

Hacking .NET Applications: The Black Arts



Jon McCoy

www.DigitalBodyGuard.com



BACKGROUND

- ◆ Why .NET
- ◆ Countermeasures
- ◆ Skill Level Needed
- ◆ Will this work on every .NET application



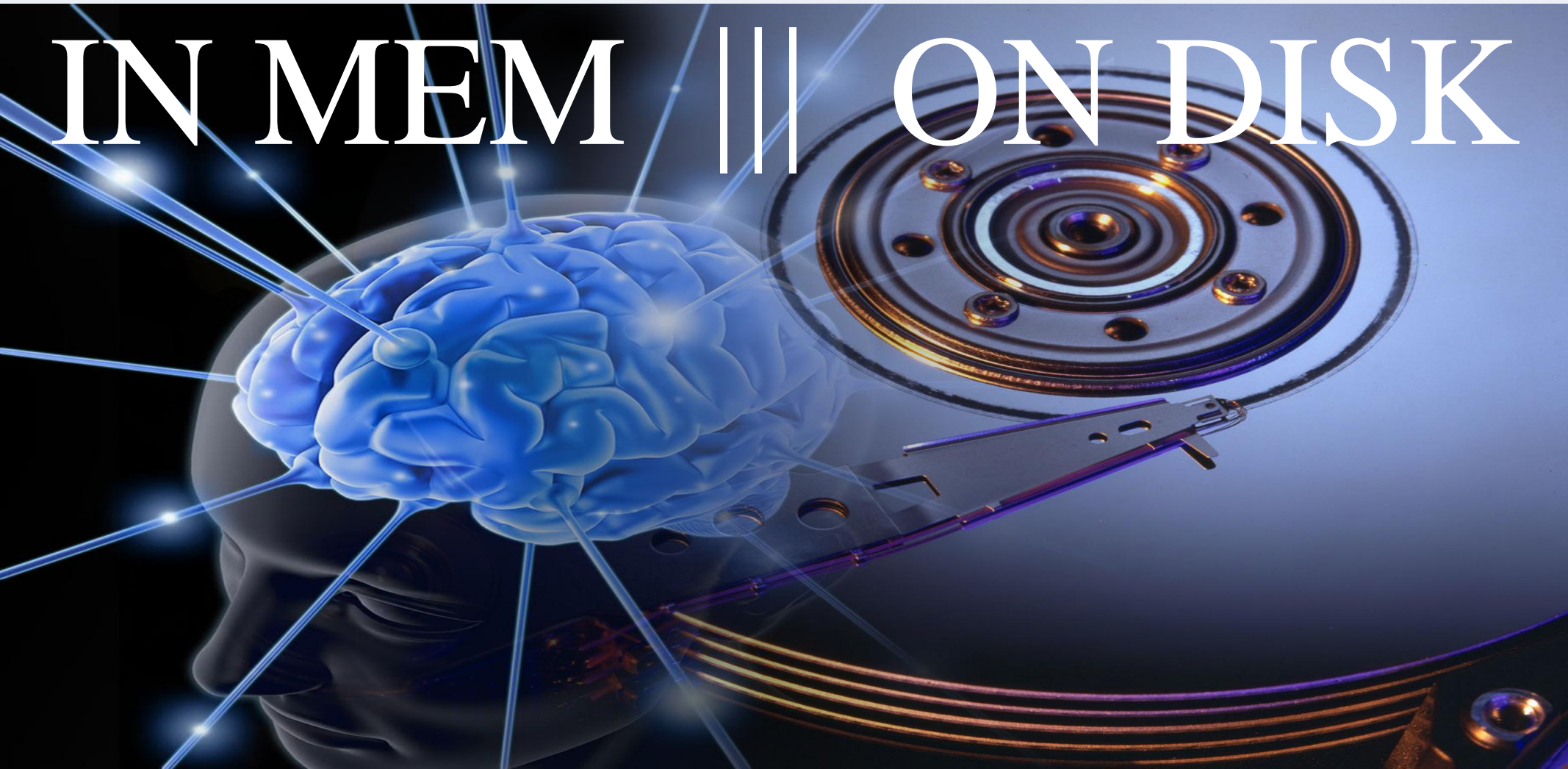
THIS WILL COVER

- ◆ How-To Attack .NET Applications
- ◆ Tools and Methodology of Attacking
- ◆ Overcome “secure” .NET Applications
- ◆ Building KeyGen/Crack/Hacks/Malware
- ◆ Reverse Engenering for Protection



Attacking/Cracking

IN MEM ||| ON DISK



ATTACK OVERVIEW



Attack on Disk

- Access Logic
- Infect Logic
- Hook Logic
- Decompile
- Recompile
- Debug



Attack in Memory/Runtime

- Inject Target App
- Navigate Structure
- Edit/Control Structure

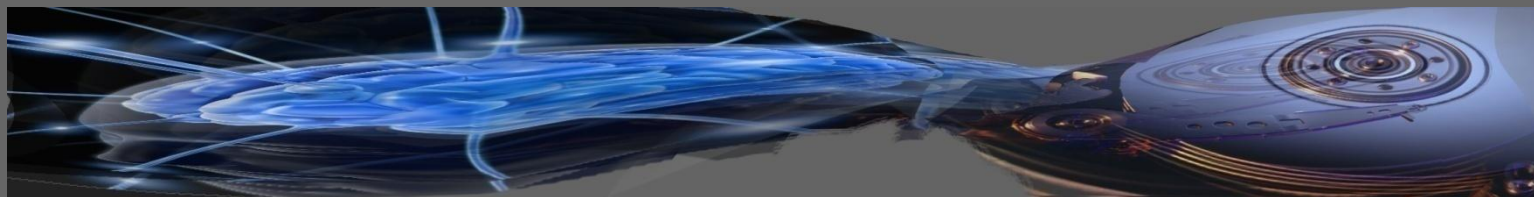


Attack The Source

In Memory

OR

On Disk



Do your Reconnaissance

Find the weak spot

Subvert the Logic/State

Control what you need

ATTACKING ON DISK



101 - DECOMPILERS





DEMO

GrayWolf – IL_Spy – Reflector



101 - ATTACK ON DISK


Connect/Open - Access Code

-  Decompile - Get code/tech
-  Infect - Change the target's code
-  Exploit - Take advantage
-  Remold Application - WIN



THE WEAK SPOTS

 Flip The Check

 Set Value is “True”

 Cut The Logic

 Return True

 Access Value

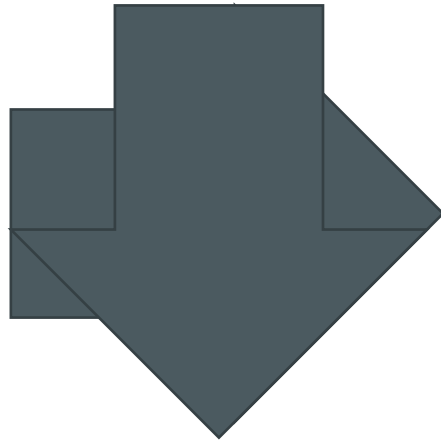




SET FLV VALUE TO THE "CRUE"

~~bool Registered = false;~~

~~If(a == b)~~





RETURN TRUE

```
bool IsRegistered()
```

```
{
```

```
    Return "TRUE";
```

```
}
```





RETURN TRUE

```
bool IsValidKey(string x)
{
    Return "TRUE";
}
```





CUT THE LOGIC

```
string sqlClean(string x)
{
    Return x;
}
```





ACCESS VALUE

```
bool ValidPassword(int x)
{
    ShowKey(Pass);
    Return (x==Pass);
}
```



ATTACK SECURITY



Microsoft Media Center



CRACK

```
public static bool CheckPin(string pin)  
{
```

```
    ParentalControl.Settings.PIN = null;  
    ParentalControl.Settings.Load();  
    string text = ParentalControl.Settings.PIN;  
    if (text == null)  
    {  
        return 1;  
    }
```

```
    if (text.Length > 0)
```

```
    {  
        if (text.get_Chars(0) == 58)  
        {  
            goto Block_6;  
        }
```

```
    }  
    ParentalControlPin.StoreNewPin(text);  
    return text == pin;
```

```
Block_6:
```

```
    return text == ParentalControlPin.HashForPin(pin);
```

```
}
```



PASSWORD



CRACK

```
public static bool CheckPin(string pin)  
{
```



Return True;

PASSWORD

```
{  
    return 1;  
}  
if (text.Length > 0)  
{  
    if (text.get_Chars(0) == 58)  
    {  
        goto Block_6;  
    }  
}  
ParentalControlPin.StoreNewPin(text);  
return text == pin;  
Block_6:  
return text == ParentalControlPin.HashForPin(pin);  
}
```



CRACK



DEMO



REGISTRATION CHECK

■ KeyGens

■ Cracks



CRACK THE KEY



Public/Private == Change Key



3/B==Name*ID*7 == ASK what is /B?



