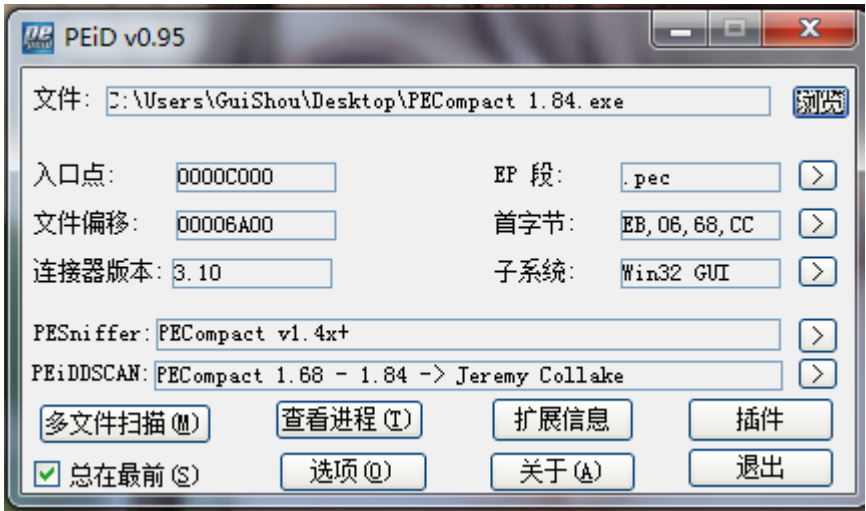


查壳  
单步跟踪到OEP  
修复导入表

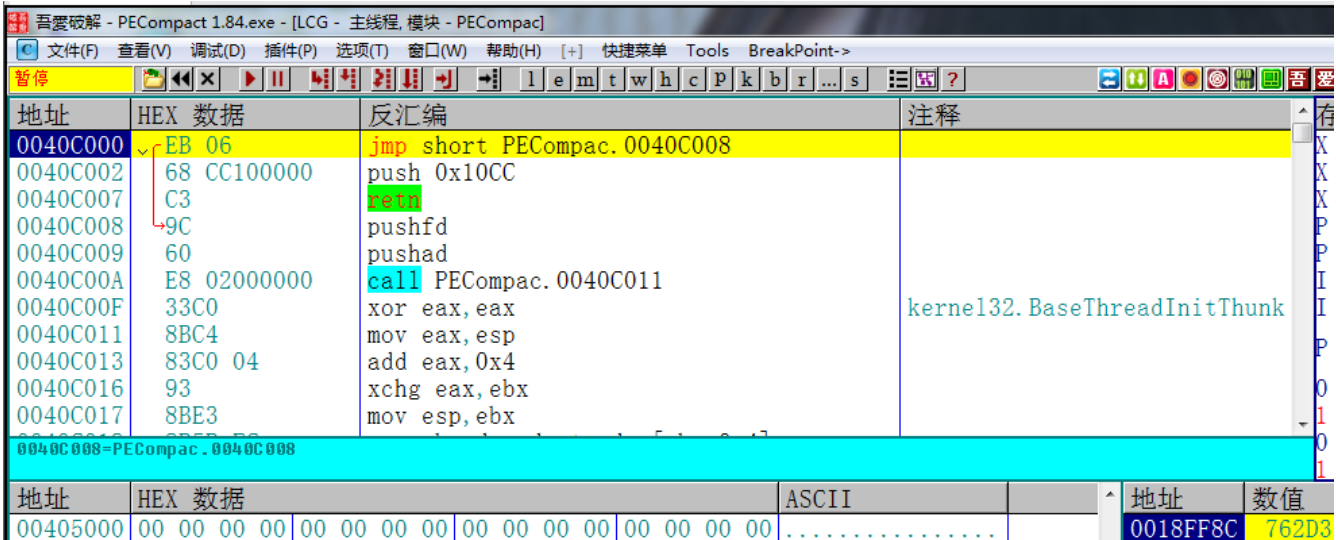
## 查壳



今天来脱一下PECompact 1.84这个壳，脱这个壳的目的是为了了解一个技巧——当遇到无法向下跟随的时候，可以找到附近没有实现的大跳转，下断点继续跟踪。

## 单步跟踪到OEP

接下来OD载入这个程序，采用单步跟踪的方法脱掉这个压缩壳。



地址	HEX 数据	反汇编	注释
0040D2A5	83C4 04	add esp,0x4	
0040D2A8	85C0	test eax,eax	
0040D2AA	74 07	je short PECompac.0040D2B3	
0040D2AC	8BC8	mov ecx,eax	
0040D2AE	5E	pop esi	PECompac.004101F0
0040D2AF	5F	pop edi	
0040D2B0	EB 9B	jmp short PECompac.0040D24D	
0040D2B2	B9 E8000000	mov ecx,0xE8	
0040D2B7	005D 81	add byte ptr ss:[ebp-0x7F],b1	
0040D2BA	ed	in eax,dx	
0040D2BB	E9 914000E8	jmp E8411351	
0040D24D=PECompac.0040D24D			

当程序执行到这个位置的时候，这里有一个jmp是往上跳的，按照常规的单步跟踪脱壳的方法，我们应该直接在下一条命令处F4，让他执行到下一条命令。但是当你在下一条命令直接F4的时候，程序会直接跑飞。

这个时候就要用到刚才的技巧了，在附近寻找一个没有实现的大跳转，一般是在上面。一直往上拉，

地址	HEX 数据	反汇编	注释
0040D24D	57	push edi	
0040D24E	AD	lods dword ptr ds:[esi]	
0040D24F	85C0	test eax,eax	
0040D251	0F84 9B000000	je PECompac.0040D2F2	
