a Firewall" on page 345

Using Solaris Secure Shell

Solaris Secure Shell provides secure access between a local shell and a remote shell. For more information, see the `ssh_config`(4) and `ssh`(1) man pages.

▼ How to Generate a Public/Private Key Pair for Use With Solaris Secure Shell

Users must generate a public/private key pair when their site implements host-based authentication or user public-key authentication. For additional options, see the `ssh-keygen`(1) man page.

Before You Begin Determine from your system administrator if host-based authentication is configured.

1. **Start the key generation program.**

```
myLocalHost% ssh-keygen -t rsa
Generating public/private rsa key pair.
...
```

where `-t` is the type of algorithm, one of `rsa`, `dsa`, or `rsa1`.

2. **Specify the path to the file that will hold the key.**

By default, the file name `id_rsa`, which represents an RSA v2 key, appears in parentheses. You can select this file by pressing the Return key. Or, you can type an alternative file name.

```
Enter file in which to save the key (/home/jdoe/.ssh/id_rsa):    <Press Return>
```

The file name of the public key is created automatically by appending the string `.pub` to the name of the private key file.

3 Type a passphrase for using your key.

This passphrase is used for encrypting your private key. A null entry is *strongly discouraged*. Note that the passphrase is not displayed when you type it in.

```
Enter passphrase (empty for no passphrase):     <Type passphrase>
```

4 Retype the passphrase to confirm it.

```
Enter same passphrase again:     <Type passphrase>
Your identification has been saved in /home/jdoe/.ssh/id_rsa.
Your public key has been saved in /home/jdoe/.ssh/id_rsa.pub.
The key fingerprint is:
0e:fb:3d:57:71:73:bf:58:b8:eb:f3:a3:aa:df:e0:d1 jdoe@myLocalHost
```

5 Check the results.

Check that the path to the key file is correct.

```
% ls ~/.ssh
id_rsa
id_rsa.pub
```

At this point, you have created a public/private key pair.

6 Choose the appropriate option:

- **If your administrator has configured host-based authentication, you might need to copy the local host's public key to the remote host.**

  You can now log in to the remote host. For details, see "How to Log In to a Remote Host With Solaris Secure Shell" on page 339.

  a. Type the command on one line with no backslash.

  ```
  % cat /etc/ssh/ssh_host_dsa_key.pub | ssh RemoteHost \
  'cat >> ~./ssh/known_hosts && echo "Host key copied"'
  ```

  b. When you are prompted, supply your login password.

  ```
  Enter password:     <Type password>
  Host key copied
  %
  ```

- **If your site uses user authentication with public keys, populate your** `authorized_keys` **file on the remote host.**

  a. **Copy your public key to the remote host.**

  Type the command on one line with no backslash.

  ```
  myLocalHost% cat $HOME/.ssh/id_rsa.pub | ssh myRemoteHost \
  'cat >> .ssh/authorized_keys && echo "Key copied"'
  ```

  b. **When you are prompted, supply your login password.**

  When the file is copied, the message "Key copied" is displayed.

  ```
  Enter password:       Type login password
  Key copied
  myLocalHost%
  ```

7 **(Optional) Reduce the prompting for passphrases.**

For a procedure, see "How to Reduce Password Prompts in Solaris Secure Shell" on page 340. For more information, see the `ssh-agent`(1) and `ssh-add`(1) man pages.

Example 19–2 Establishing a v1 RSA Key for a User

In the following example, the user can contact hosts that run v1 of the Solaris Secure Shell protocol. To be authenticated by v1 hosts, the user creates a v1 key, then copies the public key portion to the remote host.

```
myLocalHost% ssh-keygen -t rsa1 -f /home/jdoe/.ssh/identity
Generating public/private rsa key pair.
...
Enter passphrase (empty for no passphrase):     <Type passphrase>
Enter same passphrase again:     <Type passphrase>
Your identification has been saved in /home/jdoe/.ssh/identity.
Your public key has been saved in /home/jdoe/.ssh/identity.pub.
The key fingerprint is:
...
myLocalHost% ls ~/.ssh
id_rsa
id_rsa.pub
identity
identity.pub
myLocalHost% cat $HOME/.ssh/identity.pub | ssh myRemoteHost \
'cat >> .ssh/authorized_keys && echo "Key copied"'
```

▼ How to Change the Passphrase for a Solaris Secure Shell Private Key

The following procedure does not change the private key. The procedure changes the authentication mechanism for the private key, the passphrase. For more information, see the `ssh-keygen`(1) man page.

- **Change your passphrase.**

  Type the `ssh-keygen` command with the `-p` option, and answer the prompts.

```
myLocalHost% ssh-keygen -p
Enter file which contains the private key (/home/jdoe/.ssh/id_rsa):     <Press Return>
Enter passphrase (empty for no passphrase):      <Type passphrase>
Enter same passphrase again:        <Type passphrase>
```

  where `-p` requests changing the passphrase of a private key file.

▼ How to Log In to a Remote Host With Solaris Secure Shell

1. **Start a Solaris Secure Shell session.**

   Type the `ssh` command, and specify the name of the remote host.

```
myLocalHost% ssh myRemoteHost
```

   A prompt questions the authenticity of the remote host:

```
The authenticity of host 'myRemoteHost' can't be established.
RSA key fingerprint in md5 is: 04:9f:bd:fc:3d:3e:d2:e7:49:fd:6e:18:4f:9c:26
Are you sure you want to continue connecting(yes/no)?
```

   This prompt is normal for initial connections to remote hosts.

2. **If prompted, verify the authenticity of the remote host key.**

   - **If you cannot confirm the authenticity of the remote host, type `no` and contact your system administrator.**

```
Are you sure you want to continue connecting(yes/no)? no
```

     The administrator is responsible for updating the global `/etc/ssh/ssh_known_hosts` file. An updated `ssh_known_hosts` file prevents this prompt from appearing.

   - **If you confirm the authenticity of the remote host, answer the prompt and continue to the next step.**

```
Are you sure you want to continue connecting(yes/no)? yes
```

3 **Authenticate yourself to Solaris Secure Shell.**

a. **When prompted, type your passphrase.**

```
Enter passphrase for key '/home/jdoe/.ssh/id_rsa':      <Type passphrase>
```

b. **When prompted, type your account password.**

```
jdoe@myRemoteHost's password:      <Type password>
Last login: Fri Jul 20 14:24:10 2001 from myLocalHost
myRemoteHost%
```

4 **Conduct transactions on the remote host.**

The commands that you send are encrypted. Any responses that you receive are encrypted.

5 **Close the Solaris Secure Shell connection.**

When you are finished, type **`exit`** or use your usual method for exiting your shell.

```
myRemoteHost% exit
myRemoteHost% logout
Connection to myRemoteHost closed
myLocalHost%
```

▼ How to Reduce Password Prompts in Solaris Secure Shell

If you do not want to type your passphrase and your password to use Solaris Secure Shell, you can use the agent daemon. Start the daemon at the beginning of the session. Then, store your private keys with the agent daemon by using the `ssh-add` command. If you have different accounts on different hosts, add the keys that you need for the session.

You can start the agent daemon manually when needed, as described in the following procedure. Or, you can set the agent daemon to run automatically at the start of every session as described in "How to Set Up the `ssh-agent` Command to Run Automatically in CDE" on page 342.

1 **Start the agent daemon.**

```
myLocalHost% ssh-agent
```

2 **Verify that the agent daemon has been started.**

```
myLocalHost% eval 'ssh-agent'
Agent pid 9892
```

3 **Add your private key to the agent daemon.**

Type the `ssh-add` command.

```
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa:     <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
myLocalHost%
```

4 **Start a Solaris Secure Shell session.**

```
myLocalHost% ssh myRemoteHost
```

You are not prompted for a passphrase.

Example 19–3 Using `ssh-add` Options

In this example, `jdoe` adds two keys to the agent daemon. The `-l` option is used to list all keys that are stored in the daemon. At the end of the session, the `-D` option is used to remove all the keys from the agent daemon.

```
myLocalHost% ssh-agent
myLocalHost% ssh-add
Enter passphrase for /home/jdoe/.ssh/id_rsa:     <Type passphrase>
Identity added: /home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa)
myLocalHost% ssh-add /home/jdoe/.ssh/id_dsa
Enter passphrase for /home/jdoe/.ssh/id_dsa:     <Type passphrase>
Identity added:
/home/jdoe/.ssh/id_dsa(/home/jdoe/.ssh/id_dsa)

myLocalHost% ssh-add -l
md5 1024 0e:fb:3d:53:71:77:bf:57:b8:eb:f7:a7:aa:df:e0:d1
/home/jdoe/.ssh/id_rsa(RSA)
md5 1024 c1:d3:21:5e:40:60:c5:73:d8:87:09:3a:fa:5f:32:53
/home/jdoe/.ssh/id_dsa(DSA)

    User conducts Solaris Secure Shell transactions

myLocalHost% ssh-add -D
Identity removed:
/home/jdoe/.ssh/id_rsa(/home/jdoe/.ssh/id_rsa.pub)
/home/jdoe/.ssh/id_dsa(DSA)
```

▼ How to Set Up the `ssh-agent` Command to Run Automatically in CDE

If you use CDE, you can avoid providing your passphrase and password whenever you use Solaris Secure Shell by automatically starting an agent daemon, `ssh-agent`. You can start the agent daemon from the `.dtprofile` script. To add your passphrase and password to the agent daemon, see Example 19–3.

Caution – If you use the Sun Java™ Desktop System (Java DS), do not set up the `ssh-agent` command to run automatically. Because the killing of the `ssh-agent` process is controlled by a CDE interface, when you exit the Java DS, the daemon continues to run. For example, if you start the daemon in a CDE session, move to a Java DS session, and then log out, the daemon continues to run.

A running daemon uses system resources. Although no known issues are associated with leaving the `ssh-agent` daemon running, the daemon contains a password, which could create a security risk.

---

1. **Start the agent daemon automatically in a user startup script.**

   Add the following lines to the end of the `$HOME/.dtprofile` script:

```
if [ "$SSH_AUTH_SOCK" = "" -a -x /usr/bin/ssh-agent ]; then
                eval ‘/usr/bin/ssh-agent‘
fi
```

2. **Terminate the agent daemon when you exit the CDE session.**

   Add the following lines to the `$HOME/.dt/sessions/sessionexit` script:

```
if [ "$SSH_AGENT_PID" != "" -a -x /usr/bin/ssh-agent ]; then
                /usr/bin/ssh-agent -k
fi
```

   This entry ensures that no one can use the Solaris Secure Shell agent after a CDE session is terminated. Because the script uses a CDE-specific interface, `sessionexit`, this procedure does not terminate the agent daemon in a Sun Java Desktop System session.

▼ How to Use Port Forwarding in Solaris Secure Shell

You can specify that a local port be forwarded to a remote host. Effectively, a socket is allocated to listen to the port on the local side. The connection from this port is made over a secure channel to the remote host. For example, you might specify port `143` to obtain email remotely with IMAP4. Similarly, a port can be specified on the remote side.

Before You Begin To use port forwarding, the administrator must have enabled port forwarding on the remote Solaris Secure Shell server. For details, see "How to Configure Port Forwarding in Solaris Secure Shell" on page 334.

- **To use secure port forwarding, choose one of the following options:**
  - **To set a local port to receive secure communication from a remote port, specify both ports.**

    Specify the local port that listens for remote communication. Also, specify the remote host and the remote port that forward the communication.

    ```
    myLocalHost% ssh -L localPort:remoteHost:remotePort
    ```

  - **To set a remote port to receive a secure connection from a local port, specify both ports.**

    Specify the remote port that listens for remote communication. Also, specify the local host and the local port that forward the communication.

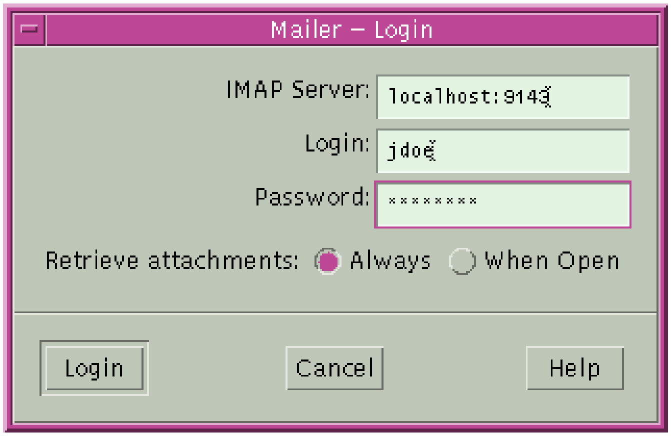
    ```
    myLocalHost% ssh -R remotePort:localhost:localPort
    ```

Example 19–4 Using Local Port Forwarding to Receive Mail

The following example demonstrates how you can use local port forwarding to receive mail securely from a remote server.

```
myLocalHost% ssh -L 9143:myRemoteHost:143 myRemoteHost
```

This command forwards connections from port `9143` on `myLocalHost` to port `143`. Port `143` is the IMAP v2 server port on `myRemoteHost`. When the user launches a mail application, the user needs to specify the local port number, as shown in the following dialog box.



IMAP Server: localhost:9143
Login: jdoe
Password: ********
Retrieve attachments: ● Always ○ When Open

Do not confuse `localhost` in the dialog box with `myLocalHost`. `myLocalHost` is a hypothetical host name. `localhost` is a keyword that identifies your local system.

Example 19–5 Using Remote Port Forwarding to Communicate Outside of a Firewall

This example demonstrates how a user in an enterprise environment can forward connections from a host on an external network to a host inside a corporate firewall.

```
myLocalHost% ssh -R 9022:myLocalHost:22 myOutsideHost
```

This command forwards connections from port `9022` on `myOutsideHost` to port `22`, the `sshd` server, on the local host.

```
myOutsideHost% ssh -p 9022 localhost
myLocalHost%
```

▼ How to Copy Files With Solaris Secure Shell

The following procedure shows how to use the `scp` command to copy encrypted files between hosts. You can copy encrypted files either between a local host and a remote host, or between two remote hosts. The command operates similarly to the `rcp` command, except that the `scp` command prompts for authentication. For more information, see the `scp`(1) man page.

You can also use the `sftp`, a more secure form of the `ftp` command. For more information, see the `sftp`(1) man page.

1. **Start the secure copy program.**

   Specify the source file, the user name at the remote destination, and the destination directory.

   ```
   myLocalHost% scp myfile.1 jdoe@myRemoteHost:~
   ```

2. **Supply your passphrase when prompted.**

   ```
   Enter passphrase for key '/home/jdoe/.ssh/id_rsa':      <Type passphrase>
   myfile.1       25% |*******                      |    640 KB  0:20 ETA
   myfile.1
   ```

   After you type the passphrase, a progress meter is displayed. See the second line in the preceding output. The progress meter displays:

   - The file name
   - The percentage of the file that has been transferred
   - A series of asterisks that indicate the percentage of the file that has been transferred
   - The quantity of data transferred
   - The estimated time of arrival, or ETA, of the complete file (that is, the remaining amount of time)

▼ How to Set Up Default Connections to Hosts Outside a Firewall

You can use Solaris Secure Shell to make a connection from a host inside a firewall to a host outside the firewall. This task is done by specifying a proxy command for `ssh` either in a configuration file or as an option on the command line. For the command-line option, see Example 19–6.

In general, you can customize your `ssh` interactions through a configuration file.

- You can customize either your own personal file in `~/.ssh/config`.
- Or, you can use the settings in the administrative configuration file, `/etc/ssh/ssh_config`.

The files can be customized with two types of proxy commands. One proxy command is for HTTP connections. The other proxy command is for SOCKS5 connections. For more information, see the `ssh_config`(4) man page.

1 **Specify the proxy commands and hosts in a configuration file.**

Use the following syntax to add as many lines as you need:

```
[Host outside-host]
ProxyCommand proxy-command [-h proxy-server] \
[-p proxy-port] outside-host|%h outside-port|%p
```

`Host` *outside-host*
Limits the proxy command specification to instances when a remote host name is specified on the command line. If you use a wildcard for *outside-host*, you apply the proxy command specification to a set of hosts.

proxy-command
Specifies the proxy command.

The command can be either of the following:

- `/usr/lib/ssh/ssh-http-proxy-connect` for HTTP connections
- `/usr/lib/ssh/ssh-socks5-proxy-connect` for SOCKS5 connections

`-h` *proxy-server* and `-p` *proxy-port*
These options specify a proxy server and a proxy port, respectively. If present, the proxies override any environment variables that specify proxy servers and proxy ports, such as `HTTPPROXY`, `HTTPPROXYPORT`, `SOCKS5_PORT`, `SOCKS5_SERVER`, and `http_proxy`. The `http_proxy` variable specifies a URL. If the options are not used, then the relevant environment variables must be set. For more information, see the `ssh-socks5-proxy-connect`(1) and `ssh-http-proxy-connect`(1) man pages.

outside-host
Designates a specific host to connect to. Use the `%h` substitution argument to specify the host on the command line.

outside-port

Designates a specific port to connect to. Use the `%p` substitution argument to specify the port on the command line. By specifying `%h` and `%p` without using the `Host` *outside-host* option, the proxy command is applied to the host argument whenever the `ssh` command is invoked.

2 **Run Solaris Secure Shell, specifying the outside host.**

For example, type the following:

```
myLocalHost% ssh myOutsideHost
```

This command looks for a proxy command specification for `myOutsideHost` in your personal configuration file. If the specification is not found, then the command looks in the system-wide configuration file, `/etc/ssh/ssh_config`. The proxy command is substituted for the `ssh` command.

Example 19–6 Connecting to Hosts Outside a Firewall From the Command Line

"How to Set Up Default Connections to Hosts Outside a Firewall" on page 345 explains how to specify a proxy command in a configuration file. In this example, a proxy command is specified on the `ssh` command line.

```
% ssh -o'Proxycommand=/usr/lib/ssh/ssh-http-proxy-connect \
-h myProxyServer -p 8080 myOutsideHost 22' myOutsideHost
```

The `-o` option to the `ssh` command provides a command-line method of specifying a proxy command. This example command does the following:

- Substitutes the HTTP proxy command for `ssh`
- Uses port `8080` and `myProxyServer` as the proxy server
- Connects to port 22 on `myOutsideHost`

◆ ◆ ◆

CHAPTER 20

Solaris Secure Shell (Reference)

This chapter describes the configuration options in Solaris Secure Shell. The following is a list of the reference information in this chapter.

- "A Typical Solaris Secure Shell Session" on page 347
- "Client and Server Configuration in Solaris Secure Shell" on page 350
- "Keywords in Solaris Secure Shell" on page 350
- "Maintaining Known Hosts in Solaris Secure Shell" on page 355
- "Solaris Secure Shell Packages and Initialization" on page 356
- "Solaris Secure Shell Files" on page 356
- "Solaris Secure Shell Commands" on page 358

For procedures to configure Solaris Secure Shell, see Chapter 19, "Using Solaris Secure Shell (Tasks)."

A Typical Solaris Secure Shell Session

The Solaris Secure Shell daemon (`sshd`) is normally started at boot time when network services are started. The daemon listens for connections from clients. A Solaris Secure Shell session begins when the user runs an `ssh`, `scp`, or `sftp` command. A new `sshd` daemon is forked for each incoming connection. The forked daemons handle key exchange, encryption, authentication, command execution, and data exchange with the client. These session characteristics are determined by client-side configuration files and server-side configuration files. Command-line arguments can override the settings in the configuration files.

The client and server must authenticate themselves to each other. After successful authentication, the user can execute commands remotely and copy data between hosts.

Session Characteristics in Solaris Secure Shell

The server-side behavior of the `sshd` daemon is controlled by keyword settings in the `/etc/ssh/sshd_config` file. For example, the `sshd_config` file controls which types of authentication are permitted for accessing the server. The server-side behavior can also be controlled by the command-line options when the `sshd` daemon is started.

The behavior on the client side is controlled by Solaris Secure Shell keywords in this order of precedence:

- Command-line options
- User's configuration file, `~/.ssh/config`
- System-wide configuration file, `/etc/ssh/ssh_config`

For example, a user can override a system-wide configuration `Cipher` setting of `blowfish` by specifying `-c 3des` on the command line.

Authentication and Key Exchange in Solaris Secure Shell

The Solaris Secure Shell protocols, v1 and v2, both support client user/host authentication and server host authentication. Both protocols involve the exchange of session cryptographic keys for the protection of Solaris Secure Shell sessions. Each protocol provides various methods for authentication and key exchange. Some methods are optional. Solaris Secure Shell supports a number of client authentication mechanisms, as shown in Table 19–1. Servers are authenticated by using known host public keys.

For the v1 protocol, Solaris Secure Shell supports user authentication with passwords. The protocol also supports user public keys and authentication with trusted host public keys. Server authentication is done with a host public key. For the v1 protocol, all public keys are RSA keys. Session key exchanges involve the use of an ephemeral server key that is periodically regenerated.

For the v2 protocol, Solaris Secure Shell supports user authentication and generic interactive authentication, which usually involves passwords. The protocol also supports authentication with user public keys and with trusted host public keys. The keys can be RSA or DSA. Session key exchanges consist of Diffie-Hellman ephemeral key exchanges that are signed in the server authentication step. Additionally, Solaris Secure Shell can use GSS credentials for authentication.

Acquiring GSS Credentials in Solaris Secure Shell

To use GSS-API for authentication in Solaris Secure Shell, the server must have GSS-API acceptor credentials and the client must have GSS-API initiator credentials. Support is available for `mech_dh` and for `mech_krb5`.

For `mech_dh`, the server has GSS-API acceptor credentials if `root` has run the `keylogin` command.

For `mech_krb5`, the server has GSS-API acceptor credentials when the host principal that corresponds to the server has a valid entry in `/etc/krb5/krb5.keytab`.

The client has initiator credentials for `mech_dh` if one of the following has been done:

- The `keylogin` command has been run.
- The `pam_dhkeys` module is used in the `pam.conf` file.

The client has initiator credentials for `mech_krb5` if one of the following has been done:

- The `kinit` command has been run.
- The `pam_krb5` module is used in the `pam.conf` file.

For the use of `mech_dh` in secure RPC, see Chapter 16, "Using Authentication Services (Tasks)." For the use of `mech_krb5`, see Chapter 21, "Introduction to the Kerberos Service." For more information on mechanisms, see the `mech`(4) and `mech_spnego`(5) man pages.

Command Execution and Data Forwarding in Solaris Secure Shell

After authentication is complete, the user can use Solaris Secure Shell, generally by requesting a shell or executing a command. Through the `ssh` command options, the user can make requests. Requests can include allocating a pseudo-tty, forwarding X11 connections or TCP/IP connections, or enabling an `ssh-agent` authentication program over a secure connection.

The basic components of a user session are as follows:

1. The user requests a shell or the execution of a command, which begins the session mode.

   In this mode, data is sent or received through the terminal on the client side. On the server side, data is sent through the shell or a command.
2. When data transfer is complete, the user program terminates.
3. All X11 forwarding and TCP/IP forwarding is stopped, except for those connections that already exist. Existing X11 connections and TCP/IP connections remain open.
4. The server sends an exit status message to the client. When all connections are closed, such as forwarded ports that had remained open, the client closes the connection to the server. Then, the client exits.

Client and Server Configuration in Solaris Secure Shell

The characteristics of a Solaris Secure Shell session are controlled by configuration files. The configuration files can be overridden to a certain degree by options on the command line.

Client Configuration in Solaris Secure Shell

In most cases, the client-side characteristics of a Solaris Secure Shell session are governed by the system-wide configuration file, `/etc/ssh/ssh_config`. The settings in the `ssh_config` file can be overridden by the user's configuration file, `~/.ssh/config`. In addition, the user can override both configuration files on the command line.

The settings in the server's `/etc/ssh/sshd_config` file determine which client requests are permitted by the server. For a list of server configuration settings, see "Keywords in Solaris Secure Shell" on page 350. For detailed information, see the `sshd_config`(4) man page.

The keywords in the client configuration file are listed in "Keywords in Solaris Secure Shell" on page 350. If the keyword has a default value, the value is given. These keywords are described in detail in the `ssh`(1), `scp`(1), `sftp`(1), and `ssh_config`(4) man pages. For a list of keywords in alphabetical order and their equivalent command-line overrides, see Table 20–8.

Server Configuration in Solaris Secure Shell

The server-side characteristics of a Solaris Secure Shell session are governed by the `/etc/ssh/sshd_config` file. The keywords in the server configuration file are listed in "Keywords in Solaris Secure Shell" on page 350. If the keyword has a default value, the value is given. For a full description of the keywords, see the `sshd_config`(4) man page.

Keywords in Solaris Secure Shell

The following tables list the keywords and their default values, if any. The keywords are in alphabetical order. The location of keywords on the client is the `ssh_config` file. Keywords that apply to the server are in the `sshd_config` file. Some keywords are set in both files. If the keyword applies to only one protocol version, the version is listed.

TABLE 20–1 Keywords in Solaris Secure Shell Configuration Files (A to Escape)

Keyword	Default Value	Location	Protocol
`AllowGroups`	No default.	Server	

TABLE 20–1 Keywords in Solaris Secure Shell Configuration Files (A to Escape) *(Continued)*

Keyword	Default Value	Location	Protocol
`AllowTcpForwarding`	`no`	Server	
`AllowUsers`	No default.	Server	
`AuthorizedKeysFile`	`~/.ssh/authorized_keys`	Server	
`Banner`	`/etc/issue`	Server	
`Batchmode`	`no`	Client	
`BindAddress`	No default.	Client	
`CheckHostIP`	`yes`	Client	
`Cipher`	`blowfish`, `3des`	Client	v1
`Ciphers`	`aes128-ctr, aes128-cbc, 3des-cbc, blowfish-cbc, arcfour`	Both	v2
`ClearAllForwardings`	No default.	Client	
`ClientAliveInterval`	`0`	Server	v2
`ClientAliveCountMax`	`3`	Server	v2
`Compression`	`yes`	Both	
`CompressionLevel`	No default.	Client	
`ConnectionAttempts`	`1`	Client	
`DenyGroups`	No default.	Server	
`DenyUsers`	No default.	Server	
`DynamicForward`	No default.	Client	
`EscapeChar`	`~`	Client	

TABLE 20–2 Keywords in Solaris Secure Shell Configuration Files (Fall to Local)

Keyword	Default Value	Location	Protocol
`FallBackToRsh`	`no`	Client	
`ForwardAgent`	`no`	Client	
`ForwardX11`	`no`	Client	
`GatewayPorts`	`no`	Both	
`GlobalKnownHostsFile`	`/etc/ssh/ssh_known_hosts`	Client	

TABLE 20–2 Keywords in Solaris Secure Shell Configuration Files (Fall to Local) *(Continued)*

Keyword	Default Value	Location	Protocol
`GSSAPIAuthentication`	`yes`	Both	v2
`GSSAPIDelegateCredentials`	`no`	Client	v2
`GSSAPIKeyExchange`	`yes`	Both	v2
`GSSAPIStoreDelegateCredentials`	`no`	Client	v2
`Host`	* For more information, see "Host-Specific Parameters in Solaris Secure Shell" on page 354.	Client	
`HostbasedAuthentication`	`no`	Both	v2
`HostbasedUsesNamesFromPacketOnly`	`no`	Server	v2
`HostKey`	`/etc/ssh/ssh_host_key`	Server	v1
`HostKey`	`/etc/ssh/host_rsa_key`, `/etc/ssh/host_dsa_key`	Server	v2
`HostKeyAlgorithms`	`ssh-rsa, ssh-dss`	Client	v2
`HostKeyAlias`	No default.	Client	v2
`IdentityFile`	`~/.ssh/identity`	Client	v1
`IdentityFile`	`~/.ssh/id_dsa, ~/.ssh/id_rsa`	Client	v2
`IgnoreRhosts`	`yes`	Server	
`IgnoreUserKnownHosts`	`yes`	Server	
`KbdInteractiveAuthentication`	`yes`	Both	
`KeepAlive`	`yes`	Both	
`KeyRegenerationInterval`	`3600` (seconds)	Server	
`ListenAddress`	No default.	Server	
`LocalForward`	No default.	Client	

TABLE 20–3 Keywords in Solaris Secure Shell Configuration Files (Login to R)

Keyword	Default Value	Location	Protocol
`LoginGraceTime`	`600` (seconds)	Server	
`LogLevel`	`info`	Both	
`LookupClientHostname`	`yes`	Server	

TABLE 20–3 Keywords in Solaris Secure Shell Configuration Files (Login to R) *(Continued)*

Keyword	Default Value	Location	Protocol
`MACs`	`hmac-sha1,hmac-md5`	Both	v2
`MaxAuthTries`	`6`	Server	
`MaxAuthTriesLog`	No default.	Server	
`MaxStartups`	`10:30:60`	Server	
`NoHostAuthenticationForLocalHost`	`no`	Client	
`NumberOfPasswordPrompts`	`3`	Client	
`PAMAuthenticationViaKBDInt`	`yes`	Server	v2
`PasswordAuthentication`	`yes`	Both	
`PermitEmptyPasswords`	`no`	Server	
`PermitRootLogin`	`no`	Server	
`PermitUserEnvironment`	`no`	Server	
`PreferredAuthentications`	`gssapi-keyex, gssapi-with-mic, hostbased, publickey, keyboard-interactive, password`	Client	v2
`Port`	`22`	Both	
`PrintMotd`	`no`	Server	
`Protocol`	`2`	Both	
`ProxyCommand`	No default.	Client	
`PubkeyAuthentication`	`yes`	Both	v2
`RemoteForward`	No default.	Client	
`RhostsAuthentication`	`no`	Both	v1
`RhostsRSAAuthentication`	`no`	Both	v1
`RSAAuthentication`	`no`	Both	v1

TABLE 20–4 Keywords in Solaris Secure Shell Configuration Files (S to X)

Keyword	Default Value	Location	Protocol
`ServerKeyBits`	`768`	Server	
`StrictHostKeyChecking`	`ask`	Client	

TABLE 20–4 Keywords in Solaris Secure Shell Configuration Files (S to X) *(Continued)*

Keyword	Default Value	Location	Protocol
`StrictModes`	`yes`	Server	
`Subsystem`	`sftp /usr/lib/ssh/sftp-server`	Server	
`SyslogFacility`	`auth`	Server	
`UseLogin`	`no` Deprecated and ignored.	Server	
`User`	No default.	Client	
`UserKnownHostsFile`	`~/.ssh/known_hosts`	Client	
`VerifyReverseMapping`	`no`	Server	
`X11Forwarding`	`yes`	Server	
`X11DisplayOffset`	`10`	Server	
`X11UseLocalHost`	`yes`	Server	
`XAuthLocation`	No default.	Both	

Host-Specific Parameters in Solaris Secure Shell

If it is useful to have different Solaris Secure Shell characteristics for different local hosts, the administrator can define separate sets of parameters in the `/etc/ssh/ssh_config` file to be applied according to host or regular expression. This task is done by grouping entries in the file by `Host` keyword. If the `Host` keyword is not used, the entries in the client configuration file apply to whichever local host a user is working on.

Solaris Secure Shell and Login Environment Variables

When the following Solaris Secure Shell keywords are not set in the `sshd_config` file, they get their value from equivalent entries in the `/etc/default/login` file:

Entry in `/etc/default/login`	Keyword and Value in `sshd_config`
`CONSOLE=*`	`PermitRootLogin=without-password`
`#CONSOLE=*`	`PermitRootLogin=yes`
`PASSREQ=YES`	`PermitEmptyPasswords=no`
`PASSREQ=NO`	`PermitEmptyPasswords=yes`

Entry in `/etc/default/login`	**Keyword and Value in** `sshd_config`
`#PASSREQ`	`PermitEmptyPasswords=no`
`TIMEOUT=`*secs*	`LoginGraceTime=`*secs*
`#TIMEOUT`	`LoginGraceTime=300`
`RETRIES` and `SYSLOG_FAILED_LOGINS`	Apply only to `password` and `keyboard-interactive` authentication methods.

When the following variables are set by the `login` command, the `sshd` daemon uses those values. When the variables are not set, the daemon uses the default value.

`TIMEZONE`	Controls the setting of the `TZ` environment variable. When not set, the `sshd` daemon uses value of `TZ` when the daemon was started.
`ALTSHELL`	Controls the setting of the `SHELL` environment variable. The default is `ALTSHELL=YES`, where the `sshd` daemon uses the value of the user's shell. When `ALTSHELL=NO`, the `SHELL` value is not set.
`PATH`	Controls the setting of the `PATH` environment variable. When the value is not set, the default path is `/usr/bin`.
`SUPATH`	Controls the setting of the `PATH` environment variable for `root`. When the value is not set, the default path is `/usr/sbin:/usr/bin`.

For more information, see the `login`(1) and `sshd`(1M) man pages.

Maintaining Known Hosts in Solaris Secure Shell

Each host that needs to communicate securely with another host must have the server's public key stored in the local host's `/etc/ssh/ssh_known_hosts` file. Although a script could be used to update the `/etc/ssh/ssh_known_hosts` files, such a practice is heavily discouraged because a script opens a major security vulnerability.

The `/etc/ssh/ssh_known_hosts` file should only be distributed by a secure mechanism as follows:

- Over a secure connection, such as Solaris Secure Shell, IPsec, or Kerberized `ftp` from a known and trusted machine
- At system install time

To avoid the possibility of an intruder gaining access by inserting bogus public keys into a `known_hosts` file, you should use a JumpStart™ server as the known and trusted source of the `ssh_known_hosts` file. The `ssh_known_hosts` file can be distributed during installation. Later, scripts that use the `scp` command can be used to pull in the latest version. This approach is secure because each host already has the public key from the JumpStart server.

Solaris Secure Shell Packages and Initialization

Solaris Secure Shell depends on core Solaris packages and the following packages:

- `SUNWgss` – Contains Generic Security Service (GSS) software
- `SUNWtcpd` – Contains TCP wrappers
- `SUNWopenssl-libraries` – Contains OpenSSL libraries
- `SUNWzlib` – Contains the zip compression library

The following packages install Solaris Secure Shell:

- `SUNWsshr` – Contains client files and utilities for the root (`/`) directory
- `SUNWsshdr` – Contains server files and utilities for the root (`/`) directory
- `SUNWsshcu` – Contains common source files for the `/usr` directory
- `SUNWsshdu` – Contains server files for the `/usr` directory
- `SUNWsshu` – Contains client files and utilities for the `/usr` directory

Upon reboot after installation, the `sshd` daemon is running. The daemon creates host keys on the system. A Solaris system that runs the `sshd` daemon is a Solaris Secure Shell server.

Solaris Secure Shell Files

The following table shows the important Solaris Secure Shell files and the suggested file permissions.

TABLE 20–5 Solaris Secure Shell Files

File Name	Description	Suggested Permissions and Owner
`/etc/ssh/sshd_config`	Contains configuration data for `sshd`, the Solaris Secure Shell daemon.	`-rw-r--r-- root`
`/etc/ssh/ssh_host_key`	Contains the host private key (v1).	`-rw-r--r-- root`
`/etc/ssh/ssh_host_dsa_key` or `/etc/ssh/ssh_host_rsa_key`	Contains the host private key (v2).	`-rw-r--r-- root`
host-private-key`.pub`	Contains the host public key, for example, `/etc/ssh/ssh_host_rsa_key.pub`. Is used to copy the host key to the local `known_hosts` file.	`-rw-r--r-- root`

TABLE 20–5 Solaris Secure Shell Files *(Continued)*

File Name	Description	Suggested Permissions and Owner
`/var/run/sshd.pid`	Contains the process ID of the Solaris Secure Shell daemon, `sshd`. If multiple daemons are running, the file contains the last daemon that was started.	`-rw-r--r-- root`
`~/.ssh/authorized_keys`	Holds the public keys of the user who is allowed to log in to the user account.	`-rw-rw-r--` *username*
`/etc/ssh/ssh_known_hosts`	Contains the host public keys for all hosts with which the client can communicate securely. The file is populated by the administrator.	`-rw-r--r-- root`
`~/.ssh/known_hosts`	Contains the host public keys for all hosts with which the client can communicate securely. The file is maintained automatically. Whenever the user connects with an unknown host, the remote host key is added to the file.	`-rw-r--r--` *username*
`/etc/default/login`	Provides defaults for the `sshd` daemon when corresponding `sshd_config` parameters are not set.	`-r--r--r-- root`
`/etc/nologin`	If this file exists, the `sshd` daemon only permits `root` to log in. The contents of this file are displayed to users who are attempting to log in.	`-rw-r--r-- root`
`~/.rhosts`	Contains the host-user name pairs that specify the hosts to which the user can log in without a password. This file is also used by the `rlogind` and `rshd` daemons.	`-rw-r--r--` *username*
`~/.shosts`	Contains the host-user name pairs that specify the hosts to which the user can log in without a password. This file is not used by other utilities. For more information, see the `sshd`(1M)man page in the `FILES` section.	`-rw-r--r--` *username*
`/etc/hosts.equiv`	Contains the hosts that are used in `.rhosts` authentication. This file is also used by the `rlogind` and `rshd` daemons.	`-rw-r--r-- root`
`/etc/ssh/shosts.equiv`	Contains the hosts that are used in host-based authentication. This file is not used by other utilities.	`-rw-r--r-- root`
`~/.ssh/environment`	Contains initial assignments at login. By default, this file is not read. The `PermitUserEnvironment` keyword in the `sshd_config` file must be set to `yes` for this file to be read.	`-rw-------` *username*
`~/.ssh/rc`	Contains initialization routines that are run before the user shell starts. For a sample initialization routine, see the sshd man page.	`-rw-------` *username*

TABLE 20–5 Solaris Secure Shell Files *(Continued)*

File Name	Description	Suggested Permissions and Owner
`/etc/ssh/sshrc`	Contains host-specific initialization routines that are specified by an administrator.	`-rw-r--r-- root`
`/etc/ssh/ssh_config`	Configures system settings on the client system.	`-rw-r--r-- root`
`~/.ssh/config`	Configures user settings. Overrides system settings.	`-rw-------` *username*

The following table lists the Solaris Secure Shell files that can be overridden by keywords or command options.

TABLE 20–6 Overrides for the Location of Solaris Secure Shell Files

File Name	Keyword Override	Command-Line Override
`/etc/ssh/ssh_config`		`ssh -F` *config-file* `scp -F` *config-file*
`~/.ssh/config`		`ssh -F` *config-file*
`/etc/ssh/host_rsa_key` `/etc/ssh/host_dsa_key`	`HostKey`	
`~/.ssh/identity` `~/.ssh/id_dsa ~/.ssh/id_rsa`	`IdentityFile`	`ssh -i` *id-file* `scp -i` *id-file*
`~/.ssh/authorized_keys`	`AuthorizedKeysFile`	
`/etc/ssh/ssh_known_hosts`	`GlobalKnownHostsFile`	
`~/.ssh/known_hosts`	`UserKnownHostsFile` `IgnoreUserKnownHosts`	

Solaris Secure Shell Commands

The following table summarizes the major Solaris Secure Shell commands.

TABLE 20–7 Commands in Solaris Secure Shell

Command	Description	Man Page
`ssh`	Logs a user in to a remote machine and securely executes commands on a remote machine. This command is the Solaris Secure Shell replacement for the `rlogin` and `rsh` commands. The `ssh` command enables secure encrypted communications between two untrusted hosts over an insecure network. X11 connections and arbitrary TCP/IP ports can also be forwarded over the secure channel.	`ssh`(1)
`sshd`	Is the daemon for Solaris Secure Shell. The daemon listens for connections from clients and enables secure encrypted communications between two untrusted hosts over an insecure network.	`sshd`(1M)
`ssh-add`	Adds RSA or DSA identities to the authentication agent, `ssh-agent`. Identities are also called *keys*.	`ssh-add`(1)
`ssh-agent`	Holds private keys that are used for public key authentication. The `ssh-agent` program is started at the beginning of an X-session or a login session. All other windows and other programs are started as clients of the `ssh-agent` program. Through the use of environment variables, the agent can be located and used for authentication when users use the `ssh` command to log in to other systems.	`ssh-agent`(1)
`ssh-keygen`	Generates and manages authentication keys for Solaris Secure Shell.	`ssh-keygen`(1)
`ssh-keyscan`	Gathers the public keys of a number of Solaris Secure Shell hosts. Aids in building and verifying `ssh_known_hosts` files.	`ssh-keyscan`(1)
`ssh-keysign`	Is used by the `ssh` command to access the host keys on the local host. Generates the digital signature that is required during host-based authentication with Solaris Secure Shell v2. The command is invoked by the `ssh` command, not by the user.	`ssh-keysign`(1M)
`scp`	Securely copies files between hosts on a network over an encrypted `ssh` transport. Unlike the `rcp` command, the `scp` command prompts for passwords or passphrases, if password information is needed for authentication.	`scp`(1)
`sftp`	Is an interactive file transfer program that is similar to the `ftp` command. Unlike the `ftp` command, the `sftp` command performs all operations over an encrypted `ssh` transport. The command connects, logs in to the specified host name, and then enters interactive command mode.	`sftp`(1)

The following table lists the command options that override Solaris Secure Shell keywords. The keywords are specified in the `ssh_config` and `sshd_config` files.

TABLE 20–8 Command-Line Equivalents for Solaris Secure Shell Keywords

Keyword	`ssh` Command-Line Override	`scp` Command-Line Override
`BatchMode`		`scp -B`
`BindAddress`	`ssh -b` *bind-addr*	`scp -a` *bind-addr*

TABLE 20–8 Command-Line Equivalents for Solaris Secure Shell Keywords *(Continued)*

Keyword	`ssh` Command-Line Override	`scp` Command-Line Override
`Cipher`	`ssh -c` *cipher*	`scp -c` *cipher*
`Ciphers`	`ssh -c` *cipher-spec*	`scp -c` *cipher-spec*
`Compression`	`ssh -C`	`scp -C`
`DynamicForward`	`ssh -D` *SOCKS4-port*	
`EscapeChar`	`ssh -e` *escape-char*	
`ForwardAgent`	`ssh -A` to enable `ssh -a` to disable	
`ForwardX11`	`ssh -X` to enable `ssh -x` to disable	
`GatewayPorts`	`ssh -g`	
`IPv4`	`ssh -4`	`scp -4`
`IPv6`	`ssh -6`	`scp -6`
`LocalForward`	`ssh -L` *localport:remotehost:remoteport*	
`MACS`	`ssh -m` *mac-spec*	
`Port`	`ssh -p` *port*	`scp -P` *port*
`Protocol`	`ssh -1` for v1 only `ssh -2` for v2 only	
`RemoteForward`	`ssh -R` *remoteport:localhost:localport*	

PART VI

Kerberos Service

This section provides information on the configuration, management and use of the Kerberos service.

◆ ◆ ◆ CHAPTER 21

Introduction to the Kerberos Service

This chapter introduces the Kerberos Service. The following is a list of the overview information in this chapter.

- "What Is the Kerberos Service?" on page 363
- "How the Kerberos Service Works" on page 364
- "Kerberos Security Services" on page 371
- "The Components of Various Kerberos Releases" on page 371

What Is the Kerberos Service?

The *Kerberos service* is a client-server architecture that provides secure transactions over networks. The service offers strong user authentication, as well as integrity and privacy. *Authentication* guarantees that the identities of both the sender and the recipient of a network transaction are true. The service can also verify the validity of data being passed back and forth (*integrity*) and encrypt the data during transmission (*privacy*). Using the Kerberos service, you can log in to other machines, execute commands, exchange data, and transfer files securely. Additionally, the service provides *authorization* services, which allows administrators to restrict access to services and machines. Moreover, as a Kerberos user, you can regulate other people's access to your account.

The Kerberos service is a *single-sign-on* system, which means that you only need to authenticate yourself to the service once per session, and all subsequent transactions during the session are automatically secured. After the service has authenticated you, you do not need to authenticate yourself every time you use a Kerberos-based command such as `ftp` or `rsh`, or to access data on an NFS file system. Thus, you do not have to send your password over the network, where it can be intercepted, each time you use these services.

The Solaris Kerberos service is based on the Kerberos V5 network authentication protocol that was developed at the Massachusetts Institute of Technology (MIT). People who have used Kerberos V5 product should therefore find the Solaris version very familiar. Because the Kerberos V5 protocol is a *de facto* industry standard for network security, the Solaris version promotes interoperability with other systems. In other words, because the Solaris Kerberos service works with systems that use the Kerberos V5 protocol, the service allows for secure transactions even over heterogeneous networks. Moreover, the service provides authentication and security both between domains and within a single domain.

The Kerberos service allows for flexibility in running Solaris applications. You can configure the service to allow both Kerberos-based and non-Kerberos-based requests for network services such as the NFS service, `telnet`, and `ftp`. As a result, current Solaris applications still work even if they are running on systems on which the Kerberos service is not enabled. Of course, you can also configure the Kerberos service to allow only Kerberos-based network requests.

The Kerberos service provides a security mechanism which allows the use of Kerberos for authentication, integrity, and privacy when using applications that use the Generic Security Service Application Programming Interface (GSS-API). However, applications do not have to remain committed to the Kerberos service if other security mechanisms are developed. Because the service is designed to integrate modularly into the GSS-API, applications that use the GSS-API can utilize whichever security mechanism best suits their needs.

How the Kerberos Service Works

The following is an overview of the Kerberos authentication system. For a more detailed description, see "How the Kerberos Authentication System Works" on page 537.

From the user's standpoint, the Kerberos service is mostly invisible after the Kerberos session has been started. Commands such as `rsh` or `ftp` work about the same. Initializing a Kerberos session often involves no more than logging in and providing a Kerberos password.

The Kerberos system revolves around the concept of a *ticket*. A ticket is a set of electronic information that identifies a user or a service such as the NFS service. Just as your driver's license identifies you and indicates what driving privileges you have, so a ticket identifies you and your network access privileges. When you perform a Kerberos-based transaction (for example, if you remote log in to another machine), you transparently send a request for a ticket to a *Key Distribution Center*, or KDC. The KDC accesses a database to authenticate your identity and returns a ticket that grants you permission to access the other machine. "Transparently" means that you do not need to explicitly request a ticket. The request happens as part of the `rlogin` command. Because only an authenticated client can get a ticket for a specific service, another client cannot use `rlogin` under an assumed identity.

Tickets have certain attributes associated with them. For example, a ticket can be *forwardable*, which means that it can be used on another machine without a new authentication process. A ticket can also be *postdated*, which means that it is not valid until a specified time. How tickets can be used, for example, to specify which users are allowed to obtain which types of ticket, is set by *policies*. Policies are determined when the Kerberos service is installed or administered.

Note – You will frequently see the terms *credential* and *ticket*. In the greater Kerberos world, they are often used interchangeably. Technically, however, a credential is a ticket plus the *session key* for that session. This difference is explained in more detail in "Gaining Access to a Service Using Kerberos" on page 538.

The following sections further explain the Kerberos authentication process.

Initial Authentication: the Ticket-Granting Ticket

Kerberos authentication has two phases: an initial authentication that allows for all subsequent authentications, and the subsequent authentications themselves.

The following figure shows how the initial authentication takes place.

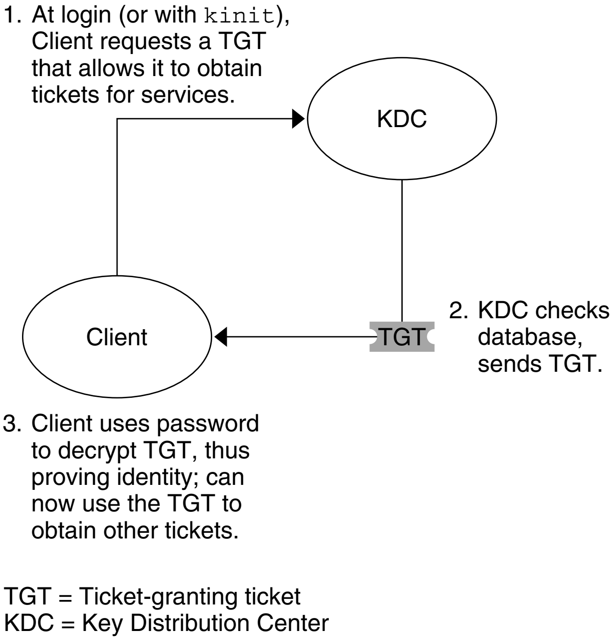

1. At login (or with kinit), Client requests a TGT that allows it to obtain tickets for services.
KDC
Client
TGT
2. KDC checks database, sends TGT.
3. Client uses password to decrypt TGT, thus proving identity; can now use the TGT to obtain other tickets.
TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 21–1 Initial Authentication for a Kerberos Session

1. A client (a user, or a service such as NFS) begins a Kerberos session by requesting a *ticket-granting ticket* (TGT) from the Key Distribution Center (KDC). This request is often done automatically at login.

A ticket-granting ticket is needed to obtain other tickets for specific services. Think of the ticket-granting ticket as similar to a passport. Like a passport, the ticket-granting ticket identifies you and allows you to obtain numerous "visas," where the "visas" (tickets) are not for foreign countries but for remote machines or network services. Like passports and visas, the ticket-granting ticket and the other various tickets have limited lifetimes. The difference is that "Kerberized" commands notice that you have a passport and obtain the visas for you. You don't have to perform the transactions yourself.

Another analogy for the ticket-granting ticket is that of a three-day ski pass that is good at four different ski resorts. You show the pass at whichever resort you decide to go to and you receive a lift ticket for that resort, as long as the pass has not expired. Once you have the lift ticket, you can ski all you want at that resort. If you go to another resort the next day, you once again show your pass, and you get an additional lift ticket for the new resort. The difference is that the Kerberos-based commands notice that you have the weekend ski pass, and they get the lift ticket for you. So you don't have to perform the transactions yourself.

2. The KDC creates a ticket–granting ticket and sends it back, in encrypted form, to the client. The client decrypts the ticket-granting ticket by using the client's password.
3. Now in possession of a valid ticket-granting ticket, the client can request tickets for all sorts of network operations, such as `rlogin` or `telnet`, for as long as the ticket-granting ticket lasts. This ticket usually lasts for a few hours. Each time the client performs a unique network operation, it requests a ticket for that operation from the KDC.

Subsequent Kerberos Authentications

After the client has received the initial authentication, each subsequent authentication follows the pattern that is shown in the following figure.

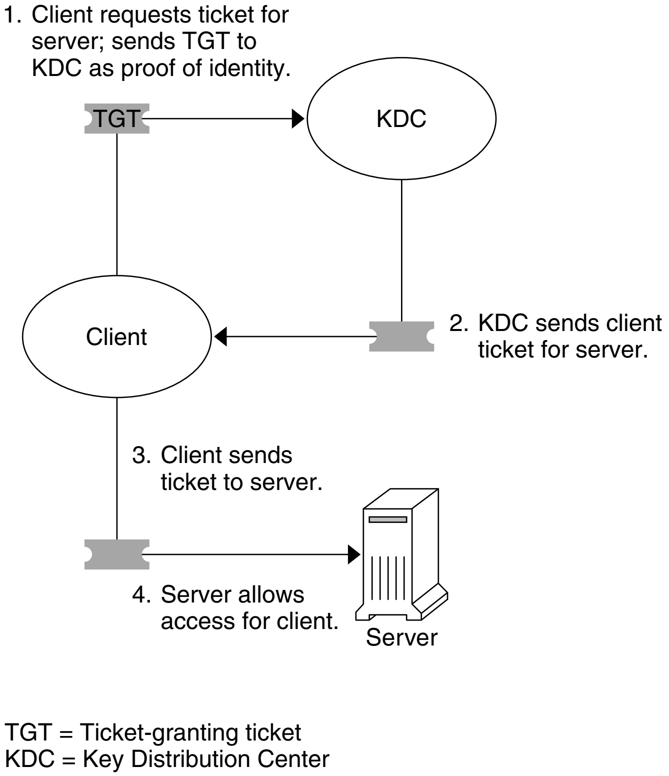

1. Client requests ticket for server; sends TGT to KDC as proof of identity.
TGT
KDC
Client
2. KDC sends client ticket for server.
3. Client sends ticket to server.
4. Server allows access for client.
Server
TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 21–2 Obtaining Access to a Service Using Kerberos Authentication

1. The client requests a ticket for a particular service, for example, to remote log in to another machine, from the KDC by sending the KDC its ticket-granting ticket as proof of identity.
2. The KDC sends the ticket for the specific service to the client.

   For example, suppose user `joe` wants to access an NFS file system that has been shared with `krb5` authentication required. Because he is already authenticated (that is, he already has a ticket-granting ticket), as he attempts to access the files, the NFS client system automatically and transparently obtains a ticket from the KDC for the NFS service.

   For example, suppose the user `joe` uses `rlogin` on the server `boston`. Because he is already authenticated, that is, he already has a ticket-granting ticket, he automatically and transparently obtains a ticket as part of the `rlogin` command. This ticket allows him to remote log in to `boston` as often as he wants until the ticket expires. If `joe` wants to remote log in to the machine `denver`, he obtains another ticket, as in Step 1.
3. The client sends the ticket to the server.

   When using the NFS service, the NFS client automatically and transparently sends the ticket for the NFS service to the NFS server.

4. The server allows the client access.

These steps make it appear that the server doesn't ever communicate with the KDC. The server does, though; it registers itself with the KDC, just as the first client does. For simplicity's sake, that part has been left out.

The Kerberos Remote Applications

The Kerberos-based (or "Kerberized") commands that a user such as `joe` can use are the following:

- `ftp`
- `rcp`
- `rdist`
- `rlogin`
- `rsh`
- `ssh`
- `telnet`

These applications are the same as the Solaris applications of the same name. However, they have been extended to use Kerberos principals to authenticate transactions, thereby providing Kerberos-based security. See "Kerberos Principals" on page 368 for information on principals.

These commands are discussed further in "Kerberos User Commands" on page 522.

Kerberos Principals

A client in the Kerberos service is identified by its *principal*. A principal is a unique identity to which the KDC can assign tickets. A principal can be a user, such as `joe`, or a service, such as `nfs` or `telnet`.

By convention, a principal name is divided into three components: the *primary*, the *instance*, and the *realm*. A typical Kerberos principal would be, for example, `joe/admin@ENG.EXAMPLE.COM`. In this example:

- `joe` is the primary. The primary can be a user name, as shown here, or a service, such as `nfs`. The primary can also be the word `host`, which signifies that this principal is a service principal that is set up to provide various network services, `ftp`, `rcp`, `rlogin`, and so on.
- `admin` is the instance. An instance is optional in the case of user principals, but it is required for service principals. For example, if the user `joe` sometimes acts as a system administrator, he can use `joe/admin` to distinguish himself from his usual user identity. Likewise, if `joe` has accounts on two different hosts, he can use two principal names with different instances, for example, `joe/denver.example.com` and `joe/boston.example.com`. Notice that the Kerberos service treats `joe` and `joe/admin` as two completely different principals.

  In the case of a service principal, the instance is the fully qualified host name. `bigmachine.eng.example.com` is an example of such an instance. The primary/instance for this example might be `ftp/bigmachine.eng.example.com` or `host/bigmachine.eng.example.com`.
- `ENG.EXAMPLE.COM` is the Kerberos realm. Realms are discussed in "Kerberos Realms" on page 369.

The following are all valid principal names:

- `joe`
- `joe/admin`
- `joe/admin@ENG.EXAMPLE.COM`
- `nfs/host.eng.example.com@ENG.EXAMPLE.COM`
- `host/eng.example.com@ENG.EXAMPLE.COM`

Kerberos Realms

A *realm* is a logical network, similar to a domain, that defines a group of systems under the same *master KDC*. Figure 21–3 shows how realms can relate to one another. Some realms are hierarchical, where one realm is a superset of the other realm. Otherwise, the realms are nonhierarchical (or "direct") and the mapping between the two realms must be defined. A feature of the Kerberos service is that it permits authentication across realms. Each realm only needs to have a principal entry for the other realm in its KDC. This Kerberos feature is called *cross-realm authentication*.

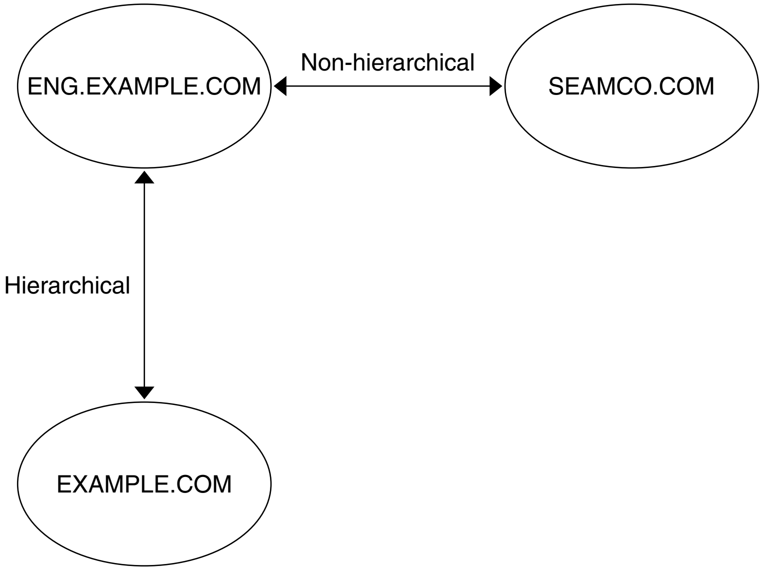


FIGURE 21–3 Kerberos Realms

Kerberos Servers

Each realm must include a server that maintains the master copy of the principal database. This server is called the *master KDC server*. Additionally, each realm should contain at least one *slave KDC server*, which contains duplicate copies of the principal database. Both the master KDC server and the slave KDC server create tickets that are used to establish authentication.

The realm can also include a Kerberos *application server*. This server provides access to Kerberized services (such as `ftp`, `telnet`, `rsh` and NFS). If you have installed SEAM 1.0 or 1.0.1, the realm might include a Kerberos network application server, but this software was not included with these releases.

The following figure shows what a hypothetical realm might contain.

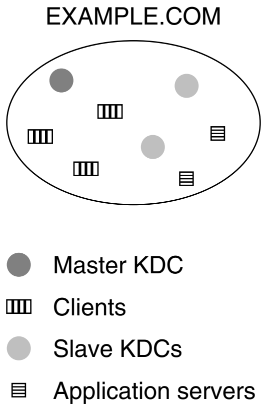


FIGURE 21–4 A Typical Kerberos Realm

Kerberos Security Services

In addition to providing secure authentication of users, the Kerberos service provides two security services:

- **Integrity –** Just as authentication ensures that clients on a network are who they claim to be, integrity ensures that the data they send is valid and has not been tampered with during transit. Integrity is done through cryptographic checksumming of the data. Integrity also includes user authentication.
- **Privacy –** Privacy takes security a step further. Privacy not only includes verifying the integrity of transmitted data, but it encrypts the data before transmission, protecting it from eavesdroppers. Privacy authenticates users, as well.

Developers can design their RPC-based applications to choose a security service by using the RPCSEC_GSS programming interface.

The Components of Various Kerberos Releases

Components of the Kerberos service have been included in many releases. Originally, the Kerberos service and changes to the base operating system to support the Kerberos service were released using the product name “Sun Enterprise Authentication Mechanism” which was shortened to SEAM. As more parts of the SEAM product were included in the Solaris software, the contents of the SEAM release decreased. For the Solaris 10 release, all parts of the SEAM product are included, so there is no longer a need for the SEAM product. The SEAM product name exists in the documentation for historical reasons.

The following table describes which components are included in each release. Each product release is listed in chronological order. All components are described in the following sections.

TABLE 21–1 Kerberos Release Contents

Release Name	Contents
SEAM 1.0 in Solaris Easy Access Server 3.0	Full release of the Kerberos service for the Solaris 2.6 and 7 releases
The Kerberos service in the Solaris 8 release	Kerberos client software only
SEAM 1.0.1 in the Solaris 8 Admin Pack	Kerberos KDC and remote applications for the Solaris 8 release
The Kerberos service in the Solaris 9 release	Kerberos KDC and client software only
SEAM 1.0.2	Kerberos remote applications for the Solaris 9 release
The Kerberos service in the Solaris 10 release	Full release of the Kerberos service with enhancements

Kerberos Components

Similar to the MIT distribution of the Kerberos V5 product, the Solaris Kerberos service includes the following:

- Key Distribution Center (KDC):
  - Kerberos database administration daemon – `kadmind`.
  - Kerberos ticket processing daemon – `krb5kdc`.
  - Database administration programs – `kadmin` (master only), `kadmin.local` and `kdb5_util`.
  - Database propagation software – `kprop` (slave only) and `kpropd`.
- User programs for managing credentials – `kinit`, `klist`, and `kdestroy`.
- User program for changing your Kerberos password – `kpasswd`.
- Remote applications – `ftp`, `rcp`, `rdist`, `rlogin`, `rsh`, `ssh`, and `telnet`.
- Remote application daemons – `ftpd`, `rlogind`, `rshd`, `sshd`, and `telnetd`.
- Keytab administration utility – `ktutil`.
- The Generic Security Service Application Programming Interface (GSS-API) – Enables applications to use multiple security mechanisms without requiring you to recompile the application every time a new mechanism is added. The GSS-API uses standard interfaces that allow applications to be portable to many operating systems. GSS-API provides applications with the ability to include the integrity and privacy security services, as well as authentication. Both `ftp` and `ssh` use the GSS-API.

- The RPCSEC_GSS Application Programming Interface (API) – Enables NFS services to use Kerberos authentication. RPCSEC_GSS is a security flavor that provides security services that are independent of the mechanisms being used. RPCSEC_GSS sits on top of the GSS-API layer. Any pluggable GSS_API-based security mechanism can be used by applications that use RPCSEC_GSS.

In addition, the Solaris Kerberos service includes the following:

- Graphical Kerberos Administration Tool (`gkadmin`) – Enables you to administer the principals and principal policies. This Java™ technology-based GUI is an alternative to the `kadmin` command.
- A Kerberos V5 service module for PAM – Provides authentication, account management, session management and password management for the Kerberos service. The module can be used to make Kerberos authentication transparent to the user.
- Kernel modules – Provides kernel-based implementations of the kerberos service for use by the NFS service, which greatly improves performance.

Kerberos Additions for the Solaris 10 5/08 Release

These enhancements are available starting in the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases:

- The Solaris Kerberos software has been synchronized with the MIT 1.4 version. In particular, the software for the KDC, the `kinit` command and the Kerberos mechanism have been updated.
- Support for accessing Kerberos principal and policy records using LDAP from a directory server has been added. This change simplifies administration and can provide greater availability, depending on the deployment of the KDCs and the DSs.
- The new `kdcmgr` command which can be used to automatically or interactively setup any KDC. This command works to create both master and slave KDC servers. Also, used with the `status` option, the `kdcmgr` command shows information about any KDC installed on the localhost.
- Support for Solaris clients that require no additional setup has been added to this release. Changes were made to the Kerberos service and to some default settings. Solaris Kerberos clients work with no client-side configuration in environments that are appropriately configured.

Kerberos Additions for the Solaris 10 8/07 Release

The MIT Kerberos V5 application programming interface (krb5-api) is supported in the Solaris 10 8/07 release. See the `libkrb5`(3LIB) and `krb5-config`(1) man pages for more information. Also, see the MIT Kerberos V5 project web pages at mit.edu for more detailed documentation as it becomes available.

Although the krb5-api is now available, Sun strongly encourages the use of the GSS-API for network authentication and integrity and privacy as the GSS-API is security-mechanism independent and an IETF standard. See the `libgss`(3LIB) man page for more information.

Kerberos Additions for the Solaris 10 6/06 Release

In the Solaris 10 6/06 release, the `ktkt_warnd` daemon can automatically renew credentials, rather than just warn the user when the credential is about to expire. The user must be logged in for the credential to be renewed automatically.

Kerberos Enhancements in the Solaris 10 3/05 Release

These Kerberos enhancements are included in the Solaris 10 Release. Several of the enhancements were introduced in prior Software Express releases and updated in the Solaris 10 Beta releases.

- Kerberos protocol support is provided in remote applications, such as `ftp`, `rcp`, `rdist`, `rlogin`, `rsh`, `ssh`, and `telnet`. See the man pages for each command or daemon and the `krb5_auth_rules`(5) man page for more information.
- The Kerberos principal database can now be transferred by incremental update instead of by transferring the entire database each time. Incremental propagation provides these advantages:
  - Increased database consistencies across servers
  - The need for fewer resources (network, CPU, and so forth)
  - Much more timely propagation of updates
  - An automated method of propagation
- A new script to help automatically configure a Kerberos client is now available. The script helps an administrator quickly and easily set up a Kerberos client. For procedures using the new script, see "Configuring Kerberos Clients" on page 421. Also, see the `kclient`(1M) man page for more information.
- Several new encryption types have been added to the Kerberos service. These new encryption types increase security and enhance compatibility with other Kerberos implementations that support these encryption types. See "Using Kerberos Encryption Types" on page 541 for more information. The encryption types include:
  - The AES encryption type can be used for high speed, high security encryption of Kerberos sessions.
  - ARCFOUR-HMAC provides better compatibility with other Kerberos implementations.
  - Triple DES (3DES) with SHA1 increases security. This encryption type also enhances interoperability with other Kerberos implementations that support this encryption type.
- The encryption types are enabled through the Cryptographic Framework. The framework can provide for hardware accelerated cryptography for the Kerberos service.

- The KDC software, the user commands, and user applications now support the use of the TCP network protocol. This enhancement provides more robust operation and better interoperability with other Kerberos implementations, including Microsoft's Active Directory. The KDC now listens on both the traditional UDP ports as well as TCP ports so it can respond to requests using either protocol. The user commands and applications first try UDP when sending a request to the KDC, and if that fails, then try TCP.
- Support for IPv6 was added to the KDC software, which includes the `kinit`, `klist` and `kprop` commands. Support for IPv6 addresses is provided by default. There are no configuration parameters to change to enable IPv6 support. No IPv6 support is available for the `kadmin` and `kadmind` commands.
- A new `-e` option has been included to several subcommands of the `kadmin` command. This new option allows for the selection of the encryption type during the creation of principals. See the `kadmin`(1M) man page for more information.
- Additions to the `pam_krb5` module to manage the Kerberos credentials cache by using the PAM framework. See the `pam_krb5`(5) man page for more information.
- Support is provided for auto-discovery of the Kerberos KDC, admin server, kpasswd server, and host or domain name-to-realm mappings by using DNS lookups. This enhancement reduces some of the steps needed to install a Kerberos client. The client is able to locate a KDC server by using DNS instead of by reading a configuration file. See the `krb5.conf`(4) man page for more information.
- A new PAM module called `pam_krb5_migrate` has been introduced. The new module helps in the automatic migration of users to the local Kerberos realm, if they do not already have Kerberos accounts. See the `pam_krb5_migrate`(5) man page for more information.
- The `~/.k5login` file can now be used with the GSS applications `ftp` and `ssh`. For more information, see the `gss_auth_rules`(5) man page.
- The `kproplog` utility has been updated to output all attribute names per log entry. For more information, see the `kproplog`(1M) man page.
- Strict TGT verification can now be disabled using a configuration option in the `krb5.conf` file. See the `krb5.conf`(4) man page for more information.
- Extensions to the password-changing utilities enable the Solaris Kerberos V5 administration server to accept password change requests from clients that do not run Solaris software. See the `kadmind`(1M) man page for more information.
- The default location of the replay cache has been moved from RAM-based file systems to persistent storage in `/var/krb5/rcache/`. The new location protects against replays if a system is rebooted. Performance enhancements were made to the rcache code. However, overall replay cache performance might be slower due to the use of persistent storage.
- The replay cache can now be configured to use file or memory only storage. Refer to the `krb5envvar`(5) man page for more information about environment variables that can be configured for key table and credential cache types or locations.

- The GSS credential table is no longer necessary for the Kerberos GSS mechanism. For more information, see "Mapping GSS Credentials to UNIX Credentials" on page 383 or the `gsscred`(1M), `gssd`(1M), and `gsscred.conf`(4) man pages.
- The Kerberos utilities, `kinit` and `ktutil`, are now based on MIT Kerberos version 1.2.1. This change added new options to the `kinit` command and new subcommands to the `ktutil` command. For more information, see the `kinit`(1) and `ktutil`(1) man pages.
- The Solaris Kerberos Key Distribution Center (KDC) and `kadmind` is now based on MIT Kerberos version 1.2.1. The KDC now defaults to a btree-based database, which is more reliable than the current hash-based database. See the `kdb5_util`(1M) man page for more information.
- The `kpropd`, `kadmind`, `krb5kdc` and `ktkt_warnd` daemons are managed by the Service Management Facility. Administrative actions on this service, such as enabling, disabling, or restarting, can be performed using the `svcadm` command. The service's status for all daemons can be queried using the `svcs` command. For an overview of the Service Management Facility refer to Chapter 15, "Managing Services (Overview)," in *System Administration Guide: Basic Administration*.

Kerberos Components in the Solaris 9 Release

The Solaris 9 release includes all components included in "Kerberos Components" on page 372, except for the remote applications.

SEAM 1.0.2 Components

The SEAM 1.0.2 release includes the remote applications. These applications are the only part of SEAM 1.0 that have not been incorporated into the Solaris 9 release. The components for the remote applications are as follows:

- Client applications – `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`
- Server daemons – `ftpd`, `rlogind`, `rshd`, and `telnetd`

Kerberos Components in the Solaris 8 Release

The Solaris 8 release includes only the client-side portions of the Kerberos service, so many components are not included. This product enables systems that run the Solaris 8 release to become Kerberos clients without requiring you to install SEAM 1.0.1 separately. To use these capabilities, you must install a KDC that uses either Solaris Easy Access Server 3.0 or the Solaris 8 Admin Pack, the MIT distribution, or Windows 2000. The client-side components are not useful without a configured KDC to distribute tickets. The following components are included in this release:

- User programs for obtaining, viewing, and destroying tickets – `kinit`, `klist`, and `kdestroy`.
- User program for changing your Kerberos password – `kpasswd`.
- Key table administration utility – `ktutil`.
- Additions to the Pluggable Authentication Module (PAM) – Enables applications to use various authentication mechanisms. PAM can be used to make logins and logouts transparent to the user.
- GSS_API plug–ins – Provides Kerberos protocol and cryptographic support.
- NFS client and server support.

SEAM 1.0.1 Components

The SEAM 1.0.1 release includes all components of the SEAM 1.0 release that are not already included in the Solaris 8 release. The components are as follows:

- Key Distribution Center (KDC) (master):
  - Kerberos database administration daemon – `kadmind`
  - Kerberos ticket processing daemon – `krb5kdc`
- Slave KDCs.
- Database administration programs – `kadmin` and `kadmin.local`.
- Database propagation software – `kprop`.
- Remote applications – `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`.
- Remote application daemons – `ftpd`, `rlogind`, `rshd`, and `telnetd`.
- Administration utility – `kdb5_util`.
- Graphical Kerberos Administration Tool (`gkadmin`) – Enables you to administerprincipals and principal policies. This Java technology-based GUI is an alternative to the `kadmin` command.
- A preconfiguration procedure – Enables you to set the parameters for installing and configuring SEAM 1.0.1, which makes SEAM installation automatic. This procedure is especially useful for multiple installations.
- Several libraries.

SEAM 1.0 Components

The SEAM 1.0 release includes all of the items included in "Kerberos Components" on page 372 as well as the following:

- A utility (`gsscred`) and a daemon (`gssd`) – These programs help map UNIX user IDs (UIDs) to principal names. These programs are needed because NFS servers use UNIX UIDs to identify users and not principal names, which are stored in a different format.
- The Generic Security Service Application Programming Interface (GSS-API) – Enables applications to use multiple security mechanisms without requiring you to recompile the application every time a new mechanism is added. Because GSS-API is machine-independent, it is appropriate for applications on the Internet. GSS-API provides applications with the ability to include the integrity and privacy security services, as well as authentication.
- The RPCSEC_GSS Application Programming Interface (API) – Enables NFS services to use Kerberos authentication. RPCSEC_GSS is a security flavor that provides security services that are independent of the mechanisms being used. RPCSEC_GSS sits on top of the GSS-API layer. Any pluggable GSS_API-based security mechanism can be used by applications that use RPCSEC_GSS.
- A preconfiguration procedure – Enables you to set the parameters for installing and configuring SEAM 1.0, which makes installation automatic. This procedure is especially useful for multiple installations.

◆ ◆ ◆

CHAPTER 22

Planning for the Kerberos Service

This chapter should be studied by administrators who are involved in the installation and maintenance of the Kerberos service. The chapter discusses several installation and configuration options that administrators must resolve before they install or configure the service.

This is a list of the topics that a system administrator or other knowledgeable support staff should study:

- "Why Plan for Kerberos Deployments?" on page 379
- "Planning Kerberos Realms" on page 380
- "Mapping Host Names Onto Realms" on page 381
- "Client and Service Principal Names" on page 381
- "Ports for the KDC and Admin Services" on page 382
- "The Number of Slave KDCs" on page 382
- "Which Database Propagation System to Use" on page 383
- "Clock Synchronization Within a Realm" on page 384
- "Client Configuration Options" on page 384
- "KDC Configuration Options" on page 385
- "Kerberos Encryption Types" on page 385
- "Online Help URL in the Graphical Kerberos Administration Tool" on page 386

Why Plan for Kerberos Deployments?

Before you install the Kerberos service, you must resolve several configuration issues. Although changing the configuration after the initial install is not impossible, some changes can be difficult to implement. In addition, some changes require that the KDC be rebuilt, so it is better to consider long-term goals when you plan your Kerberos configuration.

Deploying a Kerberos infrastructure involves such tasks as installing KDCs, creating keys for your hosts, and migrating users. Reconfiguring a Kerberos deployment can be as hard as performing an initial deployment, so plan a deployment carefully to avoid having to re-configure.

Planning Kerberos Realms

A *realm* is logical network, similar to a domain, that defines a group of systems that are under the same master KDC. As with establishing a DNS domain name, issues such as the realm name, the number and size of each realm, and the relationship of a realm to other realms for cross-realm authentication should be resolved before you configure the Kerberos service.

Realm Names

Realm names can consist of any ASCII string. Usually, the realm name is the same as your DNS domain name, except that the realm name is in uppercase. This convention helps differentiate problems with the Kerberos service from problems with the DNS namespace, while using a name that is familiar. If you do not use DNS or you choose to use a different string, then you can use any string. However, the configuration process requires more work. The use of realm names that follow the standard Internet naming structure is wise.

Number of Realms

The number of realms that your installation requires depends on several factors:

- The number of clients to be supported. Too many clients in one realm makes administration more difficult and eventually requires that you split the realm. The primary factors that determine the number of clients that can be supported are as follows:
  - The amount of Kerberos traffic that each client generates
  - The bandwidth of the physical network
  - The speed of the hosts

  Because each installation will have different limitations, no rule exists for determining the maximum number of clients.
- How far apart the clients are. Setting up several small realms might make sense if the clients are in different geographic regions.
- The number of hosts that are available to be installed as KDCs. Each realm should have at least two KDC servers, one master server and one slave server.

Alignment of Kerberos realms with administrative domains is recommended. Note that a Kerberos V realm can span multiple sub-domains of the DNS domain to which the realm corresponds.

Realm Hierarchy

When you are configuring multiple realms for cross-realm authentication, you need to decide how to tie the realms together. You can establish a hierarchical relationship among the realms, which provides automatic paths to the related domains. Of course, all realms in the hierarchical chain must be configured properly. The automatic paths can ease the administration burden. However, if there are many levels of domains, you might not want to use the default path because it requires too many transactions.

You can also choose to establish the trust relationship directly. A direct trust relationship is most useful when too many levels exist between two hierarchical realms or when no hierarchal relationship exists. The connection must be defined in the `/etc/krb5/krb5.conf` file on all hosts that use the connection. So, some additional work is required. The direct trust relationship is also referred to as a transitive relationship. For an introduction, see "Kerberos Realms" on page 369. For the configuration procedures for multiple realms, see "Configuring Cross-Realm Authentication" on page 410.

Mapping Host Names Onto Realms

The mapping of host names onto realm names is defined in the `domain_realm` section of the `krb5.conf` file. These mappings can be defined for a whole domain and for individual hosts, depending on the requirements.

DNS can also be used to look up information about the KDCs. Using DNS makes it easier to change the information because you will not need to edit the `krb5.conf` file on all of the clients each time you make a change. See the `krb5.conf`(4) man page for more information.

As of the Solaris Express Developer Edition 1/08 and the Solaris 10 5/08 releases, Solaris Kerberos clients can interoperate better with Active Directory servers. The Active Directory servers can be configured to provide the realm to host mapping.

Client and Service Principal Names

When you are using the Kerberos service, DNS must be enabled on all hosts. With DNS, the principal should contain the Fully Qualified Domain Name (FQDN) of each host. For example, if the host name is `boston`, the DNS domain name is `example.com`, and the realm name is `EXAMPLE.COM`, then the principal name for the host should be `host/boston.example.com@EXAMPLE.COM`. The examples in this book require that DNS is configured and use the FQDN for each host.

For the principal names that include the FQDN of a host, it is important to match the string that describes the DNS domain name in the `/etc/resolv.conf` file. The Kerberos service requires that the DNS domain name be in lowercase letters when you are specifying the FQDN for a principal. The DNS domain name can include uppercase and lowercase letters, but only use lowercase letters when you are creating a host principal. For example, it doesn't matter if the DNS domain name is `example.com`, `Example.COM`, or any other variation. The principal name for the host would still be `host/boston.example.com@EXAMPLE.COM`.