Call Server == Hack the Call



Demo = True; == Set Value



Complex Math == Complex Math

1% of the time the KeyGen is given





PUBLIC/PRIVATE KEY

If you can beat them

Why join them

Key = “F5PA11JS32DA”



Key = “123456ABCDE”





SERVER CALL

1. Fake the Call “Send”
SystemID = 123456789
2. Fake the Request
3. Fake the Reply Reg Code = f3V541
4. Win

Registered = True

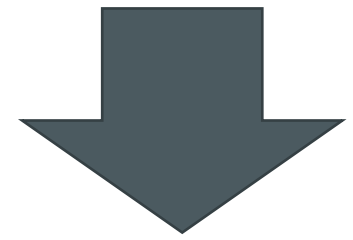




REG CODE REPLAY

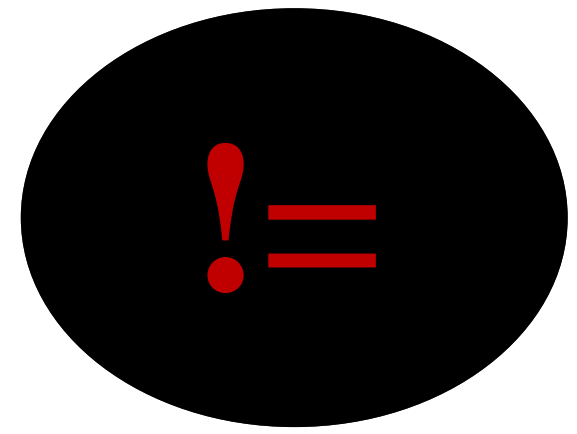
Name:   

JON DOE



Code: 

98qf3uy


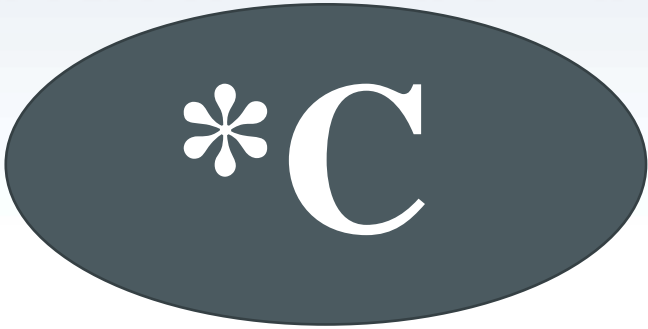



FAIL





REG CODE REPLAY

Name:   



Code:  

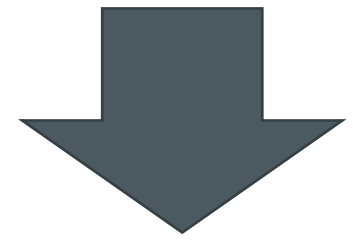




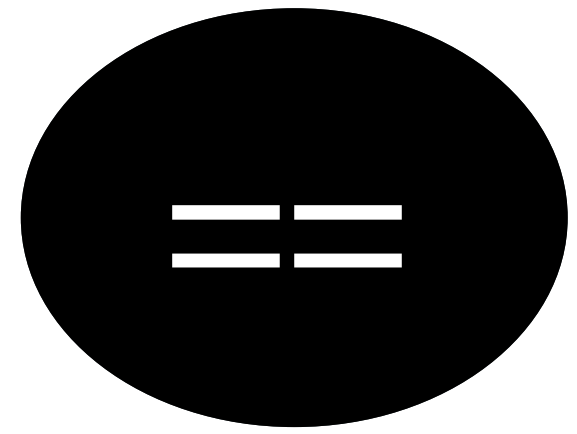
REG CODE REPLAY

Name:   

JON DOE



Code: 



5G9P3

WIN





COMPLEX MATH

1. Chop up the Math
2. Attack the Weak
3. ????????????
4. Profit



DEMO

CRACK A KEY



IL – Intermediate Language

Code of the Matrix |||| NEW ASM

1	ldarg.0		78	stloc.2		174	ldobj	System.IntPtr	235	stloc.3	
2	movsb	System.Void System.Random.Next(System.IntPtr)	79	ldloc.2		175	ldloc.0		236	stloc.3	
7	stloc.0		80	ldc.i4.0		177	ldc.i4.0	10000	237	brtrue	IL_01e8 ret
8	ldc.i4.4		81	ldstems	System.IntPtr	182	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	242	ldarg.0	
9	movsr	System.IntPtr	85	dup		187	add		248	stloc.1	
14	stloc.1		87	ldobj	System.IntPtr	188	stobj	System.IntPtr	244	ldc.i4.2	
15	ldloc.1		92	ldloc.1		199	ldarg.0		245	ldstems	
16	ldc.i4.0		98	ldc.i4.0	10	194	ldloc.1		246	rem	
17	ldloc.0		99	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	195	ldc.i4.0		247	stloc.2	
18	ldc.i4.0	10	100	add		196	ldstems		248	ldc.i4.1	
20	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	101	stobj	System.IntPtr	197	rem		249	ldstems	
25	ststems		105	ldloc.2		198	ldloc.2		250	add	
26	ldloc.1		107	ldc.i4.1		199	ldc.i4.3		251	ldc.i4.0	01F
27	ldc.i4.1		108	ldstems	System.IntPtr	200	ldstems		255	caq	
28	ldloc.0		116	dup		201	add		258	ldc.i4.0	
29	ldc.i4.0	100	114	ldobj	System.IntPtr	202	ldc.i4.0	0000	259	caq	
31	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	115	ldloc.0		207	caq		261	stloc.3	
35	ststems		120	ldc.i4.0	100	208	ldc.i4.0		262	stloc.3	
37	ldloc.1		122	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	210	caq		268	brtrue	IL_01e8 ret
38	ldc.i4.2		127	add		212	stloc.3		268	ldarg.0	
39	ldloc.0		128	stobj	System.IntPtr	213	ldloc.3		269	stloc.1	
40	ldc.i4.0	1000	133	ldloc.2		214	brtrue	IL_01e8 ret	270	ldc.i4.3	
45	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	134	ldc.i4.2		215	ldarg.0		271	ldstems	
50	ststems		135	ldstems	System.IntPtr	220	ldloc.1		272	rem	
51	ldloc.1		140	dup		221	ldc.i4.1		273	stloc.2	
52	ldc.i4.3		141	ldobj	System.IntPtr	222	ldstems		274	ldc.i4.0	
53	ldloc.0		145	ldloc.0		223	rem		275	ldstems	
54	ldc.i4.0	10000	147	ldc.i4.0	1000	224	ldloc.2		276	add	
55	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	152	callvirt	System.IntPtr System.Random.Next(System.IntPtr)	225	ldc.i4.2		277	ldc.i4.0	010
59	ststems		157	add		226	ldstems		282	caq	
65	ldarg.1		158	stobj	System.IntPtr	227	add		284	ldc.i4.0	
68	movsb	System.Void System.Random.Next(System.IntPtr)	169	ldloc.2		228	ldc.i4.0	00	285	caq	
71	stloc.0		184	ldc.i4.3		230	caq		287	stloc.3	
72	ldc.i4.4		185	ldstems	System.IntPtr	232	ldc.i4.0		288	stloc.3	



IT CAN'T BE THAT EZ

NO



PROTECTION ON DISK

● Protection - Security by *Ob\$cur17y*

- Code Obfuscation
- Logic Obfuscation
- Shells / Packers / Encrypted(code)
- Unmanaged calls.....

Try to SHUTDOWN
Decompilation



PROTECTION ON DISK

Obfuscated

```
public static bool XXXX()
{
    try
    {
        bool flag = ( & 4) == 4;
    }
    catch (Exception exception)
    {
        XXXX.XX(arg_0F_0, box(Application));
        throw;
    }
    return flag;
}
```



PROTECTION ON DISK

● Protection – Security by security

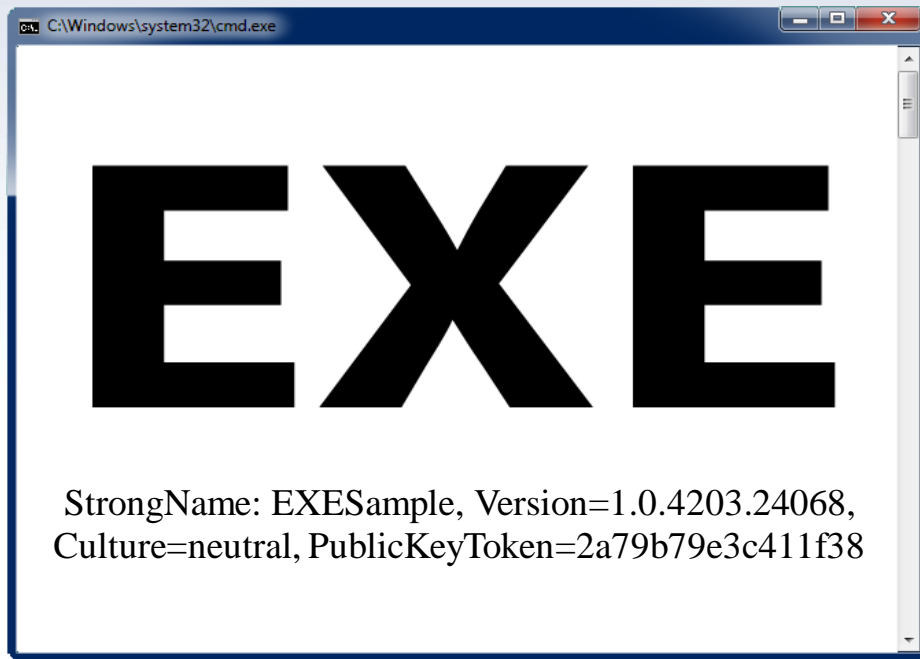
- Signed code (1024 bit CRYPTO)
- Verify the creator
- Strong Names
- ACLs..... M\$ stuff

Try to SHUTDOWN

Tampering



STRONG NAME



- Simple Name
- Version
- Culture
- Public Key Token

Example Strong Name:

EXESample, Version=1.0.4203.24068,
Culture=neutral, PublicKeyToken=2a79b79e3c411f38

