

# PE文件结构解析

20181656李涵威

1. 请解析USER32.DLL前5个导出函数的信息，要求列举AddressOfNams、AddressOfOrdinals、AddressOfFunctions的详细数据。

首先导入USER32.DLL到winhex和lordPE

找到USER32.DLL导出表位置

00081D80	00 00 00 00 74 99 24 58 00 00 00 00 C8 56 08 00	т\$X ÈV
00081D90	DC 05 00 00 EB 03 00 00 3E 03 00 00 A8 33 08 00	Ù ë > "3
00081DA0	54 43 08 00 4C 50 08 00 10 B2 02 00 30 AE 02 00	TC LP ² 0

可以得到关于导出表的一些信息：

名称	RVA
nName	0x000856C8
nBase	0x000005DC
NumberOfFunctions	0x000003E8
NumberOfNames	0x0000033E
AddressOfFunctions	0x000833A8
AddressOfNames	0x00084354
AddressOfNamesOrdinals	0x0008504C

1. AddressOfNamesOrdinals

选择先检查AddressOfNamesOrdinals，是因为可能会存在前几个函数并不具备名字的情况。

首先使用LordPE计算它的RAW，RAW=0x00083A4C

00083A40	2E 8E 08 00 38 8E 08 00 43 8E 08 00 02 00 03 00	.ž 8ž cž
00083A50	04 00 05 00 06 00 07 00 08 00 09 00 0A 00 0B 00	
00083A60	0C 00 0D 00 0E 00 0F 00 10 00 11 00 12 00 13 00	
00083A70	14 00 15 00 16 00 17 00 18 00 19 00 1A 00 1B 00	
00083A80	1C 00 1D 00 1E 00 1F 00 20 00 21 00 22 00 23 00	! " #
00083A90	24 00 25 00 26 00 27 00 28 00 29 00 2A 00 2B 00	\$ % & ' ( ) * +
00083AA0	2C 00 2D 00 2E 00 2F 00 30 00 31 00 32 00 33 00	, - . 0 1 2 3 4 5 6 7 8 9

可以看见它的序号是从2开始，而AddressOfNamesOrdinals和AddressOfNames——对应，说明前两个函数并没有名字。同时函数序号=nBase+NamesOrdinals。

2. AddressOfFunctions

首先用LordPE计算它在文件中位置，得出RAW=0x00081DA8

00081DA0	54 43 08 00 4C 50 08 00 10 B2 02 00 30 AE 02 00	TC LP ² 0
00081DB0	10 83 00 00 F0 AC 02 00 A0 35 02 00 EC 7C 01 00	f ð 5 ì
00081DC0	70 F3 07 00 C4 21 04 00 F0 7C 00 00 F8 BF 01 00	pó Å! ð   ø ç
00081DD0	8C A8 07 00 D0 D8 07 00 8C 36 01 00 70 1B 04 00	€ " ð Ø € 6 p
00081DE0	90 D1 00 00 B4 CC 00 00 10 6E 01 00 10 AD 02 00	Ñ ' ì n -

可以看见第一个函数的RVA等于0x0002B210，后续同理

3. AddressOfNames

还是用LordPE计算它在文件中位置，RAW=0x00082D54

00082D50	2C 06 04 00 D3 56 08 00 EA 56 08 00 05 57 08 00	, ÓV êV W
00082D60	16 57 08 00 29 57 08 00 34 57 08 00 4E 57 08 00	W )W 4W NW
00082D70	67 57 08 00 75 57 08 00 7E 57 08 00 8A 57 08 00	gW uW ~W ŠW
00082D80	96 57 08 00 AB 57 08 00 BD 57 08 00 D1 57 08 00	-W <W ½W ÑW
00082D90	DC 57 08 00 E7 57 08 00 F8 57 08 00 0F 58 08 00	ÜW çW øW X
00082DA0	27 58 08 00 41 58 08 00 5B 58 08 00 73 58 08 00	'X AX [X sX

可以得到指向这些名称的地址(RVA), 采用第一个进行验证

第一个RVA=0x000856D3, RAW=0x000840D3

000840C0	47 03 48 03 49 03 4A 03 55 53 45 52 33 32 2E 64	G H I J USER32.d
000840D0	6C 6C 00 41 63 74 69 76 61 74 65 4B 65 79 62 6F	ll ActivateKeybo
000840E0	61 72 64 4C 61 79 6F 75 74 00 41 64 64 43 6C 69	ardLayout AddCli
000840F0	70 62 6F 61 72 64 46 6F 72 6D 61 74 4C 69 73 74	pboardFormatList
00084100	65 6E 65 72 00 41 64 6A 75 73 74 57 69 6E 64 6F	ener AdjustWindo
00084110	77 52 65 63 74 00 41 64 6A 75 73 74 57 69 6E 64	wRect AdjustWind
00084120	6F 77 52 65 63 74 45 78 00 41 6C 69 67 6E 52 65	owRectEx AlignRe
00084130	63 74 73 00 41 6C 6C 6F 77 46 6F 72 65 67 72 6F	cts AllowForegro