In addition, the Service Management Facility has been configured so that many of the daemons or commands do not start if the DNS client service is not running. The `kdb5_util`, `kadmind`, and `kpropd` daemons, as well as the `kprop` command all are configured to depend on the DNS service. To fully utilize the features available using the Kerberos service and SMF, you must enable the DNS client service on all hosts.

Ports for the KDC and Admin Services

By default, port `88` and port `750` are used for the KDC, and port `749` is used for the KDC administration daemon. Different port numbers can be used. However, if you change the port numbers, then the `/etc/services` and `/etc/krb5/krb5.conf` files must be changed on every client. In addition to these files, the `/etc/krb5/kdc.conf` file on each KDC must be updated.

The Number of Slave KDCs

Slave KDCs generate credentials for clients just as the master KDC does. Slave KDCs provide backup if the master becomes unavailable. Each realm should have at least one slave KDC. Additional slave KDCs might be required, depending on these factors:

- The number of physical segments in the realm. Normally, the network should be set up so that each segment can function, at least minimally, without the rest of the realm. To do so, a KDC must be accessible from each segment. The KDC in this instance could be either a master or a slave.
- The number of clients in the realm. By adding more slave KDC servers, you can reduce the load on the current servers.

It is possible to add too many slave KDCs. Remember that the KDC database must be propagated to each server, so the more KDC servers that are installed, the longer it can take to get the data updated throughout the realm. Also, because each slave retains a copy of the KDC database, more slaves increase the risk of a security breach.

In addition, one or more slave KDCs can easily be configured to be swapped with the master KDC. The advantage of configuring at least one slave KDC in this way is that if the master KDC fails for any reason, you will have a system preconfigured that will be easy to swap as the master KDC. For instructions on how to configure a swappable slave KDC, see “Swapping a Master KDC and a Slave KDC” on page 436.

Mapping GSS Credentials to UNIX Credentials

The Kerberos service provides a default mapping of GSS credential names to UNIX user IDs (UIDs) for GSS applications that require this mapping, such as NFS. GSS credential names are equivalent to Kerberos principal names when using the Kerberos service. The default mapping algorithm is to take a one component Kerberos principal name and use that component, which is the primary name of the principal, to look up the UID. The look up occurs in the default realm or any realm that is allowed by using the `auth_to_local_realm` parameter in `/etc/krb5/krb5.conf`. For example, the user principal name `bob@EXAMPLE.COM` is mapped to the UID of the UNIX user named `bob` using the password table. The user principal name `bob/admin@EXAMPLE.COM` would not be mapped, because the principal name includes an instance component of `admin`. If the default mappings for the user credentials are sufficient, the GSS credential table does not need to be populated. In past releases, populating the GSS credential table was required to get the NFS service to work. If the default mapping is not sufficient, for example if you want to map a principal name which contains an instance component, then other methods should be used. For more information see:

- "How to Create a Credential Table" on page 417
- "How to Add a Single Entry to the Credential Table" on page 417
- "How to Provide Credential Mapping Between Realms" on page 418
- "Observing Mapping from GSS Credentials to UNIX Credentials" on page 473

Automatic User Migration to a Kerberos Realm

UNIX users who do not have valid user accounts in the default Kerberos realm can be automatically migrated using the PAM framework. Specifically, the `pam_krb5_migrate` module would be used in the authentication stack of the PAM service. Services would be setup up so that whenever a user, who does not have a Kerberos principal, performs a successful log in to a system using their password, a Kerberos principal would be automatically created for that user. The new principal password would be the same as the UNIX password. See "How to Configure Automatic Migration of Users in a Kerberos Realm" on page 432 for instructions on how to use the `pam_krb5_migrate` module.

Which Database Propagation System to Use

The database that is stored on the master KDC must be regularly propagated to the slave KDCs. You can configure the propagation of the database to be incremental. The incremental process propagates only updated information to the slave KDCs, rather than the entire database. For more information about database propagation, see "Administering the Kerberos Database" on page 441.

If you do not use incremental propagation, one of the first issues to resolve is how often to update the slave KDCs. The need to have up-to-date information that is available to all clients must be weighed against the amount of time it takes to complete the update.

In large installations with many KDCs in one realm, one or more slaves can propagate the data so that the process is done in parallel. This strategy reduces the amount of time that the update takes, but it also increases the level of complexity in administering the realm. For a complete description of this strategy, see "Setting Up Parallel Propagation" on page 454.

Clock Synchronization Within a Realm

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time. Known as *clock skew*, this feature provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, requests are rejected.

One way to synchronize all the clocks is to use the Network Time Protocol (NTP) software. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for more information. Other ways of synchronizing the clocks are available, so the use of NTP is not required. However, some form of synchronization should be used to prevent access failures because of clock skew.

Client Configuration Options

A new feature in the Solaris 10 release is the `kclient` configuration utility. The utility can be run in interactive mode or noninteractive mode. In interactive mode, the user is prompted for Kerberos-specific parameter values, which allows the user to make changes to the existing installation when configuring the client. In noninteractive mode, a file with previously set parameter values is used. Also, command-line options can be used in the noninteractive mode. Both interactive and noninteractive modes require less steps than the manual process, which should make the process quicker and less prone to error.

In the Solaris 10 5/08 release, changes were made to allow for a zero-configuration Kerberos client. If these rules are followed in your environment then no explicit configuration procedure is necessary for a Solaris Kerberos client:

- DNS is configured to return SRV records for KDCs.
- The realm name matches the DNS domain name or the KDC supports referrals.
- The Kerberos client does not require a `keytab`.

In some cases it may be better to explicitly configure the Kerberos client:

- If referrals are not used, the zero-configuration logic depends on the DNS domain name of the host to determine the realm. This introduces a small security risk, but the risk is much smaller than enabling `dns_lookup_realm`.
- The `pam_krb5` module relies on a host key entry in the `keytab`. This requirement may be disabled in the `krb5.conf` file however it is not recommend for security reasons. See the `krb5.conf`(4)man page.
- The zero-configuration process is less efficient than direct configuration, and has a greater reliance on DNS. The process performs more DNS lookups than a directly configured client.

See "Configuring Kerberos Clients" on page 421 for a description of all the client configuration processes.

KDC Configuration Options

Starting in the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, there are several ways to configure a KDC. The simplest versions use the `kdcmgr` utility to configure the KDC automatically or interactively. The automatic version requires that you use command line options to define the configuration parameters. This method is especially useful for scripts. The interactive version prompts you for all information that is needed. See Table 23–1 for pointers to the instructions for using this command.

In addition, starting in the Solaris 10 5/08 and Solaris Express Developer Edition 1/08 releases, support for using LDAP to manage the database files for Kerberos has been added. See "How to Configure a KDC to Use an LDAP Data Server" on page 397 for instructions. Using LDAP simplifies administration for sites that require better coordination between the Solaris Kerberos databases and their existing DS setup.

Kerberos Encryption Types

An *encryption type* is an identifier that specifies the encryption algorithm, encryption mode, and hash algorithms used in the Kerberos service. The keys in the Kerberos service have an associated encryption type to identify the cryptographic algorithm and mode to be used when the service performs cryptographic operations with the key. Here are the supported encryption types:

- `des-cbc-md5`
- `des-cbc-crc`
- `des3-cbc-sha1-kd`
- `arcfour-hmac-md5`
- `arcfour-hmac-md5-exp`
- `aes128-cts-hmac-sha1-96`

- `aes256-cts-hmac-sha1-96`

Note – In addition, the `aes256-cts-hmac-sha1-96` encryption type can be used with the Kerberos service if the unbundled Strong Cryptographic packages are installed.

If you want to change the encryption type, you should do so when creating a new principal database. Because of the interaction between the KDC, the server, and the client, changing the encryption type on an existing database is difficult. Leave these parameters unset unless you are re-creating the database. Refer to "Using Kerberos Encryption Types" on page 541 for more information.

Note – If you have a master KDC installed that is not running the Solaris 10 release, the slave KDCs must be upgraded to the Solaris 10 release before you upgrade the master KDC. A Solaris 10 master KDC will use the new encryption types, which an older slave will not be able to handle.

Online Help URL in the Graphical Kerberos Administration Tool

The online help URL is used by the Graphical Kerberos Administration Tool, `gkadmin`, so the URL should be defined properly to enable the "Help Contents" menu to work. The HTML version of this manual can be installed on any appropriate server. Alternately, you can decide to use the collections at `http://docs.sun.com`.

The URL is specified in the krb5.conf file when configuring a host to use the Kerberos service. The URL should point to the section titled "Graphical Kerberos Administration Tool" in the "Administering Principals and Policies (Tasks)" chapter in this book. You can choose another HTML page, if another location is more appropriate.

◆ ◆ ◆

CHAPTER 23

Configuring the Kerberos Service (Tasks)

This chapter provides configuration procedures for KDC servers, network application servers, NFS servers, and Kerberos clients. Many of these procedures require superuser access, so they should be used by system administrators or advanced users. Cross-realm configuration procedures and other topics related to KDC servers are also covered.

The following topics are covered.

- "Configuring the Kerberos Service (Task Map)" on page 387
- "Configuring KDC Servers" on page 389
- "Configuring Kerberos Clients" on page 421
- "Configuring Cross-Realm Authentication" on page 410
- "Configuring Kerberos Network Application Servers" on page 413
- "Configuring Kerberos NFS Servers" on page 414
- "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434
- "Swapping a Master KDC and a Slave KDC" on page 436
- "Administering the Kerberos Database" on page 441
- "Increasing Security on Kerberos Servers" on page 457

Configuring the Kerberos Service (Task Map)

Parts of the configuration process depend on other parts and must be done in a specific order. These procedures often establish services that are required to use the Kerberos service. Other procedures are not dependent on any order, and can be done when appropriate. The following task map shows a suggested order for a Kerberos installation.

Task	Description	For Instructions
1. Plan for your Kerberos installation.	Lets you resolve configuration issues before you start the software configuration process. Planning ahead saves you time and other resources in the long run.	Chapter 22, "Planning for the Kerberos Service,"
2. (Optional) Install NTP.	Configures the Network Time Protocol (NTP) software, or another clock synchronization protocol. In order for the Kerberos service to work properly, the clocks on all systems in the realm must be synchronized.	"Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434
3. Configure the KDC servers.	Configures and builds the master KDC and the slave KDC servers and the KDC database for a realm.	"Configuring KDC Servers" on page 389
4. (Optional) Increase security on the KDC servers.	Prevents security breaches on the KDC servers.	"How to Restrict Access to KDC Servers" on page 457
5. (Optional) Configure swappable KDC servers.	Makes the task of swapping the master KDC and a slave KDC easier.	"How to Configure a Swappable Slave KDC" on page 436

Configuring Additional Kerberos Services (Task Map)

Once the required steps have been completed, the following procedures can be used, when appropriate.

Task	Description	For Instructions
Configure cross-realm authentication.	Enables communications from one realm to another realm.	"Configuring Cross-Realm Authentication" on page 410
Configure Kerberos application servers.	Enables a server to support services such as `ftp`, `telnet`, and `rsh` using Kerberos authentication.	"Configuring Kerberos Network Application Servers" on page 413
Configure Kerberos clients.	Enables a client to use Kerberos services.	"Configuring Kerberos Clients" on page 421
Configure Kerberos NFS server.	Enables a server to share a file system that requires Kerberos authentication.	"Configuring Kerberos NFS Servers" on page 414
Increase security on an application server.	Increases security on an application server by restricting access to authenticated transactions only.	"How to Enable Only Kerberized Applications" on page 457

Configuring KDC Servers

After you install the Kerberos software, you must configure the KDC servers. Configuring a master KDC and at least one slave KDC provides the service that issues credentials. These credentials are the basis for the Kerberos service, so the KDCs must be installed before you attempt other tasks.

The most significant difference between a master KDC and a slave KDC is that only the master KDC can handle database administration requests. For instance, changing a password or adding a new principal must be done on the master KDC. These changes can then be propagated to the slave KDCs. Both the slave KDC and master KDC generate credentials. This feature provides redundancy in case the master KDC cannot respond.

TABLE 23–1 Configuring KDC Servers (Task Map)

Task	Description	For Instructions
Configuring a Master KDC	Configures and builds the master KDC server and database for a realm using an automatic process, which is good for scripts.	"How to Automatically Configure a Master KDC" on page 390
	Configures and builds the master KDC server and database for a realm using an interactive process, which is sufficient for most installations	"How to Interactively Configure a Master KDC" on page 390
	Configures and builds the master KDC server and database for a realm using a manual process, which is needed for more complex installations	"How to Configure a Master KDC" on page 392
	Configures and builds the master KDC server and database for a realm using a manual process and using LDAP for the KDC	"How to Configure a KDC to Use an LDAP Data Server" on page 397
Configure a slave KDC server.	Configures and builds a slave KDC server for a realm using an automatic process, which is good for scripts	"How to Automatically Configure a Slave KDC" on page 404
	Configures and builds a slave KDC server for a realm using an interactive process, which is sufficient for most installations	"How to Interactively Configure a Slave KDC" on page 405
	Configures and builds a slave KDC server for a realm using the manual process, which is needed for more complex installations	"How to Configure a Slave KDC" on page 406

How to Automatically Configure a Master KDC

Starting with the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, a master KDC can be automatically configured by using the following procedure.

1. **Become superuser.**

2. **Create the KDC.**

   Run the `kdcmgr` utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.

```
kdc1# kdcmgr -a kws/admin -r EXAMPLE.COM create master

Starting server setup
---------------------------------------

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     <Type the password>
Re-enter KDC database master key to verify:     <Type it again>

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM":     <Type the password>
Re-enter password for principal "kws/admin@EXAMPLE.COM":     <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.

Setting up /etc/krb5/kadm5.acl.

---------------------------------------------------
Setup COMPLETE.

kdc1#
```

How to Interactively Configure a Master KDC

Starting with the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, a master KDC can be interactively configured by using the following procedure.

1. **Become superuser.**

2 Create the KDC.

Run the `kdcmgr` utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.

```
kdc1# kdcmgr create master

Starting server setup
---------------------------------------

Enter the Kerberos realm: EXAMPLE.COM

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf

Initializing database '/var/krb5/principal' for realm 'EXAMPLE.COM',
master key name 'K/M@EXAMPLE.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     <Type the password>
Re-enter KDC database master key to verify:     <Type it again>

Enter the krb5 administrative principal to be created: kws/admin

Authenticating as principal root/admin@EXAMPLE.COM with password.
WARNING: no policy specified for kws/admin@EXAMPLE.COM; defaulting to no policy
Enter password for principal "kws/admin@EXAMPLE.COM":     <Type the password>
Re-enter password for principal "kws/admin@EXAMPLE.COM":     <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.

Setting up /etc/krb5/kadm5.acl.

---------------------------------------------------
Setup COMPLETE.

kdc1#
```

Example 23–1 Displaying the Status of a KDC Server

The `kdcmgr status` command can be used to display information about either a master or a slave KDC server.

▼ How to Configure a Master KDC

In this procedure, incremental propagation is configured. In addition, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- `admin` principal = `kws/admin`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the "Graphical Kerberos Administration Tool" section, as described in "Online Help URL in the Graphical Kerberos Administration Tool" on page 386.

Before You Begin This procedure requires that the host is configured to use DNS. For specific naming instructions if this master is to be swappable, see "Swapping a Master KDC and a Slave KDC" on page 436.

1. **Become superuser on the master KDC.**

2. **Edit the Kerberos configuration file (`krb5.conf`).**

   You need to change the realm names and the names of the servers. See the `krb5.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log
```

```
[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
        }
```

In this example, the lines for `default_realm`, `kdc`, `admin_server`, and all `domain_realm` entries were changed. In addition, the line that defines the `help_url` was edited.

Note – If you want to restrict the encryption types, you can set the `default_tkt_enctypes` or `default_tgs_enctypes` lines. Refer to "Using Kerberos Encryption Types" on page 541 for a description of the issues involved with restricting the encryption types.

---

3 **Edit the KDC configuration file (`kdc.conf`).**

You need to change the realm name. See the `kdc.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_master_ulogsize = 1000
        }
```

In this example, the realm name definition in the `realms` section was changed. Also, in the `realms` section, lines to enable incremental propagation and to select the number of updates the KDC master keeps in the log were added.

Note – If you want to restrict the encryption types, you can set the `permitted_enctypes`, `supported_enctypes`, or `master_key_type` lines. Refer to "Using Kerberos Encryption Types" on page 541 for a description of the issues involved with restricting the encryption types.

---

4 **Create the KDC database by using the `kdb5_util` command.**

The `kdb5_util` command creates the KDC database. Also, when used with the `-s` option, this command creates a stash file that is used to authenticate the KDC to itself before the `kadmind` and `krb5kdc` daemons are started.

```
kdc1 # /usr/sbin/kdb5_util create -s
Initializing database ’/var/krb5/principal’ for realm ’EXAMPLE.COM’
master key name ’K/M@EXAMPLE.COM’
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:     <Type the key>
Re-enter KDC database master key to verify:     <Type it again>
```

5 **Edit the Kerberos access control list file (`kadm5.acl`).**

Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC.

```
kws/admin@EXAMPLE.COM   *
```

The entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `kadm5.acl`(4) man page for more information.

6 **Start the `kadmin.local` command and add principals.**

The next substeps create principals that are used by the Kerberos service.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

a. **Add administration principals to the database.**

You can add as many `admin` principals as you need. You must add at least one `admin` principal to complete the KDC configuration process. For this example, a `kws/admin` principal is added. You can substitute an appropriate principal name instead of “`kws`.”

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM:     <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

b. **Create the `kiprop` principals.**

The `kiprop` principal is used to authorize updates from the master KDC.

```
kadmin.local: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin.local:
```

c. **Create a keytab file for the `kadmind` service.**

This command sequence creates a special keytab file with principal entries for `kadmin/<FQDN>` and `changepw/<FQDN>`. These principals are needed for the `kadmind` service and for passwords to be changed. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file. The `kadmin/changepw` principal is used to change passwords from clients that are not running a Solaris release.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

d. **Add the `kiprop` principal for the master KDC server to the `kadmind` keytab file.**

Adding the `kiprop` principal to the `kadm5.keytab` file allows the `kadmind` command to authenticate itself when incremental propagation is started.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
```

```
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

e. **Quit** `kadmin.local`**.**

You have added all of the required principals for the next steps.

```
kadmin.local: quit
```

7. **Start the Kerberos daemons.**

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

8. **Start** `kadmin` **and add more principals.**

At this point, you can add principals by using the Graphical Kerberos Administration Tool. To do so, you must log in with one of the `admin` principal names that you created earlier in this procedure. However, the following command-line example is shown for simplicity.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. **Create the master KDC** `host` **principal.**

The host principal is used by Kerberized applications, such as `kprop` to propagate changes to the slave KDCs. This principal is also used to provide secure remote access to the KDC server using applications, like `ssh`. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

b. **(Optional) Create the** `kclient` **principal.**

This principal is used by the `kclient` utility during the installation of a Kerberos client. If you do not plan on using this utility, then you do not need to add the principal. The users of the `kclient` utility need to use this password.

```
kadmin: addprinc clntconfig/admin
Enter password for principal clntconfig/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal clntconfig/admin@EXAMPLE.COM:     <Type it again>
```

```
Principal "clntconfig/admin@EXAMPLE.COM" created.
kadmin:
```

c. **Add the master KDC's `host` principal to the master KDC's keytab file.**

Adding the host principal to the keytab file allows this principal to be used by application servers, like `sshd`, automatically.

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **Quit `kadmin`.**

```
kadmin: quit
```

9 **(Optional) Synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

10 **Configure Slave KDCs.**

To provide redundancy, make sure to install at least one slave KDC. See "How to Configure a Slave KDC" on page 406 for specific instructions.

▼ How to Configure a KDC to Use an LDAP Data Server

Starting with the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, a KDC can be configured to use an LDAP data server by using the following procedure.

In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`

- Directory Server = `dsserver.example.com`
- `admin` principal = `kws/admin`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the “Graphical Kerberos Administration Tool” section, as described in “Online Help URL in the Graphical Kerberos Administration Tool” on page 386.

Before You Begin This procedure also requires that the host is configured to use DNS. For better performance, install the KDC and the LDAP Directory Service on the same server. In addition, a directory server should be running. The procedure below works with servers using the Sun Java™ Directory Server Enterprise Edition release.

1. **Become superuser on the KDC.**

2. **Create a certificate for the directory server and import the certificate.**

   The following steps configure a S10 KDC to use the Directory Server 6.1 self-signed certificate.

   a. **Export the self-signed Directory Server certificate.**

```
# /usr/sfw/bin/certutil -L -n defaultCert -d /export/sun-ds6.1/directory/alias \
        -P 'slapd-' -a > /var/tmp/ds_cert.pem
```

   b. **Create the local certificate database.**

```
# /usr/sfw/bin/certutil -N -d /var/ldap
```

   c. **Add the directory server certificate to the local certificate database.**

```
# /usr/sfw/bin/certutil -A -n defaultCert -i /var/tmp/ds_cert -a -t CT -d /var/ldap
```

   d. **Import the Directory Server certificate.**

```
# pktool setpin keystore=nss dir=/var/ldap
# chmod a+r /var/ldap/*.db
# pktool import keystore=nss objtype=cert trust="CT" infile=/tmp/defaultCert.certutil.der \
        label=defaultCert dir=/var/ldap
```

3. **Populate the LDAP directory, if necessary.**

4. **Add the Solaris Kerberos schema to the existing schema.**

```
# ldapmodify -h dsserver.example.com -D "cn=directory manager" -f /usr/share/lib/ldif/kerberos.ldif
```

5 Create the Kerberos container in the LDAP directory.

Add the following entries to the `krb5.conf` file.

a. Define the database type.

Add an entry to define the `database_module`to the `realms` section.

```
database_module = LDAP
```

b. Define the database module.

```
[dbmodules]
    LDAP = {
        ldap_kerberos_container_dn = "cn=krbcontainer,dc=example,dc=com"
        db_library = kldap
        ldap_kdc_dn = "cn=kdc service,ou=profile,dc=example,dc=com"
        ldap_kadmind_dn = "cn=kadmin service,ou=profile,dc=example,dc=com"
        ldap_cert_path = /var/ldap
        ldap_servers = ldaps://dsserver.example.com
    }
```

c. Create the KDC in the LDAP directory.

This command creates `krbcontainer` and several other objects. It also creates a `/var/krb5/.k5.EXAMPLE.COM` master key stash file.

```
# kdb5_ldap_util -D "cn=directory manager" create -P abcd1234 -r EXAMPLE.COM -s
```

6 Stash the KDC bind Distinguished Name (DN) passwords.

These passwords are used by the KDC when it binds to the DS. The KDC uses different roles depending on the type of access the KDC is using.

```
# kdb5_ldap_util stashsrvpw "cn=kdc service,ou=profile,dc=example,dc=com"
# kdb5_ldap_util stashsrvpw "cn=kadmin service,ou=profile,dc=example,dc=com"
```

7 Add KDC service roles.

a. Create a `kdc_roles.ldif` file with contents like this:

```
dn: cn=kdc service,ou=profile,dc=example,dc=com
cn: kdc service
sn: kdc service
objectclass: top
objectclass: person
userpassword: test123

dn: cn=kadmin service,ou=profile,dc=example,dc=com
cn: kadmin service
sn: kadmin service
objectclass: top
```

```
objectclass: person
userpassword: test123
```

b. **Create the role entries in the LDAP directory**

```
# ldapmodify -a -h dsserver.example.com -D "cn=directory manager" -f kdc_roles.ldif
```

8. **Set the ACLs for the KDC-related roles.**

```
# cat << EOF | ldapmodify -h dsserver.example.com -D "cn=directory manager"
# Set kadmin ACL for everything under krbcontainer.
dn: cn=krbcontainer,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=krbcontainer,dc=example,dc=com")(targetattr="krb*")(version 3.0;\
      acl kadmin_ACL; allow (all)\
      userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)

# Set kadmin ACL for everything under the people subtree if there are
# mix-in entries for krb princs:
dn: ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///ou=people,dc=example,dc=com")(targetattr="krb*")(version 3.0;\
      acl kadmin_ACL; allow (all)\
      userdn = "ldap:///cn=kadmin service,ou=profile,dc=example,dc=com";)
EOF
```

9. **Edit the Kerberos configuration file (`krb5.conf`).**

   You need to change the realm names and the names of the servers. See the `krb5.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
        kdc = kdc1.example.com
        admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
        }
```

In this example, the lines for `default_realm`, `kdc`, `admin_server`, and all `domain_realm` entries were changed. In addition, the line that defines the `help_url` was edited.

Note – If you want to restrict the encryption types, you can set the `default_tkt_enctypes` or `default_tgs_enctypes` lines. Refer to "Using Kerberos Encryption Types" on page 541 for a description of the issues involved with restricting the encryption types.

---

10 **Edit the KDC configuration file (`kdc.conf`).**

You need to change the realm name. See the `kdc.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM = {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_master_ulogsize = 1000
                }
```

In this example, the realm name definition in the `realms` section was changed. Also, in the `realms` section, lines to enable incremental propagation and to select the number of updates the KDC master keeps in the log were added.

Note – If you want to restrict the encryption types, you can set the `permitted_enctypes`, `supported_enctypes`, or `master_key_type` lines. Refer to "Using Kerberos Encryption Types" on page 541 for a description of the issues involved with restricting the encryption types.

---

11 **Edit the Kerberos access control list file (`kadm5.acl`).**

Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC.

```
kws/admin@EXAMPLE.COM   *
```

The entry gives the `kws/admin` principal in the `EXAMPLE.COM` realm the ability to modify principals or policies in the KDC. The default installation includes an asterisk (*) to match all `admin` principals. This default could be a security risk, so it is more secure to include a list of all of the `admin` principals. See the `kadm5.acl`(4) man page for more information.

12 **Start the `kadmin.local` command and add principals.**

The next substeps create principals that are used by the Kerberos service.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local:
```

a. **Add administration principals to the database.**

You can add as many `admin` principals as you need. You must add at least one `admin` principal to complete the KDC configuration process. For this example, a `kws/admin` principal is added. You can substitute an appropriate principal name instead of "`kws`."

```
kadmin.local: addprinc kws/admin
Enter password for principal kws/admin@EXAMPLE.COM:    <Type the password>
Re-enter password for principal kws/admin@EXAMPLE.COM:     <Type it again>
Principal "kws/admin@EXAMPLE.COM" created.
kadmin.local:
```

b. **Create a keytab file for the `kadmind` service.**

This command sequence creates a special keytab file with principal entries for `kadmin` and `changepw`. These principals are needed for the `kadmind` service. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc1.example.com
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
```

```
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local:
```

c. **Quit `kadmin.local`.**

You have added all of the required principals for the next steps.

```
kadmin.local: quit
```

13 Start the Kerberos daemons.

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

14 Start `kadmin` and add more principals.

At this point, you can add principals by using the Graphical Kerberos Administration Tool. To do so, you must log in with one of the `admin` principal names that you created earlier in this procedure. However, the following command-line example is shown for simplicity.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

a. **Create the master KDC `host` principal.**

The host principal is used by Kerberized applications, such as `klist` and `kprop`. Solaris 10 clients use this principal when mounting an authenticated NFS file system. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey host/kdc1.example.com
Principal "host/kdc1.example.com@EXAMPLE.COM" created.
kadmin:
```

b. **(Optional) Create the `kclient` principal.**

This principal is used by the `kclient` utility during the installation of a Kerberos client. If you do not plan on using this utility, then you do not need to add the principal. The users of the `kclient` utility need to use this password.

```
kadmin: addprinc clntconfig/admin
Enter password for principal clntconfig/admin@EXAMPLE.COM:     <Type the password>
Re-enter password for principal clntconfig/admin@EXAMPLE.COM:      <Type it again>
Principal "clntconfig/admin@EXAMPLE.COM" created.
kadmin:
```

c. **Add the master KDC's `host` principal to the master KDC's keytab file.**

Adding the host principal to the keytab file allows this principal to be used automatically.

```
kadmin: ktadd host/kdc1.example.com
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **Quit `kadmin`.**

```
kadmin: quit
```

15 **(Optional) Synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

16 **Configure Slave KDCs.**

To provide redundancy, make sure to install at least one slave KDC. See "How to Configure a Slave KDC" on page 406 for specific instructions.

▼ How to Automatically Configure a Slave KDC

Starting with the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, a slave KDC can be automatically configured by using the following procedure.

1. **Become superuser.**

2. **Create the KDC.**

   Run the `kdcmgr` utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.

```
kdc2# kdcmgr -a kws/admin -r EXAMPLE.COM create -m kdc1 slave

Starting server setup
---------------------------------------

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf
Obtaining TGT for kws/admin ...
Password for kws/admin@EXAMPLE.COM:     <Type the password>

Setting up /etc/krb5/kadm5.acl.

Setting up /etc/krb5/kpropd.acl.

Waiting for database from master...
Waiting for database from master...
Waiting for database from master...
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key
Enter KDC database master key:     <Type the password>

---------------------------------------------------
Setup COMPLETE.

kdc2#
```

▼ How to Interactively Configure a Slave KDC

Starting with the Solaris 10 5/08 and the Solaris Express Developer Edition 1/08 releases, a slave KDC can be interactively configured by using the following procedure.

1. **Become superuser.**

2. **Create the KDC.**

   Run the `kdcmgr` utility to create the KDC. You need to provide both the master key password and the password for the administrative principal.

```
kdc1# kdcmgr create slave

Starting server setup
```

```
---------------------------------------

Enter the Kerberos realm: EXAMPLE.COM
What is the master KDC's host name?: kdc1

Setting up /etc/krb5/kdc.conf

Setting up /etc/krb5/krb5.conf
Obtaining TGT for kws/admin ...
Password for kws/admin@EXAMPLE.COM:     <Type the password>

Setting up /etc/krb5/kadm5.acl.

Setting up /etc/krb5/kpropd.acl.

Waiting for database from master...
Waiting for database from master...
Waiting for database from master...
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key
Enter KDC database master key:     <Type the password>

---------------------------------------------------
Setup COMPLETE.

kdc2#
```

▼ How to Configure a Slave KDC

In this procedure, a new slave KDC named `kdc2` is configured. Also, incremental propagation is configured. This procedure uses the following configuration parameters:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com`
- `admin` principal = `kws/admin`

Before You Begin The master KDC must be configured. For specific instructions if this slave is to be swappable, see "Swapping a Master KDC and a Slave KDC" on page 436.

1 On the master KDC, become superuser.

2. **On the master KDC, start `kadmin`.**

   You must log in with one of the `admin` principal names that you created when you configured the master KDC.

   ```
   kdc1 # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin:
   ```

   a. **On the master KDC, add slave host principals to the database, if not already done.**

      For the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

      ```
      kadmin: addprinc -randkey host/kdc2.example.com
      Principal "host/kdc2.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   b. **On the master KDC, create the `kiprop` principal.**

      The `kiprop` principal is used to authorize incremental propagation from the master KDC.

      ```
      kadmin: addprinc -randkey kiprop/kdc2.example.com
      Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
      kadmin:
      ```

   c. **Quit `kadmin`.**

      ```
      kadmin: quit
      ```

3. **On the master KDC, edit the Kerberos configuration file (`krb5.conf`).**

   You need to add an entry for each slave. See the `krb5.conf`(4) man page for a full description of this file.

   ```
   kdc1 # cat /etc/krb5/krb5.conf
    .
    .
   [realms]
                   EXAMPLE.COM = {
                   kdc = kdc1.example.com
                   kdc = kdc2.example.com
                   admin_server = kdc1.example.com
           }
   ```

4. **On the master KDC, add an `kiprop` entry to `kadm5.acl`.**

   This entry allows the master KDC to receive requests for incremental propagation for the `kdc2` server.

   ```
   kdc1 # cat /etc/krb5/kadm5.acl
   */admin@EXAMPLE.COM *
   kiprop/kdc2.example.com@EXAMPLE.COM p
   ```

5 **On the master KDC, restart `kadmind` to use the new entries in the `kadm5.acl` file.**

```
kdc1 # svcadm restart network/security/kadmin
```

6 **On all slave KDCs, copy the KDC administration files from the master KDC server.**

This step needs to be followed on all slave KDCs, because the master KDC server has updated information that each KDC server needs. You can use `ftp` or a similar transfer mechanism to grab copies of the following files from the master KDC:

- `/etc/krb5/krb5.conf`
- `/etc/krb5/kdc.conf`

7 **On all slave KDCs, add an entry for the master KDC and each slave KDC into the database propagation configuration file, `kpropd.acl`.**

This information needs to be updated on all slave KDC servers.

```
kdc2 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
```

8 **On all slave KDCs, make sure that the Kerberos access control list file, `kadm5.acl`, is not populated.**

An unmodified `kadm5.acl` file would look like:

```
kdc2 # cat /etc/krb5/kadm5.acl
*/admin@___default_realm___ *
```

If the file has `kiprop` entries, remove them.

9 **On the new slave, change an entry in `kdc.conf`.**

Replace the `sunw_dbprop_master_ulogsize` entry with an entry defining `sunw_dbprop_slave_poll`. The entry sets the poll time to 2 minutes.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_slave_poll = 2m
        }
```

10 On the new slave, start the `kadmin` command.

You must log in with one of the `admin` principal names that you created when you configured the master KDC.

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin:
```

a. Add the slave's host principal to the slave's keytab file by using `kadmin`.

This entry allows `kprop` and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: ktadd host/kdc2.example.com
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc2.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

b. Add the `kiprop` principal to the slave KDC's keytab file.

Adding the `kiprop` principal to the `krb5.keytab` file allows the `kpropd` command to authenticate itself when incremental propagation is started.

```
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. Quit `kadmin`.

```
kadmin: quit
```

11 On the new slave, start the Kerberos propagation daemon.

```
kdc2 # /usr/lib/krb5/kpropd
```

12 **On the new slave, create a stash file by using `kdb5_util`.**

```
kdc2 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:     <Type the key>
```

13 **Kill the Kerberos propagation daemon.**

```
kdc2 # pkill kpropd
```

14 **(Optional) On the new slave KDC, synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

15 **On the new slave, start the KDC daemon (`krb5kdc`).**

When the `krb5kdc` service is enabled, `kpropd` also starts if the system is configured as a slave.

```
kdc2 # svcadm enable network/security/krb5kdc
```

Configuring Cross-Realm Authentication

You have several ways of linking realms together so that users in one realm can be authenticated in another realm. Cross-realm authentication is accomplished by establishing a secret key that is shared between the two realms. The relationship of the realms can be either hierarchal or directional (see "Realm Hierarchy" on page 381).

▼ How to Establish Hierarchical Cross-Realm Authentication

The example in this procedure uses two realms, `ENG.EAST.EXAMPLE.COM` and `EAST.EXAMPLE.COM`. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

Before You Begin The master KDC for each realm must be configured. To fully test the authentication process, several Kerberos clients must be configured.

1 **Become superuser on the first master KDC.**

2 **Create ticket-granting ticket service principals for the two realms.**

You must log in with one of the `admin` principal names that was created when you configured the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM
Enter password for principal krgtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM:      <Type password>
kadmin: addprinc krbtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal krgtgt/EAST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:      <Type password>
kadmin: quit
```

Note – The password that is specified for each service principal must be identical in both KDCs. Thus, the password for the service principal `krbtgt/ENG.EAST.EXAMPLE.COM@EAST.EXAMPLE.COM` must be the same in both realms.

3 **Add entries to the Kerberos configuration file (`krb5.conf`) to define domain names for every realm.**

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[domain_realm]
        .eng.east.example.com = ENG.EAST.EXAMPLE.COM
        .east.example.com = EAST.EXAMPLE.COM
```

In this example, domain names for the `ENG.EAST.EXAMPLE.COM` and `EAST.EXAMPLE.COM` realms are defined. It is important to include the subdomain first, because the file is searched top down.

4 **Copy the Kerberos configuration file to all clients in this realm.**

For cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (`/etc/krb5/krb5.conf`) installed.

5 **Repeat all of these steps in the second realm.**

▼ How to Establish Direct Cross-Realm Authentication

The example in this procedure uses two realms, `ENG.EAST.EXAMPLE.COM` and `SALES.WEST.EXAMPLE.COM`. Cross-realm authentication will be established in both directions. This procedure must be completed on the master KDC in both realms.

Before You Begin The master KDC for each realm must be configured. To fully test the authentication process, several Kerberos clients must be configured.

1 **Become superuser on one of the master KDC servers.**

2 **Create ticket-granting ticket service principals for the two realms.**

You must log in with one of the `admin` principal names that was created when you configured the master KDC.

```
# /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin: addprinc krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM
Enter password for principal
  krgtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM:       <Type the password>
kadmin: addprinc krbtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM
Enter password for principal
  krgtgt/SALES.WEST.EXAMPLE.COM@ENG.EAST.EXAMPLE.COM:       <Type the password>
kadmin: quit
```

Note – The password that is specified for each service principal must be identical in both KDCs. Thus, the password for the service principal `krbtgt/ENG.EAST.EXAMPLE.COM@SALES.WEST.EXAMPLE.COM` must be the same in both realms.

3 **Add entries in the Kerberos configuration file to define the direct path to the remote realm.**

This example shows the clients in the `ENG.EAST.EXAMPLE.COM` realm. You would need to swap the realm names to get the appropriate definitions in the `SALES.WEST.EXAMPLE.COM` realm.

```
# cat /etc/krb5/krb5.conf
[libdefaults]
 .
 .
[capaths]
    ENG.EAST.EXAMPLE.COM = {
        SALES.WEST.EXAMPLE.COM = .
    }

    SALES.WEST.EXAMPLE.COM = {
         ENG.EAST.EXAMPLE.COM = .
    }
```

4 **Copy the Kerberos configuration file to all clients in the current realm.**

For cross-realm authentication to work, all systems (including slave KDCs and other servers) must have the new version of the Kerberos configuration file (`/etc/krb5/krb5.conf`) installed.

5 **Repeat all of these steps for the second realm.**

Configuring Kerberos Network Application Servers

Network application servers are hosts that provide access using one or more of the following network applications: `ftp`, `rcp`, `rlogin`, `rsh`, `ssh`, and `telnet`. Only a few steps are required to enable the Kerberos version of these commands on a server.

▼ How to Configure a Kerberos Network Application Server

This procedure uses the following configuration parameters:

- Application server = `boston`
- `admin` principal = `kws/admin`
- DNS domain name = `example.com`
- Realm name = `EXAMPLE.COM`

Before You Begin This procedure requires that the master KDC has been configured. To fully test the process, several Kerberos clients must be configured.

1 **(Optional) Install the NTP client or another clock synchronization mechanism.**

See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

2 **Add principals for the new server and update the server's keytab.**

The following command reports the existence of the host principal:

```
boston # klist -k |grep host
4 host/boston.example.com@EXAMPLE.COM
4 host/boston.example.com@EXAMPLE.COM
4 host/boston.example.com@EXAMPLE.COM
4 host/boston.example.com@EXAMPLE.COM
```

If the command does not return a principal, then create new principals using the following steps.

How to use the Graphical Kerberos Administration Tool to add a principal is explained in "How to Create a New Kerberos Principal" on page 486. The example in the following steps shows how to add the required principals using the command line. You must log in with one of the `admin` principal names that you created when configuring the master KDC.

```
boston # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin:
```

a. **Create the server's `host` principal.**

```
kadmin: addprinc -randkey host/boston.example.com
Principal "host/boston.example.com" created.
kadmin:
```

b. **Add the server's `host` principal to the server's keytab.**

If the `kadmin` command is not running, restart it with a command similar to the following: `/usr/sbin/kadmin -p` *kws*`/admin`

```
kadmin: ktadd host/boston.example.com
Entry for principal host/boston.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/boston.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **Quit `kadmin`.**

```
kadmin: quit
```

Configuring Kerberos NFS Servers

NFS services use UNIX user IDs (UIDs) to identify a user and cannot directly use GSS credentials. To translate the credential to a UID, a credential table that maps user credentials to UNIX UIDs might need to be created. See "Mapping GSS Credentials to UNIX Credentials" on page 383 for more information on the default credential mapping. The procedures in this section focus on the tasks that are necessary to configure a Kerberos NFS server, to administer the credential table, and to initiate Kerberos security modes for NFS-mounted file systems. The following task map describes the tasks that are covered in this section.

TABLE 23–2 Configuring Kerberos NFS Servers (Task Map)

Task	Description	For Instructions
Configure a Kerberos NFS server.	Enables a server to share a file system that requires Kerberos authentication.	"How to Configure Kerberos NFS Servers" on page 415
Create a credential table.	Generates a credential table which can be used to provide mapping from GSS credentials to UNIX user IDs, if the default mapping is not sufficient.	"How to Create a Credential Table" on page 417
Change the credential table that maps user credentials to UNIX UIDs.	Updates information in the credential table.	"How to Add a Single Entry to the Credential Table" on page 417
Create credential mappings between two like realms.	Provides instructions on how to map UIDs from one realm to another if the realms share a password file.	"How to Provide Credential Mapping Between Realms" on page 418
Share a file system with Kerberos authentication.	Shares a file system with security modes so that Kerberos authentication is required.	"How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 419

▼ How to Configure Kerberos NFS Servers

In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- NFS server = `denver.example.com`
- `admin` principal = `kws/admin`

1. **Complete the prerequisites for configuring a Kerberos NFS server.**

   The master KDC must be configured. To fully test the process, you need several clients.

2. **(Optional) Install the NTP client or another clock synchronization mechanism.**

   Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be synchronized with the time on the KDC server within a maximum difference defined by the `clockskew` relation in the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

3. **Configure the NFS server as a Kerberos client.**

   Follow the instructions in "Configuring Kerberos Clients" on page 421.

4 Start `kadmin`.

You can use the Graphical Kerberos Administration Tool to add a principal, as explained in "How to Create a New Kerberos Principal" on page 486. To do so, you must log in with one of the `admin` principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin:
```

a. Create the server's NFS service principal.

Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

Repeat this step for each unique interface on the system that might be used to access NFS data. If a host has multiple interfaces with unique names, each unique name must have its own NFS service principal.

```
kadmin: addprinc -randkey nfs/denver.example.com
Principal "nfs/denver.example.com" created.
kadmin:
```

b. Add the server's NFS service principal to the server's keytab file.

Repeat this step for each unique service principal created in Step a.

```
kadmin: ktadd nfs/denver.example.com
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. Quit `kadmin`.

```
kadmin: quit
```

5 (Optional) Create special GSS credential maps, if needed.

Normally, the Kerberos service generates appropriate maps between the GSS credentials and the UNIX UIDs. The default mapping is described in "Mapping GSS Credentials to UNIX Credentials" on page 383. If the default mapping is not sufficient, see "How to Create a Credential Table" on page 417 for more information.

6 **Share the NFS file system with Kerberos security modes.**

See "How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes" on page 419 for more information.

▼ How to Create a Credential Table

The `gsscred` credential table is used by an NFS server to map Kerberos credentials to a UID. By default, the primary part of the principal name is matched to a UNIX login name. For NFS clients to mount file systems from an NFS server with Kerberos authentication, this table must be created if the default mapping is not sufficient.

1 **Edit `/etc/gss/gsscred.conf` and change the security mechanism.**

Change the mechanism to `files`.

2 **Create the credential table by using the `gsscred` command.**

```
# gsscred -m kerberos_v5 -a
```

The `gsscred` command gathers information from all sources that are listed with the `passwd` entry in the `/etc/nsswitch.conf` file. You might need to temporarily remove the `files` entry, if you do not want the local password entries included in the credential table. See the `gsscred`(1M) man page for more information.

▼ How to Add a Single Entry to the Credential Table

Before You Begin This procedure requires that the `gsscred` table has already been created on the NFS server. See "How to Create a Credential Table" on page 417 for instructions.

1 **Become superuser on the NFS server.**

2 **Add an entry to the credential table by using the `gsscred` command.**

```
# gsscred -m mech [ -n name [ -u uid ]] -a
```

mech	Defines the security mechanism to be used.
name	Defines the principal name for the user, as defined in the KDC.
uid	Defines the UID for the user, as defined in the password database.
`-a`	Adds the UID to principal name mapping.

Example 23–2 Adding a Multiple Component Principal to the Credential Table

In the following example, an entry is added for a principal named `sandy/admin`, which is mapped to UID `3736`.

```
# gsscred -m kerberos_v5 -n sandy/admin -u 3736 -a
```

Example 23–3 Adding a Principal in a Different Domain to the Credential Table

In the following example, an entry is added for a principal named `sandy/admin@EXAMPLE.COM`, which is mapped to UID `3736`.

```
# gsscred -m kerberos_v5 -n sandy/admin@EXAMPLE.COM -u 3736 -a
```

▼ How to Provide Credential Mapping Between Realms

This procedure provides appropriate credential mapping between realms that use the same password file. In this example, the realms `CORP.EXAMPLE.COM` and `SALES.EXAMPLE.COM` use the same password file. The credentials for `bob@CORP.EXAMPLE.COM` and `bob@SALES.EXAMPLE.COM` are mapped to the same UID.

1. **Become superuser.**

2. **On the client system, add entries to the `krb5.conf` file.**

```
# cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = CORP.EXAMPLE.COM
 .
[realms]
    CORP.EXAMPLE.COM = {
        .
        auth_to_local_realm = SALES.EXAMPLE.COM
        .
    }
```

Example 23–4 Mapping Credentials Between Realms Using the Same Password File

This example provides appropriate credential mapping between realms that use the same password file. In this example, the realms `CORP.EXAMPLE.COM` and `SALES.EXAMPLE.COM` use the same password file. The credentials for `bob@CORP.EXAMPLE.COM` and `bob@SALES.EXAMPLE.COM` are mapped to the same UID. On the client system, add entries to the `krb5.conf` file.

```
# cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = CORP.EXAMPLE.COM
 .
[realms]
    CORP.EXAMPLE.COM = {
        .
        auth_to_local_realm = SALES.EXAMPLE.COM
        .
    }
```

Troubleshooting See "Observing Mapping from GSS Credentials to UNIX Credentials" on page 473 to help with the process of troubleshooting credential mapping problems.

▼ How to Set Up a Secure NFS Environment With Multiple Kerberos Security Modes

This procedure enables a NFS server to provide secure NFS access using different security modes or flavors. When a client negotiates a security flavor with the NFS server, the first flavor that is offered by the server that the client has access to is used. This flavor is used for all subsequent client requests of the file system shared by the NFS server.

1. **Become superuser on the NFS server.**

2. **Verify that there is an NFS service principal in the keytab file.**

   The `klist` command reports if there is a keytab file and displays the principals. If the results show that no keytab file exists or that no NFS service principal exists, you need to verify the completion of all the steps in "How to Configure Kerberos NFS Servers" on page 415.

```
# klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab
KVNO Principal
---- ---------------------------------------------------------
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM
   3 nfs/denver.example.com@EXAMPLE.COM
```

3. **Enable Kerberos security modes in the `/etc/nfssec.conf` file.**

   Edit the `/etc/nfssec.conf` file and remove the "#" that is placed in front of the Kerberos security modes.

```
# cat /etc/nfssec.conf
 .
 .
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5           390003  kerberos_v5     default -              # RPCSEC_GSS
krb5i          390004  kerberos_v5     default integrity      # RPCSEC_GSS
krb5p          390005  kerberos_v5     default privacy        # RPCSEC_GSS
```

4. **Edit the `/etc/dfs/dfstab` file and add the `sec=` option with the required security modes to the appropriate entries.**

   `share -F nfs -o sec=`*mode file-system*

mode	Specifies the security modes to be used when sharing the file system. When using multiple security modes, the first mode in the list is used as the default.
file-system	Defines the path to the file system to be shared.

   All clients that attempt to access files from the named file system require Kerberos authentication. To access files, the user principal on the NFS client should be authenticated.

5. **Make sure that the NFS service is running on the server.**

   If this command is the first `share` command or set of `share` commands that you have initiated, the NFS daemons are likely not running. The following command restarts the daemons:

   ```
   # svcadm restart network/nfs/server
   ```

6. **(Optional) If the automounter is being used, edit the `auto_master` database to select a security mode other than the default.**

   You need not follow this procedure if you are not using the automounter to access the file system or if the default selection for the security mode is acceptable.

   file-system `auto_home -nosuid,sec=`*mode*

7. **(Optional) Manually issue the `mount` command to access the file system by using a non-default mode.**

   Alternatively, you could use the `mount` command to specify the security mode, but this alternative does not take advantage of the automounter.

   `#` **`mount -F nfs -o sec=`***mode file-system*

Example 23–5 Sharing a File System With One Kerberos Security Mode

In this example, the `dfstab` file line means that Kerberos authentication must succeed before any files can be accessed through the NFS service.

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5 /export/home
```

Example 23–6 Sharing a File System With Multiple Kerberos Security Modes

In this example, all three Kerberos security modes have been selected. Which mode is used is negotiated between the client and the NFS server. If the first mode in the command fails, then the next is tried. See the `nfssec`(5) man page for more information.

```
# grep krb /etc/dfs/dfstab
share -F nfs -o sec=krb5:krb5i:krb5p /export/home
```

Configuring Kerberos Clients

Kerberos clients include any host, that is not a KDC server, on the network that needs to use Kerberos services. This section provides procedures for installing a Kerberos client, as well as specific information about using `root` authentication to mount NFS file systems.

Configuring Kerberos Clients (Task Map)

The following task map includes all of the procedures associated with setting up Kerberos clients. Each row includes a task identifier, a description of why you would want to do that task, followed by a link to the task.

Task	Description	For Instructions
Establish a Kerberos client installation profile.	Generates a client installation profile that can be used to automatically install a Kerberos client.	"How to Create a Kerberos Client Installation Profile" on page 422
Configure a Kerberos client.	Manually installs a Kerberos client. Use this procedure if each client installation requires unique installation parameters.	"How to Manually Configure a Kerberos Client" on page 424
	Automatically installs a Kerberos client. Use this procedure if the installation parameters for each client are the same.	"How to Automatically Configure a Kerberos Client" on page 422
	Interactively installs a Kerberos client. Use this procedure if only a few of the installation parameters need to change.	"How to Interactively Configure a Kerberos Client" on page 423
Allow a client to access a NFS file system as the `root` user	Creates a `root` principal on the client, so that the client can mount a NFS file system shared with `root` access. Also, allows for the client to set up non-interactive `root` access to the NFS file system, so that cron jobs can run.	"How to Access a Kerberos Protected NFS File System as the `root` User" on page 431

▼ How to Create a Kerberos Client Installation Profile

This procedure creates a kclient profile that can be used when you install a Kerberos client. By using the kclient profile, you reduce the likelihood of typing errors. Also, using the profile reduces user intervention as compared to the interactive process.

1 **Become superuser.**

2 **Create a kclient installation profile.**

A sample kclient profile could look similar to the following:

```
client# cat /net/denver.example.com/export/install/profile
REALM EXAMPLE.COM
KDC kdc1.example.com
ADMIN clntconfig
FILEPATH /net/denver.example.com/export/install/krb5.conf
NFS 1
DNSLOOKUP none
```

▼ How to Automatically Configure a Kerberos Client

Before You Begin This procedure uses an installation profile. See "How to Create a Kerberos Client Installation Profile" on page 422.

1 **Become superuser.**

2 **Run the `kclient` installation script.**

You need to provide the password for the `clntconfig` principal to complete the process.

```
client# /usr/sbin/kclient -p /net/denver.example.com/export/install/profile

Starting client setup
---------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:     <Type the password>

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.
```

```
Copied /net/denver.example.com/export/install/krb5.conf.

---------------------------------------------------
Setup COMPLETE.

client#
```

Example 23–7 Automatically Configuring a Kerberos Client With Command-Line Overrides

The following example overrides the `DNSARG` and the `KDC` parameters that are set in the installation profile.

```
# /usr/sbin/kclient -p /net/denver.example.com/export/install/profile\
-d dns_fallback -k kdc2.example.com

Starting client setup
---------------------------------------------------

kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:      <Type the password>

nfs/client.example.com entry ADDED to KDC database.
nfs/client.example.com entry ADDED to keytab.

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Copied /net/denver.example.com/export/install/krb5.conf.

---------------------------------------------------
Setup COMPLETE.

client#
```

▼ How to Interactively Configure a Kerberos Client

This procedure uses the `kclient` installation utility without a installation profile.

1. **Become superuser.**

2. **Run the `kclient` installation script.**

You need to provide the following information:

- Kerberos realm name
- KDC master host name
- Administrative principal name
- Password for the administrative principal

Example 23–8 Running the `kclient` Installation Utility

The following output shows the results of running the `kclient` command.

```
client# /usr/sbin/kclient

Starting client setup
---------------------------------------------------

Do you want to use DNS for kerberos lookups ? [y/n]: n
        No action performed.
Enter the Kerberos realm: EXAMPLE.COM
Specify the KDC hostname for the above realm: kdc1.example.com

Setting up /etc/krb5/krb5.conf.

Enter the krb5 administrative principal to be used: clntconfig/admin
Obtaining TGT for clntconfig/admin ...
Password for clntconfig/admin@EXAMPLE.COM:      <Type the password>
Do you plan on doing Kerberized nfs ? [y/n]: n

host/client.example.com entry ADDED to KDC database.
host/client.example.com entry ADDED to keytab.

Do you want to copy over the master krb5.conf file ? [y/n]: y
Enter the pathname of the file to be copied: \
/net/denver.example.com/export/install/krb5.conf

Copied /net/denver.example.com/export/install/krb5.conf.

---------------------------------------------------
Setup COMPLETE !
#
```

▼ How to Manually Configure a Kerberos Client

In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`

- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com`
- NFS server = `denver.example.com`
- Client = `client.example.com`
- `admin` principal = `kws/admin`
- User principal = `mre`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the "Graphical Kerberos Administration Tool" section, as described in the "Online Help URL in the Graphical Kerberos Administration Tool" on page 386.

---

1 **Become superuser.**

2 **Edit the Kerberos configuration file** (`krb5.conf`)**.**

To change the file from the Kerberos default version, you need to change the realm names and the server names. You also need to identify the path to the help files for `gkadmin`.

```
kdc1 # cat /etc/krb5/krb5.conf
[libdefaults]
        default_realm = EXAMPLE.COM

[realms]
        EXAMPLE.COM = {
        kdc = kdc1.example.com
        kdc = kdc2.example.com
        admin_server = kdc1.example.com
        }

[domain_realm]
        .example.com = EXAMPLE.COM
#
# if the domain name and realm name are equivalent,
# this entry is not needed
#
[logging]
        default = FILE:/var/krb5/kdc.log
        kdc = FILE:/var/krb5/kdc.log

[appdefaults]
    gkadmin = {
        help_url = http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956
```

Note – If you want to restrict the encryption types, you can set the `default_tkt_enctypes` or `default_tgs_enctypes` lines. Refer to "Using Kerberos Encryption Types" on page 541 for a description of the issues involved with restricting the encryption types.

---

3 **(Optional) Change the process used to locate the KDCs.**

Starting with the Solaris 10 5/08 and Solaris Express Developer Edition 1/08 releases, by default the Kerberos realm to KDC mapping is determined in the following order:

- The definition in the `realms` section in `krb5.conf`.
- By looking up SRV records in DNS.

You can change this behavior by adding `dns_lookup_kdc` or `dns_fallback` to the `libdefaults` section of the `krb5.conf` file. See the `krb5.conf`(4) man page for more information. Note that referrals are always tried first.

4 **(Optional) Change the process used to determine the realm for a host.**

Starting with the Solaris 10 5/08 and Solaris Express Developer Edition 1/08 releases, by default the host to realm mapping is determined in the following order:

- If the KDC supports referrals, then the KDC may inform the client which realm the host belongs to.
- By the definition of `domain_realm` in the `krb5.conf` file.
- The DNS domainname of the host.
- The default realm.

You can change this behavior by adding `dns_lookup_kdc` or `dns_fallback` to the `libdefaults` section of the `krb5.conf`file. See the `krb5.conf`(4) man page for more information. Note that referrals will always be tried first.

5 **(Optional) Synchronize the client's clock with the master KDC's clock by using NTP or another clock synchronization mechanism.**

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be synchronized with the time on the KDC server within a maximum difference defined in the `clockskew` relation in the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

6 **Start `kadmin`.**

You can use the Graphical Kerberos Administration Tool to add a principal, as explained in "How to Create a New Kerberos Principal" on page 486. To do so, you must log in with one of the `admin` principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

```
denver # /usr/sbin/kadmin -p kws/admin
Enter password:     <Type kws/admin password>
kadmin:
```

a. **(Optional) Create a user principal if a user principal does not already exist.**

   You need to create a user principal only if the user associated with this host does not already have a principal assigned to him or her.

   ```
   kadmin: addprinc mre
   Enter password for principal mre@EXAMPLE.COM:     <Type the password>
   Re-enter password for principal mre@EXAMPLE.COM:     <Type it again>
   kadmin:
   ```

b. **(Optional) Create a `root` principal and add the principal to the server's keytab file.**

   This step is required so that the client can have `root` access to file systems mounted using the NFS service. This step is also required if non-interactive `root` access is needed, such as running cron jobs as `root`.

   If the client does not require `root` access to a remote file system which is mounted using the NFS service, then you can skip this step. The `root` principal should be a two component principal with the second component the host name of the Kerberos client system to avoid the creation of a realm wide `root` principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey root/client.example.com
Principal "root/client.example.com" created.
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **Create a `host` principal and add the principal to the server's keytab file.**

   The `host` principal is used by remote access services to provide authentication. The principal allows `root` to acquire a credential, if there is not one already in the keytab file.

```
kadmin: addprinc -randkey host/denver.example.com
Principal "host/denver.example.com@EXAMPLE.COM" created.
```

```
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

d. **(Optional) Add the server's NFS service principal to the server's keytab file.**

   This step is only required if the client needs to access NFS file systems using Kerberos authentication.

```
kadmin: ktadd nfs/denver.example.com
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal nfs/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

e. **Add the `host` principal to the server's keytab file.**

```
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

f. **Quit `kadmin`.**

```
kadmin: quit
```

7 **(Optional) Enable Kerberos with NFS.**

a. **Enable Kerberos security modes in the `/etc/nfssec.conf` file.**

Edit the `/etc/nfssec.conf` file and remove the "#" that is placed in front of the Kerberos security modes.

```
# cat /etc/nfssec.conf
 .
 .
#
# Uncomment the following lines to use Kerberos V5 with NFS
#
krb5            390003  kerberos_v5     default -               # RPCSEC_GSS
krb5i           390004  kerberos_v5     default integrity       # RPCSEC_GSS
krb5p           390005  kerberos_v5     default privacy         # RPCSEC_GSS
```

b. **Enable DNS.**

If the `/etc/resolv.conf` file has not already been created, then create this file as the service principal canonicalization is dependent upon DNS to do this. See the `resolv.conf`(4) man page for more information.

c. **Restart the `gssd` service.**

After the `/etc/resolv.conf` file has been created or modified you must then restart the `gssd` daemon to reread any changes.

```
# svcadm restart network/rpc/gss
```

8 **If you want the client to automatically renew the TGT or to warn users about Kerberos ticket expiration, create an entry in the `/etc/krb5/warn.conf` file.**

See the `warn.conf`(4) man page for more information.

Example 23–9 Setting Up a Kerberos Client Using a Non-Solaris KDC

A Kerberos client can be set up to work with a non-Solaris KDC. In this case, a line must be included in the `/etc/krb5/krb5.conf` file in the `realms` section. This line changes the protocol that is used when the client is communicating with the Kerberos password-changing server. The format of this line follows.

```
[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                admin_server = kdc1.example.com
                kpasswd_protocol = SET_CHANGE
        }
```

Example 23–10 DNS TXT Records for the Mapping of Host and Domain Name to Kerberos Realm

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
                                1989020501   ;serial
                                10800        ;refresh
                                3600         ;retry
                                3600000      ;expire
                                86400 )      ;minimum

                        IN      NS      kdc1.example.com.
kdc1                    IN      A       192.146.86.20
kdc2                    IN      A       192.146.86.21

_kerberos.example.com.             IN      TXT     "EXAMPLE.COM"
_kerberos.kdc1.example.com.        IN      TXT     "EXAMPLE.COM"
_kerberos.kdc2.example.com.        IN      TXT     "EXAMPLE.COM"
```

Example 23–11 DNS SRV Records for Kerberos Server Locations

This example defines the records for the location of the KDCs, the `admin` server, and the `kpasswd` server, respectively.

```
@ IN SOA kdc1.example.com root.kdc1.example.com (
                                1989020501   ;serial
                                10800        ;refresh
                                3600         ;retry
                                3600000      ;expire
                                86400 )      ;minimum

                                   IN      NS      kdc1.example.com.
kdc1                               IN      A       192.146.86.20
kdc2                               IN      A       192.146.86.21

_kerberos._udp.EXAMPLE.COM         IN      SRV 0 0 88  kdc2.example.com
_kerberos._tcp.EXAMPLE.COM         IN      SRV 0 0 88  kdc2.example.com
_kerberos._udp.EXAMPLE.COM         IN      SRV 1 0 88  kdc1.example.com
_kerberos._tcp.EXAMPLE.COM         IN      SRV 1 0 88  kdc1.example.com
_kerberos-adm._udp.EXAMPLE.COM     IN      SRV 0 0 749 kdc1.example.com
_kerberos-adm._tcp.EXAMPLE.COM     IN      SRV 0 0 749 kdc1.example.com
_kpasswd._udp.EXAMPLE.COM          IN      SRV 0 0 749 kdc1.example.com
```

▼ How to Access a Kerberos Protected NFS File System as the `root` User

This procedure allows a client to access a NFS file system that requires Kerberos authentication with the `root` ID privilege. In particular, if the NFS file system is shared with options like: `-o sec=krb5,root=client1.sun.com`.

1. **Become superuser.**

2. **Start `kadmin`.**

   You can use the Graphical Kerberos Administration Tool to add a principal, as explained in "How to Create a New Kerberos Principal" on page 486. To do so, you must log in with one of the `admin` principal names that you created when you configured the master KDC. However, the following example shows how to add the required principals by using the command line.

   ```
   denver # /usr/sbin/kadmin -p kws/admin
   Enter password:      <Type kws/admin password>
   kadmin:
   ```

   a. **Create a `root` principal for the NFS client.**

      This principal is used to provide root equivalent access to NFS mounted file systems that require Kerberos authentication. The `root` principal should be a two component principal with the second component the host name of the Kerberos client system to avoid the creation of a realm wide root principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

      ```
      kadmin: addprinc -randkey root/client.example.com
      Principal "root/client.example.com" created.
      kadmin:
      ```

   b. **Add the `root` principal to the server's keytab file.**

      This step is required if you added a `root` principal so that the client can have `root` access to file systems mounted using the NFS service. This step is also required if non-interactive `root` access is needed, such as running cron jobs as `root`.

```
kadmin: ktadd root/client.example.com
Entry for principal root/client.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal root/client.example.com with kvno 3, encryption type DES cbc mode
```

```
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

c. **Quit `kadmin`.**

```
kadmin: quit
```

How to Configure Automatic Migration of Users in a Kerberos Realm

Users, who do not have a Kerberos principal, can be automatically migrated to an existing Kerberos realm. The migration is achieved by using the PAM framework for the service in use by stacking the `pam_krb5_migrate` module in the service's authentication stack in `/etc/pam.conf`.

In this example, the `dtlogin` and `other` PAM service names are configured to use the automatic migration. The following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- Master KDC = `kdc1.example.com`
- Machine hosting the migration service = `server1.example.com`
- Migration service principal = `host/server1.example.com`

Before You Begin Setup `server1` as a Kerberos client of the realm `EXAMPLE.COM`. See "Configuring Kerberos Clients" on page 421 for more information.

1. **Check to see if a host service principal for `server1` exists.**

   The host service principal in the `keytab` file of `server1` is used to authenticate the server to the master KDC.

```
server1 # klist -k
Keytab name: FILE:/etc/krb5/krb5.keytab
    KVNO Principal
    ---- ------------------------------------------------
       3 host/server1.example.com@EXAMPLE.COM
       3 host/server1.example.com@EXAMPLE.COM
       3 host/server1.example.com@EXAMPLE.COM
       3 host/server1.example.com@EXAMPLE.COM
```

2. **Make changes to the PAM configuration file.**

   a. **Add entries for the `dtlogin` service.**

```
# cat /etc/pam.conf
 .
 .
```

```
#
# dtlogin service (explicit because of pam_krb5_migrate)
#
dtlogin       auth requisite          pam_authtok_get.so.1
dtlogin       auth required           pam_dhkeys.so.1
dtlogin       auth required           pam_unix_cred.so.1
dtlogin       auth sufficient         pam_krb5.so.1
dtlogin       auth requisite          pam_unix_auth.so.1
dtlogin       auth optional           pam_krb5_migrate.so.1
```

b. **(Optional) Force an immediate password change, if needed.**

The newly created Kerberos accounts can have their password expiration time set to the current time (now), in order to force an immediate Kerberos password change. To set the expiration time to now, add the `expire_pw` option to the lines which use the `pam_krb5_migrate` module. See the `pam_krb5_migrate`(5) man page for more information.

```
# cat /etc/pam.conf
 .
 .
dtlogin  auth optional           pam_krb5_migrate.so.1 expire_pw
```

c. **Add the `pam_krb5` module to the account stack.**

This addition allows for password expiration in Kerberos to block access.

```
# cat /etc/pam.conf
 .
 .
#
# Default definition for Account management
# Used when service name is not explicitly mentioned for account management
#
other   account requisite       pam_roles.so.1
other   account required        pam_krb5.so.1
other   account required        pam_unix_account.so.1
```

d. **Add the `pam_krb5` module to the password stack.**

This addition allows for passwords to be updated when the password expire.

```
# cat /etc/pam.conf
 .
 .
#
# Default definition for Password management
# Used when service name is not explicitly mentioned for password management
#
other   password required       pam_dhkeys.so.1
other   password requisite      pam_authtok_get.so.1
other   password requisite      pam_authtok_check.so.1
other   password sufficient     pam_krb5.so.1
other   password required       pam_authtok_store.so.1
```

3 **On the master KDC, update the access control file.**

The following entries grant migrate and inquire privileges to the `host/server1.example.com` service principal for all users, excepting the `root` user. It is important that users who should not be migrated are listed in the kadm5.acl file using the `U` privilege. These entries need to be before the permit all or `ui` entry. See the `kadm5.acl`(4) man page for more information.

```
kdc1 # cat /etc/krb5/kadm5.acl
host/server1.example.com@EXAMPLE.COM U root
host/server1.example.com@EXAMPLE.COM ui *
*/admin@EXAMPLE.COM *
```

4 **On the master KDC, restart the Kerberos administration daemon.**

This step allows the `kadmind` daemon to use the new `kadm5.acl` entries.

```
kdc1 # svcadm restart network/security/kadmin
```

5 **On the master KDC, add entries to the `pam.conf` file.**

The following entries enable the `kadmind` daemon to use the `k5migrate` PAM service, to validate UNIX user password for accounts that require migration.

```
# grep k5migrate /etc/pam.conf
k5migrate        auth    required        pam_unix_auth.so.1
k5migrate        account required        pam_unix_account.so.1
```

Synchronizing Clocks Between KDCs and Kerberos Clients

All hosts that participate in the Kerberos authentication system must have their internal clocks synchronized within a specified maximum amount of time (known as *clock skew*). This requirement provides another Kerberos security check. If the clock skew is exceeded between any of the participating hosts, client requests are rejected.

The clock skew also determines how long application servers must keep track of all Kerberos protocol messages, in order to recognize and reject replayed requests. So, the longer the clock skew value, the more information that application servers have to collect.

The default value for the maximum clock skew is 300 seconds (five minutes). You can change this default in the `libdefaults` section of the `krb5.conf` file.

Note – For security reasons, do not increase the clock skew beyond 300 seconds.

Because maintaining synchronized clocks between the KDCs and Kerberos clients is important, you should use the Network Time Protocol (NTP) software to synchronize them. NTP public domain software from the University of Delaware is included in the Solaris software, starting with the Solaris 2.6 release.

Note – Another way to synchronize clocks is to use the `rdate` command and `cron` jobs, a process that can be less involved than using NTP. However, this section focuses on using NTP. And, if you use the network to synchronize the clocks, the clock synchronization protocol must itself be secure.

NTP enables you to manage precise time or network clock synchronization, or both, in a network environment. NTP is basically a server-client implementation. You pick one system to be the master clock (the NTP server). Then, you set up all your other systems (the NTP clients) to synchronize their clocks with the master clock.

To synchronize the clocks, NTP uses the `xntpd` daemon, which sets and maintains a UNIX system time-of-day in agreement with Internet standard time servers. The following shows an example of this server-client NTP implementation.

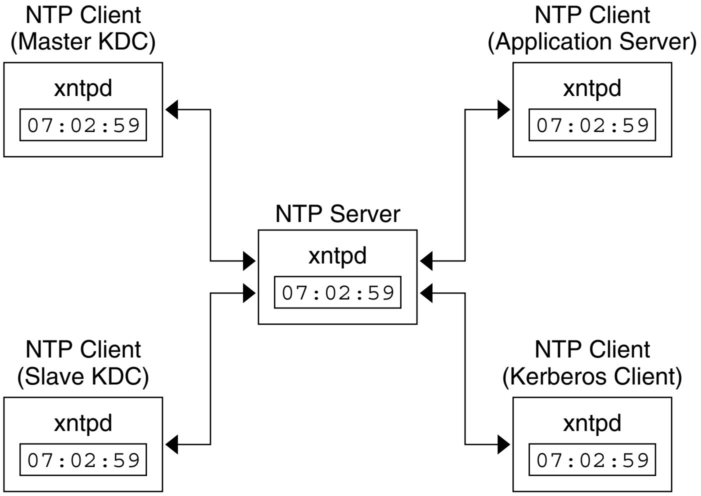

NTP Client
(Master KDC)
xntpd
07:02:59
NTP Client
(Application Server)
xntpd
07:02:59
NTP Server
xntpd
07:02:59
NTP Client
(Slave KDC)
xntpd
07:02:59
NTP Client
(Kerberos Client)
xntpd
07:02:59

FIGURE 23–1 Synchronizing Clocks by Using NTP

Ensuring that the KDCs and Kerberos clients maintain synchronized clocks involves implementing the following steps:

1. Setting up an NTP server on your network. This server can be any system, except the master KDC. See "Managing Network Time Protocol (Tasks)" in *System Administration Guide: Network Services* to find the NTP server task.
2. As you configure the KDCs and Kerberos clients on the network, setting them up to be NTP clients of the NTP server. See "Managing Network Time Protocol (Tasks)" in *System Administration Guide: Network Services* to find the NTP client task.

Swapping a Master KDC and a Slave KDC

You should use the procedures in this section to make the swap of a master KDC with a slave KDC easier. You should swap the master KDC with a slave KDC only if the master KDC server fails for some reason, or if the master KDC needs to be re-installed (for example, because new hardware is installed).

▼ How to Configure a Swappable Slave KDC

Perform this procedure on the slave KDC server that you want to have available to become the master KDC. This procedure assumes that you are using incremental propagation.

1 Use alias names for the master KDC and the swappable slave KDC during the KDC installation.

When you define the host names for the KDCs, make sure that each system has an alias included in DNS. Also, use the alias names when you define the hosts in the `/etc/krb5/krb5.conf` file.

2 Follow the steps to install a slave KDC.

Prior to any swap, this server should function as any other slave KDC in the realm. See "How to Configure a Slave KDC" on page 406 for instructions.

3 Move the master KDC commands.

To prevent the master KDC commands from being run from this slave KDC, move the `kprop`, `kadmind`, and `kadmin.local` commands to a reserved place.

```
kdc4 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc4 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc4 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
```

▼ How to Swap a Master KDC and a Slave KDC

In this procedure, the master KDC server that is being swapped out is named `kdc1`. The slave KDC that will become the new master KDC is named `kdc4`. This procedure assumes that you are using incremental propagation.

Before You Begin This procedure requires that the slave KDC server has been set up as a swappable slave. For more information, see "How to Configure a Swappable Slave KDC" on page 436).

1. **On the new master KDC, start `kadmin`.**

```
kdc4 # /usr/sbin/kadmin -p kws/admin
Enter password:      <Type kws/admin password>
kadmin:
```

   a. **Create new principals for the `kadmind` service.**

   The following example shows the first `addprinc` command on two lines, but it should be typed on one line.

```
kadmin: addprinc -randkey -allow_tgs_req +password_changing_service -clearpolicy \
       changepw/kdc4.example.com
Principal "changepw/kdc4.example.com@ENG.SUN.COM" created.
kadmin: addprinc -randkey -allow_tgs_req -clearpolicy kadmin/kdc4.example.com
Principal "kadmin/kdc4.example.com@EXAMPLE.COM" created.
kadmin:
```

   b. **Create a keytab file.**

```
kadmin: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc4.example.com
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc4.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin: ktadd -k /etc/krb5/kadm5.keytab changepw/kdc4.example.com
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
```

```
Entry for principal changepw/kdc4.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin: ktadd -k /etc/krb5/kadm5.keytab kadmin/changepw
Entry for principal kadmin/changepw with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/changepw with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin:
```

c. **Quit `kadmin`.**

```
kadmin: quit
```

2. **On the new master KDC, force synchronization.**

   The following steps force a full KDC update on the slave server.

   ```
   kdc4 # svcadm disable network/security/krb5kdc
   kdc4 # rm /var/krb5/principal.ulog
   ```

3. **On the new master KDC, verify that the update is complete.**

   ```
   kdc4 # /usr/sbin/kproplog -h
   ```

4. **On the new master KDC, restart the KDC service.**

   ```
   kdc4 # svcadm enable -r network/security/krb5kdc
   ```

5. **On the new master KDC, clear the update log.**

   These steps reinitialize the update log for the new master KDC server.

   ```
   kdc4 # svcadm disable network/security/krb5kdc
   kdc4 # rm /var/krb5/principal.ulog
   ```

6. **On the old master KDC, kill the `kadmind` and `krb5kdc` processes.**

   When you kill the `kadmind` process, you prevent any changes from being made to the KDC database.

   ```
   kdc1 # svcadm disable network/security/kadmin
   kdc1 # svcadm disable network/security/krb5kdc
   ```

7 **On the old master KDC, specify the poll time for requesting propagations.**

Comment out the `sunw_dbprop_master_ulogsize` entry in `/etc/krb5/kdc.conf` and add an entry defining `sunw_dbprop_slave_poll`. The entry sets the poll time to 2 minutes.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
#               sunw_dbprop_master_ulogsize = 1000
                sunw_dbprop_slave_poll = 2m
        }
```

8 **On the old master KDC, move the master KDC commands and the `kadm5.acl` file.**

To prevent the master KDC commands from being run, move the `kprop`, `kadmind`, and `kadmin.local` commands to a reserved place.

```
kdc1 # mv /usr/lib/krb5/kprop /usr/lib/krb5/kprop.save
kdc1 # mv /usr/lib/krb5/kadmind /usr/lib/krb5/kadmind.save
kdc1 # mv /usr/sbin/kadmin.local /usr/sbin/kadmin.local.save
kdc1 # mv /etc/krb5/kadm5.acl /etc/krb5/kadm5.acl.save
```

9 **On the DNS server, change the alias names for the master KDC.**

To change the servers, edit the `example.com` zone file and change the entry for `masterkdc`.

```
masterkdc IN CNAME kdc4
```

10 **On the DNS server, restart the Internet domain name server.**

Run the following command to reload the new alias information:

```
# svcadm refresh network/dns/server
```

11 **On the new master KDC, move the master KDC commands and the slave `kpropd.acl` file.**

```
kdc4 # mv /usr/lib/krb5/kprop.save /usr/lib/krb5/kprop
kdc4 # mv /usr/lib/krb5/kadmind.save /usr/lib/krb5/kadmind
kdc4 # mv /usr/sbin/kadmin.local.save /usr/sbin/kadmin.local
kdc4 # mv /etc/krb5/kpropd.acl /etc/krb5/kpropd.acl.save
```

12 **On the new master KDC, create the Kerberos access control list file (`kadm5.acl`).**

Once populated, the `/etc/krb5/kadm5.acl` file should contain all principal names that are allowed to administer the KDC. The file should also list all of the slaves that make requests for incremental propagation. See the `kadm5.acl`(4) man page for more information.

```
kdc4 # cat /etc/krb5/kadm5.acl
kws/admin@EXAMPLE.COM   *
kiprop/kdc1.example.com@EXAMPLE.COM p
```

13 **On the new master KDC, specify the update log size in the `kdc.conf` file.**

Comment out the `sunw_dbprop_slave_poll` entry and add an entry defining `sunw_dbprop_master_ulogsize`. The entry sets the log size to 1000 entries.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
#               sunw_dbprop_slave_poll = 2m
                sunw_dbprop_master_ulogsize = 1000
        }
```

14 **On the new master KDC, add the `kiprop` principal to the `kadmind` keytab file.**

```
kdc4 # kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc4.example.com
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc4.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

15 **On the new master KDC, start `kadmind` and `krb5kdc`.**

```
kdc4 # svcadm enable -r network/security/krb5kdc
kdc4 # svcadm enable -r network/security/kadmin
```

16 **On the old master KDC, add the `kiprop` service principal.**

Adding the `kiprop` principal to the `krb5.keytab` file allows the `kpropd` daemon to authenticate itself for the incremental propagation service.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Authenticating as pricipal kws/admin@EXAMPLE.COM with password.
Enter password:      <Type kws/admin password>
kadmin: ktadd kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

17 **On the old master KDC, add an entry for each KDC listed in `krb5.conf` to the propagation configuration file, `kpropd.acl`.**

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
host/kdc4.example.com@EXAMPLE.COM
```

18 **On the old master KDC, start `kpropd` and `krb5kdc`.**

When the `krb5kdc` daemon is started, `kpropd` also starts if the system is configured as a slave.

```
kdc1 # svcadm enable network/security/krb5kdc
```

Administering the Kerberos Database

The Kerberos database is the backbone of Kerberos and must be maintained properly. This section provides some procedures on how to administer the Kerberos database, such as backing up and restoring the database, setting up incremental or parallel propagation, and administering the stash file. The steps to initially set up the database are in "How to Configure a Master KDC" on page 392.

