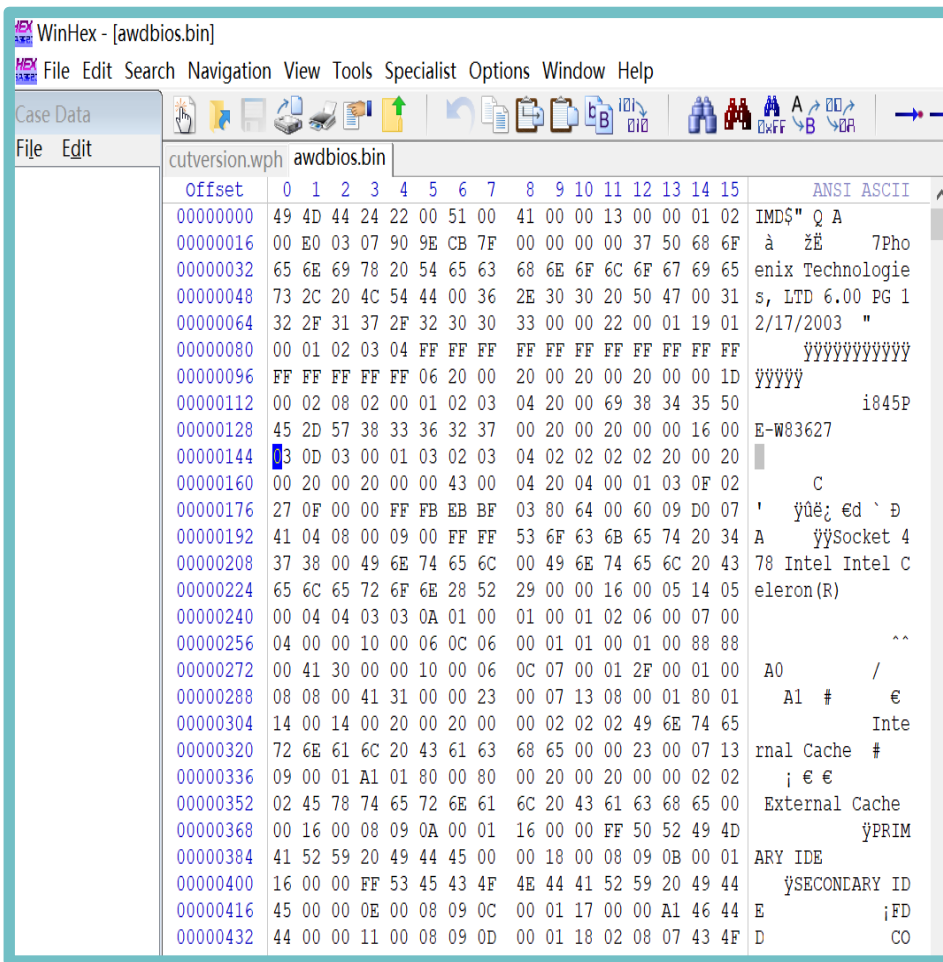


逆向分析技术

课程回顾

[illegible]

6	7	8	9	10	11	12	13	14	15	ANSI ASCII
51	00	41	00	00	13	00	00	01	02	IMD\$" Q A
CB	7F	00	00	00	00	37	50	68	6F	à žĚ 7Pho
65	63	68	6E	6F	6C	6F	67	69	65	enix Technologie
00	36	2E	30	30	20	50	47	00	31	s, LTD 6.00 PG 1
30	30	33	00	00	2	<div>Find Text</div> <div>The following text string will be searched:</div> <div>phoenix</div>				
FF	FF	FF	FF	FF	F					
20	00	20	00	20	0					
02	03	04	20	00	6					
32	37	00	20	00	2					
00	00	00	00	00	0					

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	ANSI ASCII	⬆	
49	4D	44	24	22	00	51	00	41	00	00	13	00	00	01	02	IMD\$" Q A		
00	E0	03	07	90	9E	CB	7F	00	00	00	00	37	50	68	6F	à žĚ 7Pho		
65	6E	69	78	20	54	65	63	68	6E	6F	6C	6F	67	69	65	enix Technologie		
73	2C	20	4C	54	44	00	36	2E	30	30	20	50	47	00	31	s, LTD 6.00 PG 1		
32	2F	31	37	2F	32	30	30	33	00	00	22	00	01	19	01	2/17/2003 "		
00	01	02	03	04	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	ÿÿÿÿÿÿÿÿÿÿ		
FF	FF	FF	FF	FF	06	20	00	20	00	20	00	20	00	30	00	1D	ÿÿÿÿÿ	
00	02	08	02	00	01	02	03	04	20	00	69	38	34	35	50		i845P	
15	0F	57	38	33	36	38	37	38	38	38	38	38	38	16	38	F 582687		

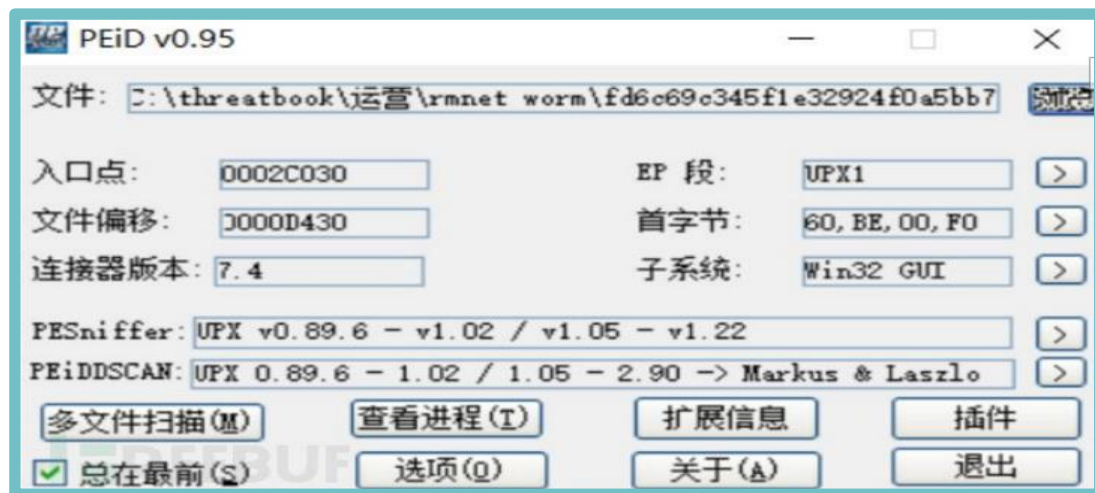
[illegible]

回顾： 二进制查壳 - PEiD

识别zprotect



识别UPX



回顾：静态分析 - IDA

IDA - D:\winhex\whxsetup.idb (whxsetup.exe)

File Edit Jump Search View Debugger Options Windows Help

Library function Data Regular function Unexplored Instruction External symbol

Functions window

- GetStdHandle
- UnhandledExceptionFilter
- WriteFile
- ExitProcess
- MessageBoxA
- FreeLibrary
- GetCommandLineA
- GetStartupInfoA
- RegCloseKey
- RegOpenKeyExA
- RegQueryValueExA
- GetCurrentThreadld
- sub_401070
- LocalFree
- VirtualFree
- EnterCriticalSection
- LeaveCriticalSection
- DeleteCriticalSection
- sub_4010BC
- sub_4010C4
- sub_4011A4
- sub_4011B0
- sub_401208
- sub_401218
- sub_401258

Line 24 of 227

Graph overview

```
sub_401218 proc near
push    esi
push    edi
mov     esi, eax
mov     edi, edx
mov     eax, ecx
cmp     edi, esi
ja      short loc_401237

jz      short loc_401255

loc_401237:
lea     esi, [ecx+esi-4]
lea     edi, [ecx+edi-4]
sar     ecx, 2
js      short loc_401255

sar     ecx, 2
js      short loc_401255

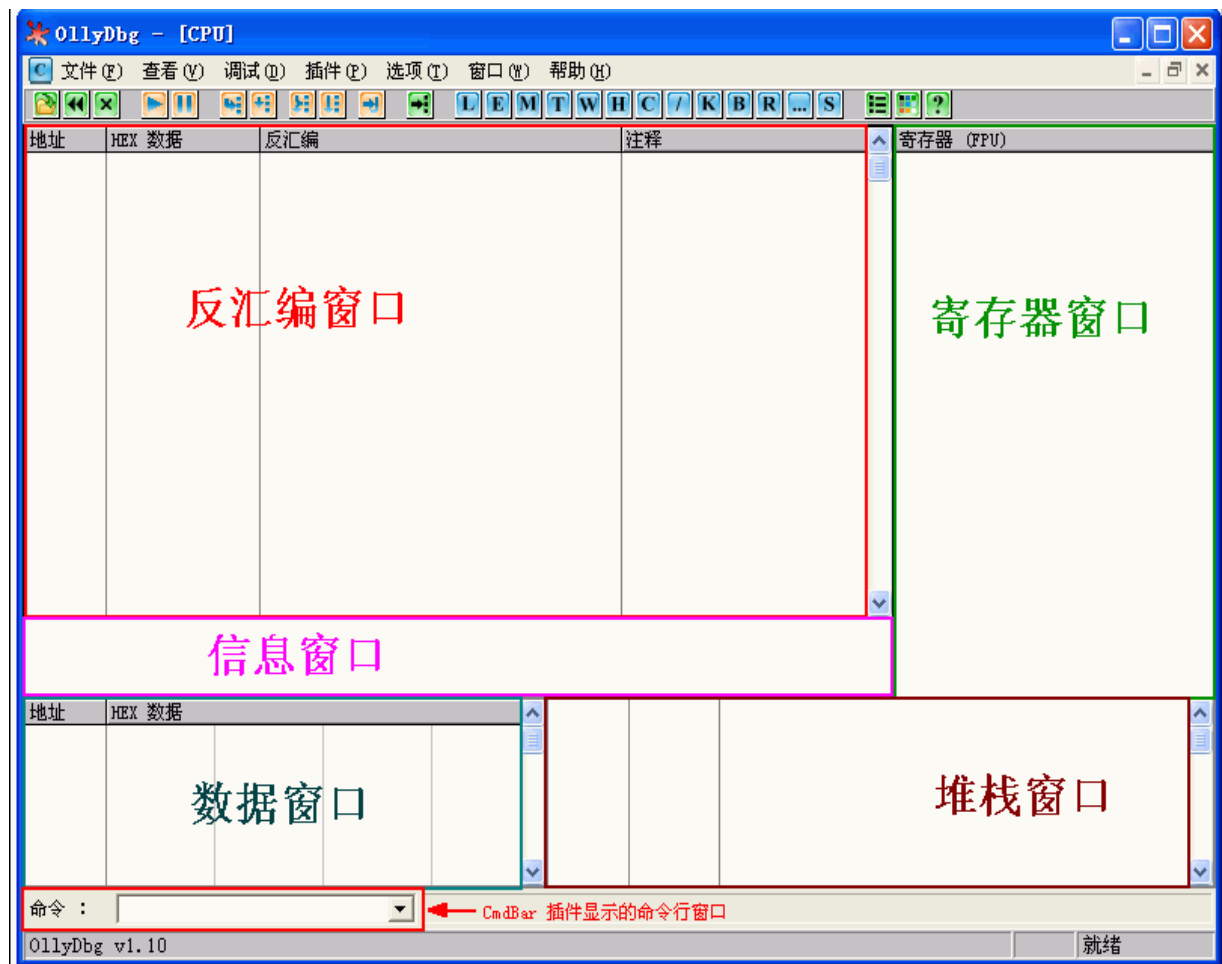
std     rep movsd
mov     ecx, eax
and     ecx, 3
add     esi, 3
add     edi, 3
rep     movsb
cld

100.008 (-5,-5) (587,5) 00000618 00401218: sub_401218
```

