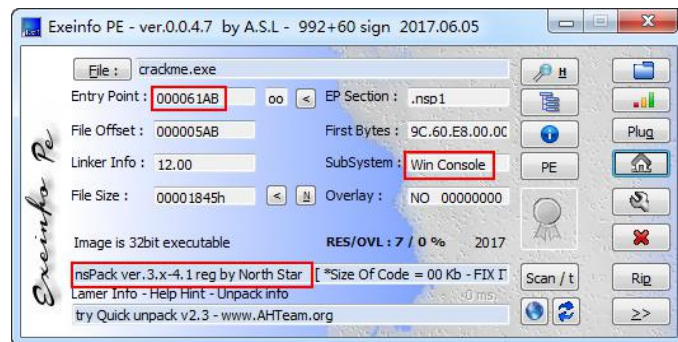


nsPack 手动脱壳

运行程序，无法运行，使用 010editor 打开，发现魔术字“pe”为小写，改为大写“PE”，可正常运行，然后查壳：



这是一个 32 位控制台程序，使用 nsPack 加壳，EP（000061AB）肯定不是 OEP 了。运行此程序，如下：

```
C:\Users\nsfocus-cyf>C:\Users\nsfocus-cyf\Desktop\crackme.exe
Please Input Flag:aaaa
error!

C:\Users\nsfocus-cyf>
```

输入 flag，然后校验，错误，返回“error!”，猜测应该使用了标准输出函数，如 printf 等。下面分析壳代码：

首先是一个保存寄存器的压栈操作：

004061AB	9C	pushfd
004061AC	60	pushad
004061AD	E8 00000000	call crackme.004061B2
004061B2	5D	pop ebp
004061B3	83ED 07	sub ebp,0x7
004061B6	8D8D D1FEFFFF	lea ecx,dword ptr ss:[ebp-0x12F]
004061BC	8039 01	cmp byte ptr ds:[ecx],0x1
004061BF	0F84 42020000	je crackme.00406407
004061C5	C601 01	mov byte ptr ds:[ecx],0x1
004061C8	8BC5	mov eax,ebp

继续跟进，动态申请了一块内存，生成一个动态的区块：

004061E2	0106	add dword ptr ds:[esi],eax	
004061E4	55	push ebp	
004061E5	56	push esi	
004061E6	6A 40	push 0x40	
004061E8	68 00100000	push 0x1000	
004061ED	68 00100000	push 0x1000	
004061F2	6A 00	push 0x0	
004061F4	FF95 FDFEFFFF	call dword ptr ss:[ebp-0x103]	
004061FA	85C0	test eax,eax	
004061FC	0F84 69030000	je crackme.0040656B	
00406202	8985 8DFEFFFF	mov dword ptr ss:[ebp-0x173],eax	
00406208	E8 00000000	call crackme.0040620D	
0040620D	5B	pop ebx	
0040620E	09 67030000	mov ecx,0x367	
00406213	0309	add ebx,ecx	
00406215	50	push eax	
00406216	53	push ebx	
00406217	E8 04020000	call crackme.004064CC	写入内容
0040621C	5E	pop esi	
0040621D	5D	pop ebp	
0040621E	8F36	mov esi,dword ptr ds:[esi]	
00406220	9F00	mov edi,ebx	
004064CC	crackme.004064CC		

地址	HEX 数据	ASCII
00030000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00030010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00030020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00030030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

返回地址是 0x00030000，大小是 0x1000，接着在 0x00406217 处调用函数，来填充该区块，crackme.004064cc 处代码如下：

004064CC	55	push ebp	
004064CD	8BEC	mov ebp,esp	
004064CF	8B75 08	mov esi,dword ptr ss:[ebp+0x8]	
004064D2	8B7D 0C	mov edi,dword ptr ss:[ebp+0xC]	
004064D5	FC	cld	
004064D6	B2 80	mov dl,0x80	
004064D8	A4	movs byte ptr es:[edi],byte ptr ds:[esi]	
004064D9	E8 6D000000	call crackme.00406548	
004064DE	73 F8	jnb short crackme.004064D8	
004064E0	33C9	xor ecx,ecx	
004064E2	E8 64000000	call crackme.00406548	
004064E7	73 19	jnb short crackme.00406502	
004064E9	33C0	xor eax,eax	

00406574处
开始读取

然后把 0x003005AA 放入 eax, 执行 call eax, 然后在 eip=0x00406289 处 free 掉动态申请的内存。

这些解密例程都不是我们关心的, 我们关心的只有两件事: OEP 和 IAT, 首先 OEP 不太好找, 先找 IAT, 找到 IAT 就离 OEP 不远了, 首先全速运行程序, 调用了 msvcrt120.printf 函数, 查找该函数地址:



在该地址处下断, 并且启用跟踪步入, 结果如下:

6.	msvcrt120	7022159F	mov eax,dword ptr ss:[esp+0x0]	EAX=0018FF11
5.	msvcrt120	702215A3	pop edi	ESP=0018FEFC, EDI=00000000
4.	msvcrt120	702215A4	retn	ESP=0018FF00
3.	crackme	00401022	mov esi,dword ptr ds:[0x402094]	ESI=70292FD9
2.	crackme	00401028	push crackme.00402100	ESP=0018FEFC
1.	crackme	0040102D	call esi	ESP=0018FEF8
0.	msvcrt120	70292FD9	push 0xC	

可以看到, 是从 0x0040102d 的 call esi 来调用的 printf, 跟进 0x0040102d, 如下:

