实验吧逆向之迷路

原创

钞sir 于 2018-10-20 18:40:47 发布 3683 收藏
分类专栏： CTF RE
钞sir
本文链接：https://blog.csdn.net/qq_40827990/article/details/83212779

CTF 同时被 2 个专栏收录
22 篇文章 2 订阅
订阅专栏

RE
5 篇文章 0 订阅
订阅专栏

后记：写了篇博客一个月后我重新分析这道题发现，这个题的flag实验吧可以提交，但是这个程序本身过不了，后来研究一下发现，当程序输入的字符串中含有‘0’时，算法不太对了，主要是v5的值受到了字符0的影响.....

今天来看看逆向中一些比较有脑洞的思路方法，以实验吧中的一道名为迷路(http://www.shiyanbar.com/ctf/1915)的题为例.

首先运行程序是这个样子的：

然后随便输入一串字符后，有窗口提示：

然后载入IDA，查看一下字符串，搜索Pls Try ag@in!等字样，再结合ollydbg动态分析之后，明白了程序的基本流程：

首先将获取的输入经atol()函数转为数字，并与0x92381221做比较，之后对输入的字符串做md5并将结果与4850B7446BBB20AAD140E7B0A964A57D进行比较，如果都相同就成功；

```
text:00401EDD                lea     ecx, [ebp-10h]
text:00401EE0                push    ecx             ; struct CString *
text:00401EE1                lea     ecx, [esi+0ACh] ; this
text:00401EE7                call    ?GetWindowTextA@CWnd@@QBEXAAVCString@@@Z ; CWnd::GetWindowTextA(CString &)
text:00401EEC                mov     edx, [ebp-10h]
text:00401EEF                push    edx             ; char *
text:00401EF0                call    _atoi           ; 通过atoi()函数将字符串转换为数字
text:00401EF5                mov     edi, ds:MessageBoxA
text:00401EFB                add     esp, 4
text:00401EFE                cmp     eax, 92381221h  ; 将数字与0x92381221比较
text:00401F03                jnz     short loc_401F47
text:00401F05                push    ecx
text:00401F06                lea     eax, [ebp-10h]
text:00401F09                mov     ecx, esp
text:00401F0B                mov     [ebp-14h], esp
text:00401F0E                push    eax
text:00401F0F                call    sub_4140E9
text:00401F14                mov     ecx, esi
text:00401F16                call    sub_401990      ; 将输入的字符串MD5加密
text:00401F1B                push    ecx             ; unsigned __int8 *
text:00401F1C                add     esi, 64h
text:00401F1F                mov     ecx, esp
text:00401F21                mov     [ebp-14h], esp
text:00401F24                push    esi
text:00401F25                call    sub_4140E9
text:00401F2A                call    sub_401DF0      ; 将加密的字符串与4850B7446BBB20AAD140E7B0A964A57D比较
text:00401F2F                add     esp, 4
text:00401F32                cmp     eax, 1
text:00401F35                jnz     short loc_401F47
text:00401F37                push    0               ; uType
text:00401F39                push    offset Caption  ; "[ 四叶草安全提示您(FourLeafCloverNetwor"...
text:00401F3E                push    offset Text     ; "你赢了!Y0u W1n"
text:00401F43                push    0               ; hWnd
text:00401F45                call    edi ; MessageBoxA
```



```
64:A1 000000[ mov     eax, dword ptr fs:[0]
6A FF         push    -0x1
68 787B4100   push    00417B78
50            push    eax
64:8925 0000[ mov     dword ptr fs:[0], esp
8B4424 10     mov     eax, dword ptr ss:[esp+0x10]
68 50F14100   push    0041F150                        4850B7446BBB20AAD140E7B0A964A57D
50            push    eax
E8 F20E0000   call    00402D06
83C4 08       add     esp, 0x8
C74424 08 FF[ mov     dword ptr ss:[esp+0x8], -0x1
```