Most of the time PublicKeyToken=null



PRIVATE KEY SIGNING



Signed code is based on

- Private Key - 1024 bit
- Signed Hash of Code
-

Identify and Verify the Author



UNPROTECTED/PROTECTED



IT CAN'T BE THAT EZ



NO

YES



ATTACK VECTOR

PRIVET KEY SIGNING



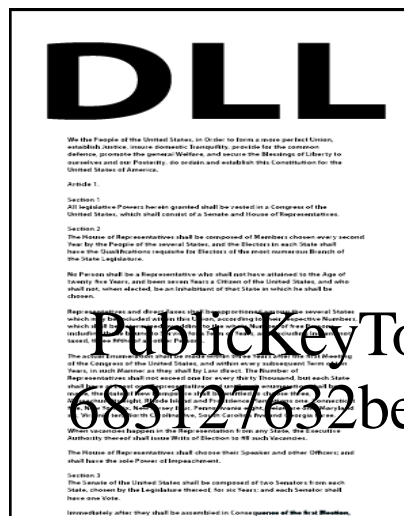
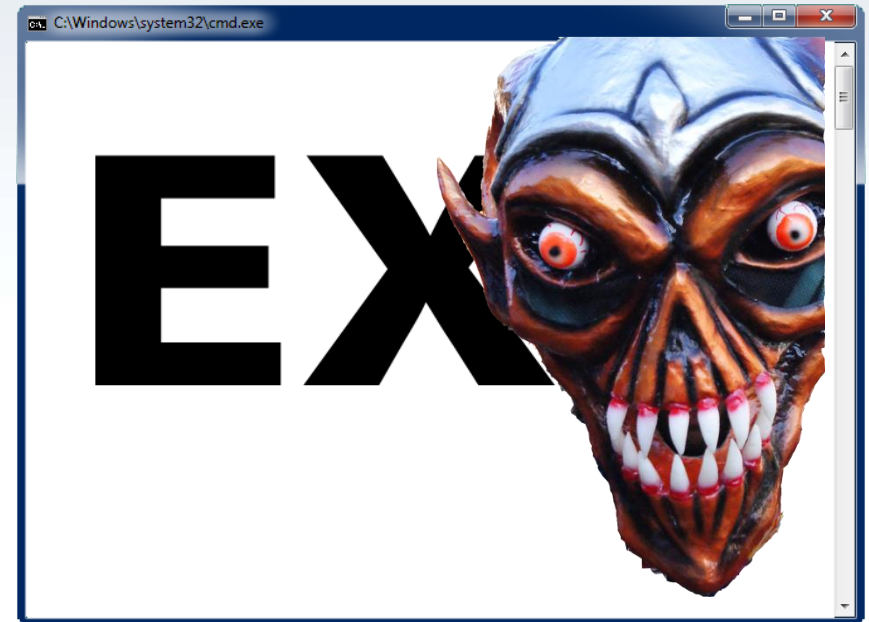
Signed code is based on

- Private Key - 1024 bit
- Signed Hash of Code
-

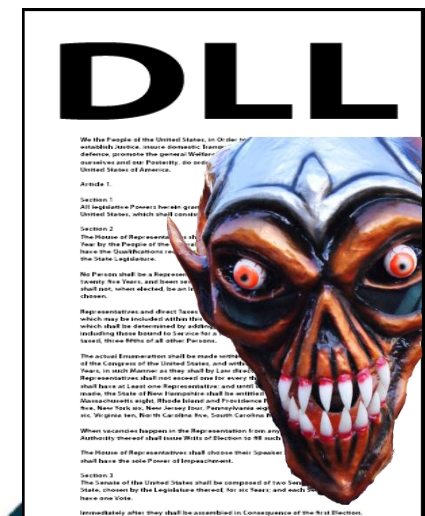
SIGNED CODE CHECKING IS
OFF BY DEFAULT



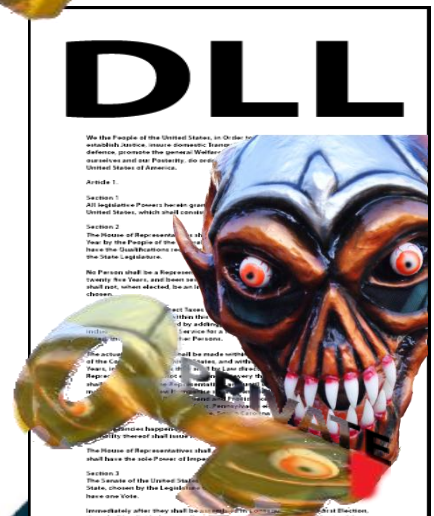
STRONG NAME HACKING



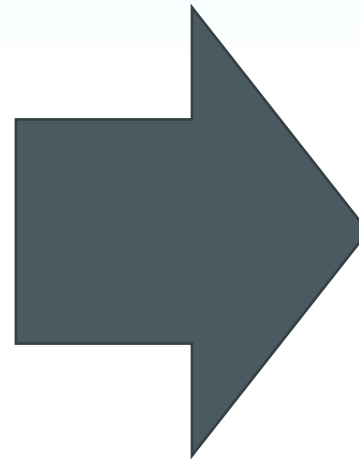
PublicKeyToken=
683127632be2c302



STRONG NAME HACKING



FAKE SIGNED DLL



FAKE SIGNED DLL

Turn Key Checking ON

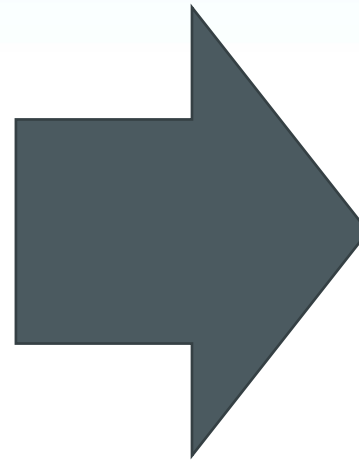
[HKEY_LOCAL_MACHINE

\SOFTWARE\Microsoft\.NETFramework]

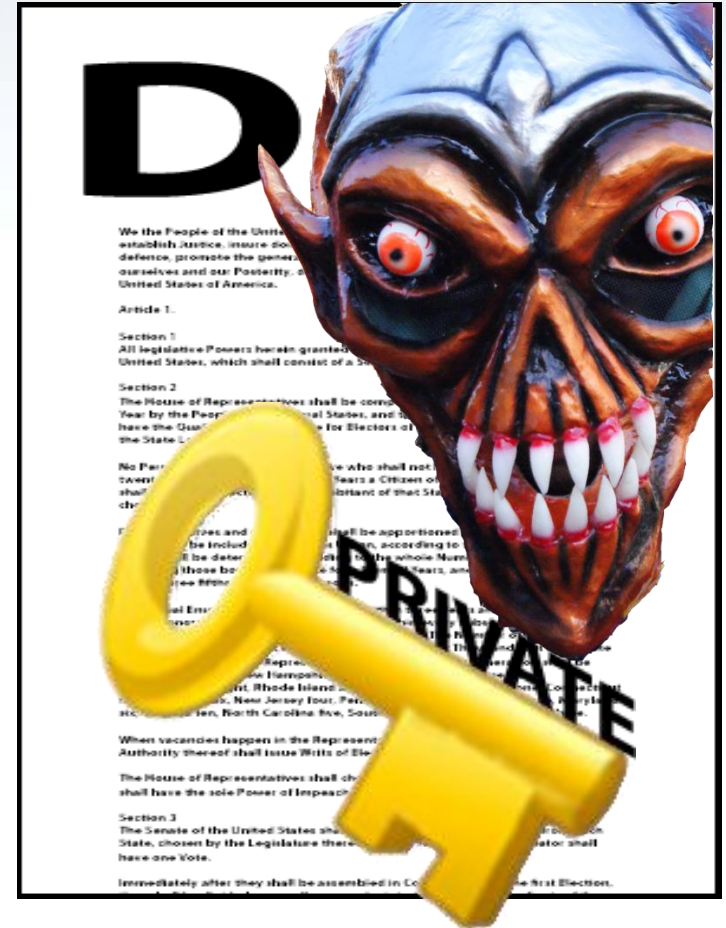
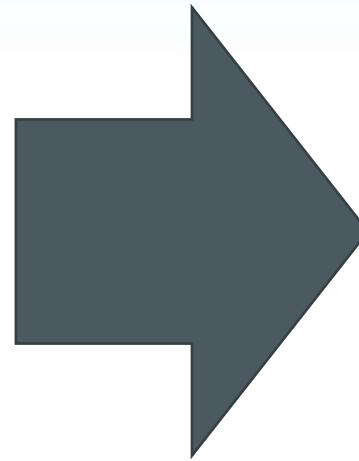
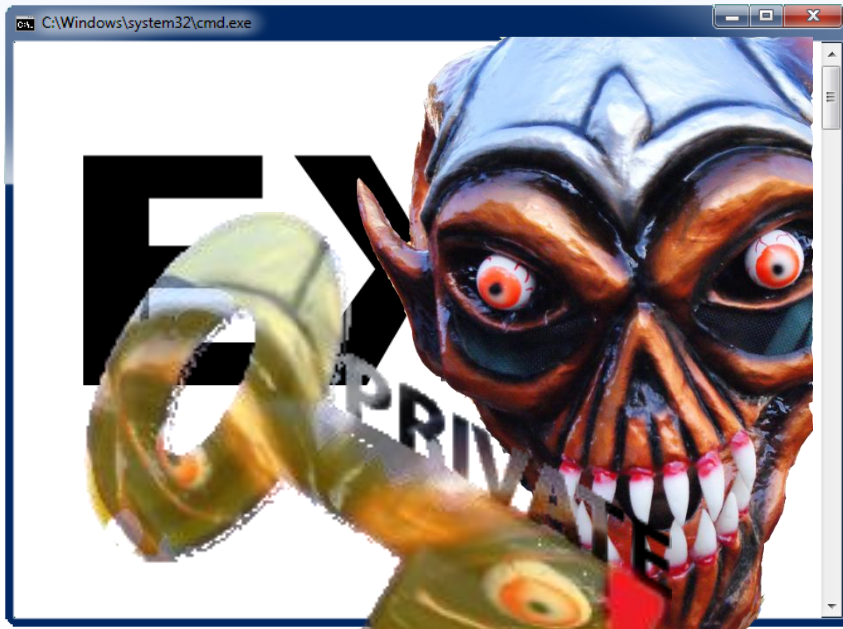
"AllowStrongNameBypass"=dword:00000000



FAKE SIGNED DLL



FAKE SIGNED EXE



FAKE SIGN DLL/EXE

Y U NO Check



Y ASK 4 PASSWORD



GLOBAL ASSEMBLY CASH --THE GAC--

What is the GAC?

