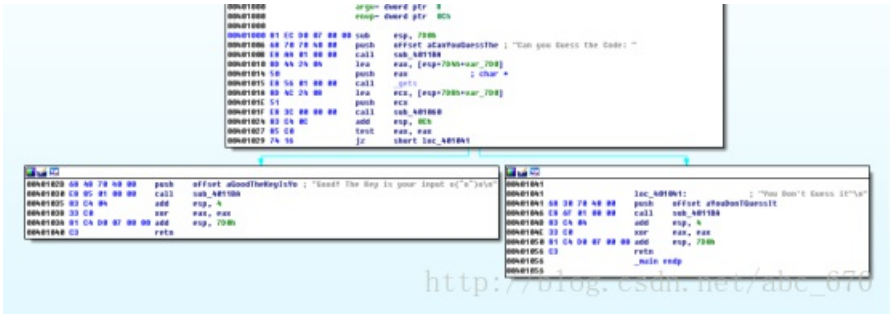
实验吧-证明自己吧（超详细）

原创

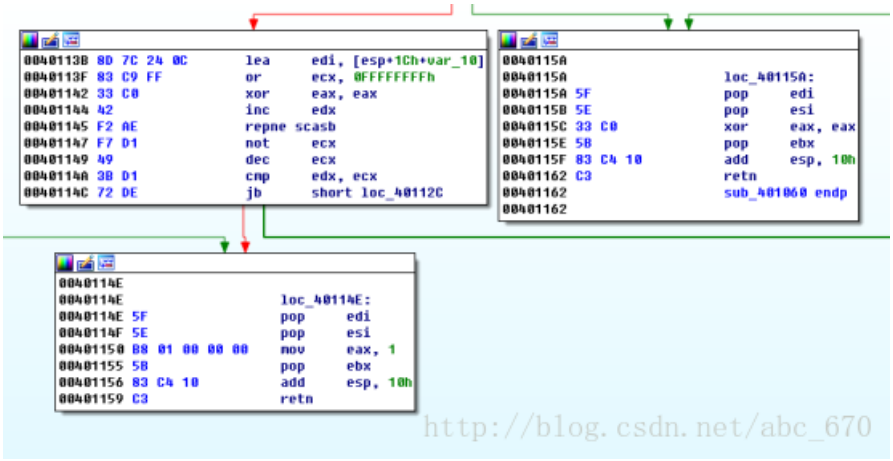
LqL_1 于 2018-03-08 10:17:21 发布 2635 收藏
分类专栏： 逆向 文章标签： CTF 实验吧

把Crackme拖到IDA里查看



可以看到，程序通过eax跳转，而eax的值很有可能是上面那个call sub_401060决定的。

进入到sub_401060查看，可以看到最后的跳转代码有两个，一个是将eax清零，是错误跳转，一个是将eax赋值1，是正确跳转



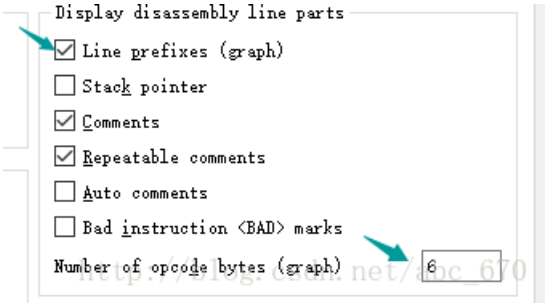
直接F5查看代码，关键代码如下：其中a1是传入函数的一个指针，指向我们输入的字符串，

这段代码大致就是将传入的字符串与0x20进行异或，然后将程序存储的一串字符进行减5，然后将两个字符串比较，如果相等就返回1，既成功，否则失败

```
if ( strlen(a1) == strlen((const char *)&v5) )
{
  v1 = 0;
  if ( strlen(a1) != 0 )
  {
    do
      a1[v1++] ^= 0x20u;
    while ( v1 < strlen(a1) );
  }
  v2 = 0;
  if ( strlen((const char *)&v5) != 0 )
  {
    do
      *((_BYTE *)&v5 + v2++) -= 5;
    while ( v2 < strlen((const char *)&v5) );
  }
  v3 = 0;
  if ( strlen((const char *)&v5) == 0 )
    return 1;
  while ( *((_BYTE *)&v5 + v3 + a1 - (const char *)&v5) == *((_BYTE *)&v5 + v3) )
  {
    if ( ++v3 >= strlen((const char *)&v5) )
      return 1;
  }
}
```

