

软件安全—恶意代码机理与防护

C3 PE文件格式

彭国军 教授

武汉大学国家网络安全学院

guojpeng@whu.edu.cn

本讲的内容提纲

3.1 PE文件及其表现形式

3.2 PE文件格式与恶意软件的关系

3.3 PE文件格式总体结构

3.4 代码节与数据节

3.5 引入函数节：PE文件的引入函数机制

3.6 引出函数节：DLL文件的函数引出机制

3.7 资源节：文件资源索引、定位与修改

3.8 重定位节：镜像地址改变后的地址自动修正

3.3 PE文件格式总体结构

□ test.exe

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
00000000h:	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	; MZ?..... ..
00000010h:	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00	; ?.....@.....
00000020h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000030h:	00	00	00	00	00	00	00	00	00	00	00	00	B0	00	00	00	;
00000040h:	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	68	; ...???L?Th
00000050h:	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6F	; is program canno
00000060h:	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20	; t be run in DOS
00000070h:	6D	6F	64	65	2E	0D	0D	0A	24	00	00	00	00	00	00	00	; mode.....\$.....
00000080h:	5D	65	FD	C8	19	04	93	9B	19	04	93	9B	19	04	93	9B	; je ..揸..揸..揸
00000090h:	97	1B	80	9B	11	04	93	9B	E5	24	81	9B	18	04	93	9B	; ?e?.揸?仁..揸
000000a0h:	52	69	63	68	19	04	93	9B	00	00	00	00	00	00	00	00	; Rich..揸.....
000000b0h:	50	45	00	00	4C	01	03	00	9B	4D	8F	42	00	00	00	00	; PE..L...跋厦...
000000c0h:	00	00	00	00	E0	00	0F	01	0B	01	05	0C	00	02	00	00	;?
000000d0h:	00	04	00	00	00	00	00	00	10	00	00	00	10	00	00	00	;
000000e0h:	00	20	00	00	00	00	40	00	10	00	00	00	02	00	00	00	;@.....
000000f0h:	04	00	00	00	00	00	00	00	04	00	00	00	00	00	00	00	;
00000100h:	00	40	00	00	00	04	00	00	00	00	00	00	02	00	00	00	; .@.....
00000110h:	00	00	10	00	00	10	00	00	00	10	00	00	10	00	00	00	;
00000120h:	00	00	00	00	10	00	00	00	00	00	00	00	00	00	00	00	;
00000130h:	14	20	00	00	3C	00	00	00	00	00	00	00	00	00	00	00	; ...<.....
00000140h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000150h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000160h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000170h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
00000180h:	00	00	00	00	00	00	00	00	00	20	00	00	14	00	00	00	;
00000190h:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	;
000001a0h:	00	00	00	00	00	00	00	00	2E	74	65	78	74	00	00	00	;text...
000001b0h:	46	00	00	00	00	10	00	00	00	02	00	00	00	04	00	00	; F.....
000001c0h:	00	00	00	00	00	00	00	00	00	00	00	00	20	00	00	60	;
000001d0h:	2E	72	64	61	74	61	00	00	A6	00	00	00	00	20	00	00	; .rdata...?....

