

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

3.5 引入函数节

□ 这个节一般名为.rdata。

■ 引入函数：是被程序调用但其执行代码又不在程序中的函数。

□ 这些函数位于一个或者多个DLL中，在调用者程序中只保留了函数信息，包括函数名及其驻留的DLL名等。

□ 动态链接库：例如kernel32.dll、user32.dll、gdi32.dll等。



A screenshot of a debugger's 'Data View' window. The window title is 'SECTION .rdata'. It contains a list of four entries, each preceded by a dotted line: 'IMPORT Address Table', 'IMPORT Directory Table', 'IMPORT Name Table', and 'IMPORT Hints/Names & DLL Names'. The first entry, 'IMPORT Address Table', is highlighted with a blue background.

引入函数节原始数据

[illegible]

引入函数节的属性

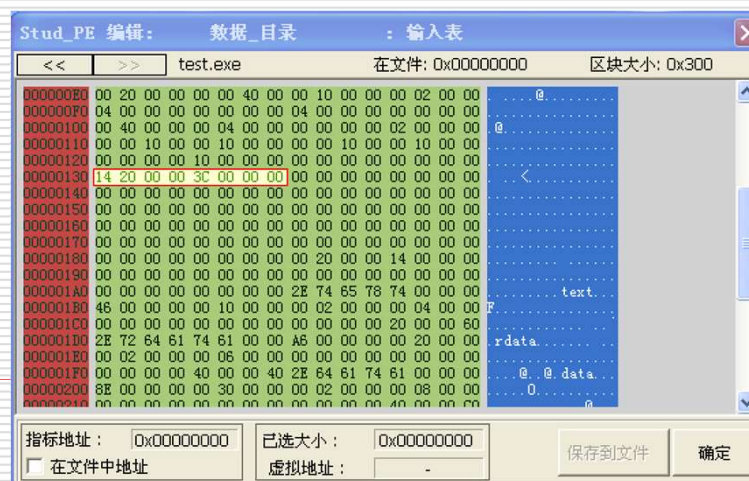
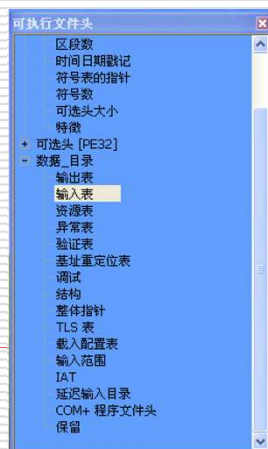


3.5.1 IMPORT Directory Table

SECTION .rdata
...IMPORT Address Table
...IMPORT Directory Table
...IMPORT Name Table
...IMPORT Hints/Names & DLL Name

□ 如何从PE文件定位到引入目录表（IDT）的起始位置？

■ PE可选文件头的DataDirectory。



3.5.1 IMPORT Directory Table



- 引入目录表由一系列的 **IMAGE_IMPORT_DESCRIPTOR** 结构组成。
 - 结构的数量取决于程序要使用的DLL文件的数量，每一个结构对应一个DLL文件。
 - 在所有这些结构的最后，由一个内容全为0的 **IMAGE_IMPORT_DESCRIPTOR** 结构作为结束。
-

IMPORT Directory Table

```
00000600h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d .....?..€...
00000610h: 00 00 00 00 50 20 00 00 00 00 00 00 00 00 00 00 : .....F.....
00000620h: 72 20 00 00 00 20 00 00 58 20 00 00 00 00 00 00 : .....X.....
00000630h: 00 00 00 00 9A 20 00 00 08 20 00 00 00 00 00 00 : .....?.....
00000640h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....
00000650h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d .....?..€...
00000660h: 00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 73 : .....€.ExitProces
00000670h: 73 00 68 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 : s.kernel32.dll..
00000680h: 62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D 85 : b.wsprintfA.?Me
00000690h: 73 73 61 67 85 42 6F 78 41 00 75 73 65 72 33 32 : ssageBoxA.user32
000006a0h: 2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00 : .dll.....
```

