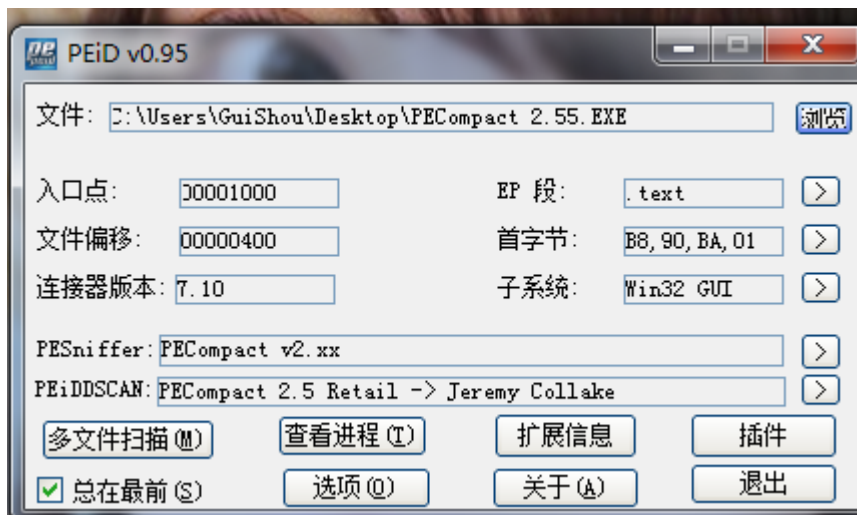


查壳
单步跟踪查找OEP
修复导入表

查壳



这个目标程序是PECompact 2.55的壳。脱这个壳的时候需要用一些技巧和套路。

单步跟踪查找OEP

OD载入,

地址	HEX 数据	反汇编	注释
01001000	B8 90BA0101	mov eax,PECompac.0101BA90	
01001005	50	push eax	kernel32.BaseT
01001006	64:FF35 00000000	push dword ptr fs:[0]	
0100100D	64:8925 00000000	mov dword ptr fs:[0],esp	
01001014	33C0	xor eax,eax	kernel32.BaseT
01001016	8908	mov dword ptr ds:[eax],ecx	
01001018	50	push eax	kernel32.BaseT
01001019	45	inc ebp	
0100101A	43	inc ebx	
0100101B	6f	outs dx,dword ptr ds:[esi]	
0100101C	6d	ins dword ptr es:[edi],dx	
0100101D	70 61	jo short PECompac.01001080	
0100101E	637432 00	arpl word ptr ds:[edx+esi],si	

地址	HEX 数据	ASCII
01009000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01009010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01009020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01009030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01009040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01009050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01009060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

M1 M2 M3 M4 M5	Command: bp VirtualFree
----------------	-------------------------

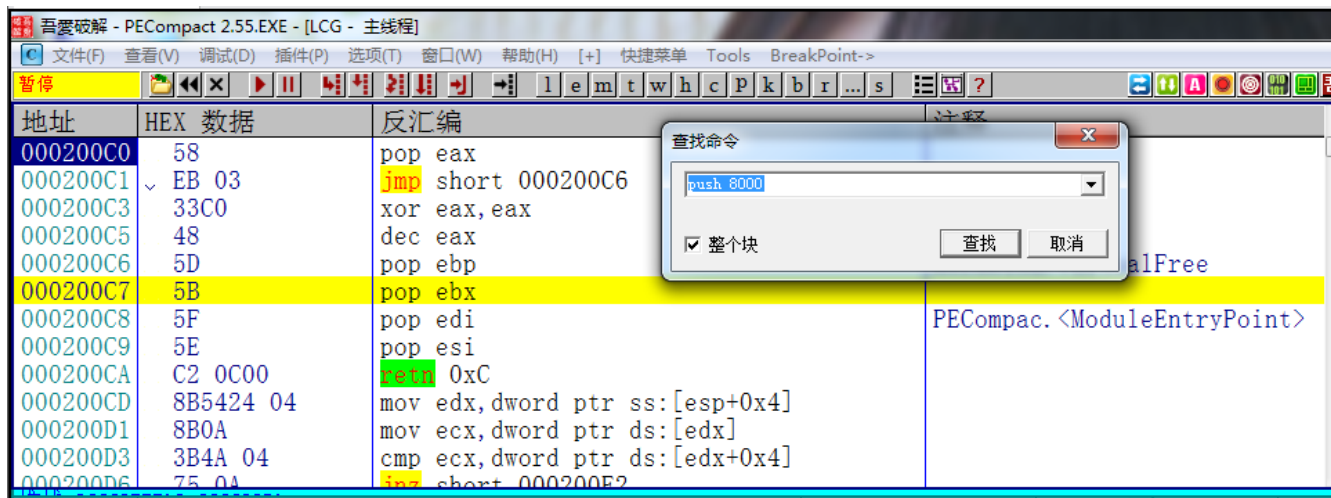
首先我们需要在VirtualFree下一个API断点，之所以下这个断点是因为壳在解压或者解密代码段的时候都需要申请一块空间来进行解密或解压操作，当VirtualFree完成之后，说明壳的解密或解压操作完成。