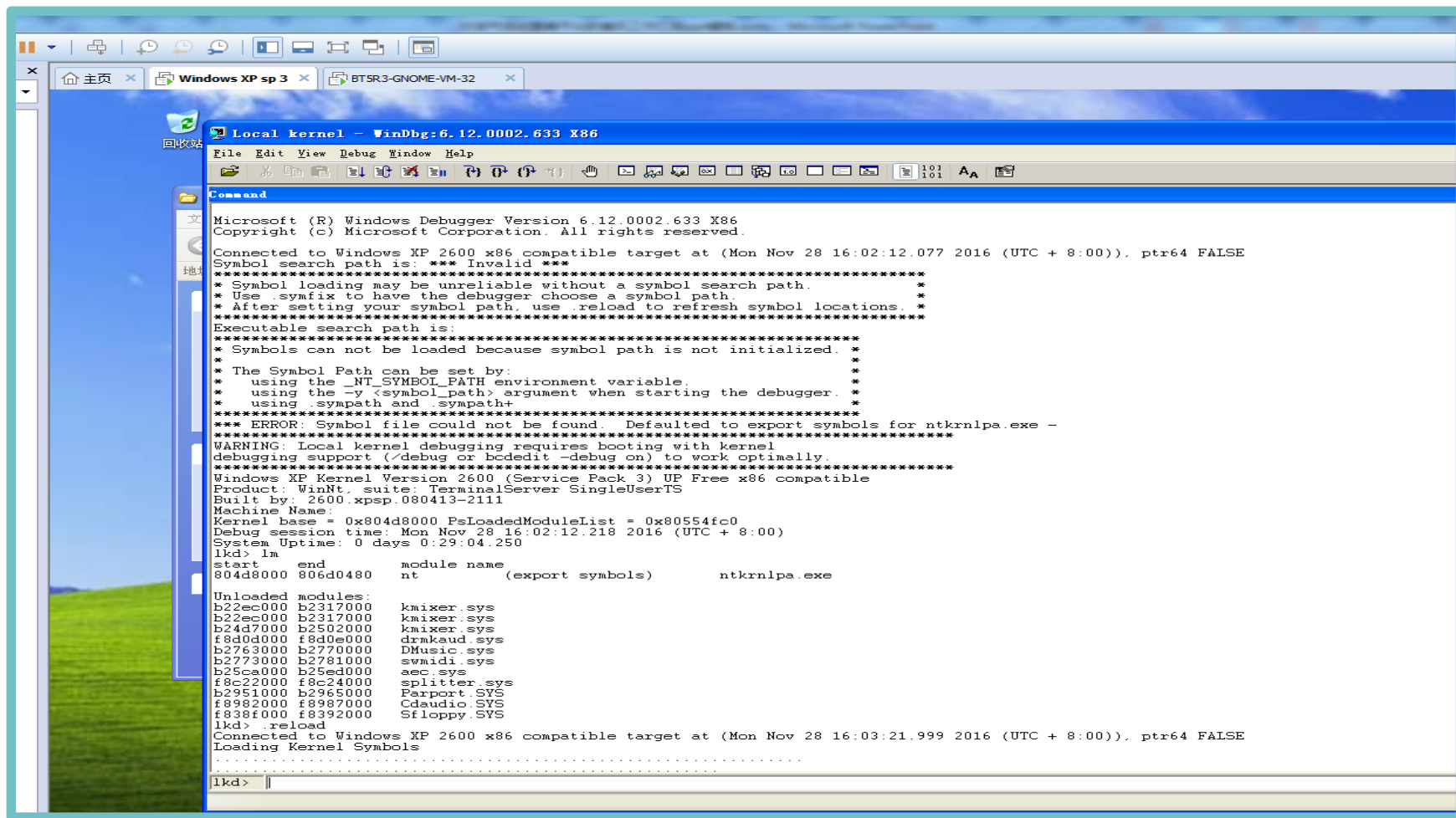
IDA View-A Hex View-1

004011D8	00 00 00 EB 0F 80 FB 18	77 0A 33 C0 8A C3 8A 98	...
004011E8	24 70 40 00 33 C0 8A C3	8B D6 E8 AD FF FF FF 5E	\$p@.
004011F8	5B C3 8B C0 83 E0 7F 8B	14 24 E9 A9 FF FF FF C3	[...
00401208	53 8B D8 E8 54 11 00 00	89 98 04 00 00 00 5B C3	S...
00401218	56 57 89 C6 89 D7 89 C8	39 F7 77 13 74 2F C1 F9	UW...
00401228	02 78 2A F3 A5 89 C1 83	E1 03 F3 A4 5F 5E C3 8D	..x*.
00401238	74 31 FC 8D 7C 39 FC C1	F9 02 78 11 FD F3 A5 89	t1..
00401248	C1 83 E1 03 83 C6 03 83	C7 03 F3 A4 FC 5F 5E C3	...
00401258	53 56 8B D8 33 F6 66 8B	43 04 66 3D B1 D7 72 2F	SU...
00401268	66 3D B3 D7 77 29 66 25	B2 07 66 3D B2 D7 75 07	f=...
00401278	8B C3 FF 53 1C 8B F0 85	F6 75 07 8B C3 FF 53 24	...S
00401288	8B F0 85 F6 74 1B 8B C6	E8 73 FF FF FF EB 12 81	...
00401298	FB 38 80 40 00 74 0A B8	67 00 00 00 E8 5F FF FF	.8M
004012A8	FF 8B C6 5E 5B C3 8B C0	57 89 C7 88 CD 89 C8 C1	...
004012B8	E0 10 66 89 C8 89 D1 C1	F9 02 78 09 F3 AB 89 D1	...f.
004012C8	83 E1 03 F3 AA 5F C3 90	53 56 8B F2 8B D8 33 C9	...
004012D8	8A 0B 8B D6 8D 43 01 E8	34 FF FF FF 33 C0 8A 03	...
004012E8	C6 04 06 00 5E 5B C3 90	53 31 DB 57 89 D7 89 C1	...
004012F8	83 EC 0A C6 07 30 DC 8B	40 70 40 00 D9 FC DC 93	...
00401308	48 70 40 00 9B DF E0 9E	72 09 DC A3 48 70 40 00	Hp@.
00401318	C6 07 31 DF 34 24 BA 08	00 00 00 47 9B 8A 04 14	..1.
00401328	8B C4 C0 E8 04 80 E4 0F	66 05 30 30 66 AB 4A 79	...
00401338	EB 83 E9 12 7C 06 B0 30	F3 AA EB 14 01 CF 80 3F	...
00401348	35 7C 0D 4F FE 07 80 3F	39 7E 05 C6 07 30 EB F3	5 .C
00401358	83 C4 0A 5F 5B C3 8B C0	53 83 EC 0C 31 D8 D9 C0	...<\$f
00401368	DB 3C 24 66 8B 44 24 08	66 85 C0 74 1B 66 2D FF	?F...
00401378	3F 66 BA 10 4D 66 F7 EA	0F BF C2 F7 D8 74 16 29	...
00401388	C3 E8 D2 01 00 00 EB D6	83 7C 24 04 00 75 DE 83	<\$..C
00401398	3C 24 00 75 D8 83 C4 0C	89 D8 5B C3 DB 6C 24 04	SUW.
004013A8	53 56 57 89 C3 89 D6 51	81 EC 00 01 00 00 81 FB	...
004013B8	FF 00 00 00 7E 05 B8 FF	00 00 00 D9 C0 D8 3C 24	1.f.
004013C8	31 C0 66 8B 44 24 08 89	C7 C1 EF 0F 66 25 FF 7F	f=...
004013D8	66 3D FF 7F 0F 84 3E 01	00 00 D9 E1 85 F6 0F 8D	...