PEView

File View Go Help

hello-2.5.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
 - IMPORT Address Table
 - IMPORT Directory Table
 - IMPORT Name Table
 - IMPORT Hints/Names & DLL Nar
 - SECTION .data

pFile	Data	Description	Value
00000614	00002050	Import Name Table RVA	
00000618	00000000	Time Date Stamp	
0000061C	00000000	Forwarder Chain	
00000620	00002072	Name RVA	kernel32.dll
00000624	00002000	Import Address Table RVA	
00000628	00002058	Import Name Table RVA	
0000062C	00000000	Time Date Stamp	
00000630	00000000	Forwarder Chain	
00000634	0000209A	Name RVA	user32.dll
00000638	00002008	Import Address Table RVA	
0000063C	00000000		
00000640	00000000		
00000644	00000000		
00000648	00000000		
0000064C	00000000		

Viewing IMPORT Directory Table

IMAGE_IMPORT_DESCRIPTOR结构

pFile	Data	Description	Value
00000614	00002050	Import Name Table RVA	
00000618	00000000	Time Date Stamp	
0000061C	00000000	Forwarder Chain	
00000620	00002072	Name RVA	kernel32.dll
00000624	00002000	Import Address Table RVA	

IMAGE_IMPORT_DESCRIPTOR结构

IMAGE_IMPORT_DESCRIPTOR STRUCT

union

Characteristics dd ?

OriginalFirstThunk dd ?

//指向Import Name Table(该DLL中所有被引入函数的名字或序号信息)

Ends

TimeStamp dd ?

ForwarderChain dd ?

Name1 dd ? //dll文件名字字符串的RVA

FirstThunk dd ? //指向Import Address Table表

IMAGE_IMPORT_DESCRIPTOR ENDS

OriginalFirstThunk和FirstThunk

- 都指向一个包含一系列IMAGE_THUNK_DATA结构的数组。
 - 每个IMAGE_THUNK_DATA结构定义了一个导入函数的信息，
实际为一个双字，不同时刻可能代表不同含义。
 - 以一个内容为0的IMAGE_THUNK_DATA结构作为结束。

pFile	Data	Description	Value
00000650	00002064	Hint/Name RVA	0080 ExitProcess
00000654	00000000	End of Imports	kernel32.dll
00000658	0000208C	Hint/Name RVA	019D MessageBoxA
0000065C	00002080	Hint/Name RVA	0262 wsprintfA
00000660	00000000	End of Imports	user32.dll