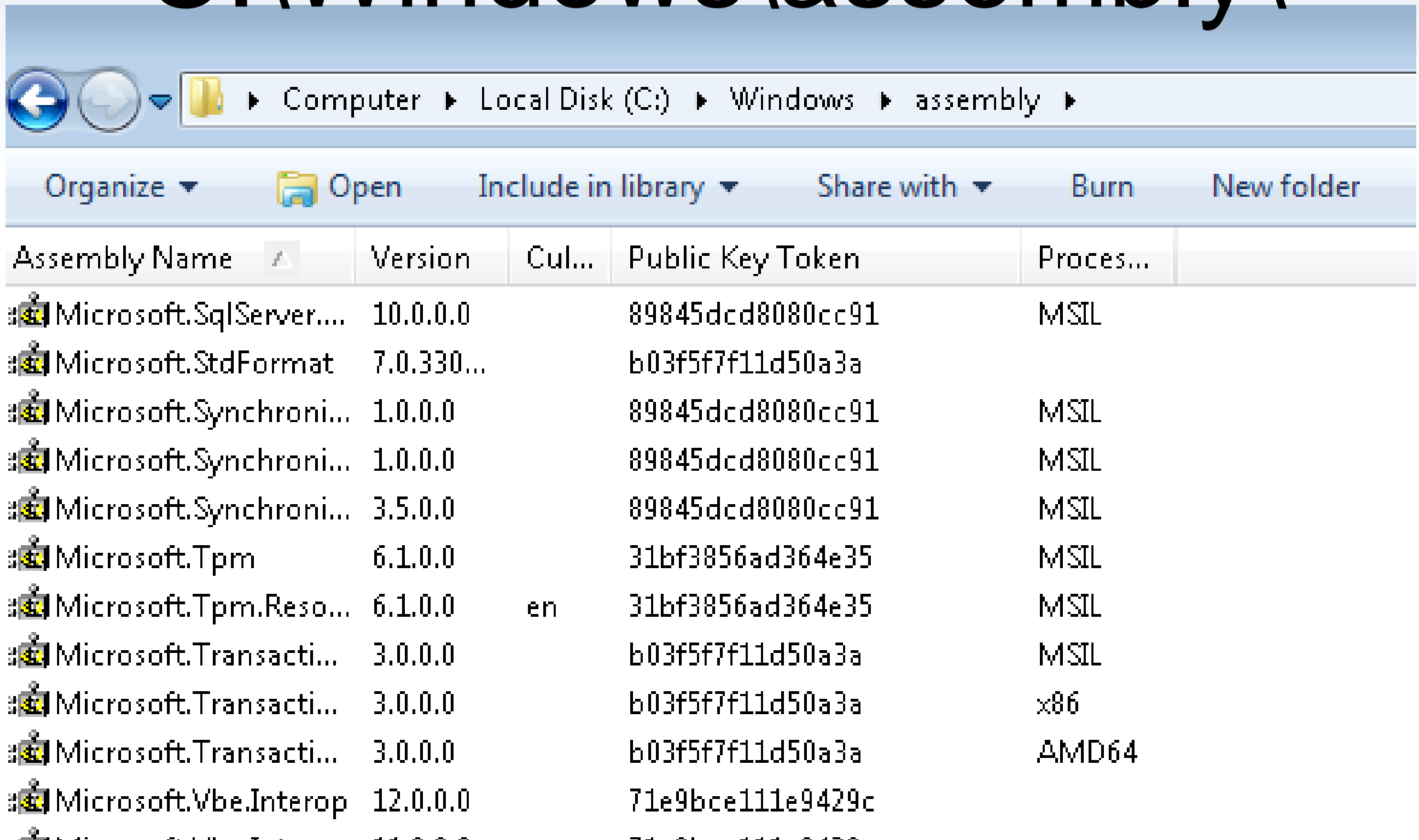
How to access the GAC?

Attacking from the GAC?



GLOBAL ASSEMBLY CASH

C:\Windows\assembly\

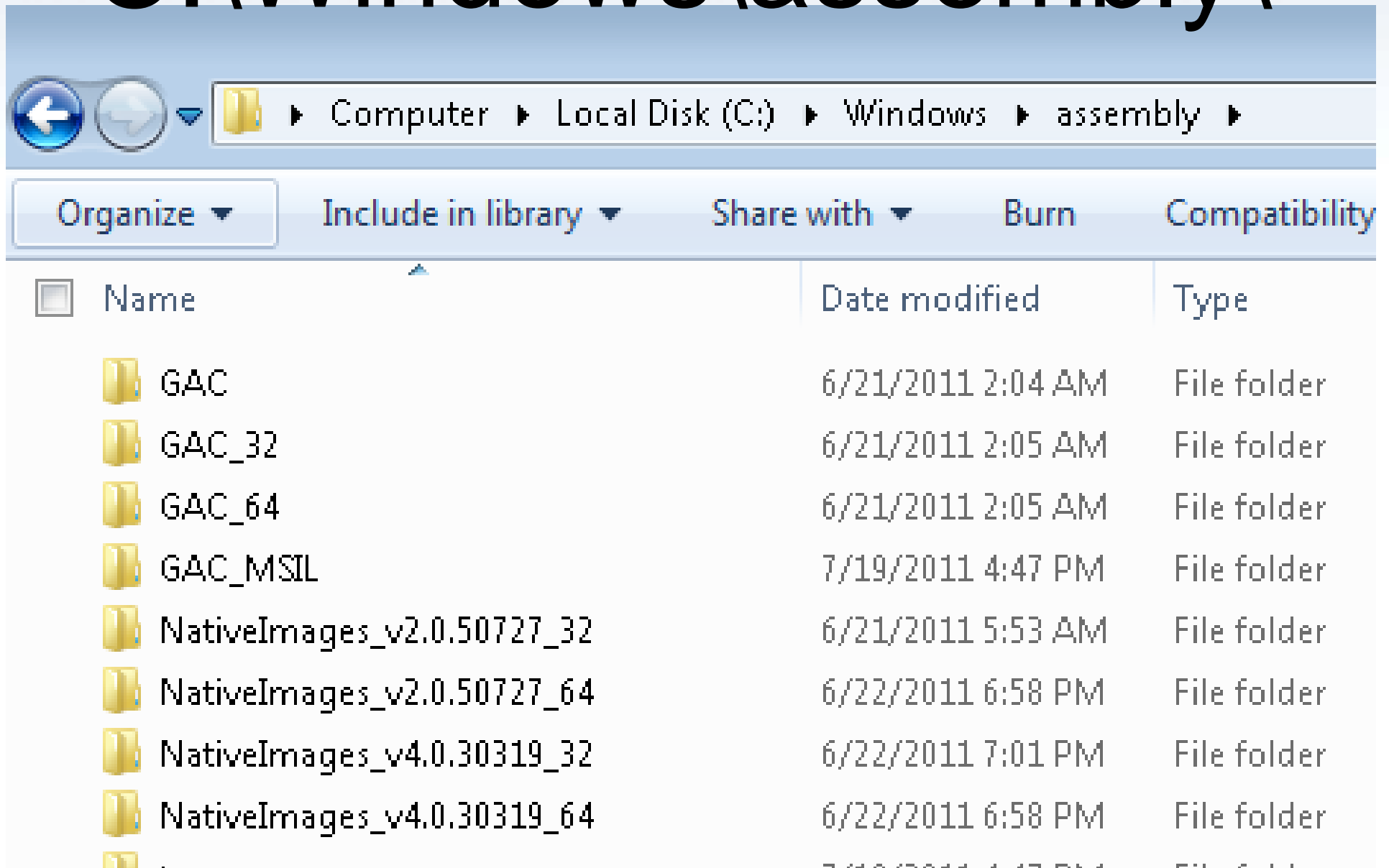


The screenshot shows a Windows Explorer window with the address bar set to "Computer > Local Disk (C:) > Windows > assembly". The window title is "GLOBAL ASSEMBLY CASH". The main area displays a table of assembly files.

Assembly Name	Version	Cul...	Public Key Token	Proces...
Microsoft.SqlServer...	10.0.0.0		89845dcd8080cc91	MSIL
Microsoft.StdFormat	7.0.330...		b03f5f7f11d50a3a	
Microsoft.Synchroni...	1.0.0.0		89845dcd8080cc91	MSIL
Microsoft.Synchroni...	1.0.0.0		89845dcd8080cc91	MSIL
Microsoft.Synchroni...	3.5.0.0		89845dcd8080cc91	MSIL
Microsoft.Tpm	6.1.0.0		31bf3856ad364e35	MSIL
Microsoft.Tpm.Reso...	6.1.0.0	en	31bf3856ad364e35	MSIL
Microsoft.Transacti...	3.0.0.0		b03f5f7f11d50a3a	MSIL
Microsoft.Transacti...	3.0.0.0		b03f5f7f11d50a3a	x86
Microsoft.Transacti...	3.0.0.0		b03f5f7f11d50a3a	AMD64
Microsoft.Vbe.Interop	12.0.0.0		71e9bce111e9429c	

GLOBAL ASSEMBLY CASH

C:\Windows\assembly\



The screenshot shows a Windows Explorer window with the address bar set to 'Computer > Local Disk (C:) > Windows > assembly'. The ribbon includes 'Organize', 'Include in library', 'Share with', 'Burn', and 'Compatibility'. The main pane displays a list of folders in the GAC, including 'GAC', 'GAC_32', 'GAC_64', 'GAC_MSIL', and several 'NativeImages' folders for different .NET Framework versions and architectures.