回顾： 动态分析 - OllyDbg



回顾：内核动态分析 - WinDbg



回顾： 动态分析 - Linux - gdb

```
(gdb) c
Continuing.

Breakpoint 2, main () at gdb-sample.c:21
21             printf("n=%d,nGlobalVar = %d /n", n, nGlobalVar);
(gdb) p nGlobalVar
$2 = 88
(gdb) c
Continuing.

Breakpoint 3, tempFunction (a=1, b=2) at gdb-sample.c:7
7             printf("tempFunction is called, a = %d, b = %d /n", a, b);
(gdb) p a
$3 = 1
(gdb) p b
$4 = 2
```


回顾：内核动态分析 - Linux - kgdb

Re-compiling linux kernel

```
linux-kgdb:/home/linux-2.6.32.12-0.7 # ls /boot/ -l
total 37648
-rw-r--r-- 1 root root 1617387 Nov 26 08:51 System.map-2.6.32.12-0.7-default
-rw-r--r-- 1 root root 1617387 Nov 26 08:51 System.map-2.6.32.12-0.7-default.old
-rw-r--r-- 1 root root 512 Nov 24 11:05 backup_mbr
lrwxrwxrwx 1 root root 1 Nov 24 10:58 boot -> .
-rw-r--r-- 1 root root 1236 May 10 2010 boot.readme
-rw-r--r-- 1 root root 107874 May 20 2010 config-2.6.32.12-0.7-default
drwxr-xr-x 2 root root 4096 Nov 26 11:15 grub
lrwxrwxrwx 1 root root 28 Nov 26 13:40 initrd -> initrd-2.6.32.12-0.7-default
-rw-r--r-- 1 root root 13777267 Nov 26 13:40 initrd-2.6.32.12-0.7-default
-rw-r--r-- 1 root root 6572832 Nov 26 09:19 initrd-2.6.32.12-0.7-default.org
-rw-r--r-- 1 root root 435712 Nov 24 11:05 message
-rw-r--r-- 1 root root 189729 May 20 2010 symsets-2.6.32.12-0.7-default.tar.gz
-rw-r--r-- 1 root root 495291 May 20 2010 sytypes-2.6.32.12-0.7-default.gz
-rw-r--r-- 1 root root 178468 May 20 2010 symvers-2.6.32.12-0.7-default.gz
-rw-r--r-- 1 root root 3774506 May 20 2010 vmlinux-2.6.32.12-0.7-default.gz
lrwxrwxrwx 1 root root 29 Nov 24 11:02 vmlinux -> vmlinux-2.6.32.12-0.7-default
-rw-r--r-- 1 root root 3205728 Nov 26 08:51 vmlinux-2.6.32.12-0.7-default
-rw-r--r-- 1 root root 3231872 Nov 26 13:42 vmlinux-2.6.32.12-0.7-default.old
-rw-r--r-- 1 root root 3231872 Nov 26 09:19 vmlinux-2.6.32.12-0.7-default.org
```

Open Kgdb options

```
root@keven-ubuntu:/home/keven/kgdb_shared# gdb vmlinux
GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1) 7.4-2012.04
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show
warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
For bug reporting instructions, please see:
<http://bugs.launchpad.net/gdb-linaro/>...
Reading symbols from /home/keven/kgdb_shared/vmlinux...done.
(gdb) set remotebaud 115200
(gdb) target remote /dev/pts/0
Remote debugging using /dev/pts/0
kgdb_breakpoint () at kernel/kgdb.c:1718
1718 kernel/kgdb.c: 没有那个文件或目录.
(gdb) target remote /dev/pts/0
A program is being debugged already. Kill it? (y or n) y

Remote debugging using /dev/pts/0
Ignoring packet error, continuing...
```




逆向分析技术

常用逆向工具



本次课程支撑的毕业要求指标点

- 毕业要求5-3:

能够分析比较所使用的技术、资源和工具的优势和不足，并理解与表述问题解决方案的局限性。

OllyDbg工具的使用 - 了解程序机制

以MessageBox的代码为例

```
#include<windows.h>
int main()
{
    MessageBox(0,"You have been hacked","warning",0);
    return 0;
}
```

运行结果：



OllyDbg工具的使用 - 了解程序机制

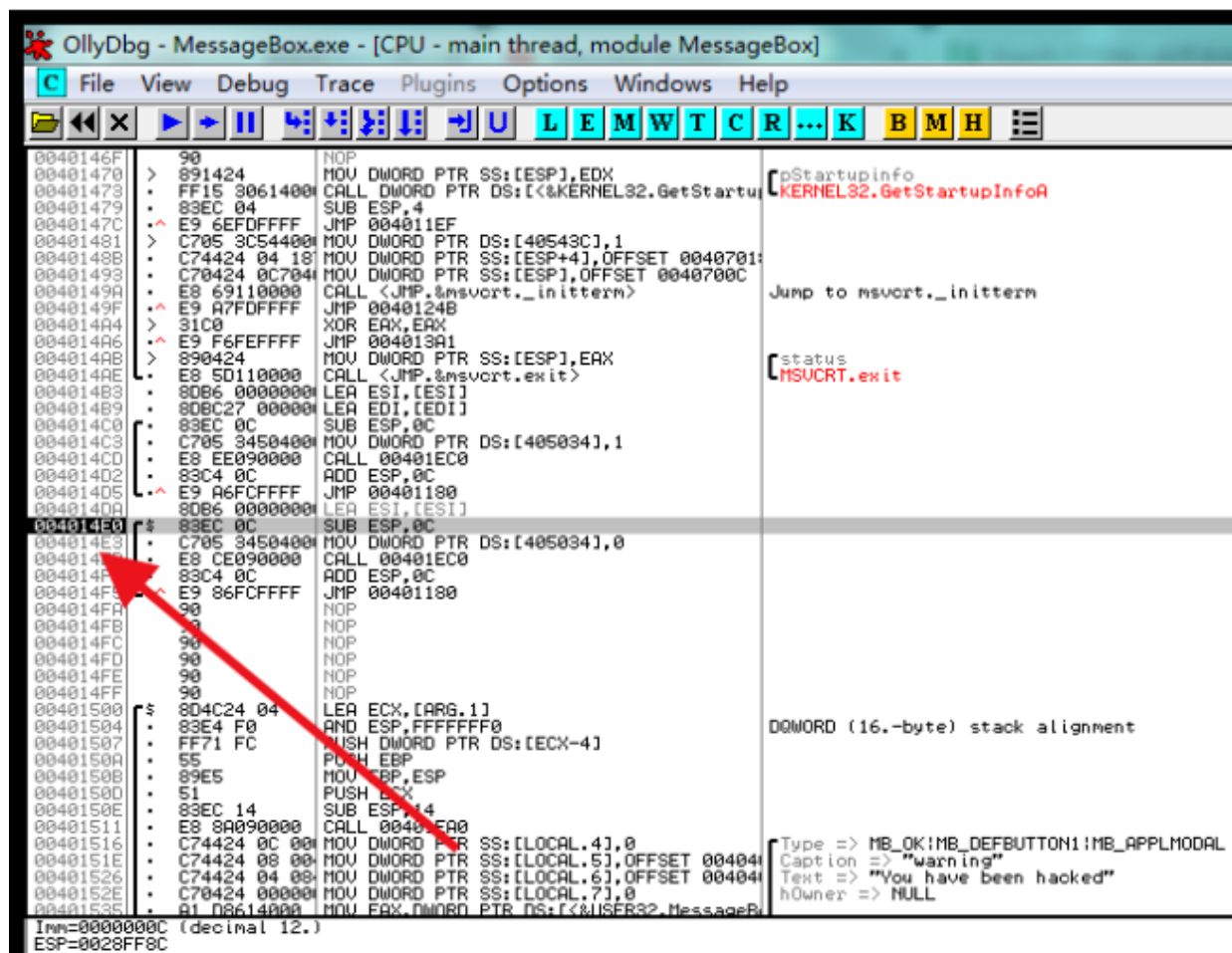
The screenshot displays the OllyDbg interface with three main windows highlighted by red boxes and Chinese labels:

- 汇编代码窗口 (Assembly Code Window):** Located at the top left, it shows the disassembled code. The current instruction is at address 004014E0: `83EC 0C SUB ESP,0C`. The address 004014E0 is highlighted in red.
- 寄存器窗口 (Registers Window):** Located at the top right, it shows the state of the CPU registers. The EIP register is highlighted in red, showing the value 004014E0, which corresponds to the current instruction address.
- 堆栈窗口 (Stack Window):** Located at the bottom, it shows the memory stack. The stack pointer (ESP) is highlighted in red, showing the value 0028FF94.

Additional details visible in the interface include the menu bar (File, View, Debug, Trace, Plugins, Options, Windows, Help), the toolbar, and the status bar at the bottom showing the current address and hex dump.