←OriginalFirstThunk指向的数据

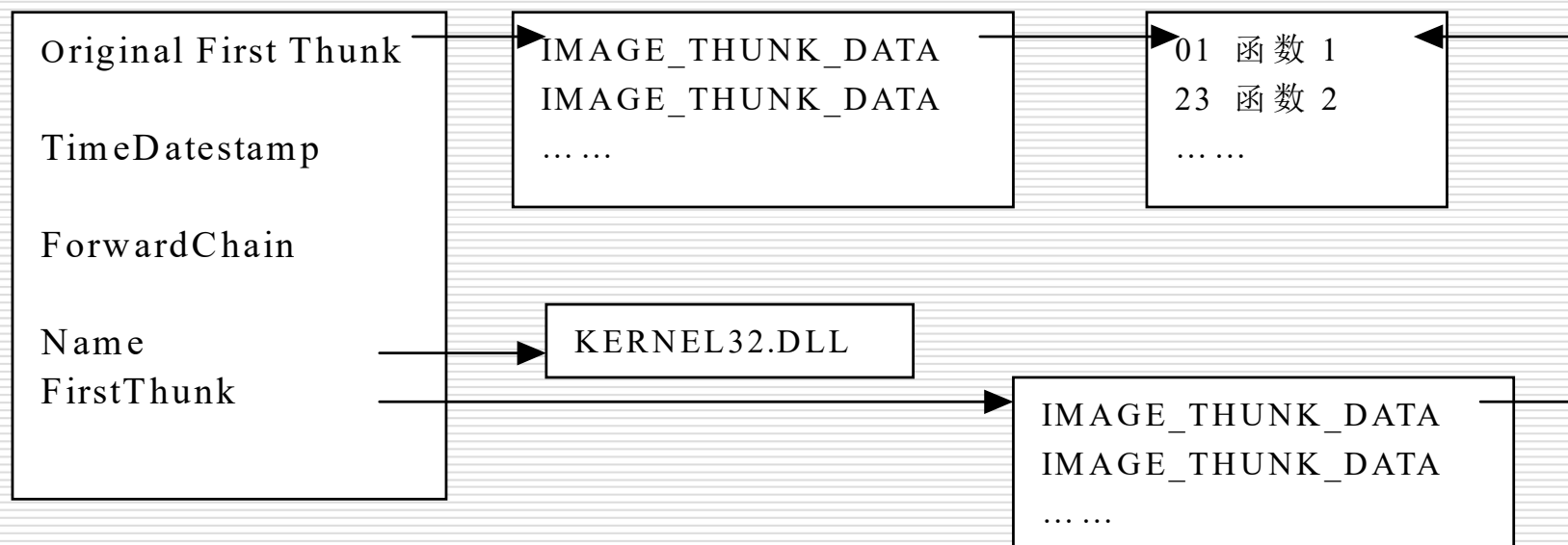
FirstThunk指向的数据→

pFile	Data	Description	Value
00000600	00002064	Hint/Name RVA	0080 ExitProcess
00000604	00000000	End of Imports	kernel32.dll
00000608	0000208C	Hint/Name RVA	019D MessageBoxA
0000060C	00002080	Hint/Name RVA	0262 wsprintfA
00000610	00000000	End of Imports	user32.dll

引入目录表（IDT）在文件中的指向

IMAGE_IMPORT_DESCRIPTOR

IMAGE_IMPORT_BY_NAME



IMAGE_THUNK_DATA结构

- 一个IMAGE_THUNK_DATA结构实际上就是一个双字，它在不同时刻有不同的含义。

IMAGE_THUNK_DATA STRUC

union u1

ForwarderString DWORD ? //指向一个转向者字符串的RVA

Function DWORD ? //被引入的函数的内存地址

Ordinal DWORD ? //被引入的API 的函数序号

AddressOfData DWORD ?

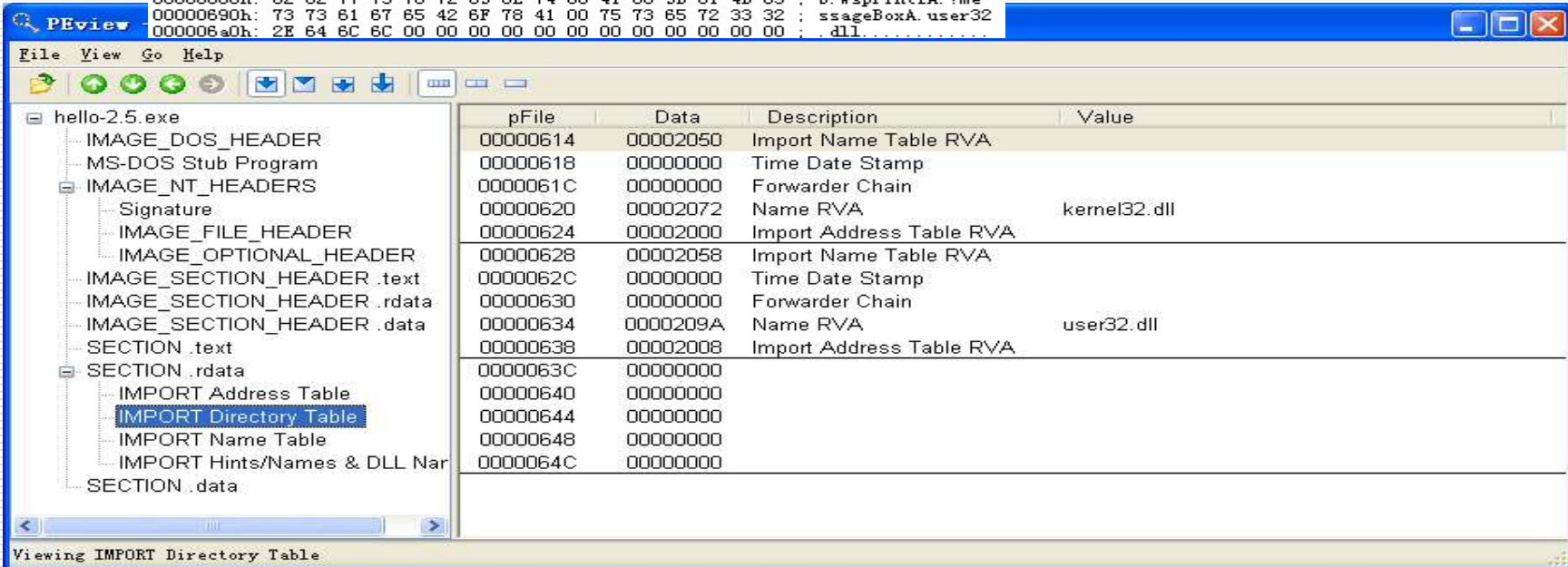
//被引入的API的Hint和函数名字字符串信息(IMAGE_IMPORT_BY_NAME结构)