Backing Up and Propagating the Kerberos Database

Propagating the Kerberos database from the master KDC to the slave KDCs is one of the most important configuration tasks. If propagation doesn't happen often enough, the master KDC and the slave KDCs will lose synchronization. So, if the master KDC goes down, the slave KDCs will not have the most recent database information. Also, if a slave KDC has been configured as a master KDC for purposes of load balancing, the clients that use that slave KDC as a master KDC will not have the latest information. Therefore, you must make sure that propagation occurs often enough or else configure the servers for incremental propagation, based on how often you change the Kerberos database. Incremental propagation is preferred over manual propagation because there is more administrative overhead when you manually propagate the database. Also, there are inefficiencies when you do full propagation of the database.

When you configure the master KDC, you set up the `kprop_script` command in a `cron` job to automatically back up the Kerberos database to the `/var/krb5/slave_datatrans` dump file and propagate it to the slave KDCs. But, as with any file, the Kerberos database can become corrupted. If data corruption occurs on a slave KDC, you might never notice, because the next automatic propagation of the database installs a fresh copy. However, if corruption occurs on the master KDC, the corrupted database is propagated to all of the slave KDCs during the next propagation. And, the corrupted backup overwrites the previous uncorrupted backup file on the master KDC.

Because there is no “safe” backup copy in this scenario, you should also set up a `cron` job to periodically copy the `slave_datatrans` dump file to another location or to create another separate backup copy by using the `dump` command of `kdb5_util`. Then, if your database becomes corrupted, you can restore the most recent backup on the master KDC by using the `load` command of `kdb5_util`.

Another important note: Because the database dump file contains principal keys, you need to protect the file from being accessed by unauthorized users. By default, the database dump file has read and write permissions only as `root`. To protect against unauthorized access, use only the `kprop` command to propagate the database dump file, which encrypts the data that is being transferred. Also, `kprop` propagates the data only to the slave KDCs, which minimizes the chance of accidentally sending the database dump file to unauthorized hosts.

Caution – If the Kerberos database is updated after it has been propagated and if the database subsequently is corrupted before the next propagation, the KDC slaves will not contain the updates. The updates will be lost. For this reason, if you add significant updates to the Kerberos database before a regularly scheduled propagation, you should manually propagate the database to avoid data loss.

The `kpropd.acl` File

The `kpropd.acl` file on a slave KDC provides a list of host principal names, one name per line, that specifies the systems from which the KDC can receive an updated database through propagation. If the master KDC is used to propagate all the slave KDCs, the `kpropd.acl` file on each slave needs to contain only the host principal name of the master KDC.

However, the Kerberos installation and subsequent configuration steps in this book instruct you to add the same `kpropd.acl` file to the master KDC and the slave KDCs. This file contains all the KDC host principal names. This configuration enables you to propagate from any KDC, in case the propagating KDCs become temporarily unavailable. And, by keeping an identical copy on all KDCs, you make the configuration easy to maintain.

The `kprop_script` Command

The `kprop_script` command uses the `kprop` command to propagate the Kerberos database to other KDCs. If the `kprop_script` command is run on a slave KDC, it propagates the slave KDC's copy of the Kerberos database to other KDCs. The `kprop_script` accepts a list of host names for arguments, separated by spaces, which denote the KDCs to propagate.

When `kprop_script` is run, it creates a backup of the Kerberos database to the `/var/krb5/slave_datatrans` file and copies the file to the specified KDCs. The Kerberos database is locked until the propagation is finished.

▼ How to Back Up the Kerberos Database

1. **Become superuser on the master KDC.**
2. **Back up the Kerberos database by using the `dump` command of the `kdb5_util` command.**

```
# /usr/sbin/kdb5_util dump [-verbose] [-d dbname] [filename [principals...]]
```

`-verbose`	Prints the name of each principal and policy that is being backed up.
dbname	Defines the name of the database to back up. Note that you can specify an absolute path for the file. If the `-d` option is not specified, the default database name is `/var/krb5/principal`.
filename	Defines the file that is used to back up the database. You can specify an absolute path for the file. If you don't specify a file, the database is dumped to standard output.
principals	Defines a list of one or more principals (separated by a space) to back up. You must use fully qualified principal names. If you don't specify any principals, the entire database is backed up.

Example 23–12 Backing Up the Kerberos Database

In the following example, the Kerberos database is backed up to a file called `dumpfile`. Because the `-verbose` option is specified, each principal is printed as it is backed up.

```
# kdb5_util dump -verbose dumpfile
kadmin/kdc1.eng.example.com@ENG.EXAMPLE.COM
krbtgt/eng.example.com@ENG.EXAMPLE.COM
kadmin/history@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
changepw/kdc1.eng.example.com@ENG.EXAMPLE.COM
```

In the following example, the `pak` and `pak/admin` principals from the Kerberos database are backed up.

```
# kdb5_util dump -verbose dumpfile pak/admin@ENG.EXAMPLE.COM pak@ENG.EXAMPLE.COM
pak/admin@ENG.EXAMPLE.COM
pak@ENG.EXAMPLE.COM
```

How to Restore the Kerberos Database

1. **Become superuser on the master KDC.**

2. **On the master, stop the KDC daemons.**

```
kdc1 # svcadm disable network/security/krb5kdc
kdc1 # svcadm disable network/security/kadmin
```

3. **Restore the Kerberos database by using the `load` command of the `kdb_util` command.**

```
# /usr/sbin/kdb5_util load [-verbose] [-d dbname] [-update] [filename]
```

`-verbose`	Prints the name of each principal and policy that is being restored.
dbname	Defines the name of the database to restore. Note you can specify an absolute path for the file. If the `-d` option is not specified, the default database name is `/var/krb5/principal`.
`-update`	Updates the existing database. Otherwise, a new database is created or the existing database is overwritten.
filename	Defines the file from which to restore the database. You can specify an absolute path for the file.

4. **Start the KDC daemons.**

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

Example 23–13 Restoring the Kerberos Database

In the following example, the database called `database1` is restored into the current directory from the `dumpfile` file. Because the `-update` option isn't specified, a new database is created by the restore.

```
# kdb5_util load -d database1 dumpfile
```

▼ How to Convert a Kerberos Database After a Server Upgrade

If your KDC database was created on a server running the Solaris 8 or Solaris 9 release,]converting the database allows you to take advantage of the improved database format.

Before You Begin Make sure that the database is using an older format. See XXX for specific instructions.

1. **On the master, stop the KDC daemons.**

```
kdc1 # svcadm disable network/security/krb5kdc
kdc1 # svcadm disable network/security/kadmin
```

2. **Create a directory to store a temporary copy of the database.**

```
kdc1 # mkdir /var/krb5/tmp
kdc1 # chmod 700 /var/krb5/tmp
```

3. **Dump the KDC database.**

```
kdc1 # kdb5_util dump /var/krb5/tmp/prdb.txt
```

4. **Save copies of the current database files.**

```
kdc1 # cd /var/krb5
kdc1 # mv princ* tmp/
```

5. **Load the database.**

```
kdc1 # kdb5_util load /var/krb5/tmp/prdb.txt
```

6. **Start the KDC daemons.**

```
kdc1 # svcadm enable -r network/security/krb5kdc
kdc1 # svcadm enable -r network/security/kadmin
```

▼ How to Reconfigure a Master KDC to Use Incremental Propagation

The steps in this procedure can be used to reconfigure an existing master KDC to use incremental propagation. In this procedure, the following configuration parameters are used:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com`
- `admin` principal = `kws/admin`

1 **Add entries to `kdc.conf`.**

You need to enable incremental propagation and select the number of updates the KDC master keeps in the log. See the `kdc.conf`(4) man page for more information.

```
kdc1 # cat /etc/krb5/kdc.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_master_ulogsize = 1000
        }
```

2 **Create the `kiprop` principal.**

The `kiprop` principal is used to authenticate the master KDC server and to authorize updates from the master KDC.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin: addprinc -randkey kiprop/kdc1.example.com
Principal "kiprop/kdc1.example.com@EXAMPLE.COM" created.
kadmin: addprinc -randkey kiprop/kdc2.example.com
Principal "kiprop/kdc2.example.com@EXAMPLE.COM" created.
kadmin:
```

3 Add the `kiprop` principal to the `kadmind` keytab file

Adding the `kiprop` principal to the `kadm5.keytab` file allows the `kadmind` command to authenticate itself when it is started.

```
kadmin: ktadd -k /etc/krb5/kadm5.keytab kiprop/kdc1.example.com
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kiprop/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin: quit
```

4 On the master KDC, add a kiprop entry to `kadm5.acl`

This entry allows the master KDC to receive requests for incremental propagation from the `kdc2` server.

```
kdc1 # cat /etc/krb5/kadm5.acl
*/admin@EXAMPLE.COM *
kiprop/kdc2.example.com@EXAMPLE.COM p
```

5 Comment out the `kprop` line in the `root` crontab file.

This step prevents the master KDC from propagating its copy of the KDC database.

```
kdc1 # crontab -e
#ident  "@(#)root       1.20    01/11/06 SMI"
#
# The root crontab should be used to perform accounting data collection.
#
# The rtc command is run to adjust the real time clock if and when
# daylight savings time changes.
#
10 3 * * * /usr/sbin/logadm
15 3 * * 0 /usr/lib/fs/nfs/nfsfind
1 2 * * * [ -x /usr/sbin/rtc ] && /usr/sbin/rtc -c > /dev/null 2>&1
30 3 * * * [ -x /usr/lib/gss/gsscred_clean ] && /usr/lib/gss/gsscred_clean
#10 3 * * * /usr/lib/krb5kprop_script kdc2.example.sun.com #SUNWkr5ma
```

6 Restart `kadmind`.

```
kdc1 # svcadm restart network/security/kadmin
```

7 **Reconfigure all slave KDC servers that use incremental propagation.**

See "How to Reconfigure a Slave KDC to Use Incremental Propagation" on page 448 for complete instructions.

How to Reconfigure a Slave KDC to Use Incremental Propagation

1 **Add entries to `krb5.conf`.**

The new entries enable incremental propagation and set the poll time to 2 minutes.

```
kdc2 # cat /etc/krb5/krb5.conf
[kdcdefaults]
        kdc_ports = 88,750

[realms]
        EXAMPLE.COM= {
                profile = /etc/krb5/krb5.conf
                database_name = /var/krb5/principal
                admin_keytab = /etc/krb5/kadm5.keytab
                acl_file = /etc/krb5/kadm5.acl
                kadmind_port = 749
                max_life = 8h 0m 0s
                max_renewable_life = 7d 0h 0m 0s
                sunw_dbprop_enable = true
                sunw_dbprop_slave_poll = 2m
        }
```

2 **Add the `kiprop` principal to the `krb5.keytab` file.**

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin: ktadd kiprop/kdc2.example.com
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal kiprop/kdc2.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

3 **Disable `kpropd`.**

```
kdc2 # svcadm disable network/security/krb5_prop
```

4 **Restart the KDC server.**

```
kdc2 # svcadm restart network/security/krb5kdc
```

▼ How to Configure a Slave KDC to Use Full Propagation

This procedure shows how to reconfigure a slave KDC server running the Solaris 10 release to use full propagation. Normally, the procedure would only need to be used if the master KDC server is running either the Solaris 9 release or an earlier release. In this case, the master KDC server can not support incremental propagation, so the slave needs to be configured to allow propagation to work.

In this procedure, a slave KDC named `kdc3` is configured. This procedure uses the following configuration parameters:

- Realm name = `EXAMPLE.COM`
- DNS domain name = `example.com`
- Master KDC = `kdc1.example.com`
- Slave KDC = `kdc2.example.com` and `kdc3.example.com`
- `admin` principal = `kws/admin`
- Online help URL = `http://denver:8888/ab2/coll.384.1/SEAM/@AB2PageView/6956`

Note – Adjust the URL to point to the "Graphical Kerberos Administration Tool" section, as described in the "Online Help URL in the Graphical Kerberos Administration Tool" on page 386.

Before You Begin The master KDC must be configured. For specific instructions if this slave is to be swappable, see "Swapping a Master KDC and a Slave KDC" on page 436.

1 **On the master KDC, become superuser.**

2 **On the master KDC, start `kadmin`.**

You must log in with one of the `admin` principal names that you created when you configured the master KDC.

```
kdc1 # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin:
```

a. **On the master KDC, add slave host principals to the database, if not already done.**

For the slave to function, it must have a host principal. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: addprinc -randkey host/kdc3.example.com
Principal "host/kdc3@EXAMPLE.COM" created.
kadmin:
```

b. **Quit `kadmin`.**

```
kadmin: quit
```

3 **On the master KDC, edit the Kerberos configuration file (`krb5.conf`).**

You need to add an entry for each slave. See the `krb5.conf`(4) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/krb5.conf
  .
  .
[realms]
                EXAMPLE.COM = {
                kdc = kdc1.example.com
                kdc = kdc2.example.com
                kdc = kdc3.example.com
                admin_server = kdc1.example.com
        }
```

4 **On the master KDC, add an entry for the master KDC and each slave KDC into the `kpropd.acl` file.**

See the `kprop`(1M) man page for a full description of this file.

```
kdc1 # cat /etc/krb5/kpropd.acl
host/kdc1.example.com@EXAMPLE.COM
host/kdc2.example.com@EXAMPLE.COM
host/kdc3.example.com@EXAMPLE.COM
```

5 **On all slave KDCs, copy the KDC administration files from the master KDC server.**

This step needs to be followed on all slave KDCs, because the master KDC server has updated information that each KDC server needs. You can use `ftp` or a similar transfer mechanism to grab copies of the following files from the master KDC:

- `/etc/krb5/krb5.conf`
- `/etc/krb5/kdc.conf`
- `/etc/krb5/kpropd.acl`

6 **On all slave KDCs, make sure that the Kerberos access control list file, `kadm5.acl`, is not populated.**

An unmodified `kadm5.acl` file would look like:

```
kdc2 # cat /etc/krb5/kadm5.acl
*/admin@___default_realm___ *
```

If the file has `kiprop` entries, remove them.

7 **On the new slave, start the `kadmin` command.**

You must log in with one of the `admin` principal names that you created when you configured the master KDC.

```
kdc2 # /usr/sbin/kadmin -p kws/admin
Enter password:       <Type kws/admin password>
kadmin:
```

a. **Add the slave's `host` principal to the slave's keytab file by using `kadmin`.**

This entry allows `kprop` and other Kerberized applications to function. Note that when the principal instance is a host name, the FQDN must be specified in lowercase letters, regardless of the case of the domain name in the `/etc/resolv.conf` file.

```
kadmin: ktadd host/kdc3.example.com
Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/kdc3.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin:
```

b. **Quit `kadmin`.**

```
kadmin: quit
```

8 **On the master KDC, add the slave KDC name to the `cron` job, which automatically runs the backups, by running `crontab -e`.**

Add the name of each slave KDC server at the end of the `kprop_script` line.

```
10 3 * * * /usr/lib/krb5/kprop_script kdc2.example.com kdc3.example.com
```

You might also want to change the time of the backups. This entry starts the backup process every day at 3:10 AM.

9 **On the new slave, start the Kerberos propagation daemon.**

```
kdc3 # svcadm enable network/security/krb5_prop
```

10 **On the master KDC, back up and propagate the database by using `kprop_script`.**

If a backup copy of the database is already available, it is not necessary to complete another backup. See "How to Manually Propagate the Kerberos Database to the Slave KDCs" on page 454 for further instructions.

```
kdc1 # /usr/lib/krb5/kprop_script kdc3.example.com
Database propagation to kdc3.example.com: SUCCEEDED
```

11 **On the new slave, create a stash file by using `kdb5_util`.**

```
kdc3 # /usr/sbin/kdb5_util stash
kdb5_util: Cannot find/read stored master key while reading master key
kdb5_util: Warning: proceeding without master key

Enter KDC database master key:      <Type the key>
```

12 **(Optional) On the new slave KDC, synchronize the master KDCs clock by using NTP or another clock synchronization mechanism.**

Installing and using the Network Time Protocol (NTP) is not required. However, every clock must be within the default time that is defined in the `libdefaults` section of the `krb5.conf` file for authentication to succeed. See "Synchronizing Clocks Between KDCs and Kerberos Clients" on page 434 for information about NTP.

13 **On the new slave, start the KDC daemon (`krb5kdc`).**

```
kdc3 # svcadm enable network/security/krb5kdc
```

▼ How to Verify That the KDC Servers Are Synchronized

If incremental propagation has been configured, this procedure ensures that the information on the slave KDC has been updated.

1 **On the KDC master server, run the `kproplog` command.**

```
kdc1 # /usr/sbin/kproplog -h
```

2 **On a KDC slave server, run the `kproplog` command.**

```
kdc2 # /usr/sbin/kproplog -h
```

3 **Check that the last serial # and the last timestamp values match.**

Example 23–14 Verifying That the KDC Servers Are Synchronized

The following is a sample of results from running the `kproplog` command on the master KDC server.

```
kdc1 # /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
    Log version #: 1
    Log state: Stable
    Entry block size: 2048
    Number of entries: 2500
    First serial #: 137966
    Last serial #: 140465
    First time stamp: Fri Nov 28 00:59:27 2004
    Last time stamp: Fri Nov 28 01:06:13 2004
```

The following is a sample of results from running the `kproplog` command on a slave KDC server.

```
kdc2 # /usr/sbin/kproplog -h

Kerberos update log (/var/krb5/principal.ulog)
Update log dump:
    Log version #: 1
    Log state: Stable
    Entry block size: 2048
    Number of entries: 0
    First serial #: None
    Last serial #: 140465
    First time stamp: None
    Last time stamp: Fri Nov 28 01:06:13 2004
```

Notice that the values for the last serial number and the last timestamp are identical, which indicates that the slave is synchronized with the master KDC server.

In the slave KDC server output, notice that no update entries exist in the slave KDC server's update log. No entries exist because the slave KDC server does not keep a set of updates, unlike the master KDC server. Also, the KDC slave server does not include information on the first serial number or the first timestamp because this is not relevant information.

▼ How to Manually Propagate the Kerberos Database to the Slave KDCs

This procedure shows you how to propagate the Kerberos database by using the `kprop` command. Use this procedure if you need to synchronize a slave KDC with the master KDC outside the periodic `cron` job. Unlike the `kprop_script`, you can use `kprop` to propagate just the current database backup without first making a new backup of the Kerberos database.

Note – Do not use this procedure if you are using incremental propagation.

---

1 **Become superuser on the master KDC.**

2 **(Optional) Back up the database by using the `kdb5_util` command.**

```
# /usr/sbin/kdb5_util dump /var/krb5/slave_datatrans
```

3 **Propagate the database to a slave KDC by using the `kprop` command.**

```
# /usr/lib/krb5/kprop -f /var/krb5/slave_datatrans slave-KDC
```

Example 23–15 Manually Propagating the Kerberos Database to the Slave KDCs Using `kprop_script`

If you want to back up the database and propagate it to a slave KDC outside the periodic `cron` job, you can also use the `kprop_script` command as follows:

```
# /usr/lib/krb5/kprop_script slave-KDC
```

Setting Up Parallel Propagation

In most cases, the master KDC is used exclusively to propagate its Kerberos database to the slave KDCs. However, if your site has many slave KDCs, you might consider load-sharing the propagation process, known as *parallel propagation*.

Note – Do not use this procedure if you are using incremental propagation.

Parallel propagation allows specific slave KDCs to share the propagation duties with the master KDC. This sharing of duties enables the propagation to be done faster and to lighten the work for the master KDC.

For example, say your site has one master KDC and six slave KDCs (shown in Figure 23–2), where `slave-1` through `slave-3` consist of one logical grouping and `slave-4` through `slave-6` consist of another logical grouping. To set up parallel propagation, you could have the master KDC propagate the database to `slave-1` and `slave-4`. In turn, those KDC slaves could propagate the database to the KDC slaves in their group.

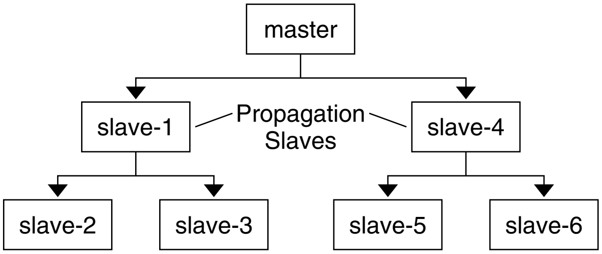


FIGURE 23–2 Example of Parallel Propagation Configuration

Configuration Steps for Setting Up Parallel Propagation

The following is not a detailed step-by-step procedure, but a high-level list of configuration steps to enable parallel propagation. These steps involve the following:

1. On the master KDC, changing the `kprop_script` entry in its `cron` job to include arguments for only the KDC slaves that will perform the succeeding propagation (the *propagation slaves*).
2. On each propagation slave, adding a `kprop_script` entry to its `cron` job, which must include arguments for the slaves to propagate. To successfully propagate in parallel, the `cron` job should be set up to run after the propagation slave is itself propagated with the new Kerberos database.

   Note – How long it will take for a propagation slave to be propagated depends on factors such as network bandwidth and the size of the Kerberos database.

   ---

3. On each slave KDC, setting up the appropriate permissions to be propagated. This step is done by adding the host principal name of its propagating KDC to its `kpropd.acl` file.

EXAMPLE 23–16 Setting Up Parallel Propagation

Using the example in Figure 23–2, the master KDC's `kprop_script` entry would look similar to the following:

```
0 3 * * * /usr/lib/krb5/kprop_script slave-1.example.com slave-4.example.com
```

EXAMPLE 23–16 Setting Up Parallel Propagation *(Continued)*

The `slave-1`'s `kprop_script` entry would look similar to the following:

```
0 4 * * * /usr/lib/krb5/kprop_script slave-2.example.com slave-3.example.com
```

Note that the propagation on the slave starts an hour after it is propagated by the master.

The `kpropd.acl` file on the propagation slaves would contain the following entry:

```
host/master.example.com@EXAMPLE.COM
```

The `kpropd.acl` file on the KDC slaves being propagated by `slave-1` would contain the following entry:

```
host/slave-1.example.com@EXAMPLE.COM
```

Administering the Stash File

The *stash file* contains the master key for the Kerberos database, which is automatically created when you create a Kerberos database. If the stash file gets corrupted, you can use the `stash` command of the `kdb5_util` utility to replace the corrupted file. The only time you should need to remove a stash file is after removing the Kerberos database with the `destroy` command of `kdb5_util`. Because the stash file is not automatically removed with the database, you have to remove the stash file to finish the cleanup.

▼ How to Remove a Stash File

1. **Become superuser on the KDC that contains the stash file.**

2. **Remove the stash file.**

   ```
   # rm stash-file
   ```

   Where *stash-file* is the path to the stash file. By default, the stash file is located at `/var/krb5/.k5.`*realm*.

Note – If you need to re-create the stash file, you can use the `-f` option of the `kdb5_util` command.

Increasing Security on Kerberos Servers

Follow these steps to increase security on Kerberos application servers and on KDC servers.

▼ How to Enable Only Kerberized Applications

This procedure restricts network access to the server that is running `telnet`, `ftp`, `rcp`, `rsh`, and `rlogin` to use Kerberos authenticated transactions only.

1 **Change the `exec` property for the `telnet` service.**

Add the `-a user` option to the `exec` property for `telnet` to restrict access to those users who can provide valid authentication information.

```
# inetadm -m svc:/network/telnet:default exec="/usr/sbin/in.telnetd -a user"
```

2 **(Optional) If not already configured, change the `exec` property for the `telnet` service.**

Add the `-a` option to the `exec` property for `ftp` to permit only Kerberos authenticated connections.

```
# inetadm -m svc:/network/ftp:default exec="/usr/sbin/in.ftpd -a"
```

3 **Disable other services.**

The `in.rshd` and `in.rlogind` daemons should be disabled.

```
# svcadm disable network/shell
# svcadm disable network/login:rlogin
```

▼ How to Restrict Access to KDC Servers

Both master KDC servers and slave KDC servers have copies of the KDC database stored locally. Restricting access to these servers so that the databases are secure is important to the overall security of the Kerberos installation.

1 **Disable remote services, as needed.**

To provide a secure KDC server, all nonessential network services should be disabled . Depending on your configuration, some of these services may already be disabled. Check the service status with the `svcs` command. In most circumstances, the only services that would need to run would be `krb5kdc` and `kadmin` if the KDC is a master. In addition, any services that use loopback tli (`ticlts`, `ticotsord`, and `ticots`) can be left enabled.

```
# svcadm disable network/comsat
# svcadm disable network/dtspc/tcp
# svcadm disable network/finger
# svcadm disable network/login:rlogin
# svcadm disable network/rexec
# svcadm disable network/shell
# svcadm disable network/talk
# svcadm disable network/tname
# svcadm disable network/uucp
# svcadm disable network/rpc_100068_2-5/rpc_udp
```

2 **Restrict access to the hardware that supports the KDC.**

To restrict physical access, make sure that the KDC server and its monitor are located in a secure facility. Users should not be able to access this server in any way.

3 **Store KDC database backups on local disks or on the KDC slaves.**

Make tape backups of your KDC only if the tapes are stored securely. Follow the same practice for copies of keytab files. It would be best to store these files on a local file system that is not shared with other systems. The storage file system can be on either the master KDC server or any of the slave KDCs.

◆ ◆ ◆ CHAPTER 24

Kerberos Error Messages and Troubleshooting

This chapter provides resolutions for error messages that you might receive when you use the Kerberos service. This chapter also provides some troubleshooting tips for various problems. This is a list of the error message and troubleshooting information in this chapter.

- "SEAM Administration Tool Error Messages" on page 459
- "Common Kerberos Error Messages (A-M)" on page 460
- "Common Kerberos Error Messages (N-Z)" on page 468
- "Problems With the Format of the `krb5.conf` File" on page 471
- "Problems Propagating the Kerberos Database" on page 471
- "Problems Mounting a Kerberized NFS File System" on page 472
- "Problems Authenticating as `root`" on page 472
- "Observing Mapping from GSS Credentials to UNIX Credentials" on page 473

Kerberos Error Messages

This section provides information about Kerberos error messages, including why each error occurs and a way to fix it.

SEAM Administration Tool Error Messages

`Unable to view the list of principals or policies; use the Name field.`

Cause: The `admin` principal that you logged in with does not have the list privilege (`l`) in the Kerberos ACL file (`kadm5.acl`). So, you cannot view the principal list or policy list.

Solution: You must type the principal and policy names in the Name field to work on them, or you need to log in with a principal that has the appropriate privileges.

`JNI: Java array creation failed`

`JNI: Java class lookup failed`

`JNI: Java field lookup failed`

`JNI: Java method lookup failed`

`JNI: Java object lookup failed`

`JNI: Java object field lookup failed`

`JNI: Java string access failed`

`JNI: Java string creation failed`

Cause: A serious problem exists with the Java Native Interface that is used by the SEAM Administration Tool (`gkadmin`).

Solution: Exit `gkadmin` and restart it. If the problem persists, please report a bug.

Common Kerberos Error Messages (A-M)

This section provides an alphabetical list (A-M) of common error messages for the Kerberos commands, Kerberos daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

`All authentication systems disabled; connection refused`

Cause: This version of `rlogind` does not support any authentication mechanism.

Solution: Make sure that `rlogind` is invoked with the `-k` option.

`Another authentication mechanism must be used to access this host`

Cause: Authentication could not be done.

Solution: Make sure that the client is using Kerberos V5 mechanism for authentication.

`Authentication negotiation has failed, which is required for encryption. Good bye.`

Cause: Authentication could not be negotiated with the server.

Solution: Start authentication debugging by invoking the `telnet` command with the `toggle authdebug` command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

`Bad krb5 admin server hostname while initializing kadmin interface`

Cause: An invalid host name is configured for `admin_server` in the `krb5.conf` file.

Solution: Make sure that the correct host name for the master KDC is specified on the `admin_server` line in the `krb5.conf` file.

`Bad lifetime value`

Cause: The lifetime value provided is not valid or incorrectly formatted.

Solution: Make sure that the value provided is consistent with the Time Formats section in the `kinit`(1) man page.

`Bad start time value`

Cause: The start time value provided is not valid or incorrectly formatted.

Solution: Make sure that the value provided is consistent with the Time Formats section in the `kinit`(1) man page.

`Cannot contact any KDC for requested realm`

Cause: No KDC responded in the requested realm.

Solution: Make sure that at least one KDC (either the master or a slave) is reachable or that the `krb5kdc` daemon is running on the KDCs. Check the `/etc/krb5/krb5.conf` file for the list of configured KDCs (`kdc` = *kdc-name*).

`Cannot determine realm for host`

Cause: Kerberos cannot determine the realm name for the host.

Solution: Make sure that there is a default realm name, or that the domain name mappings are set up in the Kerberos configuration file (`krb5.conf`).

`Cannot find KDC for requested realm`

Cause: No KDC was found in the requested realm.

Solution: Make sure that the Kerberos configuration file (`krb5.conf`) specifies a KDC in the `realm` section.

`cannot initialize realm` *realm-name*

Cause: The KDC might not have a stash file.

Solution: Make sure that the KDC has a stash file. If not, create a stash file by using the `kdb5_util` command, and try restarting the `krb5kdc` command.

`Cannot resolve KDC for requested realm`

Cause: Kerberos cannot determine any KDC for the realm.

Solution: Make sure that the Kerberos configuration file (`krb5.conf`) specifies a KDC in the `realm` section.

`Cannot reuse password`

Cause: The password that you specified has been used before by this principal.

Solution: Choose a password that has not been chosen before, at least not within the number of passwords that are kept in the KDC database for each principal. This policy is enforced by the principal's policy.

`Can’t get forwarded credentials`

Cause: Credential forwarding could not be established.

Solution: Make sure that the principal has forwardable credentials.

`Can’t open/find Kerberos configuration file`

Cause: The Kerberos configuration file (`krb5.conf`) was unavailable.

Solution: Make sure that the `krb5.conf` file is available in the correct location and has the correct permissions. This file should be writable by `root` and readable by everyone else.

`Client did not supply required checksum--connection rejected`

Cause: Authentication with checksum was not negotiated with the client. The client might be using an old Kerberos V5 protocol that does not support initial connection support.

Solution: Make sure that the client is using a Kerberos V5 protocol that supports initial connection support.

`Client/server realm mismatch in initial ticket request`

Cause: A realm mismatch between the client and server occurred in the initial ticket request.

Solution: Make sure that the server you are communicating with is in the same realm as the client, or that the realm configurations are correct.

`Client or server has a null key`

Cause: The principal has a null key.

Solution: Modify the principal to have a non-null key by using the `cpw` command of `kadmin`.

`Communication failure with server while initializing kadmin interface`

Cause: The host that was specified for the admin server, also called the master KDC, did not have the `kadmind` daemon running.

Solution: Make sure that you specified the correct host name for the master KDC. If you specified the correct host name, make sure that `kadmind` is running on the master KDC that you specified.

`Credentials cache file permissions incorrect`

Cause: You do not have the appropriate read or write permissions on the credentials cache (`/tmp/krb5cc_`*uid*).

Solution: Make sure that you have read and write permissions on the credentials cache.

`Credentials cache I/O operation failed` *XXX*

Cause: Kerberos had a problem writing to the system's credentials cache (`/tmp/krb5cc_`*uid*).

Solution: Make sure that the credentials cache has not been removed, and that there is space left on the device by using the `df` command.

`Decrypt integrity check failed`

Cause: You might have an invalid ticket.

Solution: Verify both of these conditions:

- Make sure that your credentials are valid. Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.
- Make sure that the target host has a keytab file with the correct version of the service key. Use `kadmin` to view the key version number of the service principal (for example, `host/`*FQDN-hostname*) in the Kerberos database. Also, use `klist -k` on the target host to make sure that it has the same key version number.

`Encryption could not be enabled. Goodbye.`

Cause: Encryption could not be negotiated with the server.

Solution: Start authentication debugging by invoking the `telnet` command with the `toggle encdebug` command and look at the debug messages for further clues.

`failed to obtain credentials cache`

Cause: During `kadmin` initialization, a failure occurred when `kadmin` tried to obtain credentials for the `admin` principal.

Solution: Make sure that you used the correct principal and password when you executed `kadmin`.

`Field is too long for this implementation`

Cause: The message size that was being sent by a Kerberized application was too long. This error could be generated if the transport protocol is UDP. which has a default maximum message size 65535 bytes. In addition, there are limits on individual fields within a protocol message that is sent by the Kerberos service.

Solution: Verify that you have not restricted the transport to UDP in the KDC server's `/etc/krb5/kdc.conf` file.

`GSS-API (or Kerberos) error`

Cause: This message is a generic GSS-API or Kerberos error message and can be caused by several different problems.

Solution: Check the `/var/krb5/kdc.log` file to find the more specific error message that was logged when this error occurred.

`Hostname cannot be canonicalized`

Cause: Kerberos cannot make the host name fully qualified.

Solution: Make sure that the host name is defined in DNS and that the host-name-to-address and address-to-host-name mappings are consistent.

`Illegal cross-realm ticket`

Cause: The ticket sent did not have the correct cross-realms. The realms might not have the correct trust relationships set up.

Solution: Make sure that the realms you are using have the correct trust relationships.

`Improper format of Kerberos configuration file`

Cause: The Kerberos configuration file has invalid entries.

Solution: Make sure that all the relations in the `krb5.conf` file are followed by the "=" sign and a value. Also, verify that the brackets are present in pairs for each subsection.

`Inappropriate type of checksum in message`

Cause: The message contained an invalid checksum type.

Solution: Check which valid checksum types are specified in the `krb5.conf` and `kdc.conf` files.

`Incorrect net address`

Cause: There was a mismatch in the network address. The network address in the ticket that was being forwarded was different from the network address where the ticket was processed. This message might occur when tickets are being forwarded.

Solution: Make sure that the network addresses are correct. Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

`Invalid credential was supplied`

`Service key not available`

Cause: The service ticket in the credentials cache may be incorrect.

Solution: Destroy current credential cache and rerun `kinit` before trying to use this service.

`Invalid flag for file lock mode`

Cause: An internal Kerberos error occurred.

Solution: Please report a bug.

`Invalid message type specified for encoding`

Cause: Kerberos could not recognize the message type that was sent by the Kerberized application.

Solution: If you are using a Kerberized application that was developed by your site or a vendor, make sure that it is using Kerberos correctly.

`Invalid number of character classes`

Cause: The password that you specified for the principal does not contain enough password classes, as enforced by the principal's policy.

Solution: Make sure that you specify a password with the minimum number of password classes that the policy requires.

`KADM err: Memory allocation failure`

Cause: There is insufficient memory to run `kadmin`.

Solution: Free up memory and try running `kadmin` again.

`KDC can’t fulfill requested option`

Cause: The KDC did not allow the requested option. A possible problem might be that postdating or forwardable options were being requested, and the KDC did not allow them. Another problem might be that you requested the renewal of a TGT, but you didn't have a renewable TGT.

Solution: Determine if you are either requesting an option that the KDC does not allow or a type of ticket that is not available.

`KDC policy rejects request`

Cause: The KDC policy did not allow the request. For example, the request to the KDC did not have an IP address in its request. Or forwarding was requested, but the KDC did not allow it.

Solution: Make sure that you are using `kinit` with the correct options. If necessary, modify the policy that is associated with the principal or change the principal's attributes to allow the request. You can modify the policy or principal by using `kadmin`.

`KDC reply did not match expectations`

Cause: The KDC reply did not contain the expected principal name, or other values in the response were incorrect.

Solution: Make sure that the KDC you are communicating with complies with RFC1510, that the request you are sending is a Kerberos V5 request, or that the KDC is available.

`kdestroy: Could not obtain principal name from cache`

Cause: The credentials cache is missing or corrupted.

Solution: Check that the cache location provided is correct. Remove and obtain a new TGT using `kinit`, if necessary.

`kdestroy: No credentials cache file found while destroying cache`

Cause: The credentials cache (`/tmp/krb5c_`*uid*) is missing or corrupted.

Solution: Check that the cache location provided is correct. Remove and obtain a new TGT using `kinit`, if necessary.

`kdestroy: TGT expire warning NOT deleted`

Cause: The credentials cache is missing or corrupted.

Solution: Check that the cache location provided is correct. Remove and obtain a new TGT using `kinit`, if necessary.

`Kerberos authentication failed`

Cause: The Kerberos password is either incorrect or the password might not be synchronized with the UNIX password.

Solution: If the password are not synchronized, then you must specify a different password to complete Kerberos authentication. It is possible that the user has forgotten their original password.

`Kerberos V5 refuses authentication`

Cause: Authentication could not be negotiated with the server.

Solution: Start authentication debugging by invoking the `telnet` command with the `toggle authdebug` command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

`Key table entry not found`

Cause: No entry exists for the service principal in the network application server's keytab file.

Solution: Add the appropriate service principal to the server's keytab file so that it can provide the Kerberized service.

`Key version number for principal in key table is incorrect`

Cause: A principal's key version in the keytab file is different from the version in the Kerberos database. Either a service's key has been changed, or you might be using an old service ticket.

Solution: If a service's key has been changed (for example, by using `kadmin`), you need to extract the new key and store it in the host's keytab file where the service is running.

Alternately, you might be using an old service ticket that has an older key. You might want to run the `kdestroy` command and then the `kinit` command again.

`kinit: gethostname failed`

Cause: An error in the local network configuration is causing `kinit` to fail.

Solution: Make sure that the host is configured correctly.

`login: load_modules: can not open module /usr/lib/security/pam_krb5.so.1`

Cause: Either the Kerberos PAM module is missing or it is not a valid executable binary.

Solution: Make sure that the Kerberos PAM module is in the `/usr/lib/security` directory and that it is a valid executable binary. Also, make sure that the `/etc/pam.conf` file contains the correct path to `pam_krb5.so.1`.

`Looping detected inside krb5_get_in_tkt`

Cause: Kerberos made several attempts to get the initial tickets but failed.

Solution: Make sure that at least one KDC is responding to authentication requests.

`Master key does not match database`

Cause: The loaded database dump was not created from a database that contains the master key. The master key is located in `/var/krb5/.k5.`*REALM*.

Solution: Make sure that the master key in the loaded database dump matches the master key that is located in `/var/krb5/.k5.`*REALM*.

`Matching credential not found`

Cause: The matching credential for your request was not found. Your request requires credentials that are unavailable in the credentials cache.

Solution: Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

`Message out of order`

Cause: Messages that were sent using sequential-order privacy arrived out of order. Some messages might have been lost in transit.

Solution: You should reinitialize the Kerberos session.

`Message stream modified`

Cause: There was a mismatch between the computed checksum and the message checksum. The message might have been modified while in transit, which can indicate a security leak.

Solution: Make sure that the messages are being sent across the network correctly. Because this message can also indicate the possible tampering of messages while they are being sent, destroy your tickets using `kdestroy` and reinitialize the Kerberos services that you are using.

Common Kerberos Error Messages (N-Z)

This section provides an alphabetical list (N-Z) of common error messages for the Kerberos commands, Kerberos daemons, PAM framework, GSS interface, the NFS service, and the Kerberos library.

`No credentials cache file found`

Cause: Kerberos could not find the credentials cache (`/tmp/krb5cc_`*uid*).

Solution: Make sure that the credential file exists and is readable. If it isn't, try performing `kinit` again.

`No credentials were supplied, or the credentials were unavailable or inaccessible`

`No credential cache found`

Cause: The user's credential cache is incorrect or does not exist.

Solution: The user should run `kinit` before trying to start the service.

`No credentials were supplied, or the credentials were unavailable or inaccessible`

`No principal in keytab matches desired name`

Cause: An error occurred while trying to authenticate the server.

Solution: Make sure that the host or service principal is in the server's keytab file.

`Operation requires "`*privilege*`" privilege`

Cause: The `admin` principal that was being used does not have the appropriate privilege configured in the `kadm5.acl` file.

Solution: Use a principal that has the appropriate privileges. Or, configure the principal that was being used to have the appropriate privileges by modifying the `kadm5.acl` file. Usually, a principal with `/admin` as part of its name has the appropriate privileges.

`PAM-KRB5 (auth): krb5_verify_init_creds failed: Key table entry not found`

Cause: The remote application tried to read the host's service principal in the local `/etc/krb5/krb5.keytab` file, but one does not exist.

Solution: Add the host's service principal to the host's keytab file.

`Password is in the password dictionary`

Cause: The password that you specified is in a password dictionary that is being used. Your password is not a good choice for a password.

Solution: Choose a password that has a mix of password classes.

`Permission denied in replay cache code`

Cause: The system's replay cache could not be opened. Your server might have been first run under a user ID different than your current user ID.

Solution: Make sure that the replay cache has the appropriate permissions. The replay cache is stored on the host where the Kerberized server application is running. The replay cache file is called `/var/krb5/rcache/rc_`*service_name_uid* for non-`root` users. For root users the replay cache file is called `/var/krb5/rcache/root/rc_`*service_name*.

`Protocol version mismatch`

Cause: Most likely, a Kerberos V4 request was sent to the KDC. The Kerberos service supports only the Kerberos V5 protocol.

Solution: Make sure that your applications are using the Kerberos V5 protocol.

`Request is a replay`

Cause: The request has already been sent to this server and processed. The tickets might have been stolen, and someone else is trying to reuse the tickets.

Solution: Wait for a few minutes, and reissue the request.

`Requested principal and ticket don’t match`

Cause: The service principal that you are connecting to and the service ticket that you have do not match.

Solution: Make sure that DNS is functioning properly. If you are using another vendor's software, make sure that the software is using principal names correctly.

`Requested protocol version not supported`

Cause: Most likely, a Kerberos V4 request was sent to the KDC. The Kerberos service supports only the Kerberos V5 protocol.

Solution: Make sure that your applications are using the Kerberos V5 protocol.

`Server refused to negotiate authentication, which is required for encryption. Good bye.`

Cause: The remote application is not capable or has been configured not to accept Kerberos authentication from the client.

Solution: Provide a remote application that can negotiate authentication or configure the application to use the appropriate flags to turn on authentication.

`Server refused to negotiate encryption. Good bye.`

Cause: Encryption could not be negotiated with the server.

Solution: Start authentication debugging by invoking the `telnet` command with the `toggle encdebug`command and look at the debug messages for further clues.

`Server rejected authentication (during sendauth exchange)`

Cause: The server that you are trying to communicate with rejected the authentication. Most often, this error occurs during Kerberos database propagation. Some common causes might be problems with the `kpropd.acl` file, DNS, or the keytab file.

Solution: If you get this error when you are running applications other than `kprop`, investigate whether the server's keytab file is correct.

`The ticket isn’t for us`

`Ticket/authenticator don’t match`

Cause: There was a mismatch between the ticket and the authenticator. The principal name in the request might not have matched the service principal's name. Either because the ticket was being sent with an FQDN name of the principal while the service expected a non-FQDN name, or a non-FDQN name was sent when the service expected an FQDN name.

Solution: If you get this error when you are running applications other than `kprop`, investigate whether the server's keytab file is correct.

`Ticket expired`

Cause: Your ticket times have expired.

Solution: Destroy your tickets with `kdestroy`, and create new tickets with `kinit`.

`Ticket is ineligible for postdating`

Cause: The principal does not allow its tickets to be postdated.

Solution: Modify the principal with `kadmin` to allow postdating.

`Ticket not yet valid`

Cause: The postdated ticket is not valid yet.

Solution: Create a new ticket with the correct date, or wait until the current ticket is valid.

`Truncated input file detected`

Cause: The database dump file that was being used in the operation is not a complete dump file.

Solution: Create the dump file again, or use a different database dump file.

`Unable to securely authenticate user ... exit`

Cause: Authentication could not be negotiated with the server.

Solution: Start authentication debugging by invoking the `telnet` command with the `toggle authdebug` command and look at the debug messages for further clues. Also, make sure that you have valid credentials.

`Wrong principal in request`

Cause: There was an invalid principal name in the ticket. This error might indicate a DNS or FQDN problem.

Solution: Make sure that the principal of the service matches the principal in the ticket.

Kerberos Troubleshooting

This section provides troubleshooting information for the Kerberos software.

Problems With the Format of the `krb5.conf` File

If the `krb5.conf` file is not formatted properly, the `telnet` command will fail. However, the `dtlogin` and `login` commands will still succeed, even if the `krb5.conf` file is specified as required for the commands. If this problem occurs, the following error message is displayed:

```
Error initializing krb5: Improper format of Kerberos configuration
```

In addition, an incorrectly formatted `krb5.conf` file, prevents the applications that use the GSSAPI from using the `krb5` mechanisms.

If there is a problem with the format of the `krb5.conf` file, you are vulnerable to security breaches. You should fix the problem before you allow Kerberos features to be used.

Problems Propagating the Kerberos Database

If propagating the Kerberos database fails, try `/usr/bin/rlogin -x` between the slave KDC and master KDC, and from the master KDC to the slave KDC server.

If the KDCs have been set up to restrict access, `rlogin` is disabled and cannot be used to troubleshoot this problem. To enable `rlogin` on a KDC, you must enable the `eklogin` service.

```
# svcadm enable svc:/network/login:eklogin
```

After you finish troubleshooting the problem, you need to disable the `eklogin` service..

If `rlogin` does not work, problems are likely because of the keytab files on the KDCs. If `rlogin` does work, the problem is not in the keytab file or the name service, because `rlogin` and the propagation software use the same `host/`*host-name* principal. In this case, make sure that the `kpropd.acl` file is correct.

Problems Mounting a Kerberized NFS File System

- If mounting a Kerberized NFS file system fails, make sure that the `/var/rcache/root` file exists on the NFS server. If the file system is not owned by `root`, remove it and try the mount again.
- If you have a problem accessing a Kerberized NFS file system, make sure that the `gssd` service is enabled on your system and the NFS server.
- If you see either the `invalid argument` or `bad directory` error message when you are trying to access a Kerberized NFS file system, the problem might be that you are not using a fully qualified DNS name when you are trying to mount the NFS file system. The host that is being mounted is not the same as the host name part of the service principal in the server's keytab file.

  This problem might also occur if your server has multiple Ethernet interfaces, and you have set up DNS to use a “name per interface” scheme instead of a “multiple address records per host” scheme. For the Kerberos service, you should set up multiple address records per host as follows[1] :

```
my.host.name.   A       1.2.3.4
                A       1.2.4.4
                A       1.2.5.4

my-en0.host.name.       A       1.2.3.4
my-en1.host.name.       A       1.2.4.4
my-en2.host.name.       A       1.2.5.4

4.3.2.1         PTR     my.host.name.
4.4.2.1         PTR     my.host.name.
4.5.2.1         PTR     my.host.name.
```

In this example, the setup allows one reference to the different interfaces and a single service principal instead of three service principals in the server's keytab file.

Problems Authenticating as `root`

If authentication fails when you try to become superuser on your system and you have already added the `root` principal to your host's keytab file, there are two potential problems to check. First, make sure that the `root` principal in the keytab file has a fully qualified host name as its instance. If it does, check the `/etc/resolv.conf` file to make sure that the system is correctly set up as a DNS client.

---

[1] Ken Hornstein, “Kerberos FAQ,” [`http://www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html`], accessed 11 December 1998.

Observing Mapping from GSS Credentials to UNIX Credentials

To be able to monitor the credential mappings, first uncomment this line from the `/etc/gss/gsscred.conf` file.

```
SYSLOG_UID_MAPPING=yes
```

Next instruct the `gssd` service to get information from the `/etc/gss/gsscred.conf` file.

```
# pkill -HUP gssd
```

Now you should be able to monitor the credential mappings as `gssd` requests them. The mappings are recorded by `syslogd`, if the `syslog.conf` file is configured for the `auth` system facility with the `debug` severity level.

◆ ◆ ◆ CHAPTER 25

Administering Kerberos Principals and Policies (Tasks)

This chapter provides procedures for administering principals and the policies that are associated with them. This chapter also shows how to administer a host's keytab file.

This chapter should be used by anyone who needs to administer principals and policies. Before you use this chapter, you should be familiar with principals and policies, including any planning considerations. Refer to Chapter 21, "Introduction to the Kerberos Service," and Chapter 22, "Planning for the Kerberos Service," respectively.

This is a list of the information in this chapter.

- "Ways to Administer Kerberos Principals and Policies" on page 475
- "SEAM Administration Tool" on page 476
- "Administering Kerberos Principals" on page 480
- "Administering Kerberos Policies" on page 493
- "SEAM Tool Reference" on page 501
- "Administering Keytab Files" on page 505

Ways to Administer Kerberos Principals and Policies

The Kerberos database on the master KDC contains all of your realm's Kerberos principals, their passwords, policies, and other administrative information. To create and delete principals, and to modify their attributes, you can use either the `kadmin` or `gkadmin` command.

The `kadmin` command provides an interactive command-line interface that enables you to maintain Kerberos principals, policies, and keytab files. There are two versions of the `kadmin` command:

- `kadmin` – Uses Kerberos authentication to operate securely from anywhere on the network
- `kadmin.local` – Must be run directly on the master KDC

Other than `kadmin` using Kerberos to authenticate the user, the capabilities of the two versions are identical. The local version is necessary to enable you to set up enough of the database so that you can use the remote version.

Also, the Solaris release provides the SEAM Administration Tool, `gkadmin`, which is an interactive graphical user interface (GUI) that provides essentially the same capabilities as the `kadmin` command. See "SEAM Administration Tool" on page 476 for more information.

SEAM Administration Tool

The SEAM Administration Tool (SEAM Tool) is an interactive graphical user interface (GUI) that enables you to maintain Kerberos principals and policies. This tool provides much the same capabilities as the `kadmin` command. However, this tool does not support the management of keytab files. You must use the `kadmin` command to administer keytab files, which is described in "Administering Keytab Files" on page 505.

Similar to the `kadmin` command, the SEAM Tool uses Kerberos authentication and encrypted RPC to operate securely from anywhere on the network. The SEAM Tool enables you to do the following:

- Create new principals that are based on default values or existing principals.
- Create new policies that are based on existing policies.
- Add comments for principals.
- Set up default values for creating new principals.
- Log in as another principal without exiting the tool.
- Print or save principal lists and policy lists.
- View and search principal lists and policy lists.

The SEAM Tool also provides context-sensitive help and general online help.

The following task maps provide pointers to the various tasks that you can do with the SEAM Tool:

- "Administering Kerberos Principals (Task Map)" on page 481
- "Administering Kerberos Policies (Task Map)" on page 493

Also, go to "SEAM Tool Panel Descriptions" on page 501 for descriptions of all the principal attributes and policy attributes that you can either specify or view in the SEAM Tool.

Command-Line Equivalents of the SEAM Tool

This section lists the `kadmin` commands that provide the same capabilities as the SEAM Tool. These commands can be used without running an X Window system. Even though most procedures in this chapter use the SEAM Tool, many procedures also provide corresponding examples that use the command-line equivalents.

TABLE 25–1 Command-Line Equivalents of the SEAM Tool

SEAM Tool Procedure	Equivalent `kadmin` Command
View the list of principals.	`list_principals` or `get_principals`
View a principal's attributes.	`get_principal`
Create a new principal.	`add_principal`
Duplicate a principal.	No command-line equivalent
Modify a principal.	`modify_principal` or `change_password`
Delete a principal.	`delete_principal`
Set up defaults for creating new principals.	No command-line equivalent
View the list of policies.	`list_policies` or `get_policies`
View a policy's attributes.	`get_policy`
Create a new policy.	`add_policy`
Duplicate a policy.	No command-line equivalent
Modify a policy.	`modify_policy`
Delete a policy.	`delete_policy`

The Only File Modified by the SEAM Tool

The only file that the SEAM Tool modifies is the `$HOME/.gkadmin` file. This file contains the default values for creating new principals. You can update this file by choosing Properties from the Edit menu.

Print and Online Help Features of the SEAM Tool

The SEAM Tool provides both print features and online help features. From the Print menu, you can send the following to a printer or a file:

- List of available principals on the specified master KDC
- List of available policies on the specified master KDC
- The currently selected principal or the loaded principal
- The currently selected policy or the loaded policy

From the Help menu, you can access context-sensitive help and general help. When you choose Context-Sensitive Help from the Help menu, the Context-Sensitive Help window is displayed and the tool is switched to help mode. In help mode, when you click on any fields, labels, or buttons on the window, help on that item is displayed in the Help window. To switch back to the tool's normal mode, click Dismiss in the Help window.

You can also choose Help Contents, which opens an HTML browser that provides pointers to the general overview and task information that is provided in this chapter.

Working With Large Lists in the SEAM Tool

As your site starts to accumulate a large number of principals and policies, the time it takes the SEAM Tool to load and display the principal and policy lists will become increasingly longer. Thus, your productivity with the tool will decrease. There are several ways to work around this problem.

First, you can completely eliminate the time to load the lists by not having the SEAM Tool load the lists. You can set this option by choosing Properties from the Edit menu, and unchecking the Show Lists field. Of course, when the tool doesn't load the lists, it can't display the lists, and you can no longer use the list panels to select principals or policies. Instead, you must type a principal or policy name in the new Name field that is provided, then select the operation that you want to perform on it. In effect, typing a name is equivalent to selecting an item from the list.

Another way to work with large lists is to cache them. In fact, caching the lists for a limited time is set as the default behavior for the SEAM Tool. The SEAM Tool must still initially load the lists into the cache. But after that, the tool can use the cache rather than retrieve the lists again. This option eliminates the need to keep loading the lists from the server, which is what takes so long.

You can set list caching by choosing Properties from the Edit menu. There are two cache settings. You can choose to cache the list forever, or you can specify a time limit when the tool must reload the lists from the server into the cache.

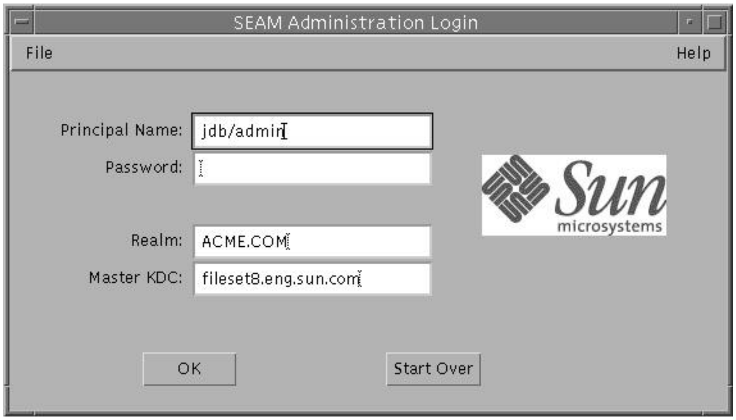
Caching the lists still enables you to use the list panels to select principals and policies, so it doesn't affect how you use the SEAM Tool as the first option does. Also, even though caching doesn't enable you to see the changes of other users, you can still see the latest list information based on your changes, because your changes update the lists both on the server and in the cache. And, if you want to update the cache to see other changes and get the lastest copy of the lists, you can use the Refresh menu whenever you want to refresh the cache from the server.

▼ How to Start the SEAM Tool

1 **Start the SEAM Tool by using the `gkadmin` command.**

```
$ /usr/sbin/gkadmin
```

The SEAM Administration Login window is displayed.



Principal Name: jdb/admin

Password:

Realm: ACME.COM

Master KDC: fileset8.eng.sun.com

2 **If you don't want to use the default values, specify new default values.**

The window automatically fills in with default values. The default principal name is determined by taking your current identity from the `USER` environment variable and appending `/admin` to it (*username*`/admin`). The default Realm and Master KDC fields are selected from the `/etc/krb5/krb5.conf` file. If you ever want to retrieve the default values, click Start Over.

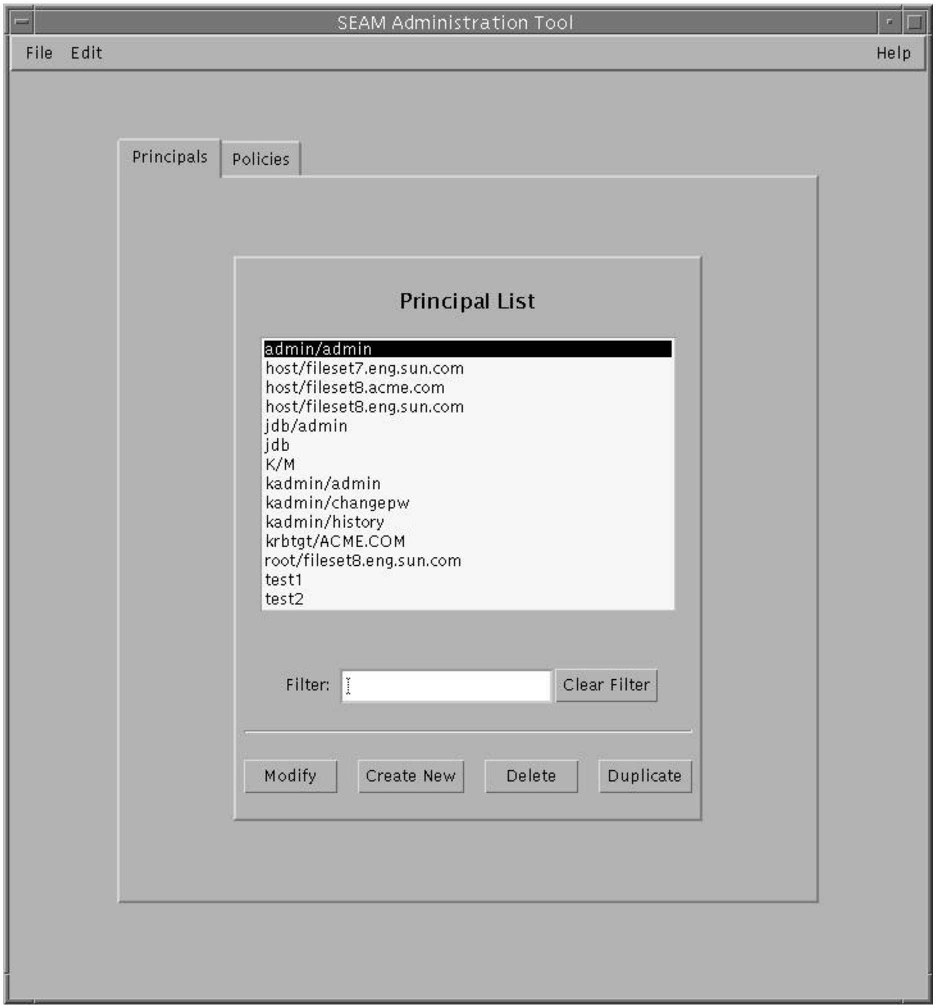
Note – The administration operations that each Principal Name can perform are dictated by the Kerberos ACL file, `/etc/krb5/kadm5.acl`. For information about limited privileges, see "Using the SEAM Tool With Limited Kerberos Administration Privileges" on page 504.

---

3 **Type a password for the specified principal name.**

4 **Click OK.**

The following window is displayed.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Principal List
Filter:
Clear Filter
Modify
Create New
Delete
Duplicate

admin/admin
host/fileset7.eng.sun.com
host/fileset8.acme.com
host/fileset8.eng.sun.com
jdb/admin
jdb
K/M
kadmin/admin
kadmin/changepw
kadmin/history
krbtgt/ACME.COM
root/fileset8.eng.sun.com
test1
test2

Administering Kerberos Principals

This section provides the step-by-step instructions used to administer principals with the SEAM Tool. This section also provides examples of command-line equivalents, when available.

Administering Kerberos Principals (Task Map)

Task	Description	For Instructions
View the list of principals.	View the list of principals by clicking the Principals tab.	"How to View the List of Kerberos Principals" on page 482
View a principal's attributes.	View a principal's attributes by selecting the Principal in the Principal List, then clicking the Modify button.	"How to View a Kerberos Principal's Attributes" on page 484
Create a new principal.	Create a new principal by clicking the Create New button in the Principal List panel.	"How to Create a New Kerberos Principal" on page 486
Duplicate a principal.	Duplicate a principal by selecting the principal to duplicate in the Principal List, then clicking the Duplicate button.	"How to Duplicate a Kerberos Principal" on page 488
Modify a principal.	Modify a principal by selecting the principal to modify in the Principal List, then clicking the Modify button. Note that you cannot modify a principal's name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.	"How to Modify a Kerberos Principal" on page 488
Delete a principal.	Delete a principal by selecting the principal to delete in the Principal List, then clicking the Delete button.	"How to Delete a Kerberos Principal" on page 489
Set up defaults for creating new principals.	Set up defaults for creating new principals by choosing Properties from the Edit menu.	"How to Set Up Defaults for Creating New Kerberos Principals" on page 490
Modify the Kerberos administration privileges (`kadm5.acl` file).	*Command-line only.* The Kerberos administration privileges determine what operations a principal can perform on the Kerberos database, such as add and modify. You need to edit the `/etc/krb5/kadm5.acl` file to modify the Kerberos administration privileges for each principal.	"How to Modify the Kerberos Administration Privileges" on page 491

Automating the Creation of New Kerberos Principals

Even though the SEAM Tool provides ease-of-use, it doesn't provide a way to automate the creation of new principals. Automation is especially useful if you need to add 10 or even 100 new principals in a short time. However, by using the `kadmin.local` command in a Bourne shell script, you can do just that.

The following shell script line is an example of how to automate the creation of new principals:

```
awk '{ print "ank +needchange -pw", $2, $1 }' < /tmp/princnames |
        time /usr/sbin/kadmin.local> /dev/null
```

This example is split over two lines for readability. The script reads in a file called `princnames` that contains principal names and their passwords, and adds them to the Kerberos database. You would have to create the `princnames` file, which contains a principal name and its password on each line, separated by one or more spaces. The `+needchange` option configures the principal so that the user is prompted for a new password during login with the principal for the first time. This practice helps to ensure that the passwords in the `princnames` file are not a security risk.

You can build more elaborate scripts. For example, your script could use the information in the name service to obtain the list of user names for the principal names. What you do and how you do it is determined by your site's needs and your scripting expertise.

▼ How to View the List of Kerberos Principals

An example of the command-line equivalent follows this procedure.

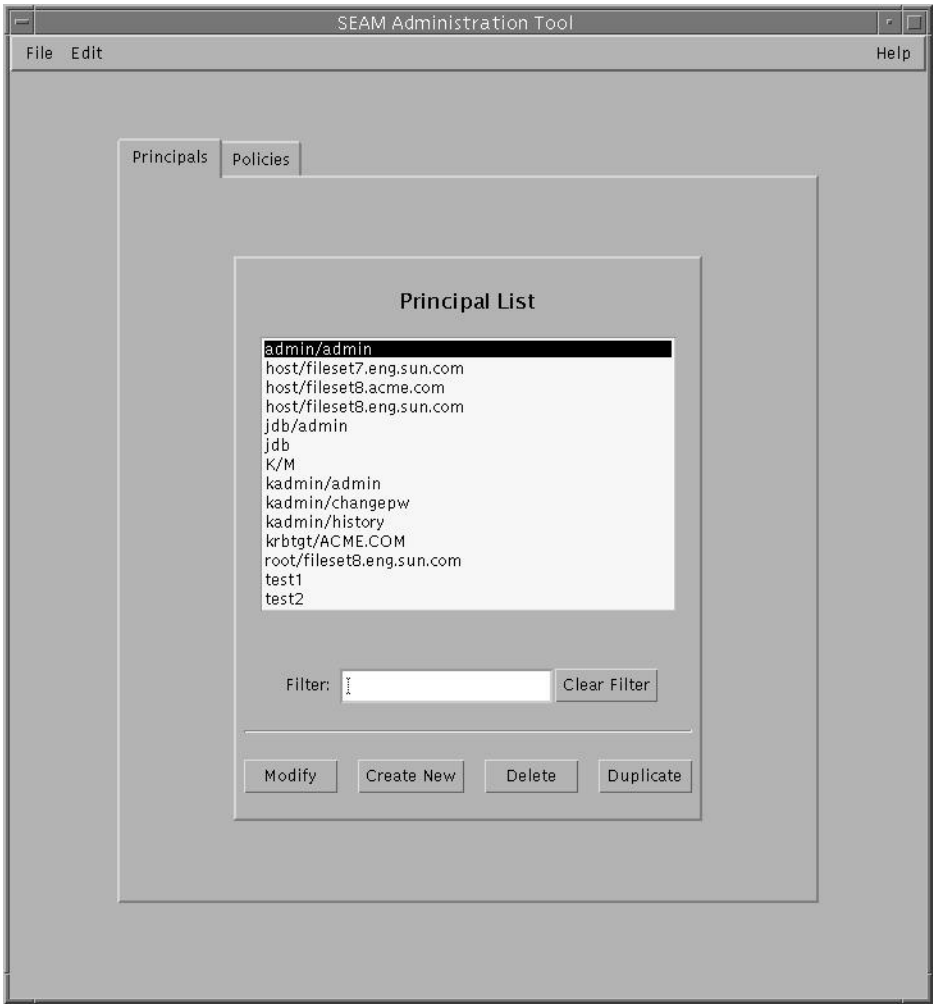
1. **If necessary, start the SEAM Tool.**

   See "How to Start the SEAM Tool" on page 478 for more information.

   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Principals tab.**

   The list of principals is displayed.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Principal List
Filter:
Clear Filter
Modify
Create New
Delete
Duplicate

admin/admin
host/fileset7.eng.sun.com
host/fileset8.acme.com
host/fileset8.eng.sun.com
jdb/admin
jdb
K/M
kadmin/admin
kadmin/changepw
kadmin/history
krbtgt/ACME.COM
root/fileset8.eng.sun.com
test1
test2

3 Display a specific principal or a sublist of principals.

Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of principals that match the filter is displayed.

The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you type the filter string `ge`, the filter mechanism displays only the principals with the `ge` string in them (for example, `george` or `edge`).

If you want to display the entire list of principals, click Clear Filter.

Example 25–1 Viewing the List of Kerberos Principals (Command Line)

In the following example, the `list_principals` command of `kadmin` is used to list all the principals that match `test*`. Wildcards can be used with the `list_principals` command.

```
kadmin: list_principals test*
test1@EXAMPLE.COM
test2@EXAMPLE.COM
kadmin: quit
```

▼ How to View a Kerberos Principal's Attributes

An example of the command-line equivalent follows this procedure.

1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Principals tab.**

3 **Select the principal in the list that you want to view, then click Modify.**

The Principal Basics panel that contains some of the principal's attributes is displayed.

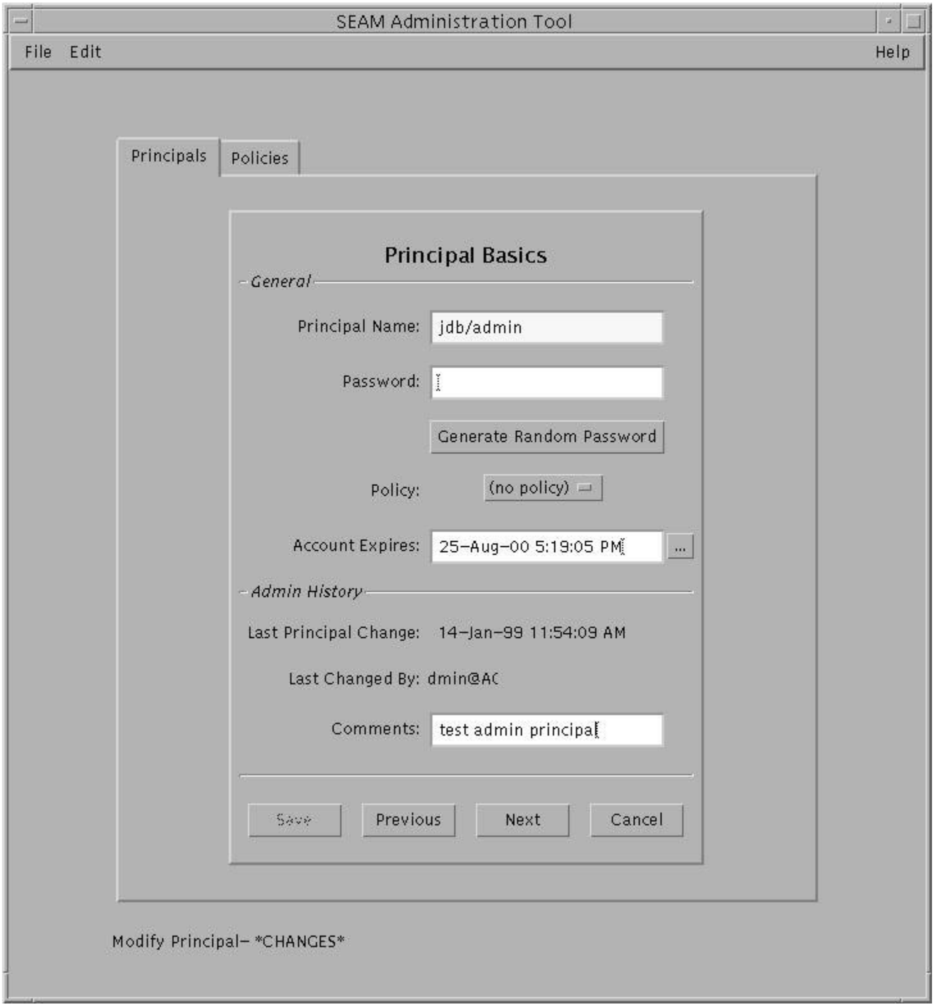
4 **Continue to click Next to view all the principal's attributes.**

Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 501.

5 **When you are finished viewing, click Cancel.**

Example 25–2 Viewing a Kerberos Principal's Attributes

The following example shows the first window when you are viewing the `jdb/admin` principal.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Generate Random Password
...
Save
Previous
Next
Cancel

Principal Basics

General

Principal Name: jdb/admin

Password:

Policy: (no policy)

Account Expires: 25–Aug–00 5:19:05 PM

Admin History

Last Principal Change: 14–Jan–99 11:54:09 AM

Last Changed By: dmin@AC

Comments: test admin principal

Modify Principal– *CHANGES*

Example 25–3 Viewing a Kerberos Principal's Attributes (Command Line)

In the following example, the `get_principal` command of `kadmin` is used to view the attributes of the `jdb/admin` principal.

```
kadmin: getprinc jdb/admin
Principal: jdb/admin@EXAMPLE.COM
Expiration date: Fri Aug 25 17:19:05 PDT 2004
Last password change: [never]
Password expiration date: Wed Apr 14 11:53:10 PDT 2003
Maximum ticket life: 1 day 16:00:00
Maximum renewable life: 1 day 16:00:00
```

```
Last modified: Thu Jan 14 11:54:09 PST 2003 (admin/admin@EXAMPLE.COM)
Last successful authentication: [never]
Last failed authentication: [never]
Failed password attempts: 0
Number of keys: 1
Key: vno 1, DES cbc mode with CRC-32, no salt
Attributes: REQUIRES_HW_AUTH
Policy: [none]
kadmin: quit
```

▼ How to Create a New Kerberos Principal

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**

   See "How to Start the SEAM Tool" on page 478 for more information.

   Note – If you are creating a new principal that might need a new policy, you should create the new policy before you create the new principal. Go to "How to Create a New Kerberos Policy" on page 497.

   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Principals tab.**

3. **Click New.**

   The Principal Basics panel that contains some attributes for a principal is displayed.

4. **Specify a principal name and a password.**

   Both the principal name and the password are mandatory.

5. **Specify values for the principal's attributes, and continue to click Next to specify more attributes.**

   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 501.

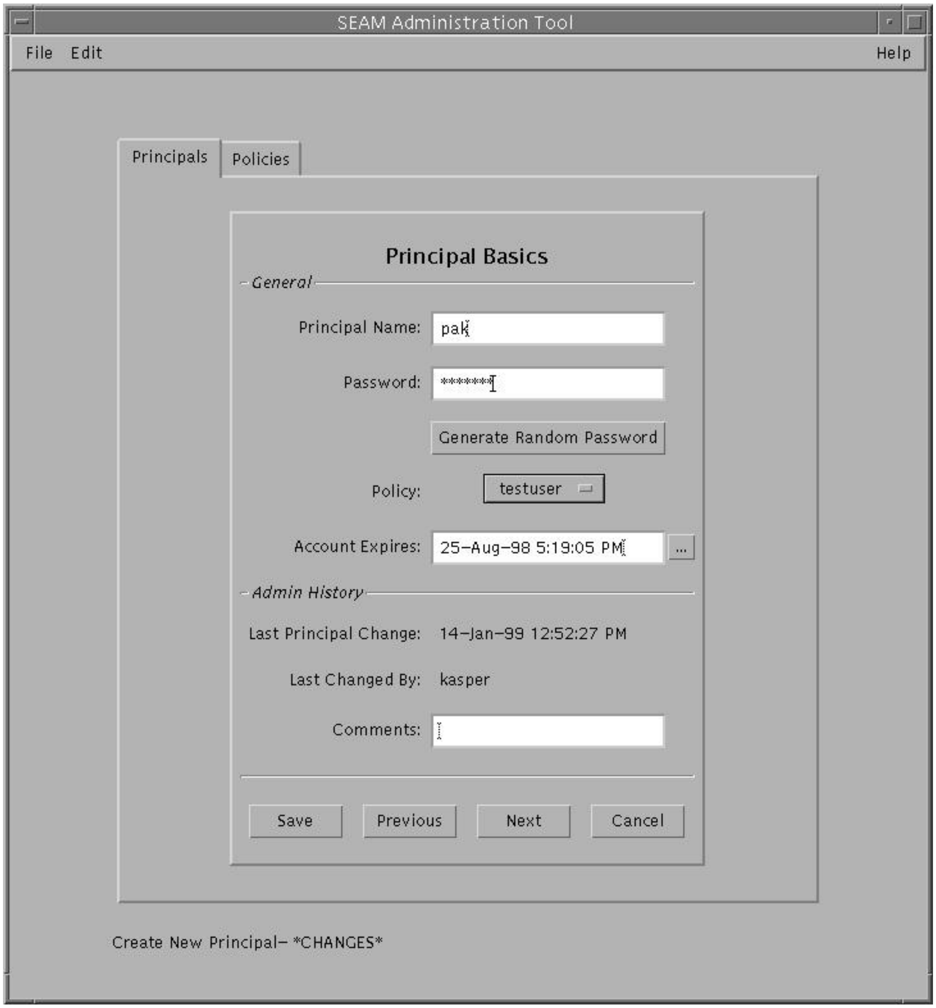
6. **Click Save to save the principal, or click Done on the last panel.**

7. **If needed, set up Kerberos administration privileges for the new principal in the** `/etc/krb5/kadm5.acl` **file.**

   See "How to Modify the Kerberos Administration Privileges" on page 491 for more details.

Example 25–4 Creating a New Kerberos Principal

The following example shows the Principal Basics panel when a new principal called `pak` is created. The policy is set to `testuser`.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Principal Basics
General
Generate Random Password
Admin History
Save
Previous
Next
Cancel

Principal Name: pak

Password: *******

Policy: testuser

Account Expires: 25–Aug–98 5:19:05 PM

Last Principal Change: 14–Jan–99 12:52:27 PM

Last Changed By: kasper

Comments:

Create New Principal– *CHANGES*

Example 25–5 Creating a New Kerberos Principal (Command Line)

In the following example, the `add_principal` command of `kadmin` is used to create a new principal called `pak`. The principal's policy is set to `testuser`.

```
kadmin: add_principal -policy testuser pak
Enter password for principal "pak@EXAMPLE.COM": <Type the password>
Re-enter password for principal "pak@EXAMPLE.COM": <Type the password again>
```

```
Principal "pak@EXAMPLE.COM" created.
kadmin: quit
```

▼ How to Duplicate a Kerberos Principal

This procedure explains how to use all or some of the attributes of an existing principal to create a new principal. No command-line equivalent exists for this procedure.

1. **If necessary, start the SEAM Tool.**

   See "How to Start the SEAM Tool" on page 478 for more information.

   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Principals tab.**

3. **Select the principal in the list that you want to duplicate, then click Duplicate.**

   The Principal Basics panel is displayed. All the attributes of the selected principal are duplicated, except for the Principal Name and Password fields, which are empty.

4. **Specify a principal name and a password.**

   Both the principal name and the password are mandatory. To make an exact duplicate of the principal you selected, click Save and skip to Step 7.

5. **Specify different values for the principal's attributes, and continue to click Next to specify more attributes.**

   Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 501.

6. **Click Save to save the principal, or click Done on the last panel.**

7. **If needed, set up Kerberos administration privileges for the principal in `/etc/krb5/kadm5.acl` file.**

   See "How to Modify the Kerberos Administration Privileges" on page 491 for more details.

▼ How to Modify a Kerberos Principal

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**

   See "How to Start the SEAM Tool" on page 478 for more information.

   ```
   $ /usr/sbin/gkadmin
   ```

2 **Click the Principals tab.**

3 **Select the principal in the list that you want to modify, then click Modify.**

The Principal Basics panel that contains some of the attributes for the principal is displayed.

4 **Modify the principal's attributes, and continue to click Next to modify more attributes.**

Three windows contain attribute information. Choose Context-Sensitive Help from the Help menu to get information about the various attributes in each window. Or, for all the principal attribute descriptions, go to "SEAM Tool Panel Descriptions" on page 501.

Note – You cannot modify a principal's name. To rename a principal, you must duplicate the principal, specify a new name for it, save it, and then delete the old principal.

---

5 **Click Save to save the principal, or click Done on the last panel.**

6 **Modify the Kerberos administration privileges for the principal in the `/etc/krb5/kadm5.acl` file.**

See "How to Modify the Kerberos Administration Privileges" on page 491 for more details.

Example 25–6 Modifying a Kerberos Principal's Password (Command Line)

In the following example, the `change_password` command of `kadmin` is used to modify the password for the `jdb` principal. The `change_password` command does not let you change the password to a password that is in the principal's password history.