OllyDbg工具的使用 - 了解程序机制

用OD打开程序



OllyDbg - MessageBox.exe - [CPU - main thread, module MessageBox]

File View Debug Trace Plugins Options Windows Help

0040146F 90 NOP
00401470 > 891424 MOV DWORD PTR SS:[ESP],EDX
00401473 • FF15 30614000 CALL DWORD PTR DS:[<KERNEL32.GetStartupInfoA
00401479 • 83EC 04 SUB ESP,4
0040147C ^ E9 6EFDFFFF JMP 004011EF
00401481 > C705 3C544000 MOV DWORD PTR DS:[40543C],1
00401488 • C74424 04 18 MOV DWORD PTR SS:[ESP+4],OFFSET 00407018
00401493 • C70424 0C704000 MOV DWORD PTR SS:[ESP],OFFSET 0040700C
0040149A • E8 69110000 CALL <JMP.&msvort._initterm> Jump to msvort._initterm
0040149F ^ E9 A7FDFFFF JMP 0040124B
004014A4 > 31C0 XOR EAX,EAX
004014A6 ^ E9 F6FEFFFF JMP 004013A1
004014AB > 890424 MOV DWORD PTR SS:[ESP],EAX
004014AE [E8 5D110000 CALL <JMP.&msvort.exit> [status] MSUCRT.exit
004014B3 • 8DB6 00000000 LEA ESI,[ESI]
004014B9 • 8DBC27 00000000 LEA EDI,[EDI]
004014C0 • 83EC 0C SUB ESP,0C
004014C3 • C705 34504000 MOV DWORD PTR DS:[405034],1
004014CD • E8 EE090000 CALL 00401EC0
004014D2 • 83C4 0C ADD ESP,0C
004014D5 ^ E9 A6FCFFFF JMP 00401180
004014DA • 8DB6 00000000 LEA ESI,[ESI]
004014E0 \$ 83EC 0C SUB ESP,0C
004014E3 • C705 34504000 MOV DWORD PTR DS:[405034],0
004014E6 • E8 CE090000 CALL 00401EC0
004014F0 • 83C4 0C ADD ESP,0C
004014F3 ^ E9 86FCFFFF JMP 00401180
004014FA 90 NOP
004014FB 90 NOP
004014FC 90 NOP
004014FD 90 NOP
004014FE 90 NOP
004014FF 90 NOP
00401500 \$ 804C24 04 LEA ECX,[ARG.1]
00401504 • 83E4 F0 AND ESP,FFFFFFF0 DQWORD (16.-byte) stack alignment
00401507 • FF71 FC PUSH DWORD PTR DS:[ECX-4]
0040150A • 55 PUSH EBP
0040150B • 89E5 MOV BP,ESP
0040150D • 51 PUSH ECX
0040150E • 83EC 14 SUB ESP,14
00401511 • E8 8A090000 CALL 00401FA0
00401516 • C74424 0C 00 MOV DWORD PTR SS:[LOCAL.4],0
0040151E • C74424 08 00 MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404040
00401526 • C74424 04 00 MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404040
0040152E • C70424 00000000 MOV DWORD PTR SS:[LOCAL.7],0
00401535 • 01 08614000 MOV EAX,DWORD PTR DS:[<USER32.MessageB
Type => MB_OK|MB_DEFBUTTON1|MB_APPLMODAL
Caption => "warning"
Text => "You have been hacked"
hOwner => NULL
EIP=00401500 (decimal 12.)
ESP=0028FF8C

OllyDbg工具的使用 - 找到程序真正入口

借助IDA

The screenshot displays the IDA Pro interface with three main windows: Functions window, IDA View-A, and Hex View-1. The Functions window on the left lists various functions, with `_main` highlighted. A red arrow labeled '1' points from this list to the IDA View-A window. The IDA View-A window shows the assembly code for the `_main` function, which is circled with a red '2'. A red arrow labeled '3' points from the `_main` function name in the IDA View-A window to the Hex View-1 window. The Hex View-1 window shows the memory address `401500` highlighted in red, which is the entry point of the `_main` function. The status bar at the bottom indicates the current address is `100.00% (-345, -45) (14, 381) 00000900 0000000000401500: _main (Synchronized with Hex View-1)`.

Function name

- `__mingw_invalidParameterHandler`
- `_pre_c_init`
- `_pre_cpp_init`
- `__tmainCRTStartup`
- `_mainCRTStartup`
- `_main`**
- `__dyn_tls_dtor@12`
- `__dyn_tls_init@12`
- `__tlregdtor`
- `__my_lconv_init`
- `__decode_pointer`
- `__encode_pointer`
- `__mingw_onexit`
- `__atexit`
- `__gnu_exception_handler@4`
- `__setargv`
- `__mingw_raise_matherr`
- `__mingw_setusermatherr`
- `__matherr`
- `__report_error`
- `__pei386_runtime_relocator`
- `__fpreset`
- `__do_global_dtors`
- `__do_global_ctors`
- `__main`
- `__security_init_cookie`

Line 6 of 65

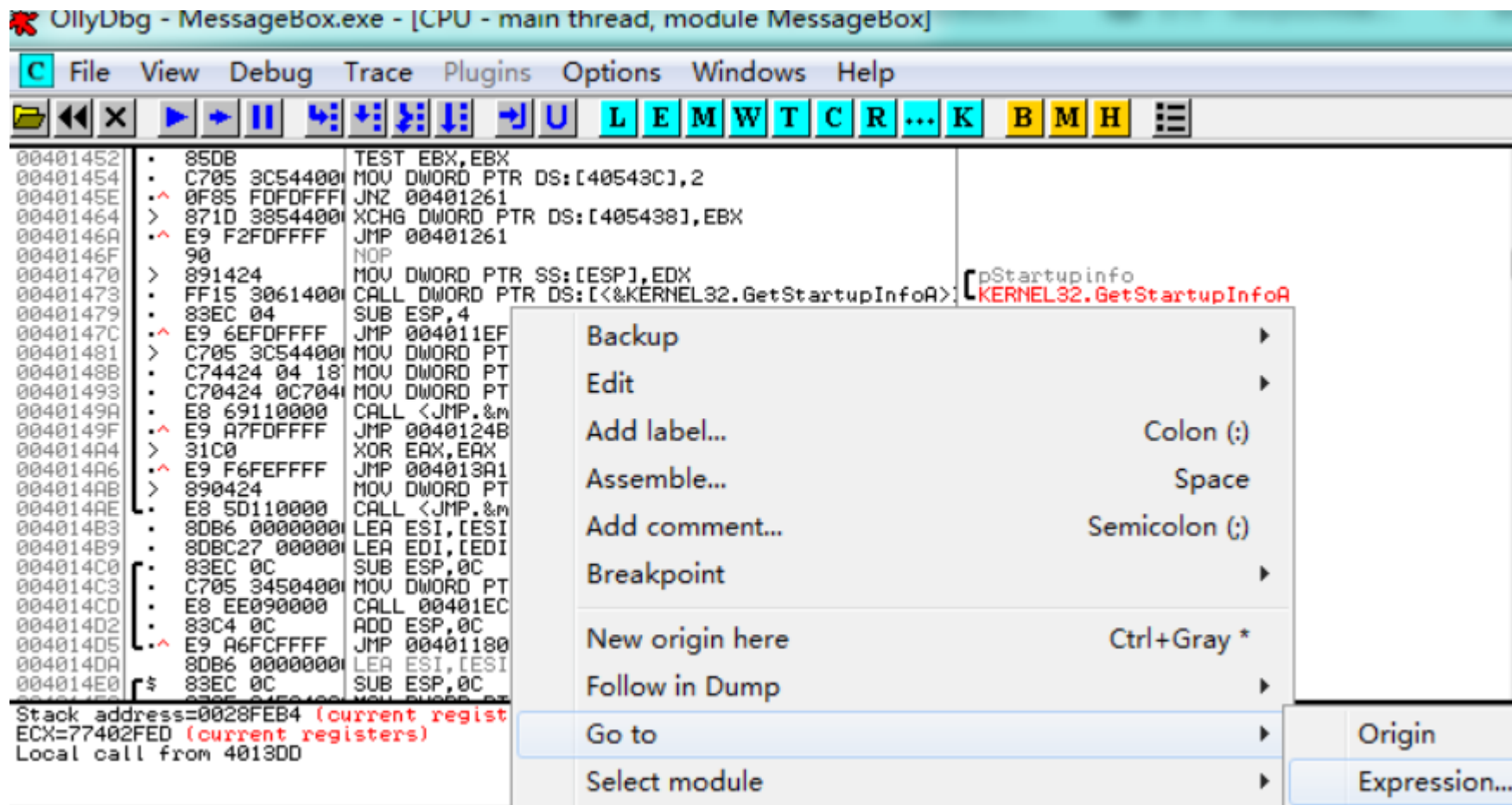
Graph overview

```
; Attributes: bp-based frame fuzzy-sp  
  
; int main()  
public _main  
_main proc near  
  
var_4= dword ptr -4  
  
lea     ecx, [esp+4]  
and     esp, 0FFFFFFFh  
push    dword ptr [ecx-4]  
push    ebp  
mov     ebp, esp  
push    ecx  
sub     esp, 14h  
call    __main  
mov     dword ptr [esp+0Ch], 0 ; uType  
mov     dword ptr [esp+8], offset Caption ; "warning"  
mov     dword ptr [esp+4], offset hWnd ; lpText  
mov     dword ptr [esp], 0 ; hWnd  
mov     eax, ds: __imp_MessageBoxA@16  
call    eax ; __imp_MessageBoxA@16  
sub     esp, 10h  
mov     eax, 0  
mov     ecx, [ebp+var_4]  
leave  
lea     esp, [ecx-4]  
retn  
_main endp
```

100.00% (-345, -45) (14, 381) 00000900 0000000000401500: _main (Synchronized with Hex View-1)