ends

IMAGE_THUNK_DATA ENDS

IMPORT Directory Table

00000600h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d?..€..
00000610h: 00 00 00 00 50 20 00 00 00 00 00 00 00 00 00 00 :F.....
00000620h: 72 20 00 00 00 20 00 00 58 20 00 00 00 00 00 00 :X.....
00000630h: 00 00 00 00 9A 20 00 00 08 20 00 00 00 00 00 00 :?.....
00000640h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 :
00000650h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d?..€..
00000660h: 00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 73 :€.ExitProces
00000670h: 73 00 68 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 : s.kernel32.dll..
00000680h: 62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D 85 : b.wsprintfA.?Me
00000690h: 73 73 61 67 85 42 6F 78 41 00 75 73 65 72 33 32 : ssageBoxA.user32
000006a0h: 2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00 : .dll.....

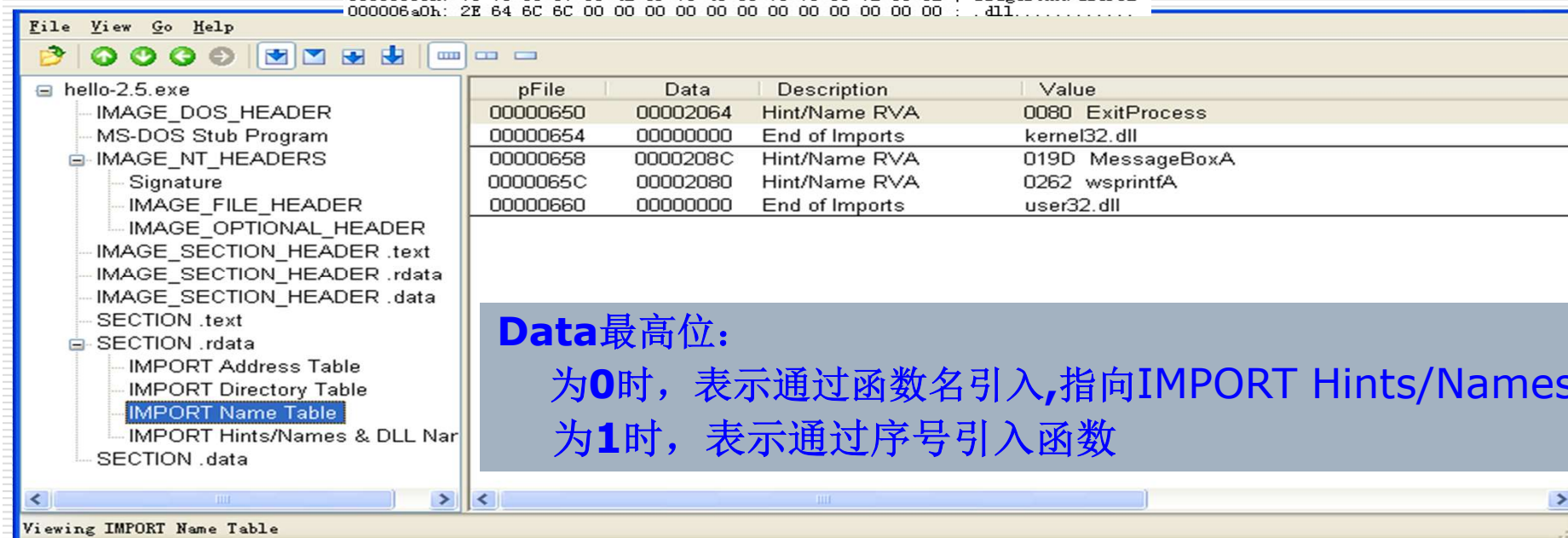


pFile	Data	Description	Value
00000614	00002050	Import Name Table RVA	
00000618	00000000	Time Date Stamp	
0000061C	00000000	Forwarder Chain	
00000620	00002072	Name RVA	kernel32.dll
00000624	00002000	Import Address Table RVA	
00000628	00002058	Import Name Table RVA	
0000062C	00000000	Time Date Stamp	
00000630	00000000	Forwarder Chain	
00000634	0000209A	Name RVA	user32.dll
00000638	00002008	Import Address Table RVA	
0000063C	00000000		
00000640	00000000		
00000644	00000000		
00000648	00000000		
0000064C	00000000		

Viewing IMPORT Directory Table