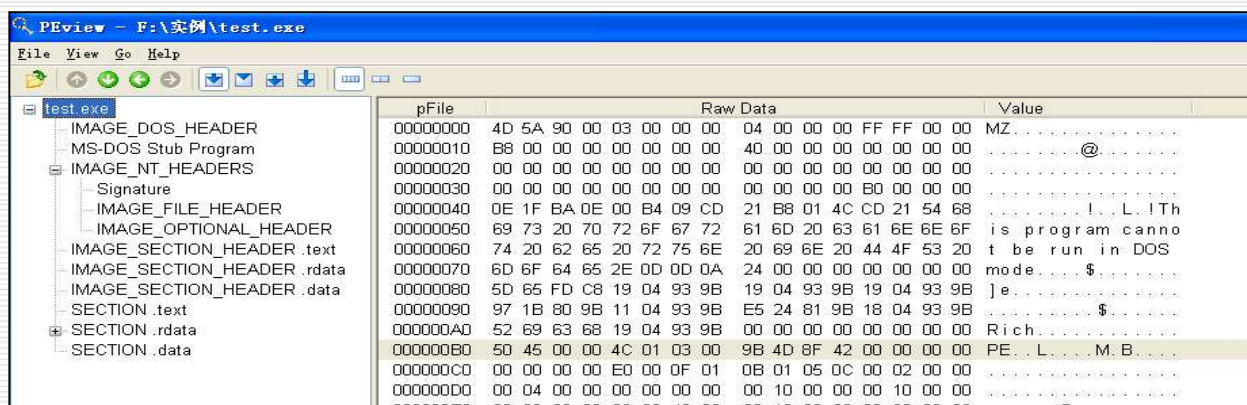


看雪学院工具集： <http://tools.pediy.com>



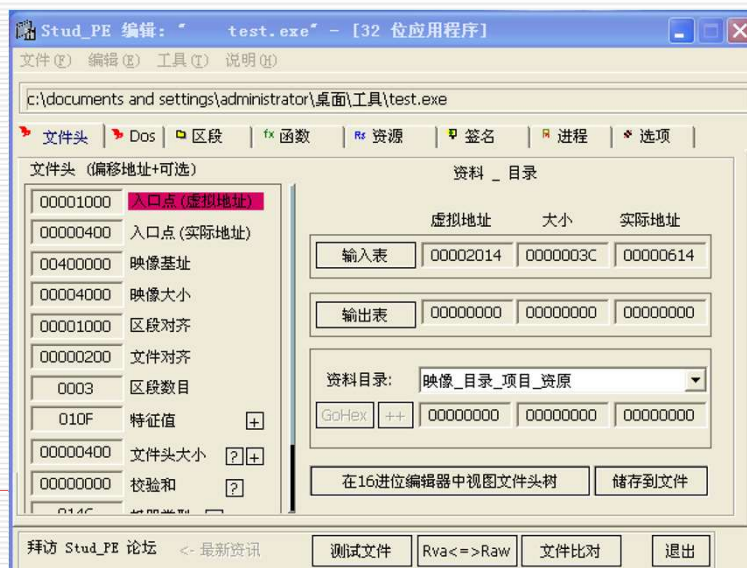
PE文件格式查看工具1-PEView

□ PEView: 可按照PE文件格式对目标文件的各字段进行详细解析。



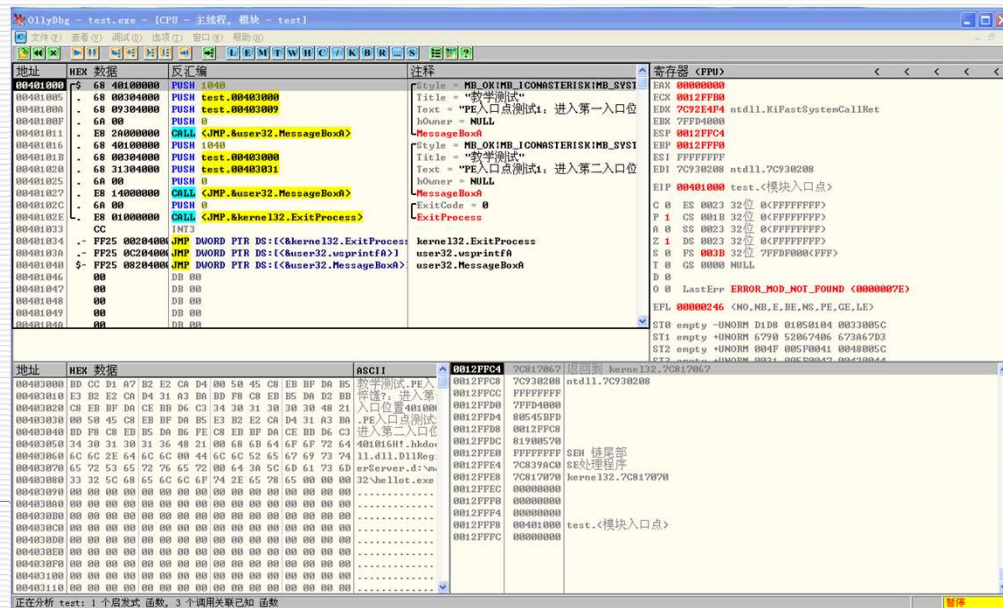
PE文件格式查看工具2-Stud_PE

□ Stud_PE：可按照PE文件格式对目标文件的各字段进行详细解析。



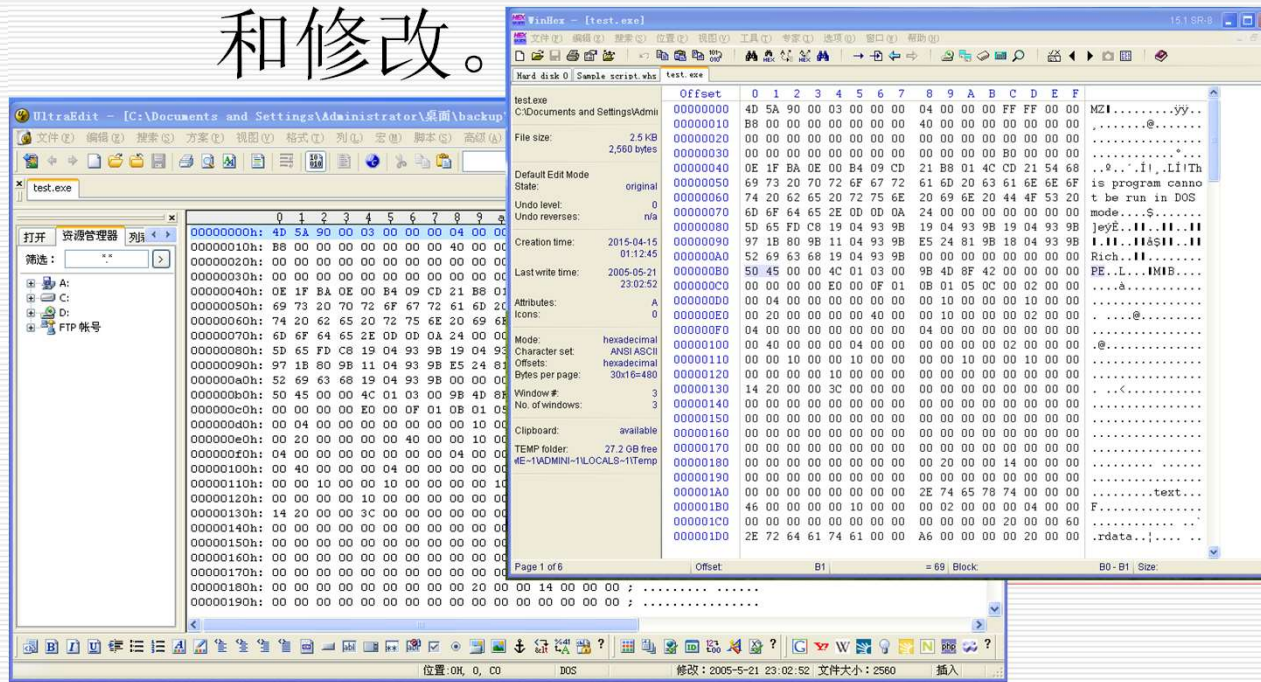
PE程序调试工具-Ollydbg

□ Ollydbg：可跟踪目标程序的执行过程，属于用户态调试工具。



16进制文件编辑工具-UltraEdit

□ UltraEdit: 可对目标文件进行16进制查看和修改。



PE文件格式总体结构

test.exe
... IMAGE_DOS_HEADER
... MS-DOS Stub Program
... IMAGE_NT_HEADERS
... Signature
... IMAGE_FILE_HEADER
... IMAGE_OPTIONAL_HEADER
... IMAGE_SECTION_HEADER .text
... IMAGE_SECTION_HEADER .rdata
... IMAGE_SECTION_HEADER .data
... SECTION .text
+ SECTION .rdata
... SECTION .data

1.DOS MZ header
2.DOS stub
3.PE header
4.Section table
5-1 Section 1
5-2 Section 2
Section ...
5.3 Section n

(1) MS-DOS MZ文件头(0x40)+ (2) DOS Stub

□ 作用:

- 定位PE文件头开始位置，也可用于PE文件合法性检测
- DOS下运行该程序时，将提示用户：“This Program cannot be run in DOS mode”！

```
00000000h: 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 ; MZ?..... ..
00000010h: B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00
00000020h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 B0 00 00 00 ; .....?.
00000040h: 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 ; ..?.???L?Th
00000050h: 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F ; is program canno
00000060h: 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 ; t be run in DOS
00000070h: 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 ; mode....$.
00000080h: 5D 65 FD C8 19 04 93 9B 19 04 93 9B 19 04 93 9B ; le
00000090h: 97 1B 80 9B 11 04 93 9B E5 24 81 9B 18 04 93 9B ; ?
000000a0h: 52 69 63 68 19 04 93 9B 00 00 00 00 00 00 00 00 ; Rich..搵.....
```

MZ文件头(0x40)