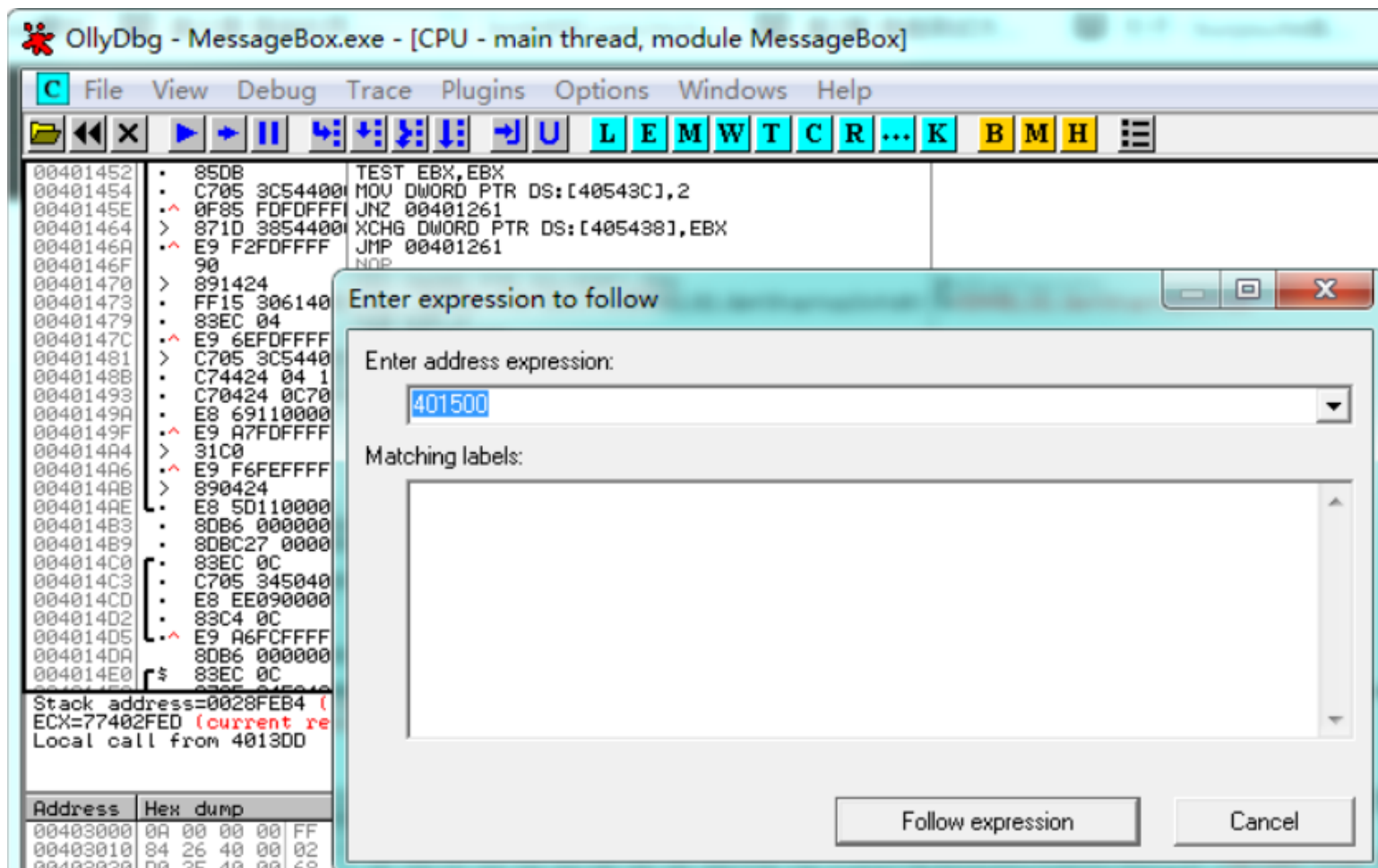
OllyDbg工具的使用 - 找到程序真正入口

回到OD



OllyDbg工具的使用 - 找到程序真正入口

跳转到main函数



OllyDbg工具的使用 - 主函数运行过程

main函数处F2下断点，按F9运行

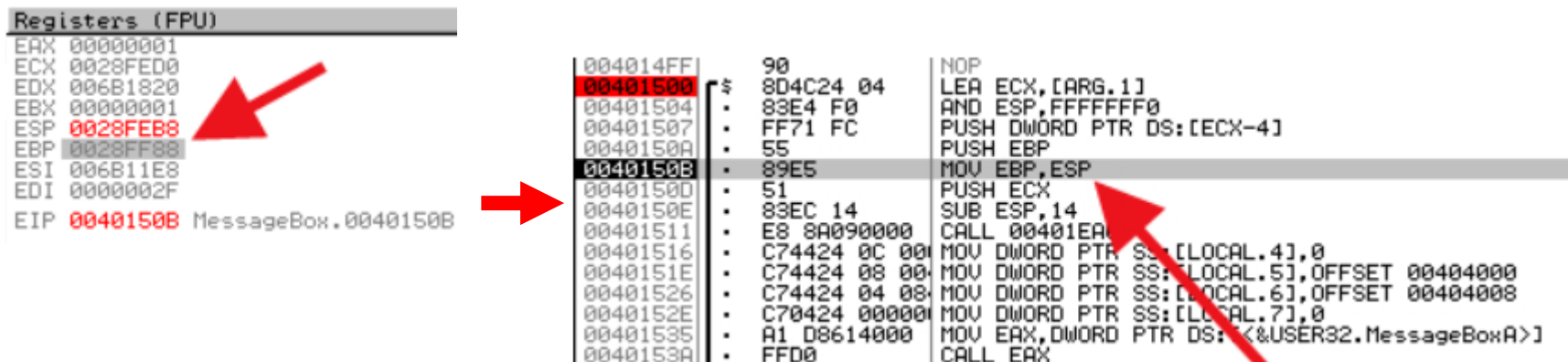
004014FD	90	NOP	
004014FE	90	NOP	
004014FF	90	NOP	
00401500	804C24 04	LEA ECX,[ARG.1]	
00401504	83E4 F0	AND ESP,FFFFFFF0	DQWORD (16.-byte) stack alignment
00401507	FF71 FC	PUSH DWORD PTR DS:[ECX-4]	
0040150A	55	PUSH EBP	
0040150D	89E5	MOV EBP,ESP	
00401510	51	PUSH ECX	
00401513	83EC 14	SUB ESP,14	
00401516	E8 8A090000	CALL 00401EA0	
00401519	C74424 0C 00	MOV DWORD PTR SS:[LOCAL.4],0	Type => MB_OK!MB_DEFBUTTON1!MB_APPLMODAL
0040151C	C74424 08 00	MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000	Caption => "warning"
0040151F	C74424 04 00	MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008	Text => "You have been hacked"
00401522	C74424 000000	MOV DWORD PTR SS:[LOCAL.7],0	hOwner => NULL
00401525	A1 D8614000	MOV EAX,DWORD PTR DS:[&USER32.MessageBoxA]	USER32.MessageBoxA
00401528	FFD0	CALL EAX	
0040152B	83EC 10	SUB ESP,10	
0040152E	B8 00000000	MOV EAX,0	

F8运行到0x40150B

004014FF	90	NOP	
00401500	804C24 04	LEA ECX,[ARG.1]	
00401504	83E4 F0	AND ESP,FFFFFFF0	
00401507	FF71 FC	PUSH DWORD PTR DS:[ECX-4]	
0040150A	55	PUSH EBP	
0040150D	89E5	MOV EBP,ESP	
00401510	51	PUSH ECX	
00401513	83EC 14	SUB ESP,14	
00401516	E8 8A090000	CALL 00401EA0	
00401519	C74424 0C 00	MOV DWORD PTR SS:[LOCAL.4],0	
0040151C	C74424 08 00	MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000	
0040151F	C74424 04 00	MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008	
00401522	C74424 000000	MOV DWORD PTR SS:[LOCAL.7],0	
00401525	A1 D8614000	MOV EAX,DWORD PTR DS:[&USER32.MessageBoxA]	
00401528	FFD0	CALL EAX	

OllyDbg工具的使用 - 栈

目前栈的情况，esp栈顶是0x28FEB8，ebp栈底是0x28FF88。



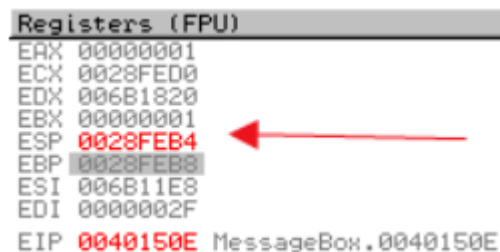
Registers (FPU)

EAX	00000001
ECX	0028FED0
EDX	006B1820
EBX	00000001
ESP	0028FEB8
EBP	0028FF88
ESI	006B11E8
EDI	0000002F
EIP	0040150B MessageBox.0040150B

Assembly Code:

004014FF	90	NOP
00401500	8D4C24 04	LEA ECX,[ARG.1]
00401504	83E4 F0	AND ESP,FFFFFFF0
00401507	FF71 FC	PUSH DWORD PTR DS:[ECX-4]
0040150A	55	PUSH EBP
0040150B	89E5	MOV EBP,ESP
0040150D	51	PUSH ECX
0040150E	83EC 14	SUB ESP,14
00401511	E8 8A090000	CALL 00401EA
00401516	C74424 0C 00	MOV DWORD PTR SS:[LOCAL.4],0
0040151E	C74424 08 00	MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000
00401526	C74424 04 00	MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008
0040152E	C74424 000000	MOV DWORD PTR SS:[LOCAL.7],0
00401535	A1 D8614000	MOV EAX,DWORD PTR DS:[&USER32.MessageBoxA]
0040153A	FFD0	CALL EAX

继续F8两次，现在的esp是0x28FEB4，这里指令SUB ESP,14是让esp直接减14的意思



Registers (FPU)

EAX	00000001
ECX	0028FED0
EDX	006B1820
EBX	00000001
ESP	0028FEB4
EBP	0028FF88
ESI	006B11E8
EDI	0000002F
EIP	0040150E MessageBox.0040150E

OllyDbg工具的使用 - 栈

$0x28FEB4 - 0x14 = 0x28FEA0$ ，因此执行完SUB指令后，栈顶（ESP）应当为0x28FEA0



OllyDbg工具的使用 - 栈

F8之后，栈顶为0x28FEA0

0028FEA0	00000000		
0028FEA4	00405000	P@	
0028FEA8	0028FEC8	u=(
0028FEAC	00401E7B	(A@	RETURN from MessageBox.00401720 to MessageBox.00401E7B
0028FEB0	00401E20	A@	Entry point
0028FEB4	0028FED0	u=(
0028FEB8	0028FF88	e(
0028FEBc	004013E2	r!!@	RETURN from MessageBox.00401500 to MessageBox.004013E2
0028FEC0	006B1108	4k	
0028FEC4	0000002F	/	
0028FEC8	00000001	0	
0028FECC	004013E2	r!!@	RETURN from MessageBox.00401500 to MessageBox.004013E2
0028FED0	00000001	0	