然后直接F9运行，

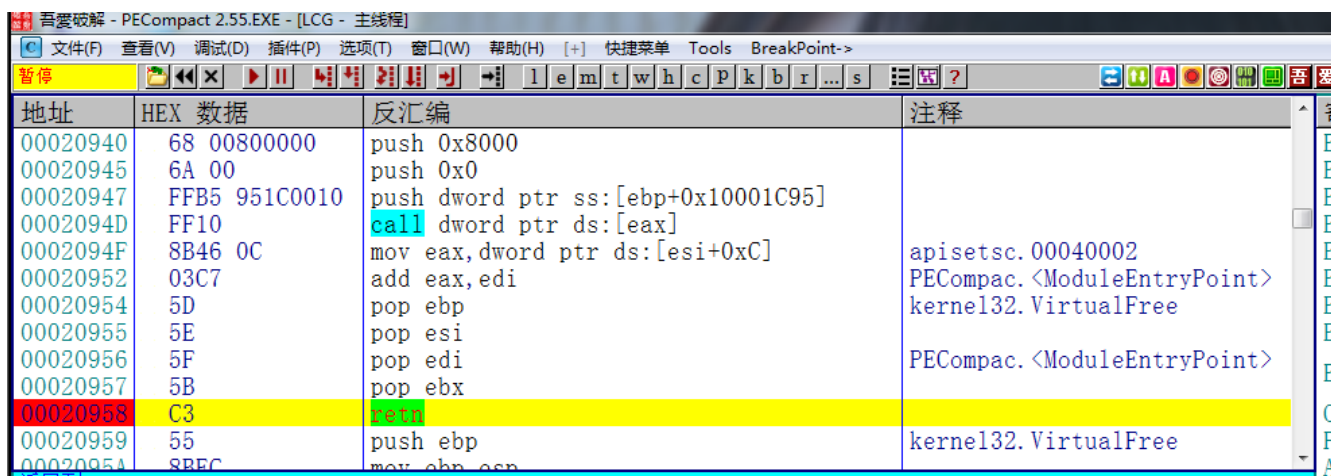
地址	HEX 数据	反汇编	注释
74BBEF39	8BFF	mov edi,edi	PECompac.<ModuleEntryPoint>
74BBEF3B	55	push ebp	kernel32.VirtualFree
74BBEF3C	8BEC	mov ebp,esp	
74BBEF3E	FF75 10	push dword ptr ss:[ebp+0x10]	
74BBEF41	FF75 0C	push dword ptr ss:[ebp+0xC]	
74BBEF44	FF75 08	push dword ptr ss:[ebp+0x8]	
74BBEF47	6A FF	push -0x1	
74BBEF49	E8 B5FEFFFF	call KernelBa.VirtualFreeEx	
74BBEF4E	5D	pop ebp	000200C0
74BBEF4F	C2 0C00	retn 0xC	
74BBEF52	CC	int3	
74BBEF53	CC	int3	
74BBEF54	CC	int3	

