

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](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/abc_670/article/details/79480560

版权



[逆向 专栏收录该内容](#)

12 篇文章 2 订阅

订阅专栏

把Crackme拖到IDA里查看

```
00401000      org= dword ptr 1
00401000      org= dword ptr 004
00401000
00401000  81 5C 06 07 00  sub     esp, 700h
00401000  7E 78 10 00     push  offset 00401000: "Can you guess the Code? "
00401000  E8 00 01 00 00  call   sub_401100
00401000  8B 00 01 00 00  lea   eax, [esp+700+var_700]
00401000  58             push  eax
00401000  E8 56 01 00 00  call   sub_401100
00401000  8B 0C 2A 00     lea   ecx, [esp+700+var_700]
00401000  51             push  ecx
00401000  7E 3C 00 00 00  call   sub_401060
00401000  83 C4 00 00     add   esp, 004
00401000  85 C6 00 00     test  eax, eax
00401000  74 76 00 00     jr    short loc_401061

sub_401100
00401100  8B 78 10 00     push  offset_00401100: "Exact the key to your input c(\"")\n"
00401100  28 01 00 00     call   sub_401100
00401100  83 C4 00 00     add   esp, 4
00401100  8B 00 01 00 00  mov   eax, eax
00401100  83 C4 00 00 00  add   esp, 700h
00401100  C3             retn

sub_401060
00401060  8B 78 10 00     push  offset_00401060: "You Don't Guess It?"
00401060  28 01 00 00     call   sub_401100
00401060  83 C4 00 00     add   esp, 4
00401060  33 C6 00 00     xor   eax, eax
00401060  83 C4 00 00 00  add   esp, 700h
00401060  C3             retn
00401060      _main endp
00401060
```

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

```
0040113B 8D 7C 24 0C     lea  edi, [esp+1Ch+var_10]
0040113F 83 C9 FF 00     or   ecx, 0FFFFFFFh
00401142 33 C0 00 00     xor  eax, eax
00401144 42 00 00 00     inc  edx
00401145 F2 AE 00 00     repne scasb
00401147 F7 D1 00 00     not  ecx
00401149 49 00 00 00     dec  ecx
0040114A 3B D1 00 00     cmp  edx, ecx
0040114C 72 DE 00 00     jb   short loc_40112C

loc_40114E:
0040114E      loc_40114E:
0040114E 5F 00 00 00     pop  edi
0040114F 5E 00 00 00     pop  esi
00401150 8B 01 00 00 00  mov  eax, 1
00401155 5B 00 00 00     pop  ebx
00401156 83 C4 10 00     add  esp, 10h
00401159 C3 00 00 00     retn

loc_40115A:
0040115A      loc_40115A:
0040115A 5F 00 00 00     pop  edi
0040115B 5E 00 00 00     pop  esi
0040115C 33 C6 00 00     xor  eax, eax
0040115E 5B 00 00 00     pop  ebx
0040115F 83 C4 10 00     add  esp, 10h
00401162 C3 00 00 00     retn
00401162      sub_401060 endp
00401162
```

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

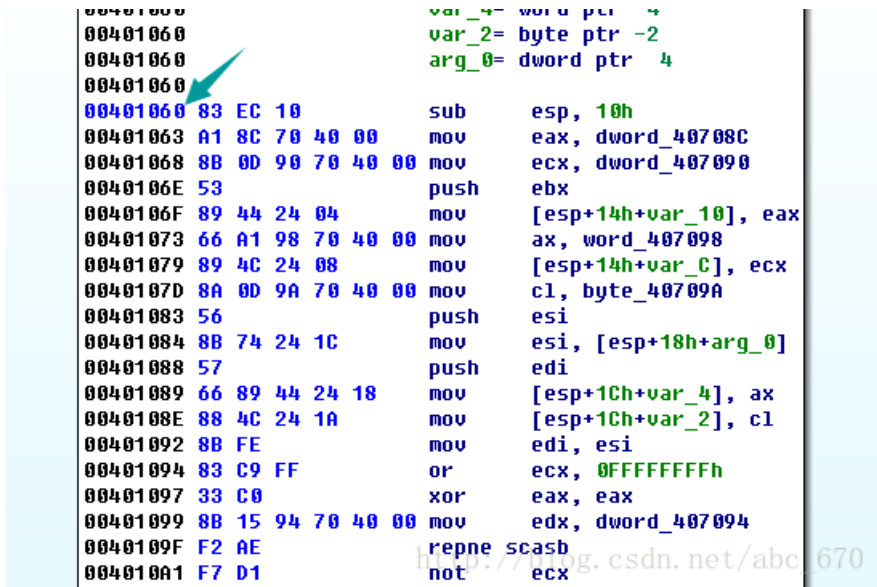
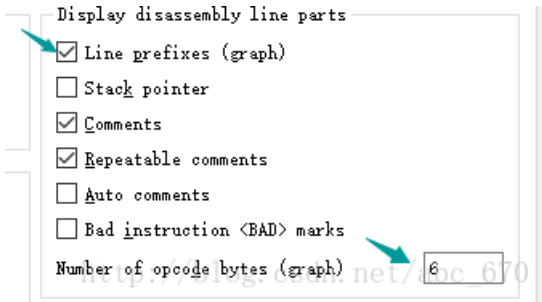
```

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;
    }
}

```

http://blog.csdn.net/abc_670

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，



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

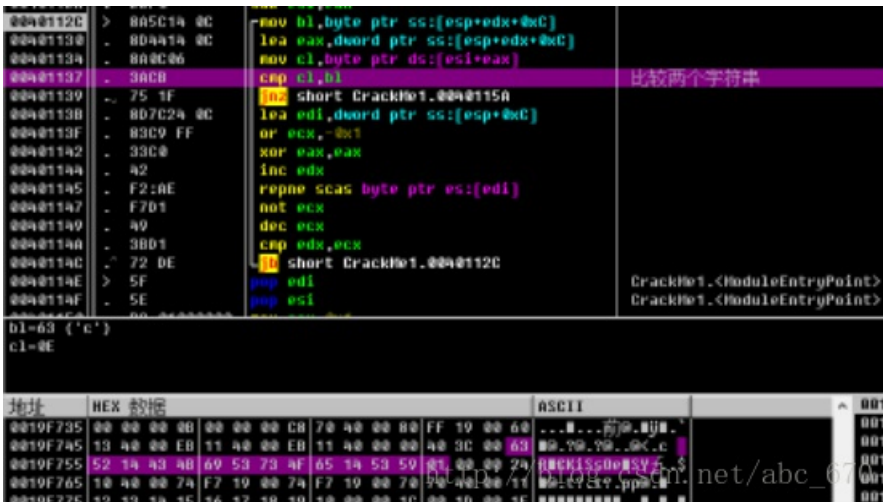
```

lea    edi, [esp+1Ch+var_10]
or     ecx, 0FFFFFFFh      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="")
```