【二进制】【WP】MOCTF逆向题解

转载

weixin_30684743 于 2019-09-12 20:54:00 发布 117 收藏
文章标签： 数据结构与算法
原文链接：http://www.cnblogs.com/yichen115/p/11515136.html

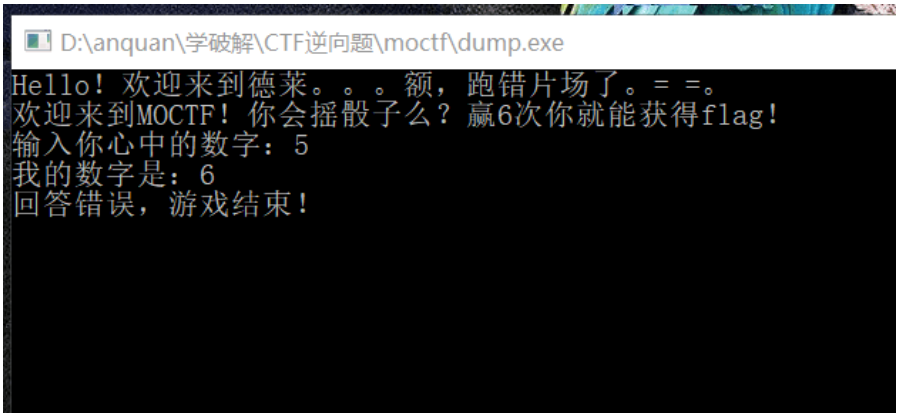
moctf 逆向第一题：SOEASY

这个是个 64 位的软件，OD 打不开，只能用 IDA64 打开，直接搜字符串（shift+F12）就可以看到

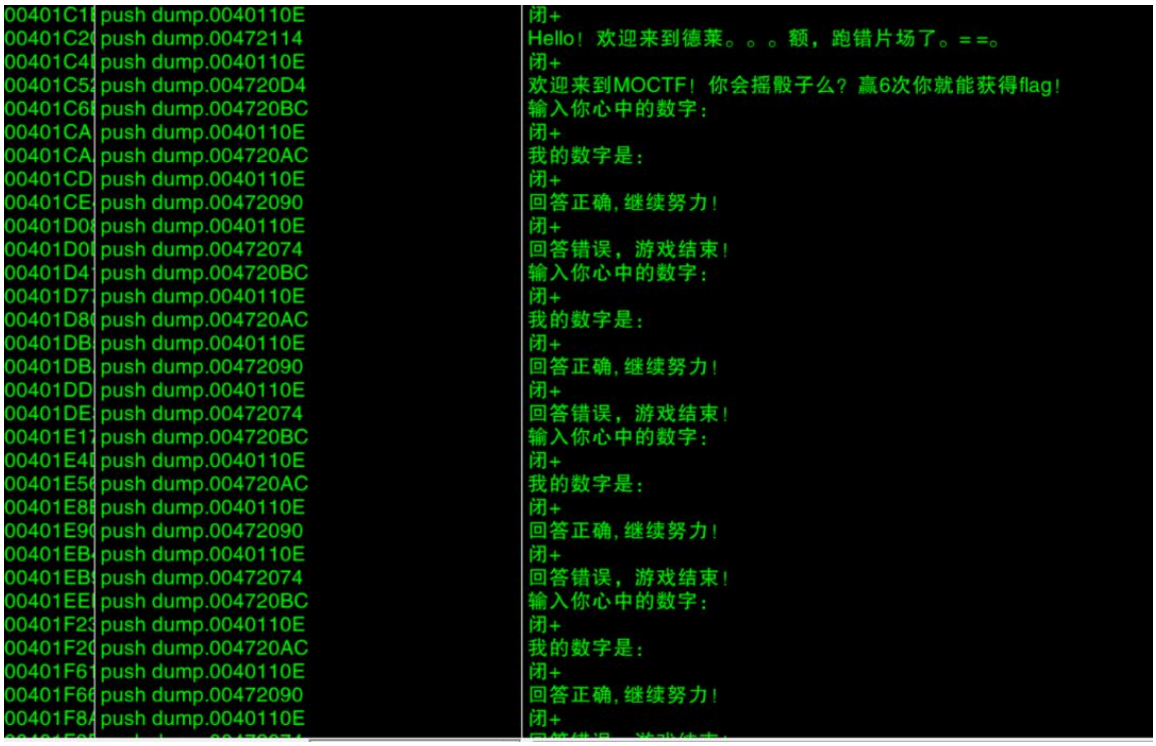
moctf 逆向第二题：跳跳跳

这个题当初给了初学逆向的我很大的成就感，当时就学了改指令爆破，根本不会分析算法，这就能做出一道题还是很舒服的

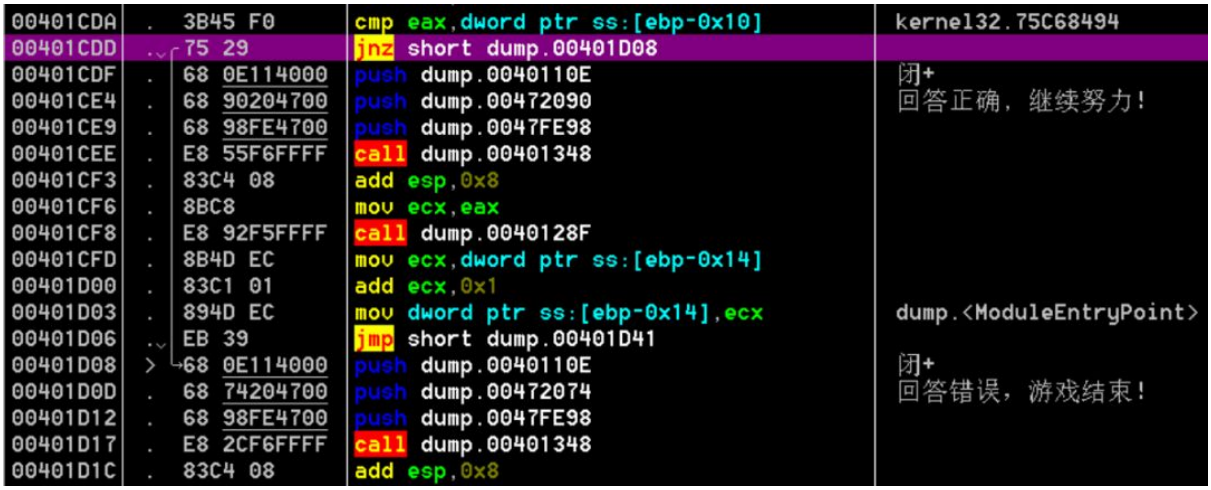
打开程序，是个猜数游戏



载入 OD 搜索字符串



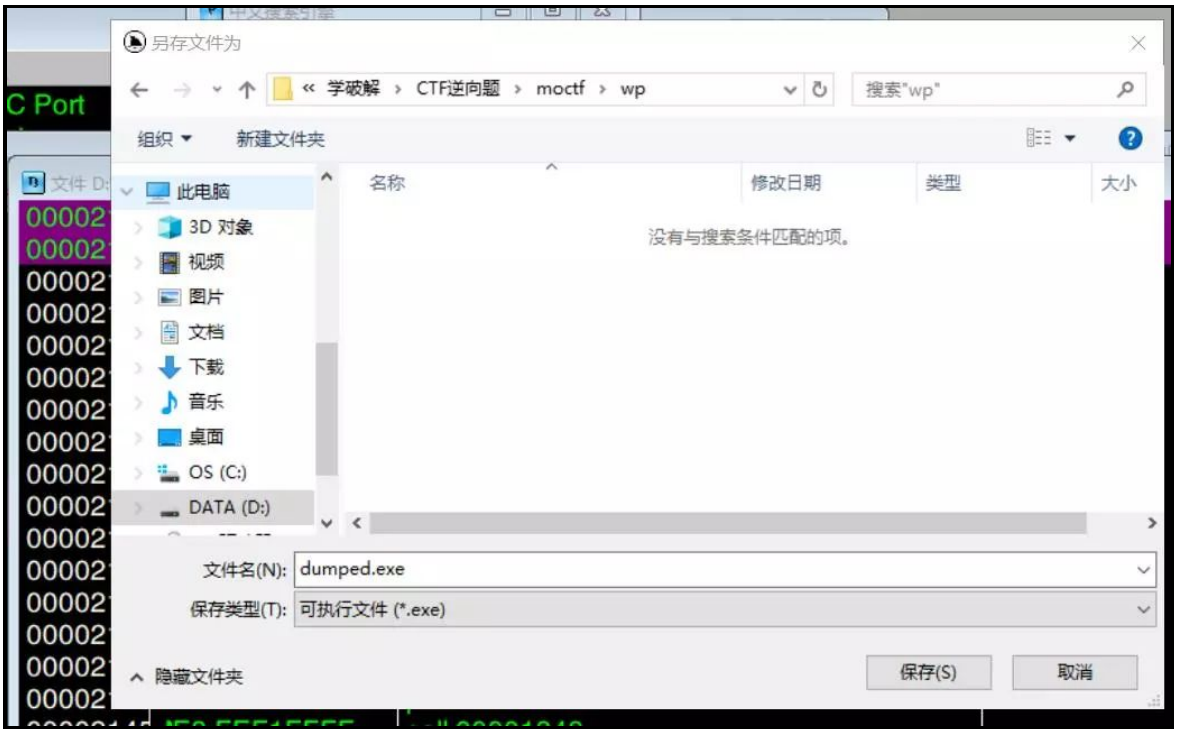
双击跟过去看看，很清晰了，那个 jnz 是关键，只要不让这个跳转实现就正确



那就直接把他 NOP 掉吧

```
.  3B45 F0      cmp eax,dword ptr ss:[ebp-0x10]   kernel32.75C68494
   90           nop
   90           nop
.  68 0E114000  push dump.0040110E                闭+
.  68 90204700  push dump.00472090                回答正确，继续努力!
.  68 98FE4700  push dump.0047FE98
.  E8 55F6FFFF  call dump.00401348
.  83C4 08      add esp,0x8