0040D257	8BD0	mov edx,eax	
0040D259	0395 E6904000	add edx,dword ptr ss:[ebp+0x4090E6]	PECompac.00400000
0040D25F	AD	lods dword ptr ds:[esi]	
0040D260	56	push esi	PECompac.004101F0
0040D261	8BC8	mov ecx,eax	
0040D263	57	push edi	
0040D264	52	push edx	PECompac.00401004
跳转未实现 0040D2F2=PECompac.0040D2F2			

在程序上面我们看到一个je，后面的OpCode的offset代表这是一个大跳转，而且是没有实现的。

地址	HEX 数据	反汇编	注释
0040D2F2	5F	pop edi	00220000
0040D2F3	8BB5 E2904000	mov esi,dword ptr ss:[ebp+0x4090E2]	PECompac.004101C4
0040D2F9	AD	lods dword ptr ds:[esi]	
0040D2FA	83F8 FF	cmp eax,-0x1	
0040D2FD	74 74	je short PECompac.0040D373	
0040D2FF	0385 E6904000	add eax,dword ptr ss:[ebp+0x4090E6]	PECompac.00400000
0040D305	8BD8	mov ebx,eax	
0040D307	AD	lods dword ptr ds:[esi]	
0040D308	0385 E6904000	add eax,dword ptr ss:[ebp+0x4090E6]	PECompac.00400000
0040D30E	8BD0	mov edx,eax	
0040D310	AD	lods dword ptr ds:[esi]	
堆栈 [0018FF64]=00220000 (00220000) edi=00220000			