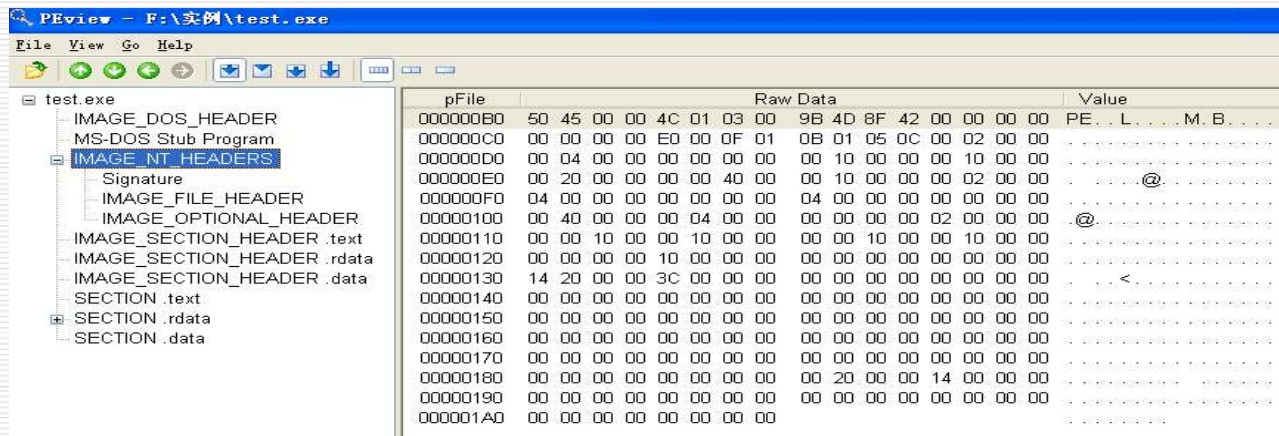
DOS Stub 搵

PEview - F:\实例\test.exe				
File View Go Help				
<div> <div>test.exe</div> <div> <div>IMAGE_DOS_HEADER</div> <div>MS-DOS Stub Program</div> <div>IMAGE_NT_HEADERS</div> <div>IMAGE_SECTION_HEADER .text</div> <div>IMAGE_SECTION_HEADER .rdata</div> <div>IMAGE_SECTION_HEADER .data</div> <div>SECTION .text</div> <div>SECTION .rdata</div> <div>SECTION .data</div> </div> </div>				
pFile	Data	Description	Value	
00000000	5A4D	Signature	IMAGE_DOS_SIGNATURE MZ	
00000002	0090	Bytes on Last Page of File		
00000004	0003	Number of Pages in File		
00000006	0000	Relocations		
00000008	0004	Size of Header in Paragraphs		
0000000A	0000	Minimum Extra Paragraphs		
0000000C	FFFF	Maximum Extra Paragraphs		
0000000E	0000	Initial Stack Segment SS		
00000010	00B8	Initial Stack Pointer SP		
00000012	0000	Checksum		
00000014	0000	Initial Instruction Pointer IP		
00000016	0000	Initial Code Segment CS		
00000018	0040	Offset to Relocation Table		
0000001A	0000	Overlay Number		
0000001C	0000	Reserved		
0000001E	0000	Reserved		
00000020	0000	Reserved		
00000022	0000	Reserved		
00000024	0000	OEM Identifier		
00000026	0000	OEM Information		
00000028	0000	Reserved		
0000002A	0000	Reserved		
0000002C	0000	Reserved		
0000002E	0000	Reserved		
00000030	0000	Reserved		
00000032	0000	Reserved		
00000034	0000	Reserved		
00000036	0000	Reserved		
00000038	0000	Reserved		
0000003A	0000	Reserved		
0000003C	000000B0	Offset to New EXE Header		

3C处的值: 000000B0
指向PE文件头开始位置

(3) PE header

- ❑ PE header 由三部分组成【始于000000B0】
 - 字符串“PE\0\0” (**Signature**)
 - 映像文件头 (**FileHeader**)
 - 可选映像头 (**OptionalHeader**)

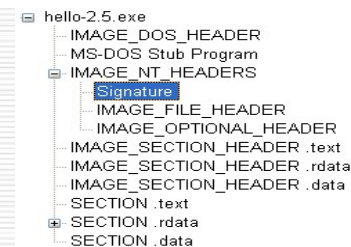


	pFile	Raw Data	Value
000000B0	50 45 00 00 4C 01 03 00	9B 4D 8F 42 00 00 00 00	PE...L...M.B....
000000C0	00 00 00 00 E0 00 0F 01	0B 01 05 0C 00 02 00 00	
000000D0	00 04 00 00 00 00 00 00	00 10 00 00 00 10 00 00	
000000E0	00 20 00 00 00 00 40 00	00 10 00 00 00 02 00 00	
000000F0	04 00 00 00 00 00 00 00	04 00 00 00 00 00 00 00	
00000100	00 40 00 00 00 04 00 00	00 00 00 00 02 00 00 00	
00000110	00 00 10 00 00 10 00 00	00 00 10 00 00 10 00 00	
00000120	00 00 00 00 10 00 00 00	00 00 00 00 00 00 00 00	
00000130	14 20 00 00 3C 00 00 00	00 00 00 00 00 00 00 00	
00000140	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000150	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000170	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
00000180	00 00 00 00 00 00 00 00	00 20 00 00 14 00 00 00	
00000190	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	
000001A0	00 00 00 00 00 00 00 00		

PE Header结构

```
000000b0h: 50 45 00 00 4C 01 03 00 9B 4D 8F 42 00 00 00 00 ; PE..L... 坂厦...
000000c0h: 00 00 00 00 E0 00 0F 01 0B 01 05 0C 00 02 00 00 ; ....?.
000000d0h: 00 04 00 00 00 00 00 00 00 10 00 00 00 10 00 00 ; .....
000000e0h: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00 ; .....@
000000f0h: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 ; .....
00000100h: 00 40 00 00 00 04 00 00 00 00 00 00 02 00 00 00 ; .@.....
00000110h: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 ; .....
00000120h: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000130h: 14 20 00 00 3C 00 00 00 00 00 00 00 00 00 00 00 ; . . <.....
00000140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000170h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000180h: 00 00 00 00 00 00 00 00 00 20 00 00 14 00 00 00 ; .....
00000190h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
000001a0h: 00 00 00 00 00 00 00 00
```