地址	HEX 数据	ASCII	地址	数值
01009000	00 26 00 46 00 6E 00 6E 00 74 00 2E 00 2E 00 2E	& E < n +	000CEFE4	000200C0

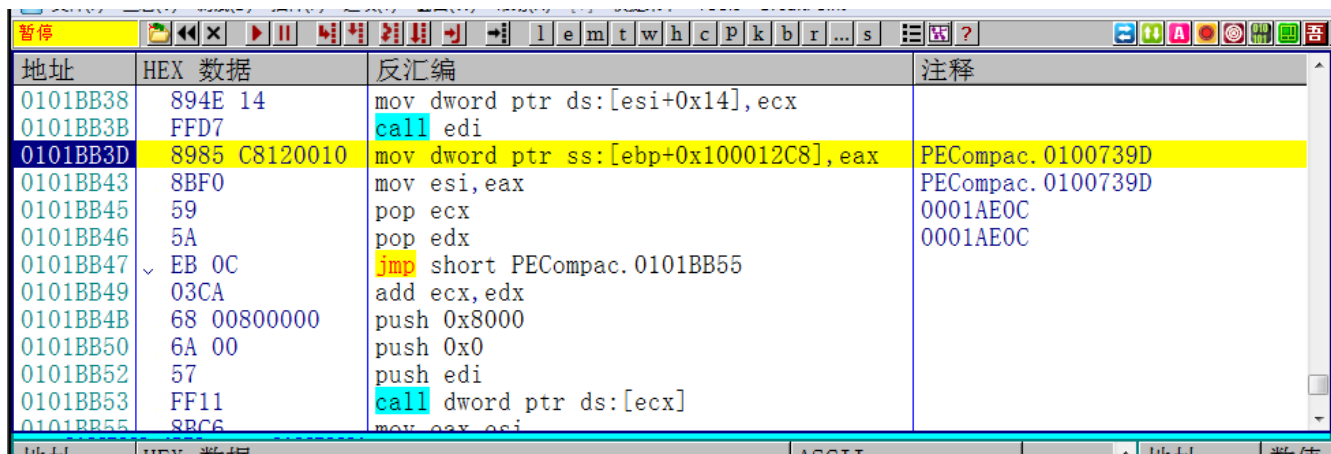
断在了VirtualFree处，接下来在retn处下断点，返回到用户代码区，我的Alt+F9执行到用户代码没有用，所以只好采取这种方法了。



接着我们查找push 8000这条命令，为什么要查找这条命令我也不知道，大概是前辈们总结出来的经验吧。就跟FSG壳的脱壳套路类似，每个壳都有一个套路。



找到push 8000之后，在retn处下断点。F9执行，就来到了这个地方



一直单步几下，

地址	HEX 数据	反汇编	注释
0101BB53	FF11	call dword ptr ds:[ecx]	
0101BB55	8BC6	mov eax,esi	
0101BB57	5A	pop edx	kernel32.762D33AA
0101BB58	5E	pop esi	kernel32.762D33AA
0101BB59	5F	pop edi	kernel32.762D33AA
0101BB5A	59	pop ecx	kernel32.762D33AA
0101BB5B	5B	pop ebx	kernel32.762D33AA
0101BB5C	5D	pop ebp	kernel32.762D33AA
0101BB5D	- FFE0	jmp eax	PECompact.0100739D