```
kadmin: change_password jdb
Enter password for principal "jdb": <Type the new password>
Re-enter password for principal "jdb": <Type the password again>
Password for "jdb@EXAMPLE.COM" changed.
kadmin: quit
```

To modify other attributes for a principal, you must use the `modify_principal` command of `kadmin`.

▼ How to Delete a Kerberos Principal

An example of the command-line equivalent follows this procedure.

1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Principals tab.**

3 **Select the principal in the list that you want to delete, then click Delete.**

After you confirm the deletion, the principal is deleted.

4 **Remove the principal from the Kerberos access control list (ACL) file,** `/etc/krb5/kadm5.acl`**.**

See "How to Modify the Kerberos Administration Privileges" on page 491 for more details.

Example 25–7 Deleting a Kerberos Principal (Command Line)

In the following example, the `delete_principal` command of `kadmin` is used to delete the `jdb` principal.

```
kadmin: delete_principal pak
Are you sure you want to delete the principal "pak@EXAMPLE.COM"? (yes/no): yes
Principal "pak@EXAMPLE.COM" deleted.
Make sure that you have removed this principal from all ACLs before reusing.
kadmin: quit
```

▼ How to Set Up Defaults for Creating New Kerberos Principals

No command-line equivalent exists for this procedure.

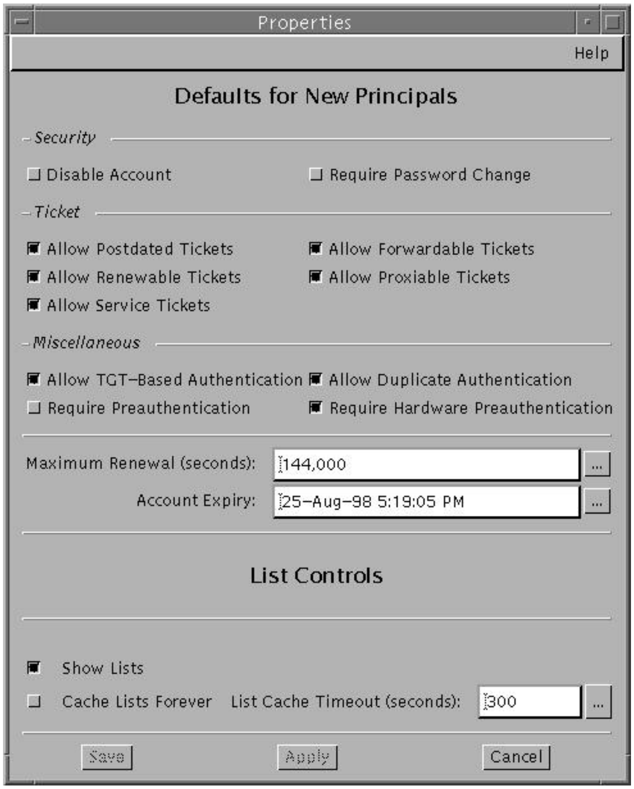
1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Choose Properties from the Edit Menu.**

The Properties window is displayed.



Defaults for New Principals

Security

☐ Disable Account
☐ Require Password Change

Ticket

☑ Allow Postdated Tickets
☑ Allow Forwardable Tickets
☑ Allow Renewable Tickets
☑ Allow Proxiable Tickets
☑ Allow Service Tickets

Miscellaneous

☑ Allow TGT–Based Authentication
☑ Allow Duplicate Authentication
☐ Require Preauthentication
☑ Require Hardware Preauthentication

Maximum Renewal (seconds): 144,000

Account Expiry: 25–Aug–98 5:19:05 PM

List Controls

☑ Show Lists

☐ Cache Lists Forever

List Cache Timeout (seconds): 300

3 **Select the defaults that you want to use when you create new principals.**

Choose Context-Sensitive Help from the Help menu for information about the various attributes in each window.

4 **Click Save.**

▼ How to Modify the Kerberos Administration Privileges

Even though your site probably has many user principals, you usually want only a few users to be able to administer the Kerberos database. Privileges to administer the Kerberos database are determined by the Kerberos access control list (ACL) file, `kadm5.acl`. The `kadm5.acl` file enables you to allow or disallow privileges for individual principals. Or, you can use the '*' wildcard in the principal name to specify privileges for groups of principals.

1 **Become superuser on the master KDC.**

2 **Edit the `/etc/krb5/kadm5.acl` file.**

An entry in the `kadm5.acl` file must have the following format:

principal privileges [*principal-target*]

principal	Specifies the principal to which the privileges are granted. Any part of the principal name can include the '*' wildcard, which is useful for providing the same privileges for a group of principals. For example, if you want to specify all principals with the `admin` instance, you would use `*/admin@`*realm*. Note that a common use of an `admin` instance is to grant separate privileges (such as administration access to the Kerberos database) to a separate Kerberos principal. For example, the user `jdb` might have a principal for his administrative use, called `jdb/admin`. This way, the user `jdb` obtains `jdb/admin` tickets only when he or she actually needs to use those privileges.
privileges	Specifies which operations can or cannot be performed by the principal. This field consists of a string of one or more of the following list of characters or their uppercase counterparts. If the character is uppercase (or not specified), then the operation is disallowed. If the character is lowercase, then the operation is permitted.

`a`	[Dis]allows the addition of principals or policies.
`d`	[Dis]allows the deletion of principals or policies.
`m`	[Dis]allows the modification of principals or polices.
`c`	[Dis]allows the changing of passwords for principals.
`i`	[Dis]allows inquiries to the Kerberos database.
`l`	[Dis]allows the listing of principals or policies in the Kerberos database.
`x` or `*`	Allows all privileges (`admcil`).

principal-target	When a principal is specified in this field, the *privileges* apply to the *principal* only when the *principal* operates on the *principal-target*. Any part of the principal name can include the '*' wildcard, which is useful to group principals.

Example 25–8 Modifying the Kerberos Administration Privileges

The following entry in the `kadm5.acl` file gives any principal in the `EXAMPLE.COM` realm with the `admin` instance all the privileges on the Kerberos database:

```
*/admin@EXAMPLE.COM *
```

The following entry in the `kadm5.acl` file gives the `jdb@EXAMPLE.COM` principal the privileges to add, list, and inquire about any principal that has the `root` instance.

```
jdb@EXAMPLE.COM ali */root@EXAMPLE.COM
```

Administering Kerberos Policies

This section provides step-by-step instructions used to administer policies with the SEAM Tool. This section also provides examples of command-line equivalents, when available.

Administering Kerberos Policies (Task Map)

Task	Description	For Instructions
View the list of policies.	View the list of policies by clicking the Policies tab.	"How to View the List of Kerberos Policies" on page 493
View a policy's attributes.	View a policy's attributes by selecting the policy in the Policy List, then clicking the Modify button.	"How to View a Kerberos Policy's Attributes" on page 495
Create a new policy.	Create a new policy by clicking the Create New button in the Policy List panel.	"How to Create a New Kerberos Policy" on page 497
Duplicate a policy.	Duplicate a policy by selecting the policy to duplicate in the Policy List, then clicking the Duplicate button.	"How to Duplicate a Kerberos Policy" on page 499
Modify a policy.	Modify a policy by selecting the policy to modify in the Policy List, then clicking the Modify button. Note that you cannot modify a policy's name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.	"How to Modify a Kerberos Policy" on page 499
Delete a policy.	Delete a policy by selecting the policy to delete in the Policy List, then clicking the Delete button.	"How to Delete a Kerberos Policy" on page 500

▼ How to View the List of Kerberos Policies

An example of the command-line equivalent follows this procedure.

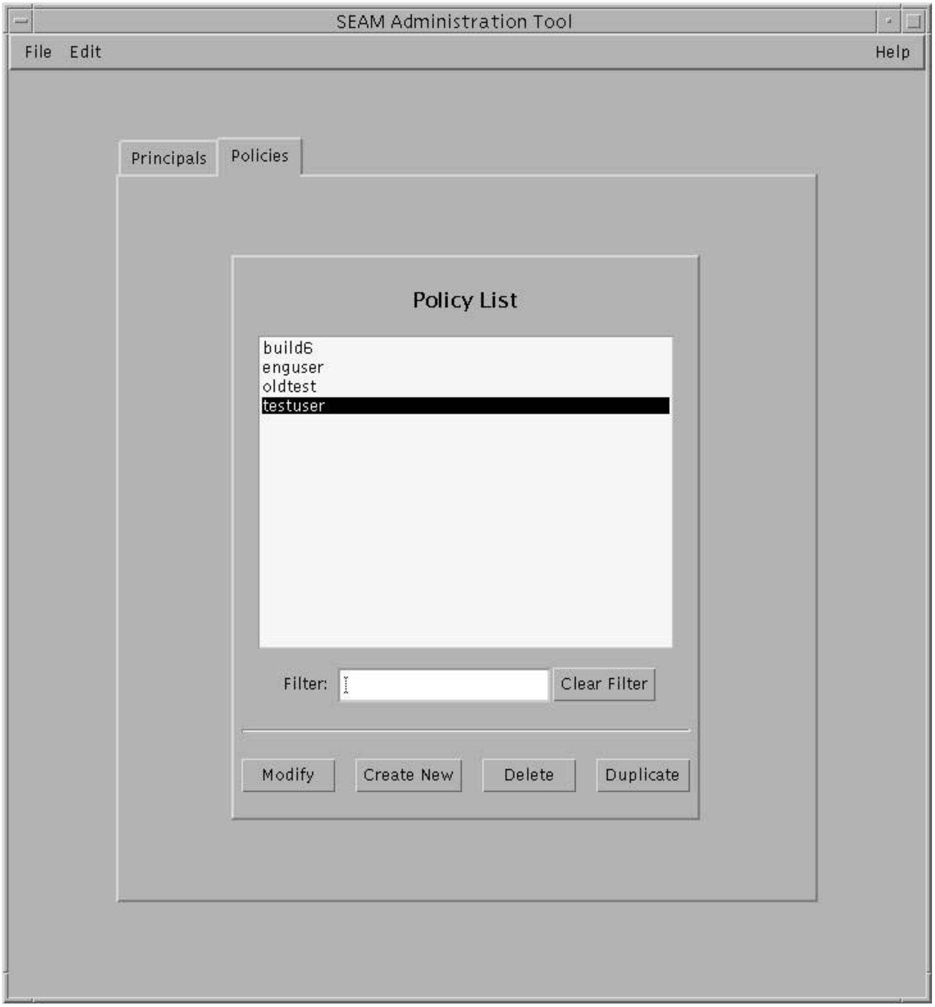
1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Policies tab.**

The list of policies is displayed.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Policy List
Filter:
Clear Filter
Modify
Create New
Delete
Duplicate

build6
enguser
oldtest
testuser

3. **Display a specific policy or a sublist of policies.**

   Type a filter string in the Filter field, and press Return. If the filter succeeds, the list of policies that match the filter is displayed.

   The filter string must consist of one or more characters. Because the filter mechanism is case sensitive, you need to use the appropriate uppercase and lowercase letters for the filter. For example, if you type the filter string `ge`, the filter mechanism displays only the policies with the `ge` string in them (for example, `george` or `edge`).

   If you want to display the entire list of policies, click Clear Filter.

Example 25–9 Viewing the List of Kerberos Policies (Command Line)

In the following example, the `list_policies` command of `kadmin` is used to list all the policies that match `*user*`. Wildcards can be used with the `list_policies` command.

```
kadmin: list_policies *user*
testuser
enguser
kadmin: quit
```

How to View a Kerberos Policy's Attributes

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**

   See "How to Start the SEAM Tool" on page 478 for more information.

   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Policies tab.**

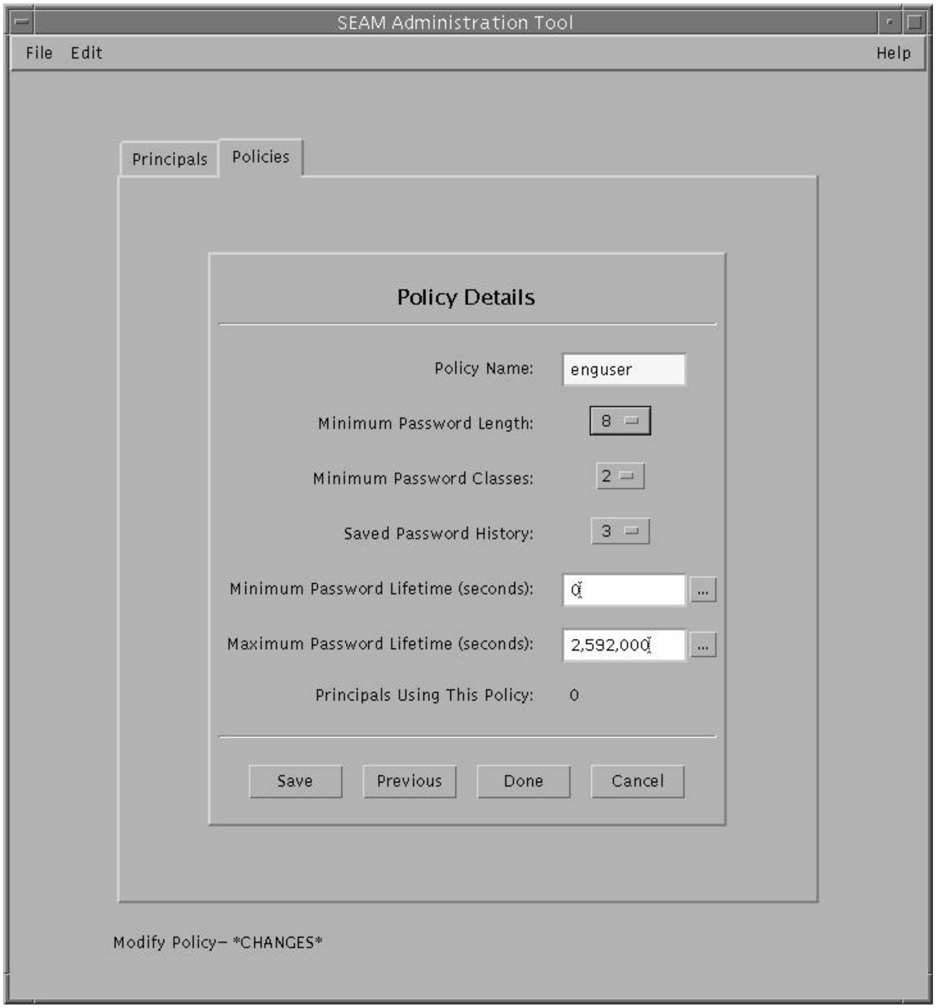
3. **Select the policy in the list that you want to view, then click Modify.**

   The Policy Details panel is displayed.

4. **When you are finished viewing, click Cancel.**

Example 25–10 Viewing a Kerberos Policy's Attributes

The following example shows the Policy Details panel when you are viewing the `test` policy.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Policy Details
Save
Previous
Done
Cancel

Policy Name: enguser

Minimum Password Length: 8

Minimum Password Classes: 2

Saved Password History: 3

Minimum Password Lifetime (seconds): 0

Maximum Password Lifetime (seconds): 2,592,000

Principals Using This Policy: 0

Modify Policy– *CHANGES*

Example 25–11 Viewing a Kerberos Policy's Attributes (Command Line)

In the following example, the `get_policy` command of `kadmin` is used to view the attributes of the `enguser` policy.

```
kadmin: get_policy enguser
Policy: enguser
Maximum password life: 2592000
Minimum password life: 0
Minimum password length: 8
Minimum number of password character classes: 2
Number of old keys kept: 3
```

```
Reference count: 0
kadmin: quit
```

The Reference count is the number of principals that use this policy.

▼ How to Create a New Kerberos Policy

An example of the command-line equivalent follows this procedure.

1. **If necessary, start the SEAM Tool.**

   See "How to Start the SEAM Tool" on page 478 for more information.

   ```
   $ /usr/sbin/gkadmin
   ```

2. **Click the Policies tab.**

3. **Click New.**

   The Policy Details panel is displayed.

4. **Specify a name for the policy in the Policy Name field.**

   The policy name is mandatory.

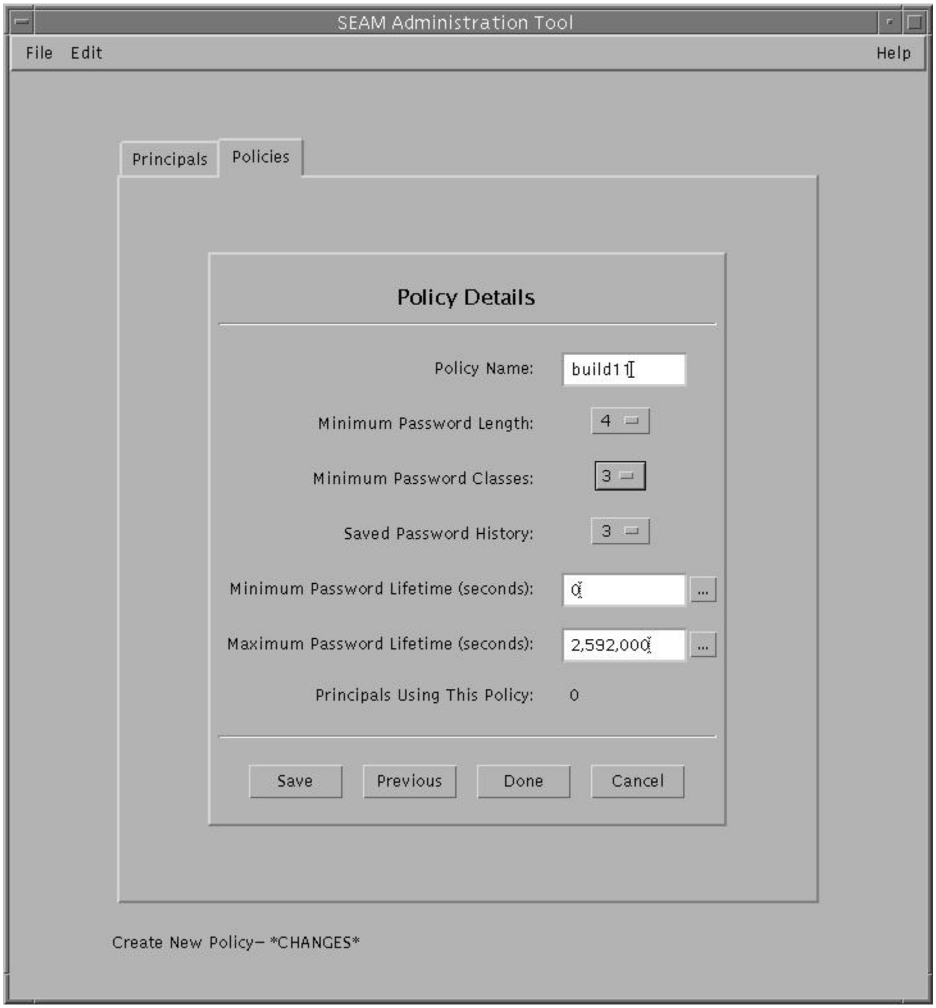
5. **Specify values for the policy's attributes.**

   Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 25–5 for all the policy attribute descriptions.

6. **Click Save to save the policy, or click Done.**

Example 25–12 Creating a New Kerberos Policy

In the following example, a new policy called `build11` is created. The Minimum Password Classes is set to 3.


SEAM Administration Tool
File
Edit
Help
Principals
Policies
Policy Details
Save
Previous
Done
Cancel

Policy Name: build11

Minimum Password Length: 4

Minimum Password Classes: 3

Saved Password History: 3

Minimum Password Lifetime (seconds): 0

Maximum Password Lifetime (seconds): 2,592,000

Principals Using This Policy: 0

Create New Policy- *CHANGES*

Example 25–13 Creating a New Kerberos Policy (Command Line)

In the following example, the `add_policy` command of `kadmin` is used to create the `build11` policy. This policy requires at least 3 character classes in a password.

```
$ kadmin
kadmin: add_policy -minclasses 3 build11
kadmin: quit
```

▼ How to Duplicate a Kerberos Policy

This procedure explains how to use all or some of the attributes of an existing policy to create a new policy. No command-line equivalent exists for this procedure.

1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Policies tab.**

3 **Select the policy in the list that you want to duplicate, then click Duplicate.**

The Policy Details panel is displayed. All the attributes of the selected policy are duplicated, except for the Policy Name field, which is empty.

4 **Specify a name for the duplicated policy in the Policy Name field.**

The policy name is mandatory. To make an exact duplicate of the policy you selected, skip to Step 6.

5 **Specify different values for the policy's attributes.**

Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 25–5 for all the policy attribute descriptions.

6 **Click Save to save the policy, or click Done.**

▼ How to Modify a Kerberos Policy

An example of the command-line equivalent follows this procedure.

1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for details.

```
$ /usr/sbin/gkadmin
```

2 **Click the Policies tab.**

3 **Select the policy in the list that you want to modify, then click Modify.**

The Policy Details panel is displayed.

4 **Modify the policy's attributes.**

Choose Context-Sensitive Help from the Help menu for information about the various attributes in this window. Or, go to Table 25–5 for all the policy attribute descriptions.

Note – You cannot modify a policy's name. To rename a policy, you must duplicate the policy, specify a new name for it, save it, and then delete the old policy.

---

5 **Click Save to save the policy, or click Done.**

Example 25–14 Modifying a Kerberos Policy (Command Line)

In the following example, the `modify_policy` command of `kadmin` is used to modify the minimum length of a password to five characters for the `build11` policy.

```
$ kadmin
kadmin: modify_policy -minlength 5 build11
kadmin: quit
```

▼ How to Delete a Kerberos Policy

An example of the command-line equivalent follows this procedure.

Note – Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to modify the principals' Policy attribute. The policy cannot be deleted if any principal is using it.

---

1 **If necessary, start the SEAM Tool.**

See "How to Start the SEAM Tool" on page 478 for more information.

```
$ /usr/sbin/gkadmin
```

2 **Click the Policies tab.**

3 **Select the policy in the list that you want to delete, then click Delete.**

After you confirm the deletion, the policy is deleted.

Example 25–15 Deleting a Kerberos Policy (Command Line)

In the following example, the `delete_policy` command of the `kadmin` command is used to delete the `build11` policy.

```
kadmin: delete_policy build11
Are you sure you want to delete the policy "build11"? (yes/no): yes
kadmin: quit
```

Before you delete a policy, you must cancel the policy from all principals that are currently using it. To do so, you need to use the `modify_principal -policy` command of `kadmin` on the affected principals. The `delete_policy` command fails if the policy is in use by a principal.

SEAM Tool Reference

This section provides descriptions of each panel in the SEAM Tool. Also, information about using limited privileges with SEAM Tool are provided.

SEAM Tool Panel Descriptions

This section provides descriptions for each principal and policy attribute that you can either specify or view in the SEAM Tool. The attributes are organized by the panel in which they are displayed.

TABLE 25–2 Attributes for the Principal Basics Panel of the SEAM Tool

Attribute	Description
Principal Name	The name of the principal (which is the *primary*/*instance* part of a fully qualified principal name). A principal is a unique identity to which the KDC can assign tickets. If you are modifying a principal, you cannot edit its name.
Password	The password for the principal. You can use the Generate Random Password button to create a random password for the principal.
Policy	A menu of available policies for the principal.
Account Expires	The date and time on which the principal's account expires. When the account expires, the principal can no longer get a ticket-granting ticket (TGT) and might be unable to log in.
Last Principal Change	The date on which information for the principal was last modified. (Read only)
Last Changed By	The name of the principal that last modified the account for this principal. (Read only)
Comments	Comments that are related to the principal (for example, “Temporary Account”).

TABLE 25–3 Attributes for the Principal Details Panel of the SEAM Tool

Attribute	Description
Last Success	The date and time when the principal last logged in successfully. (Read only)
Last Failure	The date and time when the last login failure for the principal occurred. (Read only)
Failure Count	The number of times a login failure has occurred for the principal. (Read only)

TABLE 25–3 Attributes for the Principal Details Panel of the SEAM Tool *(Continued)*

Attribute	Description
Last Password Change	The date and time when the principal's password was last changed. (Read only)
Password Expires	The date and time when the principal's current password expires.
Key Version	The key version number for the principal. This attribute is normally changed only when a password has been compromised.
Maximum Lifetime (seconds)	The maximum length of time for which a ticket can be granted for the principal (without renewal).
Maximum Renewal (seconds)	The maximum length of time for which an existing ticket can be renewed for the principal.

TABLE 25–4 Attributes of the Principal Flags Panel of the SEAM Tool

Attribute (Radio Buttons)	Description
Disable Account	When checked, prevents the principal from logging in. This attribute provides an easy way to temporarily freeze a principal account.
Require Password Change	When checked, expires the principal's current password, which forces the user to use the `kpasswd` command to create a new password. This attribute is useful if a security breach occurs, and you need to make sure that old passwords are replaced.
Allow Postdated Tickets	When checked, allows the principal to obtain postdated tickets. For example, you might need to use postdated tickets for `cron` jobs that must run after hours, but you cannot obtain tickets in advance because of short ticket lifetimes.
Allow Forwardable Tickets	When checked, allows the principal to obtain forwardable tickets. Forwardable tickets are tickets that are forwarded to the remote host to provide a single-sign-on session. For example, if you are using forwardable tickets and you authenticate yourself through `ftp` or `rsh`, then other services, such as NFS services, are available without your being prompted for another password.
Allow Renewable Tickets	When checked, allows the principal to obtain renewable tickets. A principal can automatically extend the expiration date or time of a ticket that is renewable (rather than having to get a new ticket after the first ticket expires). Currently, the NFS service is the ticket service that can renew tickets.
Allow Proxiable Tickets	When checked, allows the principal to obtain proxiable tickets. A proxiable ticket is a ticket that can be used by a service on behalf of a client to perform an operation for the client. With a proxiable ticket, a service can take on the identity of a client and obtain a ticket for another service. However, the service cannot obtain a ticket-granting ticket (TGT).

TABLE 25–4 Attributes of the Principal Flags Panel of the SEAM Tool *(Continued)*

Attribute (Radio Buttons)	Description
Allow Service Tickets	When checked, allows service tickets to be issued for the principal. You should not allow service tickets to be issued for the `kadmin/`*hostname* and `changepw/`*hostname* principals. This practice ensures that only these principals can update the KDC database.
Allow TGT-Based Authentication	When checked, allows the service principal to provide services to another principal. More specifically, this attribute allows the KDC to issue a service ticket for the service principal. This attribute is valid only for service principals. When unchecked, service tickets cannot be issued for the service principal.
Allow Duplicate Authentication	When checked, allows the user principal to obtain service tickets for other user principals. This attribute is valid only for user principals. When unchecked, the user principal can still obtain service tickets for service principals, but not for other user principals.
Required Preauthentication	When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through software) that the principal is really the principal that is requesting the TGT. This preauthentication is usually done through an extra password, for example, from a DES card. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.
Required Hardware Authentication	When checked, the KDC will not send a requested ticket-granting ticket (TGT) to the principal until the KDC can authenticate (through hardware) that the principal is really the principal that is requesting the TGT. Hardware preauthentication can occur, for example, on a Java ring reader. When unchecked, the KDC does not need to preauthenticate the principal before the KDC sends a requested TGT to the principal.

TABLE 25–5 Attributes for the Policy Basics Pane of the SEAM Tool

Attribute	Description
Policy Name	The name of the policy. A policy is a set of rules that govern a principal's password and tickets. If you are modifying a policy, you cannot edit its name.
Minimum Password Length	The minimum length for the principal's password.
Minimum Password Classes	The minimum number of different character types that are required in the principal's password. For example, a minimum classes value of 2 means that the password must have at least two different character types, such as letters and numbers (hi2mom). A value of 3 means that the password must have at least three different character types, such as letters, numbers, and punctuation (hi2mom!). And so on. A value of 1 sets no restriction on the number of password character types.

TABLE 25–5 Attributes for the Policy Basics Pane of the SEAM Tool *(Continued)*

Attribute	Description
Saved Password History	The number of previous passwords that have been used by the principal, and a list of the previous passwords that cannot be reused.
Minimum Password Lifetime (seconds)	The minimum length of time that the password must be used before it can be changed.
Maximum Password Lifetime (seconds)	The maximum length of time that the password can be used before it must be changed.
Principals Using This Policy	The number of principals to which this policy currently applies. (Read only)

Using the SEAM Tool With Limited Kerberos Administration Privileges

All features of the SEAM Administration Tool are available if your `admin` principal has all the privileges to administer the Kerberos database. However, you might have limited privileges, such as only being allowed to view the list of principals or to change a principal's password. With limited Kerberos administration privileges, you can still use the SEAM Tool. However, various parts of the SEAM Tool change based on the Kerberos administration privileges that you do not have. Table 25–6 shows how the SEAM Tool changes based on your Kerberos administration privileges.

The most visual change to the SEAM Tool occurs when you don't have the list privilege. Without the list privilege, the List panels do not display the list of principals and polices for you to manipulate. Instead, you must use the Name field in the List panels to specify a principal or a policy that you want to manipulate.

If you log in to the SEAM Tool, and you do not have sufficient privileges to perform tasks with it, the following message displays and you are sent back to the SEAM Administration Login window:

```
Insufficient privileges to use gkadmin: ADMCIL. Please try using another principal.
```

To change the privileges for a principal so that it can administer the Kerberos database, go to “How to Modify the Kerberos Administration Privileges” on page 491.

TABLE 25–6 Using the SEAM Tool With Limited Kerberos Administration Privileges

Disallowed Privilege	How the SEAM Tool Changes
a (add)	The Create New and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the add privilege, you cannot create new principals or policies, or duplicate them.

TABLE 25–6 Using the SEAM Tool With Limited Kerberos Administration Privileges *(Continued)*

Disallowed Privilege	How the SEAM Tool Changes
`d` (delete)	The Delete button is unavailable in the Principal List and Policy List panels. Without the delete privilege, you cannot delete principals or policies.
`m` (modify)	The Modify button is unavailable in the Principal List and Policy List panels. Without the modify privilege, you cannot modify principals or policies. Also, with the Modify button unavailable, you cannot modify a principal's password, even if you have the change password privilege.
`c` (change password)	The Password field in the Principal Basics panel is read only and cannot be changed. Without the change password privilege, you cannot modify a principal's password. Note that even if you have the change password privilege, you must also have the modify privilege to change a principal's password.
`i` (inquiry to database)	The Modify and Duplicate buttons are unavailable in the Principal List and Policy List panels. Without the inquiry privilege, you cannot modify or duplicate a principal or a policy. Also, with the Modify button unavailable, you cannot modify a principal's password, even if you have the change password privilege.
`l` (list)	The list of principals and policies in the List panels are unavailable. Without the list privilege, you must use the Name field in the List panels to specify the principal or the policy that you want to manipulate.

Administering Keytab Files

Every host that provides a service must have a local file, called a *keytab* (short for “key table”). The keytab contains the principal for the appropriate service, called a *service key*. A service key is used by a service to authenticate itself to the KDC and is known only by Kerberos and the service itself. For example, if you have a Kerberized NFS server, that server must have a keytab file that contains its `nfs` service principal.

To add a service key to a keytab file, you add the appropriate service principal to a host's keytab file by using the `ktadd` command of `kadmin`. Because you are adding a service principal to a keytab file, the principal must already exist in the Kerberos database so that `kadmin` can verify its existence. On the master KDC, the keytab file is located at `/etc/krb5/kadm5.keytab`, by default. On application servers that provide Kerberized services, the keytab file is located at `/etc/krb5/krb5.keytab`, by default.

A keytab is analogous to a user's password. Just as it is important for users to protect their passwords, it is equally important for application servers to protect their keytab files. You should always store keytab files on a local disk, and make them readable only by the `root` user. Also, you should never send a keytab file over an unsecured network.

There is also a special instance in which to add a `root` principal to a host's keytab file. If you want a user on the Kerberos client to mount Kerberized NFS file systems that require root-equivalent access, you must add the client's `root` principal to the client's keytab file. Otherwise, users must use the `kinit` command as `root` to obtain credentials for the client's `root` principal whenever they want to mount a Kerberized NFS file system with `root` access, even when they are using the automounter.

Note – When you set up a master KDC, you need to add the `kadmind` and `changepw` principals to the `kadm5.keytab` file.

Another command that you can use to administer keytab files is the `ktutil` command. This interactive command enables you to manage a local host's keytab file without having Kerberos administration privileges, because `ktutil` doesn't interact with the Kerberos database as `kadmin` does. So, after a principal is added to a keytab file, you can use `ktutil` to view the keylist in a keytab file or to temporarily disable authentication for a service.

Note – When you change a principal in a keytab file using the `ktadd` command in `kadmin`, a new key is generated and added to the keytab file.

Administering Keytab Files (Task Map)

Task	Description	For Instructions
Add a service principal to a keytab file.	Use the `ktadd` command of `kadmin` to add a service principal to a keytab file.	"How to Add a Kerberos Service Principal to a Keytab File" on page 507
Remove a service principal from a keytab file.	Use the `ktremove` command of `kadmin` to remove a service from a keytab file.	"How to Remove a Service Principal From a Keytab File" on page 509
Display the keylist (list of principals) in a keytab file.	Use the `ktutil` command to display the keylist in a keytab file.	"How to Display the Keylist (Principals) in a Keytab File" on page 510

Task	Description	For Instructions
Temporarily disable authentication for a service on a host.	This procedure is a quick way to temporarily disable authentication for a service on a host without requiring `kadmin` privileges. Before you use `ktutil` to delete the service principal from the server's keytab file, copy the original keytab file to a temporary location. When you want to enable the service again, copy the original keytab file back to its proper location.	"How to Temporarily Disable Authentication for a Service on a Host" on page 510

▼ How to Add a Kerberos Service Principal to a Keytab File

1. **Make sure that the principal already exists in the Kerberos database.**

   See "How to View the List of Kerberos Principals" on page 482 for more information.

2. **Become superuser on the host that needs a principal added to its keytab file.**

3. **Start the `kadmin` command.**

   ```
   # /usr/sbin/kadmin
   ```

4. **Add a principal to a keytab file by using the `ktadd` command.**

   ```
   kadmin: ktadd [-e enctype] [-k keytab] [-q] [principal | -glob principal-exp]
   ```

`-e` *enctype*	Overrides the list of encryption types defined in the `krb5.conf` file.
`-k` *keytab*	Specifies the keytab file. By default, `/etc/krb5/krb5.keytab` is used.
`-q`	Displays less verbose information.
principal	Specifies the principal to be added to the keytab file. You can add the following service principals: `host`, `root`, `nfs`, and `ftp`.
`-glob` *principal-exp*	Specifies the principal expressions. All principals that match the *principal-exp* are added to the keytab file. The rules for principal expression are the same as for the `list_principals` command of `kadmin`.

5. **Quit the `kadmin` command.**

   ```
   kadmin: quit
   ```

Example 25–16 Adding a Service Principal to a Keytab File

In the following example, the `kadmin/kdc1.example.com` and `changepw/kdc1.example.com` principals are added to a master KDC's keytab file. For this example, the keytab file must be the file that is specified in the `kdc.conf` file.

```
kdc1 # /usr/sbin/kadmin.local
kadmin.local: ktadd -k /etc/krb5/kadm5.keytab kadmin/kdc1.example.com changepw/kdc1.example.com
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-256 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal kadmin/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.exmaple.com with kvno 3, encryption type AES-256 CTS
          mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.exmaple.com with kvno 3, encryption type AES-128 CTS
          mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type Triple DES cbc
          mode with HMAC/sha1 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal changepw/kdc1.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/kadm5.keytab.
kadmin.local: quit
```

In the following example, `denver`'s `host` principal is added to `denver`'s keytab file, so that the KDC can authenticate `denver`'s network services.

```
denver # /usr/sbin/kadmin
kadmin: ktadd host/denver.example.com
Entry for principal host/denver.exmaple.com with kvno 3, encryption type AES-256 CTS
          mode with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/kadm5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type AES-128 CTS mode
          with 96-bit SHA-1 HMAC added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type Triple DES cbc mode
          with HMAC/sha1 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type ArcFour
          with HMAC/md5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
Entry for principal host/denver.example.com with kvno 3, encryption type DES cbc mode
          with RSA-MD5 added to keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ How to Remove a Service Principal From a Keytab File

1 **Become superuser on the host with a service principal that must be removed from its keytab file.**

2 **Start the `kadmin` command.**

```
# /usr/sbin/kadmin
```

3 **(Optional) To display the current list of principals (keys) in the keytab file, use the `ktutil` command.**

See "How to Display the Keylist (Principals) in a Keytab File" on page 510 for detailed instructions.

4 **Remove a principal from the keytab file by using the `ktremove` command.**

```
kadmin: ktremove [-k keytab] [-q] principal [kvno | all | old ]
```

`-k` *keytab*	Specifies the keytab file. By default, `/etc/krb5/krb5.keytab` is used.
`-q`	Displays less verbose information.
principal	Specifies the principal to be removed from the keytab file.
kvno	Removes all entries for the specified principal whose key version number matches *kvno*.
`all`	Removes all entries for the specified principal.
`old`	Removes all entries for the specified principal, except those principals with the highest key version number.

5 **Quit the `kadmin` command.**

```
kadmin: quit
```

Example 25–17 Removing a Service Principal From a Keytab File

In the following example, `denver`'s `host` principal is removed from `denver`'s keytab file.

```
denver # /usr/sbin/kadmin
kadmin: ktremove host/denver.example.com@EXAMPLE.COM
kadmin: Entry for principal host/denver.example.com@EXAMPLE.COM with kvno 3
  removed from keytab WRFILE:/etc/krb5/krb5.keytab.
kadmin: quit
```

▼ How to Display the Keylist (Principals) in a Keytab File

1. **Become superuser on the host with the keytab file.**

   Note – Although you can create keytab files that are owned by other users, using the default location for the keytab file requires `root` ownership.

   ---

2. **Start the `ktutil` command.**

   ```
   # /usr/bin/ktutil
   ```

3. **Read the keytab file into the keylist buffer by using the `read_kt` command.**

   ```
   ktutil: read_kt keytab
   ```

4. **Display the keylist buffer by using the `list` command.**

   ```
   ktutil: list
   ```

   The current keylist buffer is displayed.

5. **Quit the `ktutil` command.**

   ```
   ktutil: quit
   ```

Example 25–18 Displaying the Keylist (Principals) in a Keytab File

The following example displays the keylist in the `/etc/krb5/krb5.keytab` file on the `denver` host.

```
denver # /usr/bin/ktutil
    ktutil: read_kt /etc/krb5/krb5.keytab
    ktutil: list
slot KVNO Principal
---- ---- ---------------------------------------
   1    5 host/denver@EXAMPLE.COM
    ktutil: quit
```

▼ How to Temporarily Disable Authentication for a Service on a Host

At times, you might need to temporarily disable the authentication mechanism for a service, such as `rlogin` or `ftp`, on a network application server. For example, you might want to stop users from logging in to a system while you are performing maintenance procedures. The `ktutil` command enables you to accomplish this task by removing the service principal from the server's keytab file, without requiring `kadmin` privileges. To enable authentication again, you just need to copy the original keytab file that you saved back to its original location.

Note – By default, most services are set up to require authentication. If a service is not set up to require authentication, then the service still works, even if you disable authentication for the service.

---

1 **Become superuser on the host with the keytab file.**

Note – Although you can create keytab files that are owned by other users, using the default location for the keytab file requires `root` ownership.

---

2 **Save the current keytab file to a temporary file.**

3 **Start the `ktutil` command.**

```
# /usr/bin/ktutil
```

4 **Read the keytab file into the keylist buffer by using the `read_kt` command.**

```
ktutil: read_kt keytab
```

5 **Display the keylist buffer by using the `list` command.**

```
ktutil: list
```

The current keylist buffer is displayed. Note the slot number for the service that you want to disable.

6 **To temporarily disable a host's service, remove the specific service principal from the keylist buffer by using the `delete_entry` command.**

```
ktutil: delete_entry slot-number
```

Where *slot-number* specifies the slot number of the service principal to be deleted, which is displayed by the `list` command.

7 **Write the keylist buffer to a new keytab file by using the `write_kt` command.**

```
ktutil: write_kt new-keytab
```

8 **Quit the `ktutil` command.**

```
ktutil: quit
```

9 **Move the new keytab file.**

```
# mv new-keytab keytab
```

10 **When you want to re-enable the service, copy the temporary (original) keytab file back to its original location.**

Example 25–19 Temporarily Disabling a Service on a Host

In the following example, the `host` service on the `denver` host is temporarily disabled. To re-enable the host service on `denver`, you would copy the `krb5.keytab.temp` file to the `/etc/krb5/krb5.keytab` file.

```
denver # cp /etc/krb5/krb5.keytab /etc/krb5/krb5.keytab.temp
denver # /usr/bin/ktutil
    ktutil:read_kt /etc/krb5/krb5.keytab
    ktutil:list
slot KVNO Principal
---- ---- ---------------------------------------
   1    8 root/denver@EXAMPLE.COM
   2    5 host/denver@EXAMPLE.COM
    ktutil:delete_entry 2
    ktutil:list
slot KVNO Principal
---- ---- --------------------------------------
   1    8 root/denver@EXAMPLE.COM
    ktutil:write_kt /etc/krb5/new.krb5.keytab
    ktutil: quit
denver # cp /etc/krb5/new.krb5.keytab /etc/krb5/krb5.keytab
```

◆ ◆ ◆ CHAPTER 26

Using Kerberos Applications (Tasks)

This chapter is intended for anyone on a system with the Kerberos service configured on it. This chapter explains how to use the “Kerberized” commands and services that are provided. You should already be familiar with these commands (in their non-Kerberized versions) before you read about them here.

Because this chapter is intended for the general reader, it includes information on tickets: obtaining, viewing, and destroying them. This chapter also includes information on choosing or changing a Kerberos password.

This is a list of the information in this chapter:

- “Kerberos Ticket Management” on page 513
- “Kerberos Password Management” on page 517
- “Kerberos User Commands” on page 522

For an overview of the Solaris Kerberos product, see Chapter 21, “Introduction to the Kerberos Service.”

Kerberos Ticket Management

This section explains how to obtain, view, and destroy tickets. For an introduction to tickets, see “How the Kerberos Service Works” on page 364.

Do You Need to Worry About Tickets?

With any of the SEAM releases or the Solaris 10 release installed, Kerberos is built into the `login` command, and you will obtain tickets automatically when you log in. The Kerberized commands `rsh`, `rcp`, `rdist`, `telnet`, and `rlogin` are usually set up to forward copies of your tickets to the other machines, so you don't have to explicitly ask for tickets to get access to those machines. Your configuration might not include this automatic forwarding, but it is the default behavior. See “Overview of Kerberized Commands” on page 522 and “Forwarding Kerberos Tickets” on page 525 for more information on forwarding tickets.

For information on ticket lifetimes, see “Ticket Lifetimes” on page 535.

Creating a Kerberos Ticket

Normally, if PAM is configured properly, a ticket is created automatically when you log in, and you need not do anything special to obtain a ticket. However, you might need to create a ticket if your ticket expires. Also, you might need to use a different principal besides your default principal, for example, if you use `rlogin -l` to log in to a machine as someone else.

To create a ticket, use the `kinit` command.

```
% /usr/bin/kinit
```

The `kinit` command prompts you for your password. For the full syntax of the `kinit` command, see the `kinit`(1) man page.

EXAMPLE 26–1 Creating a Kerberos Ticket

This example shows a user, `jennifer`, creating a ticket on her own system.

```
% kinit
Password for jennifer@ENG.EXAMPLE.COM:      <Type password>
```

Here, the user `david` creates a ticket that is valid for three hours with the `-l` option.

```
% kinit -l 3h david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:      <Type password>
```

This example shows the user `david` creating a forwardable ticket (with the `-f` option) for himself. With this forwardable ticket, he can, for example, log in to a second system, and then `telnet` to a third system.

```
% kinit -f david@EXAMPLE.ORG
Password for david@EXAMPLE.ORG:         <Type password>
```

For more information on how forwarding tickets works, see “Forwarding Kerberos Tickets” on page 525 and “Types of Tickets” on page 534.

Viewing Kerberos Tickets

Not all tickets are alike. One ticket might, for example, be *forwardable*. Another ticket might be *postdated*. While a third ticket might be both forwardable and postdated. You can see which tickets you have, and what their attributes are, by using the `klist` command with the `-f` option:

```
% /usr/bin/klist -f
```

The following symbols indicate the attributes that are associated with each ticket, as displayed by `klist`:

A	Preauthenticated
D	Postdatable
d	Postdated
F	Forwardable
f	Forwarded
I	Initial
i	Invalid
P	Proxiable
p	Proxy
R	Renewable

"Types of Tickets" on page 534 describes the various attributes that a ticket can have.

EXAMPLE 26–2 Viewing Kerberos Tickets

This example shows that the user `jennifer` has an *initial* ticket, which is *forwardable* (F) and *postdated* (d), but not yet validated (i).

```
% /usr/bin/klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: jennifer@ENG.EXAMPLE.COM

Valid starting                 Expires                 Service principal
09 Mar 04 15:09:51  09 Mar 04 21:09:51  nfs/EXAMPLE.SUN.COM@EXAMPLE.SUN.COM
        renew until 10 Mar 04 15:12:51, Flags: Fdi
```

The following example shows that the user `david` has two tickets that were *forwarded* (f) to his host from another host. The tickets are also *forwardable* (F).

EXAMPLE 26–2 Viewing Kerberos Tickets *(Continued)*

```
% klist -f
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting                 Expires                 Service principal
07 Mar 04 06:09:51  09 Mar 04 23:33:51  host/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 17:09:51, Flags: fF

Valid starting                 Expires                 Service principal
08 Mar 04 08:09:51  09 Mar 04 12:54:51  nfs/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 15:22:51, Flags: fF
```

The following example shows how to display the encryption types of the session key and the ticket by using the `-e` option. The `-a` option is used to map the host address to a host name if the name service can do the conversion.

```
% klist -fea
Ticket cache: /tmp/krb5cc_74287
Default principal: david@EXAMPLE.SUN.COM

Valid starting                 Expires                 Service principal
07 Mar 04 06:09:51  09 Mar 04 23:33:51  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 10 Mar 04 17:09:51, Flags: FRIA
        Etype(skey, tkt): DES cbc mode with RSA-MD5, DES cbc mode with CRC-32
        Addresses: client.example.com
```

Destroying Kerberos Tickets

If you want to destroy all Kerberos tickets acquired during your current session, use the `kdestroy` command. The command destroys you credential cache, which destroys all your credentials and tickets. While this is not usually necessary, running `kdestroy` reduces the chance of the credential cache being compromised during times that you are not logged in.

To destroy your tickets, use the `kdestroy` command.

```
% /usr/bin/kdestroy
```

The `kdestroy` command destroys *all* your tickets. You cannot use this command to selectively destroy a particular ticket.

If you are going to be away from your system and are concerned about an intruder using your permissions, you should use either `kdestroy` or a screen saver that locks the screen.

Kerberos Password Management

With the Kerberos service configured, you now have two passwords: your regular Solaris password and a Kerberos password. You can make both passwords the same, or they can be different.

Advice on Choosing a Password

Your password can include almost any character that you can type. The main exceptions are the Control keys and the Return key. A good password is a password that you can remember readily, but no one else can easily guess. Examples of bad passwords include the following:

- Words that can be found in a dictionary
- Any common or popular name
- The name of a famous person or character
- Your name or user name in any form (for example: your name spelled backward, repeated twice, and so forth)
- A spouse's name, child's name, or pet's name
- Your birth date or a relative's birth date
- Your social security number, driver's license number, passport number, or other similar identifying number
- Any sample password that appears in this manual or any other manual

A good password is at least eight characters long. Moreover, a password should include a mix of characters, such as uppercase and lowercase letters, numbers, and punctuation marks. Examples of passwords that would be good if they didn't appear in this manual include the following:

- Acronyms, such as “I2LMHinSF” (which is recalled as “I too left my heart in San Francisco”)
- Easy-to-pronounce nonsense words, such as “WumpaBun” or “WangDangdoodle!”
- Deliberately misspelled phrases, such as “6o'cluck” or “RrriotGrrrlsRrrule!”

Caution – Don't use these examples. Passwords that appear in manuals are the first passwords that an intruder will try.

Changing Your Password

If PAM is properly configured, you can change your Kerberos password in two ways:

- With the usual UNIX `passwd` command. With the Kerberos service configured, the Solaris `passwd` command also automatically prompts for a new Kerberos password.

  The advantage of using `passwd` instead of `kpasswd` is that you can set both UNIX and Kerberos passwords at the same time. However, you generally do not *have* to change both passwords with `passwd`. Often, you can change only your UNIX password and leave the Kerberos password untouched, or vice-versa.

  Note – The behavior of `passwd` depends on how the PAM module is configured. You might be required to change both passwords in some configurations. For some sites, the UNIX password must be changed, while other sites require the Kerberos password to change.

- With the `kpasswd` command. `kpasswd` is very similar to `passwd`. One difference is that `kpasswd` changes only Kerberos passwords. You must use `passwd` if you want to change your UNIX password.

  Another difference is that `kpasswd` can change a password for a Kerberos principal that is not a valid UNIX user. For example, `david/admin` is a Kerberos principal, but not an actual UNIX user, so you must use `kpasswd` instead of `passwd`.

After you change your password, it takes some time for the change to propagate through a system (especially over a large network). Depending on how your system is set up, this delay might take anywhere from a few minutes to an hour or more. If you need to get new Kerberos tickets shortly after you change your password, try the new password first. If the new password doesn't work, try again using the old password.

Kerberos V5 protocol enables system administrators to set criteria about allowable passwords for each user. Such criteria is defined by the *policy* set for each user (or by a default policy). See "Administering Kerberos Policies" on page 493 for more on policies.

For example, suppose that user `jennifer`'s policy (call it `jenpol`) mandates that passwords be at least eight letters long and include a mix of at least two types of characters. `kpasswd` will therefore reject an attempt to use "sloth" as a password.

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:        <Jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM’s password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:     <Jennifer types 'sloth'>
New password (again):      <Jennifer re-types 'sloth'>
kpasswd: New password is too short.
Please choose a password which is at least 4 characters long.
```

Here, `jennifer` uses “slothrop49” as a password. “slothrop49” meets the criteria, because it is over eight letters long and contains two different types of characters (numbers and lowercase letters).

```
% kpasswd
kpasswd: Changing password for jennifer@ENG.EXAMPLE.COM.
Old password:      <Jennifer types her existing password>
kpasswd: jennifer@ENG.EXAMPLE.COM’s password is controlled by
the policy jenpol
which requires a minimum of 8 characters from at least 2 classes
(the five classes are lowercase, uppercase, numbers, punctuation,
and all other characters).
New password:      <Jennifer types 'slothrop49'>
New password (again):      <Jennifer re-types 'slothrop49'>
Kerberos password changed.
```

EXAMPLE 26–3 Changing Your Password

In the following example, user `david` changes both his UNIX password and Kerberos password with `passwd`.

```
% passwd
    passwd:  Changing password for david
    Enter login (NIS+) password:             <Type the current UNIX password>
    New password:                            <Type the new UNIX password>
    Re-enter password:                       <Confirm the new UNIX password>
    Old KRB5 password:                       <Type the current Kerberos password>
    New KRB5 password:                       <Type the new Kerberos password>
    Re-enter new KRB5 password:              <Confirm the new Kerberos password>
```

Note that `passwd` asks for both the UNIX password and the Kerberos password. This behavior is established by the default configuration. In that case, user `david` must use `kpasswd` to set his Kerberos password to something else, as shown next.

This example shows user `david` changing only his Kerberos password with `kpasswd`.

```
% kpasswd
kpasswd: Changing password for david@ENG.EXAMPLE.COM.
Old password:              <Type the current Kerberos password>
New password:              <Type the new Kerberos password>
New password (again):      <Confirm the new Kerberos password>
```

EXAMPLE 26–3 Changing Your Password *(Continued)*

```
Kerberos password changed.
```

In this example, user `david` changes the password for the Kerberos principal `david/admin` (which is not a valid UNIX user). He must use `kpasswd`.

```
% kpasswd david/admin
kpasswd:  Changing password for david/admin.
Old password:              <Type the current Kerberos password>
New password:              <Type the new Kerberos password>
New password (again):      <Type the new Kerberos password>
Kerberos password changed.
```

Granting Access to Your Account

If you need to give someone access to log in to your account (as you), you can do so through Kerberos, without revealing your password, by putting a `.k5login` file in your home directory. A `.k5login` file is a list of one or more Kerberos principals corresponding to each person for whom you want to grant access. Each principal must be on a separate line.

Suppose that the user `david` keeps a `.k5login` file in his home directory that looks like the following:

```
jennifer@ENG.EXAMPLE.COM
joe@EXAMPLE.ORG
```

This file allows the users `jennifer` and `joe` to assume `david`'s identity, provided that they already have Kerberos tickets in their respective realms. For example, jennifer can remotely log in to `david`'s machine (`boston`), as him, without having to give his password.

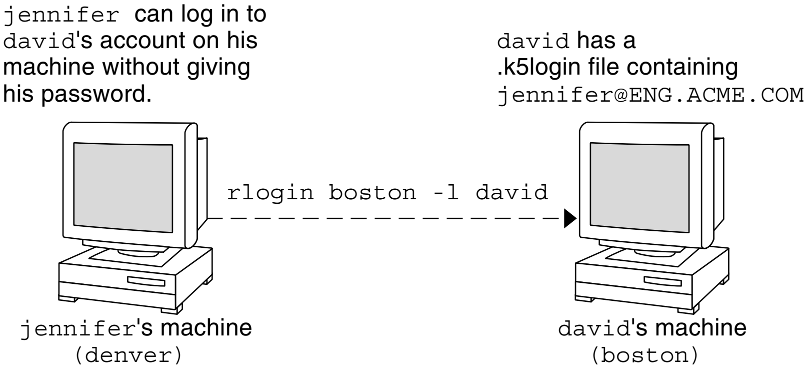

jennifer can log in to david's account on his machine without giving his password.
david has a .k5login file containing jennifer@ENG.ACME.COM
rlogin boston -l david
jennifer's machine (denver)
david's machine (boston)

FIGURE 26–1 Using the `.k5login` File to Grant Access to Your Account

In the case where `david`'s home directory is NFS-mounted, using Kerberos V5 protocols, from another (third) machine, `jennifer` must have a forwardable ticket in order to access his home directory. See "Creating a Kerberos Ticket" on page 514 for an example of using a forwardable ticket.

If you will be logging in to other machines across a network, you'll want to include your own Kerberos principal in `.k5login` files on those machines.

Using a `.k5login` file is much safer than giving out your password for these reasons:

- You can take access away any time by removing the principal from your `.k5login` file.
- Although users principals named in the `.k5login` file in your home directory have full access to your account on that machine (or sets of machines, if the `.k5login` file is shared, for example, over NFS). However, any Kerberized services will authorize access based on that user's identity, not yours. So `jennifer` can log in to `joe`'s machine and perform tasks there. However, if she uses a Kerberized program such as `ftp` or `rlogin`, she does so as herself.
- Kerberos keeps a log of who obtains tickets, so a system administrator can find out, if necessary, who is capable of using your user identity at a particular time.

One common way to use the `.k5login` file is to put it in `root`'s home directory, giving `root` access for that machine to the Kerberos principals listed. This configuration allows system administrators to become `root` locally, or to log in remotely as `root`, without having to give out the `root` password, and without requiring anyone to type the `root` password over the network.

EXAMPLE 26–4 Using the `.k5login` File to Grant Access to Your Account

Suppose `jennifer` decides to log in to the machine `boston.example.com` as `root`. Because she has an entry for her principal name in the `.k5login` file in `root`'s home directory on `boston.example.com`, she again does not have to type in her password.

EXAMPLE 26–4 Using the `.k5login` File to Grant Access to Your Account *(Continued)*

```
% rlogin boston.example.com -l root -x
This rlogin session is using DES encryption for all data transmissions.
Last login: Thu Jun 20 16:20:50 from daffodil
SunOS Release 5.7 (GENERIC) #2: Tue Nov 14 18:09:31 EST 1998
boston[root]%
```

Kerberos User Commands

Kerberos V5 product is a *single-sign-on* system, which means that you only have to type your password once. The Kerberos V5 programs do the authenticating (and optional encrypting) for you, because Kerberos has been built into each of a suite of existing, familiar network programs. The Kerberos V5 applications are versions of existing UNIX network programs with Kerberos features added.

For example, when you use a Kerberized program to connect to a remote host, the program, the KDC, and the remote host perform a set of rapid negotiations. When these negotiations are completed, your program has proven your identity on your behalf to the remote host, and the remote host has granted you access.

Note that Kerberized commands try to authenticate with Kerberos first. If Kerberos authentication fails, an error occurs or UNIX authentication is attempted, depending on what options were used with the command. Refer to the `Kerberos Security` section in each Kerberos command man page for more detailed information.

Overview of Kerberized Commands

The Kerberized network services are programs that connect to another machine somewhere on the Internet. These programs are the following:

- `ftp`
- `rcp`
- `rdist`
- `rlogin`
- `rsh`
- `ssh`
- `telnet`

These programs have features that transparently use your Kerberos tickets for negotiating authentication and optional encryption with the remote host. In most cases, you'll notice only that you no longer have to type your password to use them, because Kerberos will provide proof of your identity for you.

The Kerberos V5 network programs include options that enable you to do the following:

- Forward your tickets to the another host (if you initially obtained forwardable tickets).
- Encrypt data transmitted between you and the remote host.

Note – This section assumes you are already familiar with the non-Kerberos versions of these programs, and highlights the Kerberos functionality added by the Kerberos V5 package. For detailed descriptions of the commands described here, see their respective man pages.

The following Kerberos options have been added to `ftp`, `rcp`, `rlogin`, `rsh`, and `telnet`:

`-a`	Attempts automatic login using your existing tickets. Uses the username as returned by `getlogin()`, unless the name is different from the current user ID. See the `telnet(1)` man page for details.
`-f`	Forwards a *non-reforwardable* ticket to a remote host. This option is mutually exclusive with the `-F` option. They cannot be used together in the same command. You'll want to forward a ticket if you have reason to believe you'll need to authenticate yourself to other Kerberos-based services on a third host. For example, you might want to remotely log in to another machine and then remotely log in from it to a third machine. You should definitely use a forwardable ticket if your home directory on the remote host is NFS-mounted using the Kerberos V5 mechanism. Otherwise, you won't be able to access your home directory. That is, suppose you initially log in to System 1. From System 1, you remotely log in to your home machine, System 2, which mounts your home directory from System 3. Unless you've used the `-f` or `-F` option with `rlogin`, you won't be able to get to your home directory because your ticket can't be forwarded to System 3. By default, `kinit` obtains forwardable ticket-granting tickets (TGTs). However, your configuration might differ in this respect. For more information on forwarding tickets, see "Forwarding Kerberos Tickets" on page 525.
`-F`	Forwards a *reforwardable* copy of your TGT to a remote system. It is similar to `-f`, but it allows for access to a further (say, fourth or fifth) machine. The `-F` option can therefore be regarded as being a superset of the `-f` option. The `-F` option is mutually exclusive with the `-f` option. They cannot be used together in the same command.

	For more information on forwarding tickets, see "Forwarding Kerberos Tickets" on page 525.
`-k` *realm*	Requests tickets for the remote host in the specified *realm*, instead of determining the realm itself using the `krb5.conf` file.
`-K`	Uses your tickets to authenticate to the remote host, but does not automatically log in.
`-m` *mechanism*	Specifies the GSS-API security mechanism to use, as listed in the `/etc/gss/mech` file. Defaults to `kerberos_v5`.
`-x`	Encrypts this session.
`-X` *auth-type*	Disables the *auth-type* type of authentication.

The following table shows which commands have specific options. An "X" indicates that the command has that option.

TABLE 26–1 Kerberos Options for Network Commands

	`ftp`	`rcp`	`rlogin`	`rsh`	`telnet`
`-a`					X
`-f`	X		X	X	X
`-F`			X	X	X
`-k`		X	X	X	X
`-K`					X
`-m`	X				
`-x`	X	X	X	X	X
`-X`					X

Additionally, `ftp` allows the protection level for a session to be set at its prompt:

clear	Sets the protection level to "clear" (no protection). This protection level is the default.
private	Sets the protection level to "private." Data transmissions are confidentiality-protected and integrity-protected by encryption. The privacy service might not be available to all Kerberos users, however.
safe	Sets the protection level to "safe." Data transmissions are integrity-protected by cryptographic checksum.

You can also set the protection level at the `ftp` prompt by typing `protect` followed by any of the protection levels shown above (`clear`, `private`, or `safe`).

Forwarding Kerberos Tickets

As described in "Overview of Kerberized Commands" on page 522, some commands allow you to forward tickets with either the `-f` or `-F` option. Forwarding tickets allows you to "chain" your network transactions. You can, for example, remotely log in to one machine and then remotely log in from it to another machine. The `-f` option allows you to forward a ticket, while the `-F` option allows you to reforward a forwarded ticket.

In Figure 26–2, the user `david` obtains a non-forwardable ticket-granting ticket (TGT) with `kinit`. The ticket is non-forwardable because he did not specify the `-f` option. In scenario 1, he is able to remotely log in to machine B, but he can go no further. In scenario 2, the `rlogin -f` command fails because he is attempting to forward a ticket that is non-forwardable.

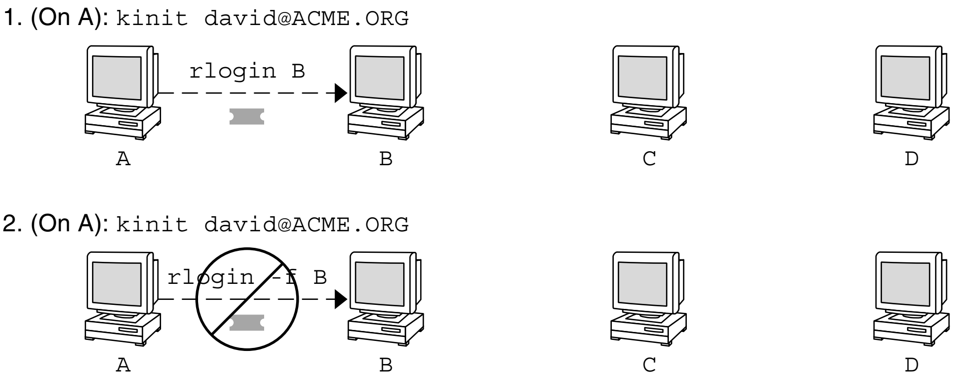

1. (On A): kinit david@ACME.ORG
rlogin B
A
B
C
D
2. (On A): kinit david@ACME.ORG
rlogin -f B
A
B
C
D

FIGURE 26–2 Using Non-Forwardable Tickets

In actuality, Kerberos configuration files are set up so that `kinit` obtains forwardable tickets by default. However, your configuration might differ. For the sake of explanation, assume that `kinit` does *not* obtain forwardable TGTs unless it is invoked with `kinit -f`. Notice, by the way, that `kinit` does not have a `-F` option. TGTs are either forwardable or not.

In Figure 26–3, the user `david` obtains forwardable TGTs with `kinit -f`. In scenario 3, he is able to reach machine C because he uses a forwardable ticket with `rlogin`. In scenario 4, the second `rlogin` fails because the ticket is not reforwardable. By using the `-F` option instead, as in scenario 5, the second `rlogin` succeeds and the ticket can be reforwarded on to machine D.

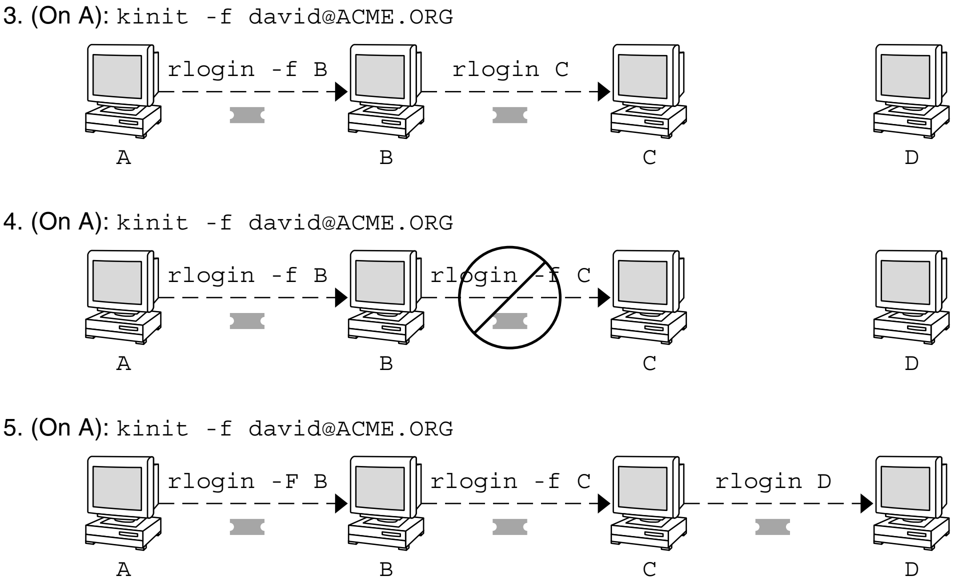

3. (On A): kinit -f david@ACME.ORG
rlogin -f B
rlogin C
A
B
C
D
4. (On A): kinit -f david@ACME.ORG
rlogin -f B
rlogin -f C
A
B
C
D
5. (On A): kinit -f david@ACME.ORG
rlogin -F B
rlogin -f C
rlogin D
A
B
C
D

FIGURE 26–3 Using Forwardable Tickets

Using Kerberized Commands (Examples)

The following examples show how the options to the Kerberized commands work.

EXAMPLE 26–5 Using the `-a`, `-f`, and `-x` Options With `telnet`

In this example, the user `david` has already logged in, and wants to `telnet` to the machine `denver.example.com`. He uses the `-f` option to forward his existing tickets, the `-x` option to encrypt the session, and the `-a` option to perform the login automatically. Because he does not plan to use the services of a third host, he can use `-f` instead of `-F`.

```
% telnet -a -f -x denver.example.com
Trying 128.0.0.5...
Connected to denver.example.com. Escape character is ’^]’.
[ Kerberos V5 accepts you as "david@eng.example.com" ]
[ Kerberos V5 accepted forwarded credentials ]
SunOS 5.9: Tue May 21 00:31:42 EDT 2004  Welcome to SunOS
%
```

Notice that `david`'s machine used Kerberos to authenticate him to `denver.example.com`, and logged him in automatically as himself. He had an encrypted session, a copy of his tickets already waiting for him, and he never had to type his password. If he had used a non-Kerberos

EXAMPLE 26–5 Using the `-a`, `-f`, and `-x` Options With `telnet` *(Continued)*

version of `telnet`, he would have been prompted for his password, and it would have been sent over the network unencrypted. If an intruder had been watching network traffic at the time, the intruder would have known `david`'s password.

If you forward your Kerberos tickets, `telnet` (as well as the other commands discussed here) destroys them when it exits.

EXAMPLE 26–6 Using `rlogin` With the `-F` Option

Here, the user `jennifer` wants to log in to her own machine, `boston.example.com`. She forwards her existing tickets with the `-F` option, and encrypts the session with the `-x` option. She chooses `-F` rather than `-f` because after she is logged in to `boston`, she might want to perform other network transactions requiring tickets to be reforwarded. Also, because she is forwarding her existing tickets, she does not have to type her password.