3.5.2 IMPORT Name Table

```
00000600h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d .....?..€..
00000610h: 00 00 00 00 50 20 00 00 00 00 00 00 00 00 00 00 : ....P.....
00000620h: 72 20 00 00 00 20 00 00 58 20 00 00 00 00 00 00 : r .....X.....
00000630h: 00 00 00 00 9A 20 00 00 08 20 00 00 00 00 00 00 : .....?.....
00000640h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....
00000650h: 84 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d .....?..€..
00000660h: 00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 73 : .....€..ExitProces
00000670h: 73 00 68 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 : s..kernel32.dll..
00000680h: 62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D 65 : b..wsprintfA.?Me
00000690h: 73 73 61 67 65 42 6F 78 41 00 75 73 65 72 33 32 : ssageBoxA.user32
000006a0h: 2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00 : .dll.....
```



The screenshot shows a debugger window with the file 'hello-2.5.exe' loaded. The left pane displays the file's structure, with 'IMPORT Name Table' selected under the 'SECTION .rdata' section. The right pane shows a table of imports:

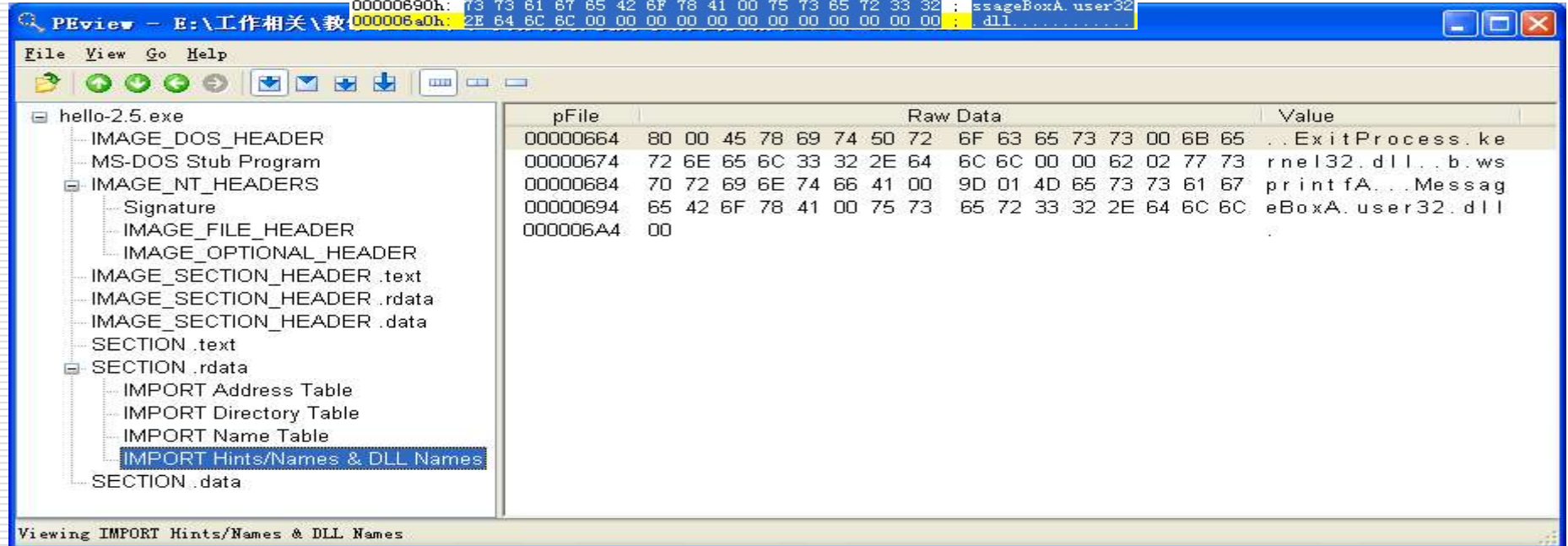
pFile	Data	Description	Value
00000650	00002064	Hint/Name RVA	0080 ExitProcess
00000654	00000000	End of Imports	kernel32.dll
00000658	0000208C	Hint/Name RVA	019D MessageBoxA
0000065C	00002080	Hint/Name RVA	0262 wsprintfA
00000660	00000000	End of Imports	user32.dll