0x28FEA0到0x28FEB4为SUB指令开辟出的空间

0028FEA0	00000000		
0028FEA4	00405000	P@	
0028FEA8	0028FEC8	u=(
0028FEAC	00401E7B	(A@	RETURN from MessageBox.00401720 to MessageBox.00401E7B
0028FEB0	00401E20	A@	Entry point
0028FEB4	0028FED0	u=(
0028FEB8	0028FF88	e(
0028FEBc	004013E2	r!!@	RETURN from MessageBox.00401500 to MessageBox.004013E2
0028FEC0	006B1108	4k	
0028FEC4	0000002F	/	
0028FEC8	00000001	0	
0028FECC	004013E2	r!!@	RETURN from MessageBox.00401500 to MessageBox.004013E2
0028FED0	00000001	0	

开辟出来的
栈空间

OllyDbg工具的使用 - 栈

目前这5个地址的情况

0028FEA4	00405000	P@	
0028FEA8	0028FEC8	▬@	
0028FEAC	00401E7B	▬@	RETURN from MessageBox.00401720 to MessageBox.00401E7B
0028FEB0	00401E20	▬@	Entry point
0028FEB4	0028FED0	▬@	

继续F8，运行过CALL 00401EA0时发现这五个地址的值没有变化。运行过MOV DWORD PTR SS:[LOCAL.4],0时现0x28FEAC的值变成了0。

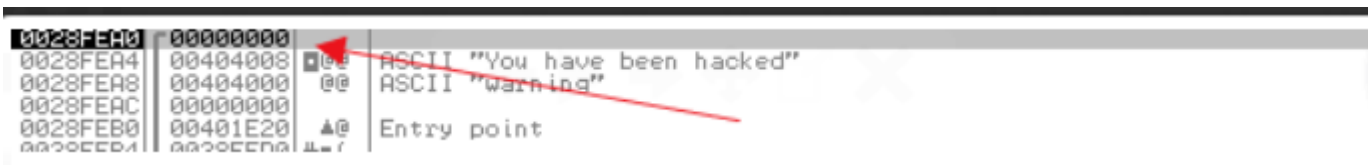
0028FEA4	00405000	P@	
0028FEA8	0028FEC8	▬@	
0028FEAC	00000000	▬@	
0028FEB0	00401E20	▬@	Entry point
0028FEB4	0028FED0	▬@	

011yDbg工具的使用 - 栈

运行过MOV DWORD PTR SS:[LOCAL.5],OFFSET 00404000，这时0x28FEA8的值变成了0x404000。

运行 MOV DWORD PTR SS:[LOCAL.6],OFFSET 00404008，这时0x28FEA4的值变成了0x404008。

运行 MOV DWORD PTR SS:[LOCAL.7],0，



0028FEA4	00000000		
0028FEA8	00404008	@@	ASCII "You have been hacked"
0028FEA8	00404000	@@	ASCII "Warning"
0028FEAC	00000000		
0028FEB0	00401E20	Δ@	Entry point
0028FEB4	0020CCE0	ΔΔ	

OllyDbg工具的使用 - 栈

0x404000和0x404008 是什么?

0028FEH0	00000000		
0028FEA4	00404008	00	ASCII "You have been hacked"
0028FEA8	00404000	00	ASCII "warning"
0028FEAC	00000000		
0028FEB0	00401E20	▲0	Entry point
0028FEB4	00000000	▲0	

验证一下，在左下角二进制窗口依次右键—>go to—>expression接着在弹出窗口中输入404000，查找

Address	Hex dump	ASCII
00404000	77 61 72 6E 69 6E 67 00 59 6F 75 20 68 61 76 65	warning You have
00404010	20 62 65 65 6E 20 68 61 63 68 65 64 00 00 00 00	been hacked
00404020	A0 15 40 00 55 6E 68 6E 6F 77 6E 20 65 72 6F 6F	â Unknown erro
00404030	72 00 00 00 5F 6D 61 74 68 65 72 72 28 29 3A 20	r _matherr():
00404040	25 73 20 69 6E 20 25 73 28 25 67 2C 20 25 67 29	%s in %s(%g, %g)
00404050	20 20 28 72 65 74 76 61 6C 3D 25 67 29 0A 00 00	(retval=%g)â
00404060	41 72 67 75 6D 65 6E 74 20 64 6F 6D 61 69 6E 20	Argument domain
00404070	65 72 72 6F 72 20 28 44 4F 4D 41 49 4E 29 00 41	error (DOMAIN) A
00404080	72 67 75 6D 65 6E 74 20 73 69 6E 67 75 6C 61 72	rgument singular
00404090	69 74 79 20 28 53 49 47 4E 29 00 00 4F 76 65 72	ity (SIGN) Over
004040A0	66 6C 6F 77 20 72 61 6E 67 65 20 65 72 72 6F 72	flow range error
004040B0	20 28 4F 56 45 52 46 4C 4F 57 29 00 54 68 65 20	(OVERFLOW) The

warning在线通过工具转换成ASC码的结果，刚好与上图前七个字节对上

\u0077\u0061\u0072\u006e\u0069\u006e\u0067

```
#include<windows.h>
int main()
{
    MessageBox(0,"You have been hacked","warning",0);
    return 0;
}
```

windbg工具的使用 - 工具介绍

- 什么是WinDbg

WinDbg是windows平台下对内核，应用程序，服务程序进行调试的工具。相对VS而言更加小巧，功能却比VS丰富，支持内核模式和用户模式的调试。

- WinDbg的下载

- WinDbg包含在Windows调试开发包当中
- <https://docs.microsoft.com/zh-cn/windows-hardware/drivers/download-the-wdk> (WDK)

windbg工具的使用 - 工具介绍

- WinDbg的调试模式
 - 内核模式 (Kernel-Mode) 与用户模式 (User-Mode)
- 内核模式(Kernel-Mode)
 - 为了不让程序任意存取资源，对应x86的ring0层，操作系统的核心部分，包括设备驱动程序都运行在该模式。
- 用户模式(User-Mode)
 - 当CPU运行于User Mode时，对应于x86的ring3层，操作系统的用户接口部分以及所有的用户应用程序都运行在该级别。

windbg工具的使用 - 符号表

- PDB文件

- 链接器自动生成
- 文件由两个部分构成，私有符号数据 (*private symbol data*) 和公共符号表 (*public symbol table*)

- 私有符号数据 (Private Symbol Data)

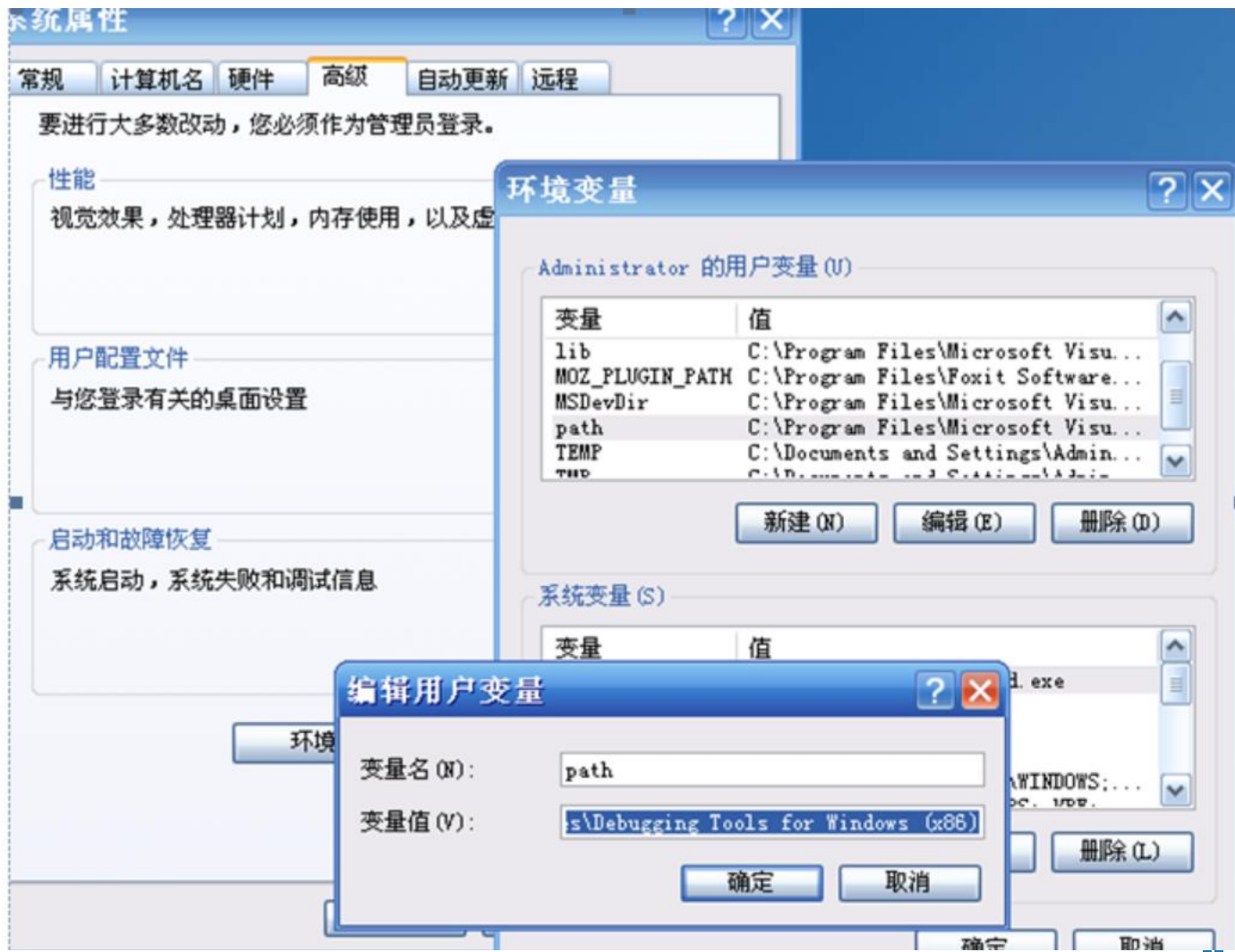
- 函数
- 全局变量
- 局部变量
- 用户定义的结构体，类，数据类型
- 源文件的名称和源文件中每个二进制指令的行号