```
% rlogin boston.example.com -F -x
This rlogin session is using encryption for all transmissions.
Last login Mon May 19 15:19:49 from daffodil
SunOS Release 5.9 (GENERIC) #2 Tue Nov 14 18:09:3 EST 2003
%
```

EXAMPLE 26–7 Setting the Protection Level in `ftp`

Suppose that `joe` wants to use `ftp` to get his mail from the directory `~joe/MAIL` from the machine `denver.example.com`, encrypting the session. The exchange would look like the following:

```
% ftp -f denver.example.com
Connected to denver.example.com
220 denver.example.org FTP server (Version 6.0) ready.
334 Using authentication type GSSAPI; ADAT must follow
GSSAPI accepted as authentication type
GSSAPI authentication succeeded Name (daffodil.example.org:joe)
232 GSSAPI user joe@MELPOMENE.EXAMPLE.COM is authorized as joe
230 User joe logged in.
Remote system type is UNIX.
Using BINARY mode to transfer files.
ftp> protect private
200 Protection level set to Private
ftp> cd ~joe/MAIL
250 CWD command successful.
ftp> get RMAIL
227 Entering Passive Mode (128,0,0,5,16,49)
```

EXAMPLE 26–7 Setting the Protection Level in `ftp` *(Continued)*

```
150 Opening BINARY mode data connection for RMAIL (158336 bytes).
226 Transfer complete. 158336 bytes received in 1.9 seconds (1.4e+02 Kbytes/s)
ftp> quit
%
```

To encrypt the session, `joe` sets the protection level to `private`.

CHAPTER 27

The Kerberos Service (Reference)

This chapter lists many of the files, commands, and daemons that are part of the Kerberos product. In addition, this chapter provides detailed information about how Kerberos authentication works.

This is a list of the reference information in this chapter.

- "Kerberos Files" on page 529
- "Kerberos Commands" on page 531
- "Kerberos Daemons" on page 532
- "Kerberos Terminology" on page 532
- "How the Kerberos Authentication System Works" on page 537
- "Gaining Access to a Service Using Kerberos" on page 538
- "Using Kerberos Encryption Types" on page 541
- "Using the `gsscred` Table" on page 543
- "Notable Differences Between Solaris Kerberos and MIT Kerberos" on page 543

Kerberos Files

TABLE 27–1 Kerberos Files

File Name	Description
`~/.gkadmin`	Default values for creating new principals in the SEAM Administration Tool
`~/.k5login`	List of principals that grant access to a Kerberos account
`/etc/krb5/kadm5.acl`	Kerberos access control list file, which includes principal names of KDC administrators and their Kerberos administration privileges

TABLE 27–1 Kerberos Files *(Continued)*

File Name	Description
`/etc/krb5/kadm5.keytab`	Keytab file for the `kadmin` service on the master KDC
`/etc/krb5/kdc.conf`	KDC configuration file
`/etc/krb5/kpropd.acl`	Kerberos database propagation configuration file
`/etc/krb5/krb5.conf`	Kerberos realm configuration file
`/etc/krb5/krb5.keytab`	Keytab file for network application servers
`/etc/krb5/warn.conf`	Kerberos ticket expiration warning and automatic renewal configuration file
`/etc/pam.conf`	PAM configuration file
`/tmp/krb5cc_`*uid*	Default credentials cache, where *uid* is the decimal UID of the user
`/tmp/ovsec_adm.`*xxxxxx*	Temporary credentials cache for the lifetime of the password changing operation, where *xxxxxx* is a random string
`/var/krb5/.k5.`*REALM*	KDC stash file, which contains a copy of the KDC master key
`/var/krb5/kadmin.log`	Log file for `kadmind`
`/var/krb5/kdc.log`	Log file for the KDC
`/var/krb5/principal`	Kerberos principal database
`/var/krb5/principal.kadm5`	Kerberos administrative database, which contains policy information
`/var/krb5/principal.kadm5.lock`	Kerberos administrative database lock file
`/var/krb5/principal.ok`	Kerberos principal database initialization file that is created when the Kerberos database is initialized successfully
`/var/krb5/principal.ulog`	Kerberos update log, which contains updates for incremental propagation
`/var/krb5/slave_datatrans`	Backup file of the KDC that the `kprop_script` script uses for propagation
`/var/krb5/slave_datatrans_`*slave*	Temporary dump file that is created when full updates are made to the specified *slave*

Kerberos Commands

This section lists some commands that are included in the Kerberos product.

TABLE 27–2 Kerberos Commands

Command	Description
`/usr/bin/ftp`	File Transfer Protocol program
`/usr/bin/kdestroy`	Destroys Kerberos tickets
`/usr/bin/kinit`	Obtains and caches Kerberos ticket-granting tickets
`/usr/bin/klist`	Displays current Kerberos tickets
`/usr/bin/kpasswd`	Changes a Kerberos password
`/usr/bin/ktutil`	Manages Kerberos keytab files
`/usr/bin/rcp`	Remote file copy program
`/usr/bin/rdist`	Remote file distribution program
`/usr/bin/rlogin`	Remote login program
`/usr/bin/rsh`	Remote shell program
`/usr/bin/telnet`	Kerberized `telnet` program
`/usr/lib/krb5/kprop`	Kerberos database propagation program
`/usr/sbin/gkadmin`	Kerberos database administration GUI program, which is used to manage principals and policies
`/usr/sbin/gsscred`	Manage gsscred table entries
`/usr/sbin/kadmin`	Remote Kerberos database administration program (run with Kerberos authentication), which is used to manage principals, policies, and keytab files
`/usr/sbin/kadmin.local`	Local Kerberos database administration program (run without Kerberos authentication and must be run on master KDC), which is used to manage principals, policies, and keytab files
`/usr/sbin/kclient`	Kerberos client installation script which is used with or without a installation profile
`/usr/sbin/kdb5_ldap_util`	Creates LDAP containers for Kerberos databases
`/usr/sbin/kdb5_util`	Creates Kerberos databases and stash files
`/usr/sbin/kgcmgr`	Configures Kerberos master and slave KDCs
`/usr/sbin/kproplog`	Lists a summary of update entries in the update log

Kerberos Daemons

The following table lists the daemons that the Kerberos product uses.

TABLE 27–3 Kerberos Daemons

Daemon	Description
`/usr/sbin/in.ftpd`	File Transfer Protocol daemon
`/usr/lib/krb5/kadmind`	Kerberos database administration daemon
`/usr/lib/krb5/kpropd`	Kerberos database propagation daemon
`/usr/lib/krb5/krb5kdc`	Kerberos ticket processing daemon
`/usr/lib/krb5/ktkt_warnd`	Kerberos ticket expiration warning and automatic renewal daemon
`/usr/sbin/in.rlogind`	Remote login daemon
`/usr/sbin/in.rshd`	Remote shell daemon
`/usr/sbin/in.telnetd`	`telnet` daemon

Kerberos Terminology

The following section presents Kerberos terms and their definitions. These terms are used throughout the Kerberos documentation. To grasp Kerberos concepts, an understanding of these terms is essential.

Kerberos-Specific Terminology

You need to understand the terms in this section in order to administer KDCs.

The *Key Distribution Center* or *KDC* is the component of Kerberos that is responsible for issuing credentials. These credentials are created by using information that is stored in the KDC database. Each realm needs at least two KDCs, a master and at least one slave. All KDCs generate credentials, but only the master KDC handles any changes to the KDC database.

A *stash file* contains the master key for the KDC. This key is used when a server is rebooted to automatically authenticate the KDC before starting the `kadmind` and `krb5kdc` commands. Because this file includes the master key, the file and any backups of the file should be kept secure. The file is created with read-only permissions for `root`. To keep the file secure, do not change the permissions. If the file is compromised, then the key could be used to access or modify the KDC database.

Authentication-Specific Terminology

You need to know the terms in this section to understand the authentication process. Programmers and system administrators should be familiar with these terms.

A *client* is the software that runs on a user's workstation. The Kerberos software that runs on the client makes many requests during this process. So, differentiating the actions of this software from the user is important.

The terms *server* and *service* are often used interchangeably. To clarify, the term *server* is used to define the physical system that Kerberos software is running on. The term *service* corresponds to a particular function that is being supported on a server (for example, `ftp` or `nfs`). Documentation often mentions servers as part of a service, but this definition clouds the meaning of the terms. Therefore, the term *server* refers to the physical system. The term *service* refers to the software.

The Kerberos product uses two types of keys. One type of key is a password derived key. The password derived key is given to each user principal and is known only to the user and to the KDC. The other type of key used by the Kerberos product is a random key that is not associated with a password and so is not suitable for use by user principals. Random keys are typically used for service principals that have entries in a keytab and session keys generated by the KDC. Service principals can use random keys since the service can access the key in the keytab which allows it to run non-interactively. Session keys are generated by the KDC (and shared between the client and service) to provide secure transactions between a client and a service.

A *ticket* is an information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains:

- Principal name of the service
- Principal name of the user
- IP address of the user's host
- Timestamp
- Value which defines the lifetime of the ticket
- Copy of the session key

All of this data is encrypted in the server's service key. Note, the KDC issues the ticket embedded in a credential described below. After a ticket has been issued, it can be reused until the ticket expires.

A *credential* is a packet of information that includes a ticket and a matching session key. The credential is encrypted with the requesting principal's key. Typically, the KDC generates a credential in response to a ticket request from a client.

An *authenticator* is information used by the server to authenticate the client user principal. An authenticator includes the principal name of the user, a timestamp, and other data. Unlike a ticket, an authenticator can be used once only, usually when access to a service is requested. An authenticator is encrypted by using the session key shared by the client and server. Typically, the client creates the authenticator and sends it with the server's or service's ticket in order to authenticate to the server or service.

Types of Tickets

Tickets have properties that govern how they can be used. These properties are assigned to the ticket when it is created, although you can modify a ticket's properties later. For example, a ticket can change from being `forwardable` to being `forwarded`. You can view ticket properties with the `klist` command. See "Viewing Kerberos Tickets" on page 515.

Tickets can be described by one or more of the following terms:

Forwardable/forwarded	A forwardable ticket can be sent from one host to another host, obviating the need for a client to reauthenticate itself. For example, if the user `david` obtains a forwardable ticket while on user `jennifer`'s machine, he can log in to his own machine without having to get a new ticket (and thus authenticate himself again). See Example 26–1 for an example of a forwardable ticket.
Initial	An initial ticket is a ticket that is issued directly, not based on a ticket-granting ticket. Some services, such as applications that change passwords, can require tickets to be marked initial in order to assure themselves that the client can demonstrate a knowledge of its secret key. An initial ticket indicates that the client has recently authenticated itself, instead of relying on a ticket-granting ticket, which might have been around for a long time.
Invalid	An invalid ticket is a postdated ticket that has not yet become usable. An invalid ticket will be rejected by an application server until it becomes validated. To be validated, a ticket must be presented to the KDC by the client in a ticket–granting service request, with the `VALIDATE` flag set, after its start time has passed.
Postdatable/postdated	A postdated ticket is a ticket that does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to be run late at night, because the ticket, if stolen, cannot be used until the batch job is to be run. When a postdated ticket is issued, it is issued as invalid and remains that way until its start time has passed, and the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the ticket is marked renewable, its lifetime is normally set to be equal to the duration of the full life of the ticket-granting ticket.

Proxiable/proxy	At times, it is necessary for a principal to allow a service to perform an operation on its behalf. The principal name of the proxy must be specified when the ticket is created. The Solaris release does not support proxiable or proxy tickets. A proxiable ticket is similar to a forwardable ticket, except that it is valid only for a single service, whereas a forwardable ticket grants the service the complete use of the client's identity. A forwardable ticket can therefore be thought of as a sort of super-proxy.
Renewable	Because it is a security risk to have tickets with very long lives, tickets can be designated as renewable. A renewable ticket has two expiration times: the time at which the current instance of the ticket expires, and the maximum lifetime for any ticket, which is one week. If a client wants to continue to use a ticket, the client renews it before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of 10 hours. If the client that is holding the ticket wants to keep it for more than an hour, the client must renew it within that hour. When a ticket reaches the maximum ticket lifetime (10 hours), it automatically expires and cannot be renewed.

For information on how to view the attributes of tickets, see "Viewing Kerberos Tickets" on page 515.

Ticket Lifetimes

Any time a principal obtains a ticket, including a ticket–granting ticket (TGT), the ticket's lifetime is set as the smallest of the following lifetime values:

- The lifetime value that is specified by the `-l` option of `kinit`, if `kinit` is used to get the ticket. By default, `kinit` used the maximum lifetime value.
- The maximum lifetime value (`max_life`) that is specified in the `kdc.conf` file.
- The maximum lifetime value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/`*realm*.
- The maximum lifetime value that is specified in the Kerberos database for the user principal that requests the ticket.

Figure 27–1 shows how a TGT's lifetime is determined and where the four lifetime values come from. Even though this figure shows how a TGT's lifetime is determined, basically the same thing happens when any principal obtains a ticket. The only differences are that `kinit` doesn't provide a lifetime value, and the service principal that provides the ticket provides a maximum lifetime value (instead of the `krbtgt/`*realm* principal).

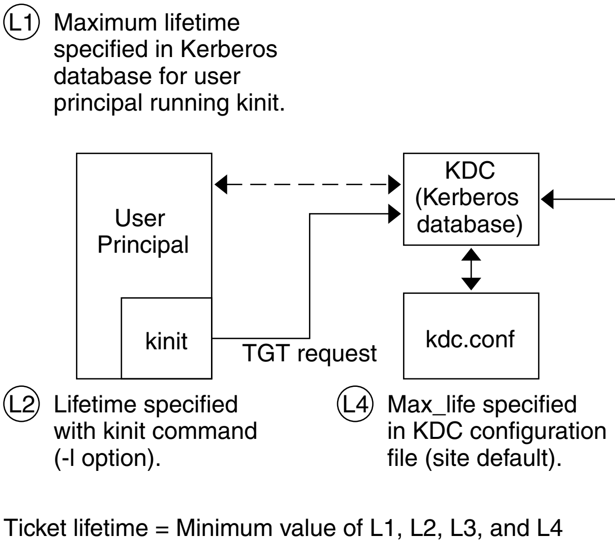

L1 Maximum lifetime specified in Kerberos database for user principal running kinit.
L3 Maximum lifetime specified in Kerberos database for service principal providing the TGT.
User Principal
kinit
KDC (Kerberos database)
krbgt/realm Principal
kdc.conf
TGT request
L2 Lifetime specified with kinit command (-l option).
L4 Max_life specified in KDC configuration file (site default).
Ticket lifetime = Minimum value of L1, L2, L3, and L4

FIGURE 27–1 How a TGT's Lifetime is Determined

The renewable ticket lifetime is also determined from the minimum of four values, but renewable lifetime values are used instead, as follows:

- The renewable lifetime value that is specified by the `-r` option of `kinit`, if `kinit` is used to obtain or renew the ticket.
- The maximum renewable lifetime value (`max_renewable_life`) that is specified in the `kdc.conf` file.
- The maximum lifetime renewable value that is specified in the Kerberos database for the service principal that provides the ticket. In the case of `kinit`, the service principal is `krbtgt/`*realm*.
- The maximum lifetime renewable value that is specified in the Kerberos database for the user principal that requests the ticket.

Kerberos Principal Names

Each ticket is identified by a principal name. The principal name can identify a user or a service. Here are examples of several principal names.

TABLE 27–4 Examples of Kerberos Principal Names

Principal Name	Description
`changepw/kdc1.example.com@EXAMPLE.COM`	A principal for the master KDC server that allows access to the KDC when you are changing passwords.

TABLE 27–4 Examples of Kerberos Principal Names *(Continued)*

Principal Name	Description
`clntconfig/admin@EXAMPLE.COM`	A principal that is used by the `kclient` installation utility.
`ftp/boston.example.com@EXAMPLE.COM`	A principal used by the `ftp` service. This principal can be used instead of a `host` principal.
`host/boston.example.com@EXAMPLE.COM`	A principal that is used by the Kerberized applications (`klist` and `kprop`, for example) and services (such as `ftp` and `telnet`). This principal is called a `host` or service principal. The principal is used to authenticate NFS mounts.
`K/M@EXAMPLE.COM`	The master key name principal. One master key name principal is associated with each master KDC.
`kadmin/history@EXAMPLE.COM`	A principal that includes a key used to keep password histories for other principals. Each master KDC has one of these principals.
`kadmin/kdc1.example.com@EXAMPLE.COM`	A principal for the master KDC server that allows access to the KDC by using `kadmind`.
`kadmin/changepw.example.com@EXAMPLE.COM`	A principal that is used to accept password change requests from clients that are not running a Solaris release.
`krbtgt/EXAMPLE.COM@EXAMPLE.COM`	This principal is used when you generate a ticket-granting ticket.
`krbtgt/EAST.EXAMPLE.COM@WEST.EXAMPLE.COM`	This principal is an example of a cross-realm ticket-granting ticket.
`nfs/boston.example.com@EXAMPLE.COM`	A principal that is used by the NFS service. This principal can be used instead of a `host` principal.
`root/boston.example.com@EXAMPLE.COM`	A principal that is associated with the `root` account on a client. This principal is called a `root` principal and provides `root` access to NFS mounted file systems..
username`@EXAMPLE.COM`	A principal for a user.
username`/admin@EXAMPLE.COM`	An `admin` principal that can be used to administer the KDC database.

How the Kerberos Authentication System Works

Applications allow you to log in to a remote system if you can provide a ticket that proves your identity, and a matching session key. The session key contains information that is specific to the user and the service that is being accessed. A ticket and session key are created by the KDC for all users when they first log in. The ticket and the matching session key form a credential. While using multiple networking services, a user can gather many credentials. The user needs to have a credential for each service that runs on a particular server. For example, access to the `ftp` service on a server named `boston` requires one credential. Access to the `ftp` service on another server requires its own credential.

The process of creating and storing the credentials is transparent. Credentials are created by the KDC that sends the credential to the requester. When received, the credential is stored in a credential cache.

Gaining Access to a Service Using Kerberos

To access a specific service on a specific server, the user must obtain two credentials. The first credential is for the ticket-granting ticket (known as the TGT). Once the ticket-granting service has decrypted this credential, the service creates a second credential for the server that the user is requesting access to. This second credential can then be used to request access to the service on the server. After the server has successfully decrypted the second credential, then the user is given access. The following sections describe this process in more detail.

Obtaining a Credential for the Ticket-Granting Service

1. To start the authentication process, the client sends a request to the authentication server for a specific user principal. This request is sent without encryption. No secure information is included in the request, so it is not necessary to use encryption.
2. When the request is received by the authentication service, the principal name of the user is looked up in the KDC database. If a principal matches the entry in the database, the authentication service obtains the private key for that principal. The authentication service then generates a session key to be used by the client and the ticket-granting service (call it Session key 1) and a ticket for the ticket-granting service (Ticket 1). This ticket is also known as the *ticket-granting ticket* (TGT). Both the session key and the ticket are encrypted by using the user's private key, and the information is sent back to the client.
3. The client uses this information to decrypt Session Key 1 and Ticket 1, by using the private key for the user principal. Because the private key should only be known by the user and the KDC database, the information in the packet should be safe. The client stores the information in the credentials cache.

During this process, a user is normally prompted for a password. If the password the user specifies is the same as the password that was used to build the private key stored in the KDC database, then the client can successfully decrypt the information that is sent by the authentication service. Now the client has a credential to be used with the ticket-granting service. The client is ready to request a credential for a server.

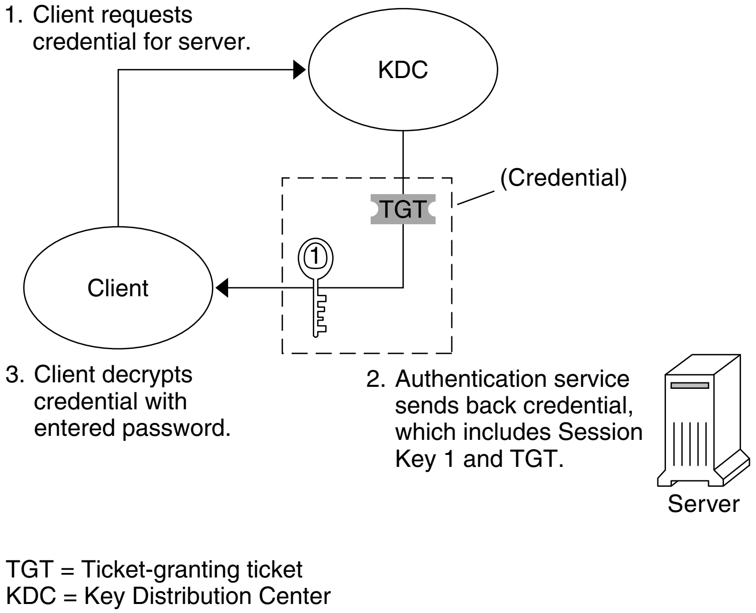

1. Client requests credential for server.
KDC
(Credential)
TGT
1
Client
3. Client decrypts credential with entered password.
2. Authentication service sends back credential, which includes Session Key 1 and TGT.
Server
TGT = Ticket-granting ticket
KDC = Key Distribution Center

FIGURE 27–2 Obtaining a Credential for the Ticket-Granting Service

Obtaining a Credential for a Server

1. To request access to a specific server, a client must first have obtained a credential for that server from the authentication service. See "Obtaining a Credential for the Ticket-Granting Service" on page 538. The client then sends a request to the ticket-granting service, which includes the service principal name, Ticket 1, and an authenticator that was encrypted with Session Key 1. Ticket 1 was originally encrypted by the authentication service by using the service key of the ticket-granting service.
2. Because the service key of the ticket-granting service is known to the ticket-granting service, Ticket 1 can be decrypted. The information in Ticket 1 includes Session Key 1, so the ticket-granting service can decrypt the authenticator. At this point, the user principal is authenticated with the ticket-granting service.
3. Once the authentication is successful, the ticket-granting service generates a session key for the user principal and the server (Session Key 2), and a ticket for the server (Ticket 2). Session Key 2 and Ticket 2 are then encrypted by using Session Key 1. Because Session Key 1 is known only to the client and the ticket-granting service, this information is secure and can be safely sent over the network.
4. When the client receives this information packet, the client decrypts the information by using Session Key 1, which it had stored in the credential cache. The client has obtained a credential to be used with the server. Now the client is ready to request access to a particular service on that server.

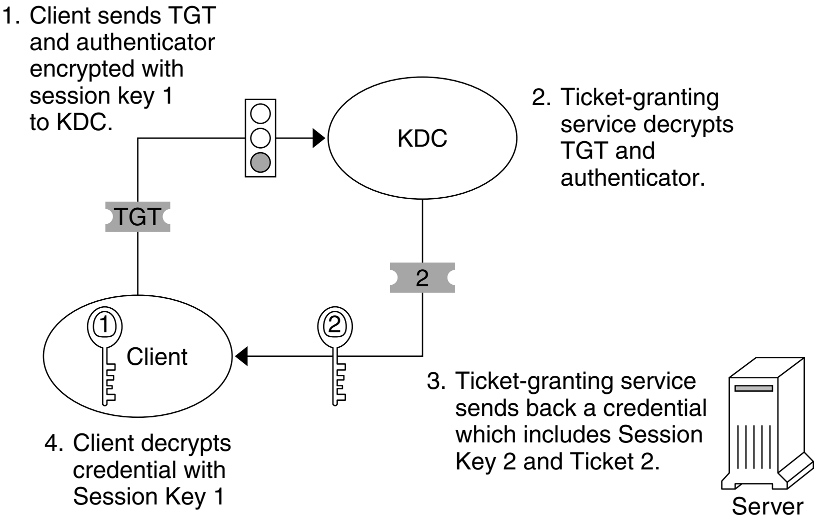

1. Client sends TGT and authenticator encrypted with session key 1 to KDC.
KDC
2. Ticket-granting service decrypts TGT and authenticator.
TGT
2
1
Client
2
3. Ticket-granting service sends back a credential which includes Session Key 2 and Ticket 2.
4. Client decrypts credential with Session Key 1
Server

FIGURE 27–3 Obtaining a Credential for a Server

Obtaining Access to a Specific Service

1. To request access to a specific service, the client must first have obtained a credential for the ticket-granting service from the authentication server, and a server credential from the ticket-granting service. See "Obtaining a Credential for the Ticket-Granting Service" on page 538 and "Obtaining a Credential for a Server" on page 539. The client can then send a request to the server including Ticket 2 and another authenticator. The authenticator is encrypted by using Session Key 2.
2. Ticket 2 was encrypted by the ticket-granting service with the service key for the service. Because the service key is known by the service principal, the service can decrypt Ticket 2 and get Session Key 2. Session Key 2 can then be used to decrypt the authenticator. If the authenticator is successfully decrypted, the client is given access to the service.

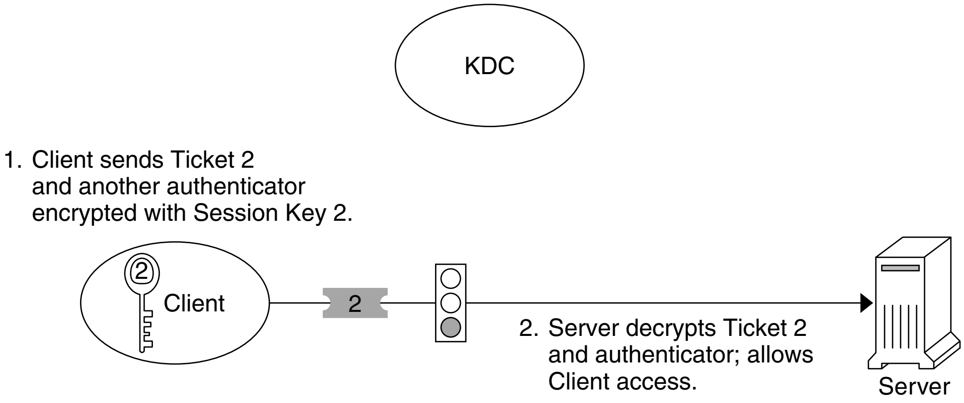

KDC
1. Client sends Ticket 2
and another authenticator
encrypted with Session Key 2.
2
Client
2
2. Server decrypts Ticket 2
and authenticator; allows
Client access.
Server

FIGURE 27–4 Obtaining Access to a Specific Service

Using Kerberos Encryption Types

Encryption types identify which cryptographic algorithms and mode to use when cryptographic operations are performed. The `aes`, `des3-cbc-sha1` and `rc4–hmac` encryption types enable the creation of keys that can be used for higher strength cryptographic operations. These higher strength operations enhance the overall security of the Kerberos service.

Note – The `aes256-cts-hmac-sha1-96` encryption type can be used with the Kerberos service if the unbundled Strong Cryptographic packages are installed.

When a client requests a ticket from the KDC, the KDC must use keys whose encryption type is compatible with both the client and the server. While the Kerberos protocol allows the client to request that the KDC use particular encryption types for the client's part of the ticket reply, the protocol does not allow the server to specify encryption types to the KDC.

Note – If you have a master KDC installed that is not running the Solaris 10 release, the slave KDCs must be upgraded to the Solaris 10 release before you upgrade the master KDC. A Solaris 10 master KDC will use the new encryption types, which an older slave will not be able to handle.

The following lists some of the issues that must be considered before you change the encryption types.

- The KDC assumes that the first key/enctype associated with the server principal entry in the principal database is supported by the server.

- On the KDC, you should make sure that the keys generated for the principal are compatible with the systems on which the principal will be authenticated. By default, the `kadmin` command creates keys for all supported encryption types. If the systems that the principal is used on do not support this default set of encryption types, then you should restrict the encryption types when creating a principal. You can restrict the encryption types through use of the `-e` flag in `kadmin addprinc` or by setting the `supported_enctypes` parameter in the `kdc.conf` file to this subset. The `supported_enctypes` parameter should be used when most of the systems in a Kerberos realm support a subset of the default set of encryption types. Setting `supported_enctypes` specifies the default set of encryption types `kadmin addprinc` uses when it creates a principal for a particular realm. As a general rule, it is best to control the encryption types used by Kerberos using one of these two methods.
- When determining the encryption types a system supports, consider both the version of Kerberos running on the system as well as the cryptographic algorithms supported by the server application for which a server principal is being created. For example, when creating an `nfs/hostname` service principal, you should restrict the encryption types to the types supported by the NFS server on that host. Note that in the Solaris 10 release, all supported Kerberos encryption types are also supported by the NFS server.
- The `master_key_enctype` parameter in the `kdc.conf` file can be used to control the encryption type of the master key that encrypts the entries in the principal database. Do not use this parameter if the KDC principal database has already been created. The `master_key_enctype` parameter can be used at database creation time to change the default master key encryption type from `des-cbc-crc` to a stronger encryption type. Make sure that all slave KDCs support the chosen encryption type and that they have an identical `master_key_enctype` entry in their `kdc.conf` when configuring the slave KDCs. Also, make sure that the `master_key_enctype` is set to one of the encryption types in `supported_enctypes`, if `supported_enctypes` is set in `kdc.conf`. If either of these issues are not handled properly, then the master KDC might not be able to work with the slave KDCs.
- On the client, you can control which encryption types the client requests when getting tickets from the KDC through a couple of parameters in `krb5.conf`. The `default_tkt_enctypes` parameter specifies the encryption types the client is willing to use when the client requests a ticket-granting ticket (TGT) from the KDC. The TGT is used by the client to acquire other server tickets in a more efficient manner. The effect of setting `default_tkt_enctypes` is to give the client some control over the encryption types used to protect the communication between the client and KDC when the client requests a server ticket using the TGT (this is called a TGS request). Note, that the encryption types specified in `default_tkt_enctypes` must match at least one of the principal key encryption types in the principal database stored on the KDC. Otherwise, the TGT request will fail. In most situations, it is best not to set `default_tkt_enctypes` because this parameter can be a source of interoperability problems. By default, the client code requests that all supported encryption types and the KDC choose the encryption types based on the keys the KDC finds in the principal database.

- The `default_tgs_enctypes` parameter restricts the encryption types the client requests in its TGS requests, which are used to acquire server tickets. This parameter also restricts the encryption types the KDC uses when creating the session key that the client and server share. For example, if a client wants to only use 3DES encryption when doing secure NFS, you should set `default_tgs_enctypes` = `des3-cbc-sha1`. Make sure that the client and server principals have a `des-3-cbc-sha1` key in the principal database. As with `default_tkt_enctype`, it is probably best in most cases not to set this because it can cause interoperability problems if the credentials are not setup properly both on the KDC and the server.
- On the server, you can control the encryption types accepted by the server with the `permitted_enctypes` in `kdc.conf`. In addition, you can specify the encryption types used when creating `keytab` entries. Again, it is generally best not to use either of these methods to control encryption types and instead let the KDC determine the encryption types to use because the KDC does not communicate with the server application to determine which key or encryption type to use.

Using the `gsscred` Table

The `gsscred` table is used by an NFS server when the server is trying to identify a Kerberos user, if the default mappings are not sufficient. The NFS service uses UNIX IDs to identify users. These IDs are not part of a user principal or a credential. The `gsscred` table provides additional mapping from GSS credentials to UNIX UIDs (from the password file). The table must be created and administered after the KDC database is populated. See "Mapping GSS Credentials to UNIX Credentials" on page 383 for more information.

When a client request comes in, the NFS service tries to map the credential name to a UNIX ID. If the mapping fails, the `gsscred` table is checked.

Notable Differences Between Solaris Kerberos and MIT Kerberos

The Solaris 10 version of the Kerberos service is based on MIT Kerberos version 1.2.1. The following lists the enhancements included in the Solaris 10 release that are not included in the MIT 1.2.1 version:

- Kerberos support of Solaris remote applications
- Incremental propagation for the KDC database
- Client configuration script
- Localized error messages
- BSM audit record support
- Thread safe use of Kerberos using GSS-API
- Use of the Encryption Framework for cryptography

This version also includes some post MIT 1.2.1 bug fixes. In particular, 1.2.5 btree bug fixes and 1.3 TCP support have been added.

PART VII

Solaris Auditing

This section provides information on the configuration, management, and use of the Solaris auditing subsystem.

◆ ◆ ◆

CHAPTER 28

Solaris Auditing (Overview)

Solaris auditing keeps a record of how the system is being used. The auditing service includes tools to assist with the analysis of the auditing data.

This chapter introduces how auditing works in the Solaris Operating System. The following is a list of the information in this chapter.

- "What Is Auditing?" on page 547
- "How Does Auditing Work?" on page 548
- "How Is Auditing Related to Security?" on page 549
- "Audit Terminology and Concepts" on page 550
- "Auditing on a System With Zones" on page 555
- "Solaris Auditing Enhancements in the Solaris 10 Release" on page 556

For planning suggestions, see Chapter 29, "Planning for Solaris Auditing." For procedures to configure auditing at your site, see Chapter 30, "Managing Solaris Auditing (Tasks)." For reference information, see Chapter 31, "Solaris Auditing (Reference)."

What Is Auditing?

Auditing is the collecting of data about the use of system resources. The audit data provides a record of security-related system events. This data can then be used to assign responsibility for actions that take place on a host. Successful auditing starts with two security features: identification and authentication. At each login, after a user supplies a user name and password, a unique audit session ID is generated and associated with the user's process. The audit session ID is inherited by every process that is started during the login session. Even if a user changes identity within a single session, all user actions are tracked with the same audit session ID. For more details about changing identity, see the `su`(1M) man page.

The auditing service makes the following possible:

- Monitoring security-relevant events that take place on the host
- Recording the events in a network-wide audit trail
- Detecting misuse or unauthorized activity
- Reviewing patterns of access and the access histories of individuals and objects
- Discovering attempts to bypass the protection mechanisms
- Discovering extended use of privilege that occurs when a user changes identity

During system configuration, you preselect which classes of audit records to monitor. You can also fine-tune the degree of auditing that is done for individual users.

After audit data is collected, postselection tools enable you to reduce and examine interesting parts of the audit trail. For example, you can choose to review audit records for individual users or specific groups. You can examine all records for a certain type of event on a specific day. Or, you can select records that were generated at a certain time of day.

Systems that install non-global zones can audit all zones identically from the global zone. These systems can also be configured to collect different records in the non-global zones. For more information, see "Auditing and Solaris Zones" on page 616.

How Does Auditing Work?

Auditing generates audit records when specified events occur. Most commonly, events that generate audit records include the following:

- System startup and system shutdown
- Login and logout
- Process creation or process destruction, or thread creation or thread destruction
- Opening, closing, creating, destroying, or renaming of objects
- Use of privilege capabilities or role-based access control (RBAC)
- Identification actions and authentication actions
- Permission changes by a process or user
- Administrative actions, such as installing a package
- Site-specific applications

Audit records are generated from three sources:

- By an application
- As a result of an asynchronous event
- As a result of a process system call

Once the relevant event information has been captured, the information is formatted into an audit record. The record is then written to audit files. Complete audit records are stored in binary format. With the Solaris 10 release, audit records can also be logged by the `syslog` utility.

Audit files that are stored in binary format can be stored in a local partition. The files can also be stored on NFS-mounted file servers. The location can include multiple partitions on the same system, partitions on different systems, or partitions on systems on different but linked networks. The collection of audit files that are linked together is considered an *audit trail*. Audit records accumulate in audit files chronologically. Contained in each audit record is information that identifies the event, what caused the event, the time of the event, and other relevant information.

Audit records can also be monitored by using the `syslog` utility. These audit logs can be stored locally. Or, the logs can be sent to a remote system over the UDP protocol. For more information, see "Audit Files" on page 553.

How Is Auditing Related to Security?

Solaris auditing helps to detect potential security breaches by revealing suspicious or abnormal patterns of system usage. Solaris auditing also provides a means to trace suspect actions back to a particular user, thus serving as a deterrent. Users who know that their activities are being audited are less likely to attempt malicious activities.

To protect a computer system, especially a system on a network, requires mechanisms that control activities before system processes or user processes begin. Security requires tools that monitor activities as the activities occur. Security also requires reports of activities after the activities have happened. Initial configuration of Solaris auditing requires that parameters be set before users log in or system processes begin. Most auditing activities involve monitoring current events and reporting those events that meet the specified parameters. How Solaris auditing monitors and reports these events is discussed in detail in Chapter 29, "Planning for Solaris Auditing," and Chapter 30, "Managing Solaris Auditing (Tasks)."

Auditing cannot prevent hackers from unauthorized entry. However, the auditing service can report, for example, that a specific user performed specific actions at a specific time and date. The audit report can identify the user by entry path and user name. Such information can be reported immediately to your terminal and to a file for later analysis. Thus, the auditing service provides data that helps you determine the following:

- How system security was compromised
- What loopholes need to be closed to ensure the desired level of security

Audit Terminology and Concepts

The following terms are used to describe the auditing service. Some definitions include pointers to more complete descriptions.

TABLE 28–1 Solaris Auditing Terms

Term	Definition
Audit class	A grouping of audit events. Audit classes provide a way to select a group of events to be audited. For more information, see "Audit Classes and Preselection" on page 551.
Audit directory	A repository of audit files in binary format. For a description of the types of audit directories, see "Audit Files" on page 553.
Audit event	A security-related system action that is audited. For ease of selection, events are grouped into audit classes. For a discussion of the system actions that can be audited, see "Audit Events" on page 551.
Audit policy	A set of auditing options that you can enable or disable at your site. These options include whether to record certain kinds of audit data. The options also include whether to suspend auditable actions when the audit trail is full. For more information, see "Determining Audit Policy" on page 564.
Audit record	Audit data that is stored in audit files. An audit record describes a single audit event. Each audit record is composed of audit tokens. For more information about audit records, see "Audit Records and Audit Tokens" on page 552.
Audit token	A field of an audit record or event. Each audit token describes an attribute of an audit event, such as a user, a program, or other object. For descriptions of all the audit tokens, see "Audit Token Formats" on page 623.
Audit trail	A collection of one or more audit files that store the audit data from all systems that run the auditing service. For more information, see "Audit Trail" on page 620.
Preselection	Preselection is the choice of which audit classes to monitor before you enable the auditing service. The audit events of preselected audit classes appear in the audit trail. Audit classes that are not preselected are not audited, so their events do not appear in the audit trail. A postselection tool, the `auditreduce` command, selects records from the audit trail. For more information, see "Audit Classes and Preselection" on page 551.
Public objects	A public object is a file that is owned by the `root` user and readable by the world. For example, files in the `/etc` directory and the `/usr/bin` directory are public objects. Public objects are not audited for read-only events. For example, even if the `file_read` (`fr`) audit class is preselected, the reading of public objects is not audited. You can override the default by changing the `public` audit policy option.

Audit Events

Security-relevant system actions can be audited. These auditable actions are defined as *audit events*. Audit events are listed in the `/etc/security/audit_event` file. Each audit event is defined in the file by an event number, a symbolic name, a short description, and the set of audit classes to which the event belongs. For more information on the `audit_event` file, see the `audit_event`(4) man page.

For example, the following entry defines the audit event for the `exec()` system call:

```
7:AUE_EXEC:exec(2):ps,ex
```

When you preselect for auditing either the audit class `ps` or the audit class `ex`, then `exec()` system calls are recorded in the audit trail.

Solaris auditing handles *attributable* and *nonattributable* events. The `exec()` system call can be attributed to a user, so the call is considered an attributable event. Events are nonattributable if the events occur at the kernel-interrupt level. Events that occur before a user is authenticated are also nonattributable. The `na` audit class handles audit events that are nonattributable. For example, booting the system is a nonattributable event.

```
113:AUE_SYSTEMBOOT:system booted:na
```

When the class to which an audit event belongs is preselected for auditing, the event is recorded in the audit trail. For example, when you preselect the `ps` and `na` audit classes for auditing, the `exec()` system calls and system boot actions, among other events, are recorded in the audit trail.

In addition to the audit events that are defined by the Solaris auditing service, third-party applications can generate audit events. Audit event numbers from 32768 to 65535 are available for third-party applications.

Audit Classes and Preselection

Each audit event belongs to an *audit class* or classes. Audit classes are convenient containers for large numbers of audit events. When you *preselect* a class to be audited, you specify that all the events in that class should be recorded in the audit trail. You can preselect for events on a system and for events initiated by a particular user. After the auditing service is running, you can dynamically add or remove audit classes from the preselected classes.

- **System-wide preselection –** Specify system-wide defaults for auditing in the `flags`, `naflags`, and `plugin` lines in the `audit_control` file. The `audit_control` file is described in "`audit_control` File" on page 612. See also the `audit_control`(4) man page.
- **User-specific preselection –** Specify additions to the system-wide auditing defaults for individual users in the `audit_user` database.

The audit preselection mask determines which classes of events are audited for a user. The user's audit preselection mask is a combination of the system-wide defaults and the audit classes that are specified for the user. For a more detailed discussion, see "Process Audit Characteristics" on page 620.

The `audit_user` database can be administered locally or by a name service. The Solaris Management Console provides the graphical user interface (GUI) to administer the database. For details, see the `audit_user`(4) man page.

- **Dynamic preselection** – Specify audit classes as arguments to the `auditconfig` command to add or remove those audit classes from a process or session. For more information, see the `auditconfig`(1M) man page.

A postselection command, `auditreduce`, enables you to select records from the preselected audit records. For more information, see "Examining the Audit Trail" on page 555 and the `auditreduce`(1M) man page.

Audit classes are defined in the `/etc/security/audit_class` file. Each entry contains the audit mask for the class, the name for the class, and a descriptive name for the class. For example, the `ps` and `na` class definitions appear in the `audit_class` file as follows:

```
0x00100000:ps:process start/stop
0x00000400:na:non-attribute
```

There are 32 possible audit classes. The classes include the two global classes: `all` and `no`. The audit classes are described in the `audit_class`(4) man page.

The mapping of audit events to classes is configurable. You can remove events from a class, add events to a class, and create a new class to contain selected events. For the procedure, see "How to Change an Audit Event's Class Membership" on page 578.

Audit Records and Audit Tokens

Each *audit record* records the occurrence of a single audited event. The record includes information such as who did the action, which files were affected, what action was attempted, and where and when the action occurred. The following example shows a `login` audit record:

```
header,81,2,login - local,,2003-10-13 11:23:31.050 -07:00
subject,root,root,other,root,other,378,378,0 0 example_system
text,successful login
return,success,0
```

The type of information that is saved for each audit event is defined by a set of *audit tokens*. Each time an audit record is created for an event, the record contains some or all of the tokens that are defined for the event. The nature of the event determines which tokens are recorded. In the preceding example, each line begins with the name of the audit token. The content of the audit token follows the name. Together, the four audit tokens comprise the `login` audit record.

For a detailed description of the structure of each audit token with an example of `praudit` output, see "Audit Token Formats" on page 623. For a description of the binary stream of audit tokens, see the `audit.log`(4) man page.

Audit Files

Audit records are collected in audit logs. Solaris auditing provides two output modes for audit logs. Logs that are called *audit files* store audit records in binary format. The set of audit files from a system or site provide a complete audit record. The complete audit record is called the *audit trail*.

The `syslog` utility collects and stores text version summaries of the audit record. A `syslog` record is not complete. The following example shows a `syslog` entry for a `login` audit record:

```
Oct 13  11:24:11 example_system auditd: [ID 6472 audit.notice] \
        login - login ok session 378 by root as root:other
```

A site can store audit records in both formats. You can configure the systems at your site to use binary mode, or to use both modes. The following table compares binary audit records with `syslog` audit records.

TABLE 28–2 Comparison of Binary Audit Records With `syslog` Audit Records

Feature	Binary Records	`syslog` Records
Protocol	Writes to the file system	Uses UDP for remote logging
Data type	Binary	Text
Record length	No limit	Up to 1024 characters per audit record
Location	Stored on local disk, and in directories that are mounted by using NFS	Stored in a location that is specified in the `syslog.conf` file
How to configure	Edit `audit_control` file, and protect and NFS-mount audit directories	Edit `audit_control` file, and edit `syslog.conf` file
How to read	Typically, in batch mode Browser output in XML	In real time, or searched by scripts that you have created for `syslog` Plain text output
Completeness	Guaranteed to be complete, and to appear in the correct order	Are not guaranteed to be complete
Timestamp	Greenwich Mean Time (GMT)	Time on the system that is being audited

Binary records provide the greatest security and coverage. Binary output meets the requirements of security certifications, such as the Common Criteria Controlled Access Protection Profile (CAPP). The records are written to a file system that is protected from snooping. On a single system, all binary records are collected and are displayed in order. The GMT timestamp on binary logs enables accurate comparison when systems on one audit trail are distributed across time zones. The `praudit -x` command enables you to view the records in a browser in XML. You can also use scripts to parse the XML output.

In contrast, the `syslog` records provide greater convenience and flexibility. For example, you can collect the `syslog` data from a variety of sources. Also, when you monitor `audit.notice` events in the `syslog.conf` file, the `syslog` utility logs an audit record summary with the current timestamp. You can use the same management and analysis tools that you have developed for `syslog` messages from a variety of sources, including workstations, servers, firewalls, and routers. The records can be viewed in real time, and can be stored on a remote system.

By using `syslog.conf` to store audit records remotely, you protect log data from alteration or deletion by an attacker. On the other hand, when audit records are stored remotely, the records are susceptible to network attacks such as denial of service and spoofed source addresses. Also, UDP can drop packets or can deliver packets out of order. The limit on `syslog` entries is 1024 characters, so some audit records could be truncated in the log. On a single system, not all audit records are collected. The records might not display in order. Because each audit record is stamped with the local system's date and time, you can not rely on the timestamp to construct an audit trail for several systems.

For more information on audit logs, refer to the following:

- `audit_syslog`(5) man page
- `audit.log`(4) man page
- "How to Configure `syslog` Audit Logs" on page 573

Audit Storage

An *audit directory* holds audit files in binary format. A typical installation uses many audit directories. The contents of all audit directories comprise the *audit trail*. Audit records are stored in audit directories in the following order:

- **Primary audit directory –** A directory where the audit files for a system are placed under normal conditions
- **Secondary audit directory –** A directory where the audit files for a system are placed if the primary audit directory is full or not available
- **Directory of last resort –** A local audit directory that is used if the primary audit directory and all secondary audit directories are not available

The directories are specified in the `audit_control` file. A directory is not used until a directory that is earlier in the list is full. For an annotated `audit_control` file with a list of directory entries, see Example 30–3.

Examining the Audit Trail

The auditing service provides commands to combine and reduce files from the audit trail. The `auditreduce` command can merge audit files from the audit trail. The command can also filter files to locate particular events. The `praudit` command reads the binary files. Options to the `praudit` command provide output that is suitable for scripting and for browser display.

Auditing on a System With Zones

A zone is a virtualized operating system environment that is created within a single instance of the Solaris OS. The auditing service audits the entire system, including activities in zones. A system that has installed non-global zones can run a single audit service to audit all zones identically. Or, it can configure one audit service per zone, including the global zone.

Sites that satisfy the following conditions can run a single audit service:

- The site requires a single-image audit trail.
- The non-global zones are used as application containers. The zones are part of one administrative domain. That is, no non-global zone has customized name service files.

  If all the zones on a system are within one administrative domain, the `zonename` audit policy can be used to distinguish audit events that execute in different zones.
- Administrators want low audit overhead. The global zone administrator audits all zones identically. Also, the global zone's audit daemon serves all zones on the system.

Sites that satisfy the following conditions can run one audit service per zone:

- The site does not require a single-image audit trail.
- The non-global zones have customized name service files. These separate administrative domains typically function as servers.
- Individual zone administrators want to control auditing in the zones that they administer. In per-zone auditing, zone administrators can decide to enable or to disable auditing for the zone that they administer.

The advantages of per-zone auditing are a customized audit trail for each zone, and the ability to disable auditing on a zone by zone basis. These advantages can be offset by the administrative overhead. The zone administrator customizes every audit configuration file. Each zone runs its own audit daemon, and has its own audit queue and audit logs. The zone's audit log files must be managed.

Solaris Auditing Enhancements in the Solaris 10 Release

Since the Solaris 9 release, the following features have been introduced to Solaris auditing:

- Solaris auditing can use the `syslog` utility to store audit records in text format. For discussion, see "Audit Files" on page 553. To set up the `audit_control` file to use the `syslog` utility, see "How to Configure `syslog` Audit Logs" on page 573.
- The `praudit` command has an additional output format, XML. XML is a standard, portable, processable format. The XML format enables the output to be read in a browser, and provides source for XML scripting for reports. The `-x` option to the `praudit` command is described in "`praudit` Command" on page 609.
- The default set of audit classes has been restructured. Audit metaclasses provide an umbrella for finer-grained audit classes. For a list of the default set of classes, see "Definitions of Audit Classes" on page 617.
- The `bsmconv` command no longer disables the use of the Stop-A key. The Stop-A event can be audited.
- The timestamp in audit records is reported in ISO 8601 format. For information about the standard, see `http://www.iso.org`.
- Three audit policy options have been added:
  - **public** – Public objects are no longer audited for read-only events. By not auditing public files, the audit log size is greatly reduced. Attempts to read sensitive files are therefore easier to monitor. For more on public objects, see "Audit Terminology and Concepts" on page 550.
  - **perzone** – The `perzone` policy has broad effects. A separate audit daemon runs in each zone. The daemon uses audit configuration files that are specific to the zone. Also, the audit queue is specific to the zone. For details, see the `auditd`(1M) and `auditconfig`(1M) man pages. For more on zones, see "Auditing and Solaris Zones" on page 616. For more on policy, see "How to Plan Auditing in Zones" on page 560.
  - **zonename** – The name of the Solaris zone in which an audit event occurred can be included in audit records. For more on zones, see "Auditing and Solaris Zones" on page 616. For a discussion of when to use the option, see "Determining Audit Policy" on page 564.
- Five audit tokens have been added:
  - The `cmd` token records the list of arguments and the list of environment variables that are associated with a command. For more information, see "`cmd` Token" on page 627.
  - The `path_attr` token records the sequence of attribute file objects that are below the `path` token object. For more information, see "`path_attr` Token" on page 633.
  - The `privilege` token records the use of privilege on a process. For more information, see "`privilege` Token" on page 633.
  - The `uauth` token records the use of authorization with a command or action. For more information, see "`uauth` Token" on page 639.

- The `zonename` token records the name of the non-global zone in which an audit event occurred. The `zonename` audit policy option determines whether the `zonename` token is included in the audit record. For more information, see "`zonename` Token" on page 639.

  For overview information, see "Auditing and Solaris Zones" on page 616. To learn about zones, see Part II, "Zones," in *System Administration Guide: Virtualization Using the Solaris Operating System*.

◆ ◆ ◆

CHAPTER 29

Planning for Solaris Auditing

This chapter describes how to set up the auditing service for your Solaris installation. In particular, the chapter covers issues that you need to consider before you enable the auditing service. The following is a list of the planning information in this chapter:

- "Planning Solaris Auditing (Task Map)" on page 559
- "Determining Audit Policy" on page 564
- "Controlling Auditing Costs" on page 566
- "Auditing Efficiently" on page 568

For an overview of auditing, see Chapter 28, "Solaris Auditing (Overview)." For procedures to configure auditing at your site, see Chapter 30, "Managing Solaris Auditing (Tasks)." For reference information, see Chapter 31, "Solaris Auditing (Reference)."

Planning Solaris Auditing (Task Map)

The following task map points to the major tasks that are required for planning disk space and what events to record.

Task	For Instructions
Determine auditing strategy for non-global zones	"How to Plan Auditing in Zones" on page 560
Plan storage space for the audit trail	"How to Plan Storage for Audit Records" on page 561
Determine who and what to audit	"How to Plan Who and What to Audit" on page 562

Planning Solaris Auditing (Tasks)

You want to be selective about what kinds of activities are audited. At the same time, you want to collect useful audit information. Audit files can quickly grow to fill the available space, so you should allocate enough disk space. You also need to carefully plan who to audit and what to audit.

▼ How to Plan Auditing in Zones

If your system has implemented zones, you have two audit configuration possibilities:

- You can configure a single audit service in the global zone for all zones.
- You can configure one audit service per zone.

For a discussion of the trade-offs, see "Auditing on a System With Zones" on page 555.

● **Choose one of the following methods.**

- **OPTION 1 - Configure a single audit service for all zones.**

  Auditing all zones identically can create a single-image audit trail. A single-image audit trail occurs when all zones on a system are part of one administrative domain. The audit records can then be easily compared, because the records in every zone are preselected with identical settings.

  This configuration treats all zones as part of one system. The global zone runs the only audit daemon on a system, and collects audit logs for every zone. You customize audit configuration files only in the global zone, then copy the audit configuration files to every non-global zone.

  a. **Copy the `audit_control` file from the global zone to every non-global zone.**

  b. **Use the same `audit_user` database for every zone.**

     The `audit_user` database might be a local file, or you might get it from a shared naming service.

  c. **Enable the audit records to be selected by zone.**

     To put the zone name as part of the audit record, set the `zonename` policy in the global zone. The `auditreduce` command can then select audit events by zone. For an example, see the `auditreduce`(1M) man page.

  To plan a single-image audit trail, use "How to Plan Who and What to Audit" on page 562 to plan. Start with the first step. The global zone administrator must also set aside storage, as described in "How to Plan Storage for Audit Records" on page 561.

- **OPTION 2 - Configure one audit service per zone.**

  Choose to configure per-zone auditing if different zones have different name service files, or if zone administrators want to control auditing in their zones.

  - When you configure per-zone auditing, you must configure the global zone for auditing. You set the `perzone` audit policy in the global zone.

    Note – If name service files are customized in non-global zones, and `perzone` policy is not set, then careful use of the audit tools is required to select usable records. A user ID in one zone can refer to a different user from the same ID in a different zone.

  - To generate records that can be traced to their originating zone, set the `zonename` audit policy in the global zone. In the global zone, run the `auditreduce` command with the `zonename` option. Then, in the `zonename` zone, run the `praudit` command on the `auditreduce` output.
  - Each zone administrator configures the audit files for the zone.

    A non-global zone administrator can set all policy options except `perzone` and `ahlt`.
  - Each zone administrator can enable or disable auditing in the zone.

  If you customize audit configuration files in every zone, use "How to Plan Who and What to Audit" on page 562 to plan for every zone. You can skip the first step. Each zone administrator must also set aside storage for every zone, as described in "How to Plan Storage for Audit Records" on page 561.

▼ How to Plan Storage for Audit Records

The audit trail requires dedicated file space. The dedicated file space for audit files must be available and secure. Each system should have several audit directories that are configured for audit files. You should decide how to configure the audit directories as one of the first tasks before you enable auditing on any systems. The following procedure covers the issues to be resolved when you plan for audit trail storage.

Before You Begin If you are implementing non-global zones, complete "How to Plan Auditing in Zones" on page 560 before using this procedure.

1. **Determine how much auditing your site needs.**

   Balance your site's security needs against the availability of disk space for the audit trail.

   For guidance on how to reduce space requirements while still maintaining site security, as well as how to design audit storage, see "Controlling Auditing Costs" on page 566 and "Auditing Efficiently" on page 568.

2 **Determine which systems are to be audited.**

On those systems, allocate space for at least one local audit directory. To specify the audit directories, see Example 30–3.

3 **Determine which systems are to store audit files.**

Decide which servers are to hold the primary and secondary audit directories. For examples of configuring disks for audit directories, see "How to Create Partitions for Audit Files" on page 580.

4 **Name the audit directories.**

Create a list of all the audit directories that you plan to use. For the naming conventions, see "Conventions for Binary Audit File Names" on page 621

5 **Determine which systems are to use which audit directories.**

Create a map that shows which system should use which audit directory. The map helps you to balance the auditing activity. For an illustration, see Figure 31–1 and Figure 31–2.

▼ How to Plan Who and What to Audit

Before You Begin If you are implementing non-global zones, complete "How to Plan Auditing in Zones" on page 560 before using this procedure.

1 **Determine if you want a single-system image audit trail.**

Systems within a single administrative domain can create a single-system image audit trail. If your systems use different naming services, start with the next step. You should complete the rest of the planning steps for every system.

A single-system image audit trail treats the systems that are being audited as one machine. To create a single-system image audit trail for a site, every system in the installation should be configured as follows:

- Use the same naming service.

  To interpret the audit records, two commands are used, `auditreduce` and `praudit`. For correct interpretation of the audit records, the `passwd`, `hosts`, and `audit_user` files must be consistent.
- Use the same `audit_warn`, `audit_event`, `audit_class`, and `audit_startup` files as every other system.
- Use the same `audit_user` database. The database can be in a naming service such as NIS or LDAP.
- Have identical `flags`, `naflags`, and `plugin` entries in the `audit_control` file.

2 Determine the audit policy.

Use the `auditconfig -lspolicy` command to see a short description of available policy options. By default, only the `cnt` policy is turned on. For a fuller discussion, see Step 8.

For the effects of the policy options, see "Determining Audit Policy" on page 564. To set audit policy, see "How to Configure Audit Policy" on page 584.

3 Determine if you want to modify event-to-class mappings.

In many situations, the default mapping is sufficient. However, if you add new classes, change class definitions, or determine that a record of a specific system call is not useful, you might also need to move an event to a different class.

For an example, see "How to Change an Audit Event's Class Membership" on page 578.

4 Determine which audit classes to preselect.

The best time to add audit classes or to change the default classes is before you start the auditing service.

The audit class values of the `flags`, `naflags`, and `plugin` entries in the `audit_control` file apply to all users and processes. The preselected classes determine whether an audit class is audited for success, for failure, or for both.

To preselect audit classes, see "How to Modify the `audit_control` File" on page 571.

5 Determine user exceptions to the system-wide preselected audit classes.

If you decide that some users should be audited differently from the system-wide preselected audit classes, modify the individual users' entries in the `audit_user` database.

For an example, see "How to Change a User's Audit Characteristics" on page 575.

6 Determine the minimum free disk space.

When disk space on an audit file system drops below the `minfree` percentage, the `auditd` daemon switches to the next available audit directory. The daemon then sends a warning that the soft limit has been exceeded.

To set the minimum free disk space, see Example 30–4.

7 Decide how to manage the `audit_warn` email alias.

The `audit_warn` script is run whenever the audit system needs to notify you of a situation that requires administrative attention. By default, the `audit_warn` script sends email to an `audit_warn` alias and sends a message to the console.

To set up the alias, see "How to Configure the `audit_warn` Email Alias" on page 583.

8 **Decide what action to take when all the audit directories are full.**

By default, when the audit trail overflows, the system continues to work. The system counts the audit records that are dropped, but does not record the events. For greater security, you can disable the `cnt` policy, and enable the `ahlt` policy. The `ahlt` policy stops the system when the audit audit trail overflows.

To configure these policy options, see Example 30–14.

9 **Decide whether to collect audit records in `syslog` format as well as in binary logs.**

For overview information, see "Audit Files" on page 553.

For an example, see "How to Configure `syslog` Audit Logs" on page 573.

Determining Audit Policy

Audit policy determines the characteristics of the audit records for the local system. The policy options are set by a startup script. The `bsmconv` script, which enables the auditing service, creates the `/etc/security/audit_startup` script. The `audit_startup` script executes the `auditconfig` command to establish audit policy. For details about the script, see the `audit_startup`(1M) man page.

Most audit policy options are disabled by default to minimize storage requirements and system processing demands. You can dynamically enable and disable audit policy options with the `auditconfig` command. You can permanently enable and disable the policy options with the `audit_startup` script.

Use the following table to determine if the needs of your site justify the additional overhead that results from enabling one or more audit policy options.

TABLE 29–1 Effects of Audit Policy Options

Policy Name	Description	Why Change the Policy Option?
`ahlt`	This policy applies to asynchronous events only. When disabled, this policy allows the event to complete without an audit record being generated. When enabled, this policy stops the system when the audit file systems are full. Administrative intervention is required to clean up the audit queue, make space available for audit records, and reboot. This policy can only be enabled in the global zone. The policy affects all zones.	The disabled option makes sense when system availability is more important than security. The enabled option makes sense in an environment where security is paramount.

TABLE 29–1 Effects of Audit Policy Options *(Continued)*

Policy Name	Description	Why Change the Policy Option?
`arge`	When disabled, this policy omits environment variables of an executed program from the `exec` audit record. When enabled, this policy adds the environment variables of an executed program to the `exec` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have suspicions about the environment variables that are being used in `exec` programs.
`argv`	When disabled, this policy omits the arguments of an executed program from the `exec` audit record. When enabled, this policy adds the arguments of an executed program to the `exec` audit record. The resulting audit records contain much more detail than when this policy is disabled.	The disabled option collects much less information than the enabled option. The enabled option makes sense when you are auditing a few users. The option is also useful when you have reason to believe that unusual `exec` programs are being run.
`cnt`	When disabled, this policy blocks a user or application from running. The blocking happens when audit records cannot be added to the audit trail because no disk space is available. When enabled, this policy allows the event to complete without an audit record being generated. The policy maintains a count of audit records that are dropped.	The disabled option makes sense in an environment where security is paramount. The enabled option makes sense when system availability is more important than security.
`group`	When disabled, this policy does not add a groups list to audit records. When enabled, this policy adds a groups list to every audit record as a special token.	The disabled option usually satisfies requirements for site security. The enabled option makes sense when you need to audit which groups are generating audit events.
`path`	When disabled, this policy records in an audit record at most one path that is used during a system call. When enabled, this policy records every path that is used in conjunction with an audit event to every audit record.	The disabled option places at most one path in an audit record. The enabled option enters each file name or path that is used during a system call in the audit record as a `path` token.
`perzone`	When disabled, this policy maintains a single audit configuration for a system. One audit daemon runs in the global zone. Audit events in non-global zones can be located in the audit record by preselecting the `zonename` audit token. When enabled, this policy maintains separate audit configuration, audit queue, and audit logs for each zone. A separate version of the audit daemon runs in each zone. This policy can be enabled in the global zone only.	The disabled option is useful when you have no special reason to maintain a separate audit log, queue, and daemon for each zone. The enabled option is useful when you cannot monitor your system effectively by simply preselecting the `zonename` audit token.

TABLE 29–1 Effects of Audit Policy Options *(Continued)*

Policy Name	Description	Why Change the Policy Option?
`public`	When disabled, this policy does not add read-only events of public objects to the audit trail when the reading of files is preselected. Audit classes that contain read-only events include `fr`, `fa`, and `cl`. When enabled, this policy records every read-only audit event of public objects if an appropriate audit class is preselected.	The disabled option usually satisfies requirements for site security. The enabled option is rarely useful.
`seq`	When disabled, this policy does not add a sequence number to every audit record. When enabled, this policy adds a sequence number to every audit record. The `sequence` token holds the sequence number.	The disabled option is sufficient when auditing is running smoothly. The enabled option makes sense when the `cnt` policy is enabled. The `seq` policy enables you to to determine when data was discarded.
`trail`	When disabled, this policy does not add a `trailer` token to audit records. When enabled, this policy adds a `trailer` token to every audit record.	The disabled option creates a smaller audit record. The enabled option clearly marks the end of each audit record with a `trailer` token. The `trailer` token is often used in conjunction with the `sequence` token. The `trailer` token provides easier and more accurate resynchronization of audit records.
`zonename`	When disabled, this policy does not include a `zonename` token in audit records. When enabled, this policy includes a `zonename` token in every audit record from a non-global zone.	The disabled option is useful when you do not need to compare audit behavior across zones. The enabled option is useful when you want to isolate and compare audit behavior across zones.

Controlling Auditing Costs

Because auditing consumes system resources, you must control the degree of detail that is recorded. When you decide what to audit, consider the following costs of auditing:

- Cost of increased processing time
- Cost of analysis of audit data
- Cost of storage of audit data

Cost of Increased Processing Time of Audit Data

The cost of increased processing time is the least significant of the costs of auditing. The first reason is that auditing generally does not occur during computation-intensive tasks, such as image processing, complex calculations, and so forth. The other reason is that the cost for single-user systems is usually small enough to ignore.

Cost of Analysis of Audit Data

The cost of analysis is roughly proportional to the amount of audit data that is collected. The cost of analysis includes the time that is required to merge and review audit records. Cost also includes the time that is required to archive the records and keep the records in a safe place.

The fewer records that you generate, the less time that is required to analyze the audit trail. Upcoming sections, "Cost of Storage of Audit Data" on page 567 and "Auditing Efficiently" on page 568, describe ways to audit efficiently. Efficient auditing reduces the amount of audit data, while still providing enough coverage to achieve your site's security goals.

Cost of Storage of Audit Data

Storage cost is the most significant cost of auditing. The amount of audit data depends on the following:

- Number of users
- Number of systems
- Amount of use
- Degree of traceability and accountability that is required

Because these factors vary from site to site, no formula can predetermine the amount of disk space to set aside for audit data storage. Use the following information as a guide:

- Preselect audit classes judiciously to reduce the volume of records that are generated.

  Full auditing, that is, with the `all` class, fills disks quickly. Even a simple task such as compiling a program could generate a large audit file. A program of modest size could generate thousands of audit records in less than a minute.

  For example, by omitting the `file_read` audit class, `fr`, you can significantly reduce audit volume. By choosing to audit for failed operations only, you can at times reduce audit volume. For example, by auditing for failed `file_read` operations, `-fr`, you can generate far fewer records than by auditing for all `file_read` events.
- Efficient audit file management is also important. After the audit records are created, file management reduces the amount of storage that is required.
- Understand the audit classes

  Before you configure auditing, you should understand the types of events that the classes contain. You can change the audit event-class mappings to optimize audit record collection.
- Develop a philosophy of auditing for your site.

  Base your philosophy on sensible measures. Such measures include the amount of traceability that your site requires, and the types of users that you administer.

Auditing Efficiently

The following techniques can help you achieve your organization's security goals while auditing more efficiently.

- Randomly audit only a certain percentage of users at any one time.
- Reduce the disk-storage requirements for audit files by combining, reducing, and compressing the files. Develop procedures for archiving the files, for transferring the files to removable media, and for storing the files offline.
- Monitor the audit data in real time for unusual behaviors. You can extend management and analysis tools that you have already developed to handle audit records in `syslog` files.

  You can also set up procedures to monitor the audit trail for certain activities. You can write a script to trigger an automatic increase in the auditing of certain users or certain systems in response to detection of unusual events.

  For example, you could write a script that does the following:

  1. Monitors the creation of audit files on all the audit file servers.
  2. Processes the audit files with the `tail` command.

     The piping of the output from the `tail -0f` command through the `praudit` command can yield a stream of audit records as the records are generated. For more information, see the `tail`(1) man page.
  3. Analyzes this stream for unusual message types or other indicators, and delivers the analysis to the auditor.

     Or, the script can be used to trigger automatic responses.
  4. Constantly monitors the audit directories for the appearance of new `not_terminated` audit files.
  5. Terminates outstanding `tail` processes when their files are no longer being written to.

◆ ◆ ◆

CHAPTER 30

Managing Solaris Auditing (Tasks)

This chapter presents procedures to help you set up and manage a Solaris system that is audited. This chapter also includes instructions for administering the audit trail. The following is a list of the information in this chapter.

- "Solaris Auditing (Task Map)" on page 569
- "Configuring Audit Files (Task Map)" on page 570
- "Configuring and Enabling the Auditing Service (Task Map)" on page 579
- "Managing Audit Records (Task Map)" on page 594

For an overview of the auditing service, see Chapter 28, "Solaris Auditing (Overview)." For planning suggestions, see Chapter 29, "Planning for Solaris Auditing." For reference information, see Chapter 31, "Solaris Auditing (Reference)."

Solaris Auditing (Task Map)

The following task map points to the major tasks that are required to manage auditing. The tasks are ordered.

Task	Description	For Instructions
1. Plan for auditing	Contains configuration issues to decide before you configure the auditing service.	"Planning Solaris Auditing (Task Map)" on page 559
2. Configure audit files	Defines which events, classes, and users require auditing.	"Configuring Audit Files (Task Map)" on page 570

Task	Description	For Instructions
3. Configure and enable auditing	Configures each host for disk space and other auditing service requirements. Then, starts the auditing service.	"Configuring and Enabling the Auditing Service (Task Map)" on page 579
	On a host that has installed non-global zones, configure one auditing service for the system, or one auditing service per zone.	"Configuring the Auditing Service in Zones (Tasks)" on page 590
4. Manage audit records	Collects and analyzes the audit data.	"Managing Audit Records (Task Map)" on page 594

Configuring Audit Files (Task Map)

The following task map points to the procedures for configuring files to customize auditing at your site. Most of the tasks are optional.

Task	Description	For Instructions
Select audit classes, and customize `audit_control` settings	Involves: ▪ Preselecting system-wide audit classes ▪ Specifying the audit directories for each system ▪ Setting disk space limits on audit file systems	"How to Modify the `audit_control` File" on page 571
(Optional) Log audit events in two modes	Enables you to monitor audit events in real time, in addition to storing audit records in binary format.	"How to Configure `syslog` Audit Logs" on page 573
(Optional) Change audit characteristics for users	Sets user-specific exceptions to the system-wide preselected audit classes.	"How to Change a User's Audit Characteristics" on page 575
(Optional) Add audit classes	Reduces the number of audit records by creating a new audit class to hold events.	"How to Add an Audit Class" on page 577
(Optional) Change event-to-class mappings	Reduces the number of audit records by changing the event-class mapping.	"How to Change an Audit Event's Class Membership" on page 578

Configuring Audit Files (Tasks)

Before you enable auditing on your network, you can customize the audit configuration files for your site auditing requirements. You can also restart the auditing service or reboot the local system to read changed configuration files after the auditing service has been enabled. However, the recommended practice is to customize your audit configuration as much as possible before you start the auditing service.

If you have implemented zones, you can choose to audit all zones from the global zone. To differentiate between zones in the audit output, you can set the `zonename` policy option.

Alternatively, to audit non-global zones individually, you can set the `perzone` policy in the global zone and customize the audit configuration files in the non-global zones. For an overview, see "Auditing and Solaris Zones" on page 616. For planning, see "How to Plan Auditing in Zones" on page 560.

▼ How to Modify the `audit_control` File

The `/etc/security/audit_control` file configures system-wide auditing. The file determines which events are audited, when audit warnings are issued, and the location of the audit files.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **(Optional) Save a backup copy of the `audit_control` file.**

   ```
   # cp /etc/security/audit_control /etc/security/audit_control.orig
   ```

3. **Modify the `audit_control` file for your site.**

   Each entry has the following format:

   keyword`:`*value*

keyword	Defines the type of line. The types are `dir`, `flags`, `minfree`, `naflags`, and `plugin`. The `dir` line can be repeated. For explanations of the keywords, see the following examples. For an example of a `plugin` entry, see "How to Configure `syslog` Audit Logs" on page 573
value	Specifies data that is associated with the line type.

Example 30–1 Preselecting Audit Classes for All Users

The `flags` line in the `audit_control` file defines which classes of attributable events are audited for all users on the system. The classes are separated by commas. White space is allowed. In this example, the events in the `lo` class are audited for all users.

```
# audit_control file
dir:/var/audit
flags:lo
minfree:20
naflags:lo
```

To see which events are in the `lo` class, read the `audit_event` file. You can also use the `bsmrecord` command, as shown in Example 30–25.

Example 30–2 Preselecting Nonattributable Events

In this example, all events in the `na` class, and all `login` events that are not attributable, are audited.

```
# audit_control file
dir:/var/audit
flags:lo
minfree:20
naflags:lo,na
```

Example 30–3 Specifying the Location of Binary Audit Data

The `dir` lines in the `audit_control` file list which audit file systems to use for binary audit data. In this example, three locations for binary audit data are defined.

```
# audit_control file
#
# Primary audit directory - NFS-mounted from audit server
dir:/var/audit/egret.1/files
#
# Secondary audit directory - NFS-mounted from audit server
dir:/var/audit/egret.2/files
#
# Directory of last resort local directory
dir:/var/audit
flags:lo
minfree:20
naflags:lo,na
plugin:
```

To set up file systems to hold audit binary audit data, see "How to Create Partitions for Audit Files" on page 580.

Example 30–4 Changing the Soft Limit for Warnings

In this example, the minimum free-space level for all audit file systems is set so that a warning is issued when only 10 percent of the file system is available.

```
# audit_control file
#
dir:/var/audit/examplehost.1/files
dir:/var/audit/examplehost.2/files
dir:/var/audit/localhost/files
flags:lo
```

```
minfree:10
naflags:lo,na
```

The `audit_warn` alias receives the warning. To set up the alias, see "How to Configure the `audit_warn` Email Alias" on page 583.

How to Configure `syslog` Audit Logs

You can instruct the auditing service to collect only binary audit data, or you can instruct the auditing service to collect binary data and text data. In the following procedure, you collect binary audit data and text audit data. The collected text audit data is a subset of the binary data.

Before You Begin Preselected audit classes must be specified on the `flags` line or the `naflags` line of the `audit_control` file. The text data is a subset of the preselected binary data.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **(Optional) Save a backup copy of the `audit_control` file.**

   ```
   # cp /etc/security/audit_control /etc/security/audit_control.save
   ```

3. **Add a `plugin` entry.**

   Plugins in the auditing service implement binary output and `syslog` output of audit data. The binary plugin is not specified. The `syslog` plugin must be specified. For more information, see "`auditd` Daemon" on page 606.

   A `plugin` entry has the following format:

   ```
   plugin:name=value; p_flags=classes
   ```

value	Lists the name of the plugin to use. Currently, the only valid value is the `audit_syslog.so.1` plugin.
classes	Lists a subset of the audit classes that are specified in the `flags` line and the `naflags` line.

   For more information about the `plugin` value, see the `audit_syslog`(5) man page.

4 **Add an `audit.notice` entry to the `syslog.conf` file.**

The entry includes the location of the log file.

```
# cat /etc/syslog.conf
...
audit.notice       /var/adm/auditlog
```

Text logs should not be stored where the binary audit files are stored. The `auditreduce` command assumes that all files in an audit partition are binary audit files.

5 **Create the log file.**

```
# touch /var/adm/auditlog
```

6 **Refresh the configuration information for the `syslog` service.**

```
# svcadm refresh system/system-log
```

7 **Regularly archive the `syslog` log files.**

The auditing service can generate extensive output. To manage the logs, see the `logadm`(1M) man page.

Example 30–5 Specifying Audit Classes for `syslog` Output

In the following example, the `syslog` utility collects a subset of the preselected audit classes.

```
# audit_control file
dir:/var/audit/host.1/files
dir:/var/audit/host.2/files
dir:/var/audit/localhost/files
flags:lo,ss
minfree:10
naflags:lo,na
plugin:name=audit_syslog.so.1; p_flags=-lo,-na,-ss
```

The `flags` and `naflags` entries instruct the system to collect all login/logout, nonattributable, and change of system state audit records in binary format. The `plugin` entry instructs the `syslog` utility to collect only failed logins, failed nonattributable events, and failed changes of system state.

Example 30–6 Putting `syslog` Audit Records on a Remote System

You can change the `audit.notice` entry in the `syslog.conf` file to point to a remote system. In this example, the name of the local system is `example1`. The remote system is `remote1`.

```
example1 # cat /etc/syslog.conf
...
audit.notice       @remote1
```

The `audit.notice` entry in the `syslog.conf` file on the `remote1` system points to the log file.

```
remote1 # cat /etc/syslog.conf
...
audit.notice       /var/adm/auditlog
```

▼ How to Change a User's Audit Characteristics

Definitions for each user are stored in the `audit_user` database. These definitions modify, for the specified user, the preselected classes in the `audit_control` file. The `nsswitch.conf` file determines if a local file or if a name service database is used. To calculate the user's final audit preselection mask, see "Process Audit Characteristics" on page 620.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **(Optional) Save a backup copy of the `audit_user` database.**

   ```
   # cp /etc/security/audit_user /etc/security/audit_user.orig
   ```

3. **Add new entries to the `audit_user` database.**

   In the local database, each entry has the following format:

   username:*always-audit*:*never-audit*

username	Selects the name of the user to be audited.
always-audit	Selects the list of audit classes that should always be audited for the specified user.
never-audit	Selects the list of audit classes that should never be audited for the specified user.

   You can specify multiple classes by separating the audit classes with commas.

   The `audit_user` entries are in effect at the user's next login.

Example 30–7 Changing Which Events Are Audited for One User

In this example, the `audit_control` file contains the preselected audit classes for the system:

```
# audit_control file
...
flags:lo,ss
minfree:10
naflags:lo,na
```

The `audit_user` file shows an exception. When the user `jdoe` uses a profile shell, that use is audited:

```
# audit_user file
jdoe:pf
```

The audit preselection mask for `jdoe` is a combination of the `audit_user` settings with the `audit_control` settings. The `auditconfig -getaudit` command shows the preselection mask for `jdoe`:

```
# auditconfig -getaudit
audit id = jdoe(1234567)
process preselection mask = ss,pf,lo(0x13000,0x13000)
terminal id (maj,min,host) = 242,511,example1(192.168.160.171)
audit session id = 2138517656
```

Example 30–8 Auditing Users Only, Not the System

In this example, the login and role activities of four users only are audited on this system. The `audit_control` file does not preselect audit classes for the system:

```
# audit_control file
...
flags:
minfree:10
naflags:
```

The `audit_user` file preselects two audit classes for four users:

```
# audit_user file
jdoe:lo,pf
kdoe:lo,pf
pdoe:lo,pf
sdoe:lo,pf
```

The following `audit_control` file protects the system from unwarranted intrusion. In combination with the `audit_user` file, this file protects the system more than the first `audit_control` file in this example.

```
# audit_control file
...
flags:
minfree:10
naflags:lo
```

▼ How to Add an Audit Class

When you create your own audit class, you can place into it just those audit events that you want to audit for your site. When you add the class on one system, you should copy the change to all systems that are being audited.

1 **Assume the Primary Administrator role, or become superuser.**

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2 **(Optional) Save a backup copy of the `audit_class` file.**

```
# cp /etc/security/audit_class /etc/security/audit_class.orig
```

3 **Add new entries to the `audit_class` file.**

Each entry has the following format:

`0x`*number*`:`*name*`:`*description*

0x	Identifies *number* as hexadecimal.
number	Defines the unique audit class mask.
name	Defines the letter name of the audit class.
description	Defines the descriptive name of the audit class.

The entry must be unique in the file. Do not use existing audit class masks.

Example 30–9 Creating a New Audit Class

This example creates a class to hold a small set of audit events. The added entry to the `audit_class` file is as follows:

```
0x01000000:pf:profile command
```

The entry creates a new audit class that is called `pf`. Example 30–10 populates the new audit class.

Troubleshooting If you have customized the `audit_class` database, make sure that any modifications to `audit_user` are consistent with the new audit classes. Errors occur when the audit classes in `audit_user` are not a subset of the `audit_class` database.

▼ How to Change an Audit Event's Class Membership

You might want to change an audit event's class membership to reduce the size of an existing audit class, or to place the event in a class of its own. When you reconfigure audit event-class mappings on one system, you should copy the change to all systems that are being audited.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, “Working With the Solaris Management Console (Tasks),” in *System Administration Guide: Basic Administration*.

2. **(Optional) Save a backup copy of the `audit_event` file.**

   ```
   # cp /etc/security/audit_event /etc/security/audit_event.orig
   ```

3. **Change the class to which particular events belong by changing the *class-list* of the events.**

   Each entry has the following format:

   number:*name*:*description*:*class-list*

number	Is the audit event ID.
name	Is the name of the audit event.
description	Typically, the system call or executable that triggers the creation of an audit record.
class-list	Is a comma-separated list of audit classes.

Example 30–10 Mapping Existing Audit Events to a New Class

This example maps an existing audit event to the new class that was created in Example 30–9. In the `audit_control` file, the binary audit record captures successes and failures of events in the `pf` class. The `syslog` audit log contains only failures of events in the `pf` class.

```
# grep pf | /etc/security/audit_class
0x01000000:pf:profile command
# vi /etc/security/audit_event
6180:AUE_prof_cmd:profile command:ua,as,pf
# vi audit_control
...
flags:lo,pf
```

```
plugin:name=audit_syslog.so.1; p_flags=-lo,-pf
```

Example 30–11 Auditing the Use of `setuid` Programs

This example creates a class to hold events that monitor calls to the `setuid` and `setgid` programs. The `audit_control` entries audit all successful invocations of the events in the `st` class.

```
# vi /etc/security/audit_class
0x00000800:st:setuid class
# vi /etc/security/audit_event
26:AUE_SETGROUPS:setgroups(2):st
27:AUE_SETPGRP:setpgrp(2):st
40:AUE_SETREUID:setreuid(2):st
41:AUE_SETREGID:setregid(2):st
214:AUE_SETEGID:setegid(2):st
215:AUE_SETEUID:seteuid(2):st
# vi audit_control
...
flags:lo,+st
plugin:name=audit_syslog.so.1; p_flags=-lo,+st
```

Configuring and Enabling the Auditing Service (Task Map)

The following task map points to procedures for configuring and enabling the auditing service. The tasks are ordered.

Task	Description	For Instructions
1. (Optional) Change the audit configuration files	Selects which events, classes, and users require auditing.	"Configuring Audit Files (Task Map)" on page 570
2. Create audit partitions	Creates disk space for the audit files, and protects them with file permissions.	"How to Create Partitions for Audit Files" on page 580
3. Create the `audit_warn` alias	Defines who should get email warnings when the auditing service needs attention.	"How to Configure the `audit_warn` Email Alias" on page 583
4. (Optional) Change audit policy	Defines additional audit data that your site requires.	"How to Configure Audit Policy" on page 584
5. Configure auditing in non-global zones	Enable non-global zones to collect auditing records	"Configuring the Auditing Service in Zones (Tasks)" on page 590

Task	Description	For Instructions
6. Enable auditing	Turns on the auditing service.	"How to Enable the Auditing Service" on page 586
	When `perzone` auditing is turned on, enables auditing in a non-global zone.	Example 30–18
7. (Optional) Disable auditing	Turns off the auditing service.	"How to Disable the Auditing Service" on page 587
	When `perzone` auditing is turned on, disables auditing in a non-global zone.	Example 30–23
8. (Optional) Reread auditing configuration changes	Reads audit configuration changes into the kernel while the `auditd` daemon is running.	"How to Update the Auditing Service" on page 589

Configuring and Enabling the Auditing Service (Tasks)

After the configuration files have been set up for your site, you need to set up disk space for your audit files. You also need to set up other attributes of the auditing service, and then enable the service. This section also contains procedures to refresh the auditing service when you change configuration settings.

When a non-global zone is installed, you can choose to audit the zone exactly as the global zone is being audited. Alternatively, to audit the non-global zone individually, you can modify the audit configuration files in the non-global zone. To customize audit configuration files, see "Configuring Audit Files (Task Map)" on page 570.

▼ How to Create Partitions for Audit Files

The following procedure shows how to create partitions for audit files, as well as the corresponding file systems and directories. Skip steps as necessary, depending on if you already have an empty partition, or if you have already mounted an empty file system.

1. **Assume the Primary Administrator role, or become superuser.**

   The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2. **Determine the amount of disk space that is required.**

   Assign at least 200 Mbytes of disk space per host. However, how much auditing you require dictates the disk space requirements. So, your disk space requirements might be far greater than this figure. Remember to include a local partition for a directory of last resort.

3 **Create dedicated audit partitions, as needed.**

This step is most easily done during server installation. You can also create the partitions on disks that have not yet been mounted on the server. For complete instructions on how to create the partitions, see Chapter 11, "Administering Disks (Tasks)," in *System Administration Guide: Devices and File Systems*.

```
# newfs /dev/rdsk/cwtxdysz
```

where `/dev/rdsk/cwtxdysz` is the raw device name for the partition.

If the local host is to be audited, also create an audit directory of last resort for the local host.

4 **Create mount points for each new partition.**

```
# mkdir /var/audit/server-name.n
```

where *server-name.n* is the name of the server plus a number that identifies each partition. The number is optional, but the number is useful when there are many audit directories.

5 **Add entries to automatically mount the new partitions.**

Add a line to the `/etc/vfstab` file that resembles the following:

```
/dev/dsk/cwtxdysz /dev/rdsk/cwtxdysz /var/audit/server-name.n   ufs  2  yes
```

6 **(Optional) Remove the minimum free space threshold on each partition.**

If you use the default configuration, a warning is generated when the directory is 80 percent full. The warning removes the reason to reserve free space on the partition.

```
# tunefs -m 0 /var/audit/server-name.n
```

7 **Mount the new audit partitions.**

```
# mount /var/audit/server-name.n
```

8 **Create audit directories on the new partitions.**

```
# mkdir /var/audit/server-name.n/files
```

9 **Correct the permissions on the mount points and new directories.**

```
# chmod -R 750 /var/audit/server-name.n/files
```

10 **On a file server, define the file systems to be made available to other hosts.**

Often, disk farms are installed to store the audit records. If an audit directory is to be used by several systems, then the directory must be shared through the NFS service. Add an entry that resembles the following for each directory to the `/etc/dfs/dfstab` file:

```
share -F nfs /var/audit/server-name.n/files
```

11 **On a file server, restart the NFS service.**

If this command is the first `share` command or set of `share` commands that you have initiated, the NFS daemons might not be running.

- **If the NFS service is offline, enable the service.**

  ```
  % svcs \*nfs\*
  disabled       Nov_02   svc:/network/nfs/rquota:default
  offline        Nov_02   svc:/network/nfs/server:default
  # svcadm enable network/nfs/server
  ```

- **If the NFS service is running, restart the service.**

  ```
  % svcs \*nfs\*
  online         Nov_02   svc:/network/nfs/client:default
  online         Nov_02   svc:/network/nfs/server:default
  # svcadm restart network/nfs/server
  ```

For more information about the NFS service, refer to “Setting Up NFS Services” in *System Administration Guide: Network Services*. For information on managing persistent services, see Chapter 15, “Managing Services (Overview),” in *System Administration Guide: Basic Administration* and the `smf`(5) man page.

Example 30–12 Creating an Audit Directory of Last Resort

All systems that run the auditing service should have a local file system that can be used if no other file system is available. In this example, a file system is being added to a system that is named `egret`. Because this file system is only used locally, none of the steps for a file server are necessary.

```
# newfs /dev/rdsk/c0t2d0
# mkdir /var/audit/egret
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret ufs  2  yes  -
# tunefs -m 0 /var/audit/egret
# mount /var/audit/egret
# mkdir /var/audit/egret/files
# chmod -R 750 /var/audit/egret/files
```

Example 30–13 Creating New Audit Partitions

In this example, a new file system is created on two new disks that are to be used by other systems in the network.