.  8BC8         mov ecx,eax
.  E8 92F5FFFF  call dump.0040128F
.  8B4D EC      mov ecx,dword ptr ss:[ebp-0x14]
.  83C1 01      add ecx,0x1
.  894D EC      mov dword ptr ss:[ebp-0x14],ecx   dump.<ModuleEntryPoint>
.  EB 39        jmp short dump.00401D41
>  68 0E114000  push dump.0040110E                闭+
.  68 74204700  push dump.00472074                回答错误，游戏结束!
.  68 98FE4700  push dump.0047FE98
.  E8 2CF6FFFF  call dump.00401348
.  83C4 08      add esp,0x8
.  8BC8         mov ecx,eax
```

保存出来



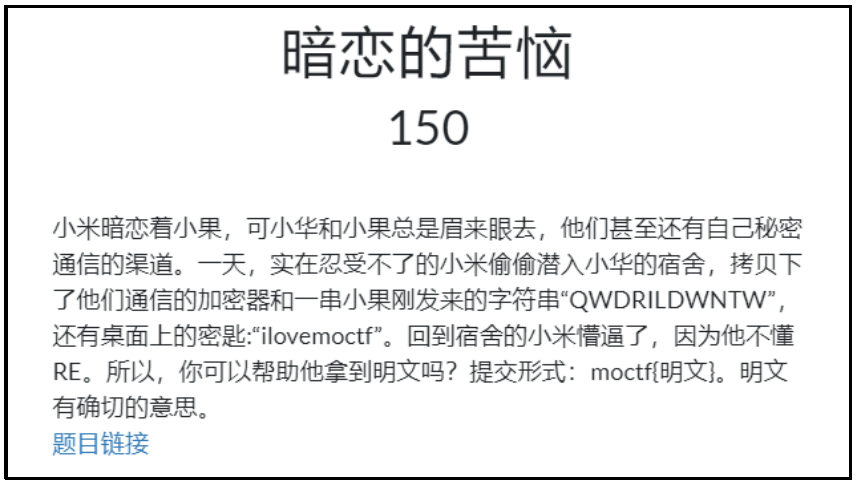
随便输入，最后还是可以成功出来 flag

```
Hello! 欢迎来到德莱。。。额，跑错片场了。= =。
欢迎来到MOCTF! 你会摇骰子么？赢6次你就能获得flag!
输入你心中的数字：5
我的数字是：1
回答正确，继续努力!
输入你心中的数字：3
我的数字是：2
回答正确，继续努力!
输入你心中的数字：1
我的数字是：2
回答正确，继续努力!
输入你心中的数字：4
我的数字是：6
回答正确，继续努力!
输入你心中的数字：5
我的数字是：6
回答正确，继续努力!
输入你心中的数字：2
我的数字是：4
回答正确，继续努力!
哇呜~命运之子，给你flag~
ZHUtcF8xc19FYVN577yB
10秒后自动退出
```