1) 字符串 “PE\0\0”



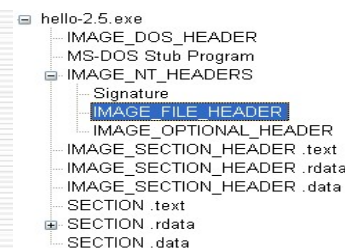
❑ **Signature** 一dword类型，值为50h, 45h, 00h, 00h（PE\0\0）。

■ 本域为PE标记，可以此识别给定文件是否为有效PE文件。

PE\0\0

```
000000b0h: 50 45 00 00 4C 01 03 00 9B 4D 8F 42 00 00 00 00 ; PE..L... 坂廈...
000000c0h: 00 00 00 00 E0 00 0F 01 0B 01 05 0C 00 02 00 00 ; ....?.....
```


2) 映像文件头 (0x14)



- 该结构域包含了关于PE文件物理分布的信息，比如节数目、后续可选文件头大小、机器类型等。

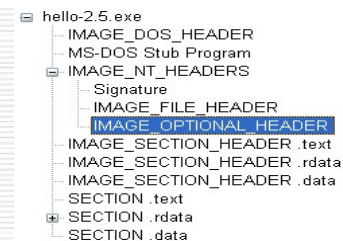
	X86	3个节	
000000b0h:	50 45 00 00	4C 01 03 00	9B 4D 8F 42 00 00 00 00 ; PE..L... 沝度....
000000c0h:	00 00 00 00	E0 00 0F 01	0B 01 05 0C 00 02 00 00 ;?.....
	可选 文件头 大小		

映像文件头的结构

顺序	名字	大小 (字节)	描述
1	Machine *	2	机器类型, x86为14CH
2	NumberOfSection **	2	文件中节的个数
3	TimeDataStamp	4	生成该文件的时间
4	PointerToSymbleTable	4	COFF符号表的偏移
5	NumberOfSymbols	4	符号数目
6	SizeOfOptionalHeader*	2	可选头的大小
7	Characteristics *	2	关于文件信息的标记, 比如文件是exe还是dll

```
000000b0h: 50 45 00 00 4C 01 03 00 9B 4D 8F 42 00 00 00 00 ; PE..L... 汧廔...
000000c0h: 00 00 00 00 E0 00 0F 01 0B 01 05 0C 00 02 00 00 ; ....?.....
```

3) 可选文件头



- ❑ 定义了PE文件的很多关键信息
 - ❑ 内存镜像加载地址（ImageBase）
 - ❑ 程序入口点（代码从哪里开始执行？）
 - ❑ 节在文件和内存中的对齐粒度
 - ❑ 本程序在内存中的镜像大小、文件头大小等

```
000000b0h: 50 45 00 00 4C 01 03 00 9B 4D 8F 42 00 00 00 00 ; PE..L... 坂厦...
000000c0h: 00 00 00 00 E0 00 0F 01 0B 01 05 0C 00 02 00 00 ; ....?.....
000000d0h: 00 04 00 00 00 00 00 00 00 10 00 00 00 10 00 00 ; .....
000000e0h: 00 20 00 00 00 00 40 00 00 10 00 00 00 02 00 00 ; .....@.....
000000f0h: 04 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 ; .....
00000100h: 00 40 00 00 00 04 00 00 00 00 00 00 02 00 00 00 ; .@.....
00000110h: 00 00 10 00 00 10 00 00 00 00 10 00 00 10 00 00 ; .....
00000120h: 00 00 00 00 10 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000130h: 14 20 00 00 3C 00 00 00 00 00 00 00 00 00 00 00 ; ...<.....
00000140h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000150h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000160h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000170h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
00000180h: 00 00 00 00 00 00 00 00 00 20 00 00 14 00 00 00 ; .....
00000190h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; .....
000001a0h: 00 00 00 00 00 00 00 00 2E 74 65 78 74 00 00 00 ; .....text...
```