Below the table, a blue box contains the following text:

Data最高位:
为**0**时, 表示通过函数名引入, 指向IMPORT Hints/Names
为**1**时, 表示通过序号引入函数

3.5.3 IMPORT Hints/Names & DLL names

```
00000600h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d .....?..€...
00000610h: 00 00 00 00 50 20 00 00 00 00 00 00 00 00 00 00 : ....P.....
00000620h: 72 20 00 00 00 20 00 00 58 20 00 00 00 00 00 00 : r ...X.....
00000630h: 00 00 00 00 9A 20 00 00 08 20 00 00 00 00 00 00 : ....?.....
00000640h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 : .....
00000650h: 64 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00 : d .....?..€...
00000660h: 00 00 00 00 30 00 45 78 89 74 50 72 8F 63 65 73 : ....€.ExitProces
00000670h: 73 00 68 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 : s.kernel32.dll..
00000680h: 62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D 65 : b.wsprintfA.?Me
00000690h: 73 73 61 67 65 42 6F 78 41 00 75 73 65 72 33 32 : ssageBoxA.user32
000006A0h: 2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00 : .dll.....
```



pFile	Raw Data	Value
00000664	80 00 45 78 69 74 50 72 6F 63 65 73 73 00 6B 65	..ExitProcess.ke
00000674	72 6E 65 6C 33 32 2E 64 6C 6C 00 00 62 02 77 73	rnal32.dll..b.ws
00000684	70 72 69 6E 74 66 41 00 9D 01 4D 65 73 73 61 67	printfA...Messag
00000694	65 42 6F 78 41 00 75 73 65 72 33 32 2E 64 6C 6C	eBoxA.user32.dll
000006A4	00	.

Viewing IMPORT Hints/Names & DLL Names

IMPORT Hints/Names & DLL names 分解说明

```
00000660h: 00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 73 : .....ExitProces
00000670h: 73 00 6B 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00 : s.kernel32.dll.
00000680h: 62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D 65 : b.wsprintfA.?Me
00000690h: 73 73 61 67 65 42 6F 78 41 00 75 73 65 72 33 32 : ssageBoxA.user32
000006a0h: 2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00 : .dll.....
```

- ❑ IMAGE_IMPORT_BY_NAME结构
 - 80 00为Hints, ExitProcess为引入函数名
 - 62 02 为Hints, wsprintfA为引入函数名
 - 9D 01 为Hints, MessageBoxA为引入函数名
- ❑ DLL names字符串
 - kernel32.dll 为dll文件名
 - user32.dll 为dll文件名

```
IMAGE_IMPORT_BY_NAME STRUCT
    Hint dw ?
    Name1 db ?
IMAGE_IMPORT_BY_NAME ENDS
```