<input type="checkbox"/>	Name	Date modified	Type
<input type="checkbox"/>	GAC	6/21/2011 2:04 AM	File folder
<input type="checkbox"/>	GAC_32	6/21/2011 2:05 AM	File folder
<input type="checkbox"/>	GAC_64	6/21/2011 2:05 AM	File folder
<input type="checkbox"/>	GAC_MSIL	7/19/2011 4:47 PM	File folder
<input type="checkbox"/>	NativeImages_v2.0.50727_32	6/21/2011 5:53 AM	File folder
<input type="checkbox"/>	NativeImages_v2.0.50727_64	6/22/2011 6:58 PM	File folder
<input type="checkbox"/>	NativeImages_v4.0.30319_32	6/22/2011 7:01 PM	File folder
<input type="checkbox"/>	NativeImages_v4.0.30319_64	6/22/2011 6:58 PM	File folder
<input type="checkbox"/>		7/19/2011 4:47 PM	File folder

GLOBAL ASSEMBLY CASH GAC

- \GAC – Installed/Sandbox
- \GAC_32 – 32bit-(x86)
- \GAC_64 – 64bit-(x64)
- \GAC_MSIL – MSIL(ANY)



NATIVE IMAGE(NI)

VER 1.1 - is dead ☺

GAC

VER 3.0 - is dead ☺

VER 2.0 & 3.5

 \NativeImages_v2.0.50727_32

 \NativeImages_v2.0.50727_64

VER 4.0

 \NativeImages_v4.0.30319_32

 \NativeImages_v4.0.30319_64



GAC

So much GAC!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

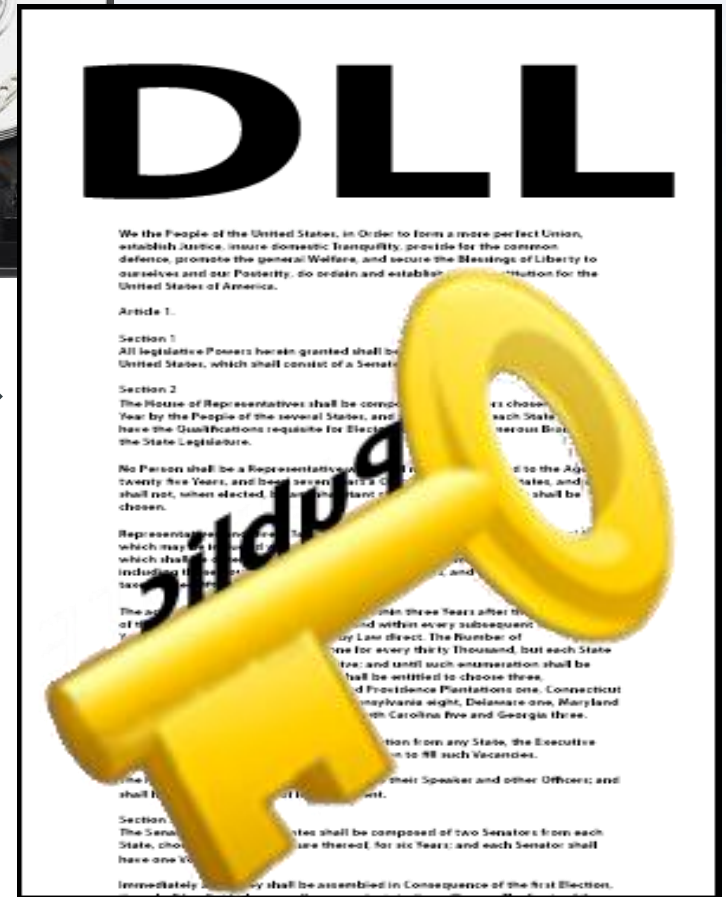
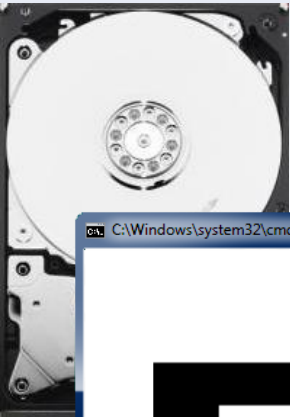
● C:\Windows\assembly\

● C:\Windows\winsxs\

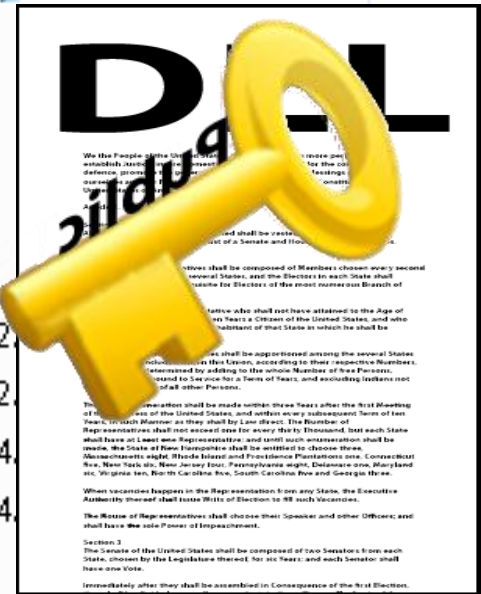
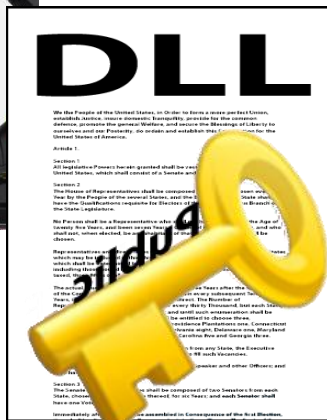
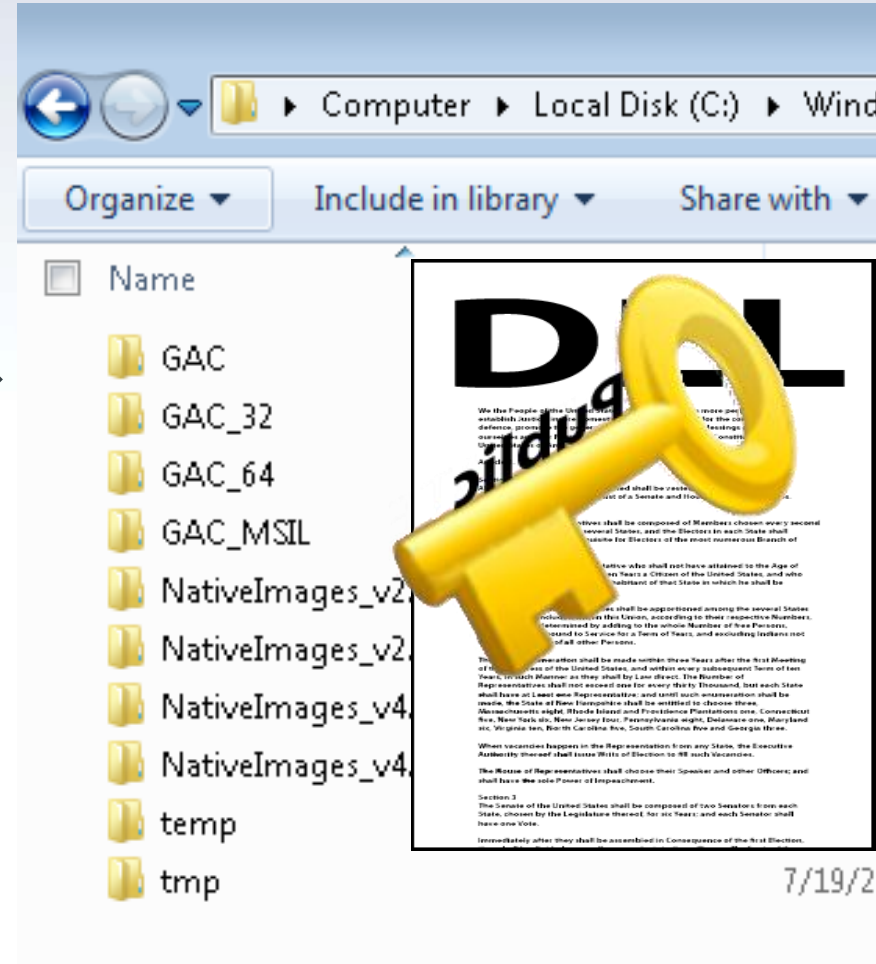
● C:\Windows\Microsoft.NET



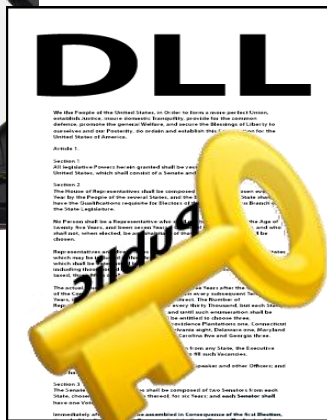
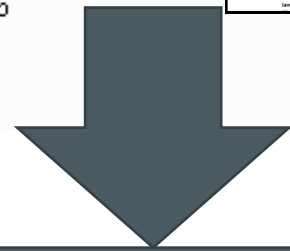
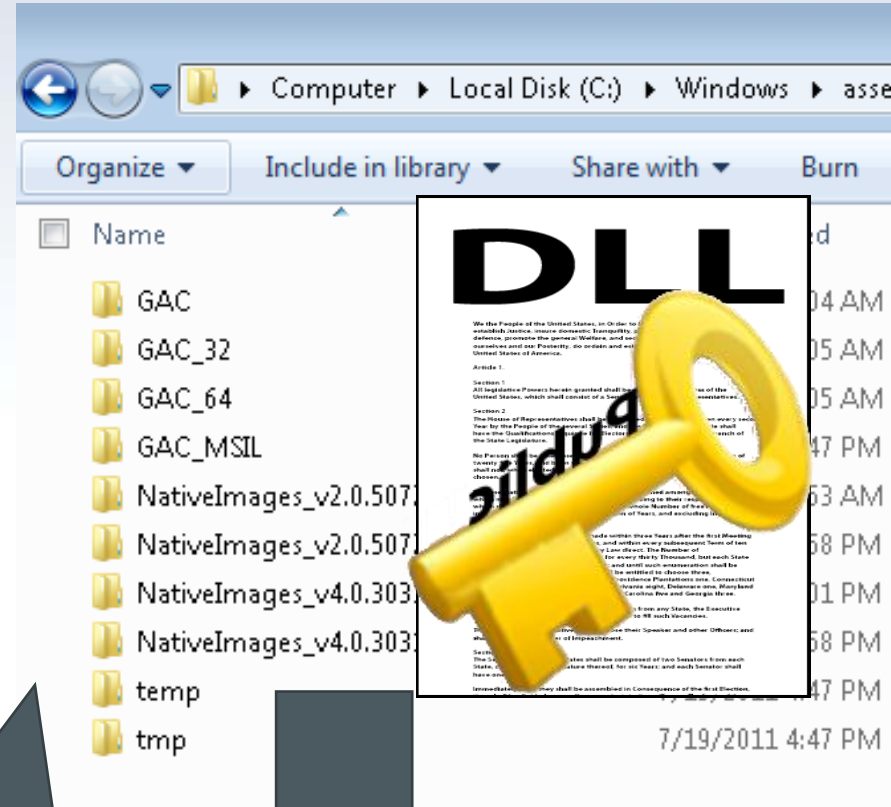
ATTACK THE GAC