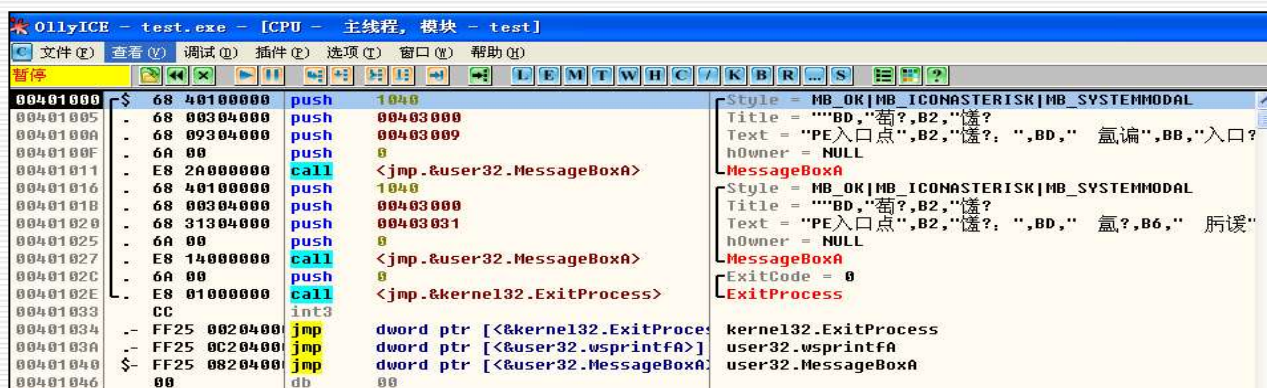
几个概念-1

□ ImageBase:

- PE文件在内存中的优先装载地址。

□ RVA地址:

- Relative Virtual Address, 相对虚拟地址, 它是相对内存中 ImageBase 的偏移位置。



The screenshot shows the OllyICE debugger interface. The main window displays assembly code for a module named 'test'. The code includes several push, call, and jmp instructions. The call stack on the right shows the following functions:

- Style = MB_OK|MB_ICONASTERISK|MB_SYSTEMMODAL
Title = ""BD,""萄?,B2,""透?
Text = "PE入口点",B2,""透?: ",BD,"" 氩谰",BB,"入口?
hOwner = NULL
MessageBoxA
- Style = MB_OK|MB_ICONASTERISK|MB_SYSTEMMODAL
Title = ""BD,""萄?,B2,""透?
Text = "PE入口点",B2,""透?: ",BD,"" 氩?,B6,"" 脔谰"
hOwner = NULL
MessageBoxA
- ExitCode = 0
ExitProcess

The assembly code in the main window is as follows:

```
00401000 68 40100000 push 1040
00401005 68 00304000 push 00403000
0040100A 68 00304000 push 00403009
0040100F 6A 00 push 0
00401011 E8 2A000000 call <jmp.&user32.MessageBoxA>
00401016 68 40100000 push 1040
0040101B 68 00304000 push 00403000
00401020 68 31304000 push 00403031
00401025 6A 00 push 0
00401027 E8 14000000 call <jmp.&user32.MessageBoxA>
0040102C 6A 00 push 0
0040102E E8 01000000 call <jmp.&kernel32.ExitProcess>
00401033 CC int3
00401034 FF25 00204000 jmp dword ptr [&kernel32.ExitProcess]
0040103A FF25 00204000 jmp dword ptr [&user32.wsprintfA]
00401040 FF25 00204000 jmp dword ptr [&user32.MessageBoxA]
00401046 00 db 00
```

几个概念-2

□ 对齐粒度

- 比喻：桶的容量为100升，现有367升水，请问需要使用多少个桶？

□ 问题：代码节的代码实际长度为0x46字节

- 文件中节对齐粒度为0x200,
- 内存中节对齐粒度为0x1000字节,

请问代码节在文件和内存中分别占用多少字节？

- 为什么PE文件中有很多“00”字节？
-

可选文件头中的一些关键字段

名字	描述
AddressOfEntryPoint * (位置D8H, 4字节)	PE装载器准备运行的PE文件的第一条指令的RVA。(病毒感染中通用关键字段)
ImageBase (位置: E4H, 4字节)	PE文件的优先装载地址。比如, 如果该值是400000h, PE装载器将尝试把文件装到虚拟地址空间的400000h处。
SectionAlignment (位置: E8H, 4字节)	内存中节对齐的粒度。
FileAlignment (位置: ECH, 4字节)	文件中节对齐的粒度。

PEview - F:\实例\test.exe

File View Go Help

test.exe

- IMAGE_DOS_HEADER
- MS-DOS Stub Program
- IMAGE_NT_HEADERS
 - Signature
 - IMAGE_FILE_HEADER
 - IMAGE_OPTIONAL_HEADER
- IMAGE_SECTION_HEADER .text
- IMAGE_SECTION_HEADER .rdata
- IMAGE_SECTION_HEADER .data
- SECTION .text
- SECTION .rdata
- SECTION .data