3.5.4 IAT (IMPORT Address Table)



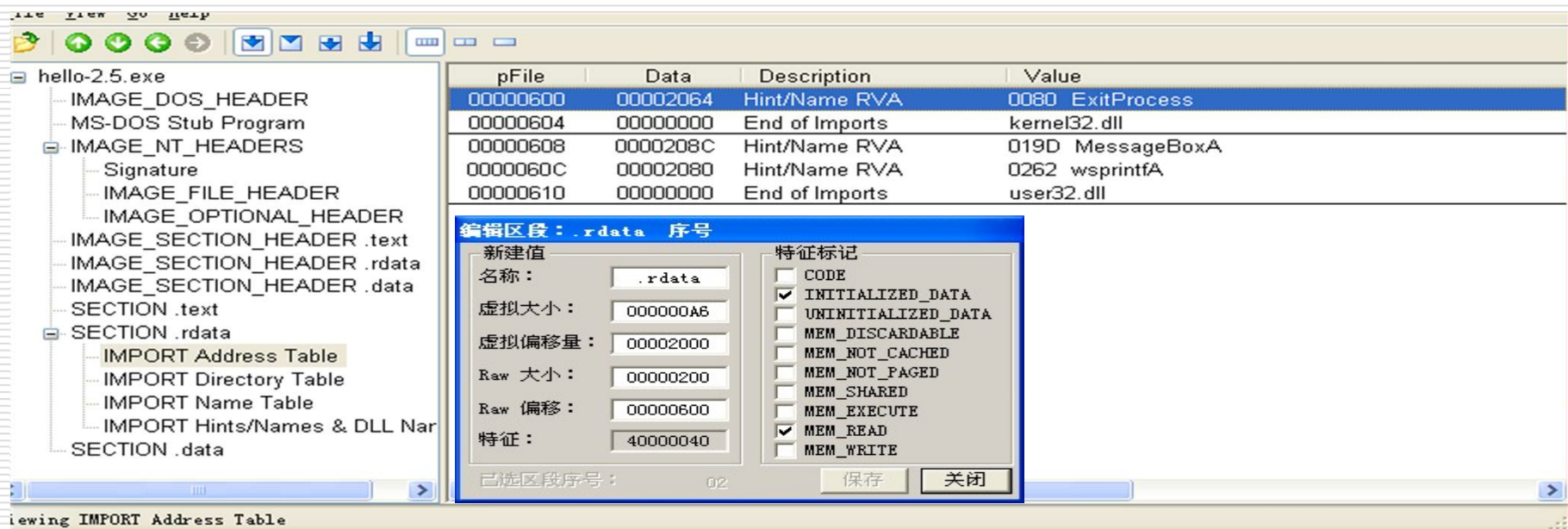
- 引入地址表：DWORD数组 [可通过可选文件头中的 **DataDirectory** 的第13项定位]
- 在文件中时，其内容与Import Name Table完全一样。
- 在内存中时，每个双字中存放着对应引入函数的地址。

00000600h:	84 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00	:	d?..€ ..
00000610h:	00 00 00 00 50 20 00 00 00 00 00 00 00 00 00 00	:P
00000620h:	72 20 00 00 00 20 00 00 58 20 00 00 00 00 00 00	:	rX
00000630h:	00 00 00 00 9A 20 00 00 08 20 00 00 00 00 00 00	:?.....
00000640h:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	:
00000650h:	84 20 00 00 00 00 00 00 8C 20 00 00 80 20 00 00	:	d?..€ ..
00000660h:	00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 73	:€.ExitProces
00000670h:	73 00 68 65 72 6E 65 6C 33 32 2E 64 6C 6C 00 00	:	s.kernel32.dll..
00000680h:	62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D 65	:	b.wsprintfA.?Me
00000690h:	73 73 61 67 65 42 6F 78 41 00 75 73 65 72 33 32	:	ssageBoxA.user32
000006a0h:	2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00 00 00	:	.dll.....

IAT (IMPORT Address Table)

00000600h:	54 20 00 00 00 00 00 8C 20 00 00 80 20 00 00	d . . . ? . € . .
00000610h:	00 00 00 00 50 20 00 00 00 00 00 00 00 00	. . . P
00000620h:	72 20 00 00 00 20 00 00 58 20 00 00 00 00	r X
00000630h:	00 00 00 00 9A 20 00 00 08 20 00 00 00 00 ?
00000640h:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 ?
00000650h:	64 20 00 00 00 00 00 8C 20 00 00 80 20 00	d ? . . € .
00000660h:	00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 € . ExitProce
00000670h:	73 00 6B 65 72 6E 65 6C 33 32 2E 64 6C 6C 00	s . kernel32 . dll . .
00000680h:	62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D	b . wsprintfA . ?Me
00000690h:	73 73 61 67 65 42 6F 78 41 00 75 73 65 72 33	ssageBoxA . user32
000006a0h:	2E 64 6C 6C 00 00 00 00 00 00 00 00 00 00	. dll