- 公共符号表(Public Symbol Table)

- 静态函数
- 全局变量(extern)

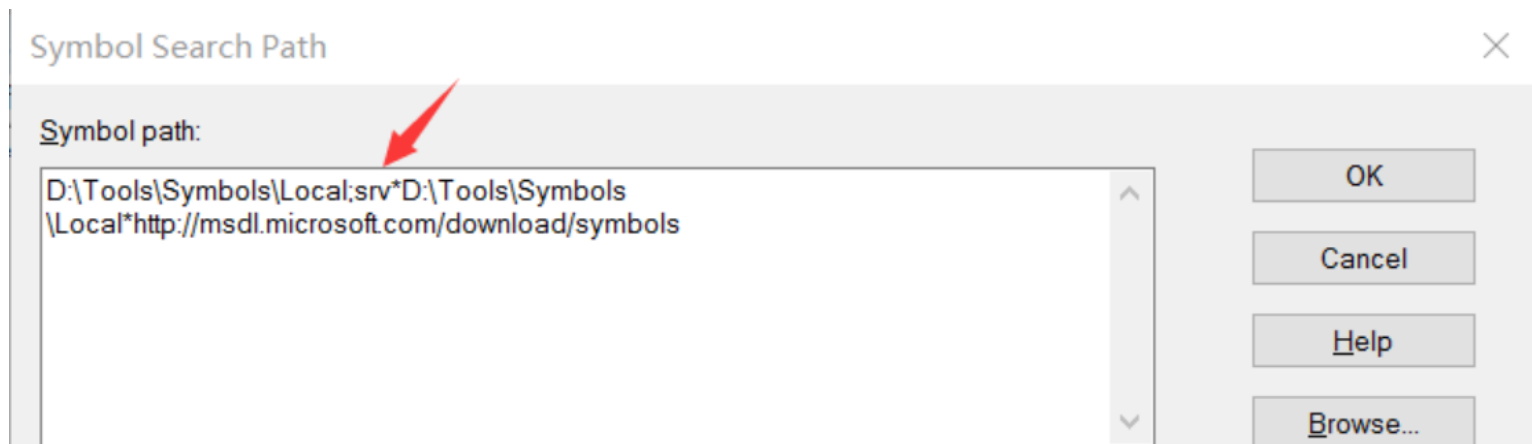
windbg工具的使用 - 内核调试环境 - windbg+vmware

增加环境变量



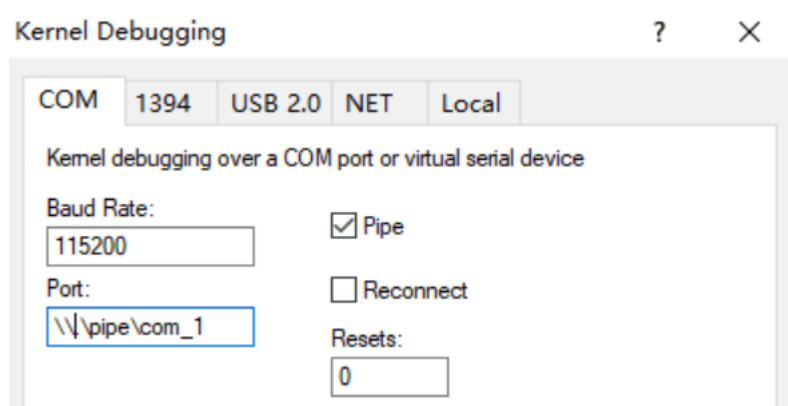
windbg工具的使用 - 内核调试环境 - windbg+vmware

WinDbg,按下Ctrl + S按键（链接如失效，需手动下载对应版本）



windbg工具的使用 - 内核调试环境 - windbg+vmware

- 1.启动WinDbg
- 2.选择File->Kernel Debug
- 3.单击COM标签



windbg工具的使用 - 内核调试环境 - windbg+vmware

增加串口



输出到命名管道



管道设置














windbg工具的使用 - 内核调试环境 - windbg+vmware

直观效果

虚拟机设置

硬件

选项

设备	摘要
 内存	512 MB
 处理器	1
 硬盘(SCSI)	40 GB
 CD/DVD (IDE)	自动检测
 软盘	自动检测
 网络适配器	NAT
 USB 控制器	存在
 声卡	自动检测
 打印机	存在
 串行端口 2	正在使用命名管道 \\.\pipe\com_1
 显示器	自动检测

设备状态

- ☐ 已连接(C)
- ☒ 启动时连接(O)

连接

☐ 使用物理串行端口(U):

COM1

☐ 使用输出文件(S):

浏览(B)...

☒ 使用命名的管道(N):

\\.\pipe\com_1

该端是服务器。

另一端是应用程序。

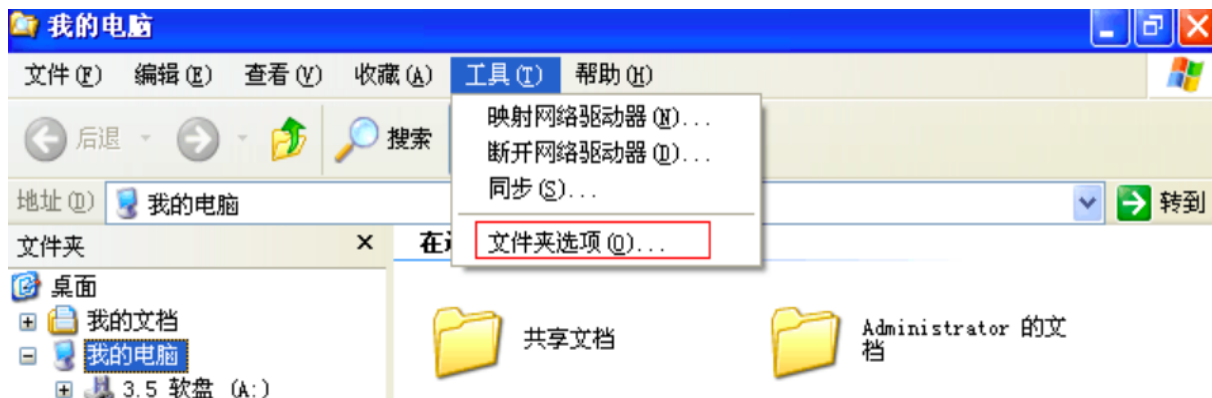
I/O 模式

☒ 轮询时主动放弃 CPU(Y)