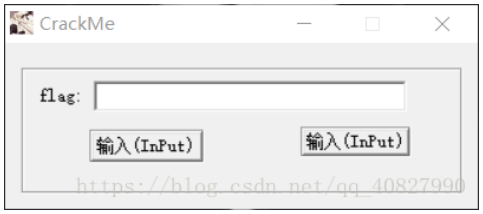

但是查询4850B7446BBB20AAD140E7B0A964A57D对应的明文是sakjflks，既要等于92381221h又要是sakjflsk显然是不可能的，所以在这里面一定还有什么东西；

之后我们再OD中查看句柄窗口，果然发现这里有两个input，说明其中一个input被隐藏起来了：

```
00020838  ActiveMovie Window  Topmost                  04CF0000 00000100 00000224 560A6540 FilterGraphWindow
L0008083E Default IME         00020838                 8C000000          00000224 7667BD10 IME
000C07A4  CrackMe             Topmost                  94CA00C4 00050101 在       76667670 #32770
 00040780
 00060750 输入(InPut)         000C07A4 0000FFFF 50000007 00000004 在       766A55C0 Button
 0008080A Default IME         000C07A4 000003EC 40012100 00020004 在       766A55C0 Button
 0007071A MSCTFIME UI         0008080A                 8C000000          在       7667BD10 IME
 00110784 flag:               000C07A4 0000FFFF 50020000 00000004 在       75A816B0 MSCTFIME UI
 00110788 输入(InPut)         000C07A4 000003E8 50010001 00020004 在       76670E70 Static
 001C076C                     000C07A4 000003EB 50010A80 00000204 在       766A55C0 Button
 001D074C                     000C07A4 0000FFFF 40020000 00000004 在       7666D720 Edit
                                                                            76670E70 Static
```


在OD中重新加载程序，Ctrl+g搜索ShowWindow下断点，可以看到有一个按钮被hide了，把它的值00000000改成00000001即可；

```
0019F518  00412782  CALL 到 ShowWindow 来自 crackme.0041277C
0019F51C  000805D8  hWnd = 000805D8 ('输入(InPut) ',class='Button',parent=00050994)
0019F520  00000000  ShowState = SW_HIDE
0019F524  00401637  返回到 crackme.00401637 来自 crackme.0041276E
0019F528  00000000
```

隐藏的input被显示出来了；

然后我们Ctrl+g搜索GetWindowTextA下断点，然后重新输入flag，点击左边的input键，然后程序在0x00410A6B的位置停了下来；

然后一步一步的跟，发现这里是有个很关键的地方0x401F96，不注意就跳到失败的地方去了，IDA里f5看看0x401740里面是什么

```
00401F80  r$  51          push    ecx
00401F81  .   56          push    esi
00401F82  .   8BF1        mov     esi, ecx
00401F84  .   51          push    ecx
00401F85  .   8D46 5C     lea     eax, dword ptr ds:[esi+0x5C]
00401F88  .   8BCC        mov     ecx, esp
00401F8A  .   896424 08   mov     dword ptr ss:[esp+0x8], esp
00401F8E  .   50          push    eax
00401F8F  .   E8 55210100 call    004140E9
00401F94  .   8BCE        mov     ecx, esi
00401F96  .   E8 A5F7FFFF call    00401740
00401F9B  .   85C0        test    eax, eax
00401F9D  .v  74 33       je      short 00401FD2
```

```
00401FA0  .   8D56 60     lea     edx, dword ptr ds:[esi+0x60]
00401FA3  .   8BCC        mov     ecx, esp
00401FA5  .   896424 08   mov     dword ptr ss:[esp+0x8], esp
00401FA9  .   52          push    edx
00401FAA  .   E8 3A210100 call    004140E9
00401FAF  .   8BCE        mov     ecx, esi
00401FB1  .   E8 AAF8FFFF call    00401860
00401FB6  .   85C0        test    eax, eax
00401FB8  .v  74 18       je      short 00401FD2
```

```
signed int __thiscall sub_401740(void *this, int a2)
{
  void *v2; // esi@1
  signed int result; // eax@11

  v2 = this;
  if ( *(_DWORD *)(a2 - 8) < 0x27u              // 判断flag的基本格式：OOCTF{}
    || *(_BYTE *)a2 != 'O'
    || *(_BYTE *)(a2 + 1) != 'O'
    || *(_BYTE *)(a2 + 2) != 'C'
    || *(_BYTE *)(a2 + 3) != 'T'
    || *(_BYTE *)(a2 + 4) != 'F'
    || *(_BYTE *)(a2 + 5) != '{'
    || *(_BYTE *)(a2 + 38) != '}' )
  {
    sub_414374(&a2);
    result = 0;
  }
  else
  {
    sub_40F42B((CString *)&a2, 38, 1);
    sub_40F42B((CString *)&a2, 0, 6);
    if ( *(_DWORD *)(a2 - 8) )
      sub_4143CB((CString *)((char *)v2 + 96), (int)&a2);
    CString::Empty((CString *)((char *)v2 + 92));
    sub_414374(&a2);
    result = 1;
  }
  return result;
}
```