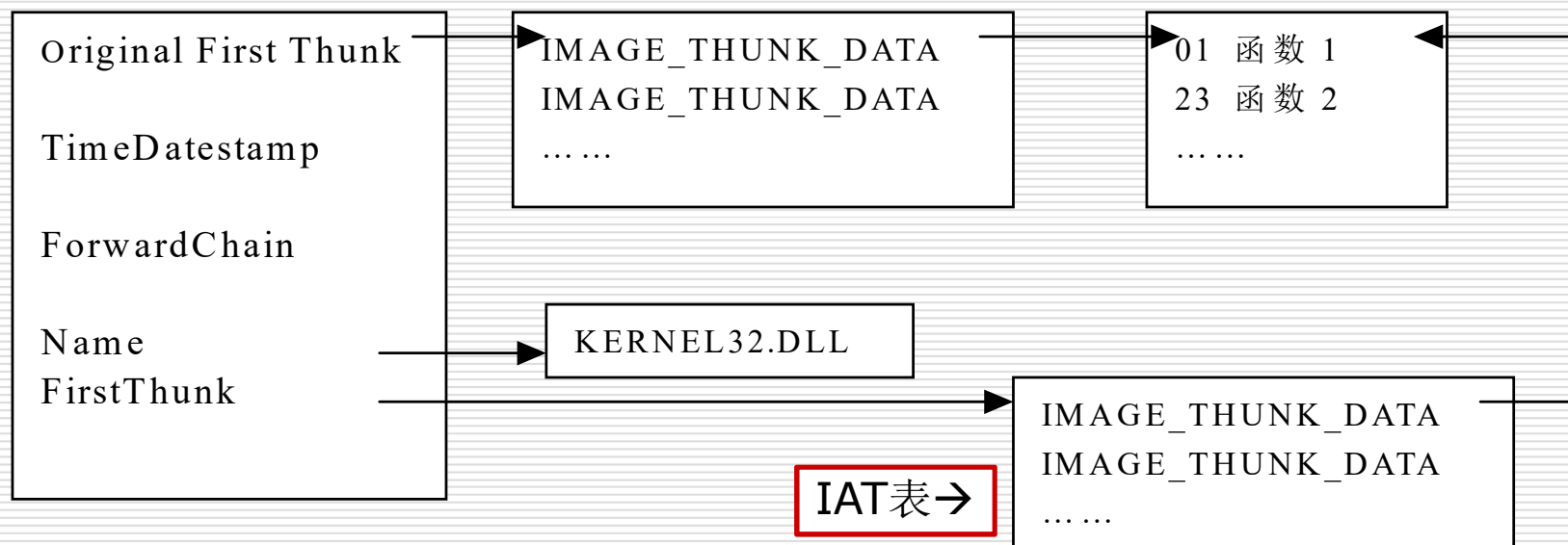
地址	HEX 数据	ASCII
00402000	DA CD 81 7C 00 00 00 00 EA 04 D5 77 AD A8 D1 77	前 ? 请
00402010	00 00 00 00 50 20 00 00 00 00 00 00 00 00 P
00402020	72 20 00 00 00 20 00 00 58 20 00 00 00 00	r X
00402030	00 00 00 00 9A 20 00 00 08 20 00 00 00 00 ?
00402040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 ?
00402050	64 20 00 00 00 00 00 8C 20 00 00 80 20 00	d ? . . € .
00402060	00 00 00 00 80 00 45 78 69 74 50 72 6F 63 65 € . Exit
00402070	73 00 6B 65 72 6E 65 6C 33 32 2E 64 6C 6C 00	s . kernel32
00402080	62 02 77 73 70 72 69 6E 74 66 41 00 9D 01 4D	b . wsprintf
00402090	73 73 61 67 65 42 6F 78 41 00 75 73 65 72 33	ssageBoxA .



在文件中的IAT表

IMAGE_IMPORT_DESCRIPTOR

IMAGE_IMPORT_BY_NAME



在内存中的IAT表

IAT表→