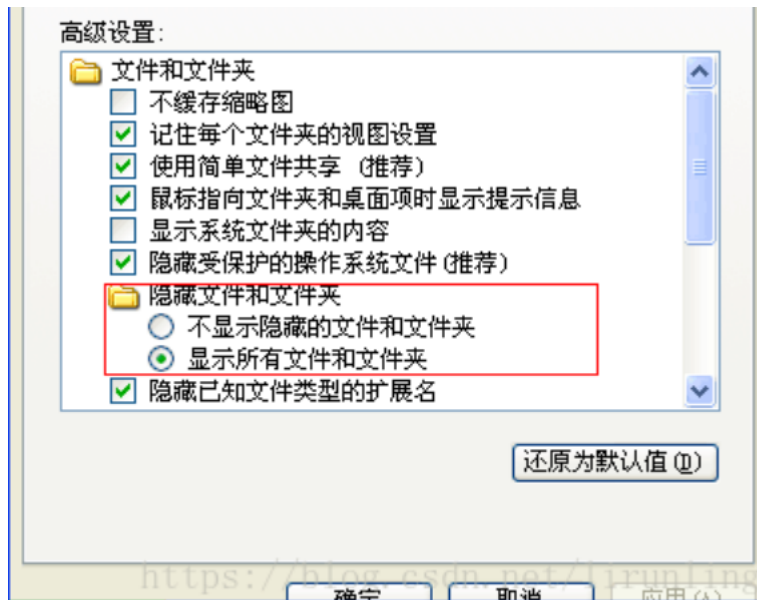
允许客户机操作系统在轮询模式下(而不是中断模式)使用此串行端口。

windbg工具的使用 - 内核调试环境 - 镜像修改

文件夹选项



打开隐藏文件夹



windbg工具的使用 - 内核调试 - (xp)修改Boot.ini

找到Boot.ini

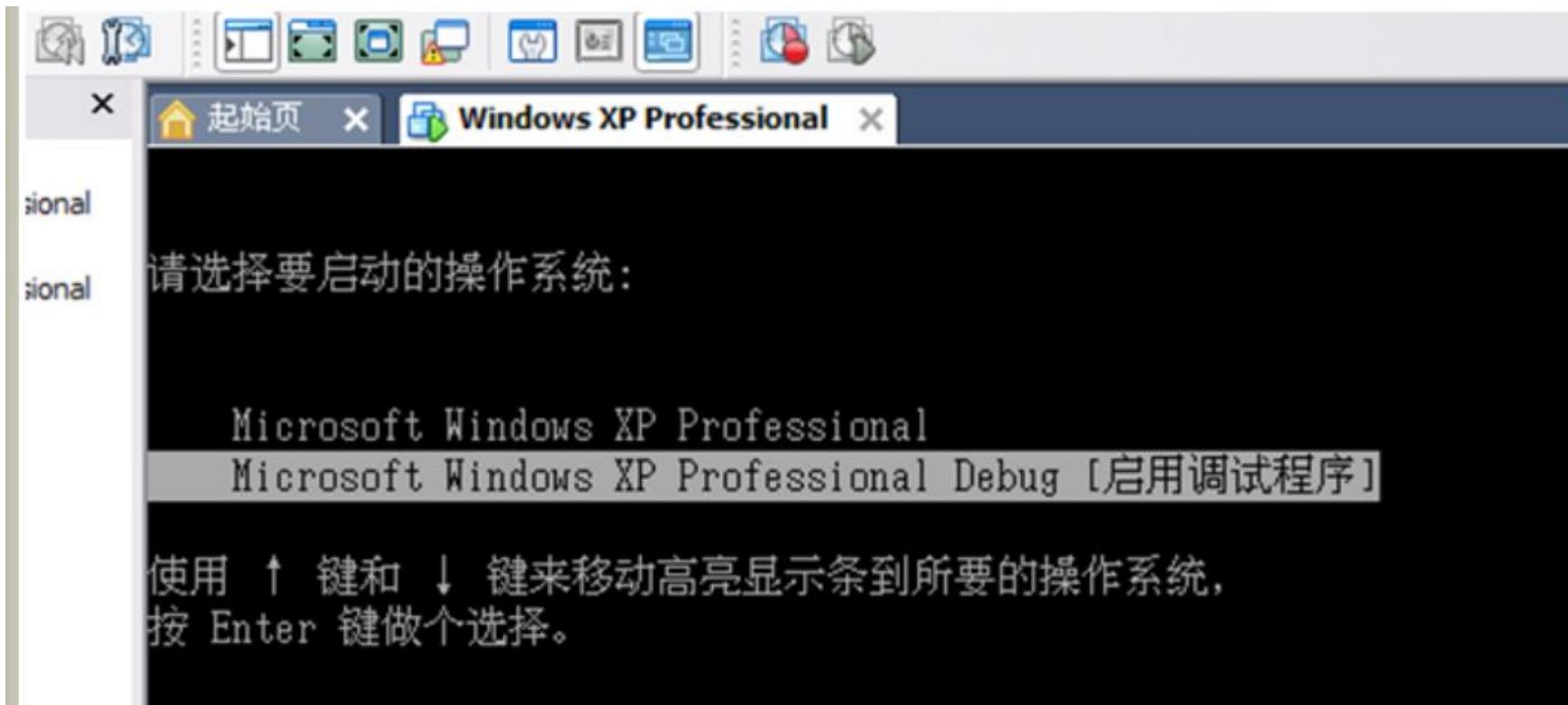


进行调试修改

```
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(1)\WINDOWS
[operating systems]
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional"
/noexecute=optin /fastdetect /noexecute=alwaysoff
multi(0)disk(0)rdisk(0)partition(1)\WINDOWS="Microsoft Windows XP Professional Debug"
/fastdetect /debug /debugport=com /baudrate=115200
```

windbg工具的使用 - 内核调试环境 - 启动目标

启动对应支持调试的OS



windbg工具的使用 - 内核调试环境 - win7/10

管理员权限的CMD

```
管理员: C:\Windows\System32\cmd.exe

Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。

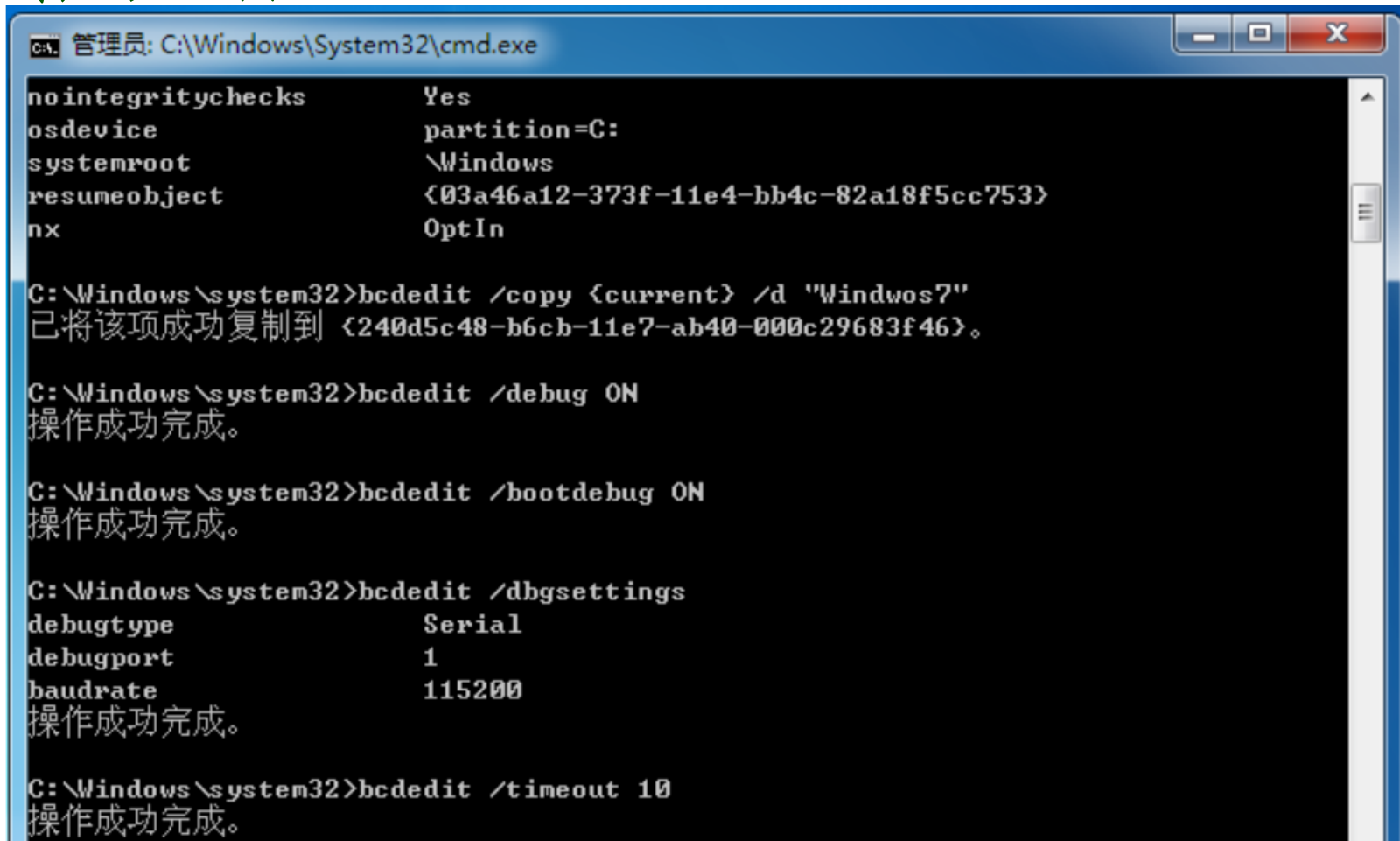
C:\Windows\system32>bcdedit

Windows 启动管理器
-----
标识符                {bootmgr}
device                partition=C:
description            Windows Boot Manager
locale                zh-CN
inherit                {globalsettings}
default                {current}
resumeobject           {03a46a12-373f-11e4-bb4c-82a18f5cc753}
displayorder           {current}
toolsdisplayorder      {memdiag}
timeout                0

Windows 启动加载器
-----
标识符                {current}
device                partition=C:
path                  \Windows\system32\winload.exe
```

windbg工具的使用 - 内核调试环境 - win7/10

对应设置调整：



```
管理员: C:\Windows\System32\cmd.exe

nointegritychecks      Yes
osdevice               partition=C:
systemroot             \Windows
resumeobject           {03a46a12-373f-11e4-bb4c-82a18f5cc753}
nx                     OptIn

C:\Windows\system32>bcdedit /copy {current} /d "Windwos?"
已将该项成功复制到 {240d5c48-b6cb-11e7-ab40-000c29683f46}。

C:\Windows\system32>bcdedit /debug ON
操作成功完成。

C:\Windows\system32>bcdedit /bootdebug ON
操作成功完成。

C:\Windows\system32>bcdedit /dbgsettings
debugtype              Serial
debugport              1
baudrate               115200
操作成功完成。

C:\Windows\system32>bcdedit /timeout 10
操作成功完成。
```

windbg工具的使用 - 内核调试环境 - 连接成功

```
Microsoft (R) Windows Debugger Version 6.11.0001.404 X86
Copyright (c) Microsoft Corporation. All rights reserved.

Opened \\.\pipe\com1
Waiting to reconnect...
Connected to Windows XP 2600 x86 compatible target at (Wed Jan 2 10:16:55.515 2013 (GMT+8)), ptr64 FALSE
Kernel Debugger connection established. (Initial Breakpoint requested)
Symbol search path is: H:\Cache\symbols
Executable search path is:
*** ERROR: Symbol file could not be found. Defaulted to export symbols for ntoskrnl.exe -
Windows XP Kernel Version 2600 (Service Pack 3) UP Free x86 compatible
Product: WinNt, suite: TerminalServer SingleUserTS
Built by: 2600.xpsp_sp3_gdr.111025-1629
Machine Name:
Kernel base = 0x804d8000 PsLoadedModuleList = 0x8055c240
Debug session time: Wed Jan 2 10:16:47.259 2013 (GMT+8)
System Uptime: 0 days 0:06:24.082
Break instruction exception - code 80000003 (first chance)
*****
*
* You are seeing this message because you pressed either
*   CTRL+C (if you run kd.exe) or,
*   CTRL+BREAK (if you run WinDBG),
* on your debugger machine's keyboard.
*
*   THIS IS NOT A BUG OR A SYSTEM CRASH
*
* If you did not intend to break into the debugger, press the "g" key, then
* press the "Enter" key now. This message might immediately reappear. If it
* does, press "g" and "Enter" again.
*
*****
nt!DbgBreakPointWithStatus+0x4:
804e45a2 cc      int     3
```

正在通过串口连接到虚拟机

已经成功地连接到虚拟机

符号文件路径, 要设置正确

虚拟机已经处于被调试状态了

windbg工具的使用 - 环境搭建总结

注意：

1. 细节：名称、空格、斜杠与反斜杠
2. 端口的名称一定要统一，考虑端口占用的问题
3. 原理决定逻辑，动手+思考
4. 别人行你不一定行，不要开口就问，自己先纠结一下
5. 确实存在人品现象
6. 百思不得其解的痛苦：从头开始

感谢大家！



南京邮电大学
Nanjing University of Posts and Telecommunications