这个 falg 是 base64 加密的，之前在这里提交好多次总是不对，之后试了试才发现编码了

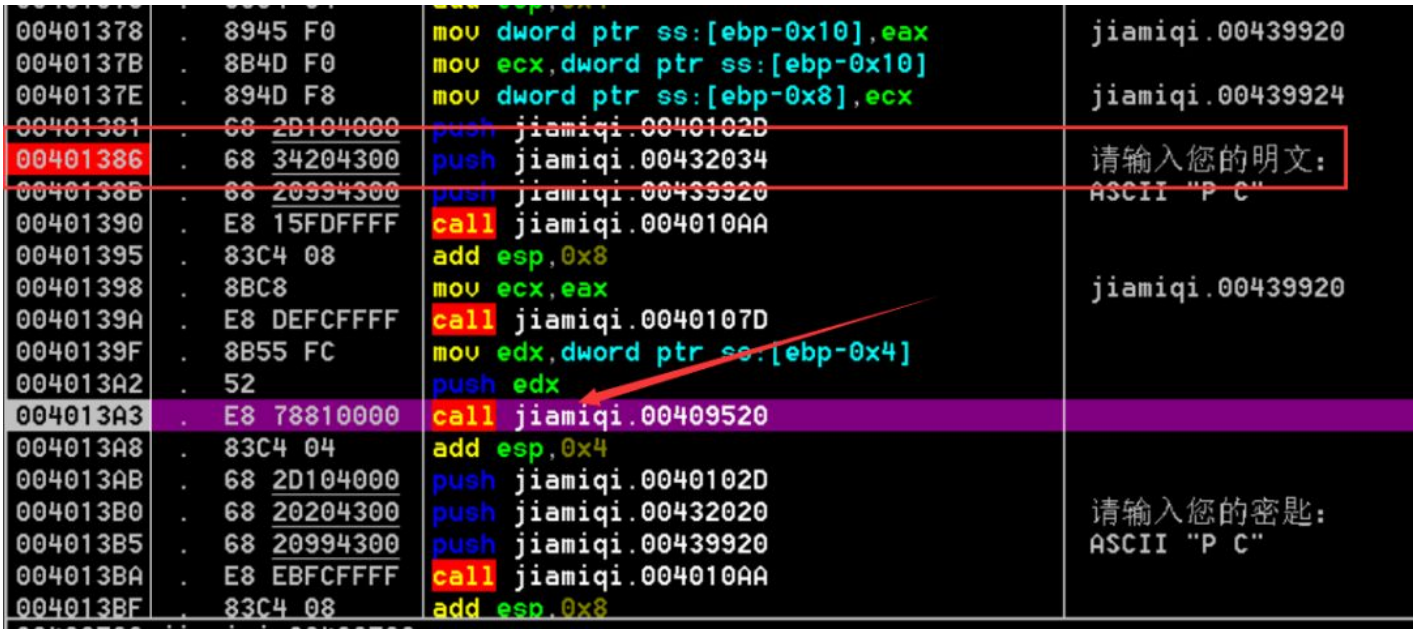
moctf 逆向第三题：暗恋的苦恼

当时可是费了好大劲才看懂这个伪代码的 (不知道现在还看不看得懂)，这次用 OD 分析的还算顺利，但是也花了一个多小时……

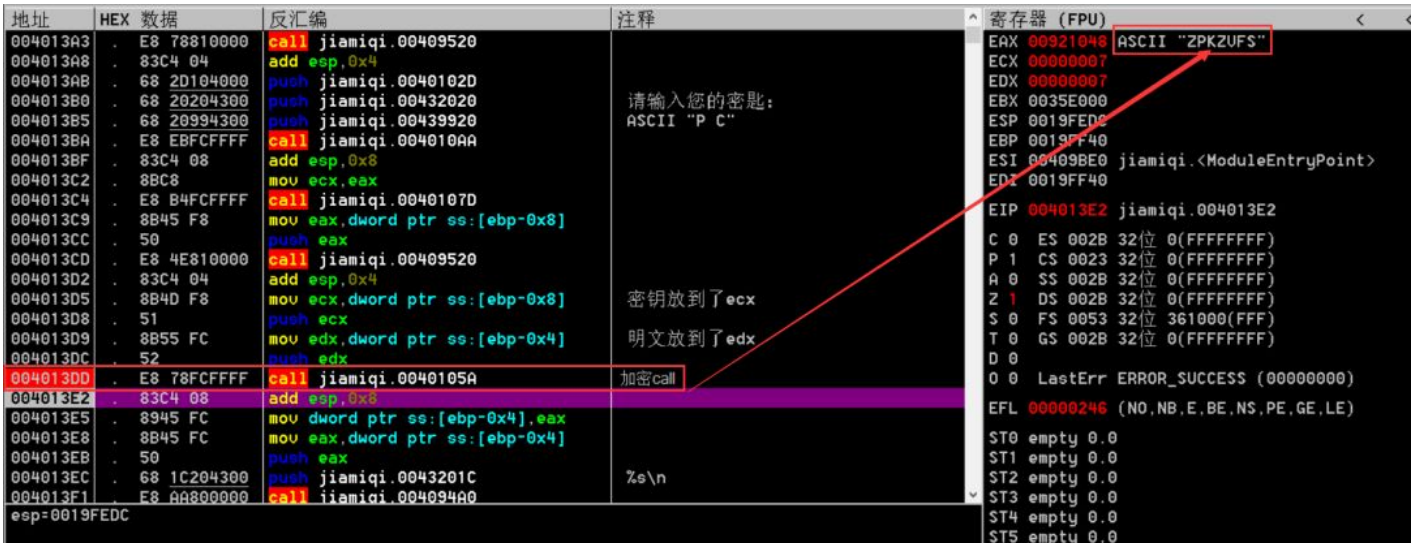


先来看一下流程：

载入 OD，通过字符串找到获取输入的地方，在下图箭头所指的地方程序运行起来等待输入



可以看到，经过一个 CALL 之后出现了加密后的字符串，那就跟进这个 CALL 看一下



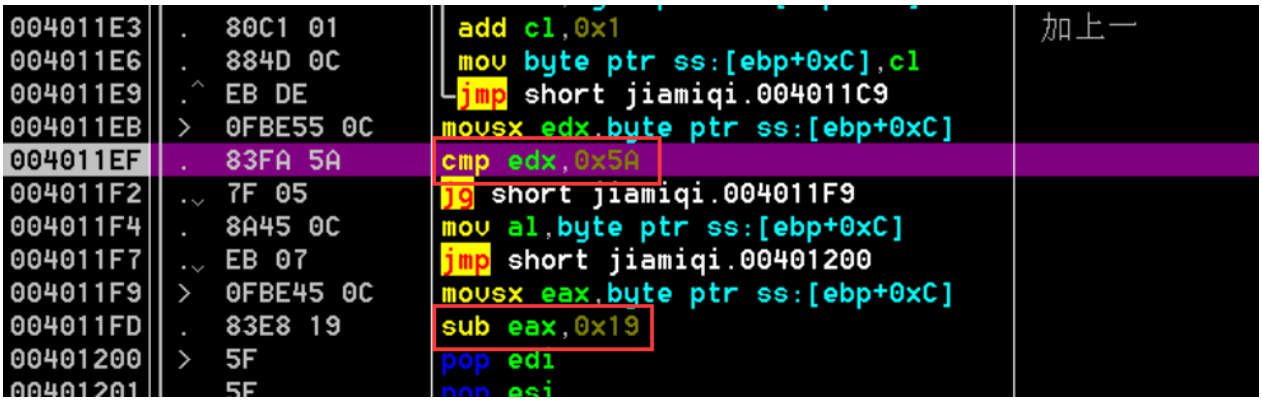
之后还得再进一个 CALL 才能发现程序算法 (程序是一个字一个字加密的刚开始会把输入的全部转换成大写)

程序算法：

1. 逐个取字符，转成大写的，然后用大写形式的 ASCII 码减去 41 (十六进制) 得到结果 (十六进制)

2. 然后用密钥的大写形式加上得到的结果

PS. 这里做了一些处理，如果加上以后结果超出了大写字母 Z (0x5A)，那么会再减去 19 (十六进制)