可以看见右边信息已经找到了函数名称。

#### 4. 综上

根据上面分析可以得到前五个函数的信息为

函数编号	函数RVA	函数名称
0x05DC	0X0002B210	无
0x05DD	0x0002AE30	无
0x05DE	0x00008310	ActivateKeyboardLayout
0x05DF	0x0002ACF0	AddClipboardFormatListener
0x05E0	0x000235A0	AdjustWindowRect

2. 编写程序并生成exe文件, 要求定义1048个字节长度的word数组, 在程序中对该数组赋随机数, 然后查找该数组的最小值, 并调用MessageBox函数和ExitProcess函数。

```

1  include C:\irvine\Irvine32.inc
2  includelib C:\irvine\kernel32.lib
3  includelib C:\irvine\User32.lib
4  includelib C:\irvine\Irvine32.lib
5  .stack 4096
6  .data
7  array word 524 DUP(?)
8  str1 byte "hello",0
9  str2 byte "dialog",0
10 .code
11 main PROC
12     mov ecx,lengthof array
13     mov esi,offset array
14     call Randomize;随机数种子初始化
15 L1:
16     mov eax,0
17     mov ax,50h;设置随机数范围0-50
18     call RandomRange;使得ax获得一个随机数
19     mov [esi],ax
20     add esi,type word
21     loop L1

```

```

22
23     mov ecx,lengthof array
24     mov esi,offset array
25     mov ax,50h
26 L2:;循环遍历得到最小值
27     cmp [esi],ax
28     JB change;遇见更小的跳入change把值给ax
29     add esi,type word
30     loop L2
31
32     INVOKE MessageBox,NULL,offset str1,offset str2,MB_OK;调用两次是担心结束
    时在L2处结束但是代码继续向下往change执行,导致ECX值出错。
33     INVOKE ExitProcess,0
34 change:
35     mov ax,[esi]
36     add esi,type word
37     loop L2
38
39     INVOKE MessageBox,NULL,offset str1,offset str2,MB_OK
40     INVOKE ExitProcess,0
41 main endp
42 end main

```

地址	HEX 数据	反汇编	寄存器 (FPU)
0040101D	. 66:8906	MOV WORD PTR DS:[ESI],AX	EAX 00000000
00401020	. 83C6 02	ADD ESI,2	ECX 00000000
00401023	^ E2 EA	LOOPD SHORT homework.0040100F	EDX 00401000
00401025	. B9 0C020000	MOV ECX,20C	EBX 7EFDE000
			ESP 0018FF8C
			EBP 0018FF94
地址	HEX 数据	ASCII	
00404000	35 00 46 00 2E 00 41 00 34 00 28 00 09 00 12 00	5.F...A.4.(...	
00404010	33 00 4C 00 2D 00 37 00 45 00 46 00 3E 00 04 00	3.L-.7.E.F.>.	
00404020	1E 00 4B 00 30 00 0D 00 32 00 1E 00 38 00 16 00	.K.0...2..8.	
00404030	46 00 33 00 08 00 34 00 4D 00 4E 00 26 00 4A 00	F.3. .4.M.N.&.J.	
00404040	0B 00 04 00 16 00 2C 00 24 00 48 00 4A 00 2F 00	. . ...\$.H.J./.	
00404050	3D 00 2E 00 28 00 25 00 19 00 2A 00 03 00 34 00	=...(.%. .*. .4.	
00404060	3C 00 01 00 4F 00 0F 00 2A 00 35 00 31 00 0B 00	<. .O. .*.5.1.	
00404070	18 00 0C 00 1B 00 2B 00 39 00 3A 00 43 00 43 00	... .+9...C.C.	
00404080	21 00 01 00 4D 00 27 00 25 00 07 00 2B 00 0B 00	!...M.'%. .+	
00404090	17 00 4F 00 13 00 44 00 0D 00 2E 00 08 00 25 00	.O. .D.... . .	
004040A0	2A 00 46 00 2E 00 42 00 13 00 4D 00 2A 00 20 00	*.F...B. .M.*..	
004040B0	0A 00 16 00 1F 00 41 00 15 00 05 00 41 00 1B 00	.. ...A. ...A.	
004040C0	27 00 3F 00 34 00 21 00 25 00 47 00 0C 00 38 00	'?4! .%.G...8.	
004040D0	01 00 41 00 2E 00 33 00 11 00 41 00 2D 00 15 00	.A...3. .A.-.	
004040E0	48 00 4C 00 1D 00 45 00 2B 00 25 00 13 00 14 00	H.L..E.+%. .	
004040F0	4C 00 40 00 22 00 28 00 11 00 41 00 0E 00 14 00	L.@."(. .A. .	
00404100	4D 00 2D 00 3B 00 3C 00 25 00 26 00 2D 00 04 00	M.-.;<%.&-	
00404110	3B 00 23 00 49 00 21 00 05 00 25 00 12 00 16 00	;#.I! . .	
00404120	16 00 31 00 0C 00 47 00 23 00 0C 00 3C 00 19 00	.1...G.#...<	
00404130	3E 00 39 00 45 00 3E 00 3D 00 0D 00 1B 00 4C 00	>.9.E.>=... L.	
00404140	04 00 1A 00 02 00 07 00 35 00 46 00 1F 00 11 00	. . . .5.F.	
00404150	36 00 14 00 44 00 10 00 09 00 48 00 37 00 07 00	6. .D. ...H.7.	
00404160	25 00 37 00 4C 00 2A 00 4B 00 24 00 25 00 0D 00	%.7.L.*.K.\$.%...	
00404170	21 00 43 00 48 00 45 00 19 00 08 00 08 00 35 00	!.C.H.E. . . .5.	
00404180	0A 00 18 00 09 00 01 00 15 00 36 00 40 00 0E 00	.. ... . .6.@.	