```
# newfs /dev/rdsk/c0t2d0
# newfs /dev/rdsk/c0t2d1
# mkdir /var/audit/egret.1
```

```
# mkdir /var/audit/egret.2
# grep egret /etc/vfstab
/dev/dsk/c0t2d0s1  /dev/rdsk/c0t2d0s1  /var/audit/egret.1 ufs  2  yes  -
/dev/dsk/c0t2d1s1  /dev/rdsk/c0t2d1s1  /var/audit/egret.2 ufs  2  yes  -
# tunefs -m 0 /var/audit/egret.1
# tunefs -m 0 /var/audit/egret.2
# mount /var/audit/egret.1
# mount /var/audit/egret.2
# mkdir /var/audit/egret.1/files
# mkdir /var/audit/egret.2/files
# chmod -R 750 /var/audit/egret.1/files /var/audit/egret.2/files
# grep egret /etc/dfs/dfstab
 share -F nfs /var/audit/egret.1/files
 share -F nfs /var/audit/egret.2/files
# svcadm enable network/nfs/server
```

▼ How to Configure the `audit_warn` Email Alias

The `audit_warn` script generates mail to an email alias that is called `audit_warn`. To send this mail to a valid email address, you can follow one of the options that are described in Step 2:

1 Assume the Primary Administrator role, or become superuser.

The Primary Administrator role includes the Primary Administrator profile. To create the role and assign the role to a user, see Chapter 2, "Working With the Solaris Management Console (Tasks)," in *System Administration Guide: Basic Administration*.

2 Configure the `audit_warn` email alias.

Choose one of the following options:

- **OPTION 1 –** Replace the `audit_warn` email alias with another email account in the `audit_warn` script.

  Change the email alias in the following line of the script:

  ```
  ADDRESS=audit_warn            # standard alias for audit alerts
  ```

- **OPTION 2 –** Redirect the `audit_warn` email to another mail account.

  In this case, you would add the `audit_warn` email alias to the appropriate mail aliases file. You could add the alias to the local `/etc/mail/aliases` file or to the `mail_aliases` database in the name space. The new entry would resemble the following if the `root` mail account was made a member of the `audit_warn` email alias:

  ```
  audit_warn: root
  ```

▼ How to Configure Audit Policy

Audit policy determines the characteristics of the audit records for the local host. When auditing is enabled, the contents of the `/etc/security/audit_startup` file determine the audit policy.

You can inspect, enable, or disable the current audit policy options with the the `auditconfig` command. You can also modify the policy options to the `auditconfig` command in the `audit_startup` script to make permanent audit policy changes.

1 **Assume a role that includes the Audit Control profile, or become superuser.**

To create a role that includes the Audit Control profile and to assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2 **Review the audit policy.**

Before auditing is enabled, the contents of the `audit_startup` file determine the audit policy:

```
#! /bin/sh
...
/usr/bin/echo "Starting BSM services."
/usr/sbin/auditconfig -setpolicy +cnt       Counts rather than drops records
/usr/sbin/auditconfig -conf       Configures event-class mappings
/usr/sbin/auditconfig -aconf      Configures nonattributable events
```

3 **View the available policy options.**

```
$ auditconfig -lspolicy
```

Note – The `perzone` and `ahlt` policy options can be set only in the global zone.

---

4 **Enable or disable selected audit policy options.**

```
# auditconfig -setpolicy prefixpolicy
```

prefix	A *prefix* value of + enables the policy option. A *prefix* value of - disables the policy option.
policy	Selects the policy to be enabled or to be disabled.

The policy is in effect until the next boot, or until the policy is modified by the `auditconfig -setpolicy` command.

For a description of each policy option, see "Determining Audit Policy" on page 564.

Example 30–14 Setting the `cnt` and `ahlt` Audit Policy Options

In this example, the `cnt` policy is disabled and the `ahlt` policy is enabled. With these settings, system use is halted when the audit partitions are full. These settings are appropriate when security is more important than availability. For restrictions on setting this policy, see Step 3.

The following `audit_startup` entries disable the `cnt` policy option and enable the `ahlt` policy option across reboots:

```
# cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy -cnt
/usr/sbin/auditconfig -setpolicy +ahlt
```

Example 30–15 Setting the `seq` Audit Policy Temporarily

In this example, the `auditd` daemon is running and the `ahlt` audit policy has been set. The `seq` audit policy is added to the current policy. The `seq` policy adds a `sequence` token to every audit record. This is useful for debugging the auditing service when audit records are corrupted, or when records are being dropped.

The + prefix adds the `seq` option to the audit policy, rather than replaces the current audit policy with `seq`. The `auditconfig` command puts the policy in effect until the next invocation of the command, or until the next boot.

```
$ auditconfig -setpolicy +seq
$ auditconfig -getpolicy
audit policies = ahlt,seq
```

Example 30–16 Setting the `perzone` Audit Policy

In this example, the `perzone` audit policy is set in the `audit_startup` script in the global zone. When a zone boots, the non-global zone collects audit records according to the audit configuration settings in its zone.

```
$ cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
```

```
/usr/sbin/auditconfig -setpolicy +perzone
/usr/sbin/auditconfig -setpolicy +cnt
```

Example 30–17

Changing an Audit Policy

In this example, the audit daemon is running and audit policy has been set. The `auditconfig` command changes the `ahlt` and `cnt` policies for the duration of the session. With these settings, audit records are dropped, but counted, when the audit file system is full. For restrictions on setting the `ahlt` policy, see Step 3.

```
$ auditconfig -setpolicy +cnt
$ auditconfig -setpolicy -ahlt
$ auditconfig -getpolicy
audit policies = cnt,seq
```

When the changes are put in the `audit_startup` file, the policies are permanently in effect:

```
$ cat /etc/security/audit_startup
#!/bin/sh
/usr/bin/echo "Starting BSM services."
/usr/sbin/deallocate -Is
/usr/sbin/auditconfig -conf
/usr/sbin/auditconfig -aconf
/usr/sbin/auditconfig -setpolicy +cnt
```

The `-ahlt` option does not have to be specified in the file, because the `ahlt` policy option is disabled by default. This setting is appropriate when availability is more important than the security that audit records provide.

▼ How to Enable the Auditing Service

This procedure enables the auditing service for all zones. To start the audit daemon in a non-global zone, see Example 30–18.

When auditing is configured securely, the system is in single-user mode until auditing is enabled. You can also enable auditing in multiuser mode.

Before You Begin You should perform this procedure as superuser after completing the following tasks:

- Planning – "Planning Solaris Auditing (Task Map)" on page 559
- Customizing audit files – "Configuring Audit Files (Task Map)" on page 570
- Setting up audit partitions – "How to Create Partitions for Audit Files" on page 580
- Setting up audit warning messages – "How to Configure the `audit_warn` Email Alias" on page 583

- Setting audit policy – "How to Configure Audit Policy" on page 584

1 **Run the script that enables the auditing service.**

Go to the `/etc/security` directory, and execute the `bsmconv` script there.

```
# cd /etc/security
# ./bsmconv
This script is used to enable the Basic Security Module (BSM).
Shall we continue with the conversion now? [y/n] y
bsmconv: INFO: checking startup file.
bsmconv: INFO: turning on audit module.
bsmconv: INFO: initializing device allocation.

The Basic Security Module is ready.
If there were any errors, please fix them now.
Configure BSM by editing files located in /etc/security.
Reboot this system now to come up with BSM enabled.
```

For the effects of the script, see the `bsmconv`(1M) man page.

2 **Reboot the system.**

```
# reboot
```

The startup file `/etc/security/audit_startup` causes the `auditd` daemon to run automatically when the system enters multiuser mode.

Another effect of the script is to turn on device allocation. To configure device allocation, see "Managing Device Allocation (Task Map)" on page 79.

Example 30–18 Enabling Auditing in a Non-Global Zone

In the following example, the global zone administrator turned on `perzone` policy after auditing was enabled in the global zone and after the non-global zone had booted. The zone administrator of the non-global zone has configured the audit files for the zone, and then starts the audit daemon in the zone.

```
zone1# svcadm enable svc:/system/auditd
```

▼ How to Disable the Auditing Service

If the auditing service is no longer required at some point, this procedure returns the system to the system state before auditing was enabled. If non-global zones are being audited, their auditing service is also disabled.

Caution – This command also disables device allocation. Do not run this command if you want to be able to allocate devices. To disable auditing and retain device allocation, see Example 30–19.

---

1. **Become superuser and bring the system into single-user mode.**

```
% su
Password:       <Type root password>
# init S
```

For more information, see the `init`(1M) man page.

2. **Run the script to disable auditing.**

Change to the `/etc/security` directory, and execute the `bsmunconv` script.

```
# cd /etc/security
# ./bsmunconv
```

Another effect of the script is to disable device allocation.

For information on the full effect of the `bsmunconv` script, see the `bsmconv`(1M) man page.

3. **Bring the system into multiuser mode.**

```
# init 6
```

Example 30–19

Disabling Auditing and Keeping Device Allocation

In this example, the auditing service stops collecting records, but device allocation continues to work. All values from the `flags`, `naflags`, and `plugin` entries in the `audit_control` file are removed, as are all user entries in the `audit_user` file.

```
# audit_control file
...
flags:
minfree:10
naflags:
plugin:

# audit_user file
```

The `auditd` daemon runs, but no audit records are kept.

Example 30–20 Disabling Auditing on a Per-Zone Basis

In this example, the auditing service stops running in the zone where the auditing service is disabled. Device allocation continues to work. When this command is run in the global zone, and the `perzone` audit policy is not set, auditing is disabled for all zones, not just the global zone.

```
# svcadm disable svc:/system/auditd
```

▼ How to Update the Auditing Service

This procedure restarts the `auditd` daemon when you have made changes to audit configuration files after the daemon has been running.

1 **Assume a role that includes the Audit Control rights profile, or become superuser.**

To create a role that includes the Audit Control rights profile and assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2 **Choose the appropriate command.**

- **If you modify the `naflags` line in the `audit_control` file, change the kernel mask for nonattributable events.**

  ```
  $ /usr/sbin/auditconfig -aconf
  ```

  You can also reboot.

- **If you modify other lines in the `audit_control` file, reread the `audit_control` file.**

  The audit daemon stores information from the `audit_control` file internally. To use the new information, either reboot the system or instruct the audit daemon to read the modified file.

  ```
  $ /usr/sbin/audit -s
  ```

  Note – Audit records are generated based on the audit preselection mask that is associated with each process. Executing `audit -s` does *not* change the masks in existing processes. To change the preselection mask for an existing process, you must restart the process. You can also reboot.

  The `audit -s` command causes the audit daemon to re-read the `directory` and `minfree` values from the `audit_control` file. The command changes the generation of the preselection mask for processes spawned by subsequent logins.

- **If you modify the `audit_event` file or the `audit_class` file while the audit daemon is running, refresh the auditing service.**

  Read the modified event-class mappings into the system, and ensure that each user who uses the machine is correctly audited.

  ```
  $ auditconfig -conf
  $ auditconfig -setumask auid classes
  ```

auid	Is the user ID.
classes	Are the preselected audit classes.

- **To change audit policy on a running system, see Example 30–15.**

Example 30–21 Restarting the Audit Daemon

In this example, the system is brought down to single-user mode, then back up to multiuser mode. When the system is brought into multiuser mode, modified audit configuration files are read into the system.

```
# init S
# init 6
```

Configuring the Auditing Service in Zones (Tasks)

The auditing service audits the entire system, including audit events in zones. A system that has installed non-global zones can audit all zones identically, or can control auditing per zone. For background, see “Auditing on a System With Zones” on page 555. To plan, see “How to Plan Auditing in Zones” on page 560.

▼ How to Configure All Zones Identically for Auditing

This procedure enables audits every zone identically. This method requires the least computer overhead and administrative resources.

1. **Configure the global zone for auditing.**

   a. **Complete the tasks in “Configuring Audit Files (Task Map)” on page 570.**

   b. **Complete the tasks in “Configuring and Enabling the Auditing Service (Task Map)” on page 579, with the following exceptions.**

      - Do not enable `perzone` audit policy.

- Do not enable the auditing service. You enable the auditing service after you have configured the non-global zones for auditing.

2 Copy the audit configuration files from the global zone to every non-global zone.

Copy any of the following files that you have edited: `audit_class`, `audit_control`, `audit_event`, `audit_user`. Do not copy `audit_startup` or `audit_warn`. You do not have to copy files that you have not edited.

You have two options. As superuser, you can copy the files, or loopback mount the files. The non-global zone must be running.

- **Copy the files.**

  a. From the global zone, list the `/etc/security` directory in the non-global zone.

  ```
  # ls /zone/zonename/etc/security/
  ```

  b. Copy the audit configuration files to the zone's `/etc/security` directory.

  ```
  # cp /etc/security/audit-file /zone/zonename/etc/security/audit-file
  ```

  Later, if you modify an audit configuration file in the global zone, you re-copy the file to the non-global zones.

- **Loopback mount the configuration files.**

  a. From the global zone, halt the non-global zone.

  ```
  # zoneadm -z non-global-zone halt
  ```

  b. Create a loopback mount for every audit configuration file that you modified in the global zone.

  ```
  # zonecfg -z non-global-zone
   add filesystem
      set special=/etc/security/audit-file
      set directory=/etc/security/audit-file
      set type=lofs
      end
   exit
  ```

Example 30–22 Loopback Mounting Audit Configuration Files

In this example, the system administrator has modified the `audit_class`, `audit_event`, `audit_control`, `audit_user`, `audit_startup`, and `audit_warn` files.

The `audit_startup` and `audit_warn` files are read in the global zone only, so do not have to be loopback mounted into the non-global zones.

On this system, `machine1`, the administrator has created two non-global zones, `machine1–webserver` and `machine1–appserver`.

```
# zoneadm -z machine1-webserver halt
# zoneadm -z machine1-appserver halt
# zonecfg -z machine1-webserver
 add filesystem
    set special=/etc/security/audit_class
    set directory=/etc/security/audit_class
    set type=lofs
    end
 add filesystem
    set special=/etc/security/audit_event
    set directory=/etc/security/audit_event
    set type=lofs
    end
 add filesystem
    set special=/etc/security/audit_control
    set directory=/etc/security/audit_control
    set type=lofs
    end add filesystem
    set special=/etc/security/audit_user
    set directory=/etc/security/audit_user
    set type=lofs
    end
 exit
# zonecfg -z machine1-appserver
 add filesystem
    set special=/etc/security/audit_class
    set directory=/etc/security/audit_class
    set type=lofs
    end
...
 exit
```

When the zones are rebooted, the audit configuration files are read-only in the zones. When the global administrator modifies the files in the global zone, the changes are immediately effective in the non-global zones.

▼ How to Configure Per-Zone Auditing

This procedure enables separate zone administrators to control the auditing service in their zone. For the complete list of policy options, see the `auditconfig`(1M) man page.

1 **In the global zone, configure auditing, but do not enable the auditing service.**

   a. **Complete the tasks in "Configuring Audit Files (Task Map)" on page 570.**

   b. **Complete the tasks in "Configuring and Enabling the Auditing Service (Task Map)" on page 579, with the following exceptions.**

      - Add the `perzone` audit policy. For an example, see Example 30–16.
      - Do not enable the auditing service. You enable the auditing service after the non-global zones are configured for auditing.

2 **In each non-global zone, configure the audit files.**

Note – If you are planning to disable auditing in the non-global zone, you can skip this step. To disable auditing, see Example 30–23.

---

   a. **Complete the tasks in "Configuring Audit Files (Task Map)" on page 570.**

   b. **Follow the procedures that are described in "Configuring and Enabling the Auditing Service (Task Map)" on page 579.**

   c. **Do not configure system-wide audit settings.**

   Specifically, do not add the `perzone` or `ahlt` to the non-global zone's `audit_startup` file. And do not run the `bsmconv` command from the non-global zone.

   d. **Enable auditing in your zone.**

   When the global zone reboots after auditing is configured, auditing is automatically enabled in your zone.

   If the global zone administrator activates the `perzone` audit policy after the system is booted, individual zone administrators must enable auditing. For details, see Example 30–18.

3 **In the global zone, enable the auditing service.**

For the procedure, see "How to Enable the Auditing Service" on page 586.

Example 30–23 Disabling Auditing in a Non-Global Zone

This example works if the global zone has set the `perzone` audit policy. The zone administrator of the `noaudit` zone disables auditing for that zone. Because the administrator planned to disable auditing, she did not edit the audit configuration files.

```
noauditzone # svcadm disable svc:/system/auditd
```

Managing Audit Records (Task Map)

The following task map points to procedures for selecting, analyzing, and managing audit records.

Task	Description	For Instructions
Display the formats of audit records	Shows the kind of information that is collected for an audit event, and the order in which the information is presented.	"How to Display Audit Record Formats" on page 594
Merge audit records	Combines audit files from several machines into one audit trail.	"How to Merge Audit Files From the Audit Trail" on page 596
Select records to examine	Selects particular events for study.	"How to Select Audit Events From the Audit Trail" on page 598
Display audit records	Enables you to view binary audit records.	"How to View the Contents of Binary Audit Files" on page 600
Clean up incorrectly named audit files	Provides an end timestamp to audit files that were inadvertently left open by the auditing service.	"How to Clean Up a `not_terminated` Audit File" on page 602
Prevent audit trail overflow	Prevents the audit file systems from becoming full.	"How to Prevent Audit Trail Overflow" on page 603

Managing Audit Records

By managing the audit trail, you can monitor the actions of users on your network. Auditing can generate large amounts of data. The following tasks show you how to work with all this data.

▼ How to Display Audit Record Formats

To write scripts that can find the audit data that you want, you need to know the order of tokens in an audit event. The `bsmrecord` command displays the audit event number, audit class, selection mask, and record format of an audit event.

- **Put the format of all audit event records in an HTML file.**

  The `-a` option lists all audit event record formats. The `-h` option puts the list in HTML format that can be displayed in a browser.

  ```
  % bsmrecord -a -h > audit.events.html
  ```

  When you display the `*html` file in a browser, use the browser's Find tool to find specific records.

For more information, see the `bsmrecord`(1M) man page.

Example 30–24

Displaying the Audit Record Formats of a Program

In this example, the format of all audit records that are generated by the `login` program are displayed. The login programs include `rlogin`, `telnet`, `newgrp`, role login to the Solaris Management Console, and Solaris Secure Shell.

```
% bsmrecord -p login
terminal login
  program     /usr/sbin/login      See login(1)
              /usr/dt/bin/dtlogin  See dtlogin
  event ID    6152                 AUE_login
  class       lo                   (0x00001000)
      header
      subject
      text                         error message or "successful login"
      return

login: logout
  program     various              See login(1)
  event ID    6153                 AUE_logout
...

newgrp
  program     newgrp               See newgrp login
  event ID    6212                 AUE_newgrp_login
...

rlogin
  program     /usr/sbin/login      See login(1) - rlogin
  event ID    6155                 AUE_rlogin
...

SMC: role login
  program     SMC server           See role login
  event ID    6173                 AUE_role_login
...

/usr/lib/ssh/sshd
  program     /usr/lib/ssh/sshd    See login - ssh
  event ID    6172                 AUE_ssh
...

telnet login
  program     /usr/sbin/login      See login(1) - telnet
  event ID    6154                 AUE_telnet
  ...
```

Example 30–25 Displaying the Audit Record Formats of an Audit Class

In this example, the format of all audit records in the `fd` class are displayed.

```
% bsmrecord -c fd

rmdir
  system call rmdir                     See rmdir(2)
  event ID    48                        AUE_RMDIR
  class       fd                        (0x00000020)
      header
      path
      [attribute]
      subject
      [use_of_privilege]
      return

unlink
  system call unlink                    See unlink(2)
  event ID    6                         AUE_UNLINK
  ...

unlinkat
  system call unlinkat                  See openat(2)
  event ID    286                       AUE_UNLINKAT
  ...
```

▼ How to Merge Audit Files From the Audit Trail

By merging all audit files in all the audit directories, you can analyze the contents of the entire audit trail. The `auditreduce` command merges all the records from its input files into a single output file. The input files can then be deleted. When the output file is placed in a directory that is named `/etc/security/audit`*server-name*`/files`, the `auditreduce` command can find the output file without your specifying the full path.

Note – This procedure applies only to binary audit records.

---

1 **Assume a role that includes the Audit Review profile, or become superuser.**

The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2 **Create a directory for storing merged audit files.**

`# mkdir` *audit-trail-directory*

3 **Limit access to the directory.**

```
# chmod 700 audit-trail-directory
# ls -la audit-trail-directory
drwx------   3 root     sys          512 May 12 11:47 .
drwxr-xr-x   4 root     sys         1024 May 12 12:47 ..
```

4 **Merge the audit records in the audit trail.**

Change directories to the *audit-trail-directory* and merge the audit records into a file with a named suffix. All directories that are listed in the `dir` lines of the `audit_control` file on the local system are merged.

```
# cd audit-trail-directory
# auditreduce -Uppercase-option -O suffix
```

The uppercase options to the `auditreduce` command manipulate files in the audit trail. The uppercase options include the following:

`-A`	Selects all of the files in the audit trail.
`-C`	Selects complete files only. This option ignores files with the suffix `not_terminated`.
`-M`	Selects files with a particular suffix. The suffix can be a machine name, or it can be a suffix that you have specified for a summary file.
`-O`	Creates an audit file with 14-character timestamps for both the start time and the end time, with the suffix *suffix* in the current directory.

Example 30–26 Copying Audit Files to a Summary File

In the following example, the System Administrator role, `sysadmin`, copies all files from the audit trail into a merged file.

```
$ whoami
sysadmin
$ mkdir /var/audit/audit_summary.dir
$ chmod 700 /var/audit/audit_summary.dir
$ cd /var/audit/audit_summary.dir
$ auditreduce -A -O All
$ ls *All
20030827183214.20030827215318.All
```

In the following example, only complete files are copied from the audit trail into a merged file.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O Complete
$ ls *Complete
20030827183214.20030827214217.Complete
```

In the following example, only complete files are copied from the `example1` machine into a merged file.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -M example1 -O example1summ
$ ls *summ
20030827183214.20030827214217.example1summ
```

Example 30–27 Moving Audit Files to a Summary File

The `-D` option to the `auditreduce` command deletes an audit file when you copy it to another location. In the following example, the complete audit files from one system are copied to the summary directory for later examination.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -C -O daily_example1 -D example1
$ ls *example1
20030827183214.20030827214217.daily_example1
```

The audit files from the `example1` system that were the input to the `*daily_example1` file are removed when this command successfully completes.

▼ How to Select Audit Events From the Audit Trail

You can filter audit records for examination. For the complete list of filtering options, see the `auditreduce`(1M) man page.

1. **Assume a role that includes the Audit Review profile, or become superuser.**

   The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2. **Select the kinds of records that you want from the audit trail, or from a specified audit file.**

   `auditreduce` -*lowercase-option* *argument* [*optional-file*]

argument	Specific argument that a lowercase option requires. For example, the `-c` option requires an *argument* of an audit class, such as `ua`.
`-d`	Selects all of the events on a particular date. The date format for *argument* is *yyymmdd*. Other date options, `-b` and `-a`, select events before and after a particular date.

`-u`	Selects all of the events attributable to a particular user. The *argument* is a user name. Another user option, `-e`, selects all of the events attributable to an effective user ID.
`-c`	Selects all of the events in a preselected audit class. The *argument* is an audit class name.
`-m`	Selects all of the instances of a particular audit event. The *argument* is an audit event.
optional-file	Is the name of an audit file.

Example 30–28 Combining and Reducing Audit Files

The `auditreduce` command can eliminate the less interesting records as it combines the input files. For example, you might use the `auditreduce` command to retain only the login and logout records in audit files that are over a month old. If you need to retrieve the complete audit trail, you could recover the trail from backup media.

```
# cd /var/audit/audit_summary.dir
# auditreduce -O lo.summary -b 20030827 -c lo; compress *lo.summary
```

Example 30–29 Copying `na` Audit Records to a Summary File

In this example, all the records of nonattributable audit events in the audit trail are collected into one file.

```
$ whoami
sysadmin
$ cd /var/audit/audit_summary.dir
$ auditreduce -c na -O nasumm
$ ls *nasumm
20030827183214.20030827215318.nasumm
```

The merged `nasumm` audit file is time stamped with the beginning and ending date of the `na` records.

Example 30–30 Finding Audit Events in a Specified Audit File

You can select audit files manually to search just the named set of files. For example, you can further process the `*nasumm` file in the previous example to find system boot events. To do so, you would specify the file name as the final argument to the `auditreduce` command.

```
$ auditreduce -m 113 -O systemboot 20030827183214.20030827215318.nasumm
20030827183214.20030827183214.systemboot
```

The `20030827183214.20030827183214.systemboot` file contains only system boot audit events.

Example 30–31 Copying One User's Audit Records to a Summary File

In this example, the records in the audit trail that contain the name of a particular user are merged. The `-e` option finds the effective user. The `-u` option finds the audit user.

```
$ cd /var/audit/audit_summary.dir
$ auditreduce -e tamiko -O tamiko
```

You can look for specific events in this file. In the following example, what time the user logged in and out on Sept 7, 2003, your time, is checked. Only those files with the user's name as the file suffix are checked. The short form of the date is *yyyymmdd*.

```
# auditreduce -M tamiko -O tamikolo -d 20030907 -u tamiko -c lo
```

Example 30–32 Copying Selected Records to a Single File

In this example, login and logout messages for a particular day are selected from the audit trail. The messages are merged into a target file. The target file is written in a directory other than the normal audit root directory.

```
# auditreduce -c lo -d 20030827 -O /var/audit/audit_summary.dir/logins
# ls /var/audit/audit_summary.dir/*logins
/var/audit/audit_summary.dir/20030827183936.20030827232326.logins
```

▼ How to View the Contents of Binary Audit Files

The `praudit` command enables you to view the contents of binary audit files. You can pipe the output from the `auditreduce` command, or you can read a particular audit file. The `-x` option is useful for further processing.

1. **Assume a role that includes the Audit Review profile, or become superuser.**

   The System Administrator role includes the Audit Review profile. You can also create a separate role that includes the Audit Review profile. To create a role and assign the role to a user, see "Configuring RBAC (Task Map)" on page 196.

2. **Use one of the following `praudit` commands to produce the output that is best for your purposes.**

   The following examples show `praudit` output from the same audit event. Audit policy has been set to include the `sequence` and `trailer` tokens.

- The `praudit -s` command displays audit records in a short format, one token per line. Use the `-l` option to place each record on one line.

  ```
  $ auditreduce -c lo | praudit -s
  header,101,2,AUE_rlogin,,example1,2003-10-13 11:23:31.050 -07:00
  subject,jdoe,jdoe,staff,jdoe,staff,749,749,195 1234 server1
  text,successful login
  return,success,0
  sequence,1298
  ```

- The `praudit -r` command displays audit records in their raw format, one token per line. Use the `-l` option to place each record on one line.

  ```
  $ auditreduce -c lo | praudit -r
  21,101,2,6155,0x0000,192.168.60.83,1062021202,64408258
  36,2026700,2026700,10,2026700,10,749,749,195 1234 192.168.60.17
  40,successful login
  39,0,0
  47,1298
  ```

- The `praudit -x` command displays audit records in XML format, one token per line. Use the `-l` option to place the XML output for one record on one line.

  ```
  $ auditreduce -c lo | praudit -x
  <record version="2" event="login - rlogin" host="example1"
  time="Wed Aug 27 14:53:22 PDT 2003" msec="64">
  <subject audit-uid="jdoe" uid="jdoe" gid="staff" ruid="jdoe"
  rgid="staff" pid="749" sid="749" tid="195 1234 server1"/>
  <text>successful login</text>
  <return errval="success" retval="0"/>
  <sequence seq-num="1298"/>

  </record>
  ```

Example 30–33 Printing the Entire Audit Trail

With a pipe to the `lp` command, the output for the entire audit trail goes to the printer. The printer should have limited access.

```
# auditreduce | praudit | lp -d example.protected.printer
```

Example 30–34 Viewing a Specific Audit File

In this example, a summary login file is examined in a terminal window.

```
# cd /var/audit/audit_summary.dir/logins
# praudit 20030827183936.20030827232326.logins | more
```

Example 30–35 Putting Audit Records in XML Format

In this example, the audit records are converted to XML format.

```
# praudit -x 20030827183214.20030827215318.logins > 20030827.logins.xml
```

The `*xml` file can be displayed in a browser. The contents of the file can be operated on by a script to extract the relevant information.

▼ How to Clean Up a `not_terminated` Audit File

Occasionally, an audit daemon exits while its audit file is still open. Or, a server becomes inaccessible and forces the machine to switch to a new server. In such instances, an audit file remains with the string `not_terminated` as the end timestamp, even though the file is no longer used for audit records. Use the `auditreduce -O` command to give the file the correct timestamp.

1. **List the files with the `not_terminated` string on your audit file system in order of creation.**

   ```
   # ls -R1t audit-directory*/files/* | grep not_terminated
   ```

`-R`	Lists files in subdirectories.
`-t`	Lists files from most recent to oldest.
`-1`	Lists the files in one column.

2. **Clean up the old `not_terminated` file.**

   Specify the name of the old file to the `auditreduce -O` command.

   ```
   # auditreduce -O system-name old-not-terminated-file
   ```

3. **Remove the old `not_terminated` file.**

   ```
   # rm system-name old-not-terminated-file
   ```

Example 30–36 Cleaning Up Closed `not_terminated` Audit Files

In the following example, `not_terminated` files are found, renamed, then the originals are removed.

```
ls -R1t */files/* | grep not_terminated
.../egret.1/20030908162220.not_terminated.egret
.../egret.1/20030827215359.not_terminated.egret
# cd */files/egret.1
# auditreduce -O egret 20030908162220.not_terminated.egret
# ls -1t
```

```
20030908162220.not_terminated.egret       Current audit file
20030827230920.20030830000909.egret       Input (old) audit file
20030827215359.not_terminated.egret
# rm 20030827215359.not_terminated.egret
# ls -1t
20030908162220.not_terminated.egret       Current audit file
20030827230920.20030830000909.egret       Cleaned up audit file
```

The start timestamp on the new file reflects the time of the first audit event in the `not_terminated` file. The end timestamp reflects the time of the last audit event in the file.

▼ How to Prevent Audit Trail Overflow

If your security policy requires that all audit data be saved, do the following:

1. **Set up a schedule to regularly archive audit files.**

   Archive audit files by backing up the files to offline media. You can also move the files to an archive file system.

   If you are collecting text audit logs with the `syslog` utility, archive the text logs. For more information, see the `logadm`(1M) man page.

2. **Set up a schedule to delete the archived audit files from the audit file system.**

3. **Save and store auxiliary information.**

   Archive information that is necessary to interpret audit records along with the audit trail.

4. **Keep records of which audit files have been archived.**

5. **Store the archived media appropriately.**

6. **Reduce the volume of audit data that you store by creating summary files.**

   You can extract summary files from the audit trail by using options to the `auditreduce` command. The summary files contain only records for specified types of audit events. To extract summary files, see Example 30–28 and Example 30–32.

CHAPTER 31

Solaris Auditing (Reference)

This chapter describes the important components of Solaris auditing. The following is a list of the reference information in this chapter.

- "Audit Commands" on page 605
- "Files Used in the Auditing Service" on page 611
- "Rights Profiles for Administering Auditing" on page 616
- "Auditing and Solaris Zones" on page 616
- "Audit Classes" on page 617
- "Audit Policy" on page 620
- "Process Audit Characteristics" on page 620
- "Audit Trail" on page 620
- "Conventions for Binary Audit File Names" on page 621
- "Audit Record Structure" on page 622
- "Audit Token Formats" on page 623

For an overview of Solaris auditing, see Chapter 28, "Solaris Auditing (Overview)." For planning suggestions, see Chapter 29, "Planning for Solaris Auditing." For procedures to configure auditing at your site, see Chapter 30, "Managing Solaris Auditing (Tasks)."

Audit Commands

This section provides information about the following commands:

- "`auditd` Daemon" on page 606
- "`audit` Command" on page 606
- "`bsmrecord` Command" on page 607
- "`auditreduce` Command" on page 607
- "`praudit` Command" on page 609
- "`auditconfig` Command" on page 610

`auditd` Daemon

The following list summarizes what the `auditd` daemon does.

- The `auditd` daemon opens and closes audit files in the directories that are specified in the `audit_control` file. The files are opened in order.
- The `auditd` daemon loads one or more plugins. Sun provides two plugins. The `/lib/security/audit_binfile.so.1` plugin writes binary audit data to a file. The `/lib/security` `audit_syslog.so.1` plugin sends text summaries of audit records to the `syslogd` daemon.
- The `auditd` daemon reads audit data from the kernel and outputs the data by using an `auditd` plugin.
- The `auditd` daemon executes the `audit_warn` script to warn of configuration errors. The `binfile.so.1` plugin executes the `audit_warn` script. The script, by default, sends warnings to the `audit_warn` email alias and to the console. The `syslog.so.1` plugin does not execute the `audit_warn` script.
- By default, when all audit directories are full, processes that generate audit records are suspended. In addition, the `auditd` daemon writes a message to the console and to the `audit_warn` email alias. At this point, only the system administrator can fix the auditing service. The administrator can log in to write audit files to offline media, delete audit files from the system, and do other cleanup tasks.

  The audit policy can be reconfigured with the `auditconfig` command.

The `auditd` daemon can be started automatically when the system is brought up to multiuser mode. Or, you can start the daemon from the command line. When the `auditd` daemon is started, it calculates the amount of free space necessary for audit files.

The `auditd` daemon uses the list of audit directories in the `audit_control` file as possible locations for creating audit files. The daemon maintains a pointer into this list of directories, starting with the first directory. Every time the `auditd` daemon needs to create an audit file, the daemon puts the file into the first available directory in the list. The list starts at the `auditd` daemon's current pointer. You can reset the pointer to the beginning of the list by running the `audit -s` command. The `audit -n` command instructs the daemon to switch to a new audit file. The new file is created in the same directory as the current file.

`audit` Command

The `audit` command controls the actions of the `auditd` daemon. The `audit` command can do the following tasks:

- Enable and disable auditing
- Reset the `auditd` daemon
- Adjust the auditing preselection mask on the local system

- Write audit records to a different audit file

For a discussion of the available options, see the `audit`(1M) man page.

`bsmrecord` Command

The `bsmrecord` command displays the format of audit events that are defined in the `/etc/security/audit_event` file. The output includes the event's audit ID, audit class, audit flag, and the record's audit tokens in order. With no option, the `bsmrecord` output displays in a terminal window. With the `-h` option, the output is suitable for viewing in a browser. For examples of the use of the `bsmrecord` command, see "How to Display Audit Record Formats" on page 594. Also, see the `bsmrecord`(1M) man page.

`auditreduce` Command

The `auditreduce` command summarizes audit records that are stored in binary format. The command can merge audit records from one or more input audit files. The command can also be used to perform a post selection of audit records. The records remain in binary format. To merge the entire audit trail, run this command on the audit server. The audit server is the system that mounts all the audit file systems for the installation. For more information, see the `auditreduce`(1M) man page.

The `auditreduce` command enables you to track all audited actions on multiple systems from a single location. The command can read the logical combination of all audit files as a single audit trail. You must identically configure all systems at a site for auditing, and create servers and local directories for the audit files. The `auditreduce` command ignores how the records were generated or where the records are stored. Without options, the `auditreduce` command merges audit records from all the audit files in all of the subdirectories in the audit root directory. Typically, `/etc/security/audit` is the audit root directory. The `auditreduce` command sends the merged results to standard output. You can also place the results into a single, chronologically ordered output file. The file contains binary data.

The `auditreduce` command can also select particular types of records for analysis. The merging functions and selecting functions of the `auditreduce` command are logically independent. The `auditreduce` command captures data from the input files as the records are read, before the files are merged and then written to disk.

By specifying options to the `auditreduce` command, you can also do the following:

- Request audit records that were generated by specified audit classes
- Request audit records that were generated by one particular user
- Request audit records that were generated on specific dates

With no arguments, the `auditreduce` command checks the subdirectories within the `/etc/security/audit` directory, the default audit root directory. The command checks for a `files` directory in which the *start-time.end-time.hostname* files reside. The `auditreduce` command is very useful when audit data resides in separate directories. Figure 31–1 illustrates audit data in separate directories for different hosts. Figure 31–2 illustrates audit data in separate directories for different audit servers.

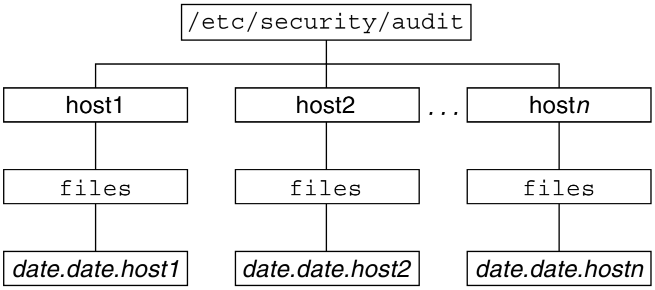

/etc/security/audit
host1
host2
. . .
hostn
files
files
files
date.date.host1
date.date.host2
date.date.hostn

FIGURE 31–1 Audit Trail Storage Sorted by Host

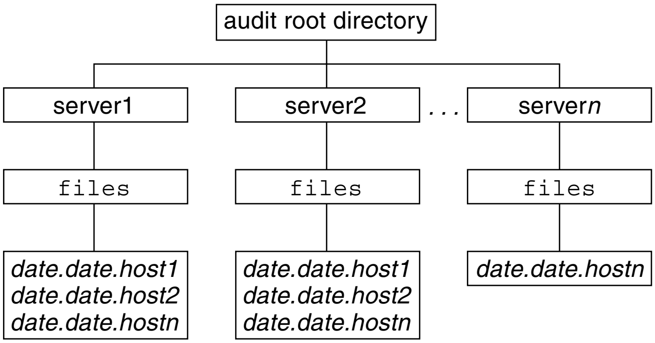

audit root directory
server1
server2
. . .
servern
files
files
files
date.date.host1
date.date.host2
date.date.hostn
date.date.host1
date.date.host2
date.date.hostn
date.date.hostn

FIGURE 31–2 Audit Trail Storage Sorted by Server

If the partition for the `/etc/security/audit` directory is very small, you might not store audit data in the default directory. You can pass the `auditreduce` command another directory by using the `-R` option:

```
# auditreduce -R /var/audit-alt
```

You can also specify a particular subdirectory by using the `-S` option:

```
# auditreduce -S /var/audit-alt/host1
```

For other options and more examples, see the `auditreduce`(1M) man page.

`praudit` Command

The `praudit` command makes the binary output of the `auditreduce` command readable. The `praudit` command reads audit records in binary format from standard input and displays the records in a presentable format. The input can be piped from the `auditreduce` command or from a single audit file. Input can also be produced with the `cat` command to concatenate several files, or the `tail` command for a current audit file.

The `praudit` command can generate four output formats. A fifth option, `-l` (long), prints one audit record per line of output. The default is to place one audit token per line of output. The `-d` option changes the delimiter that is used between token fields and between tokens. The default delimiter is a comma.

- **Default –** The `praudit` command with no options displays one audit token per line. The command displays the audit event by its description, such as the `ioctl(2)` system call. Any value that can be displayed as text is displayed in text format. For example, a user is displayed as the user name, not as the user ID.
- **–r option –** The raw option displays as a number any value that could be numeric. For example, a user is displayed by user ID, Internet addresses are in hexadecimal format, and modes are in octal format. The audit event is displayed as its event number, such as `158`.
- **–s option –** The short option displays the audit event by its table name, for example, `AUE_IOCTL`. The option displays the other tokens as the default option displays them.
- **–x option –** The XML option displays the audit record in XML format. This option is useful as input to browsers, or as input to scripts that manipulate XML.

  The XML is described by a DTD that the auditing service provides. Solaris software also provides a style sheet. The DTD and the style sheet are in the `/usr/share/lib/xml` directory.

In the default output format of the `praudit` command, each record is easily identified as a sequence of audit tokens. Each token is presented on a separate line. Each record begins with a `header` token. You could, for example, further process the output with the `awk` command.

Here is the output from the `praudit -l` command for a `header` token:

```
header,173,2,settppriv(2),,example1,2003-10-13 13:46:02.174 -07:00
```

Here is the output from the `praudit -r` command for the same `header` token:

```
121,173,2,289,0x0000,192.168.86.166,1066077962,174352445
```

EXAMPLE 31–1 Processing `praudit` Output With a Script

You might want to process output from the `praudit` command as lines of text. For example, you might want to select records that the `auditreduce` command cannot select. You can use a

EXAMPLE 31–1 Processing `praudit` Output With a Script *(Continued)*

simple shell script to process the output of the `praudit` command. The following simple example script puts one audit record on one line, searches for a user-specified string, then returns the audit file to its original form.

```
#!/bin/sh
#
## This script takes an argument of a user-specified string.
#  The sed command prefixes the header tokens with Control-A
#  The first tr command puts the audit tokens for one record
#  onto one line while preserving the line breaks as Control-A
#
praudit | sed -e ’1,2d’ -e ’$s/^file.*$//’ -e ’s/^header/^aheader/’ \\
| tr ’\\012\\001’ ’\\002\\012’ \\
| grep "$1" \\        Finds the user-specified string
| tr ’\\002’ ’\\012’       Restores the original newline breaks
```

Note that the `^a` in the script is Control-A, not the two characters `^` and `a`. The prefix distinguishes the `header` token from the string `header` that might appear as text.

`auditconfig` Command

The `auditconfig` command provides a command-line interface to retrieve and set audit configuration parameters. The `auditconfig` command can do the following tasks:

- Display, check, and configure audit policy
- Determine if auditing is turned on or turned off
- Turn auditing off and turn auditing on
- Manage the audit directory and the audit file
- Manage the audit queue
- Get and set preselection masks
- Get and set audit event to audit class mappings
- Get and set configuration information, such as session ID and audit ID
- Configure audit characteristics for a process, a shell, and a session
- Reset audit statistics

For a discussion of the command options, see the `auditconfig`(1M) man page.

Files Used in the Auditing Service

The auditing service uses the following files:

- "`system` File" on page 611
- "`syslog.conf` File" on page 611
- "`audit_class` File" on page 611
- "`audit_control` File" on page 612
- "`audit_event` File" on page 613
- "`audit_startup` Script" on page 613
- "`audit_user` Database" on page 613
- "`audit_warn` Script" on page 614
- "`bsmconv` Script" on page 616

`system` File

The `/etc/system` file contains commands that the kernel reads during initialization to customize the system operations. The `bsmconv` and `bsmunconv` shell scripts, which are used to activate and deactivate auditing, modify the `/etc/system` file. The `bsmconv` shell script adds the following line to the `/etc/system` file:

```
set c2audit:audit_load=1
```

The `set c2audit:audit_load=1` entry causes the kernel module for auditing to be loaded when the system is booted. The `bsmunconv` shell script disables auditing when the system is rebooted. The command removes the `c2audit` line from the `/etc/system` file.

`syslog.conf` File

The `/etc/syslog.conf` file works with the `audit_control` file to store audit records in text format. The `syslog.conf` file can be configured to enable the `syslog` utility to store audit records. For an example, see "How to Configure `syslog` Audit Logs" on page 573.

`audit_class` File

The `/etc/security/audit_class` file defines the audit classes. Audit classes are groups of audit events. You use the class name in the `audit_control` file to preselect the classes whose events you want to audit. The classes accept prefixes to select only failed events or only successful events. For more information, see "Audit Class Syntax" on page 618.

The superuser, or an administrator in an equivalent role, can modify the definitions of audit classes. This administrator can define new audit classes, rename existing classes, or otherwise change existing classes by editing the `audit_class` file in a text editor. For more information, see the `audit_class`(4) man page.

`audit_control` File

The `/etc/security/audit_control` file on each system contains configuration information for the `auditd` daemon. The file enables every system to mount a remote audit file system to store their audit records.

You can specify five kinds of information in the `audit_control` file. Each line of information begins with a keyword.

- `flags` **keyword** – Begins the entry that preselects which classes of events are audited for all users on the system. The audit classes that are specified here determine the *system-wide audit preselection mask*. The audit classes are separated by commas.
- `naflags` **keyword** – Begins the entry that preselects which classes of events are audited when an action cannot be attributed to a specific user. The audit classes are separated by commas. The `na` event class belongs in this entry. The `naflags` entry can be used to log other event classes that are normally attributable but cannot be attributed. For example, if a program that starts at boot reads a file, then an `fr` in the `naflags` entry would create a record for that event.
- `minfree` **keyword** – Begins the entry that defines the minimum free-space level for all audit file systems. The `minfree` percentage must be equal to `0` or greater than `0`. The default is 20 percent. When an audit file system is 80 percent full, the audit data is then stored in the next available audit directory. For more information, see the `audit_warn`(1M) man page.
- `dir` **keyword** – Begins the *directory definition* lines. Each line defines an audit file system and directory that the system uses to store its audit files. You can define one or more directory definition lines. The order of the `dir` lines is significant. The `auditd` daemon creates audit files in the directories in the specified order. The first directory is the *primary audit directory* for the system. The second directory is the *secondary audit directory* where the `auditd` daemon creates audit files when the first directory becomes full, and so on. For more information, see the `audit`(1M) man page.
- `plugin` **keyword** – Specifies the *plugin path* and the audit classes for the `syslog` plugin module. The module provides real-time conversion of Solaris audit records to text. The audit classes in the `plugin` line must be a subset of the audit classes in the `flags` line and the `naflags` line.

For more information about the `audit_control` file, see the `audit_control`(4) man page.

EXAMPLE 31–2 Sample `audit_control` File

The following is a sample `audit_control` file for the system `noddy`. `noddy` uses two audit file systems on the audit server `blinken`, and a third audit file system that is mounted from the second audit server `winken`. The third file system is used only when the audit file systems on `blinken` become full or unavailable. The `minfree` value of 20 percent specifies that the warning script is run when the file systems are 80 percent full. The settings specify that logins and administrative operations are to be audited. The operations are audited for success and for

EXAMPLE 31–2 Sample `audit_control` File *(Continued)*

failure. Failures of all types, except failures to create a file system object, are to be audited. Nonattributable events are also audited. The `syslog` audit log records fewer audit events. This log contains text summaries of failed logins and failed administrative operations.

```
flags:lo,am,-all,^-fc
naflags:lo,nt
minfree:20
dir:/etc/security/audit/blinken/files
dir:/etc/security/audit/blinken.1/files
#
# Audit filesystem used when blinken fills up
#
dir:/etc/security/audit/winken
plugin:name=audit_syslog.so.1; p_flags=-lo,-am
```

`audit_event` File

The `/etc/security/audit_event` file contains the default audit event-class mappings. You can edit this file to change the class mappings. When you change class mappings, you must reboot the system or run the `auditconfig -conf` command to read the changed mappings into the kernel. For more information, see the `audit_event`(4) man page.

`audit_startup` Script

The `/etc/security/audit_startup` script automatically configures the auditing service when the system enters multiuser mode. The `auditd` daemon starts after the script performs the following tasks:

- Configures the audit event-class mappings
- Sets the audit policy options

For more information, see the `audit_startup`(1M) man page.

`audit_user` Database

The `/etc/security/audit_user` database modifies the system-wide preselected classes for an individual user. The classes that you add to a user's entry in the `audit_user` database modify the settings in the `audit_control` file in two ways:

- By specifying audit classes that are always to be audited for this user
- By specifying audit classes that are never to be audited for this user

Each user entry in the `audit_user` database contains three fields:

username`:`*always-audit-classes*`:`*never-audit-classes*

The audit fields are processed in sequence. The *always-audit-classes* field turns on the auditing of the classes in that field. The *never-audit-classes* field turns off the auditing of the classes in that field.

Note – Avoid the common mistake of placing the `all` audit class in the *never-audit-classes* field. This mistake causes all auditing to be turned off for that user, which overrides the settings in the *always-audit-classes* field. The setting also overrides system-wide audit class settings in the `audit_control` file.

The *never-audit-classes* settings for a user override the system defaults. You might not want to override system defaults. For example, suppose you want to audit everything for user `tamiko`, except for successful reads of file system objects. You also want to apply the system defaults to `tamiko`. Note the placement of the second colon (`:`) in the following `audit_user` entries:

`tamiko:all,^+fr:`	*correct entry*

The correct entry means, "always audit everything, except for successful file-reads."

`tamiko:all:+fr`	*incorrect entry*

The incorrect entry means, "always audit everything, but never audit successful file-reads." The *never-audit-classes* field, which follows the second colon, overrides the system defaults. In the correct entry, the *always-audit-classes* field includes the exception to the `all` audit class. Because no audit class is in the *never-audit-classes* field, the system defaults from the `audit_control` file are not overridden.

Note – Successful events and failed events are treated separately. A process could generate more audit records for failed events than for successful events.

`audit_warn` Script

The `/etc/security/audit_warn` script notifies an email alias when the `auditd` daemon encounters an unusual condition while writing audit records. You can customize this script for your site to warn of conditions that might require manual intervention. Or, you could specify how to handle those conditions automatically. For all error conditions, the `audit_warn` script writes a message to `syslog` with the severity of `daemon.alert`. You can use `syslog.conf` to configure console display of `syslog` messages. The `audit_warn` script also sends a message to the `audit_warn` email alias. You should set up this alias when you enable auditing.

When the `auditd` daemon detects the following conditions, the daemon invokes the `audit_warn` script. The script sends email to the `audit_warn` alias.

- An audit directory has become more full than the `minfree` value allows. The `minfree` value or soft limit is a percentage of the available space on an audit file system.

  The `audit_warn` script is invoked with the string `soft` and the name of the directory whose available space is below the minimum value. The `auditd` daemon switches automatically to the next suitable directory. The daemon writes the audit files in this new directory until the directory reaches its `minfree` limit. The `auditd` daemon then goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is at its `minfree` limit.
- All the audit directories have reached the `minfree` threshold.

  The `audit_warn` script is invoked with the string `allsoft`. A message is written to the console. Email is also sent to the `audit_warn` alias.

  When all audit directories that are listed in the `audit_control` file have reached their `minfree` threshold, the `auditd` daemon switches back to the first directory. The daemon writes audit records until the directory becomes completely full.
- An audit directory has become completely full with no space remaining.

  The `audit_warn` script is invoked with the string `hard` and the name of the directory. A message is written to the console. Email is also sent to the `audit_warn` alias.

  The `auditd` daemon switches automatically to the next suitable directory with any space available. The `auditd` daemon goes to each remaining directory in the order that is listed in the `audit_control` file. The daemon writes audit records until each directory is full.
- All the audit directories are completely full. The `audit_warn` script is invoked with the string `allhard` as an argument.

  By default, a message is written to the console. Email is also sent to the `audit_warn` alias. Processes that would otherwise generate audit records continue to occur, but audit records are counted. Audit records are not generated. For an example of how to handle this situation, see Example 30–14 and "How to Prevent Audit Trail Overflow" on page 603.
- An internal error occurs. Possible internal errors include the following:
  - `ebusy` – Another `auditd` daemon process is already running
  - `tmpfile` – A temporary file cannot be used
  - `postsigterm` – A signal was received during auditing shutdown
- A problem is discovered with the syntax of the `audit_control` file. By default, a message is sent to the console. Email is also sent to the `audit_warn` alias.

For further information, see the `audit_warn`(1M) man page.

`bsmconv` Script

The `/etc/security/bsmconv` script enables the auditing service. The `bsmunconv` command disables the auditing service. After the `bsmconv` script is run, you configure the audit directories and audit configuration files. Upon reboot, auditing is enabled.

For further information, see the `bsmconv`(1M) man page.

Rights Profiles for Administering Auditing

The Solaris OS provides rights profiles for configuring the auditing service and for analyzing the audit trail.

- **Audit Control –** Enables a role to configure Solaris auditing. This rights profile grants authorizations to configure files that are used by the auditing service. The profile also enables a role to run audit commands. A role with the Audit Control profile can run the following commands: `audit`, `auditd`, `auditconfig`, `bsmconv`, and `bsmunconv`.
- **Audit Review –** Enables a role to analyze Solaris audit records. This rights profile grants authorization to read audit records with the `praudit` and `auditreduce` commands. A role with this rights profile can also run the `auditstat` command.
- **System Administrator –** Includes the Audit Review rights profile. A role with the System Administrator rights profile can analyze audit records.

To configure roles to handle the auditing service, see "Configuring RBAC (Task Map)" on page 196.

Auditing and Solaris Zones

Non-global zones can be audited exactly as the global zone is audited, or nonglobal zones can set their own flags, storage, and audit policy.

When all zones are being audited identically, the configuration files in the global zone provide the settings for auditing in every zone. The `+zonename` policy option is useful. When this option is set, the audit records from all zones include the name of the zone. Audit records can then be postselected by zone name. To understand audit policy, see "Determining Audit Policy" on page 564. For an example, see "How to Configure Audit Policy" on page 584.

Zones can also be audited individually. When the policy option, `perzone`, is set in the global zone, each non-global zone runs its own audit daemon, handles its own audit queue, and specifies the content and location of its audit records. A non-global zone can also set most audit policy options. It cannot set policy that affects the entire system, so a non-global zone cannot set the `ahlt` or `perzone` policy. For further discussion, see "Auditing on a System With Zones" on page 555 and "How to Plan Auditing in Zones" on page 560.

To learn about zones, see Part II, “Zones,” in *System Administration Guide: Virtualization Using the Solaris Operating System*.

Audit Classes

System-wide defaults for Solaris auditing are preselected by specifying one or more classes of events. The classes are preselected for each system in the system's `audit_control` file. Anyone who uses the system is audited for these classes of events. The file is described in “`audit_control` File” on page 612.

You can configure audit classes and make new audit classes. Audit class names can be up to 8 characters in length. The class description is limited to 72 characters. Numeric and non-alphanumeric characters are allowed.

You can modify what is audited for individual users by adding audit classes to a user's entry in the `audit_user` database. The audit classes are also used as arguments to the `auditconfig` command. For details, see the `auditconfig`(1M) man page.

Definitions of Audit Classes

The following table shows each predefined audit class, the descriptive name for each audit class, and a short description.

TABLE 31–1 Predefined Audit Classes

Audit Class	Descriptive Name	Description
`all`	`all`	All classes (meta-class)
`no`	`no_class`	Null value for turning off event preselection
`na`	`non_attrib`	Nonattributable events
`fr`	`file_read`	Read of data, open for reading
`fw`	`file_write`	Write of data, open for writing
`fa`	`file_attr_acc`	Access of object attributes: `stat`, `pathconf`
`fm`	`file_attr_mod`	Change of object attributes: `chown`, `flock`
`fc`	`file_creation`	Creation of object
`fd`	`file_deletion`	Deletion of object
`cl`	`file_close`	`close` system call

TABLE 31–1 Predefined Audit Classes *(Continued)*

Audit Class	Descriptive Name	Description
`ap`	`application`	Application-defined event
`ad`	`administrative`	Administrative actions (old administrative meta-class)
`am`	`administrative`	Administrative actions (meta-class)
`ss`	`system state`	Change system state
`as`	`system-wide administration`	System-wide administration
`ua`	`user administration`	User administration
`aa`	`audit administration`	Audit utilization
`ps`	`process start`	Process start and process stop
`pm`	`process modify`	Process modify
`pc`	`process`	Process (meta-class)
`ex`	`exec`	Program execution
`io`	`ioctl`	`ioctl()` system call
`ip`	`ipc`	System V IPC operations
`lo`	`login_logout`	Login and logout events
`nt`	`network`	Network events: `bind`, `connect`, `accept`
`ot`	`other`	Miscellaneous, such as device allocation and `memcntl()`

You can define new classes by modifying the `/etc/security/audit_class` file. You can also rename existing classes. For more information, see the `audit_class`(4) man page.

Audit Class Syntax

Events can be audited for success, events can be audited for failure, and events can be audited for both. Without a prefix, a class of events is audited for success and for failure. With a plus (+) prefix, a class of events is audited for success only. With a minus (`-`) prefix, a class of events is audited for failure only. The following table shows some possible representations of audit classes.

TABLE 31–2 Plus and Minus Prefixes to Audit Classes

[*prefix*]*class*	**Explanation**
`lo`	Audit all successful attempts to log in and log out, and all failed attempts to log in. A user cannot fail an attempt to log out.
`+lo`	Audit all successful attempts to log in and log out.
`-all`	Audit all failed events.
`+all`	Audit all successful events.

Caution – The `all` class can generate large amounts of data and quickly fill audit file systems. Use the `all` class only if you have extraordinary reasons to audit all activities.

Audit classes that were previously selected can be further modified by a caret prefix, `^`. The following table shows how the caret prefix modifies a preselected audit class.

TABLE 31–3 Caret Prefix That Modifies Already-Specified Audit Classes

^[*prefix*]*class*	**Explanation**
`-all,^-fc`	Audit all failed events, except do not audit failed attempts to create file objects
`am,^+aa`	Audit all administrative events for success and for failure, except do not audit successful attempts to administer auditing
`am,^ua`	Audit all administrative events for success and for failure, except do not audit user administration events

The audit classes and their prefixes can be used in the following files and commands:

- In the `flags` line in the `audit_control` file
- In the `plugin ...p_flags=` line in the `audit_control` file
- In the user's entry in the `audit_user` database
- As arguments to `auditconfig` command options

See "`audit_control` File" on page 612 for an example of using the prefixes in the `audit_control` file.

Audit Policy

Audit policy determines if additional information is added to the audit trail. The effects of the different audit policy options are described in "Determining Audit Policy" on page 564.

Process Audit Characteristics

The following audit characteristics are set at initial login:

- **Process preselection mask –** A combination of the audit classes from the `audit_control` file and the `audit_user` database. When a user logs in, the login process combines the preselected classes to establish the *process preselection mask* for the user's processes. The process preselection mask specifies whether events in each audit class are to generate audit records.

  The following algorithm describes how the system obtains the user's process preselection mask:

  ```
  (flags line + always-audit-classes) - never-audit-classes
  ```

  Add the audit classes from the `flags` line in the `audit_control` file to the classes from the *always-audit-classes* field in the user's entry in the `audit_user` database. Then, subtract from the total the classes from the user's *never-audit-classes* field.
- **Audit ID –** A process acquires an audit ID when the user logs in. The audit ID is inherited by all child processes that were started by the user's initial process. The audit ID helps enforce accountability. Even after a user becomes `root`, the audit ID remains the same. The audit ID that is saved in each audit record always allows you to trace actions back to the original user who had logged in.
- **Audit Session ID –** The audit session ID is assigned at login. The session ID is inherited by all child processes.
- **Terminal ID (port ID, machine ID) –** The terminal ID consists of the host name and the Internet address, followed by a unique number that identifies the physical device on which the user logged in. Most often, the login is through the console. The number that corresponds to the console device is `0`.

Audit Trail

The *audit trail* is contains binary audit files. The trail is created by the `auditd` daemon. Once the auditing service has been enabled with the `bsmconv` command, the `auditd` daemon starts when the system is booted. The `auditd` daemon is responsible for collecting the audit trail data and writing the audit records.

The audit records are stored in binary format on file systems that are dedicated to audit files. Even though you can physically locate audit directories within file systems that are not dedicated to auditing, do *not* do so except for directories of last resort. Directories of last resort are directories where audit files are written only when no other suitable directory is available.

There is one other scenario where locating audit directories outside of dedicated audit file systems could be acceptable. You might do so in a software development environment where auditing is optional. To make full use of disk space might be more important than to keep an audit trail. However, in a security-conscious environment, the placement of audit directories within other file systems is not acceptable.

You should also consider the following factors when administering audit file systems:

- A host should have at least one local audit directory. The local directory can be used as a directory of last resort if the host is unable to communicate with the audit server.
- Mount audit directories with the read-write (`rw`) option. When you mount audit directories remotely, also use the `intr` and `noac` options.
- List the audit file systems on the audit server where they reside. The export list should include all systems that are being audited at the site.

Conventions for Binary Audit File Names

Each binary audit file is a self-contained collection of records. The file's name identifies the time span during which the records were generated and the system that generated them.

Binary Audit File Names

Audit files that are complete have names of the following form:

start-time.end-time.system

start-time	Is the time that the first audit record in the audit file was generated
end-time	Is the time that the last record was written to the file
system	Is the name of the system that generated the file

An audit file that is still active has a name of the following form:

start-time`.not_terminated.`*system*

For examples of `not_terminated` and closed audit file names, see “How to Clean Up a `not_terminated` Audit File” on page 602.

Binary Audit File Timestamps

The timestamps in file names are used by the `auditreduce` command to locate records within a specific time range. These timestamps are important because there can be a month's accumulation or more of audit files online. To search all the files for records that were generated in the last 24 hours would be unacceptably expensive.

The *start-time* and *end-time* are timestamps with one-second resolution. They are specified in Greenwich Mean Time (GMT). The format is four digits for the year, followed by two digits for each month, day, hour, minute, and second, as follows:

YYYYMMDDHHMMSS

The timestamps are in GMT to ensure that they sort in proper order, even across time zones. Because they are in GMT, the date and hour must be translated to the current time zone to be meaningful. Beware of this point whenever you manipulate these files with standard file commands rather than with the `auditreduce` command.

Audit Record Structure

An audit record is a sequence of audit tokens. Each audit token contains event information such as user ID, time, and date. A `header` token begins an audit record, and an optional `trailer` token concludes the record. Other audit tokens contain information relevant to the audit event. The following figure shows a typical audit record.

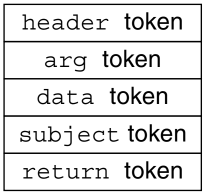


FIGURE 31–3 Typical Audit Record Structure

Audit Record Analysis

Audit record analysis involves postselecting records from the audit trail. You can use one of two approaches to parsing the binary data that was collected.

- You can parse the binary data stream. To parse the data stream, you need to know the order of the fields in each token, and the order of tokens in each record. You also need to know the variants of an audit record. For example, the `ioctl()` system call creates an audit record for “Bad file name” that contains different tokens from the audit record for “Invalid file descriptor”.

  - For a description of the order of binary data in each audit token, see the `audit.log`(4) man page.
  - For a description of the order of tokens in an audit record, use the `bsmrecord` command. Output from the `bsmrecord` command includes the different formats that occur under different conditions. Square brackets (`[]`) indicate that an audit token is optional. For more information, see the `bsmrecord`(1M) man page. For examples, see also "How to Display Audit Record Formats" on page 594.
- You can use the `praudit` command. Options to the command provide different text outputs. For example, the `praudit -x` command provides XML for input into scripts and browsers. `praudit` outputs do not include fields whose sole purpose is to help to parse the binary data. The outputs do not necessarily follow the order of the binary fields. Also, the order and format of `praudit` output is not guaranteed between Solaris releases.

  For examples of `praudit` output, see "How to View the Contents of Binary Audit Files" on page 600, and the `praudit`(1M) man page.

  For a description of the `praudit` output for each audit token, see the individual tokens in the "Audit Token Formats" on page 623 section.

Audit Token Formats

Each audit token has a token type identifier, which is followed by data that is specific to the token. Each token type has its own format. The following table shows the token names with a brief description of each token. Obsolete tokens are maintained for compatibility with previous Solaris releases.

TABLE 31–4 Audit Tokens for Solaris Auditing

Token Name	Description	For More Information
`acl`	Access Control List (ACL) information	"`acl` Token" on page 625
`arbitrary`	Data with format and type information	"`arbitrary` Token (Obsolete)" on page 625
`arg`	System call argument value	"`arg` Token" on page 626
`attribute`	File vnode tokens	"`attribute` Token" on page 626
`cmd`	Command arguments and environment variables	"`cmd` Token" on page 627
`exec_args`	Exec system call arguments	"`exec_args` Token" on page 627
`exec_env`	Exec system call environment variables	"`exec_env` Token" on page 628
`exit`	Program exit information	"`exit` Token (Obsolete)" on page 628
`file`	Audit file information	"`file` Token" on page 628

TABLE 31–4 Audit Tokens for Solaris Auditing *(Continued)*

Token Name	Description	For More Information
`group`	Process groups information	"`group` Token (Obsolete)" on page 629
`groups`	Process groups information	"`groups` Token" on page 629
`header`	Indicates start of audit record	"`header` Token" on page 629
`in_addr`	Internet address	"`in_addr` Token" on page 630
`ip`	IP header information	"`ip` Token (Obsolete)" on page 630
`ipc`	System V IPC information	"`ipc` Token" on page 631
`ipc_perm`	System V IPC object tokens	"`ipc_perm` Token" on page 631
`iport`	Internet port address	"`iport` Token" on page 632
`opaque`	Unstructured data (unspecified format)	"`opaque` Token (Obsolete)" on page 632
`path`	Path information	"`path` Token" on page 632
`path_attr`	Access path information	"`path_attr` Token" on page 633
`privilege`	Privilege set information	"`privilege` Token" on page 633
`process`	Process token information	"`process` Token" on page 634
`return`	Status of system call	"`return` Token" on page 635
`sequence`	Sequence number token	"`sequence` Token" on page 636
`socket`	Socket type and addresses	"`socket` Token" on page 636
`subject`	Subject token information (same format as `process` token)	"`subject` Token" on page 637
`text`	ASCII string	"`text` Token" on page 638
`trailer`	Indicates end of audit record	"`trailer` Token" on page 639
`uauth`	Use of authorization	"`uauth` Token" on page 639
`zonename`	Name of zone	"`zonename` Token" on page 639

An audit record always begins with a `header` token. The `header` token indicates where the audit record begins in the audit trail. In the case of attributable events, the `subject` and the `process` tokens refer to the values of the process that caused the event. In the case of nonattributable events, the `process` token refers to the system.

`acl` Token

The `acl` token records information about Access Control Lists (ACLs). This token consists of four fixed fields:

- A token ID that identifies this token as an `acl` token
- A field that specifies the ACL type
- An ACL value field
- A field that lists the permissions associated with this ACL

The `praudit` command displays the `acl` token as follows:

```
acl,jdoe,staff,0755
```

`arbitrary` Token (Obsolete)

The `arbitrary` token encapsulates data for the audit trail. This token consists of four fixed fields and an array of data. The fixed fields are as follows:

- A token ID that identifies this token as an `arbitrary` token
- A suggested print format field, such as hexadecimal
- An item size field that specifies the size of the data that is encapsulated, such as short
- A count field that provides the number of following items

The remainder of the token is composed of *count* of the specified type. The `praudit` command displays the `arbitrary` token as follows:

```
arbitrary,decimal,int,1
42
```

The following table shows the possible values of the print format field.

TABLE 31–5 Values for the `arbitrary` Token's Print Format Field

Value	Action
`AUP_BINARY`	Prints the date in binary format
`AUP_OCTAL`	Prints the date in octal format
`AUP_DECIMAL`	Prints the date in decimal format
`AUP_HEX`	Prints the date in hexadecimal format
`AUP_STRING`	Prints the date as a string

The following table shows the possible values of the item size field.

TABLE 31–6 Values for the `arbitrary` Token's Item Size Field

Value	Action
`AUR_BYTE`	Data is printed in units of bytes in 1 byte
`AUR_SHORT`	Data is printed in units of shorts in 2 bytes
`AUR_LONG`	Data is printed in units of longs in 4 bytes

`arg` Token

The `arg` token contains information about the arguments to a system call: the argument number of the system call, the argument value, and an optional description. This token allows a 32-bit integer system-call argument in an audit record. The `arg` token has five fields:

- A token ID that identifies this token as an `arg` token
- An argument ID that tells which system call argument that the token refers to
- The argument value
- The length of the descriptive text string
- The text string

The `praudit` command displays the `arg` token without the fourth field, as follows:

```
argument,4,0xffbfe0ac,pri
```

The `praudit -x` command includes the names of the fields that are displayed:

```
<argument arg-num="4" value="0xffbfe0ac" desc="pri"/>
```

`attribute` Token

The `attribute` token contains information from the file vnode. This token has seven fields:

- A token ID that identifies this token as an `attribute` token
- The file access mode and type
- The owner user ID
- The owner group ID
- The file system ID
- The node ID
- The device ID that the file might represent

For further information about the file system ID and the device ID, see the `statvfs`(2) man page.

The `attribute` token usually accompanies a `path` token. The `attribute` token is produced during path searches. If a path-search error occurs, there is no vnode available to obtain the necessary file information. Therefore, the `attribute` token is not included as part of the audit record. The `praudit` command displays the `attribute` token as follows:

```
attribute,20666,root,root,247,4829,450971566127
```

`cmd` Token

The `cmd` token records the list of arguments and the list of environment variables that are associated with a command.

The `cmd` token contains the following fields:

- A token ID that identifies this token as a `cmd` token
- A count of the command's arguments
- The argument list
- The length of the next field
- The content of the arguments
- A count of the environment variables
- The list of environment variables
- The length of the next field
- The content of the environment variables

The `praudit` command displays the `cmd` token as follows:

```
cmd,argcnt,3,ls,-l,/etc,envcnt,0,
```

`exec_args` Token

The `exec_args` token records the arguments to an `exec()` system call. The `exec_args` token has two fixed fields:

- A token ID field that identifies this token as an `exec_args` token
- A count that represents the number of arguments that are passed to the `exec()` system call

The remainder of this token is composed of *count* strings. The `praudit` command displays the `exec_args` token as follows:

```
exec_args,2,vi,/etc/security/audit_user
```

Note – The `exec_args` token is output only when the `argv` audit policy option is active.

`exec_env` Token

The `exec_env` token records the current environment variables to an `exec()` system call. The `exec_env` token has two fixed fields:

- A token ID field that identifies this token as an `exec_env` token
- A count that represents the number of arguments that are passed to the `exec()` system call

The remainder of this token is composed of *count* strings. The `praudit` command displays the `exec_env` token as follows:

```
exec_env,25,
GROUP=staff,HOME=/export/home/jdoe,HOST=exm1,HOSTTYPE=sun4u,HZ=100,
LC_COLLATE=en_US.ISO8859-1,LC_CTYPE=en_US.ISO8859-1,LC_MESSAGES=C,
LC_MONETARY=en_US.ISO8859-1,LC_NUMERIC=en_US.ISO8859-1,
LC_TIME=en_US.ISO8859-1,LOGNAME=jdoe,MACHTYPE=sparc,
MAIL=/var/mail/jdoe,OSTYPE=solaris,PATH=/usr/sbin:/usr/bin,PS1=#,
PWD=/var/audit,REMOTEHOST=192.168.13.5,SHELL=/usr/bin/csh,SHLVL=1,
TERM=dtterm,TZ=US/Pacific,USER=jdoe,VENDOR=sun
```

Note – The `exec_env` token is output only when the `arge` audit policy option is active.

`exit` Token (Obsolete)

The `exit` token records the exit status of a program. The `exit` token contains the following fields:

- A token ID that identifies this token as an `exit` token
- A program exit status as passed to the `exit()` system call
- A return value that describes the exit status or that provides a system error number

The `praudit` command displays the `exit` token as follows:

```
exit,Error 0,0
```

`file` Token

The `file` token is a special token that is generated by the `auditd` daemon. The token marks the beginning of a new audit file and the end of an old audit file as the old file is deactivated. The `auditd` daemon builds a special audit record that contains this token to “link” together successive audit files into one audit trail.

The `file` token has four fields:

- A token ID that identifies this token as a `file` token
- A timestamp that identifies the date and the time that the file was created or was closed
- The file name length
- A field that holds the file null-terminated name

The `praudit -x` command shows the fields of the `file` token:

```
file,2003-10-13 11:21:35.506 -07:00,
    /var/audit/localhost/files/20031013175058.20031013182135.example1
```

`group` Token (Obsolete)

This token has been replaced by the the `groups` token. See “`groups` Token” on page 629.

`groups` Token

The `groups` token replaces the `group` token. The `groups` token records the group entries from the process's credential. The `groups` token has two fixed fields:

- A token ID field that identifies this token as a `groups` token
- A count that represents the number of groups that are contained in this audit record

The remainder of this token is composed of *count* group entries. The `praudit` command displays the `groups` token as follows:

```
groups,staff,admin
```

Note – The `groups` token is output only when the `group` audit policy option is active.

`header` Token

The `header` token is special in that it marks the beginning of an audit record. The `header` token combines with the `trailer` token to bracket all the other tokens in the record.

The `header` token has eight fields:

- A token ID field that identifies this token as a `header` token
- A byte count of the total length of the audit record, including both the `header` and the `trailer` tokens

- A version number that identifies the version of the audit record structure
- The audit event ID that identifies the audit event that the record represents
- The ID modifier that identifies special characteristics of the audit event
- The address type, either IPv4 or IPv6
- The machine's IP address
- The time and date that the record was created

On 64-bit systems, the `header` token is displayed with a 64-bit timestamp, in place of the 32-bit timestamp.

The `praudit` command displays the `header` token for a `ioctl()` system call as follows:

```
header,176,2,ioctl(2),fe,example1,2003-09-08 11:23:31.050 -07:00
```

The ID modifier field has the following flags defined:

```
0x4000            PAD_NOTATTR                        nonattributable event
0x8000            PAD_FAILURE                        failed audit event
```

`in_addr` Token

The `in_addr` token contains an Internet Protocol address. Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address. The `in_addr` token has three fields:

- A token ID that identifies this token as an `in_addr` token
- The IP address type, either IPv4 or IPv6
- An IP address

The `praudit` command displays the `in_addr` token, without the second field, as follows:

```
ip address,192.168.113.7
```

`ip` Token (Obsolete)

The `ip` token contains a copy of an Internet Protocol header. The `ip` token has two fields:

- A token ID that identifies this token as an `ip` token
- A copy of the IP header, that is, all 20 bytes

The `praudit` command displays the `ip` token as follows:

```
ip address,0.0.0.0
```

The IP header structure is defined in the `/usr/include/netinet/ip.h` file.

`ipc` Token

The `ipc` token contains the System V IPC message handle, semaphore handle, or shared-memory handle that is used by the caller to identify a particular IPC object. The `ipc` token has three fields:

- A token ID that identifies this token as an `ipc` token
- A type field that specifies the type of IPC object
- The handle that identifies the IPC object

Note – The IPC object identifiers violate the context-free nature of the Solaris audit tokens. No global "name" uniquely identifies IPC objects. Instead, IPC objects are identified by their handles. The handles are valid only during the time that the IPC objects are active. However, the identification of IPC objects should not be a problem. The System V IPC mechanisms are seldom used, and the mechanisms all share the same audit class.

The following table shows the possible values for the IPC object type field. The values are defined in the `/usr/include/bsm/audit.h` file.

TABLE 31–7 Values for the IPC Object Type Field

Name	Value	Description
`AU_IPC_MSG`	1	IPC message object
`AU_IPC_SEM`	2	IPC semaphore object
`AU_IPC_SHM`	3	IPC shared-memory object

The `praudit` command displays the `ipc` token as follows:

```
IPC,msg,3
```

`ipc_perm` Token

The `ipc_perm` token contains a copy of the System V IPC access permissions. This token is added to audit records that are generated by IPC shared-memory events, IPC semaphore events, and IPC message events. The `ipc_perm` token has eight fields:

- A token ID that identifies this token as an `ipc_perm` token
- The user ID of the IPC owner

- The group ID of the IPC owner
- The user ID of the IPC creator
- The group ID of the IPC creator
- The access mode of the IPC
- The sequence number of the IPC
- The IPC key value

The `praudit` command displays the `ipc_perm` token as follows:

```
IPC perm,root,sys,root,sys,0,0,0x00000000
```

The values are taken from the `ipc_perm` structure that is associated with the IPC object.

`iport` Token

The `iport` token contains the TCP or UDP port address. The `iport` token has two fields:

- A token ID that identifies this token as an `iport` token
- The TCP or UDP port address

The `praudit` command displays the `iport` token as follows:

```
ip port,0xf6d6
```

`opaque` Token (Obsolete)

The `opaque` token contains unformatted data as a sequence of bytes. The `opaque` token has three fields:

- A token ID that identifies this token as an `opaque` token
- A byte count of the data
- An array of byte data

The `praudit` command displays the `opaque` token as follows:

```
opaque,12,0x4f5041515545204441544100
```

`path` Token

The `path` token contains access path information for an object. This token contains the following fields:

- A token ID that identifies this token as an `path` token
- The path length

- The absolute path to the object that is based on the real root of the system

The `praudit` command displays the `path` token, without the second field, as follows:

```
path,/etc/security/audit_user
```

The `praudit -x` command displays the `path` token as follows:

```
<path>/etc/security/audit_user</path>
```

The following figure shows the format of a `path` token.

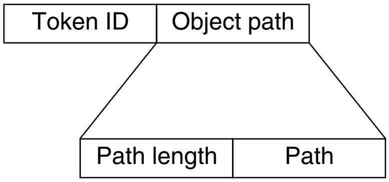


FIGURE 31–4 `path` Token Format

`path_attr` Token

The `path_attr` token contains access path information for an object. The access path specifies the sequence of attribute file objects below the `path` token object. Systems calls such as `openat()` access attribute files. For more information on attribute file objects, see the `fsattr`(5) man page.

The `path_attr` token contains the following fields:

- A token ID that identifies this token as a `path_attr` token
- A count that represents the number of sections of attribute file paths
- *count* null-terminated strings

The `praudit` command displays the `path_attr` token as follows:

```
path_attr,1,attr_file_name
```

`privilege` Token

The `privilege` token records the use of privileges on a process. The `privilege` token is not recorded for privileges in the basic set. If a privilege has been removed from the basic set by administrative action, then the use of that privilege is recorded. For more information on privileges, see "Privileges (Overview)" on page 185

The `privilege` token contains the following fields:

- A token ID that identifies this token as a `privilege` token
- The length of the following field
- The name of privilege set
- The length of the following field
- The list of privileges

The `praudit` command displays the `privilege` token as follows:

```
privilege,effective,
```

`process` Token

The `process` token contains information about a user who is associated with a process, such as the recipient of a signal. The `process` token has nine fields:

- A token ID that identifies this token as a `process` token
- The audit ID
- The effective user ID
- The effective group ID
- The real user ID
- The real group ID
- The process ID
- The audit session ID
- A terminal ID that consists of a device ID and a machine ID

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note – The `process` token fields for the session ID, the real user ID, or the real group ID might be unavailable. The value is then set to `-1`.