然而还有个问题，我发现给出的密文长度是比密钥长的，那怎么办？记得之前看的 writeup 说密钥循环使用的，即：ilovemoctfilovemoctf，这样一直排下去，不管多出来多少密文都可以加密

为了验证这个问题，我做了一个实验

这样输入的明文个数比密钥多一个，而且多的是与第一个一样的，我们看一下是不是按照那样循环是用的

加密结果让我很欣慰，证明我记性还不错 

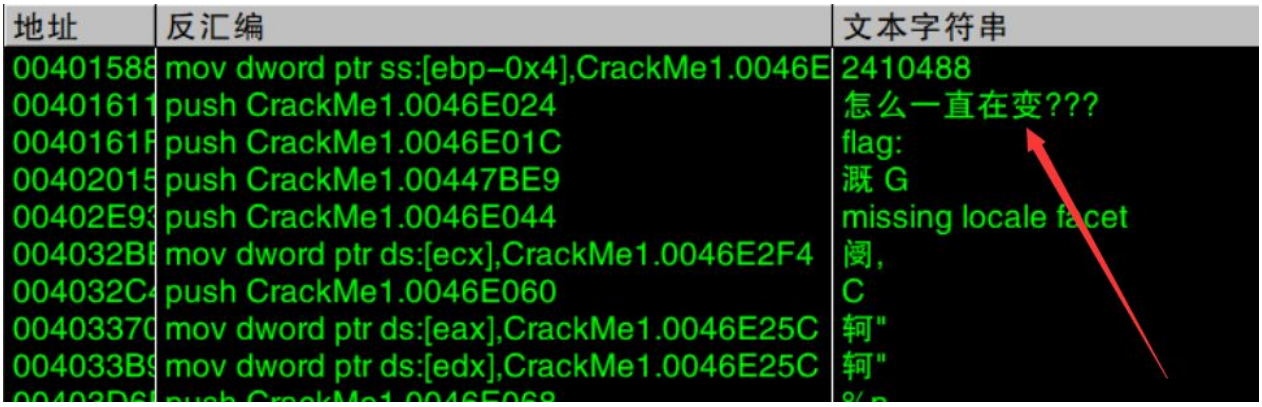
参考脚本，这或许会帮你更好的认识这个程序的流程

```
l="QWDRILDWNTW"
key="ILOVEMOCTFI"
flag=""
for i in range(len(l)):
    for temp1 in range(65,91):
        temp2=temp1-65
        if(temp2+ord(key[i])>90):
            temp2-=25
        if(temp2+ord(key[i])==ord(l[i])):
            flag+=chr(temp1)
print(flag)
```