pFile	Data	Description	Value
000000C8	010B	Magic	
000000CA	05	Major Linker Version	
000000CB	0C	Minor Linker Version	
000000CC	00000200	Size of Code	
000000D0	00000400	Size of Initialized Data	
000000D4	00000000	Size of Uninitialized Data	
000000D8	00001000	Address of Entry Point	
000000DC	00001000	Base of Code	
000000E0	00002000	Base of Data	
000000F4	00400000	Image Base	
000000E8	00001000	Section Alignment	
000000EC	00000200	File Alignment	
000000F0	0004	Major O/S Version	
000000F2	0000	Minor O/S Version	
000000F4	0000	Major Image Version	
000000F6	0000	Minor Image Version	
000000F8	0004	Major Subsystem Version	
000000FA	0000	Minor Subsystem Version	
000000FC	00000000	Win32 Version Value	
00000100	00004000	Size of Image	
00000104	00000400	Size of Headers	
00000108	00000000	Checksum	
0000010C	0002	Subsystem	
0000010E	0000	DLL Characteristics	
00000110	00100000	Size of Stack Reserve	
00000114	00001000	Size of Stack Commit	
00000118	00100000	Size of Heap Reserve	
0000011C	00001000	Size of Heap Commit	
00000120	00000000	Loader Flags	
00000124	00000010	Number of Directories	

第一条指令在内存中的地址是多少?

- 401000H

=400000H+1000H

AddressOfEntryPoint的作用

□ 是否可以修改AddressOfEntryPoint指向任意代码？

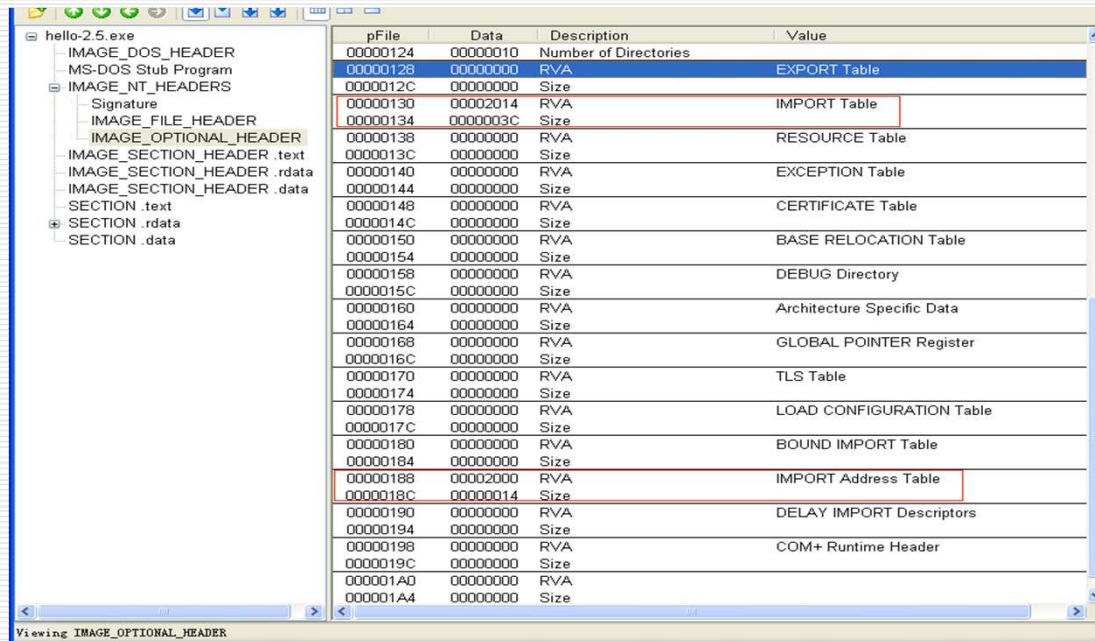
■ D8H处的4个字节

■ 演示：修改1000H为1016H

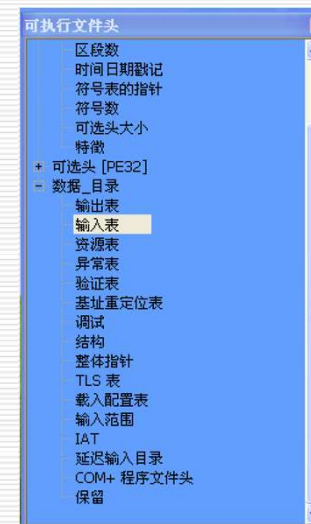
```
000000b0h: 50 45 00 00 4C 01 03 00 9B 4D 8F 42 00 00 00 00
000000c0h: 00 00 00 00 E0 00 0F 01 0B 01 05 0C 00 02 00 00
000000d0h: 00 04 00 00 00 00 00 00 00 10 00 00 00 10 00 00
```