Any token that contains a terminal ID has several variations. The `praudit` command hides these variations. So, the terminal ID is handled the same way for any token that contains a terminal ID. The terminal ID is either an IP address and port number, or a device ID. A device ID, such as the serial port that is connected to a modem, can be zero. The terminal ID is specified in one of several formats.

The terminal ID for device numbers is specified as follows:

- **32-bit applications –** 4-byte device number, 4 bytes unused
- **64-bit applications –** 8-byte device number, 4 bytes unused

In releases prior to the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- **32-bit applications –** 4-byte port number, 4-byte IP address
- **64-bit applications –** 8-byte port number, 4-byte IP address

Since the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- **32-bit with IPv4 –** 4-byte port number, 4-byte IP type, 4-byte IP address
- **32-bit with IPv6 –** 4-byte port number, 4-byte IP type, 16-byte IP address
- **64-bit with IPv4 –** 8-byte port number, 4-byte IP type, 4-byte IP address
- **64-bit with IPv6 –** 8-byte port number, 4-byte IP type, 16-byte IP address

The `praudit` command displays the `process` token as follows:

```
process,root,root,sys,root,sys,0,0,0,0.0.0.0
```

The following figure shows the format of a `process` token.

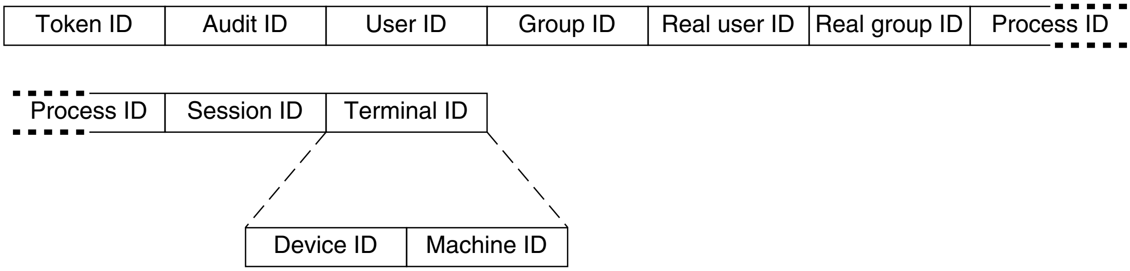


FIGURE 31–5 `process` Token Format

`return` Token

The `return` token contains the return status of the system call (`u_error`) and the process return value (`u_rval1`). This token has three fields:

- A token ID that identifies this token as a `return` token
- The error status of the system call
- The return value of the system call

The `return` token is always returned as part of kernel-generated audit records for system calls. In application auditing, this token indicates exit status and other return values.

The `praudit` command displays the `return` token for a system call as follows:

```
return,failure: Operation now in progress,-1
```

The `praudit -x` command displays the `return` token as follows:

```
<return errval="failure: Operation now in progress" retval="-1/">
```

`sequence` Token

The `sequence` token contains a sequence number. This token is useful for debugging. The `sequence` token has two fields:

- A token ID that identifies this token as a `sequence` token
- A 32-bit unsigned long field that contains the sequence number

The sequence number is incremented every time an audit record is added to the audit trail. The `praudit` command displays the `sequence` token as follows:

```
sequence,1292
```

The `praudit -x` command displays the `sequence` token as follows:

```
<sequence seq-num="1292"/>
```

Note – The `sequence` token is output only when the `seq` audit policy option is active.

`socket` Token

The `socket` token contains information that describes an Internet socket. In some instances, the token has four fields:

- A token ID that identifies this token as a `socket` token
- A socket type field that indicates the type of socket referenced, either TCP, UDP, or UNIX
- The local port
- The local IP address

The `praudit` command displays this instance of the `socket` token as follows:

```
socket,0x0002,0x83b1,localhost
```

In most instances, the token has eight fields:

- A token ID that identifies this token as a `socket` token
- The socket domain
- A socket type field that indicates the type of socket referenced, either TCP, UDP, or UNIX
- The local port
- The address type, either IPv4 or IPv6
- The local IP address
- The remote port

- The remote IP address

Since the Solaris 8 release, the Internet address can be displayed in IPv4 format or IPv6 format. The IPv4 address uses 4 bytes. The IPv6 address uses 1 byte to describe the address type, and 16 bytes to describe the address.

The `praudit` command displays the `socket` token as follows:

```
socket,0x0002,0x0002,0x83cf,example1,0x2383,server1.Subdomain.Domain.COM
```

The `praudit -x` command describes the `socket` token fields. The lines are wrapped for display purposes.

```
<socket sock_domain="0x0002" sock_type="0x0002" lport="0x83cf"
laddr="example1" fport="0x2383" faddr="server1.Subdomain.Domain.COM"/>
```

`subject` Token

The `subject` token describes a user who performs or attempts to perform an operation. The format is the same as the `process` token. The `subject` token has nine fields:

- A token ID that identifies this token as a `subject` token
- The audit ID
- The effective user ID
- The effective group ID
- The real user ID
- The real group ID
- The process ID
- The audit session ID
- A terminal ID that consists of a device ID and a machine ID

The audit ID, user ID, group ID, process ID, and session ID are long instead of short.

Note – The `subject` token fields for the session ID, the real user ID, or the real group ID might be unavailable. The value is then set to `-1`.

Any token that contains a terminal ID has several variations. The `praudit` command hides these variations. So, the terminal ID is handled the same way for any token that contains a terminal ID. The terminal ID is either an IP address and port number, or a device ID. A device ID, such as the serial port that is connected to a modem, can be zero. The terminal ID is specified in one of several formats.

The terminal ID for device numbers is specified as follows:

- **32-bit applications –** 4-byte device number, 4 bytes unused
- **64-bit applications –** 8-byte device number, 4 bytes unused

In releases prior to the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- **32-bit applications –** 4-byte port number, 4-byte IP address
- **64-bit applications –** 8-byte port number, 4-byte IP address

Since the Solaris 8 release, the terminal ID for port numbers is specified as follows:

- **32-bit with IPv4 –** 4-byte port number, 4-byte IP type, 4-byte IP address
- **32-bit with IPv6 –** 4-byte port number, 4-byte IP type, 16-byte IP address
- **64-bit with IPv4 –** 8-byte port number, 4-byte IP type, 4-byte IP address
- **64-bit with IPv6 –** 8-byte port number, 4-byte IP type, 16-byte IP address

The `subject` token is always returned as part of kernel-generated audit records for system calls. The `praudit` command displays the `subject` token as follows:

```
subject,jdoe,root,staff,root,staff,424,223,0 0 example1
```

The following figure shows the format of the `subject` token.

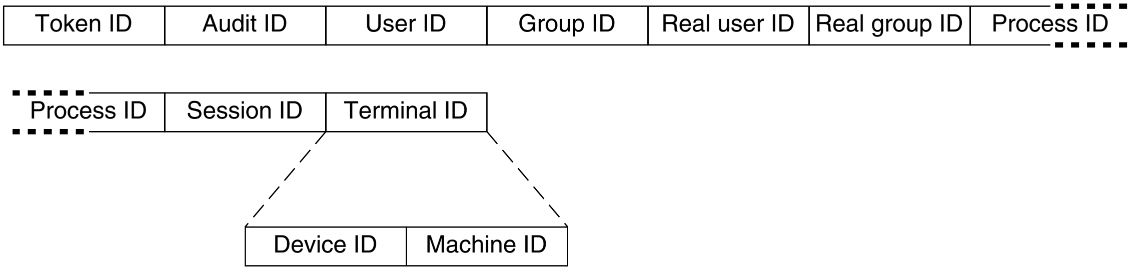


FIGURE 31–6 `subject` Token Format

`text` Token

The `text` token contains a text string. This token has three fields:

- A token ID that identifies this token as a `text` token
- The length of the text string
- The text string itself

The `praudit` command displays the `text` token as follows:

```
text,logout jdoe
```

`trailer` Token

The two tokens, `header` and `trailer`, are special in that they distinguish the end points of an audit record and bracket all the other tokens. A `header` token begins an audit record. A `trailer` token ends an audit record. The `trailer` token is an optional token. The `trailer` token is added as the last token of each record only when the `trail` audit policy option has been set.

When an audit record is generated with trailers turned on, the `auditreduce` command can verify that the trailer correctly points back to the record header. The `trailer` token supports backward seeks of the audit trail.

The `trailer` token has three fields:

- A token ID that identifies this token as a `trailer` token
- A pad number to aid in marking the end of the record
- The total number of characters in the audit record, including both the `header` and `trailer` tokens

The `praudit` command displays the `trailer` token, without the second field, as follows:

```
trailer,136
```

`uauth` Token

The `uauth` token records the use of authorization with a command or action.

The `uauth` token contains the following fields:

- A token ID that identifies this token as a `uauth` token
- The length of the text in the following field
- A list of authorizations

The `praudit` command displays the `uauth` token as follows:

```
use of authorization,solaris.admin.printer.delete
```

`zonename` Token

The `zonename` token records the zone in which the audit event occurred. The string “global” indicates audit events that occur in the global zone.

The `zonename` token contains the following fields:

- A token ID that identifies this token as a `zonename` token
- The length of the text in the following field

- The name of the zone

The `praudit` command displays the `zonename` token as follows:

```
zonename,graphzone
```

Glossary

Access Control List (ACL)	An access control list (ACL) provides finer-grained file security than traditional UNIX file protection provides. For example, an ACL enables you to allow group read access to a file, while allowing only one member of that group to write to the file.
admin principal	A user principal with a name of the form *username*`/admin` (as in `jdoe/admin`). An admin principal can have more privileges (for example, to change policies) than a regular user principal. See also principal name, user principal.
AES	Advanced Encryption Standard. A symmetric 128-bit block data encryption technique. The U.S. government adopted the Rijndael variant of the algorithm as its encryption standard in October 2000. AES replaces user principal encryption as the government standard.
algorithm	A cryptographic algorithm. This is an established, recursive computational procedure that encrypts or hashes input.
application server	See network application server.
audit files	Binary audit logs. Audit files are stored separately in an audit partition.
audit partition	A hard disk partition that is configured to hold audit files.
audit policy	The global and per-user settings that determine which audit events are recorded. The global settings that apply to the audit service typically affect which pieces of optional information are included in the audit trail. Two settings, `cnt` and `ahlt`, affect the operation of the system when the audit queue fills. For example, audit policy might require that a sequence number be part of every audit record.
audit trail	The collection of all audit files from all hosts.
authentication	The process of verifying the claimed identity of a principal.
authenticator	Authenticators are passed by clients when requesting tickets (from a KDC) and services (from a server). They contain information that is generated by using a session key known only by the client and server, that can be verified as of recent origin, thus indicating that the transaction is secure. When used with a ticket, an authenticator can be used to authenticate a user principal. An authenticator includes the principal name of the user, the IP address of the user's host, and a time stamp. Unlike a ticket, an authenticator can be used only once, usually when access to a service is requested. An authenticator is encrypted by using the session key for that client and that server.
authorization	1. In Kerberos, the process of determining if a principal can use a service, which objects the principal is allowed to access, and the type of access that is allowed for each object.

	2. In role-based access control (RBAC), a permission that can be assigned to a role or user (or embedded in a rights profile) for performing a class of actions that are otherwise prohibited by security policy.
Basic Security Module (BSM)	The Solaris auditing service and device allocation. Together, these features satisfy the C2 level of security.
basic set	The set of privileges that are assigned to a user's process at login. On an unmodified system, each user's initial inheritable set equals the basic set at login.
Blowfish	A symmetric block cipher algorithm that takes a variable-length key from 32 bits to 448 bits. Its author, Bruce Schneier, claims that Blowfish is optimized for applications where the key does not change often.
client	Narrowly, a process that makes use of a network service on behalf of a user; for example, an application that uses `rlogin`. In some cases, a server can itself be a client of some other server or service. More broadly, a host that a) receives a Kerberos credential, and b) makes use of a service that is provided by a server. Informally, a principal that makes use of a service.
client principal	(RPCSEC_GSS API) A client (a user or an application) that uses RPCSEC_GSS-secured network services. Client principal names are stored in the form of `rpc_gss_principal_t` structures.
clock skew	The maximum amount of time that the internal system clocks on all hosts that are participating in the Kerberos authentication system can differ. If the clock skew is exceeded between any of the participating hosts, requests are rejected. Clock skew can be specified in the `krb5.conf` file.
confidentiality	See privacy.
consumer	In the Solaris Cryptographic Framework, a consumer is a user of the cryptographic services that come from providers. Consumers can be applications, end users, or kernel operations. Kerberos, IKE, and IPsec are examples of consumers. For examples of providers, see provider.
credential	An information package that includes a ticket and a matching session key. Used to authenticate the identity of a principal. See also ticket, session key.
credential cache	A storage space (usually a file) that contains credentials that are received from the KDC.
cryptographic algorithm	See algorithm.
DES	Data Encryption Standard. A symmetric-key encryption method developed in 1975 and standardized by ANSI in 1981 as ANSI X.3.92. DES uses a 56-bit key.
device allocation	Device protection at the user level. Device allocation enforces the exclusive use of a device by one user at a time. Device data is purged before device reuse. Authorizations can be used to limit who is permitted to allocate a device.
device policy	Device protection at the kernel level. Device policy is implemented as two sets of privileges on a device. One set of privileges controls read access to the device. The second set of privileges controls write access to the device. See also policy.

Diffie-Hellman protocol	Also known as public key cryptography. An asymmetric cryptographic key agreement protocol that was developed by Diffie and Hellman in 1976. The protocol enables two users to exchange a secret key over an insecure medium without any prior secrets. Diffie-Hellman is used by Kerberos.
digest	See message digest.
DSA	Digital Signature Algorithm. A public key algorithm with a variable key size from 512 to 4096 bits. The U.S. Government standard, DSS, goes up to 1024 bits. DSA relies on SHA1 for input.
effective set	The set of privileges that are currently in effect on a process.
flavor	Historically, *security flavor* and *authentication flavor* had the same meaning, as a flavor that indicated a type of authentication (AUTH_UNIX, AUTH_DES, AUTH_KERB). RPCSEC_GSS is also a security flavor, even though it provides integrity and privacy services in addition to authentication.
forwardable ticket	A ticket that a client can use to request a ticket on a remote host without requiring the client to go through the full authentication process on that host. For example, if the user `david` obtains a forwardable ticket while on user `jennifer`'s machine, he can log in to his own machine without being required to get a new ticket (and thus authenticate himself again). See also proxiable ticket.
FQDN	Fully qualified domain name. For example, `central.example.com` (as opposed to simply `denver`).
GSS-API	The Generic Security Service Application Programming Interface. A network layer that provides support for various modular security services, including the Kerberos service. GSS-API provides for security authentication, integrity, and privacy services. See also authentication, integrity, privacy.
hardening	The modification of the default configuration of the operating system to remove security vulnerabilities that are inherent in the host.
hardware provider	In the Solaris Cryptographic Framework, a device driver and its hardware accelerator. Hardware providers offload expensive cryptographic operations from the computer system, thus freeing CPU resources for other uses. See also provider.
host	A machine that is accessible over a network.
host principal	A particular instance of a service principal in which the principal (signified by the primary name `host`) is set up to provide a range of network services, such as `ftp`, `rcp`, or `rlogin`. An example of a host principal is `host/central.example.com@EXAMPLE.COM`. See also server principal.
inheritable set	The set of privileges that a process can inherit across a call to `exec`.
initial ticket	A ticket that is issued directly (that is, not based on an existing ticket-granting ticket). Some services, such as applications that change passwords, might require tickets to be marked *initial* so as to assure themselves that the client can demonstrate a knowledge of its secret key. This assurance is important because an initial ticket indicates that the client has recently authenticated itself (instead of relying on a ticket-granting ticket, which might existed for a long time).
instance	The second part of a principal name, an instance qualifies the principal's primary. In the case of a service principal, the instance is required. The instance the host's fully qualified domain name, as in `host/central.example.com`. For user principals, an instance is optional. Note, however, that `jdoe` and `jdoe/admin` are unique principals. See also primary, principal name, service principal, user principal.

integrity A security service that, in addition to user authentication, provides for the validity of transmitted data through cryptographic checksumming. See also authentication, privacy.

invalid ticket A postdated ticket that has not yet become usable. An invalid ticket is rejected by an application server until it becomes validated. To be validated, an invalid ticket must be presented to the KDC by the client in a TGS request, with the `VALIDATE` flag set, after its start time has passed. See also postdated ticket.

KDC Key Distribution Center. A machine that has three Kerberos V5 components:

- Principal and key database
- Authentication service
- Ticket-granting service

Each realm has a master KDC and should have one or more slave KDCs.

Kerberos An authentication service, the protocol that is used by that service, or the code that is used to implement that service.

The Solaris Kerberos implementation that is closely based on Kerberos V5 implementation.

While technically different, “Kerberos” and “Kerberos V5” are often used interchangeably in the Kerberos documentation.

Kerberos (also spelled Cerberus) was a fierce, three-headed mastiff who guarded the gates of Hades in Greek mythology.

Kerberos policy A set of rules that governs password usage in the Kerberos service. Policies can regulate principals' accesses, or ticket parameters, such as lifetime.

key 1. Generally, one of two main types of keys:

- A *symmetric key* – An encryption key that is identical to the decryption key. Symmetric keys are used to encrypt files.
- An *asymmetric key* or *public key* – A key that is used in public key algorithms, such as Diffie-Hellman or RSA. Public keys include a private key that is known only by one user, a public key that is used by the server or general resource, and a private-public key pair that combines the two. A private key is also called a *secret* key. The public key is also called a *shared* key or *common* key.
- 2. An entry (principal name) in a keytab file. See also keytab file.

3. In Kerberos, an encryption key, of which there are three types:

- A *private key* – An encryption key that is shared by a principal and the KDC, and distributed outside the bounds of the system. See also private key.
- A *service key* – This key serves the same purpose as the private key, but is used by servers and services. See also service key.
- A *session key* – A temporary encryption key that is used between two principals, with a lifetime limited to the duration of a single login session. See also session key.

keytab file A key table file that contains one or more keys (principals). A host or service uses a keytab file in the much the same way that a user uses a password.

kvno	Key version number. A sequence number that tracks a particular key in order of generation. The highest kvno is the latest and most current key.
limit set	The outside limit of what privileges are available to a process and its children.
MAC	1. See message authentication code (MAC). 2. Also called labeling. In government security terminology, MAC is Mandatory Access Control. Labels such as Top Secret and Confidential are examples of MAC. MAC contrasts with DAC, which is Discretionary Access Control. UNIX permissions are an example of DAC. 3. In hardware, the unique machine address on a LAN. If the machine is on an Ethernet, the MAC is the Ethernet address.
master KDC	The main KDC in each realm, which includes a Kerberos administration server, `kadmind`, and an authentication and ticket-granting daemon, `krb5kdc`. Each realm must have at least one master KDC, and can have many duplicate, or slave, KDCs that provide authentication services to clients.
MD5	An iterative cryptographic hash function that is used for message authentication, including digital signatures. The function was developed in 1991 by Rivest.
mechanism	1. A software package that specifies cryptographic techniques to achieve data authentication or confidentiality. Examples: Kerberos V5, Diffie-Hellman public key. 2. In the Solaris Cryptographic Framework, an implementation of an algorithm for a particular purpose. For example, a DES mechanism that is applied to authentication, such as CKM_DES_MAC, is a separate mechanism from a DES mechanism that is applied to encryption, CKM_DES_CBC_PAD.
message authentication code (MAC)	MAC provides assurance of data integrity and authenticates data origin. MAC does not protect against eavesdropping.
message digest	A message digest is a hash value that is computed from a message. The hash value almost uniquely identifies the message. A digest is useful for verifying the integrity of a file.
minimization	The installation of the minimal operating system that is necessary to run the server. Any software that does not directly relate to the operation of the server is either not installed, or deleted after the installation.
name service scope	The scope in which a role is permitted to operate, that is, an individual host or all hosts that are served by a specified name service such as NIS, NIS+, or LDAP. Scopes are applied to Solaris Management Console toolboxes.
network application server	A server that provides a network application, such as `ftp`. A realm can contain several network application servers.
nonattributable audit event	An audit event whose initiator cannot be determined, such as the AUE_BOOT event.
NTP	Network Time Protocol. Software from the University of Delaware that enables you to manage precise time or network clock synchronization, or both, in a network environment. You can use NTP to maintain clock skew in a Kerberos environment. See also clock skew.

PAM	Pluggable Authentication Module. A framework that allows for multiple authentication mechanisms to be used without having to recompile the services that use them. PAM enables Kerberos session initialization at login.
passphrase	A phrase that is used to verify that a private key was created by the passphrase user. A good passphrase is 10-30 characters long, mixes alphabetic and numeric characters, and avoids simple prose and simple names. You are prompted for the passphrase to authenticate use of the private key to encrypt and decrypt communications.
password policy	The encryption algorithms that can be used to generate passwords. Can also refer to more general issues around passwords, such as how often the passwords must be changed, how many mis-entries are permitted, and other security considerations. Security policy requires passwords. Password policy might require passwords to be encrypted with the MD5 algorithm, and might make further requirements related to password strength.
permitted set	The set of privileges that are available for use by a process.
policy	Generally, a plan or course of action that influences or determines decisions and actions. For computer systems, policy typically means security policy. Your site's security policy is the set of rules that define the sensitivity of the information that is being processed and the measures that are used to protect the information from unauthorized access. For example, security policy might require that systems be audited, that devices be protected with privileges, and that passwords be changed every six weeks. For the implementation of policy in specific areas of the Solaris OS, see audit policy, policy in the cryptographic framework, device policy, Kerberos policy, password policy, and RBAC policy.
policy for public key technologies	In the Key Management Framework (KMF), policy is the management of certificate usage. The KMF policy database can put constraints on the use of the keys and certificates that are managed by the KMF library.
policy in the cryptographic framework	In the Solaris Cryptographic Framework, policy is the disabling of existing cryptographic mechanisms. The mechanisms then cannot be used. Policy in the cryptographic framework might prevent the use of a particular mechanism, such as `CKM_DES_CBC`, from a provider, such as DES.
postdated ticket	A postdated ticket does not become valid until some specified time after its creation. Such a ticket is useful, for example, for batch jobs that are intended to run late at night, since the ticket, if stolen, cannot be used until the batch job is run. When a postdated ticket is issued, it is issued as *invalid* and remains that way until a) its start time has passed, and b) the client requests validation by the KDC. A postdated ticket is normally valid until the expiration time of the ticket-granting ticket. However, if the postdated ticket is marked *renewable*, its lifetime is normally set to be equal to the duration of the full life time of the ticket-granting ticket. See also invalid ticket, renewable ticket.
primary	The first part of a principal name. See also instance, principal name, realm.
principal	1. A uniquely named client/user or server/service instance that participates in a network communication. Kerberos transactions involve interactions between principals (service principals and user principals) or between principals and KDCs. In other words, a principal is a unique entity to which Kerberos can assign tickets. See also principal name, service principal, user principal. 2. (RPCSEC_GSS API) See client principal, server principal.

principal name	1. The name of a principal, in the format *primary/instance@REALM*. See also instance, primary, realm. 2. (RPCSEC_GSS API) See client principal, server principal.
privacy	A security service, in which transmitted data is encrypted before being sent. Privacy also includes data integrity and user authentication. See also authentication, integrity, service.
private key	A key that is given to each user principal, and known only to the user of the principal and to the KDC. For user principals, the key is based on the user's password. See also key.
private-key encryption	In private-key encryption, the sender and receiver use the same key for encryption. See also public-key encryption.
privilege	A discrete right on a process in a Solaris system. Privileges offer a finer-grained control of processes than does `root`. Privileges are defined and enforced in the kernel. For a full description of privileges, see the `privileges`(5) man page.
privilege model	A stricter model of security on a computer system than the superuser model. In the privilege model, processes require privilege to run. Administration of the system can be divided into discrete parts that are based on the privileges that administrators have in their processes. Privileges can be assigned to an administrator's login process. Or, privileges can be assigned to be in effect for certain commands only.
privilege set	A collection of privileges. Every process has four sets of privileges that determine whether a process can use a particular privilege. See limit set, effective set set, permitted set set, and inheritable set set. Also, the basic set set of privileges is the collection of privileges that are assigned to a user's process at login.
privileged application	An application that can override system controls. The application checks for security attributes, such as specific UIDs, GIDs, authorizations, or privileges.
profile shell	In RBAC, a shell that enables a role (or user) to run from the command line any privileged applications that are assigned to the role's rights profiles. The profile shells are `pfsh`, `pfcsh`, and `pfksh`. They correspond to the Bourne shell (`sh`), C shell (`csh`), and Korn shell (`ksh`), respectively.
provider	In the Solaris Cryptographic Framework, a cryptographic service that is provided to consumers. PKCS #11 libraries, kernel cryptographic modules, and hardware accelerators are examples of providers. Providers plug in to the Solaris Cryptographic Framework, so are also called *plugins*. For examples of consumers, see consumer.
proxiable ticket	A ticket that can be used by a service on behalf of a client to perform an operation for the client. Thus, the service is said to act as the client's proxy. With the ticket, the service can take on the identity of the client. The service can use a proxiable ticket to obtain a service ticket to another service, but it cannot obtain a ticket-granting ticket. The difference between a proxiable ticket and a forwardable ticket is that a proxiable ticket is only valid for a single operation. See also forwardable ticket.
public-key encryption	An encryption scheme in which each user has two keys, one public key and one private key. In public-key encryption, the sender uses the receiver's public key to encrypt the message, and the receiver uses a private key to decrypt it. The Kerberos service is a private-key system. See also private-key encryption.
QOP	Quality of Protection. A parameter that is used to select the cryptographic algorithms that are used in conjunction with the integrity service or privacy service.

RBAC	Role-Based Access Control. An alternative to the all-or-nothing superuser model. RBAC lets an organization separate superuser's capabilities and assign them to special user accounts called roles. Roles can be assigned to specific individuals according to their responsibilities.
RBAC policy	The security policy that is associated with a command. Currently, `suser` and `solaris` are the valid policies. The `solaris` policy recognizes privileges and `setuid` security attributes. The `suser` policy recognizes only `setuid` security attributes. Trusted Solaris™ systems, which can interoperate with a Solaris system, provide a `tsol` policy, which recognizes privileges, `setuid` security attributes, and labels on processes.
realm	1. The logical network that is served by a single Kerberos database and a set of Key Distribution Centers (KDCs). 2. The third part of a principal name. For the principal name `jdoe/admin@ENG.EXAMPLE.COM`, the realm is `ENG.EXAMPLE.COM`. See also principal name.
relation	A configuration variable or relationship that is defined in the `kdc.conf` or `krb5.conf` files.
renewable ticket	Because having tickets with very long lives is a security risk, tickets can be designated as *renewable*. A renewable ticket has two expiration times: a) the time at which the current instance of the ticket expires, and b) maximum lifetime for any ticket. If a client wants to continue to use a ticket, the client renews the ticket before the first expiration occurs. For example, a ticket can be valid for one hour, with all tickets having a maximum lifetime of ten hours. If the client that holds the ticket wants to keep it for more than an hour, the client must renew the ticket. When a ticket reaches the maximum ticket lifetime, it automatically expires and cannot be renewed.
rights profile	Also referred to as a right or a profile. A collection of overrides used in RBAC that can be assigned to a role or user. A rights profile can consist of authorizations, commands with security attributes, and other rights profiles.
role	A special identity for running privileged applications that only assigned users can assume.
RSA	A method for obtaining digital signatures and public key cryptosystems. The method was first described in 1978 by its developers, Rivest, Shamir, and Adleman.
SEAM	Sun Enterprise Authentication Mechanism. The product name for the initial versions of a system for authenticating users over a network, based on the Kerberos V5 technology that was developed at the Massachusetts Institute of Technology. The product is now called the Kerberos service. SEAM refers to parts the Kerberos service that were not included in various Solaris releases.
secret key	See private key.
Secure Shell	A special protocol for secure remote login and other secure network services over an insecure network.
security attributes	In RBAC, overrides to security policy that enable an administrative command to succeed when the command is run by a user other than superuser. In the superuser model, the `setuid` and `setgid` programs are security attributes. When these attributes are applied to a command, the command succeeds no matter who runs the command. In the privilege model, security attributes are privileges. When a privilege is given to a command, the command succeeds. The privilege model is compatible with the superuser model, in that the privilege model also recognizes the `setuid` and `setgid` programs as security attributes.

security flavor	See flavor.
security mechanism	See mechanism.
security policy	See policy.
security service	See service.
seed	A numeric starter for generating random numbers. When the starter originates from a random source, the seed is called a *random seed*.
server	A principal that provides a resource to network clients. For example, if you `rlogin` to the machine `central.example.com`, then that machine is the server that provides the `rlogin` service. See also service principal.
server principal	(RPCSEC_GSS API) A principal that provides a service. The server principal is stored as an ASCII string in the form *service@host*. See also client principal.
service	1. A resource that is provided to network clients, often by more than one server. For example, if you `rlogin` to the machine `central.example.com`, then that machine is the server that provides the `rlogin` service. 2. A security service (either integrity or privacy) that provides a level of protection beyond authentication. See also integrity and privacy.
service key	An encryption key that is shared by a service principal and the KDC, and is distributed outside the bounds of the system. See also key.
service principal	A principal that provides Kerberos authentication for a service or services. For service principals, the primary name is a name of a service, such as `ftp`, and its instance is the fully qualified host name of the system that provides the service. See also host principal, user principal.
session key	A key that is generated by the authentication service or the ticket-granting service. A session key is generated to provide secure transactions between a client and a service. The lifetime of a session key is limited to a single login session. See also key.
SHA1	Secure Hashing Algorithm. The algorithm operates on any input length less than 2^{64} to produce a message digest. The SHA1 algorithm is input to DSA.
single-system image	A single-system image is used in Solaris auditing to describe a group of audited machines that use the same naming service. These machines send their audit records to a central audit server, where the records can be compared as if the records came from one machine.
slave KDC	A copy of a master KDC, which is capable of performing most functions of the master. Each realm usually has several slave KDCs (and only one master KDC). See also KDC, master KDC.
software provider	In the Solaris Cryptographic Framework, a kernel software module or a PKCS #11 library that provides cryptographic services. See also provider.

stash file	A stash file contains an encrypted copy of the master key for the KDC. This master key is used when a server is rebooted to automatically authenticate the KDC before it starts the `kadmind` and `krb5kdc` processes. Because the stash file includes the master key, the stash file and any backups of it should be kept secure. If the encryption is compromised, then the key could be used to access or modify the KDC database.
superuser model	The typical UNIX model of security on a computer system. In the superuser model, an administrator has all-or-nothing control of the machine. Typically, to administer the machine, a user becomes superuser (`root`) and can do all administrative activities.
TGS	Ticket-Granting Service. That portion of the KDC that is responsible for issuing tickets.
TGT	Ticket-Granting Ticket. A ticket that is issued by the KDC that enables a client to request tickets for other services.
ticket	An information packet that is used to securely pass the identity of a user to a server or service. A ticket is valid for only a single client and a particular service on a specific server. A ticket contains the principal name of the service, the principal name of the user, the IP address of the user's host, a time stamp, and a value that defines the lifetime of the ticket. A ticket is created with a random session key to be used by the client and the service. Once a ticket has been created, it can be reused until the ticket expires. A ticket only serves to authenticate a client when it is presented along with a fresh authenticator. See also authenticator, credential, service, session key.
ticket file	See credential cache.
user principal	A principal that is attributed to a particular user. A user principal's primary name is a user name, and its optional instance is a name that is used to described the intended use of the corresponding credentials (for example, `jdoe` or `jdoe/admin`). Also known as a user instance. See also service principal.
virtual private network (VPN)	A network that provides secure communication by using encryption and tunneling to connect users over a public network.

Index

Numbers and Symbols

$$ (double dollar sign), parent shell process number, 240
`[]` (square brackets), `bsmrecord` output, 623
* (asterisk)
  - checking for in RBAC authorizations, 221
  - `device_allocate` file, 93, 94
  - wildcard character
    - in ASET, 161, 163
    - in RBAC authorizations, 228, 231

@ (at sign), `device_allocate` file, 94
\ (backslash)
  - `device_allocate` file, 93
  - `device_maps` file, 92

^ (caret) in audit class prefixes, 619
. (dot)
  - authorization name separator, 228
  - displaying hidden files, 131
  - path variable entry, 47

= (equal sign), file permissions symbol, 126
- (minus sign)
  - audit class prefix, 618
  - file permissions symbol, 126
  - file type symbol, 122
  - `sulog` file, 70

+ (plus sign)
  - ACL entry, 138
  - audit class prefix, 618
  - file permissions symbol, 126
  - `sulog` file, 70

# (pound sign)
  - `device_allocate` file, 93

# (pound sign) *(Continued)*
  - `device_maps` file, 92

? (question mark), ASET tune files, 163
; (semicolon)
  - `device_allocate` file, 93
  - separator of security attributes, 235

> (redirect output), preventing, 48
>> (append output), preventing, 48
`~/.gkadmin` file, description, 529
`~/.k5login` file, description, 529
`~/.rhosts` file, description, 357
`~/.shosts` file, description, 357
`~/.ssh/authorized_keys` file
  - description, 357
  - override, 358

`~/.ssh/config` file
  - description, 358
  - override, 358

`~/.ssh/environment` file, description, 357
`~/.ssh/id_dsa` file, override, 358
`~/.ssh/id_rsa` file, override, 358
`~/.ssh/identity` file, override, 358
`~/.ssh/known_hosts` file
  - description, 357
  - override, 358

`~/.ssh/rc` file, description, 357
`3des-cbc` encryption algorithm, `ssh_config` file, 351
`3des` encryption algorithm, `ssh_config` file, 351

A

- `-A` option, `auditreduce` command, 597
- `-a` option
  - `bsmrecord` command, 594
  - `digest` command, 270
  - `encrypt` command, 273
  - `getfacl` command, 142
  - Kerberized commands, 523
  - `mac` command, 272
  - `smrole` command, 204-205
- absolute mode
  - changing file permissions, 126, 135-136
  - changing special file permissions, 136-137
  - description, 126
  - setting special permissions, 127
- access
  - control lists
    - *See* ACL
  - getting to server
    - with Kerberos, 538-540
  - granting to your account, 520-522
  - login authentication with Solaris Secure Shell, 340-341
  - obtaining for a specific service, 540
  - restricting for
    - devices, 44-46, 76
    - system hardware, 72-74
  - restricting for KDC servers, 457-458
  - `root` access
    - displaying attempts on console, 71-72
    - monitoring `su` command attempts, 46, 70-71
    - preventing login (RBAC), 207-208
    - restricting, 51-52, 71-72
  - Secure RPC authentication, 299
  - security
    - ACLs, 51, 127-130
    - controlling system usage, 46-50
    - devices, 76
    - file access restriction, 48
    - firewall setup, 55-56
    - login access restrictions, 39
    - login authentication, 340-341
    - login control, 39
    - monitoring system usage, 50

- access, security *(Continued)*
    - network control, 52-56
    - NFS client-server, 301-304
    - `PATH` variable setting, 47
    - peripheral devices, 44
    - physical security, 38
    - remote systems, 327
    - reporting problems, 56
    - `root` login tracking, 46
    - saving failed logins, 61
    - `setuid` programs, 48
    - system hardware, 72-74
  - sharing files, 51
  - system logins, 42
- access control list
  - *See* ACL
- Access Control Lists (ACLs), *See* ACL
- ACL
  - changing entries, 140-141
  - checking entries, 138
  - commands, 129-130
  - copying ACL entries, 140
  - default entries for directories, 129
  - deleting entries, 129-130, 141-142
  - description, 51, 127-130
  - directory entries, 129
  - displaying entries, 129-130, 142-143
  - format of entries, 127-130
  - `kadm5.acl` file, 486, 488, 492
  - modifying entries, 140
  - restrictions on copying entries, 128
  - setting entries, 138-140
  - setting on a file, 138
  - task map, 137-143
  - user procedures, 137-143
  - valid file entries, 128-129
- `acl` audit token, format, 625
- `add_drv` command, description, 90
- adding
  - ACL entries, 138-140
  - administration principals (Kerberos), 394, 402
  - allocatable device, 80-81
  - attributes to a rights profile, 215-218
  - audit classes, 577-578

adding *(Continued)*
  audit directories, 580-583
  audit policy, 585
  auditing of roles, 206-207
  auditing of zones, 560-564
  cryptomgt role, 206
  custom roles (RBAC), 204-205
  customized role, 204-205
  DH authentication to mounted file systems, 304
  dial-up passwords, 63-65
  hardware provider mechanisms and features, 286
  keys for DH authentication, 305-306
  library plugin, 280
  local user, 207
  new rights profile, 215-218
  Operator role, 200
  PAM modules, 316
  password encryption module, 68-69
  plugins to cryptographic framework, 279-280
  privileges directly to user or role, 244-245
  privileges to command, 243-244
  RBAC properties to legacy applications, 219-221
  rights profiles with Solaris Management Console, 217
  roles
    for particular profiles, 199-202
    from command line, 202-205
    to a user, 201
    with limited scope, 201
  security attributes to legacy applications, 219-221
  security-related role, 206
  security-related roles, 201
  security to devices, 77-78, 80-85
  security to system hardware, 72-73
  service principal to keytab file (Kerberos), 507-508
  software provider, 279-280
  System Administrator role, 200
  user-level software provider, 280
`admin_server` section
  `krb5.conf` file, 393, 401
administering
  ACLs, 137-143
  auditing
    audit classes, 551-552, 617
administering, auditing *(Continued)*
    audit events, 551
    audit files, 600-602
    audit records, 552
    audit trail overflow prevention, 603
    `auditreduce` command, 596-598
    cost control, 566
    description, 548
    efficiency, 568
    process preselection mask, 606
    reducing storage-space requirements, 567
    task map, 569
    in zones, 555, 616-617
  auditing in zones, 560-561
  cryptographic framework, 264-265
  cryptographic framework and zones, 266
  cryptographic framework task map, 275-276
  device allocation, 79-80
  device policy, 76
  dial-up logins, 63
  file permissions, 131
  Kerberos
    keytabs, 505-512
    policies, 493-501
    principals, 480-492
  metaslot, 264-265
  NFS client-server file security, 301-304
  password algorithms, 65
  privileges, 239
  properties of a role, 213-215
  RBAC properties, 215-218
  remote logins with Solaris Secure Shell, 336-338
  rights profiles, 215-218
  roles, 199-202
  roles to replace superuser, 197-199
  Secure RPC task map, 304
  Solaris Secure Shell
    clients, 350
    overview, 347-349
    servers, 350
    task map, 331
  without privileges, 187
`administrative (old)` audit class, 618
`administrative` audit class, 618

- AES kernel provider, 277
- `aes128-cbc` encryption algorithm, `ssh_config` file, 351
- `aes128-ctr` encryption algorithm, `ssh_config` file, 351
- agent daemon, Solaris Secure Shell, 340-341
- `ahlt` audit policy
  - description, 564
  - setting, 585
- algorithms
  - definition in cryptographic framework, 263
  - listing in the cryptographic framework, 276-278
  - password
    - configuration, 66-67
  - password encryption, 40
- `all`, in user audit fields, 614
- All (RBAC), rights profile, 227
- `all` audit class
  - caution for using, 619
  - description, 617
- `allhard` string, `audit_warn` script, 615
- `allocate` command
  - allocate error state, 92
  - authorizations for, 91
  - authorizations required, 238
  - description, 91
  - tape drive, 86
  - user authorization, 81
  - using, 85-86
- allocate error state, 92
- allocating devices
  - by users, 85-86
  - forcibly, 82
  - task map, 85
- `AllowGroups` keyword, `sshd_config` file, 350
- `AllowTcpForwarding` keyword
  - changing, 335
  - `sshd_config` file, 351
- `AllowUsers` keyword, `sshd_config` file, 351
- `allsoft` string, `audit_warn` script, 615
- `ALTSHELL` in Solaris Secure Shell, 355
- always-audit classes
  - `audit_user` database, 614
  - process preselection mask, 620
- analysis, `praudit` command, 609
- appending arrow (>>), preventing appending, 48
- `application` audit class, 618
- application server, configuring, 413-414
- `arbitrary` audit token
  - format, 625-626
  - item size field, 625
  - print format field, 625
- `arcfour` encryption algorithm, `ssh_config` file, 351
- ARCFOUR kernel provider, 277
- Archive tape drive device-clean script, 94
- archiving, audit files, 603
- `arg` audit token, format, 626
- `arge` audit policy
  - and `exec_env` token, 628
  - description, 565
- `argv` audit policy
  - and `exec_args` token, 627-628
  - description, 565
- ASET
  - aliases file
    - description, 155
    - examples, 164
    - `UID_ALIASES` variable, 158
  - `aset` command
    - interactive version, 165-166
    - `-p` option, 166
    - starting, 148
  - `aset.restore` command, 158
  - `ASETDIR` variable, 160
  - `asetenv` file, 156
  - `ASETSECLEVEL` variable, 161
  - `CKLISTPATH_level` variable, 162
  - collecting reports, 167-168
  - configuring, 156-158, 158
  - description, 48, 147-164
  - environment file, 156
  - environment variables, 160
  - error messages, 168
  - execution log, 151
  - master files, 149, 155
  - NFS services and, 159
  - `PERIODIC_SCHEDULE` variable, 157, 161
  - restoring original system state, 158

ASET *(Continued)*
  running ASET periodically, 166
  running interactively, 165-166
  running periodically, 166
  scheduling ASET execution, 157, 161
  stopping from running periodically, 167
  task map, 164-168
  `TASKS` variable, 156, 162
  troubleshooting, 168
  tune file examples, 163
  tune files, 155, 158
  `uid_aliases` file, 155
  `UID_ALIASES` variable, 155, 158, 162
  working directory, 160
  `YPCHECK` variable, 158, 162
assigning
  privileges to commands in a rights profile, 243-244
  privileges to commands in a script, 247
  privileges to user or role, 244-245
  role to a user, 200, 201
  role to a user locally, 205-206
assuming role
  how to, 209
  in a terminal window, 209-211
  in Solaris Management Console, 211-212
  Primary Administrator, 210
  `root`, 210
  System Administrator, 211
asterisk (*)
  checking for in RBAC authorizations, 221
  `device_allocate` file, 93, 94
  wildcard character
    in ASET, 161, 163
    in RBAC authorizations, 228, 231
`at` command, authorizations required, 237
at sign (@), `device_allocate` file, 94
`atq` command, authorizations required, 237
`attribute` audit token, 626-627
attributes, keyword in BART, 117
audio devices, security, 95
`audit administration` audit class, 618
audit characteristics
  audit ID, 620
  process preselection mask, 606
audit characteristics *(Continued)*
  processes, 620
  session ID, 620
  terminal ID, 620
  user process preselection mask, 620
`audit_class` file, adding a class, 577-578
audit_class file, description file, 611
audit class preselection, effect on public objects, 550
audit classes
  adding, 577-578
  definitions, 617
  description, 550, 551
  entries in `audit_control` file, 612
  exceptions in `audit_user` database, 613-614
  exceptions to system-wide settings, 551
  mapping events, 552
  modifying default, 577-578
  overview, 551-552
  prefixes, 618
  preselecting, 571-573
  preselection, 550
  process preselection mask, 620
  setting system-wide, 617
  syntax, 618, 619
  system-wide, 612
`audit` command
  description, 606-607
  preselection mask for existing processes (`-s` option), 589
  rereading audit files (`-s` option), 606
  resetting directory pointer (`-n` option), 606
  updating auditing service, 589-590
audit configuration file, *See* `audit_control` file
`audit_control` file
  audit daemon rereading after editing, 589
  changing kernel mask for nonattributable events, 589
  configuring, 571-573
  description, 612
  entries, 612
  entries and zones, 616-617
  examples, 612
  exceptions in `audit_user` database, 613-614

`audit_control` file *(Continued)*
- `flags` line
  - process preselection mask, 620
- `minfree` warning, 615
- overview, 549
- prefixes in `flags` line, 619
- syntax problem, 615

Audit Control rights profile, 616

audit daemon, *See* `auditd` daemon

audit directory
- creating, 582
- description, 550
- partitioning for, 580-583
- sample structure, 608

`audit_event` file
- changing class membership, 578-579
- description, 551

audit events
- `audit_event` file, 551
- changing class membership, 578-579
- description, 551
- mapping to classes, 552
- selecting from audit trail, 598-600
- selecting from audit trail in zones, 616
- summary, 550
- viewing from binary files, 600-602

audit files
- `auditreduce` command, 607
- combining, 596-598, 607
- configuring, 570-579
- copying messages to single file, 600
- managing, 603
- minimum free space for file systems, 612
- names, 621, 622
- order for opening, 612
- partitioning disk for, 580-583
- printing, 601
- reducing, 596-598, 607
- reducing storage-space requirements, 567, 568
- switching to new file, 606
- time stamps, 622

audit ID
- mechanism, 620
- overview, 547-548

audit logs
- *See also* audit files
- comparing binary and textual, 553
- configuring textual audit logs, 573-575
- in text, 612
- modes, 553

audit messages, copying to single file, 600

`audit.notice` entry, `syslog.conf` file, 574

audit policy
- defaults, 564-566
- description, 550
- effects of, 564-566
- `public`, 566
- setting, 584-586
- setting `ahlt`, 585
- setting in global zone, 555, 616-617
- setting `perzone`, 585-586

audit preselection mask, modifying for individual users, 575-577

audit records
- audit directories full, 606, 615
- converting to readable format, 601, 609
- description, 550
- displaying, 600-602
- displaying formats of
  - procedure, 594-596
  - summary, 607
- displaying formats of a program, 595
- displaying formats of an audit class, 596
- displaying in XML format, 602
- events that generate, 548
- format, 622
- formatting example, 595
- merging, 596-598
- overview, 552
- reducing audit files, 596-598
- sequence of tokens, 622
- `syslog.conf` file, 549
- `/var/adm/auditlog` file, 574

Audit Review rights profile, 616

audit session ID, 620

`audit_startup` script
- configuring, 584-586
- description, 613

audit threshold, 612
audit tokens
  See also individual audit token names
  audit record format, 622
  description, 550, 553
  format, 623
  list of, 623
  new in current release, 556
audit trail
  analysis costs, 567
  analysis with `praudit` command, 609
  cleaning up not terminated files, 602-603
  creating
    `auditd` daemon's role, 606
  description, 550
  effect of audit policy on, 564
  events included, 552
  merging all files, 608
  monitoring in real time, 568
  no public objects, 550
  overview, 549
  preventing overflow, 603
  selecting events from, 598-600
  viewing events from, 600-602
  viewing events from different zones, 616
`audit_user` database
  exception to system-wide audit classes, 551
  prefixes for classes, 619
  process preselection mask, 620
  specifying user exceptions, 575-577
  user audit fields, 613-614
`audit_warn` script
  `auditd` daemon execution of, 606
  conditions invoking, 615
  configuring, 583
  description, 614
  strings, 615
`auditconfig` command
  audit classes as arguments, 552, 617
  description, 610
  prefixes for classes, 619
  setting audit policy, 585
`auditd` daemon
  audit trail creation, 606, 620
`auditd` daemon *(Continued)*
  `audit_warn` script
    description, 614, 615
    execution of, 606
  functions, 606
  order audit files are opened, 612
  rereading information for the kernel, 589
  rereading the `audit_control` file, 589
auditing
  changes in current release, 556-557
  changes in device policy, 78
  configuring identically for all zones, 590-592
  configuring in global zone, 560, 585
  configuring per-zone, 592-593
  device allocation, 84-85
  disabling, 587-589
  enabling, 586-587
  planning, 560-564
  planning in zones, 560-561
  preselection definition, 550
  privileges and, 257
  rights profiles for, 616
  roles, 206-207
  updating information, 589-590
  zones and, 555, 616-617
`auditlog` file, text audit records, 574
`auditreduce` command, 607
  `-c` option, 600
  cleaning up audit files, 602-603
  description, 607
  examples, 596-598
  filtering options, 598
  merging audit records, 596-598
  `-O` option, 596-598
  options, 607
  selecting audit records, 598-600
  timestamp use, 622
  trailer tokens, and, 639
  using lowercase options, 598
  using uppercase options, 597
  without options, 608
`auth_attr` database
  description, 232-233
  summary, 229

`AUTH_DES` authentication, *See* `AUTH_DH` authentication
`AUTH_DH` authentication, and NFS, 299
authentication
  `AUTH_DH` client-server session, 301-304
  configuring cross-realm, 410-412
  description, 53-54
  DH authentication, 300-304
  disabling with `-X` option, 524
  Kerberos and, 363
  name services, 299
  network security, 53-54
  NFS-mounted files, 309
  overview of Kerberos, 537-538
  Secure RPC, 299
  Solaris Secure Shell
    methods, 328-330
    process, 348-349
  terminology, 533-534
  types, 53-54
  use with NFS, 299
authentication methods
  GSS-API credentials in Solaris Secure Shell, 328
  host-based in Solaris Secure Shell, 329, 331-334
  keyboard-interactive in Solaris Secure Shell, 329
  password in Solaris Secure Shell, 329
  public keys in Solaris Secure Shell, 329
  Solaris Secure Shell, 328-330
authenticator
  in Kerberos, 533, 539
`authlog` file, saving failed login attempts, 62-63
authorizations
  Kerberos and, 363
  types, 53-54
authorizations (RBAC)
  checking for wildcards, 221
  checking in privileged application, 182
  commands that require authorizations, 237-238
  database, 229-236
  definition, 181
  delegating, 228-229
  description, 178, 228-229
  for allocating device, 81
  for device allocation, 91-92
  granularity, 228
authorizations (RBAC) *(Continued)*
  naming convention, 228
  not requiring for device allocation, 83
  `solaris.device.allocate`, 81, 91
  `solaris.device.revoke`, 91
`authorized_keys` file, description, 357
`AuthorizedKeysFile` keyword, `sshd_config` file, 351
`auths` command, description, 236
`AUTHS_GRANTED` keyword, `policy.conf` file, 235
`auto_transition` option, SASL and, 325
Automated Security Enhancement Tool, *See* ASET
automatic login
  disabling, 524
  enabling, 523
automatically configuring
  Kerberos
    master KDC server, 390
    slave KDC server, 404-405
automatically enabling auditing, 613
automating principal creation, 481-482
`auxprop_login` option, SASL and, 325

B

`-b` option, `auditreduce` command, 599
backup
  Kerberos database, 442-443
  slave KDCs, 382
`Banner` keyword, `sshd_config` file, 351
BART
  components, 98-100
  overview, 97-100
  programmatic output, 119
  security considerations, 101
  task map, 100-101
  verbose output, 119
`bart` command, 97
`bart compare` command, 99
`bart create` command, 98-99, 102
Basic Audit Reporting Tool, *See* BART
basic privilege set, 190
Basic Security Module (BSM)
  See auditing
  See device allocation

Basic Solaris User rights profile, 226
`Batchmode` keyword, `ssh_config` file, 351
`BindAddress` keyword, `ssh_config` file, 351
binding control flag, PAM, 319
`blowfish-cbc` encryption algorithm, `ssh_config` file, 351
Blowfish encryption algorithm
  kernel provider, 277
  `policy.conf` file, 67
  `ssh_config` file, 351
  using for password, 67
Bourne shell, privileged version, 184
`bsmconv` script
  creating `device_maps` file, 92-93
  description, 616
  enabling auditing service, 586-587
`bsmrecord` command
  `[]` (square brackets) in output, 623
  description, 607
  displaying audit record formats, 594-596
  example, 595
  listing all formats, 594
  listing formats of class, 596
  listing formats of program, 595
  optional tokens (`[]`), 623
`bsmunconv` script, disabling auditing service, 587-589

C

`-C` option, `auditreduce` command, 597
C shell, privileged version, 184
`-c` option
  `auditreduce` command, 599, 600
  `bsmrecord` command, 596
`c2audit:audit_load` entry, `system` file, 611
cache, credential, 538
`canon_user_plugin` option, SASL and, 325
caret (^) in audit class prefixes, 619
CD-ROM drives
  allocating, 87-88
  security, 95
`cdrw` command, authorizations required, 237
certificates
  exporting for use by another system, 294-295
certificates *(Continued)*
  generating with `pktool gencert` command, 291-292
  importing into keystore, 293-294
`ChallengeResponseAuthentication` keyword, *See* `KbdInteractiveAuthentication` keyword
`changepw` principal, 506
changing
  ACL entries, 140-141
  allocatable devices, 83-84
  `audit_class` file, 577-578
  `audit_control` file, 571-573
  `audit_event` file, 578-579
  default password algorithm, 65
  device policy, 77-78
  file ownership, 133
  file permissions
    absolute mode, 135-136
    special, 136-137
    symbolic mode, 134-135
  group ownership of file, 134
  NFS secret keys, 301
  passphrase for Solaris Secure Shell, 339
  password algorithm for a domain, 67
  password algorithm task map, 65
  properties of role, 213-215
  rights profile contents, 215-218
  rights profile from command line, 216
  `root` user into role, 207-208
  special file permissions, 136-137
  user properties from command line, 219
  your password with `kpasswd`, 518
  your password with `passwd`, 518
`CheckHostIP` keyword, `ssh_config` file, 351
`chgrp` command
  description, 122
  syntax, 134
`chkey` command, 301, 308
`chmod` command
  changing special permissions, 136-137, 137
  description, 122
  syntax, 136
choosing, your password, 517-518
`chown` command, description, 122

`Cipher` keyword, `sshd_config` file, 351
`Ciphers` keyword, Solaris Secure Shell, 351
`cklist.rpt` file, 150, 154
`CKLISTPATH_level` variable (ASET), 162
classes, *See* audit classes
cleaning up, binary audit files, 602-603
clear protection level, 524
`ClearAllForwardings` keyword, Solaris Secure Shell port forwarding, 351
client names, planning for in Kerberos, 381-382
`ClientAliveCountMax` keyword, Solaris Secure Shell port forwarding, 351
`ClientAliveInterval` keyword, Solaris Secure Shell port forwarding, 351
clients
    `AUTH_DH` client-server session, 301-304
    configuring for Solaris Secure Shell, 348, 350
    configuring Kerberos, 421-434
    definition in Kerberos, 533
`clntconfig` principal
    creating, 396, 404
clock skew
    Kerberos and, 434-436
    Kerberos planning and, 384
clock synchronizing
    Kerberos master KDC and, 397, 404
    Kerberos planning and, 384
    Kerberos slave KDC and, 410
    Kerberos slave server and, 452
`cmd` audit token, 556, 627
`cnt` audit policy, description, 565
combining audit files
    `auditreduce` command, 596-598, 607
    from different zones, 616
command execution, Solaris Secure Shell, 349
command-line equivalents of SEAM Administration Tool, 476-477
commands
    See also individual commands
    ACL commands, 129-130
    auditing commands, 605-610
    cryptographic framework commands, 264-265
    determining user's privileged commands, 249-251
    device allocation commands, 91
commands *(Continued)*
    device policy commands, 89-90
    file protection commands, 121
    for administering privileges, 255
    Kerberos, 531-532
    RBAC administration commands, 236-237
    Secure RPC commands, 301
    Solaris Secure Shell commands, 358-360
    that assign privileges, 191
    that check for privileges, 182
    user-level cryptographic commands, 265
common keys
    calculating, 303
    DH authentication and, 300-304
components
    BART, 98-100
    device allocation mechanism, 90-91
    RBAC, 178-181
    Solaris Secure Shell user session, 349
`Compression` keyword, Solaris Secure Shell, 351
`CompressionLevel` keyword, `ssh_config` file, 351
Computer Emergency Response Team/Coordination Center (CERT/CC), 56
computer security, *See* system security
computing
    DH key, 307
    digest of a file, 270-271
    MAC of a file, 271-273
    secret key, 268-270
configuration decisions
    auditing
        file storage, 561-562
        policy, 564-566
        who and what to audit, 562-564
        zones, 560-561
    Kerberos
        client and service principal names, 381-382
        clients, 384-385
        clock synchronization, 384
        database propagation, 383-384
        encryption types, 385-386
        KDC server, 385
        mapping host names onto realms, 381
        number of realms, 380

configuration decisions, Kerberos *(Continued)*
    - ports, 382
    - realm hierarchy, 381
    - realm names, 380
    - realms, 380-381
    - slave KDCs, 382
  - password algorithm, 40

configuration files
  - ASET, 148
  - `audit_class` file, 611
  - `audit_control` file, 571-573, 606, 612
  - `audit_event` file, 613
  - `audit_startup` script, 613
  - `audit_user` database, 613-614
  - `device_maps` file, 92
  - `nsswitch.conf` file, 39
  - for password algorithms, 40
  - `policy.conf` file, 40, 66-67, 236
  - Solaris Secure Shell, 348
  - `syslog.conf` file, 62-63, 257, 611
  - `system` file, 611
  - with privilege information, 256-257

configuring
  - `ahlt` audit policy, 585
  - ASET, 156-158, 158
  - `audit_class` file, 577-578
  - `audit_control` file, 571-573
  - `audit_event` file, 578-579
  - audit files, 570-579
  - audit files task map, 570
  - audit policy, 584-586
  - audit policy temporarily, 585
  - `audit_startup` script, 584-586
  - audit trail overflow prevention, 603
  - `audit_user` database, 575-577
  - `audit_warn` script, 583
  - `auditconfig` command, 610
  - auditing in zones, 555, 616-617
  - auditing service task map, 579-580
  - custom roles, 204-205
  - device allocation, 79-80
  - device policy, 76
  - devices task map, 75
  - DH key for NIS+ user, 306-307

configuring *(Continued)*
  - DH key for NIS user, 308-309
  - DH key in NIS, 307-308
  - DH key in NIS+, 305-306
  - dial-up logins, 63
  - hardware security, 72-74
  - host-based authentication for Solaris Secure Shell, 331-334
  - identical auditing for non-global zones, 590-592
  - Kerberos
    - adding administration principals, 394, 402
    - clients, 421-434
    - cross-realm authentication, 410-412
    - master KDC server, 390, 392-397
    - master KDC server using LDAP, 397-404
    - NFS servers, 415-417
    - overview, 387-458
    - slave KDC server, 404-405, 405-406, 406-410
    - task map, 387-388
  - name service, 208
  - password for hardware access, 72-73
  - per-zone auditing, 592-593
  - `perzone` audit policy, 585-586
  - port forwarding in Solaris Secure Shell, 334-335
  - RBAC, 197-208
  - RBAC task map, 196-197
  - rights profile from command line, 216
  - rights profiles, 215-218
  - roles, 199-202, 213-215
    - from command line, 202-205
  - `root` user as role, 207-208
  - Solaris Secure Shell, 331
    - clients, 350
    - servers, 350
  - Solaris Secure Shell task map, 331
  - `ssh-agent` daemon, 342
  - textual audit logs, 573-575

configuring application servers, 413-414

`ConnectionAttempts` keyword, `ssh_config` file, 351

console, displaying `su` command attempts, 71-72

`CONSOLE` in Solaris Secure Shell, 354

consumers, definition in cryptographic framework, 263

context-sensitive help, SEAM Administration Tool, 477
control manifests (BART), 97
controlling
  access to system hardware, 72
  system access, 57-58
  system usage, 46-50
conversation keys
  decrypting in secure RPC, 303
  generating in secure RPC, 302
converting
  audit records to readable format, 601, 609
copying
  ACL entries, 140
  files using Solaris Secure Shell, 344
copying audit messages to single file, 600
cost control, and auditing, 566
`crammd5.so.1` plug-in, SASL and, 324
creating
  audit trail
    `auditd` daemon, 620
    `auditd` daemon's role, 606
  credential table, 417
  customized role, 204-205
  `d_passwd` file, 64
  dial-up passwords, 63-65
  `/etc/d_passwd` file, 64
  file digests, 270-271
  keytab file, 395, 402
  local user, 207
  new device-clean scripts, 95-96
  new policy (Kerberos), 486, 497-498
  new principal (Kerberos), 486-488
  Operator role, 200
  partitions for binary audit files, 580-583
  passwords for temporary user, 64
  rights profiles, 215-218
  rights profiles with Solaris Management Console, 217
  roles
    for particular profiles, 199-202
    on command line, 202-205
    with limited scope, 201
  `root` user as role, 207-208

creating *(Continued)*
  secret keys
    for encryption, 268-270
  security-related roles, 201
  Solaris Secure Shell keys, 336-338
  stash file, 410, 452
  System Administrator role, 200
  tickets with `kinit`, 514
`cred` database
  adding client credential, 306
  adding user credential, 306
  DH authentication, 300-304
`cred` table
  DH authentication and, 301
  information stored by server, 303
credential
  cache, 538
  description, 302, 533
  obtaining for a server, 539
  obtaining for a TGS, 538
  or tickets, 365
credential table, adding single entry to, 417-418
credentials, mapping, 383
`crontab` files
  authorizations required, 237
  running ASET periodically, 148
  stop running ASET periodically, 167
cross-realm authentication, configuring, 410-412
`CRYPT_ALGORITHMS_ALLOW` keyword, `policy.conf` file, 41
`CRYPT_ALGORITHMS_DEPRECATE` keyword, `policy.conf` file, 41
`crypt_bsdbf` password algorithm, 41
`crypt_bsdmd5` password algorithm, 41
`crypt` command, file security, 51
`crypt.conf` file
  changing with new password module, 68-69
  third-party password modules, 68-69
`CRYPT_DEFAULT` keyword, `policy.conf` file, 41
`CRYPT_DEFAULT` system variable, 66
`crypt_sunmd5` password algorithm, 41
`crypt_unix` password algorithm, 41, 66-69
Crypto Management (RBAC)
  creating role, 206

Crypto Management (RBAC) *(Continued)*
  use of rights profile, 280, 282
`cryptoadm` command
  description, 264
  disabling cryptographic mechanisms, 280, 282
  disabling hardware mechanisms, 285-287
  installing PKCS #11 library, 280
  listing providers, 277
  `-m` option, 281, 282
  `-p` option, 281, 282
  restoring kernel software provider, 283
`cryptoadm install` command, installing PKCS #11 library, 280
cryptographic framework
  administering with role, 206
  connecting providers, 266
  consumers, 262
  `cryptoadm` command, 264
  definition of terms, 262
  description, 262
  `elfsign` command, 264, 265
  error messages, 275
  installing providers, 266
  interacting with, 264
  listing providers, 276-278
  PKCS #11 library, 262
  providers, 262
  refreshing, 287-288
  registering providers, 266
  restarting, 287-288
  signing providers, 266
  task maps, 267
  user-level commands, 265
  zones and, 266, 287-288
cryptographic services, *See* cryptographic framework
Cryptoki, *See* PKCS #11 library
`csh` command, privileged version, 184
`.cshrc` file, path variable entry, 47
Custom Operator (RBAC), creating role, 204-205
customizing, manifests, 104-107
customizing a report (BART), 113-114

D
`-D` option
  `auditreduce` command, 598
  `ppriv` command, 242
`d_passwd` file
  creating, 64
  description, 43
  disabling dial-up logins temporarily, 65
`-d` option
  `auditreduce` command, 600
  `getfacl` command, 142
  `praudit` command, 609
  `setfacl` command, 142
daemons
  `auditd`, 606
  `kcfd`, 265
  `keyserv`, 305
  `nscd` (name service cache daemon), 200, 236
  `rpc.nispasswd`, 67
  running with privileges, 187
  `ssh-agent`, 340-341
  `sshd`, 347-349
  table of Kerberos, 532
  `vold`, 81
Data Encryption Standard, *See* DES encryption
data forwarding, Solaris Secure Shell, 349
databases
  `audit_user`, 613-614
  `auth_attr`, 232-233
  backing up and propagating KDC, 442-443
  creating KDC, 394
  `cred` for Secure RPC, 301, 306
  `exec_attr`, 234-235
  KDC propagation, 383-384
  NFS secret keys, 301
  `prof_attr`, 233-234
  `publickey` for Secure RPC, 301
  RBAC, 229-236
  `user_attr`, 231
  with privilege information, 256-257
`dd` command, generating secret keys, 268-270
`deallocate` command
  allocate error state, 92
  authorizations for, 91

`deallocate` command *(Continued)*
  authorizations required, 238
  description, 91
  device-clean scripts and, 95-96
  using, 88-89
deallocating
  devices, 88-89
  forcibly, 83
  microphone, 88-89
debugging, privileges, 242
debugging sequence number, 636
`decrypt` command
  description, 265
  syntax, 274
decrypting
  conversation keys for Secure RPC, 303
  files, 274
  NFS secret keys, 301
  secret keys, 301
`default/login` file, description, 357
`default_realm` section
  `krb5.conf` file, 393, 401
`defaultpriv` keyword, `user_attr` database, 256
defaults
  ACL entries for directories, 129
  `audit_startup` script, 613
  `praudit` output format, 609
  privilege settings in `policy.conf` file, 256
  system-wide auditing, 617
  system-wide in `policy.conf` file, 40
  `umask` value, 125
delegating, RBAC authorizations, 228-229
`delete_entry` command, `ktutil` command, 511
deleting
  ACL entries, 129-130, 141-142
  archived audit files, 603
  audit files, 596
  host's service, 511
  `not_terminated` audit files, 602-603
  policies (Kerberos), 500-501
  principal (Kerberos), 489-490
  rights profiles, 216
`DenyGroups` keyword, `sshd_config` file, 351
`DenyUsers` keyword, `sshd_config` file, 351
DES encryption
  kernel provider, 277
  Secure NFS, 300
destroying, tickets with `kdestroy`, 516
determining
  files with `setuid` permissions, 143
  if file has ACL, 138
  privileges on a process, 240-241
  privileges task map, 247
`/dev/arp` device, getting IP MIB-II information, 79
`/dev/urandom` device, 268-270
`devfsadm` command, description, 90
`device_allocate` file
  description, 93-94
  format, 93
  sample, 83, 93
device allocation
  adding devices, 79-80
  allocatable devices, 94
  `allocate` command, 91
  allocate error state, 92
  allocating devices, 85-86
  auditing, 84-85
  authorizations for commands, 91-92
  authorizing users to allocate, 81
  changing allocatable devices, 83-84
  commands, 91
  components of mechanism, 90-91
  configuration file, 92
  `deallocate` command, 91
    device-clean scripts and, 95-96
    using, 88-89
  deallocating devices, 88-89
  `device_allocate` file, 93-94
  device-clean scripts
    audio devices, 95
    CD-ROM drives, 95
    description, 94-96
    diskette drives, 95
    options, 95
    tape drives, 94
    writing new scripts, 95-96
  `device_maps` file, 92-93
  disabling, 588

device allocation *(Continued)*
  enabling, 80-81
  examples, 86
  forcibly allocating devices, 82
  forcibly deallocating devices, 83
  making device allocatable, 80-81
  managing devices, 79-80
  mounting devices, 86-88
  not requiring authorization, 83
  preventing, 84
  requiring authorization, 83-84
  task map, 79-80
  unmounting allocated device, 89
  user procedures, 85
  using, 85
  using `allocate` command, 85-86
  viewing information, 82
device-clean scripts
  and object reuse, 94-96
  audio devices, 95
  CD-ROM drives, 95
  description, 94-96
  diskette drives, 95
  options, 95
  tape drives, 94
  writing new scripts, 95-96
device management, *See* device policy
`device_maps` file
  description, 92
  format, 92
  sample entries, 92
device policy
  `add_drv` command, 89
  auditing changes, 78
  changing, 77-78
  commands, 89
  configuring, 76-79
  kernel protection, 89-96
  managing devices, 76
  overview, 44-46
  removing from device, 78
  task map, 76
  `update_drv` command, 77-78, 89
  viewing, 76-77
Device Security (RBAC), creating role, 201
devices
  adding device policy, 77-78
  allocating for use, 85
  auditing allocation of, 84-85
  auditing policy changes, 78
  authorizing users to allocate, 81
  changing device policy, 77-78
  changing which are allocatable, 83-84
  deallocating a device, 88-89
  `/dev/urandom` device, 268-270
  device allocation
    See device allocation
  forcibly allocating, 82
  forcibly deallocating, 83
  getting IP MIB-II information, 79
  listing, 76-77
  listing device names, 82
  login access control, 43
  making allocatable, 80-81
  managing, 76
  managing allocation of, 79-80
  mounting allocated devices, 86-88
  not requiring authorization for use, 83
  policy commands, 89-90
  preventing use of all, 84
  preventing use of some, 84
  privilege model and, 192
  protecting by device allocation, 44
  protecting in the kernel, 44
  removing policy, 78
  security, 44-46
  superuser model and, 192
  unmounting allocated device, 89
  viewing allocation information, 82
  viewing device policy, 76-77
  zones and, 44
`dfstab` file
  security modes, 420
  sharing files, 51
DH authentication
  configuring in NIS, 307-308
  configuring in NIS+, 305-306
  description, 300-304

DH authentication *(Continued)*
  for NIS+ client, 306
  for NIS client, 307-308
  mounting files with, 309
  sharing files with, 309
DHCP Management (RBAC), creating role, 201
dial-up passwords
  creating, 63-65
  disabling, 44
  disabling temporarily, 65
  `/etc/d_passwd` file, 44
  security, 43-44
`dialups` file, creating, 63
Diffie-Hellman authentication, *See* DH authentication
`digest` command
  description, 265
  example, 271
  syntax, 270
`digestmd5.so.1` plug-in, SASL and, 324
digests
  computing for file, 270-271
  of files, 270-271, 271
`dir` line, `audit_control` file, 612
direct realms, 411-412
directories
  See also files
  ACL entries, 129
  `audit_control` file definitions, 612
  audit directories full, 606, 615
  `auditd` daemon pointer, 606
  checklist task setting (ASET), 157, 162
  displaying files and related information, 121, 131-132
  master files (ASET), 155
  mounting audit directories, 621
  permissions
    defaults, 125
    description, 122-123
  public directories, 125
  reports (ASET), 153
  working directory (ASET), 160, 165-166
disabling
  abort sequence, 73-74
  audit policy, 584-586

disabling *(Continued)*
  auditing service, 587-589
  cryptographic mechanisms, 280
  device allocation, 588
  dial-up logins temporarily, 65
  dial-up passwords, 65
  executable stacks, 144-145
  executables that compromise security, 130
  hardware mechanisms, 285-287
  keyboard abort, 73-74
  keyboard shutdown, 73-74
  logging of executable stack messages, 145
  logins temporarily, 60-61
  programs from using executable stacks, 144-145
  remote `root` access, 71-72
  service on a host (Kerberos), 510-512
  system abort sequence, 73-74
  user logins, 60-61
disk partitioning, for binary audit files, 580-583
disk-space requirements, 567
diskette drives
  allocating, 87
  device-clean scripts, 95
displaying
  ACL entries, 129-130, 138, 142-143
  allocatable devices, 82
  ASET task status, 149, 152
  audit policies, 584
  audit record formats, 594-596
  audit records, 600-602
  audit records in XML format, 602
  device policy, 76-77
  file information, 131-132
  files and related information, 121
  format of audit records, 594-596
  providers in the cryptographic framework, 276-278
  roles you can assume, 209, 236
  `root` access attempts, 71-72
  selected audit records, 596-598
  `su` command attempts, 71-72
  sublist of principals (Kerberos), 483
  user's login status, 58-59, 59
  users with no passwords, 59-60
`dminfo` command, 92

DNS, Kerberos and, 381-382
`domain_realm` section
  `krb5.conf` file, 381, 393, 401
dot (`.`)
  authorization name separator, 228
  displaying hidden files, 131
  path variable entry, 47
double dollar sign ($$), parent shell process number, 240
`DSAAuthentication` keyword, *See* `PubkeyAuthentication` keyword
DTD for `praudit` command, 609
`.dtprofile` script, use in Solaris Secure Shell, 342
duplicating, principals (Kerberos), 488
`DynamicForward` keyword, `ssh_config` file, 351

E

`-e` option
  `auditreduce` command, 600
  `ppriv` command, 242
`ebusy` string, `audit_warn` script, 615
`eeprom` command, 38, 72-74
`eeprom.rpt` file, 151, 154
effective privilege set, 189
efficiency, auditing and, 568
`eject` command, device cleanup and, 95
`elfsign` command
  description, 264, 265
enabling
  auditing, 586-587
  auditing service, 586-587
  auditing service task map, 579-580
  cryptographic mechanisms, 282
  device allocation, 80-81
  Kerberized applications only, 457
  kernel software provider use, 283
  keyboard abort, 73-74
  mechanisms and features on hardware provider, 286
`encrypt` command
  description, 265
  error messages, 275
  syntax, 268

`encrypt` command *(Continued)*
  troubleshooting, 275
encrypting
  communications between hosts, 340
  `encrypt` command, 273-275
  files, 51, 268, 273-275
  network traffic between hosts, 327-330
  passwords, 65
  private key of NIS user, 308
  Secure NFS, 300
  using user-level commands, 265
encryption
  algorithms
    Kerberos and, 385-386
  DES algorithm, 300
  generating symmetric key
    using the `dd` command, 268-270
  installing third-party password modules, 68-69
  list of password algorithms, 40
  modes
    Kerberos and, 385-386
  password algorithm, 40
  privacy service, 363
  specifying algorithms in `ssh_config` file, 351
  specifying password algorithm
    locally, 65
  specifying password algorithms in `policy.conf` file, 40
  types
    Kerberos and, 385-386, 541-543
  with `-x` option, 524
ending, signal received during auditing shutdown, 615
`env.rpt` file, 151, 154
environment variables
  See also variables
  `ASETDIR` (ASET), 160
  `ASETSECLEVEL` (ASET), 161
  audit token for, 628
  `CKLISTPATH_level` (ASET), 157, 162
  overriding proxy servers and ports, 345
  `PATH`, 47
  `PERIODIC_SCHEDULE` (ASET), 157, 161
  presence in audit records, 565, 623
  Solaris Secure Shell and, 354-355

environment variables *(Continued)*
  summary (ASET), 160
  `TASKS` (ASET), 156, 162
  `UID_ALIASES` (ASET), 155, 158, 162
  use with `ssh-agent` command, 359
  `YPCHECK` (ASET), 158, 162
equal sign (=), file permissions symbol, 126
error messages
  `encrypt` command, 275
  Kerberos, 459-471
  with `kpasswd`, 518
errors
  allocate error state, 92
  audit directories full, 606, 615
  internal errors, 615
`EscapeChar` keyword, `ssh_config` file, 351
`/etc/d_passwd` file
  and `/etc/passwd` file, 43
  creating, 64
  disabling dial-up logins temporarily, 65
`/etc/default/kbd` file, 73-74
`/etc/default/login` file
  description, 357
  login default settings, 62
  restricting remote `root` access, 71-72
  Solaris Secure Shell and, 354-355
`/etc/default/su` file
  displaying `su` command attempts, 71-72
  monitoring access attempts, 71-72
  monitoring `su` command, 70-71
`/etc/dfs/dfstab` file
  security modes, 420
  sharing files, 51
`/etc/dialups` file, creating, 63
`/etc/group` file, ASET checks, 150
`/etc/hosts.equiv` file, description, 357
`/etc/krb5/kadm5.acl` file, description, 529
`/etc/krb5/kadm5.keytab` file, description, 530
`/etc/krb5/kdc.conf` file, description, 530
`/etc/krb5/kpropd.acl` file, description, 530
`/etc/krb5/krb5.conf` file, description, 530
`/etc/krb5/krb5.keytab` file, description, 530
`/etc/krb5/warn.conf` file, description, 530
`/etc/logindevperm` file, 43
`/etc/nologin` file
  description, 357
  disabling user logins temporarily, 60-61
`/etc/nsswitch.conf` file, 39
`/etc/pam.conf` file, Kerberos and, 530
`/etc/passwd` file, ASET checks, 150
`/etc/publickey` file, DH authentication and, 301
`/etc/security/audit_event` file, audit events and, 551
`/etc/security/audit_startup` file, 613
`/etc/security/audit_warn` script, 614
`/etc/security/bsmconv` script, 92-93
  description, 616
`/etc/security/crypt.conf` file
  changing with new password module, 68-69
  third-party password modules, 68-69
`/etc/security/device_allocate` file, 93
`/etc/security/device_maps` file, 92
`/etc/security/policy.conf` file, algorithms configuration, 66-67
`/etc/ssh_host_dsa_key.pub` file, description, 356
`/etc/ssh_host_key.pub` file, description, 356
`/etc/ssh_host_rsa_key.pub` file, description, 356
`/etc/ssh/shosts.equiv` file, description, 357
`/etc/ssh/ssh_config` file
  configuring Solaris Secure Shell, 350
  description, 358
  host-specific parameters, 354
  keywords, 350-355
  override, 358
`/etc/ssh/ssh_host_dsa_key` file, description, 356
`/etc/ssh/ssh_host_key` file
  description, 356
  override, 358
`/etc/ssh/ssh_host_rsa_key` file, description, 356
`/etc/ssh/ssh_known_hosts` file
  controlling distribution, 355
  description, 357
  override, 358
  secure distribution, 355
`/etc/ssh/sshd_config` file
  description, 356
  keywords, 350-355
`/etc/ssh/sshrc` file, description, 358

`/etc/syslog.conf` file
  auditing and, 574, 611
  executable stack messages and, 130
  failed logins and, 62-63
  PAM and, 317
`/etc/system` file, 611
event, description, 551
event modifier field flags (`header` token), 630
`exec_args` audit token
  `argv` policy and, 627-628
  format, 627-628
`exec_attr` database
  description, 234-235
  summary, 229
`exec` audit class, 618
`exec_env` audit token, format, 628
executable stacks
  disabling logging messages, 145
  logging messages, 130
  protecting against, 130, 144-145
execute permissions, symbolic mode, 126
execution log (ASET), 151
`exit` audit token, format, 628
`export` subcommand, `pktool` command, 294-295
EXTERNAL security mechanism plug-in, SASL and, 324