unsigned int

到这里已经很清楚了，然后用OD来找程序存储的另外一字符串就可以了，首先用ida查看那段代码的地址，在option中设置，



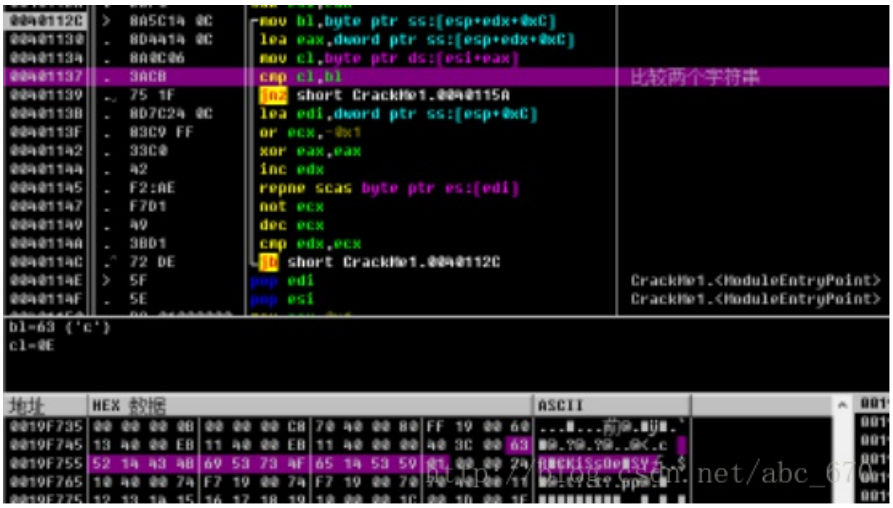

```
00401060                        var_2= byte ptr -2
00401060                        arg_0= dword ptr  4
00401060
00401060 83 EC 10               sub     esp, 10h
00401063 A1 8C 70 40 00         mov     eax, dword_40708C
00401068 8B 0D 90 70 40 00      mov     ecx, dword_407090
0040106E 53                     push    ebx
0040106F 89 44 24 04            mov     [esp+14h+var_10], eax
00401073 66 A1 98 70 40 00      mov     ax, word_407098
00401079 89 4C 24 08            mov     [esp+14h+var_C], ecx
0040107D 8A 0D 9A 70 40 00      mov     cl, byte_40709A
00401083 56                     push    esi
00401084 8B 74 24 1C            mov     esi, [esp+18h+arg_0]
00401088 57                     push    edi
00401089 66 89 44 24 18         mov     [esp+1Ch+var_4], ax
0040108E 88 4C 24 1A            mov     [esp+1Ch+var_2], cl
00401092 8B FE                  mov     edi, esi
00401094 83 C9 FF               or      ecx, 0FFFFFFFFh
00401097 33 C0                  xor     eax, eax
00401099 8B 15 94 70 40 00      mov     edx, dword_407094
0040109F F2 AE                  repne scasb
004010A1 F7 D1                  not     ecx
```


这里解释一下ida里的一串代码，repne scasb在代码中出现了很多次，作用就是计算字符串长度，edi中存放要计算的字符串，ecx中存放算好的字符串长度，eax中存放要查找的字符，如果是0的话就是计算字符串长度

```
lea     edi, [esp+1Ch+var_10]
or      ecx, 0FFFFFFFFh          设置循环次数-1
xor     eax, eax                 设置搜索内容0
xor     edx, edx
repne scasb                      一直重复搜索到EDI字符串末尾的0
not     ecx                      得到搜索次数，也就是字符串的完整长度
dec     ecx                      -1得到字符串不包含末尾0的长度
```

转到OD，转到上述地址，然后F2下断点，F9运行到刚下的断点处，运行程序，发现程序固定的字符串长度为14,

一路运行，直到到比较两个字符串的位置，



找到固定字符串所在位置，然后读取出14个字符，分别
为0x63,0x52,0x14,0x43,0x4B,0x69,0x53,0x73,0x4F,0x65,0x14,0x53,0x59,0x01

最后一步，用python解出flag，代码如下：

```
a=[0x63,0x52,0x14,0x43,0x4B,0x69,0x53,0x73,0x4F,0x65,0x14,0x53,0x59,0x01]
for i in a:
    print(chr(i^0x20),end='')
```