这一连串的指令就相当于popad了，这条指令F7之后就到达OEP了。

地址	HEX 数据	反汇编	注释
0100739D	6A 70	push 0x70	
0100739F	68 98180001	push PECompact.01001898	
010073A4	E8 BF010000	call PECompact.01007568	
010073A9	33DB	xor ebx,ebx	
010073AB	53	push ebx	
010073AC	8B3D CC100001	mov edi,dword ptr ds:[0x10010CC]	kernel32.GetModuleHandleA
010073B2	FFD7	call edi	
010073B4	66:8138 4D5A	cmp word ptr ds:[eax],0x5A4D	
010073B9	75 1F	jnz short PECompact.010073DA	
010073BB	8B48 3C	mov ecx,dword ptr ds:[eax+0x3C]	
010073BE	03C8	add ecx,eax	PECompact.0100739D
010073C0	8139 50450000	cmp dword ptr ds:[ecx],0x4550	
010073C6	75 12	jnz short PECompact.010073DA	

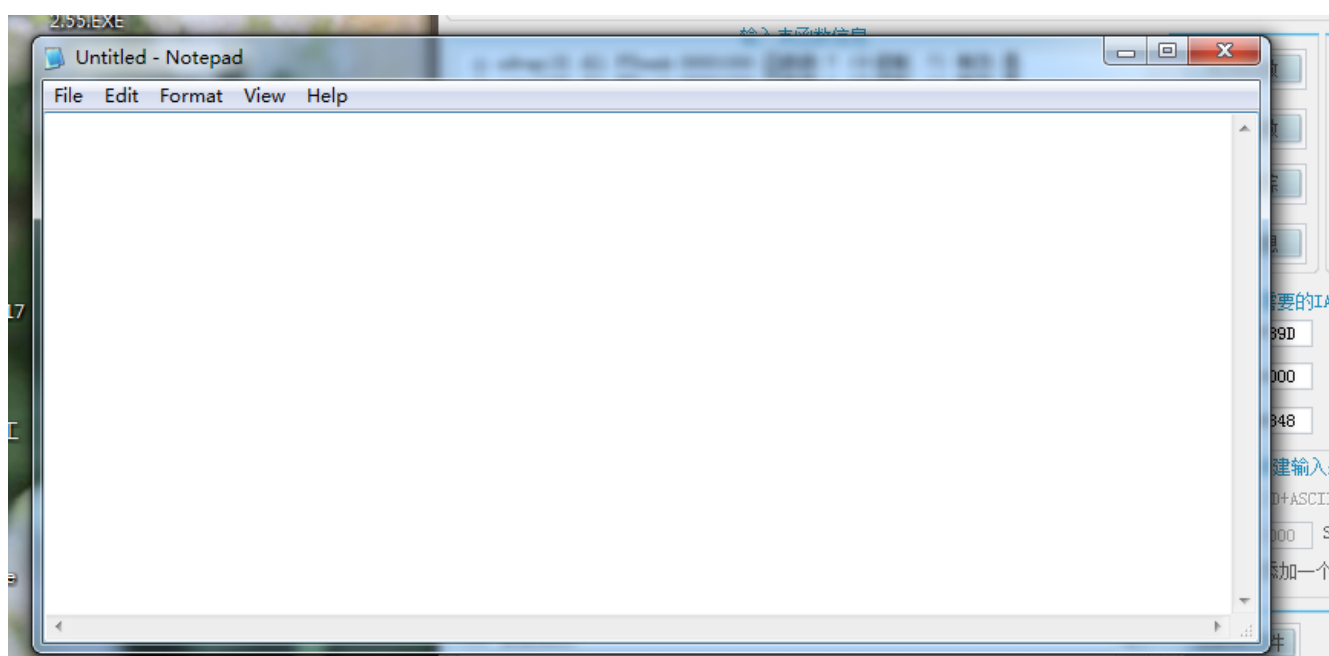
地址	HEX 数据	ASCII	地址	数值
01009000	00 00 00 00 D4 70 00 01 00 00 00 00 00 00 00 00詐.	000CFF8C	762D3

在这里看到了GetModuleHandleA，说明这个就是入口点了。

修复导入表



接下来dump文件，自动查找IAT 获取输入表，这里有无效指针不能剪切掉，如果剪切掉了转储的程序就无法正常运行。接着转储文件。



脱壳后的程序正常运行。

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