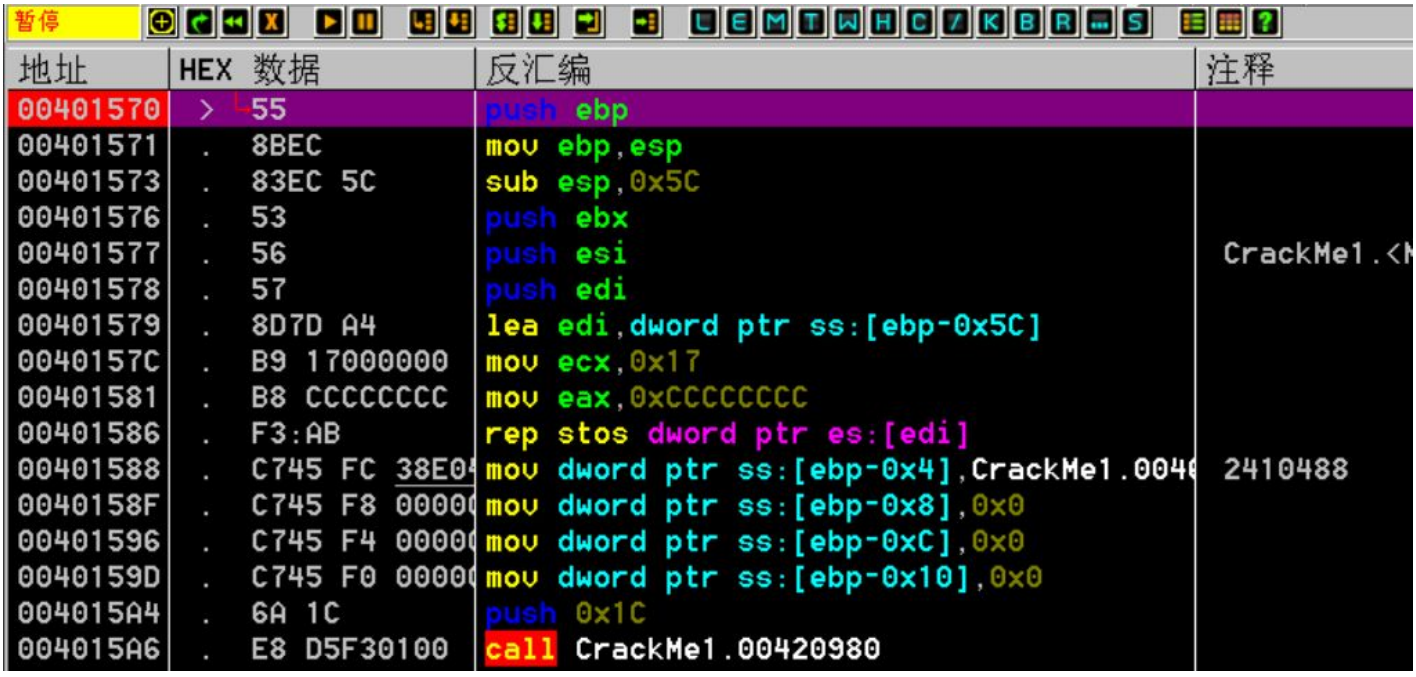
moctf 逆向第四题：crackme1

打开后啥都不要求输入，出现一串字符，说是 flag，但是每次打开都不一样

载入 OD，通过字符串找到



但是因为不知道该怎么分析，干脆直接段首下断，然后单步跟踪



找到算法部分，算法：

1. 逐个取定义好的 “2410488”

2. 乘以 2 后减去 60 (十六进制)

3. 除以 4 之后加上 3

4. 对 10 取余，得到结果

```
004015E0  .  8B4D FC      mov ecx,dword ptr ss:[ebp-0x4]
004015E3  .  034D F0      add ecx,dword ptr ss:[ebp-0x10]
004015E6  .  0FBE11       movsx edx,byte ptr ds:[ecx]        逐个放到edx
004015E9  .  8D4412 A0    lea eax,dword ptr ds:[edx+edx-0x60] 两倍的edx-60放到eax
004015ED  .  99           cdq
004015EE  .  83E2 03      and edx,0x3
004015F1  .  03C2         add eax,edx
004015F3  .  C1F8 02      sar eax,0x2                        右移两位，相当于除以4
004015F6  .  83C0 03      add eax,0x3                        再加上3
004015F9  .  99           cdq
004015FA  .  B9 0A000000  mov ecx,0xA
004015FF  .  F7F9         idiv ecx                           除以A
00401601  .  8B45 F0      mov eax,dword ptr ss:[ebp-0x10]
00401604  .  8B4D EC      mov ecx,dword ptr ss:[ebp-0x14]
```