那么我们就可以直接跟进去，下断点，让程序断下，然后继续单步跟踪。

地址	HEX 数据	反汇编	注释
0040D2F2	5F	pop edi	
0040D2F3	8BB5 E2904000	mov esi,dword ptr ss:[ebp+0x4090E2]	PECompac. 004101C4
0040D2F9	AD	lods dword ptr ds:[esi]	
0040D2FA	83F8 FF	cmp eax,-0x1	
0040D2FD	74 74	je short PECompac. 0040D373	
0040D2FF	0385 E6904000	add eax,dword ptr ss:[ebp+0x4090E6]	PECompac. 00400000
0040D305	8BD8	mov ebx,eax	
0040D307	AD	lods dword ptr ds:[esi]	
0040D308	0385 E6904000	add eax,dword ptr ss:[ebp+0x4090E6]	PECompac. 00400000
0040D30E	8BD0	mov edx,eax	
0040D310	AD	lods dword ptr ds:[esi]	
跳转未实现 0040D373=PECompac. 0040D373			

这里也是一个大跳转，按照刚才的方法，回车跟进去下断点，F9跑起来，就到了这里。继续往下跟

地址	HEX 数据	反汇编	注释
0040D373	68 00400000	push 0x4000	
0040D378	6A 00	push 0x0	
0040D37A	57	push edi	
0040D37B	FF95 45974000	call dword ptr ss:[ebp+0x409745]	kernel32.VirtualFree
0040D381	8BBD 3C964000	mov edi,dword ptr ss:[ebp+0x40963C]	
0040D387	03BD E6904000	add edi,dword ptr ss:[ebp+0x4090E6]	PECompac. 00400000
0040D38D	8BD8 40964000	mov ecx,dword ptr ss:[ebp+0x409640]	
0040D393	51	push ecx	
0040D394	57	push edi	
0040D395	33D2	xor edx,edx	
0040D397	33DB	xor ebx,ebx	PECompac. 00404FD5
跳转未实现			

跟到这里的时候，同样，回车跟进去下断点F9

地址	HEX 数据	反汇编	注释
0040D394	57	push edi	PECompac. 00401000
0040D395	33D2	xor edx,edx	
0040D397	33DB	xor ebx,ebx	
0040D399	33F6	xor esi,esi	
0040D39B	03FE	add edi,esi	
0040D39D	03DE	add ebx,esi	
0040D39F	49	dec ecx	
0040D3A0	74 72	je short PECompac. 0040D414	
0040D3A2	78 70	js short PECompac. 0040D414	
0040D3A4	66:8B07	mov ax,word ptr ds:[edi]	
0040D3A7	2C E8	sub al,0xE8	
跳转未实现			

这里还是一样的套路，继续跟

地址	HEX 数据	反汇编	注释
0040D414	5F	pop edi	PECompac. 00401000
0040D415	59	pop ecx	
0040D416	33C0	xor eax,eax	
0040D418	85C9	test ecx,ecx	
0040D41A	74 3B	je short PECompac. 0040D457	
0040D41C	8BF7	mov esi,edi	PECompac. 00401000
0040D41E	33C0	xor eax,eax	
0040D420	83F9 04	cmp ecx,0x4	
0040D423	72 32	jb short PECompac. 0040D457	
0040D425	87DB	xchg ebx,ebx	
0040D427	87DB	xchg ebx,ebx	
跳转未实现 0040D457=PECompac. 0040D457			
地址	HEX 数据	ASCII	地址 数值

地址	HEX 数据	反汇编	注释
0040D535	FFB5 3D974000	push dword ptr ss:[ebp+0x40973D]	
0040D53B	FFB5 39974000	push dword ptr ss:[ebp+0x409739]	
0040D541	E8 F40A0000	call PECompac.0040E03A	
0040D546	85C0	test eax,eax	kernel32.BaseThreadInitThunk
0040D548	0F85 9DFDFFFF	jnz PECompac.0040D2EB	
0040D54E	61	popad	
0040D54F	9D	popfd	
0040D550	50	push eax	kernel32.BaseThreadInitThunk
0040D551	68 CC104000	push PECompac.004010CC	
0040D556	C2 0400	ret 0x4	
0040D559	8BB5 5B974000	mov esi,dword ptr ss:[ebp+0x40975B]	

