xctf windows_reverse1

原创

菜逼的ctf之路 于 2020-10-24 12:53:58 发布 159 收藏 1

文章标签： 1024程序员节

本文链接：https://blog.csdn.net/weixin_45701079/article/details/109257718

xctf windows_reverse1

日常水一波

这道题是有加壳的(upx),用010分析就知道了

脱壳之后找到动态调试不行，经大佬讲解，发现win7之后加入了ALSR，然后这个文件不支持ALSR，所以动态调试不行，只能用ida打开

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [esp+4h] [ebp-804h]
  char v5; // [esp+5h] [ebp-803h]
  char v6; // [esp+404h] [ebp-404h]
  char Dst; // [esp+405h] [ebp-403h]

  v6 = 0;
  memset(&Dst, 0, 0x3FFu);
  v4 = 0;
  memset(&v5, 0, 0x3FFu);
  printf("please input code:");
  scanf("%s", &v6);
  sub_401000(&v6);
  if ( !strcmp(&v4, "DDCTF{reverseME}") )
    printf("You've got it!!%s\n", &v4);
  else
    printf("Try again later.\n");
  return 0;
}
```


乍一看，v4和v6没有任何关系，但是查看汇编

```
.text:004010A2                 call    esi ; printf
.text:004010A4                 lea     edx, [esp+824h+var_404]
.text:004010AB                 push    edx
.text:004010AC                 push    offset aS       ; "%s"
.text:004010B1                 call    ds:scanf
.text:004010B7                 lea     eax, [esp+82Ch+var_404]
.text:004010BE                 push    eax
.text:004010BF                 lea     ecx, [esp+830h+var_804]
.text:004010C3                 call    sub_401000
.text:004010C8                 add     esp, 28h
.text:004010CB                 mov     ecx, offset aDdctfReverseme ; "DDCTF{reverseME}"
.text:004010D0                 lea     eax, [esp+808h+var_804]
.text:004010D4
.text:004010D4 loc_4010D4:                             ; CODE XREF: _main+9E↓j
.text:004010D4                 mov     dl, [eax]
.text:004010D6                 cmp     dl, [ecx]
.text:004010D8                 jnz     short loc_4010F4
```

把v4的地址放入了ecx，然后看调用的函数

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1 | Structures

```
1  unsigned int __cdecl sub_401000(const char *a1)
2  {
3    _BYTE *v1; // ecx
4    unsigned int v2; // edi
5    unsigned int result; // eax
6    const char *v4; // ebx
7
8    v2 = 0;
9    result = strlen(a1);
10   if ( result )
11   {
12     v4 = (const char *)(a1 - v1);
13     do
14     {
15       *v1 = *((_BYTE *)&dword_402FF6 + (char)v1[(_DWORD)v4] + 2);
16       ++v2;
17       ++v1;
18       result = strlen(a1);
19     }
20     while ( v2 < result );
21   }
22   return result;
23 }
```


v1的保存值就是ecx，所以思路就出来了。v1的值就是main函数的v4的值

然后通过地址相减(仔细看上图会发现是个负数)，然后地址的差值作为另一个的索引，这里用了溢出，然后找到这个数据段进行索引就好了。

最后贴上源码

```
data = [ 0x7E, 0x7D, 0x7C, 0x7B, 0x7A, 0x79, 0x78, 0x77, 0x76, 0x75, 0x74, 0x73, 0x72, 0x71,
    0x70, 0x6F, 0x6E, 0x6D, 0x6C, 0x6B, 0x6A, 0x69, 0x68, 0x67, 0x66, 0x65, 0x64, 0x63, 0x62, 0x61,
    0x60, 0x5F, 0x5E, 0x5D, 0x5C, 0x5B, 0x5A, 0x59, 0x58, 0x57, 0x56, 0x55, 0x54, 0x53, 0x52, 0x51,
    0x50, 0x4F, 0x4E, 0x4D, 0x4C, 0x4B, 0x4A, 0x49, 0x48, 0x47, 0x46, 0x45, 0x44, 0x43, 0x42, 0x41,
    0x40, 0x3F, 0x3E, 0x3D, 0x3C, 0x3B, 0x3A, 0x39, 0x38, 0x37, 0x36, 0x35, 0x34, 0x33, 0x32, 0x31,
    0x30, 0x2F, 0x2E, 0x2D, 0x2C, 0x2B, 0x2A, 0x29, 0x28, 0x27, 0x26, 0x25, 0x24, 0x23, 0x22, 0x21,
    0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,


]
s='DDCTF{reverseME}'
flag=''
for i in range(len(s)):
    flag+=chr(data.index(ord(s[i]))+32)
print(flag)
```