在刚开始一直不知道这里的数是怎么出来的，在心里把 [edx+edx-0x60] 当作一个地址了，后来发现，这他妈的不是存放的刚赋值的字符嘛！！

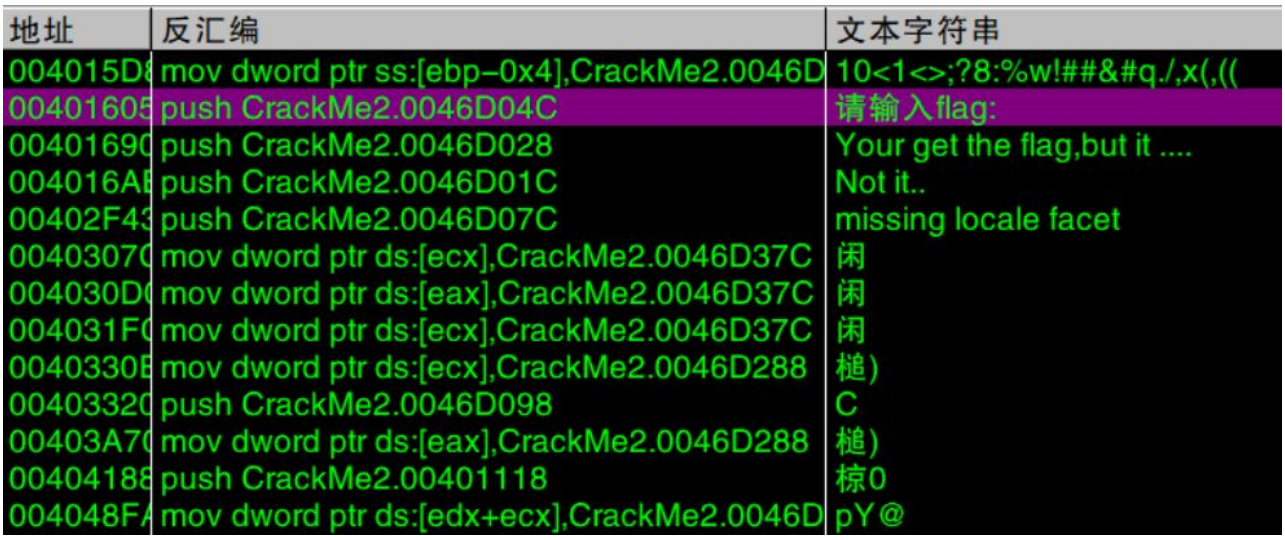
```
004015D2  .  83C2 01      add edx,0x1
004015D5  .  8955 F0      mov dword ptr ss:[ebp-0x10],edx
004015D8  >  8B45 F0      mov eax,dword ptr ss:[ebp-0x10]
004015DB  .  3B45 E8      cmp eax,dword ptr ss:[ebp-0x18]
004015DE  .v 7D 2C        jge short CrackMe1.0040160C
004015E0  .  8B4D FC      mov ecx,dword ptr ss:[ebp-0x4]
004015E3  .  034D F0      add ecx,dword ptr ss:[ebp-0x10]
004015E6  .  0FBE11       movsx edx,byte ptr ds:[ecx]        逐个放到edx
004015E9  .  8D4412 A0    lea eax,dword ptr ds:[edx+edx-0x60] 两倍的edx-60放到eax
004015ED  .  99           cdq
004015EE  .  83E2 03      and edx,0x3
004015F1  .  03C2         add eax,edx
004015F3  .  C1F8 02      sar eax,0x2                        右移两位，相当于除以4
004015F6  .  83C0 03      add eax,0x3                        再加上3
004015F9  .  99           cdq
004015FA  .  B9 0A000000  mov ecx,0xA
004015FF  .  F7F9         idiv ecx                           除以A
00401601  .  8B45 F0      mov eax,dword ptr ss:[ebp-0x10]
```