F

`-f` option
  Kerberized commands, 523, 525
  `setfacl` command, 140
  `st_clean` script, 96
`-F` option
  `deallocate` command, 92
  Kerberized commands, 523, 525
failed login attempts
  `loginlog` file, 61
  `syslog.conf` file, 62-63
failure
  audit class prefix, 618
  turning off audit classes for, 619
`FallBackToRsh` keyword, `ssh_config` file, 351
`fd_clean` script, description, 95
`file_attr_acc` audit class, 617
`file_attr_mod` audit class, 617
`file` audit token, format, 628-629
`file_close` audit class, 617
`file_creation` audit class, 617
`file_deletion` audit class, 617
file permission modes
  absolute mode, 126
  symbolic mode, 126
`FILE` privileges, 186
`file_read audit` class, 617
file systems
  NFS, 299
  security
    authentication and NFS, 299
    TMPFS file system, 125
  sharing files, 51
  TMPFS, 125
file `vnode` audit token, 626-627
`file_write` audit class, 617
files
  ACL entries
    adding or modifying, 140-141
    checking, 138
    deleting, 129-130, 141-142
    displaying, 129-130, 142-143
    setting, 138-140
    valid entries, 128-129
  ASET checks, 149, 150
  BART manifests, 115-116
  changing ACL, 140-141
  changing group ownership, 134
  changing ownership, 122, 133
  changing special file permissions, 136-137
  computing a digest, 270-271
  computing digests of, 270-271, 271
  computing MAC of, 271-273
  copying ACL entries, 140
  copying with Solaris Secure Shell, 344
  decrypting, 274
  deleting ACL, 141-142
  determining if has ACL, 138
  digest of, 270-271
  displaying ACL entries, 142-143

files *(Continued)*
  displaying file information, 131-132
  displaying hidden files, 131
  displaying information about, 121
  encrypting, 268, 273-275
  file types, 122
  finding files with `setuid` permissions, 143
  for administering Solaris Secure Shell, 356
  hashing, 268
  `kdc.conf`, 535
  Kerberos, 529-530
  manifests (BART), 115-116
  mounting with DH authentication, 309
  ownership
    and `setgid` permission, 124
    and `setuid` permission, 124
  permissions
    absolute mode, 126, 135-136
    changing, 122, 125-127, 135
    defaults, 125
    description, 122-123
    `setgid`, 124
    `setuid`, 124
    sticky bit, 124-125
    symbolic mode, 126, 134-135, 135
    `umask` value, 125
  PKCS #12, 295
  privileges relating to, 186
  protecting with ACLs, 137-143
  protecting with UNIX permissions, 131-137
  public objects, 550
  security
    access restriction, 48
    ACL, 51
    changing ownership, 133
    changing permissions, 125-127, 135
    directory permissions, 122-123
    displaying file information, 121, 132
    encryption, 51, 268
    file permissions, 122-123
    file types, 122
    special file permissions, 127
    `umask` default, 125
    UNIX permissions, 121-127

files, security *(Continued)*
    user classes, 122
  setting ACL, 138-140
  sharing with DH authentication, 309
  special files, 123-125
  symbols of file type, 122
  `syslog.conf` file, 611
  verifying integrity with `digest`, 270-271
  with privilege information, 256-257
`find` command, finding files with `setuid` permissions, 143
`firewall.rpt` file, 151, 154
firewall systems
  ASET setup, 151
  connecting from outside, 346
  outside connections with Solaris Secure Shell
    from command line, 346
    from configuration file, 345-346
  packet smashing, 56
  packet transfers, 56
  secure host connections, 345
  security, 55-56
  trusted hosts, 55
`flags` line
  `audit_control` file, 612
  `plugin` line and, 573
  process preselection mask, 620
forced cleanup, `st_clean` script, 96
format of audit records, `bsmrecord` command, 595
forwardable tickets
  definition, 534
  description, 364
  example, 514
  with `-F` option, 523, 525
  with `-f` option, 523, 525
`ForwardAgent` keyword, Solaris Secure Shell forwarded authentication, 351
`ForwardX11` keyword, Solaris Secure Shell port forwarding, 351
FQDN (Fully Qualified Domain Name), in Kerberos, 381-382
`ftp` command
  Kerberos and, 522-525, 531
  setting protection level in, 524

`ftpd` daemon, Kerberos and, 532

G

`GatewayPorts` keyword, Solaris Secure Shell, 351
gateways, *See* firewall systems
`gencert` subcommand, `pktool` command, 291-292
generating
  certificates with `pktool` command, 291-292
  keys for Solaris Secure Shell, 336-338
  NFS secret keys, 301
  passphrases with `pktool` command, 295-296
  random number
    using the `dd` command, 268-270
  Solaris Secure Shell keys, 336-338
  symmetric key
    using the `dd` command, 268-270
Generic Security Service API, *See* GSS-API
`getdevpolicy` command, description, 90
`getfacl` command
  `-a` option, 142
  `-d` option, 142
  description, 129-130
  displaying ACL entries, 142-143
  examples, 142-143
  verifying ACL entries, 139
getting
  access to a specific service, 540
  credential for a server, 539
  credential for a TGS, 538
`gkadmin` command
  See also SEAM Administration Tool
  description, 531
`.gkadmin` file
  description, 529
  SEAM Administration Tool and, 477
`GlobalKnownHostsFile` keyword, `ssh_config` file, 351
`GlobalKnownHostsFile2` keyword, *See* `GlobalKnownHostsFile` keyword
granting access to your account, 520-522
group ACL entries
  default entries for directories, 129
  description, 128-129
  setting, 138-140
`group` audit policy
  and `groups` token, 565, 629
  description, 565
`group` audit token, replaced by `groups` token, 629
group ID numbers (GIDs), special logins and, 42
groups, changing file ownership, 134
`groups` audit token, 629
GSS-API
  authentication in Solaris Secure Shell, 328
  credentials in secure RPC, 305-306
  credentials in Solaris Secure Shell, 348
  Kerberos and, 364, 378
`gssapi.so.1` plug-in, SASL and, 324
`GSSAPIAuthentication` keyword, Solaris Secure Shell, 352
`GSSAPIDelegateCredentials` keyword, Solaris Secure Shell, 352
`GSSAPIKeyExchange` keyword, Solaris Secure Shell, 352
`GSSAPIStoreDelegatedCredentials` keyword, `ssh_config` file, 352
`gsscred` command, description, 531
`gsscred` table, using, 543
`gssd` daemon, Kerberos and, 532

H

`-h` option, `bsmrecord` command, 595
hard disk, space requirements for auditing, 567
hard string, `audit_warn` script, 615
hardware
  listing attached hardware accelerators, 285
  protecting, 38, 72-74
  requiring password for access, 72-73
hardware providers
  disabling cryptographic mechanisms, 285-287
  enabling mechanisms and features on, 286
  listing, 285
  loading, 285
hash
  algorithms
    Kerberos and, 385-386
hashing, files, 268
`header` audit token
  event-modifier field flags, 630

`header` audit token *(Continued)*
  format, 629-630
  order in audit record, 629-630
help
  SEAM Administration Tool, 477-478
  URL for online, 386
Help Contents, SEAM Administration Tool, 478
hierarchical realms
  configuring, 410-411
  in Kerberos, 369, 381
high ASET security level, 148
`hmac-md5` algorithm, `ssh_config` file, 353
`hmac-sha1` encryption algorithm, `ssh_config` file, 353
host-based authentication
  configuring in Solaris Secure Shell, 331-334
  description, 328
`Host` keyword
  `ssh_config` file, 352, 354
host names, mapping onto realms, 381
`host` principal
  creating, 396, 403
`HostbasedAuthentication` keyword, Solaris Secure Shell, 352
`HostbasedUsesNamesFromPacketOnly` keyword, `sshd_config` file, 352
`HostKey` keyword, `sshd_config` file, 352
`HostKeyAlgorithms` keyword, `ssh_config` file, 352
`HostKeyAlias` keyword, `ssh_config` file, 352
hosts
  disabling Kerberos service on, 510-512
  Solaris Secure Shell hosts, 328
  trusted hosts, 55
`hosts.equiv` file, description, 357

I

`-I` option
  `bart create` command, 102
  `st_clean` script, 95
`-i` option
  `bart create` command, 102, 107
  `encrypt` command, 273
  `st_clean` script, 95
identity files (Solaris Secure Shell), naming conventions, 356
`IdentityFile` keyword, `ssh_config` file, 352
IDs
  audit
    mechanism, 620
    overview, 547-548
  audit session, 620
  mapping UNIX to Kerberos principals, 543
`IgnoreRhosts` keyword, `sshd_config` file, 352
`IgnoreUserKnownHosts` keyword, `sshd_config` file, 352
`import` subcommand, `pktool` command, 293-294
`in_addr` audit token, format, 630
`in.ftpd` daemon, Kerberos and, 532
`in.rlogind` daemon, Kerberos and, 532
`in.rshd` daemon, Kerberos and, 532
`in.telnetd` daemon, Kerberos and, 532
include control flag, PAM, 319
inheritable privilege set, 189
initial ticket, definition, 534
`install` subcommand, `cryptoadm` command, 280
installing
  password encryption module, 68-69
  providers in cryptographic framework, 266
instance, in principal names, 368-369
integrity
  Kerberos and, 363
  security service, 371
interactively configuring
  Kerberos
    master KDC server, 390-391
    slave KDC server, 405-406
interactively running ASET, 165-166
INTERNAL plug-in, SASL and, 324
Internet firewall setup, 55-56
Internet-related tokens
  `in_addr` token, 630
  `ip` token, 630-631
  `iport` token, 632
  `socket` token, 636-637
invalid ticket, definition, 534
`ioctl` audit class, 618
`ioctl()` system calls, 618

`ioctl()` system calls *(Continued)*
  `AUDIO_SETINFO()`, 95
IP addresses, Solaris Secure Shell checking, 351
`ip` audit token, format, 630-631
IP MIB-II, getting information from `/dev/arp`, 79
`ipc` audit class, 618
`ipc` audit token, 631
  format, 631
`ipc_perm` audit token, format, 631-632
`IPC` privileges, 187
ipc type field values (`ipc` token), 631
`iport` audit token, format, 632
item size field, `arbitrary` token, 625

J
JASS toolkit, pointer to, 49

K
`-k` option
  `encrypt` command, 273
  Kerberized commands, 524
  `mac` command, 272
`-K` option
  Kerberized commands, 524
  `usermod` command, 244
`.k5.`*REALM* file, description, 530
`.k5login` file
  description, 520-522, 529
  rather than revealing password, 521
`kadm5.acl` file
  description, 529
  format of entries, 492
  master KDC entry, 394, 402, 440
  new principals and, 486, 488
`kadm5.keytab` file
  description, 505, 530
`kadmin` command
  creating `host` principal, 396, 403
  description, 531
  `ktadd` command, 507-508
  `ktremove` command, 509
`kadmin` command *(Continued)*
  removing principals from keytab with, 509
  SEAM Administration Tool and, 475
`kadmin.local` command
  adding administration principals, 394, 402
  automating creation of principals, 481
  creating keytab file, 395, 402
  description, 531
`kadmin.log` file, description, 530
`kadmind` daemon
  Kerberos and, 532
  master KDC and, 532
`kadmind` principal, 506
`kbd` file, 73-74
`KbdInteractiveAuthentication` keyword, Solaris Secure Shell, 352
`kcfd` daemon, 287-288
`kclient` command, description, 531
`kdb5_ldap_util` command, description, 531
`kdb5_util` command
  creating KDC database, 394
  creating stash file, 410, 452
  description, 531
KDC
  backing up and propagating, 442-443
  configuring master
    automatic, 390
    interactive, 390-391
    manual, 392-397
    with LDAP, 397-404
  configuring server, 389-410
  configuring slave
    automatic, 404-405
    interactive, 405-406
    manual, 406-410
  copying administration files from slave to master, 408, 450
  creating database, 394
  creating `host` principal, 396, 403
  database propagation, 383-384
  master
    definition, 532
  planning, 382
  ports, 382

- KDC *(Continued)*
  - restricting access to servers, 457-458
  - slave, 382
    - definition, 532
  - slave or master, 370, 389
  - starting daemon, 410, 452
  - swapping master and slave, 436-441
  - synchronizing clocks
    - master KDC, 397, 404
    - slave KDC, 410, 452
- `kdc.conf` file
  - description, 530
  - ticket lifetime and, 535
- `kdc.log` file, description, 530
- `kdcmgr` command
  - configuring master
    - automatic, 390
    - interactive, 391
  - configuring slave
    - automatic, 405
    - interactive, 405
  - server status, 391
- `kdestroy` command
  - example, 516
  - Kerberos and, 531
- `KeepAlive` keyword, Solaris Secure Shell, 352
- Kerberos
  - administering, 475-512
  - Administration Tool
    - *See* SEAM Administration Tool
  - commands, 522-528, 531-532
  - components of, 372-373
  - configuration decisions, 379-386
  - configuring KDC servers, 389-410
  - daemons, 532
  - `dfstab` file option, 420
  - enabling Kerberized applications only, 457
  - encryption types
    - overview, 385-386
    - using, 541-543
  - error messages, 459-471
  - examples of using Kerberized commands, 526-528
  - files, 529-530
  - gaining access to server, 538-540
- Kerberos *(Continued)*
  - granting access to your account, 520-522
  - Kerberos V5 protocol, 363
  - online help, 386
  - options to Kerberized commands, 523
  - overview
    - authentication system, 364-370, 537-538
    - Kerberized commands, 522-525
  - password management, 517-522
  - planning for, 379-386
  - realms
    - *See* realms (Kerberos)
  - reference, 529-544
  - remote applications, 368
  - table of network command options, 524
  - terminology, 532-537
  - troubleshooting, 471
  - using, 513-528
- Kerberos authentication
  - and Secure RPC, 300
  - `dfstab` file option, 420
- Kerberos commands, 522-528
  - enabling only Kerberized, 457
  - examples, 526-528
- `kern.notice` entry, `syslog.conf` file, 130
- kernel providers, listing, 277
- Key Distribution Center, *See* KDC
- key management framework (KMF), *See* KMF
- `KEYBOARD_ABORT` system variable, 73-74
- `keylogin` command
  - use for Secure RPC, 301
  - verifying DH authentication setup, 306
- `KeyRegenerationInterval` keyword, `sshd_config` file, 352
- keys
  - creating DH key for NIS user, 308-309
  - creating for Solaris Secure Shell, 336-338
  - definition in Kerberos, 533
  - generating for Solaris Secure Shell, 336-338
  - generating symmetric key
    - using the `dd` command, 268-270
  - service key, 505-512
  - session keys
    - Kerberos authentication and, 537-538

keys *(Continued)*
  using for MAC, 272-273
`keyserv` daemon, 305
keyserver
  description, 301
  starting, 305
keystores
  exporting certificates, 294-295
  importing certificates, 293-294
  listing contents, 292
  managed by KMF, 290
  protecting with password in KMF, 295-296
keytab file
  adding master KDC's host principal to, 397, 404
  adding service principal to, 505, 507-508
  administering, 505-512
  administering with `ktutil` command, 506
  creating, 395, 402
  disabling a host's service with `delete_entry` command, 511
  read into keytab buffer with `read_kt` command, 510, 511
  removing principals with `ktremove` command, 509
  removing service principal from, 509
  viewing contents with `ktutil` command, 509, 510
  viewing keylist buffer with `list` command, 510, 511
`keytab` option, SASL and, 325
keywords
  See also specific keyword
  attribute in BART, 117
  command-line overrides in Solaris Secure Shell, 359
  Solaris Secure Shell, 350-355
`kgcmgr` command, description, 531
`kinit` command
  example, 514
  `-F` option, 514
  Kerberos and, 531
  ticket lifetime, 535
`klist` command
  example, 515-516
  `-f` option, 515-516
  Kerberos and, 531
KMF
  creating
    passphrases for keystores, 291
    password for keystore, 295-296
    self-signed certificate, 291-292
  exporting certificates, 294-295
  importing certificates into keystore, 293-294
  library, 290
  managing
    keystores, 290
    PKI policy, 290
    public key technologies (PKI), 289
  utilities, 290
`kmfcfg` command, 289
`known_hosts` file
  controlling distribution, 355
  description, 357
Korn shell, privileged version, 184
`kpasswd` command
  error message, 518
  example, 519
  Kerberos and, 531
  `passwd` command and, 518
`kprop` command, description, 531
`kpropd.acl` file, description, 530
`kpropd` daemon, Kerberos and, 532
`kproplog` command, description, 531
`krb5.conf` file
  description, 530
  `domain_realm` section, 381
  editing, 392, 400
  ports definition, 382
`krb5.keytab` file, description, 530
`krb5cc_`*uid* file, description, 530
`krb5kdc` daemon
  Kerberos and, 532
  master KDC and, 532
  starting, 410, 452
`ksh` command, privileged version, 184
`ktadd` command
  adding service principal, 505, 507-508
  syntax, 507
`ktkt_warnd` daemon, Kerberos and, 532
`ktremove` command, 509

`ktutil` command
    administering keytab file, 506
    `delete_entry` command, 511
    Kerberos and, 531
    `list` command, 510, 511
    `read_kt` command, 510, 511
    viewing list of principals, 509, 510

L

`-L` option, `ssh` command, 342-344
`-l` option
    `digest` command, 270
    `encrypt` command, 268
    `mac` command, 271
    `praudit` command, 609
LDAP, configuring master KDC using, 397-404
LDAP name service
    passwords, 40
    specifying password algorithm, 68
least privilege, principle of, 186
libraries, user-level providers, 277
lifetime of ticket, in Kerberos, 535-536
limit privilege set, 189
limiting, use of privileges by user or role, 245-246
`limitpriv` keyword, `user_attr` database, 256
`list` command, 510, 511
`list_devices` command
    authorizations for, 91
    authorizations required, 238
    description, 91
list privilege, SEAM Administration Tool and, 504
`list` subcommand, `pktool` command, 292
`ListenAddress` keyword, `sshd_config` file, 352
listing
    available providers in cryptographic
        framework, 276-278
    contents of keystore, 292
    cryptographic framework providers, 285
    device policy, 76-77
    hardware providers, 285
    providers in the cryptographic framework, 276-278
    roles you can assume, 209, 236
    users with no passwords, 59-60
`LocalForward` keyword, `ssh_config` file, 352
log files
    audit records, 553, 601
    BART
        programmatic output, 118-119
        verbose output, 118-119
    configuring for auditing service, 573-575
    examining audit records, 607
    execution log (ASET), 151
    failed login attempts, 62-63
    monitoring `su` command, 70-71
    space for audit records, 606
    `syslog` audit records, 611
`log_level` option, SASL and, 325
`logadm` command, archiving textual audit files, 603
logging in
    and `AUTH_DH`, 301
    disabling temporarily, 60-61
    displaying user's login status, 58-59, 59
    log of failed logins, 62-63
    monitoring failures, 61
    `root` login
        account, 42
        restricting to console, 71-72
        tracking, 46
    security
        access control on devices, 43
        access restrictions, 39
        saving failed attempts, 61
        system access control, 39
        tracking `root` login, 46
    system logins, 42
    task map, 58
    users' basic privilege set, 190
    with Solaris Secure Shell, 339-340
`login` environment variables, Solaris Secure Shell
    and, 354-355
`login` file
    login default settings, 62
`.login` file, path variable entry, 47
`login` file
    restricting remote `root` access, 71-72
`login_logout` audit class, 618
`LoginGraceTime` keyword, `sshd_config` file, 352

`loginlog` file, saving failed login attempts, 61
`logins` command
  displaying user's login status, 58-59, 59
  displaying users with no passwords, 60
  syntax, 58
`LogLevel` keyword, Solaris Secure Shell, 352
`LookupClientHostname` keyword, `sshd_config` file, 352
low ASET security level, 148

M

`-M` option, `auditreduce` command, 598
`-m` option
  `cryptoadm` command, 281, 282
  Kerberized commands, 524
`mac` command
  description, 265
  syntax, 271
machine security, *See* system security
`MACS` keyword, Solaris Secure Shell, 353
mail, using with Solaris Secure Shell, 343
`makedbm` command, description, 236
managing
  See also administering
  audit files, 596-598, 603
  audit records task map, 594
  audit trail overflow, 603
  auditing, 569
  auditing in zones, 555, 616-617
  device allocation task map, 79-80
  devices, 79-80
  file permissions, 131
  keystores with KMF, 290
  passwords with Kerberos, 517-522
  privileges task map, 239
  RBAC task map, 213
manifests
  See also `bart create`
  control, 97
  customizing, 104-107
  file format, 115-116
  test, 99
manually configuring
  Kerberos
    master KDC server, 392-397
    master KDC server using LDAP, 397-404
    slave KDC server, 406-410
mapping
  host names onto realms (Kerberos), 381
  UIDs to Kerberos principals, 543
mapping GSS credentials, 383
mappings, events to classes (auditing), 552
mask (auditing)
  description of process preselection, 620
  system-wide process preselection, 612
mask ACL entries
  default entries for directories, 129
  description, 128-129
  setting, 138-140
master files (ASET), 149, 155
master KDC
  automatically configuring, 390
  configuring with LDAP, 397-404
  definition, 532
  interactively configuring, 390-391
  manually configuring, 392-397
  slave KDCs and, 370, 389
  swapping with slave KDC, 436-441
`max_life` value, description, 535
`max_renewable_life` value, description, 536
`MaxAuthTries` keyword, `sshd_config` file, 353
`MaxAuthTriesLog` keyword, `sshd_config` file, 353
`MaxStartups` keyword, `sshd_config` file, 353
MD5 encryption algorithm
  kernel provider, 277
  `policy.conf` file, 66-67
`mech_dh` mechanism
  GSS-API credentials, 349
  secure RPC, 305-306
`mech_krb` mechanism, GSS-API credentials, 349
`mech_list` option, SASL and, 325
mechanism, definition in cryptographic framework, 263
mechanisms
  disabling all on hardware provider, 285-287
  enabling some on hardware provider, 286

medium ASET security level, 148
merging, binary audit records, 596-598
message authentication code (MAC), computing for file, 271-273
`messages` file, executable stack messages, 130
metaslot
  administering, 264-265
  definition in cryptographic framework, 263
microphone
  allocating, 85-86
  deallocating, 88-89
`minfree` line
  `audit_control` file, 612
  `audit_warn` condition, 615
minus sign (-)
  audit class prefix, 618
  entry in `sulog` file, 70
  file permissions symbol, 126
  symbol of file type, 122
mode, definition in cryptographic framework, 263
modifying
  policies (Kerberos), 499-500
  principal's password (Kerberos), 489
  principals (Kerberos), 488-489
  role assignment to a user, 201
  roles (RBAC), 213-215
  users (RBAC), 218-219
modules, password encryption, 40
monitoring
  audit trail in real time, 568
  failed logins, 61
  `su` command attempts, 46, 70-71
  superuser access attempts, 71-72
  superuser task map, 69-70
  system usage, 50
  use of privileged commands, 206-207
`mount` command, with security attributes, 81
mounting
  allocated CD-ROM, 87-88
  allocated devices, 86-88
  allocated diskette, 87
  audit directories, 621
  files with DH authentication, 309
`mt` command, tape device cleanup and, 95

N
`-n` option
  `audit` command, 606
  `bart create` command, 102
`naflags` line
  `audit_control` file, 612
  `plugin` line and, 573
name services
  See also individual name services
  scope and RBAC, 184
names
  audit classes, 617
  audit files, 621
  device names
    `device_maps` file, 93
naming conventions
  audit directories, 572, 612
  audit files, 621
  devices, 82
  RBAC authorizations, 228
  Solaris Secure Shell identity files, 356
`NET` privileges, 187
network, privileges relating to, 187
`network` audit class, 618
network security
  authentication, 53-54
  authorizations, 53-54
  controlling access, 52-56
  firewall systems
    need for, 55
    packet smashing, 56
    trusted hosts, 55
  overview, 52
  reporting problems, 56
Network Security (RBAC), creating role, 201
Network Time Protocol, *See* NTP
never-audit classes, `audit_user` database, 614
new features
  auditing enhancements, 556-557
  BART, 97-119
  commands
    `bart compare`, 99
    `bart create`, 98-99
    `cryptoadm`, 276

new features, commands *(Continued)*
        `decrypt`, 274
        `digest`, 270-271
        `encrypt`, 273-275
        `getdevpolicy`, 76-77
        `kcfd`, 287-288
        `kclient`, 374
        `kpropd`, 374
        `mac`, 271-273
        `ppriv`, 240-241
        `praudit -x`, 601
        `ssh-keyscan`, 359
        `ssh-keysign`, 359
    cryptographic framework, 261-266
    device policy, 45
    Kerberos enhancements, 374-376
    metaslot, 261
    PAM enhancements, 313-314
    privileges, 185-193
    process rights management, 185-193
    SASL, 323
    Solaris Cryptographic Framework, 261-266
    Solaris Secure Shell enhancements, 330
    strong password encryption, 40
    system security enhancements, 37-38
`newkey` command
    creating key for NIS user, 308-309
    generating keys, 301
NFS file systems
    ASET and, 159
    authentication, 299
    providing client-server security, 301-304
    secure access with `AUTH_DH`, 309
NFS servers, configuring for Kerberos, 415-417
NIS+ name service
    adding authenticated user, 307
    ASET checks, 158
    authentication, 299
    `cred` database, 306
    `cred` table, 301
    passwords, 40
    specifying password algorithm, 67
NIS name service
    authentication, 299
NIS name service *(Continued)*
    passwords, 40
    specifying password algorithm, 67
`nisaddcred` command
    adding client credential, 306
    generating keys, 301
`no_class` audit class, 617
`nobody` user, 51-52
`noexec_user_stack_log` variable, 130, 145
`noexec_user_stack` variable, 130, 145
`NoHostAuthenticationForLocalHost` keyword, `ssh_config` file, 353
`nologin` file, description, 357
`non_attrib` audit class, 617
nonattributable classes, 612
nonhierarchical realms, in Kerberos, 369
`nscd` (name service cache daemon)
    starting with `svcadm` command, 200
    use, 236
NSS, managing keystore, 290
`nsswitch.conf` file, login access restrictions, 39
NTP
    Kerberos planning and, 384
    master KDC and, 397, 404
    slave KDC and, 410, 452
`null` audit class, 617
`NumberOfPasswordPrompts` keyword, `ssh_config` file, 353

O

`-O` option, `auditreduce` command, 596-598
`-o` option, `encrypt` command, 273
object reuse requirements
    device-clean scripts
        tape drives, 94
        writing new scripts, 95-96
    for devices, 94-96
obtaining
    access to a specific service, 540
    credential for a server, 539
    credential for a TGS, 538
    forwardable tickets, 514
    privileged commands, 213-215

obtaining *(Continued)*
  privileges, 190, 191, 244-245
  privileges on a process, 240-241
  tickets with `kinit`, 514
online help
  SEAM Administration Tool, 477-478
  URL for, 386
`opaque` audit token, format, 632
OpenSSH, *See* Solaris Secure Shell
OpenSSL, managing keystore, 290
Operator (RBAC)
  contents of rights profile, 225
  creating role, 200
  recommended role, 176
optional control flag, PAM, 319
options to Kerberized commands, 523
other ACL entries, description, 128-129
`other` audit class, 618
overflow prevention, audit trail, 603
`ovsec_adm.`*xxxxx* file, description, 530
ownership of files
  ACLs and, 51, 127-130
  changing, 122, 133
  changing group ownership, 134

P
`-p` option
  `aset` command, 166
  `bart create`, 107
  `bsmrecord` command, 595
  `cryptoadm` command, 281, 282
  `logins` command, 60
packages, Solaris Secure Shell, 356
packet transfers
  firewall security, 55
  packet smashing, 56
PAM
  adding a module, 316
  configuration file
    control flags, 319
    introduction, 317
    stacking diagrams, 319
    stacking example, 321
PAM, configuration file *(Continued)*
    stacking explained, 318
    syntax, 317
  `/etc/syslog.conf` file, 317
  framework, 312
  Kerberos and, 373, 377
  overview, 311
  planning, 315
  task map, 314
`pam.conf` file
  See PAM configuration file
  Kerberos and, 530
`pam_roles` command, description, 236
`PAMAuthenticationViaKBDInt` keyword, `sshd_config` file, 353
panels, table of SEAM Administration Tool, 501-504
passphrases
  changing for Solaris Secure Shell, 339
  `encrypt` command, 273
  example, 340
  generating in KMF, 295-296
  `mac` command, 272
  storing safely, 274
  using for MAC, 272
  using in Solaris Secure Shell, 338, 340-341
`PASSREQ` in Solaris Secure Shell, 354
`passwd` command
  and `kpasswd` command, 518
  and name services, 40
`passwd` file
  and `/etc/d_passwd` file, 43
  ASET checks, 150
password authentication, Solaris Secure Shell, 328
`PasswordAuthentication` keyword, Solaris Secure Shell, 353
passwords
  authentication in Solaris Secure Shell, 328
  changing with `kpasswd` command, 518
  changing with `passwd -r` command, 40
  changing with `passwd` command, 518
  creating for dial-up, 63-65
  dial-up passwords
    disabling temporarily, 65
    `/etc/d_passwd` file, 43

passwords *(Continued)*
  disabling dial-up temporarily, 65
  displaying users with no passwords, 60
  eliminating in Solaris Secure Shell, 340-341
  eliminating in Solaris Secure Shell in CDE, 342
  encryption algorithms, 40
  finding users with no passwords, 59-60
  granting access without revealing, 520-522
  hardware access and, 72-73
  installing third-party encryption module, 68-69
  LDAP, 40
    specifying new password algorithm, 68
  local, 40
  login security, 39
  managing, 517-522
  modifying a principal's password, 489
  NIS, 40
    specifying new password algorithm, 67
  NIS+, 40
    specifying new password algorithm, 67
  policies and, 518
  PROM security mode, 38, 72-74
  protecting
    keystore, 295
    PKCS #12 file, 295
  requiring for hardware access, 72-73
  secret-key decryption for Secure RPC, 301
  specifying algorithm, 66-67
    in name services, 67
    locally, 65
  suggestions on choosing, 517-518
  system logins, 39, 42
  task map, 58
  UNIX and Kerberos, 517-522
  using Blowfish encryption algorithm for, 67
  using MD5 encryption algorithm for, 66-67
  using new algorithm, 66
`path_attr` audit token, 556, 633
`path` audit policy, description, 565
`path` audit token, format, 632-633
`PATH` environment variable
  and security, 47
  setting, 47
`PATH` in Solaris Secure Shell, 355
`PERIODIC_SCHEDULE` variable (ASET), 157, 161
permissions
  ACLs and, 51, 127-130
  ASET handling of, 148, 149
  changing file permissions
    absolute mode, 126, 135-136
    `chmod` command, 122
    symbolic mode, 126, 134-135, 135
  defaults, 125
  directory permissions, 122-123
  file permissions
    absolute mode, 126, 135-136
    changing, 125-127, 135
    description, 122-123
    special permissions, 125, 127
    symbolic mode, 126, 134-135, 135
  finding files with `setuid` permissions, 143
  `setgid` permissions
    absolute mode, 127, 137
    description, 124
    symbolic mode, 126
  `setuid` permissions
    absolute mode, 127, 137
    description, 124
    security risks, 124
    symbolic mode, 126
  special file permissions, 123-125, 125, 127
  sticky bit, 124-125
  tune files (ASET), 155, 158
  `umask` value, 125
  user classes and, 122
`PermitEmptyPasswords` keyword, `sshd_config` file, 353
`PermitRootLogin` keyword, `sshd_config` file, 353
permitted privilege set, 189
`PermitUserEnvironment` keyword, `sshd_config` file, 353
`perzone` audit policy
  description, 565
  setting, 585-586
  using, 561, 592-593, 616-617
  when to use, 555
`pfcsh` command, description, 184
`pfexec` command, description, 236

`pfksh` command, description, 184
`pfsh` command, description, 184
physical security, description, 38
PKCS #11 library
  adding as provider, 280
  in Solaris Cryptographic Framework, 262
PKCS #11 softtokens, managing keystore, 290
PKCS #12 files, protecting, 295
`pkcs11_kernel.so` user-level provider, 277
`pkcs11_softtoken.so` user-level provider, 277
`pkgadd` command
  installing third-party providers, 279
  installing third-party software, 69
PKI
  managed by KMF, 289
  policy managed by KMF, 290
`pktool` command
  creating self-signed certificate, 291-292
  `export` subcommand, 294-295
  `gencert` subcommand, 291-292
  `import` subcommand, 293-294
  `list` subcommand, 292
  managing PKI objects, 289
  `setpin` subcommand, 295-296
`plain.so.1` plug-in, SASL and, 324
planning
  auditing, 560-564
  auditing in zones, 560-561
  auditing task map, 559
  Kerberos
    client and service principal names, 381-382
    clock synchronization, 384
    configuration decisions, 379-386
    database propagation, 383-384
    number of realms, 380
    ports, 382
    realm hierarchy, 381
    realm names, 380
    realms, 380-381
    slave KDCs, 382
  PAM, 315
  RBAC, 197-199
plug-ins, SASL and, 324
pluggable authentication module, *See* PAM
`plugin` line
  `audit_control` file, 612
  `flags` line and, 573
`plugin_list` option, SASL and, 325
plugins
  in auditing service, 573
  in cryptographic framework, 262
plus sign (+)
  ACL entry, 138
  audit class prefix, 618
  entry in `sulog` file, 70
  file permissions symbol, 126
policies
  administering, 475-512
  creating (Kerberos), 486
  creating new (Kerberos), 497-498
  deleting, 500-501
  for auditing, 564-566
  modifying, 499-500
  on devices, 76-77
  overview, 33-34
  passwords and, 518
  SEAM Administration Tool panels for, 501-504
  specifying password algorithm, 65
  task map for administering, 493
  viewing attributes, 495-497
  viewing list of, 493-495
policy
  definition in cryptographic framework, 263
  definition in Solaris OS, 33-34
`policy.conf` file
  adding password encryption module, 68-69
  Basic Solaris User rights profile, 226
  description, 235-236, 236
  keywords
    for password algorithms, 41
    for privileges, 235, 256
    for RBAC authorizations, 235
    for rights profiles, 235
  specifying encryption algorithms in, 66-67
  specifying password algorithm
    in name services, 67
  specifying password algorithms, 66-67

port forwarding
    configuring in Solaris Secure Shell, 334-335
    Solaris Secure Shell, 343, 344
`Port` keyword, Solaris Secure Shell, 353
ports, for Kerberos KDC, 382
postdated ticket
    definition, 534
    description, 364
postsigterm string, `audit_warn` script, 615
pound sign (#)
    `device_allocate` file, 93
    `device_maps` file, 92
`ppriv` command
    for debugging, 242
    listing privileges, 240
`praudit` command
    converting audit records to readable format, 601, 609
    DTD for `-x` option, 609
    options, 609
    output formats, 609
    piping `auditreduce` output to, 601
    use in a script, 609-610
    viewing audit records, 600-602
    with no options, 609
    XML format, 602
`PreferredAuthentications` keyword, `ssh_config` file, 353
prefixes for audit classes, 618
preselecting, audit classes, 571-573
preselection in auditing, 550
preselection mask (auditing)
    description, 620
    reducing storage costs, 606
    system-wide, 612
preventing
    access to system hardware, 72
    audit trail overflow, 603
    executables from compromising security, 130
    kernel software provider use, 282-284
    use of hardware mechanism, 285-287
primary, in principal names, 368-369
Primary Administrator (RBAC)
    assuming role, 210
Primary Administrator (RBAC) *(Continued)*
    recommended role, 176
    rights profile contents, 224
primary audit directory, 612
principal
    adding administration, 394, 402
    adding service principal to keytab, 505, 507-508
    administering, 475-512
    automating creation of, 481-482
    creating, 486-488
    creating `clntconfig`, 396, 404
    creating `host`, 396, 403
    deleting, 489-490
    duplicating, 488
    Kerberos, 368-369
    modifying, 488-489
    principal name, 368-369
    removing from keytab file, 509
    removing service principal from keytab, 509
    SEAM Administration Tool panels for, 501-504
    service principal, 369
    setting up defaults, 490-491
    task map for administering, 481
    user ID comparison, 417
    user principal, 369
    viewing attributes, 484-486
    viewing list of, 482-484
    viewing sublist of principals, 483
`principal` file, description, 530
`principal.kadm5` file, description, 530
`principal.kadm5.lock` file, description, 530
`principal.ok` file, description, 530
`principal.ulog` file, description, 530
principle of least privilege, 186
print format field, `arbitrary` token, 625
Printer Management rights profile, 225
printing, audit log, 601
`PrintMotd` keyword, `sshd_config` file, 353
`priv.debug` entry, `syslog.conf` file, 257
`PRIV_DEFAULT` keyword
    `policy.conf` file, 235, 256
`PRIV_LIMIT` keyword
    `policy.conf` file, 235, 256
`PRIV_PROC_LOCK_MEMORY` privilege, 175, 188-189

privacy
  availability, 524
  Kerberos and, 363
  security service, 371
private keys
  See also secret keys
  definition in Kerberos, 533
  Solaris Secure Shell identity files, 356
private protection level, 524
`privilege` audit token, 556, 633-634
privilege checking, in applications, 182
privilege sets
  adding privileges to, 191
  basic, 190
  effective, 189
  inheritable, 189
  limit, 189
  listing, 190
  permitted, 189
  removing privileges from, 192
privileged application
  authorization checking, 182
  description, 178
  ID checking, 181
  privilege checking, 182
privileged ports, alternative to Secure RPC, 54
privileges
  adding to command, 243-244
  administering, 239
  assigning to a command, 191
  assigning to a script, 192
  assigning to a user, 191
  assigning to user or role, 244-245
  auditing and, 257
  categories, 186
  commands, 255
  compared to superuser model, 185-193
  debugging, 193, 242
  description, 178, 186
  determining directly assigned ones, 248-249
  devices and, 192
  differences from superuser model, 187
  effects on SEAM Administration Tool, 504
  escalation, 257
privileges *(Continued)*
  executing commands with privilege, 191
  files, 256-257
  finding missing, 242-243
  how to use, 247
  implemented in sets, 189
  inherited by processes, 190
  limiting use by user or role, 245-246
  listing on a process, 240-241
  `PRIV_PROC_LOCK_MEMORY`, 175, 188-189
  processes with assigned privileges, 190
  programs aware of privileges, 190
  protecting kernel processes, 185
  removing from a user, 192
  removing from basic set, 246
  removing from limit set, 246
  task map, 239
  using in shell script, 247
`privileges` file, description, 187
`PROC` privileges, 187
process audit characteristics
  audit ID, 620
  audit session ID, 620
  process preselection mask, 620
  terminal ID, 620
`process` audit class, 618
`process` audit token, format, 634-635
`process modify` audit class, 618
process preselection mask, description, 620
process privileges, 187
process rights management, *See* privileges
`process start` audit class, 618
processing time costs, of auditing service, 566
`prof_attr` database
  description, 233-234
  summary, 229
`.profile` file, path variable entry, 47
profile shells, description, 184
profiles, *See* rights profiles
`profiles` command, description, 236
`PROFS_GRANTED` keyword, `policy.conf` file, 235
programs
  checking for RBAC authorizations, 220
  privilege-aware, 189, 190

`project.max-locked-memory` resource control, 175, 188-189
PROM security mode, 72-74
propagation
  KDC database, 383-384
  Kerberos database, 442-443
protecting
  BIOS, pointer to, 72-73
  by using passwords with cryptographic framework, 291
  contents of keystore, 295
  files with cryptographic framework, 268
  PROM, 72-73
  system from risky programs, 143-145
protecting files
  task map, 131
  user procedures, 131-137
  with ACLs, 127-130, 137-143
  with ACLs task map, 137
  with UNIX permissions, 121-127, 131-137
  with UNIX permissions task map, 131
protection level
  clear, 524
  private, 524
  safe, 524
  setting in `ftp`, 524
`Protocol` keyword, `ssh_config` file, 353
providers
  adding library, 280
  adding software provider, 279-280
  adding user-level software provider, 280
  connecting to cryptographic framework, 266
  definition as plugins, 262
  definition in cryptographic framework, 263
  disabling hardware mechanisms, 285-287
  installing, 266
  listing hardware providers, 285
  listing in cryptographic framework, 276-278
  preventing use of kernel software provider, 282-284
  registering, 266
  restoring use of kernel software provider, 283
  signing, 266
proxiable ticket, definition, 535
proxy ticket, definition, 535
`ProxyCommand` keyword, `ssh_config` file, 353
pseudo-tty, use in Solaris Secure Shell, 349
`PubkeyAuthentication` keyword, Solaris Secure Shell, 353
`public` audit policy
  description, 566
  read-only events, 566
public directories
  auditing, 550
  sticky bit and, 125
public key authentication, Solaris Secure Shell, 328
public key cryptography
  `AUTH_DH` client-server session, 301-304
  changing NFS public keys and secret keys, 301
  common keys
    calculation, 303
  database of public keys for Secure RPC, 301
  generating keys
    conversation keys for Secure NFS, 302
    using Diffie-Hellman, 301
  NFS secret keys, 301
public key technologies, *See* PKI
public keys
  changing passphrase, 339
  DH authentication and, 300-304
  generating public-private key pair, 336-338
  Solaris Secure Shell identity files, 356
public objects, auditing, 550
`publickey` map, DH authentication, 300-304
`pwcheck_method` option, SASL and, 325

Q

question mark (?), in ASET tune files, 163
quoting syntax in BART, 117

R

`-R` option
  `bart create`, 102, 107
  `ssh` command, 342-344
`-r` option
  `bart create`, 107

- `-r` option *(Continued)*
  - `passwd` command, 40
  - `praudit` command, 609
- random numbers, `dd` command, 268-270
- raw `praudit` output format, 609
- RBAC
  - adding custom roles, 204-205
  - adding new rights profile, 217
  - adding roles, 199-202
  - adding roles from command line, 202-205
  - administration commands, 236-237
  - audit profiles, 616
  - auditing roles, 206-207
  - authorization database, 232-233
  - authorizations, 181
  - basic concepts, 178-181
  - changing user properties
    - from command line, 219
  - checking scripts or programs for
    - authorizations, 220
  - commands for managing, 236-237
  - compared to superuser model, 176-178
  - configuring, 197-208
  - database relationships, 229-230
  - databases, 229-236
  - editing rights profiles, 215-218
  - elements, 178-181
  - modifying roles, 213-215
  - modifying users, 218-219
  - name services and, 230
  - planning, 197-199
  - profile shells, 184
  - rights profile database, 233-234
  - rights profiles, 183
  - securing scripts, 220
  - using privileged applications, 211-212
- RC4, *See* ARCFOUR kernel provider
- `rcp` command
  - Kerberos and, 522-525, 531
- `rdist` command, Kerberos and, 531
- `read_kt` command, 510, 511
- read permissions, symbolic mode, 126
- readable audit record format
  - converting audit records to, 601, 609
- realms (Kerberos)
  - configuration decisions, 380-381
  - configuring cross-realm authentication, 410-412
  - contents of, 370
  - direct, 411-412
  - hierarchical, 410-411
  - hierarchical or nonhierarchical, 369
  - hierarchy, 381
  - in principal names, 368-369
  - mapping host names onto, 381
  - names, 380
  - number of, 380
  - requesting tickets for specific, 524
  - servers and, 370
- `reauth_timeout` option, SASL and, 325
- redirecting arrow (>), preventing redirection, 48
- reducing
  - audit files, 596-598, 607
  - storage-space requirements for audit files, 568
- refreshing, cryptographic services, 287-288
- registering providers, cryptographic framework, 266
- `rem_drv` command, description, 90
- remote logins
  - authentication, 53-54
  - authorization, 53-54
  - preventing superuser from, 71-72
  - security and, 303
- `RemoteForward` keyword, `ssh_config` file, 353
- removing
  - ACL entries, 141-142
  - cryptographic providers, 282
  - device policy, 78
  - policy from device, 78
  - principals with `ktremove` command, 509
  - privileges from basic set, 246
  - privileges from limit set, 246
  - service principal from keytab file, 509
  - software providers
    - permanently, 284
    - temporarily, 283
- renewable ticket, definition, 535
- replacing, superuser with roles, 197-199
- replayed transactions, 303
- reporting tool, *See* `bart compare`

reports
  ASET, 153, 154, 159
  BART, 97
  comparing (ASET), 155
  directory (ASET), 153
required control flag, PAM, 319
requisite control flag, PAM, 319
resource controls
  privileges, and, 175, 188-189
  `project.max-locked-memory`, 175, 188-189
  `zone.max-locked-memory`, 175, 188-189
restarting
  audit daemon, 589
  cryptographic services, 287-288
  `ssh` service, 335
  `sshd` daemon, 335
restoring, cryptographic providers, 283
restricted shell (`rsh`), 47
restricting
  superuser task map, 69-70
  user privileges, 246
restricting access for KDC servers, 457-458
`RETRIES` in Solaris Secure Shell, 355
`return` audit token, format, 635
`rewoffl` option
  `mt` command
    tape device cleanup and, 95
`.rhosts` file, description, 357
`RhostsAuthentication` keyword, Solaris Secure Shell, 353
`RhostsRSAAuthentication` keyword, Solaris Secure Shell, 353
right, *See* rights profiles
rights profiles
  for auditing service, 616
  changing contents of, 215-218
  changing from command line, 216
  contents of typical, 223
  creating
    in Solaris Management Console, 217
    on command line, 216
  creating roles for, 199-202
  databases
    See `prof_attr` database and `exec_attr` database
rights profiles *(Continued)*
  description, 178, 183
  major rights profiles descriptions, 223
  methods of creating, 215-218
  modifying, 215-218
  ordering, 227
  troubleshooting, 217
  using the System Administrator profile, 72
  viewing contents, 227
Rights tool, description, 215-218
`rlogin` command
  Kerberos and, 522-525, 531
`rlogind` daemon, Kerberos and, 532
role-based access control, *See* RBAC
`roleadd` command
  description, 236
  using, 202
`roledel` command, description, 236
`rolemod` command
  changing properties of role, 214
  description, 236
roles
  adding custom roles, 204-205
  adding for particular profiles, 199-202
  adding from command line, 202-205
  assigning privileges to, 244-245
  assigning with `usermod` command, 205-206
  assuming, 209-211, 211-212
  assuming after login, 183
  assuming in a terminal window, 184, 209-211
  assuming in Solaris Management Console, 211-212
  assuming Primary Administrator role, 210
  assuming `root` role, 210
  assuming System Administrator role, 211
  auditing, 206-207
  changing properties of, 213-215
  creating
    Crypto Management role, 206
    Custom Operator role, 204-205
    Device Security role, 201
    DHCP Management role, 201
    for particular profiles, 199-202
    Network Security role, 201
    on command line, 202-205

roles, creating *(Continued)*
    Operator role, 200
    role with limited scope, 201
    `root` role, 207-208
    security-related roles, 201
    System Administrator role, 200
  description, 183-184
  determining directly assigned privileges, 249
  determining role's privileged commands, 251-253
  listing local roles, 209, 236
  making `root` user into role, 207-208
  modifying, 213-215
  modifying assignment to a user, 201
  recommended roles, 176
  summary, 178
  troubleshooting, 202
  use in RBAC, 176
  using an assigned role, 209-211, 211-212
  using to access the hardware, 72-73
`roles` command
  description, 236
  using, 209
`root` principal, adding to host's keytab, 506
`root` role (RBAC), assuming role, 210
`root` user
  changing to `root` role, 207-208
  displaying access attempts on console, 71-72
  login account
    description, 42
  monitoring `su` command attempts, 46, 70-71
  replacing in RBAC, 183
  restricting access, 51-52
  restricting remote access, 71-72
  tracking logins, 46
RPCSEC_GSS API, Kerberos and, 378
RSA kernel provider, 277
`RSAAuthentication` keyword, Solaris Secure Shell, 353
`rsh` command
  Kerberos and, 522-525, 531
`rsh` command (restricted shell), 47
`rshd` daemon, Kerberos and, 532
`rstchown` system variable, 133
rules file (BART), 99-100
rules file attributes, *See* keywords
rules file format (BART), 116-117
rules file specification language, *See* quoting syntax
Running ASET task map, 164-168

S
`-S` option, `st_clean` script, 96
`-s` option
  `audit` command, 606
  `praudit` command, 609
safe protection level, 524
SASL
  environment variable, 324
  options, 325
  overview, 323
  plug-ins, 324
`saslauthd_path` option, SASL and, 325
saving, failed login attempts, 61
scope (RBAC), description, 184
`scp` command
  copying files with, 344
  description, 359
scripts
  `audit_startup` script, 613
  `audit_warn` script, 614
  `bsmconv` effect, 611
  `bsmconv` for device allocation, 80
  `bsmconv` script, 616
  `bsmconv` to enable auditing, 586-587
  checking for RBAC authorizations, 220
  device-clean scripts
    See also device-clean scripts
  for cleaning devices, 94-96
  monitoring audit files example, 568
  processing `praudit` output, 609-610
  running with privileges, 192
  securing, 220
  use of privileges in, 247
SCSI devices, `st_clean` script, 94
SEAM Administration Tool
  and limited administration privileges, 504-505
  and list privileges, 504
  and X Window system, 476-477
  command-line equivalents, 476-477

SEAM Administration Tool *(Continued)*
    context-sensitive help, 477
    creating a new policy, 486, 497-498
    creating a new principal, 486-488
    default values, 479
    deleting a principal, 489-490
    deleting policies, 500-501
    displaying sublist of principals, 483
    duplicating a principal, 488
    files modified by, 477
    Filter Pattern field, 483
    `gkadmin` command, 475
    `.gkadmin` file, 477
    help, 477-478
    Help Contents, 478
    how affected by privileges, 504
    `kadmin` command, 475
    login window, 478
    modifying a policy, 499-500
    modifying a principal, 488-489
    online help, 477-478
    or `kadmin` command, 476
    overview, 476-479
    panel descriptions, 501-504
    privileges, 504
    setting up principal defaults, 490-491
    starting, 478-479
    table of panels, 501-504
    viewing a principal's attributes, 484-486
    viewing list of policies, 493-495
    viewing list of principals, 482-484
    viewing policy attributes, 495-497
secondary audit directory, 612
secret keys
    creating, 268-270
    generating, 268-270
        using the `dd` command, 268-270
    generating for Secure RPC, 301
secure connection
    across a firewall, 345
    logging in, 339-340
Secure NFS, 300
Secure RPC
    alternative, 54
Secure RPC *(Continued)*
    and Kerberos, 300
    description, 299
    implementation of, 301-304
    keyserver, 301
    overview, 53-54
securing
    logins task map, 58
    passwords task map, 58
    scripts, 220
security
    across insecure network, 345
    auditing and, 549
    BART, 101
    computing digest of files, 270-271
    computing MAC of files, 271-273
    devices, 44-46
    DH authentication, 301-304
    encrypting files, 273-275
    Kerberos authentication, 420
    NFS client-server, 301-304
    password encryption, 40
    pointer to JASS toolkit, 49
    policy overview, 33-34
    preventing remote login, 71-72
    protecting against denial of service, 49
    protecting against Trojan horse, 47
    protecting devices, 94-96
    protecting hardware, 72-74
    protecting PROM, 72-74
    system hardware, 72-74
security attributes
    checking for, 181
    considerations when directly assigning, 184-185
    description, 178
    Printer management rights profile, 180
    privileges on commands, 182
    special ID on commands, 182
    using to mount allocated device, 81
security mechanism, specifying with `-m` option, 524
security modes, setting up environment with multiple, 419-421
security policy, default (RBAC), 229
security service, Kerberos and, 371

selecting
  audit classes, 571-573
  audit records, 598-600
  events from audit trail, 598-600
semicolon (;)
  `device_allocate` file, 93
  separator of security attributes, 235
`sendmail` command, authorizations required, 238
`seq` audit policy
  and `sequence` token, 566, 636
  description, 566
`sequence` audit token
  and `seq` audit policy, 636
  format, 636
`ServerKeyBits` keyword, `sshd_config` file, 353
servers
  `AUTH_DH` client-server session, 301-304
  configuring for Solaris Secure Shell, 350
  definition in Kerberos, 533
  gaining access with Kerberos, 538-540
  obtaining credential for, 539
  realms and, 370
service
  definition in Kerberos, 533
  disabling on a host, 510-512
  obtaining access for specific service, 540
service keys
  definition in Kerberos, 533
  keytab files and, 505-512
service management facility
  enabling keyserver, 305
  refreshing cryptographic framework, 279
  restarting cryptographic framework, 287-288
  restarting Solaris Secure Shell, 335
service principal
  adding to keytab file, 505, 507-508
  description, 369
  planning for names, 381-382
  removing from keytab file, 509
session ID, audit, 620
session keys
  definition in Kerberos, 533
  Kerberos authentication and, 537-538
`setfacl` command
  `-d` option, 142
  description, 129-130
  examples, 141
  `-f` option, 140
  syntax, 138-140
`setgid` permissions
  absolute mode, 127, 137
  description, 124
  security risks, 124
  symbolic mode, 126
`setpin` subcommand, `pktool` command, 295-296
setting
  audit policy, 584-586
  principal defaults (Kerberos), 490-491
`setuid` permissions
  absolute mode, 127, 137
  description, 124
  finding files with permissions set, 143
  security risks, 48, 124
  symbolic mode, 126
`sftp` command, description, 359
`sh` command, privileged version, 184
SHA1 kernel provider, 277
sharing files
  and network security, 51
  with DH authentication, 309
shell, privileged versions, 184
shell commands
  `/etc/d_passwd` file entries, 44
  passing parent shell process number, 240
shell process, listing its privileges, 240-241
shell scripts, writing privileged, 247
short `praudit` output format, 609
`shosts.equiv` file, description, 357
`.shosts` file, description, 357
signal received during auditing shutdown, 615
signing providers, cryptographic framework, 266
single-sign-on system, 522-528
  Kerberos and, 363
size of audit files
  reducing, 596-598, 607
  reducing storage-space requirements, 568

`slave_datatrans` file
  description, 530
  KDC propagation and, 442-443
`slave_datatrans_`*`slave`* file, description, 530
slave KDCs
  automatically configuring, 404-405
  configuring, 406-410
  definition, 532
  interactively configuring, 405-406
  master KDC and, 370
  or master, 389
  planning for, 382
  swapping with master KDC, 436-441
slot, definition in cryptographic framework, 264
smartcard documentation, pointer to, 32
`smattrpop` command, description, 237
`smexec` command, description, 237
`smmultiuser` command, description, 237
`smprofile` command
  changing rights profile, 216
  description, 237
`smrole` command
  changing properties of role, 214
  description, 237
  using, 204-205
`smuser` command
  changing user's RBAC properties, 218
  description, 237
`socket` audit token, 636-637
soft limit
  `audit_warn` condition, 615
  `minfree` line description, 612
soft string, `audit_warn` script, 615
Solaris auditing task map, 569
Solaris Cryptographic Framework, *See* cryptographic framework
`solaris.device.revoke` authorization, 91
Solaris Secure Shell
  adding to system, 356
  administering, 347-349
  administrator task map, 331
  authentication
    requirements for, 328-330
  authentication methods, 328-330
Solaris Secure Shell *(Continued)*
  authentication steps, 348-349
  basis from OpenSSH, 330
  changes in current release, 330
  changing passphrase, 339
  command execution, 349
  configuring clients, 350
  configuring port forwarding, 334-335
  configuring server, 350
  connecting across a firewall, 345
  connecting outside firewall
    from command line, 346
    from configuration file, 345-346
  copying files, 344
  creating keys, 336-338
  data forwarding, 349
  description, 327
  files, 356
  forwarding mail, 343
  generating keys, 336-338
  keywords, 350-355
  local port forwarding, 343, 344
  logging in fewer prompts, 340-341
  logging in to remote host, 339-340
  login environment variables and, 354-355
  naming identity files, 356
  packages, 356
  protocol versions, 327
  public key authentication, 328
  remote port forwarding, 344
  `scp` command, 344
  TCP and, 335
  typical session, 347-349
  user procedures, 335-336
  using port forwarding, 342-344
  using without password, 340-341
`solaris` security policy, 234
special permissions
  `setgid` permissions, 124
  `setuid` permissions, 124
  sticky bit, 124-125
square brackets (`[]`), `bsmrecord` output, 623
`sr_clean` script, description, 95

`ssh-add` command
  description, 359
  example, 340-341, 341
  storing private keys, 340-341
`ssh-agent` command
  configuring for CDE, 342
  description, 359
  from command line, 340-341
  in scripts, 342
`ssh` command
  description, 359
  overriding keyword settings, 359
  port forwarding options, 342-344
  using, 339-340
  using a proxy command, 346
`.ssh/config` file
  description, 358
  override, 358
`ssh_config` file
  configuring Solaris Secure Shell, 350
  host-specific parameters, 354
  keywords, 350-355
    See specific keyword
  override, 358
`.ssh/environment` file, description, 357
`ssh_host_dsa_key` file, description, 356
`ssh_host_dsa_key.pub` file, description, 356
`ssh_host_key` file
  description, 356
  override, 358
`ssh_host_key.pub` file, description, 356
`ssh_host_rsa_key` file, description, 356
`ssh_host_rsa_key.pub` file, description, 356
`.ssh/id_dsa` file, 358
`.ssh/id_rsa` file, 358
`.ssh/identity` file, 358
`ssh-keygen` command
  description, 359
  using, 336-338
`ssh-keyscan` command, description, 359
`ssh-keysign` command, description, 359
`.ssh/known_hosts` file
  description, 357
  override, 358
`ssh_known_hosts` file, 357
`.ssh/rc` file, description, 357
`sshd` command, description, 359
`sshd_config` file
  description, 356
  keywords, 350-355
    See specific keyword
  overrides of `/etc/default/login` entries, 354-355
`sshd.pid` file, description, 357
`sshrc` file, description, 358
`st_clean` script
  description, 94-95
  for tape drives, 94
standard cleanup, `st_clean` script, 96
starting
  ASET from shell, 148
  ASET interactively, 165-166
  audit daemon, 590
  auditing, 586-587
  device allocation, 80-81
  KDC daemon, 410, 452
  running ASET periodically, 166
  Secure RPC keyserver, 305
stash file
  creating, 410, 452
  definition, 532
sticky bit permissions
  absolute mode, 127, 137
  description, 124-125
  symbolic mode, 126
stopping, dial-up logins temporarily, 65
storage costs, and auditing, 567
storage overflow prevention, audit trail, 603
storing
  audit files, 561-562, 580-583
  passphrase, 274
`StrictHostKeyChecking` keyword, `ssh_config` file, 353
`StrictModes` keyword, `sshd_config` file, 354
`su` command
  displaying access attempts on console, 71-72
  in role assumption, 209-211, 211-212
  monitoring use, 70-71
`su` file, monitoring `su` command, 70-71

`subject` audit token, format, 637-638
`Subsystem` keyword, `sshd_config` file, 354
success
    audit class prefix, 618
    turning off audit classes for, 619
sufficient control flag, PAM, 319
`sulog` file, 70-71
    monitoring contents of, 70
`SUPATH` in Solaris Secure Shell, 355
superuser
    compared to privilege model, 185-193
    compared to RBAC model, 176-178
    differences from privilege model, 187
    eliminating in RBAC, 183
    monitoring access attempts, 71-72
`suser` security policy, 234
`svcadm` command
    administering cryptographic framework, 264, 265
    enabling cryptographic framework, 287-288
    enabling keyserver daemon, 305
    refreshing cryptographic framework, 279-280
    restarting name service, 200
    restarting NFS server, 582
    restarting Solaris Secure Shell, 335
    restarting `syslog` daemon, 62, 574
`svcs` command
    listing cryptographic services, 287-288
    listing keyserver service, 305
swapping master and slave KDCs, 436-441
symbolic links, file permissions, 123
symbolic mode
    changing file permissions, 126, 134-135, 135
    description, 126
synchronizing clocks
    master KDC, 397, 404
    overview, 434-436
    slave KDC, 410, 452
`SYS` privileges, 187
`sysconf.rpt` file, 150, 154
`syslog.conf` file
    and auditing, 611
    `audit.notice` level, 574
    audit records, 549
    executable stack messages, 130

`syslog.conf` file *(Continued)*
    `kern.notice` level, 130
    `priv.debug` entry, 257
    saving failed login attempts, 62-63
`SYSLOG_FAILED_LOGINS`
    in Solaris Secure Shell, 355
    system variable, 62
`syslog` format, audit records, 611
`SyslogFacility` keyword, `sshd_config` file, 354
System Administrator (RBAC)
    assuming role, 211
    creating role, 200
    protecting hardware, 72
    recommended role, 176
    rights profile, 224
system calls
    `arg` audit token, 626
    `close`, 617
    `exec_args` audit token, 627-628
    `exec_env` audit token, 628
    `ioctl()`, 618
    ioctl to clean audio device, 95
    `return` audit token, 635
`system` file, `bsmconv` effect on, 611
system hardware, controlling access to, 72-74
system properties, privileges relating to, 187
system security
    ACL, 127-130
    dial-up logins and passwords, 43-44
    dial-up passwords
        disabling temporarily, 65
    displaying
        user's login status, 58-59, 59
        users with no passwords, 60
    firewall systems, 55-56
    hardware protection, 38, 72-74
    login access restrictions, 39
    machine access, 38
    overview, 38
    password encryption, 40
    passwords, 39
    privileges, 185-193
    protecting from risky programs, 143-145
    restricted shell, 47, 48

system security *(Continued)*
  restricting remote `root` access, 71-72
  role-based access control (RBAC), 46, 176-178
  `root` access restrictions, 51-52, 71-72
  saving failed login attempts, 61
  special logins, 42
  `su` command monitoring, 46, 70-71
  task map, 143
`system state` audit class, 618
System V IPC
  `ipc` audit class, 618
  `ipc` audit token, 631
  `ipc_perm` audit token, 631-632
  privileges, 187
system variables
  See also variables
  `CRYPT_DEFAULT`, 66
  `KEYBOARD_ABORT`, 73-74
  `noexec_user_stack`, 145
  `noexec_user_stack_log`, 145
  `rstchown`, 133
  `SYSLOG_FAILED_LOGINS`, 62
`system-wide administration` audit class, 618
systems, protecting from risky programs, 143-145

T
tables, `gsscred`, 543
`tail` command, example of use, 568
tape drives
  allocating, 86
  cleaning of data, 94-95
  device-clean scripts, 94
task maps
  administering cryptographic framework, 275-276
  administering policies (Kerberos), 493
  administering principals (Kerberos), 481
  administering Secure RPC, 304
  allocating devices, 85
  ASET, 164-168
  auditing, 569
  changing default algorithm for password encryption, 65
  configuring audit files, 570
  configuring auditing service, 579-580
  configuring device policy, 76
  configuring devices, 75
  configuring Kerberos NFS servers, 414
  configuring RBAC, 196-197
  configuring Solaris Secure Shell, 331
  controlling access to system hardware, 72
  cryptographic framework, 267
  device allocation, 79-80
  device policy, 76
  devices, 75
  enabling auditing service, 579-580
  Kerberos configuration, 387-388
  Kerberos maintenance, 388
  managing and using privileges, 239
  managing audit records, 594
  managing device allocation, 79-80
  managing device policy, 76
  managing RBAC, 213
  monitoring and restricting superuser, 69-70
  PAM, 314
  planning auditing, 559
  protecting against programs with security risk, 143
  protecting files, 131
  protecting files with ACLs, 137
  protecting files with cryptographic mechanisms, 268
  protecting files with UNIX permissions, 131
  protecting system hardware, 72
  running ASET, 164-168
  securing logins and passwords, 58
  securing systems, 57-58
  Solaris Secure Shell, 331
  system access, 57-58
  Using BART task map, 100-101
  using device allocation, 85
  using RBAC, 195
  using roles, 209
  using Solaris Secure Shell, 335-336
  using the cryptographic framework, 267
  Using the Key Management Framework (Task Map), 291
`TASKS` variable (ASET), 156, 162

`taskstat` command (ASET), 149, 152
TCP
  addresses, 632
  Solaris Secure Shell and, 335, 349
`telnet` command
  Kerberos and, 522-525, 531
`telnetd` daemon, Kerberos and, 532
terminal ID, audit, 620
terminating, signal received during auditing
  shutdown, 615
terminology
  authentication-specific, 533-534
  Kerberos, 532-537
  Kerberos-specific, 532
test manifests, 99
`text` audit token, format, 638
TGS, getting credential for, 538
TGT, in Kerberos, 365-366
third-party password algorithms, adding, 68-69
ticket file, *See* credential cache
ticket-granting service, *See* TGS
ticket-granting ticket, *See* TGT
tickets
  creating, 513-514
  creating with `kinit`, 514
  definition, 364
  definition in Kerberos, 533
  destroying, 516
  `-F` option or `-f` option, 523
  file
    See credential cache
  forwardable, 364, 514, 525, 534
  initial, 534
  invalid, 534
  `-k` option, 524
  `klist` command, 515-516
  lifetime, 535-536
  maximum renewable lifetime, 536
  obtaining, 513-514
  or credentials, 365
  postdatable, 534
  postdated, 364
  proxiable, 535
  proxy, 535
tickets *(Continued)*
  renewable, 535
  requesting for specific realm, 524
  types of, 534-537
  viewing, 515-516
  warning about expiration, 429
`TIMEOUT` in Solaris Secure Shell, 355
timestamps
  ASET reports, 153
  audit files, 622
`/tmp/krb5cc_`*uid* file, description, 530
`/tmp/ovsec_adm.`*xxxxx* file, description, 530
tmpfile string, `audit_warn` script, 615
TMPFS file system, security, 125
token, definition in cryptographic framework, 264
`trail` audit policy
  and `trailer` token, 566
  description, 566
`trailer` audit token
  format, 639
  order in audit record, 639
  `praudit` display, 639
transparency, definition in Kerberos, 364
Trojan horse, 47
troubleshooting
  allocating a device, 86
  ASET errors, 168
  `encrypt` command, 275
  Kerberos, 471
  lack of privilege, 242-243
  `list_devices` command, 82
  mounting a device, 88
  privilege requirements, 242-243
  rights profiles, 217
  role capabilities, 202
`truss` command, for privilege debugging, 242
trusted hosts, 55
tune files (ASET)
  description, 155
  examples, 163, 164
  modifying, 158
  rules, 163
`tune.rpt` file, 149, 154
types of tickets, 534-537

`TZ` in Solaris Secure Shell, 355

U

`-U` option
  `allocate` command, 91
  `list_devices` command, 91
`uauth` audit token, 556, 639
UDP
  addresses, 632
  port forwarding and, 335
  Solaris Secure Shell and, 335
  using for remote audit logs, 553
`uid_aliases` file (ASET), 155, 158
`UID_ALIASES` variable (ASET), 155, 158, 162
`umask` value
  and file creation, 125
  typical settings, 125
`umount` command, with security attributes, 81
uninstalling, cryptographic providers, 282
UNIX file permissions, *See* files, permissions
unmounting, allocated devices, 89
`update_drv` command
  description, 90
  using, 77-78
updating, auditing service, 589-590
URL for online help, Graphical Kerberos Tool, 386
`use_authid` option, SASL and, 325
`UseLogin` keyword, `sshd_config` file, 354
user accounts
  See also users
  ASET check, 150
  displaying login status, 58-59, 59
User Accounts tool, description, 218-219
user ACL entries
  default entries for directories, 129
  description, 128-129
  setting, 138-140
`user administration` audit class, 618
`user_attr` database
  `defaultpriv` keyword, 256
  description, 229, 231
  `limitpriv` keyword, 256
  RBAC relationships, 229-230
user audit fields, `audit_user` database, 613-614
user classes of files, 122
user database (RBAC), *See* `user_attr` database
user ID
  audit ID and, 547-548, 620
  in NFS services, 417
`User` keyword, `ssh_config` file, 354
user principal, description, 369
user procedures
  allocating devices, 85
  assuming a role, 209
  `chkey` command, 308
  computing digest of a file, 270-271
  computing MAC of a file, 271-273
  creating self-signed certificate, 291-292
  decrypting files, 273-275
  encrypting files, 268
  encrypting NIS user's private key, 308
  exporting certificates, 294-295
  generating a symmetric key
    using the `dd` command, 268-270
  generating passphrase for keystore, 295-296
  importing certificates, 293-294
  protecting files, 131-137
  using ACLs, 137-143
  using an assigned role, 209
  using `pktool` command, 291
  using Solaris Secure Shell, 335-336
user scripts, configuring for `ssh-agent` daemon in CDE, 342
`useradd` command
  adding local user, 207
  description, 237
`userdel` command, description, 237
`UserKnownHostsFile` keyword, `ssh_config` file, 354
`UserKnownHostsFile2` keyword, *See* `UserKnownHostsFile` keyword
`usermod` command
  changing user's RBAC properties, 218
  description, 237
  using to assign role, 205-206
users
  adding local user, 207
  allocating devices, 85-86

users *(Continued)*
  assigning allocate authorization to, 81
  assigning privileges to, 244-245
  assigning RBAC defaults, 235-236
  basic privilege set, 190
  changing properties from command line, 219
  computing digest of files, 270-271
  computing MAC of files, 271-273
  creating local user, 207
  deallocating devices, 88-89
  determining directly assigned privileges, 248-249
  determining own privileged commands, 249-251
  disabling login, 60-61
  displaying login status, 58-59
  encrypting files, 273-275
  having no passwords, 59-60
  initial inheritable privileges, 190
  modifying audit preselection mask of, 575-577
  modifying properties (RBAC), 218-219
  mounting allocated devices, 86-88
  restricting basic privileges, 246
  unmounting allocated devices, 89
using
  ACLs, 138-140
  `allocate` command, 85-86
  ASET, 164-168
  BART, 101
  `cryptoadm` command, 276
  cryptographic framework task map, 267
  `dd` command, 268-270
  `deallocate` command, 88-89
  device allocation, 85
  `digest` command, 270-271
  `encrypt` command, 273-275
  file permissions, 131
  `mac` command, 271-273
  `mount` command, 87
  new password algorithm, 66
  `ppriv` command, 240
  privileges, 248
  privileges task map, 247
  RBAC task map, 195
  roles, 209
  roles task map, 209

using *(Continued)*
  `smrole` command, 245
  Solaris Secure Shell task map, 335-336
  `ssh-add` command, 340-341
  `ssh-agent` daemon, 340-341
  `truss` command, 242
  `umount` command, 89
  `usermod` command, 244
Using the Key Management Framework (Task Map), 291
`/usr/aset/asetenv` file, 156
`/usr/aset` directory, 148
`/usr/aset/masters/tune` files
  description, 155
  modifying, 158
  rules, 163
`/usr/aset/masters/uid_aliases` file, 155
`/usr/aset/reports` directory, structure, 153
`/usr/aset/reports` directory structure, 152
`/usr/aset/reports/latest` directory, 153
`/usr/bin/ftp` command, Kerberos and, 531
`/usr/bin/kdestroy` command, Kerberos and, 531
`/usr/bin/kinit` command, Kerberos and, 531
`/usr/bin/klist` command, Kerberos and, 531
`/usr/bin/kpasswd` command, Kerberos and, 531
`/usr/bin/ktutil` command, Kerberos and, 531
`/usr/bin/rcp` command, Kerberos and, 531
`/usr/bin/rdist` command, Kerberos and, 531
`/usr/bin/rlogin` command, Kerberos and, 531
`/usr/bin/rsh` command, Kerberos and, 531
`/usr/bin/telnet` command, Kerberos and, 531
`/usr/lib/kprop` command, description, 531
`/usr/lib/krb5/kadmind` daemon, Kerberos and, 532
`/usr/lib/krb5/kpropd` daemon, Kerberos and, 532
`/usr/lib/krb5/krb5kdc` daemon, Kerberos and, 532
`/usr/lib/krb5/ktkt_warnd` daemon, Kerberos and, 532
`/usr/lib/libsasl.so` library, overview, 323
`/usr/sbin/gkadmin` command, description, 531
`/usr/sbin/gsscred` command, description, 531
`/usr/sbin/in.ftpd` daemon, Kerberos and, 532
`/usr/sbin/in.rlogind` daemon, Kerberos and, 532
`/usr/sbin/in.rshd` daemon, Kerberos and, 532
`/usr/sbin/in.telnetd` daemon, Kerberos and, 532

`/usr/sbin/kadmin` command, description, 531
`/usr/sbin/kadmin.local` command, description, 531
`/usr/sbin/kclient` command, description, 531
`/usr/sbin/kdb5_ldap_util` command, description, 531
`/usr/sbin/kdb5_util` command, description, 531
`/usr/sbin/kgcmgr` command, description, 531
`/usr/sbin/kproplog` command, description, 531
`/usr/share/lib/xml` directory, 609
`usrgrp.rpt` file
  description, 150, 154
  example, 154
`uucico` command, login program, 64

V

v1 protocol, Solaris Secure Shell, 327
`-v` option
  `digest` command, 270
  `mac` command, 272
  `ppriv` command, 240
v2 protocol, Solaris Secure Shell, 327
`/var/adm/auditlog` file, text audit records, 574
`/var/adm/loginlog` file, saving failed login attempts, 61
`/var/adm/messages` file, executable stack messages, 130
`/var/adm/sulog` file, monitoring contents of, 70
`/var/krb5/.k5.`*REALM* file, description, 530
`/var/krb5/kadmin.log` file, description, 530
`/var/krb5/kdc.log` file, description, 530
`/var/krb5/principal` file, description, 530
`/var/krb5/principal.kadm5` file, description, 530
`/var/krb5/principal.kadm5.lock` file, description, 530
`/var/krb5/principal.ok` file, description, 530
`/var/krb5/principal.ulog` file, description, 530
`/var/krb5/slave_datatrans` file, description, 530
`/var/krb5/slave_datatrans_`*slave* file, description, 530
`/var/log/authlog` file, failed logins, 62-63
`/var/run/sshd.pid` file, description, 357
variables
  adding to audit record, 565, 628

variables *(Continued)*
  ASET environment variables
    `ASETDIR`, 160
    `ASETSECLEVEL`, 161
    `CKLISTPATH_level`, 155, 157, 162
    `PERIODIC_SCHEDULE`, 157, 161
    summary, 160
    `TASKS`, 156, 162
    `UID_ALIASES`, 155, 158, 162
    `YPCHECK`, 158, 162
  auditing those associated with a command, 627
  for proxy servers and ports, 345
  `KEYBOARD_ABORT`, 73-74
  `login` and Solaris Secure Shell, 354-355
  `noexec_user_stack`, 130
  `noexec_user_stack_log`, 130
  `rstchown`, 133
  setting in Solaris Secure Shell, 355
verifiers
  description, 302
  returned to NFS client, 303
  window, 302
`VerifyReverseMapping` keyword, `ssh_config` file, 354
viewing
  ACL entries, 142-143
  audit record formats, 594-596
  available cryptographic mechanisms, 278, 282
  binary audit files, 600-602
  contents of rights profiles, 227
  cryptographic mechanisms
    available, 278, 282
    existing, 277, 278, 282
  device allocation information, 82
  device policy, 76-77
  digest of a file, 271
  directly assigned privileges, 248-249
  existing cryptographic mechanisms, 278, 282
  file permissions, 131-132
  keylist buffer with `list` command, 510, 511
  list of policies, 493-495
  list of principals, 482-484
  MAC of a file, 272-273
  policy attributes, 495-497
  principal's attributes, 484-486

viewing *(Continued)*
  privileges in a shell, 241, 249
  privileges on a process, 240
  tickets, 515-516
  user's login status, 58-59
  users with no passwords, 59-60
  XML audit records, 601, 609
viruses
  denial of service attack, 49
  Trojan horse, 47
`vnode` audit token, format, 626-627
`vold` daemon, turned off by device allocation, 81

W

`warn.conf` file, description, 530
warning about ticket expiration, 429
wildcard characters
  for hosts in Solaris Secure Shell, 345
  in ASET files, 161
  in ASET tune files, 163
  in RBAC authorizations, 228
window verifier, 302
write permissions, symbolic mode, 126

X

`-X` option, Kerberized commands, 524
X Window system, and SEAM Administration Tool, 476-477
X11 forwarding
  configuring in `ssh_config` file, 351
  in Solaris Secure Shell, 349
`X11DisplayOffset` keyword, `sshd_config` file, 354
`X11Forwarding` keyword, `sshd_config` file, 354
`X11UseLocalHost` keyword, `sshd_config` file, 354
`-x` option
  Kerberized commands, 524
  `praudit` command, 609
`xauth` command, X11 forwarding, 354
`XAuthLocation` keyword, Solaris Secure Shell port forwarding, 354
XML format, audit records, 602
XML option, `praudit` command, 609
Xylogics tape drive device-clean script, 94

Y

`YPCHECK` variable (ASET), 158, 162

Z

`zone.max-locked-memory` resource control, 175, 188-189
`zonename` audit policy
  description, 566
  using, 561, 616-617
`zonename` audit token, 557, 639-640
zones
  auditing and, 555, 616-617
  configuring auditing in global zone, 585
  cryptographic framework and, 266
  cryptographic services and, 287-288
  devices and, 44
  `perzone` audit policy, 555, 561, 616-617
  planning auditing in, 560-561
  `zonename` audit policy, 561, 616-617