可以看见成功赋值,同时eax的后16位置ax得到最小值0。函数满足需求。

### 3. 请解析题2生成exe文件的节表,加载前、后导入函数的详细信息。

- 首先是关于节表信息

000001D0	2E 74 65 78 74 00 00 00	FA 10 00 00 00 10 00 00	.text	ú
000001E0	00 12 00 00 00 04 00 00	00 00 00 00 00 00 00 00		
000001F0	00 00 00 00 20 00 00 60	2E 72 64 61 74 61 00 00		.rdata
00000200	36 03 00 00 00 30 00 00	00 04 00 00 00 16 00 00	6	0
00000210	00 00 00 00 00 00 00 00	00 00 00 00 40 00 00 40		@ @
00000220	2E 64 61 74 61 00 00 00	25 0F 00 00 00 40 00 00	.data	% @
00000230	00 0E 00 00 00 1A 00 00	00 00 00 00 00 00 00 00		
00000240	00 00 00 00 40 00 00 C0	2E 72 73 72 63 00 00 00		@ À.rsrc
00000250	10 00 00 00 00 50 00 00	00 02 00 00 00 28 00 00	P	(
00000260	00 00 00 00 00 00 00 00	00 00 00 00 40 00 00 40		@ @
00000270	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00		

可以得到：

节表名称	内存中偏移量	内存中地址	文件中偏移量	文件中大小
.text	0x00001000	0x000010FA	0x00000400	0x00001200
.rdata	0x00003000	0x00000336	0x00001600	0x00000400
.data	0x00004000	0x00000F25	0x00001A00	0x00000E00
.rsrc	0x00005000	0x00000010	0x00002800	0x00000200

- 关于加载前、后导入函数的详细信息

主要的关注点在于 `OriginalFirstThunk` 和 `FirstThunk`，在 `winhex` 中分析为导入前的信息，在 `OllyDbg` 中分析为导入后的信息。

1. 首先是关于 `OriginalFirstThunk`

- 在 `winhex` 中

000016B0	24 31 00 00 0C 33 00 00	EC 32 00 00 5A 31 00 00	\$1	3	i2	Z1
000016C0	68 31 00 00 76 31 00 00	88 31 00 00 A2 31 00 00	h1	v1	^1	ç1
000016D0	B4 31 00 00 C6 31 00 00	E4 31 00 00 F4 31 00 00	'1	#1	ä1	ô1
000016E0	04 32 00 00 14 32 00 00	24 32 00 00 30 32 00 00	2	2	\$2	02
000016F0	44 32 00 00 54 32 00 00	68 32 00 00 74 32 00 00	D2	T2	h2	t2
00001700	90 32 00 00 A2 32 00 00	BC 32 00 00 C4 32 00 00	2	ç2	¼2	Ä2
00001710	DC 32 00 00 2A 33 00 00	00 00 00 00 40 31 00 00	Ü2	*3		@1
00001720	00 00 00 00 19 01 45 78	69 74 50 72 6F 63 65 73				ExitProces