写出脚本，跑出 flag：4533577

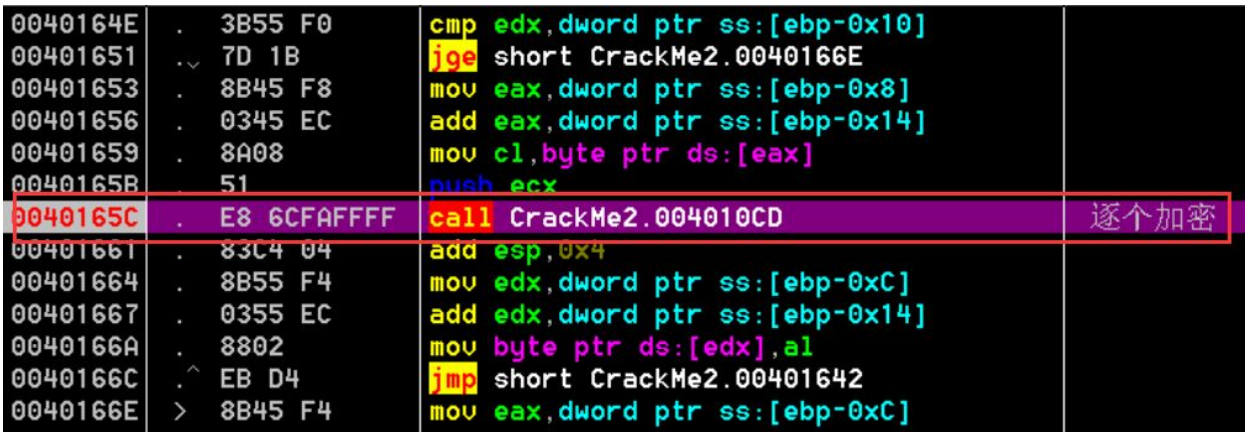
---

moctf 逆向第五题：crackme2

通过字符串定位获取输入的位置



然后按步跟踪找到加密函数

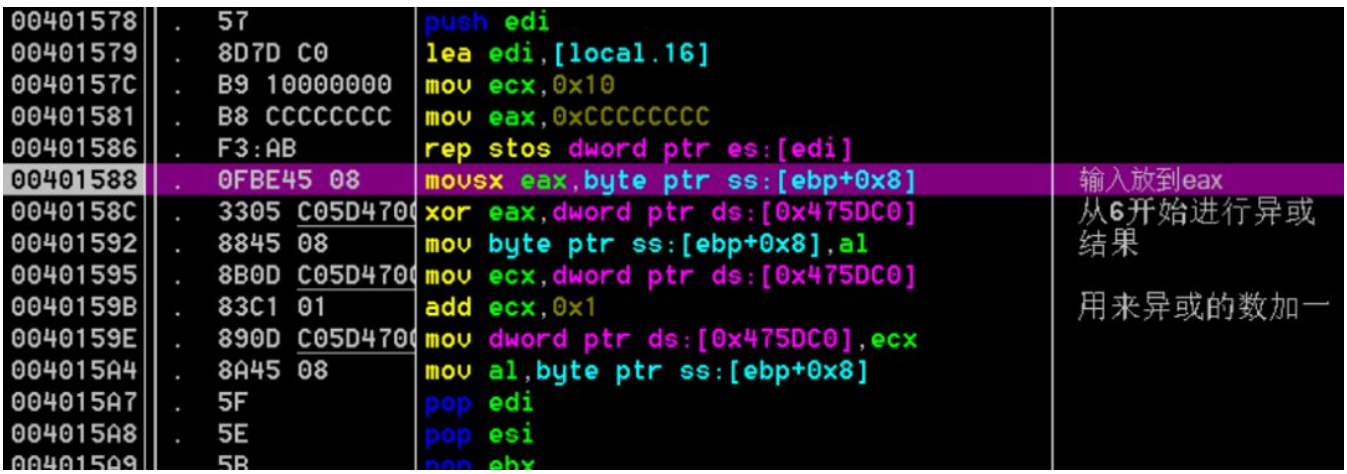


步入加密 CALL，分析算法：

1. 依次取输入的字符串

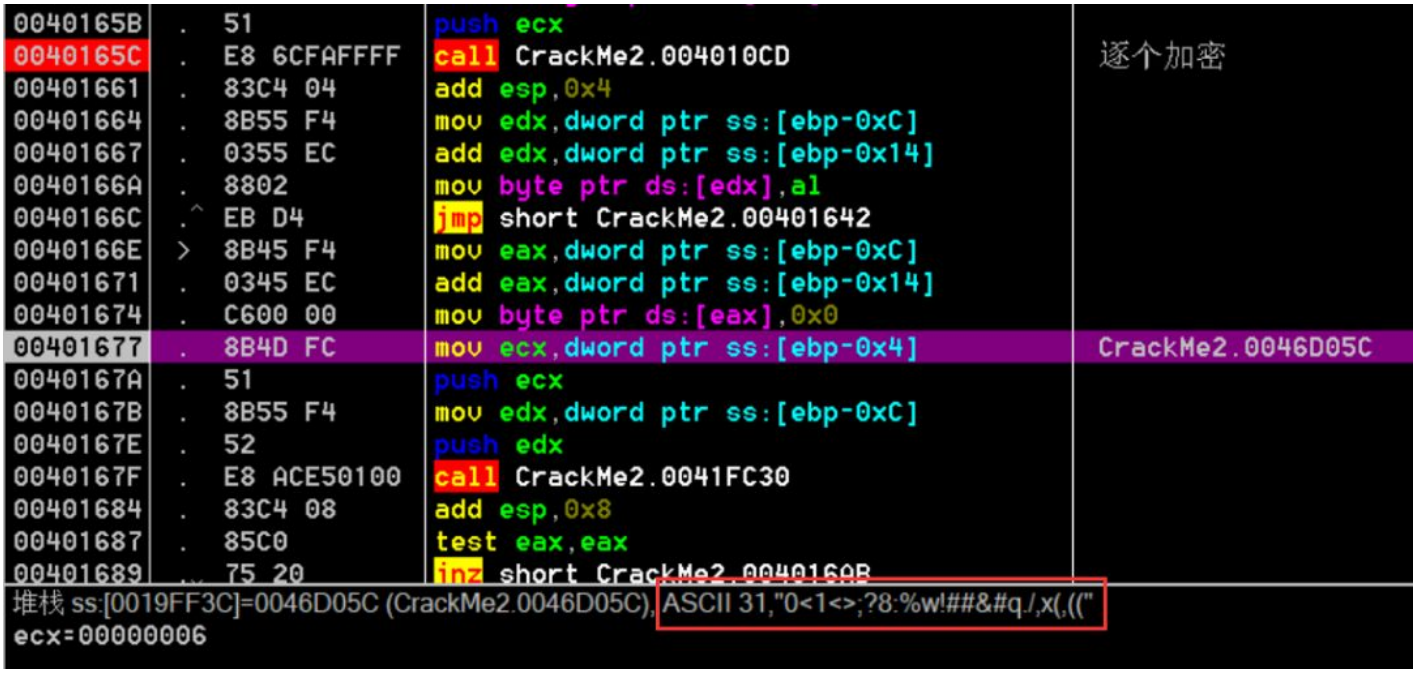
2. 从 6 开始，与输入的字符串进行异或

3. 异或结果与定义好的字符串进行比较

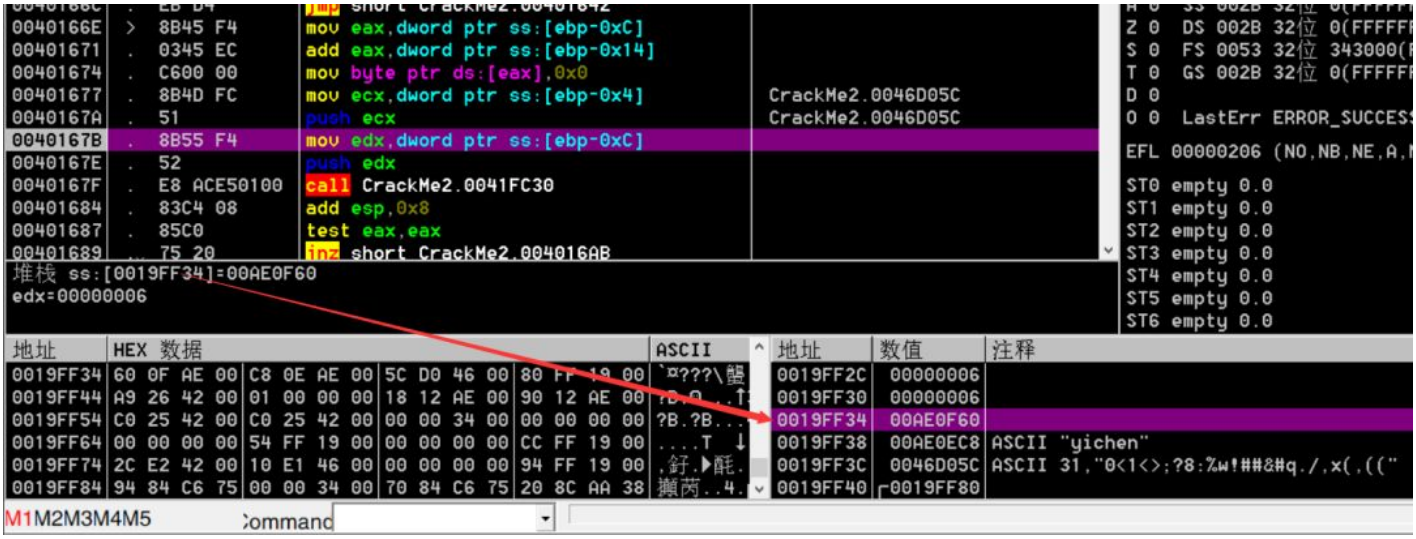


红框圈出来的是定义好的字符串

0<1<>;?8:%w!##&#q./,x(,((



下面的那个 CALL 是进行对比的



可以看到我们输入的进行加密后是 7F，而规定的应该是 31

```
0041FC38   .  F7C2 03000000  test edx,0x3
0041FC3E   .v 75 3C          jnz short CrackMe2.0041FC7C
0041FC40   >  8B02           mov eax,dword ptr ds:[edx]
0041FC42   .  3A01           cmp al,byte ptr ds:[ecx]
0041FC44   .v 75 2E          jnz short CrackMe2.0041FC74
0041FC46   .  0AC0           or al,al
0041FC48   .v 74 26          je short CrackMe2.0041FC70
0041FC4A   .  3A61 01        cmp ah,byte ptr ds:[ecx+0x1]
0041FC4D   .v 75 25          jnz short CrackMe2.0041FC74
0041FC4F   .  0AE4           or ah,ah
ds:[0046D05C]=31 ('1')
al=7F
```

写出脚本跑出应该输入的字符串

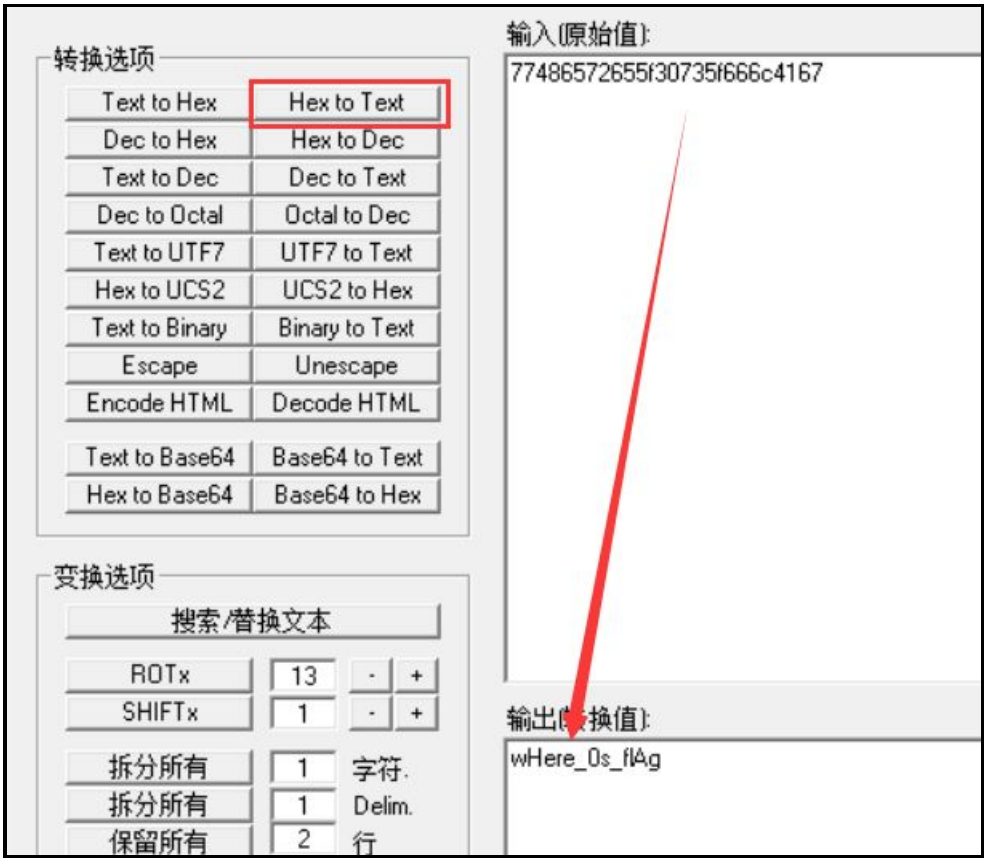
```
a = 6
s = "10<1<>;?8:%w!##&#q./,x(,(("
flag = ""

for i in range(len(s)):
    result_temp = ord(s[i])^a
    a += 1
    flag += chr(result_temp)
    print(flag)