ATTACK FROM THE GAC



ATTACK FROM THE GAC

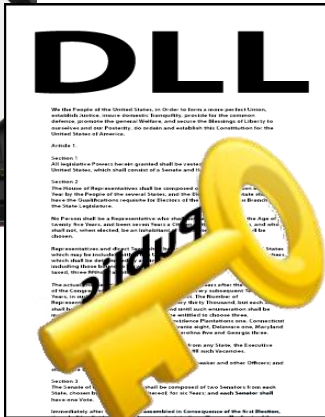
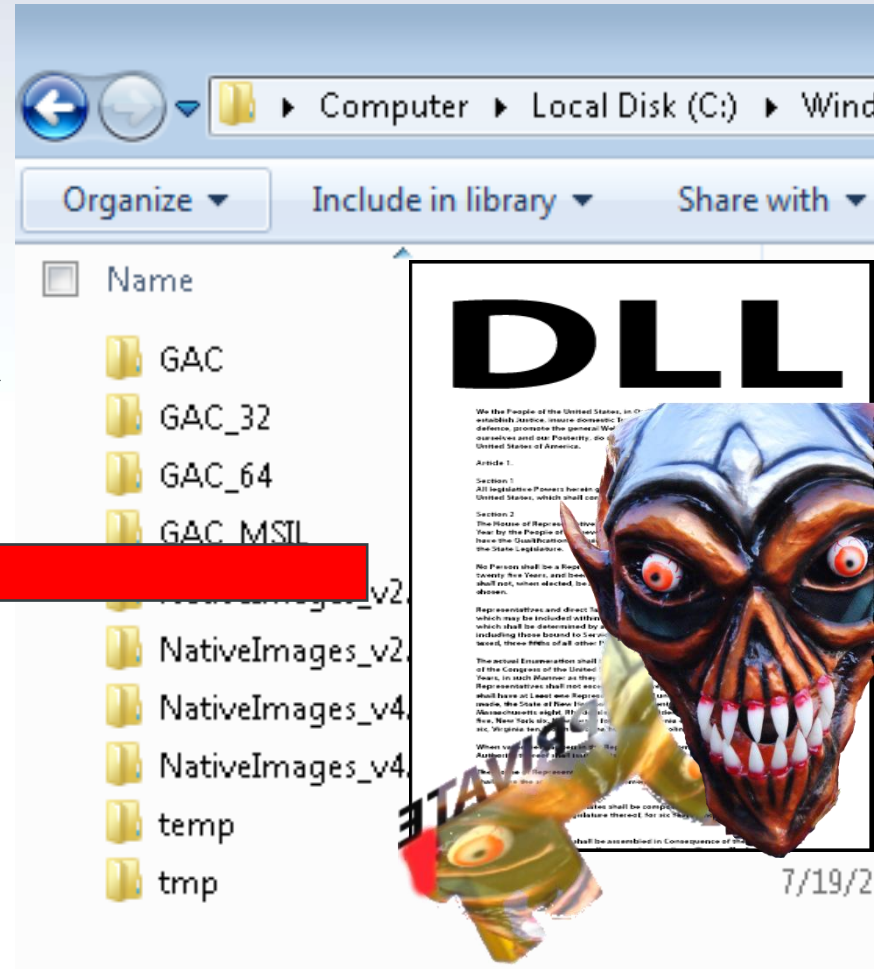
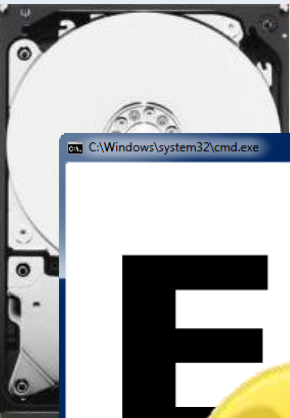


ATTACK THE GAC

1. Delete the Native Image
2. Replace File in GAC
3. Hack Target from GAC



ATTACK FROM THE GAC

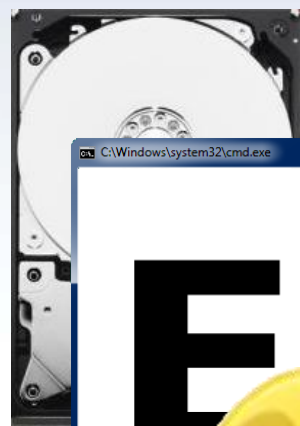


ATTACK FROM FRAMEWORK



DLL

Microsoft Word document with text and a yellow key icon.



C:\Windows\system32\cmd.exe

EXE

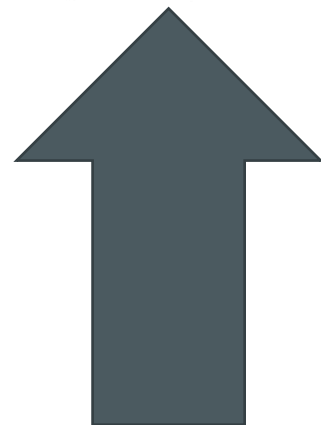
Yellow key icon with the word PRIVATE written on it.



C:\Windows\system32\cmd.exe

PROCESS

Blue brain icon.



.NET Framework



ATTACK VECTOR

ASM THE OLD IS NEW

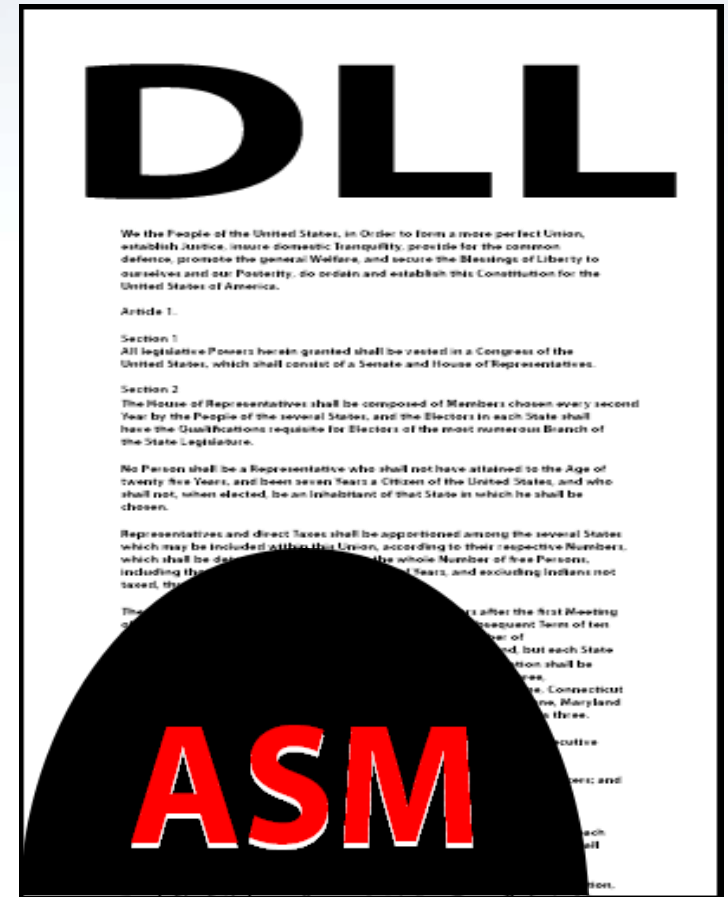
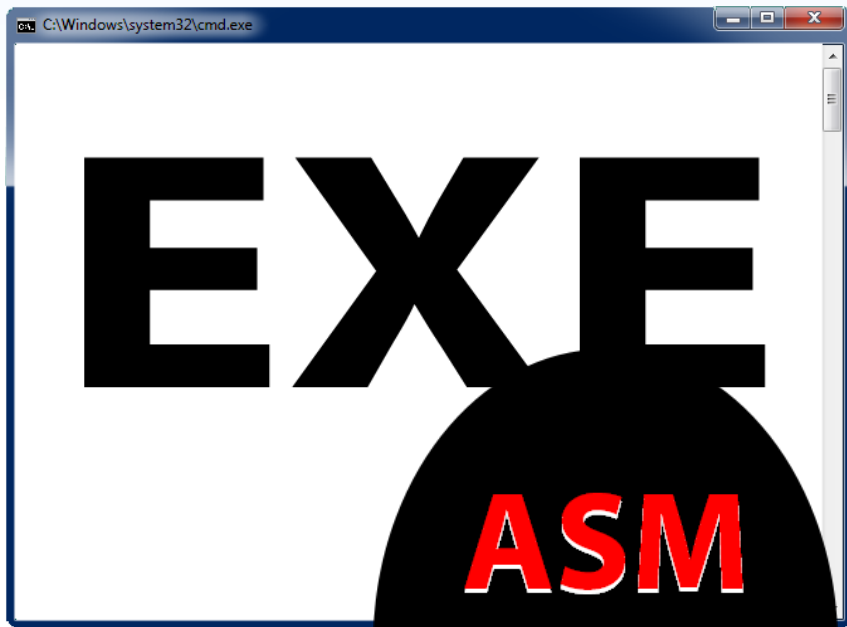


- Shell Code - ASM
- UNmanaged
- NO .NET Security
-

Execute ASM Attack with
Unmanaged Calls or Reflection



FAKE SIGNED DLL



THE OLD IS NEW AGAIN

ASM-SHELLS

● The Power of ASM

- Attack from a lower level
- Brake the “safe” security
- Attack the Runtime
- ASM-Shells.... 😊 shells...