- 在 `OD` 中

004030B0	24 31 00 00 0C 33 00 00	EC 32 00 00 5A 31 00 00	\$1...	3...	?...Z1..
004030C0	68 31 00 00 76 31 00 00	88 31 00 00 A2 31 00 00	h1..	v1..	?...?..
004030D0	B4 31 00 00 C6 31 00 00	E4 31 00 00 F4 31 00 00	?...?	?...?	?...?
004030E0	04 32 00 00 14 32 00 00	24 32 00 00 30 32 00 00	■2..	■2..	\$2..02..
004030F0	44 32 00 00 54 32 00 00	68 32 00 00 74 32 00 00	D2..	T2..	h2..t2..
00403100	90 32 00 00 A2 32 00 00	BC 32 00 00 C4 32 00 00	?...?	?...?	?...?
00403110	DC 32 00 00 2A 33 00 00	00 00 00 00 40 31 00 00	?...*	3.....	@1..
00403120	00 00 00 00 19 01 45 78	69 74 50 72 6F 63 65 73	....	■	ExitProces

可以看见他们没有任何区别，因为无论加载前、后，它都指向函数编号和名称

2. 其次是关于 `FirstThunk`

- 在 `winhex` 中

可以发现导入前 `FirstThunk` 和 `OriginalFirstThunk` 一样。

00001600	24 31 00 00 0C 33 00 00	EC 32 00 00 5A 31 00 00	\$1	3	i2	Z1
00001610	68 31 00 00 76 31 00 00	88 31 00 00 A2 31 00 00	h1	v1	^1	ç1
00001620	B4 31 00 00 C6 31 00 00	E4 31 00 00 F4 31 00 00	'1	#1	ä1	ô1
00001630	04 32 00 00 14 32 00 00	24 32 00 00 30 32 00 00	2	2	\$2	02
00001640	44 32 00 00 54 32 00 00	68 32 00 00 74 32 00 00	D2	T2	h2	t2
00001650	90 32 00 00 A2 32 00 00	BC 32 00 00 C4 32 00 00	2	ç2	¼2	Ä2
00001660	DC 32 00 00 2A 33 00 00	00 00 00 00 40 31 00 00	Ü2	*3		@1
00001670	00 00 00 00 B0 30 00 00	00 00 00 00 00 00 00 00		°0		

- 在 `OD` 中

00403000	28 7A CC 76 2D 7A D6 76 E1 79 D6 76 F0 13 CC 76	(z魏-z講醒講?魏
00403010	B6 5D CC 76 35 61 CE 76 47 83 D6 76 A1 50 CC 76	禡魏5a蜚G卩v 魏
00403020	28 13 CC 76 AD C6 CE 76 C0 11 CC 76 2E 5A CC 76	(■魏 蜚?魏.Z魏
00403030	B3 50 CC 76 1E 5A CC 76 4C 2F CC 76 C5 77 D6 76	砵魏Z魏L/魏航講
00403040	D9 7B D6 76 0B 78 D6 76 E5 3F CC 76 87 83 D6 76	賒講■x講?魏嗎講
00403050	E5 A7 CD 76 63 7C CE 76 FF 10 CC 76 06 5A CC 76	襖蚰c 蜚 ■魏IZ魏
00403060	FC 12 CC 76 82 12 CC 76 00 00 00 00 AE FD 46 77	?魏?魏.... Fw
00403070	00 00 00 00 B0 30 00 00 00 00 00 00 00 00	.....

可以看见这时候就出现不同了，因为在加载后，调用的函数调入内存，FirstThunk则不再跟OriginalFirstThunk指向相同的地方，而是指向了函数执行地址。可以以MessageBox函数为例，如下图所示40306C地址处的DWORD值是它的函数地址，对比上图的得到验证，其函数地址为0x7746FDAE

00401066	. 68 18444000	PUSH homework.00404418		Text =
0040106B	. 6A 00	PUSH 0		hOwner
0040106D	. E8 0E000000	CALL <JM		Message
00401072	. 6A 00	PUSH 0		ExitCo
00401074	. E8 01000000	CALL <JM		ExitPr
00401079	CC	INT3		
0040107A	.- FF25 00304000	JMP DWORD PTR DS:[00304000]		kernel
00401080	\$- FF25 6C304000	JMP DWORD PTR DS:[<&USER32.MessageBoxA>]		USER32
00401086	CC	INT3		
00401087	CC	INT3		