```

```
77486572655
77486572655f
77486572655f3
77486572655f30
77486572655f307
77486572655f3073
77486572655f30735
77486572655f30735f
77486572655f30735f6
77486572655f30735f66
77486572655f30735f666
77486572655f30735f666c
77486572655f30735f666c4
77486572655f30735f666c41
77486572655f30735f666c416
77486572655f30735f666c4167
```

输入正确的后在进行测试，nice !!!

```
0041FC34   .  8B4C24 08      mov ecx,dword ptr ss:[esp+0x8]
0041FC38   .  F7C2 03000000  test edx,0x3
0041FC3E   .v 75 3C          jnz short CrackMe2.0041FC7C
0041FC40   >  8B02           mov eax,dword ptr ds:[edx]
0041FC42   .  3A01           cmp al,byte ptr ds:[ecx]
0041FC44   .v 75 2E          jnz short CrackMe2.0041FC74
0041FC46   .  0AC0           or al,al
0041FC48   .v 74 26          je short CrackMe2.0041FC70
0041FC4A   .  3A61 01        cmp ah,byte ptr ds:[ecx+0x1]
0041FC4D   .v 75 25          jnz short CrackMe2.0041FC74
0041FC4F   .  0AE4           or ah,ah
0041FC51   .v 74 1D          je short CrackMe2.0041FC70
0041FC53   .  C1E8 10        shr eax,0x10
ds:[0046D05C]=31 ('1')
al=31 ('1')
```

然而，这并不是 flag，需要将它转换成字符串形式



ps. 这是 N 个月前在公众号发的文章，转过来博客，主要是希望能够让更多人看见

转载于:https://www.cnblogs.com/yichen115/p/11515136.html