VISUAL STUDIO

Exploit – Run arbitrary code

First noted in 2004

Demo

PowerShell - Matrix

Get developer Keys

Attack the SVN & DB

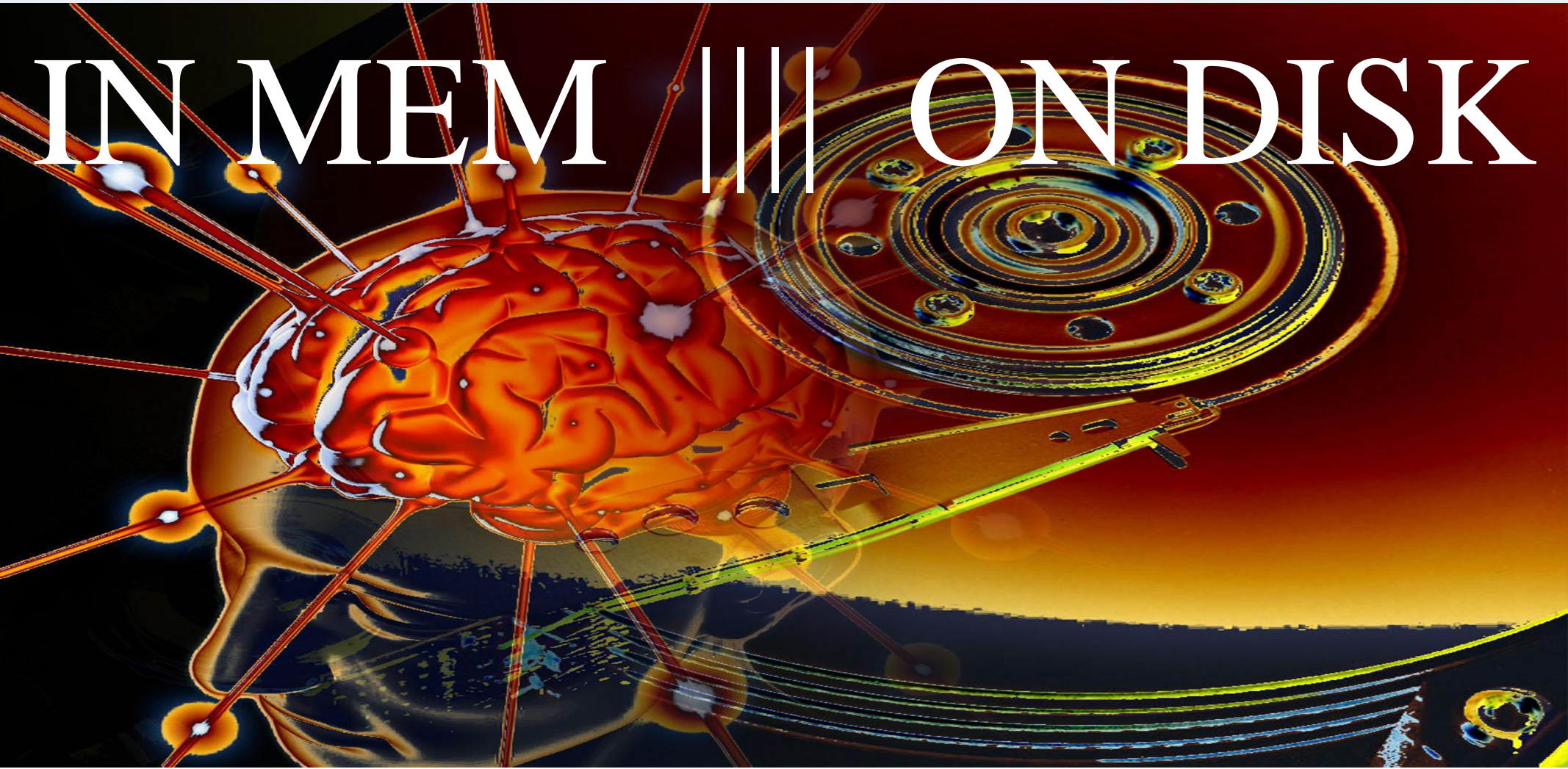
Virus

Malware

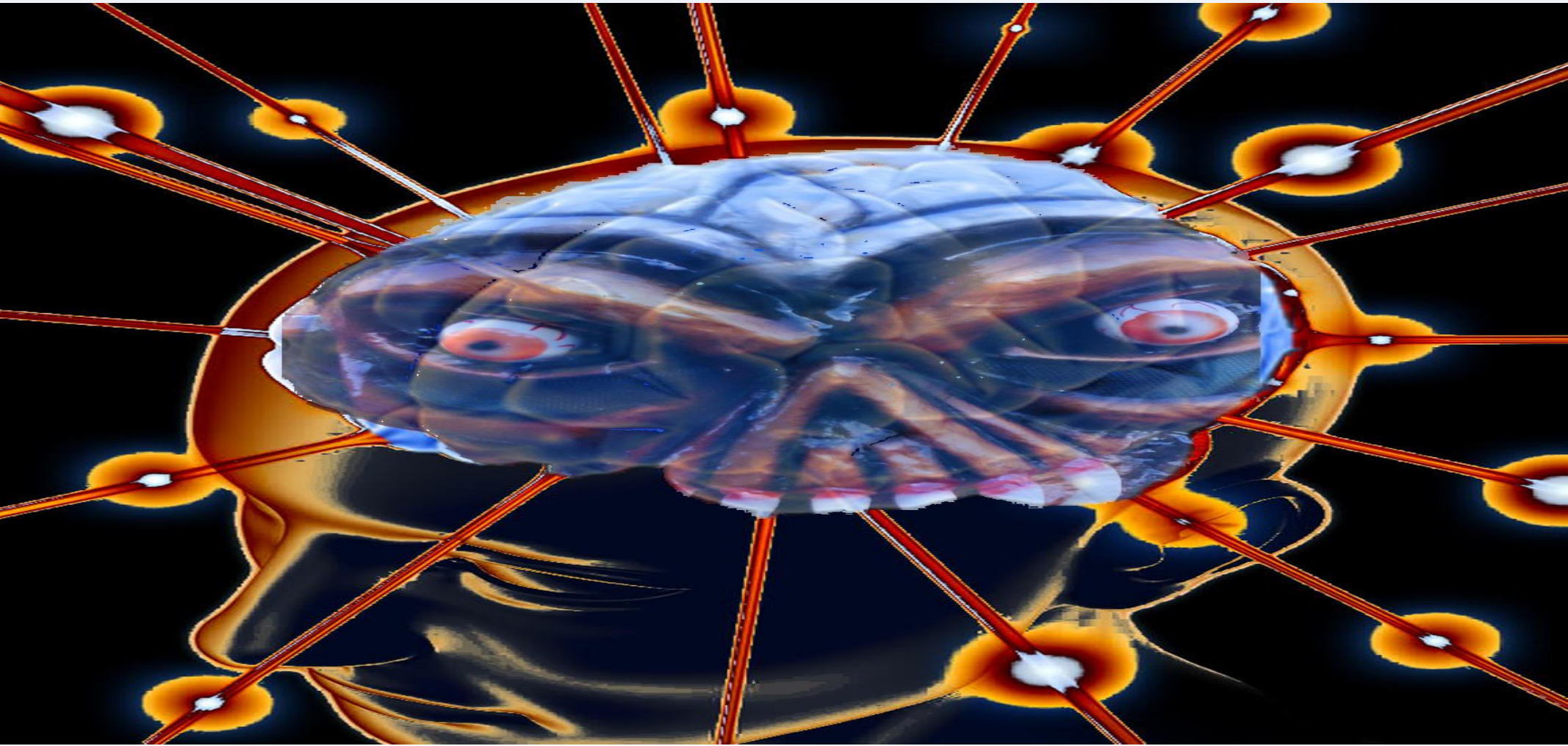


Attacking/Cracking

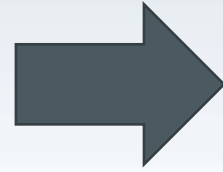
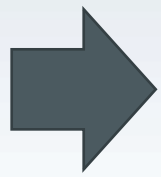
IN MEM ||| ON DISK



ATTACKING .NET APPLICATIONS: AT RUNTIME



WHY AT RUNTIME

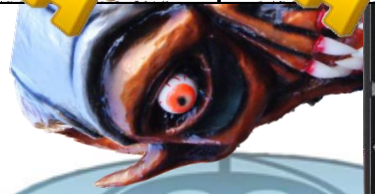
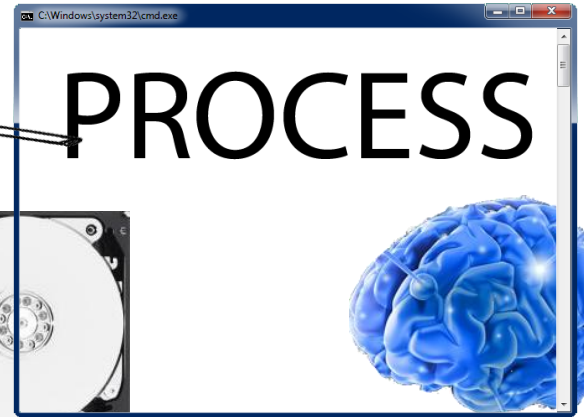
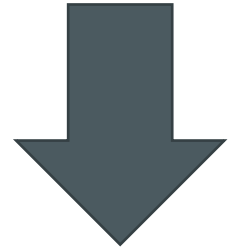
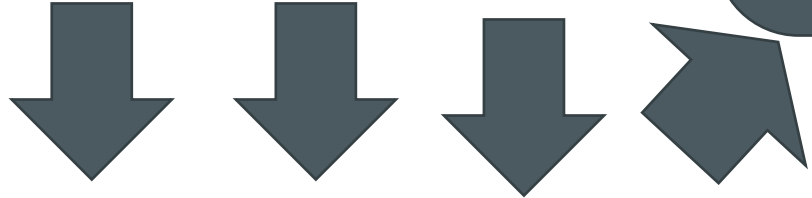