可以看到输入需要满足格式OOCTF{里面有32位}，满足这些条件后，程序运行到这里，调用0x00401860处函数的函数处理后，得到另外一串字符串，且传入的参数正是{}里面的32位字符串，最后处理结果由0x00402d06函数判断是否等于b5h760h64R867618bBwB48BrW92H4w5r

```
if ( v5 < 0 )
{
  v5 = 3;                                   // 这里对V5有处理，第一次为0x3，第二次之后就为0x5
  v12 = 3;
}
v6 = *(_DWORD *)(a2 - 8) - dword_41F130 - 2;
if ( *(_DWORD *)(a2 - 8) > 0 )
{
  do
  {
    v7 = *(_BYTE *)(v3 + v4);
    v8 = 0;
    if ( v7 > 57 || v7 < 48 )               // 判断是否为字母
    {
      if ( v7 > 122 || v7 < 97 )            // 将字母转换为字母表中对应的0~25之间的数字
      {
        if ( v7 <= 90 && v7 >= 65 )
          v7 -= 65;
      }
      else
      {
        v7 -= 97;
        v8 = 1;
      }
      v9 = (v6 + v5 * v7) % 26 + ((v6 + v5 * v7) % 26 < 0 ? 0x1A : 0);// v6=0x1C
      if ( v8 )
      {
        if ( v8 == 1 )
          LOBYTE(v9) = v9 + 97;
      }
      else
      {
        LOBYTE(v9) = v9 + 65;
      }
      *(_BYTE *)(v3 + v4) = v9;
      v5 = v12;
```

该函数对于字符串中的数字不做处理，字母转化成字母表中对应的0-25之间的数值，之后将得到的v9再转化成相应的字母；

知道了算法之后，就可以倒退出flag了：

```
name="b5h760h64R867618bBwB48BrW92H4w5r"
lenth=len(name)
def fun(a,i):
        edx=0x5
        edi=0x1c
        if(i==0):
                edx=0x3 #第一次调用为0x3
        eax=a*edx
        eax=eax+edi
        edx=eax%0x1A
        #print(edx)
        return edx


key=""
le=0
while(le!=lenth):
        for i in range(ord('0'),ord('z')):
                if(i>=ord('A') and i<=ord('Z')):
                        a=i-ord('A')
                        b=fun(a,i)+ord('A')
                        if(name[le]==chr(b)):
                                key=key+chr(i)
                                le=le+1
                                break
                elif(i>=ord('a') and i<=ord('z')):
                        a=i-ord('a')
                        b=fun(a,i)+ord('a')
                        if(name[le]==chr(b)):
                                key=key+chr(i)
                                le=le+1
                                break
                elif(i>=ord('0') and i<=ord('9')):
                        if(name[le]==chr(i)):
                                key=key+chr(i)
                                le=le+1
                                break
                else:
                        b=fun(i,i)+ord('A')
                        if(name[le]==chr(b)):
                                key=key+chr(i)
                                le=le+1
                                break

print("flag:  OOCTF{"+key+"}")
```

代码的处理和OD中跟出来的步骤基本一样，fun函数的结果加ord('A')或ord('a')就是v9；