00401000	55	push ebp		
00401001	8BEC	mov ebp,esp		
00401003	83EC 38	sub esp,0x38		
00401006	A1 00304000	mov eax,dword ptr ds:[0x403000]		
00401008	33C5	xor eax,ebp		
0040100D	8945 FC	mov [local.1],eax		
00401010	56	push esi		
00401011	6A 31	push 0x31		
00401013	8D45 C9	lea eax,dword ptr ss:[ebp-0x37]		
00401016	C645 C8 00	mov byte ptr ss:[ebp-0x38],0x0		
0040101A	6A 00	push 0x0		
0040101C	50	push eax		
0040101D	E8 D2000000	call crackme.004018F4		
00401022	8B35 94204000	mov esi,dword ptr ds:[0x402094]		
00401028	68 00214000	push crackme.00402100		
0040102D	FFD6	call esi		
0040102F	8D45 C8	lea eax,[local.14]		
00401032	6A 2C	push 0x2C		
00401034	50	push eax		
00401035	FF15 90204000	call dword ptr ds:[0x402090]		
0040103B	8D4D C8	lea ecx,[local.14]		
0040103E	83C4 18	add esp,0x18		
00401041	8D51 01	lea edx,dword ptr ds:[ecx+0x1]		
00401044	8A01	mov al,byte ptr ds:[ecx]		
00401046	41	inc ecx		
00401047	84C0	test al,al		
00401049	75 F9	jnz short crackme.00401044		
0040104B	2BCA	sub ecx,edx		
0040104D	83F9 2A	cmp ecx,0x2A		
00401050	74 1C	je short crackme.0040106E		
00401052	68 14214000	push crackme.00402114		
00401057	FFD6	call esi		

OEP

IAT
地址

开头是压栈和 ebp 赋值, 后续是程序逻辑, 所以这里就是 OEP。然后看调用 printf 的语句 call esi, 查找 esi 值得来源, 即 0x00402094 地址中的值, 则 0x00402094 就是 IAT 表的位置。查看 IAT 表:


```

1 signed int start()
2 {
3     signed int result; // eax@2
4     signed int v1; // eax@3
5     char Buf; // [sp+4h] [bp-38h]@1
6     char Dst; // [sp+5h] [bp-37h]@1
7
8     Buf = 0;
9     memset(&Dst, 0, 0x31u);
10    printf("Please Input Flag:");
11    gets_s(&Buf, 0x2Cu);
12    if ( strlen(&Buf) == 42 )
13    {
14        v1 = 0;
15        while ( (*(&Buf + v1) ^ byte_402130[v1 % 16]) == dword_402150[v1] )
16        {
17            if ( ++v1 >= 42 )
18            {
19                printf("right!\n");
20                goto LABEL_8;
21            }
22        }
23        printf("error!\n");
24    LABEL_8:
25        result = 0;
26    }
27    else
28    {
29        printf("error!\n");
30        result = -1;
31    }
32    return result;
33 }

```

在 nsp0 段中

写出 c 正向算法:

```

#include<stdio.h>
#include<string.h>
int main()
{
    char ans[50];
    char flag[]="0x12,0x04,0x08,0x14,0x24,0x5C,0x4A,0x3D,0x56,0x0F,
    char key[]="this_is_not_flag";
    scanf("%s",ans);
    if(strlen(ans)==42)
    {
        int i=0;
        while(ans[i]^key[i%16]==flag[i])
        {
            ++i;
            if(i==42)
            {
                printf("right.\n");
                result=0;
            }
        }
        printf("error.\n");
    }
    else
    {
        printf("error.\n");
        result=-1;
    }
    return result;
}

```

这个算法大致思路是：存储一个数组 flag，里面是乱码，存有一个字符串 key "this_is_not_flag"，然后取下标 i 与 16 取余，去除 key 中该余数下表的字符，然后与下标为 i 的 flag 中字符去异或，得到 flag。

然后写逆算法:

```

#include<stdio.h>
#include<string.h>
int main()
{
    char flag[]="\x12\x04\x08\x14\x24\x5C\x4A\x3D\x56\x0A\x10"
    "\x67\x00\x41\x00\x01\x46\x5A\x44\x42\x6E\x0C"
    "\x44\x72\x0C\x0D\x40\x3E\x4B\x5F\x02\x01\x4C"
    "\x5E\x5B\x17\x6E\x0C\x16\x68\x5B\x12";
    char key[]="this_is_not_flag";
    char ans[50];
    for(int i=0;i<42;i++)
        ans[i]=flag[i]^key[i%16];
    printf("%s",ans);
}

```

编译逆算法 c 文件，运行，即可得到 flag:


```
#include<stdio.h>
#include<string.h>
int main()
{
    char flag[]="\x12\x04\x08\x14\x24\x5C\x4A\x3D\x56\x0A\x10"
               "\x67\x00\x41\x00\x01\x46\x5A\x44\x42\x6E\x0C"
               "\x44\x72\x0C\x0D\x40\x3E\x4B\x5F\x02\x01\x4C"
               "\x5E\x5B\x17\x6E\x0C\x16\x68\x5B\x12";

    char key[]="this_is_not_flag";
    char ans[50];
    for(int i=0;i<42;i++)
        ans[i]=flag[i]^key[i%16];
    printf("%s",ans);
}
```

C:\Users\nsfocus-cyf\Desktop\re-crackme.exe
flag{59b8ed6f-af22-11e7-bb4a-3cf862d1ee75}o?E
Process returned 0 (0x0) execution time : 0.094 s
Press any key to continue.