● Hacks

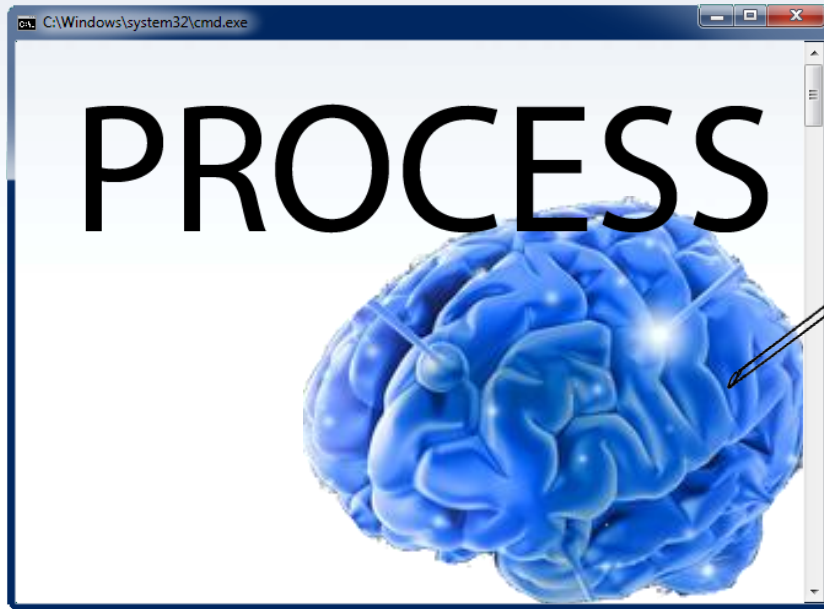
● Cracks

● Malware

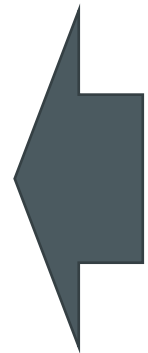
● Backdoors



Inject at Runtime



↑
.NET
DLL



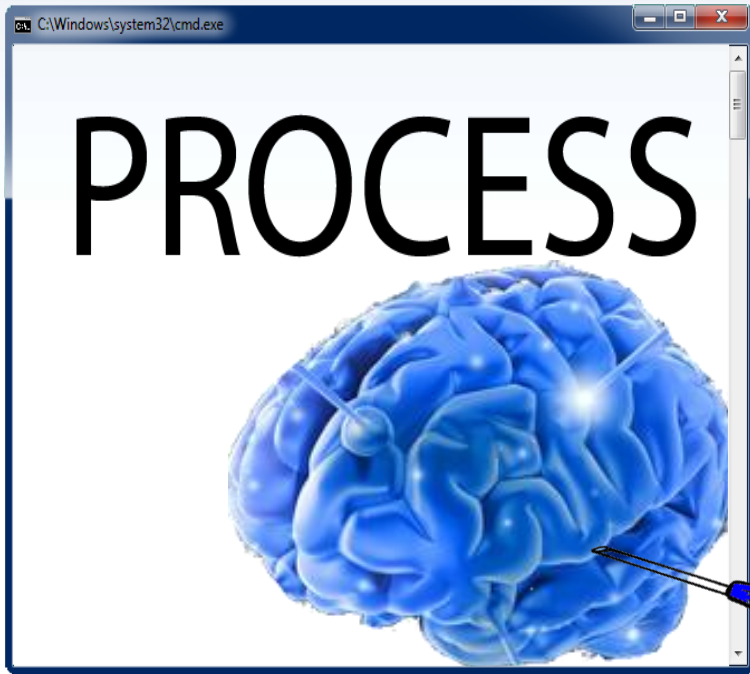
←
C++
DLL



←
GrayDragon



Inject At Runtime



ATTACKING APPS

- 🎯 Gain Full Access
 - 🎯 Reverse Engineer
 - 🎯 Attack (in MEM)
 - 🎯 Take out the “Security”
 - 🎯 Control the Program



PAST TALKS

Hacking .NET Application: A Runtime Attack

Control the Runtime
Control the Application



DEMO: GOD MODE

Inject and Control



SO YOU'RE NOT A HACKER
WHY SHOULD YOU CARE?

Defend your Applications

Defend your Systems

Verify your Tools\Programs



VERIFY YOUR APPLICATIONS

What is the Crypto & KEY

What info is it sending home

Does it have Backdoors?

Is your data Secure?



REVERSE ENGINEERING

What is going on?

What technology is used?

What is the security?

Any MaLWare?

Threat Level?



LOOK INSIDE

Take Control
Don't be helpless
Know your Threats



DON'T

LOOK

Keys

Crypto

DB

BackDoors

MalWare

Good Code

Passwords

Technology

Weak Spots

Data Leaks

Reg Code

Bad Code





SECURITY



The Login security check is

- Does $A == B$
- Does $MD5\%5 == X$
- Is the Pass the Crypto Key





DATA LEAK



The Data sent home is

- Application Info
- User / Registration Info
- Security / System Info





KEY



The Crypto Key is

- A Hard Coded Key
- The Licence Number
- A MD5 Hash of the Pass
- 6Salt 6MD5 Hash of the Pass





CRYPTO



The Crypto is

- DES 64
- Tripple DES 192
- Rijndael AES 256
- Home MIX (secure/unsecure)



MALWARE
MALWARE

T1M3
T1M3



MALWARE
MALWARE

T1M3
T1M3

So your malware

How do you hide



MALWARE T1M3
MALWARE T1M3

Protection (Shell Crypto)

Attack (Unmanaged Calls)

Protection (Obfuscated Code)

Fake (Signed DLL Protection)



MALWARE HIDE WANT TO BE A HIDE



REUSE THE TARGET

- Intelligent names
- Code style
- Don't use loops
- Don't use one area for your Vars
- Access the normal program

- Call back into your target
- Use Timers
- Link to Events
- Spread out your Vars and Code



MALWARE T1M3



MALWARE FIGHT



Protect Me! 2010



Androsa FileProtector

Androsa FileProtector

Version 1.4.4

Copyright © AndrosaSoft 2009



Protect Me! 2010

```
{
```



```
    this.filesToAdd = new List<string>();  
    base..ctor();
```

```
    this.InitializeComponent();
```

```
    this.Text = this.AssemblyTitle + " " + this.AssemblyName;
```

```
    if ((int)par.Length == 1)
```

```
    {
```

```
        if (par[0].Contains("en"))
```

```
        {
```

```
            this.langParEn = 1;
```

```
        }
```

```
    }
```

Androsa FileProtector

```
private void x03a69b6bf16c508c()
{
    var arg_ÅÅ_0 = this.xef9c50c23fdde0e7;
    object[] array = new object()[6];
    array[0] = this.x991baafb3e2f1814.getTranslation(
('meifjaagfahgfaog', 127490266));
    array[1] = string.Intern(x1110bdd110cdcea4._d57);
    array[2] = box(System.Int32, this.xe25232a1a3e32);
    array[3] = string.Intern(x1110bdd110cdcea4._d57);
    array[4] = box(System.Int32, this.xe25232a1a3e32);
    array[5] = string.Intern(x1110bdd110cdcea4._d57);
    arg_ÅÅ_0.Text = string.C
}
```

Androsa FileProtector

Version 1.4.4

Copyright © AndrosaSoft 2009

MALWARE FIGHT



Androsa File Protector

Version 1.4.4

Copyright © AndrosaSoft 2009



MALWARE FIGHT

WATNMAE EICHL



Protect Me! 2010

Good Crypto

Salt & VI

Encrypted Pass

Password SHA512

Custom Crypto LIB's

Possible Back Door

Ob\$cur17y

Androsa FileProtector

Androsa FileProtector

Version 1.4.4

Copyright © AndrosaSoft 2009



MALWARE FIGHT



DEMO



Androsa File Protector

Version 1.4.4

Copyright © AndrosaSoft 2009



PROTECTION FOR WHO?



Obfu\$ca73



WHAT M\$ DID **RIGHT**

Un-obfuscated Code

- Σ Good for user security

- Σ User can see what they are running

.NET Framework Security

- Σ Targeted Security Access

- Σ Protect the Computer from the app

Giving Reduced Rights Inside Code

- Σ Put venerable code in a box

- Σ Mitigate & Segment Risk



WHAT M\$ DID **WRONG**

MixModeCode – Bad for security

- ∑ This allows ASM\C++\C code

- ∑ This breaks out of .NET security

GAC & Native Image Override

- ∑ Removes ability to secure code

Not Hash Checking Code

- ∑ Good for hackers



ATTACKING APPS

- 🎯 Read my papers: Reflections Hidden Power & Attacking .NET at Runtime
- 🎯 Watch 2010 Presentations on Attacking .NET DefCon 18, AppSec-DC, DojoCon
- 🎯 Look up Presentations and Research from Andrew Willson, Erez Ezule, Arndet Mandent
- 🎯 Use tools: Visual Studio/MonoDev Reflector/GrayWolf/ILspy/.../ILASM/ILDASM



FIN



MORE INFORMATION @:
www.DigitalBodyGuard.com

FIN = 1



HACKER VS ATTACKER



101 - Recon



● File Location

C:\Windows\ehome\ehshell.dll

● StrongName KEY

d:\w7rtm.public.x86fre\internal\strongnamekeys\fake\windows.snk

● Registry CurrentUser OR LocalMachine

SOFTWARE\Microsoft\Windows\CurrentVersion\Media Center\

● Web Host Address

www.microsoft.com/WindowsMedia/Services/2003/10/10/movie



101 - Recon

EHSHELL ● .NET Framework
Ver 3.5



Windows
Media Center

● Un-~~ob~~*fu\$ca7ed*

● Crash Reporting
Watson

● Coded in C#