指令入口点

Directory—16项 * 8字节



pFile	Data	Description	Value
00000124	00000010	Number of Directories	
00000128	00000000	RVA	EXPORT Table
0000012C	00000000	Size	
00000130	00002014	RVA	IMPORT Table
00000134	0000003C	Size	
00000138	00000000	RVA	RESOURCE Table
0000013C	00000000	Size	
00000140	00000000	RVA	EXCEPTION Table
00000144	00000000	Size	
00000148	00000000	RVA	CERTIFICATE Table
0000014C	00000000	Size	
00000150	00000000	RVA	BASE RELOCATION Table
00000154	00000000	Size	
00000158	00000000	RVA	DEBUG Directory
0000015C	00000000	Size	
00000160	00000000	RVA	Architecture Specific Data
00000164	00000000	Size	
00000168	00000000	RVA	GLOBAL POINTER Register
0000016C	00000000	Size	
00000170	00000000	RVA	TLS Table
00000174	00000000	Size	
00000178	00000000	RVA	LOAD CONFIGURATION Table
0000017C	00000000	Size	
00000180	00000000	RVA	BOUND IMPORT Table
00000184	00000000	Size	
00000188	00002000	RVA	IMPORT Address Table
0000018C	00000014	Size	
00000190	00000000	RVA	DELAY IMPORT Descriptors
00000194	00000000	Size	
00000198	00000000	RVA	COM+ Runtime Header
0000019C	00000000	Size	
000001A0	00000000	RVA	
000001A4	00000000	Size	



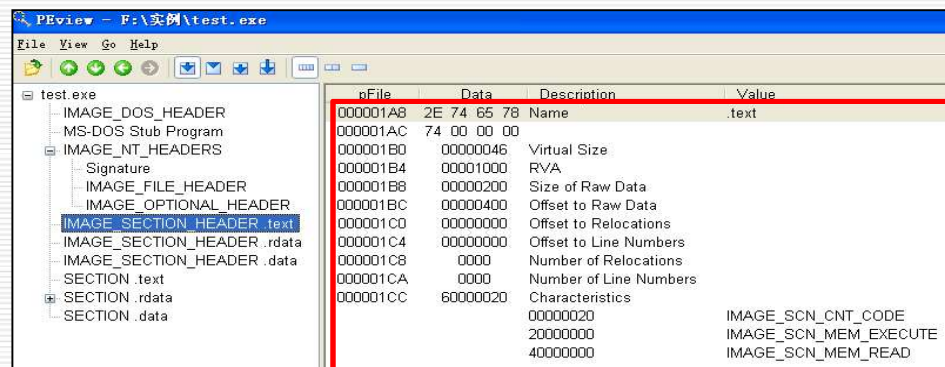
可执行文件头
区段数
时间/日期戳记
符号表的指针
符号数
可选头大小
特徵
+ 可选头 [PE32]
- 数据_目录
输出表
输入表
资源表
异常表
验证表
基址重定位表
调试
结构
整体指针
TLS 表
载入配置表
输入范围
IAT
延迟输入目录
COM+ 程序文件头
保留

(4) 节表

- 节表，是紧挨着 PE header 的一个结构数组。
 - 每一个节均对应一个节表项：
 - 节名；
 - 节在文件和内存中的开始地址；
 - 长度；
 - 节属性等。
-

test.exe的节表

```
000001a0h: 2E 74 65 78 74 00 00 00 : .....text...
000001b0h: 46 00 00 00 00 10 00 00 00 02 00 00 00 04 00 00 : F.....
000001c0h: 00 00 00 00 00 00 00 00 00 00 00 00 20 00 00 60 : .....
000001d0h: 2E 72 64 61 74 61 00 00 A6 00 00 00 00 20 00 00 : .rdata..?...
000001e0h: 00 02 00 00 00 06 00 00 00 00 00 00 00 00 00 00 : .....
000001f0h: 00 00 00 00 40 00 00 40 2E 64 61 74 61 00 00 00 : ....@..@.data...
00000200h: 58 00 00 00 00 30 00 00 00 02 00 00 00 08 00 00 : ?...0.....
00000210h: 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 C0 : .....@..?
```



Characteristics-节属性

0x00000020? 这个块包含代码。置位

0x00000040? 这个块包含已初始化的数据。

0x00000080? 这个块包含未初始化的数据（如 .bss 块）

0x00000200? 这个块包含注释或其它的信息。

0x00000800? 这个块的内容不应放进最终的EXE文件中。

0x02000000? 这个块可以被丢弃，因为一旦它被载入，其进程就不需要它。最通常的可丢弃块是基本重定位块（.reloc）。

0x10000000? 这个块是可共享的。

0x20000000? 这个块是可执行的。

0x40000000? 这个块是可读的。

0x80000000? 这个块是可写的。



(5) 节

- 可执行文件的核心部分。
 - PE文件一般都有多个“节”，比较常见的有：
 - 代码节
 - 数据节
 - 引入函数节
 - 资源节等(如图标)
 - 引出函数节（DLL文件中常见）
 - 重定位节（DLL文件中常见）
-