当跟到这里的时候，可以知道马上就到了OEP了，和大部分壳的套路一样，会push一个地址，然后返回回去。

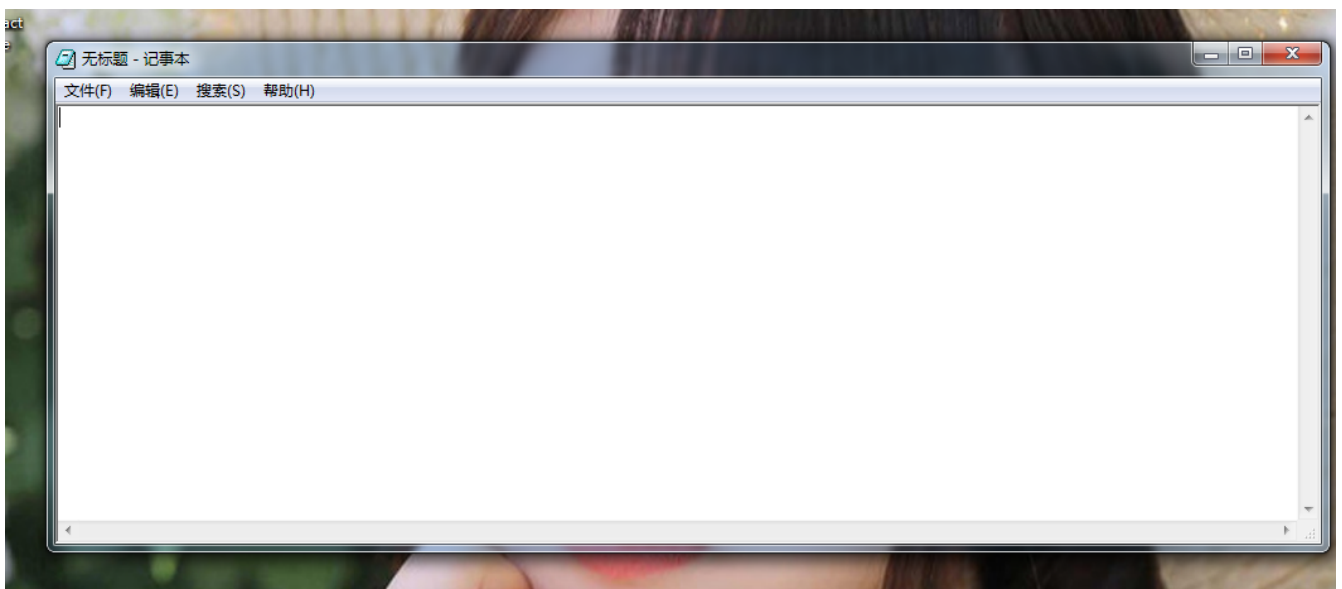
地址	HEX 数据	反汇编	注释
004010CC	55	push ebp	
004010CD	8BEC	mov ebp,esp	
004010CF	83EC 44	sub esp,0x44	
004010D2	56	push esi	
004010D3	FF15 E4634000	call dword ptr ds:[0x4063E4]	kernel32.GetCommandLineA
004010D9	8BF0	mov esi,eax	kernel32.BaseThreadInitThunk
004010DB	8A00	mov al,byte ptr ds:[eax]	
004010DD	3C 22	cmp al,0x22	
004010DF	75 1B	jnz short PECompac.004010FC	
004010E1	56	push esi	
004010E2	FF15 F4644000	call dword ptr ds:[0x4064F4]	user32.CharNextA

接着就来到了熟悉的OEP。

## 修复导入表



然后在OD里dump文件，然后修复导入表，输入OEP，自动查找IAT，获取输入表，转储文件。



脱壳后的程序正常运行，这个壳也就完成了。当然了，这个壳用ESP定律也是可以秒脱的，我这里纯粹是为了练习这个技巧而已。

需要相关文件可以到我的Github下载:<https://github.com/TonyChen56/Unpack-Practice>