攻防世界 三 （进阶篇）

原创

chan3301 于 2019-06-09 16:21:30 发布 823 收藏

分类专栏： 逆向题目练习 文章标签： 逆向 攻防世界

本文链接：https://blog.csdn.net/sjt670994562/article/details/91195673

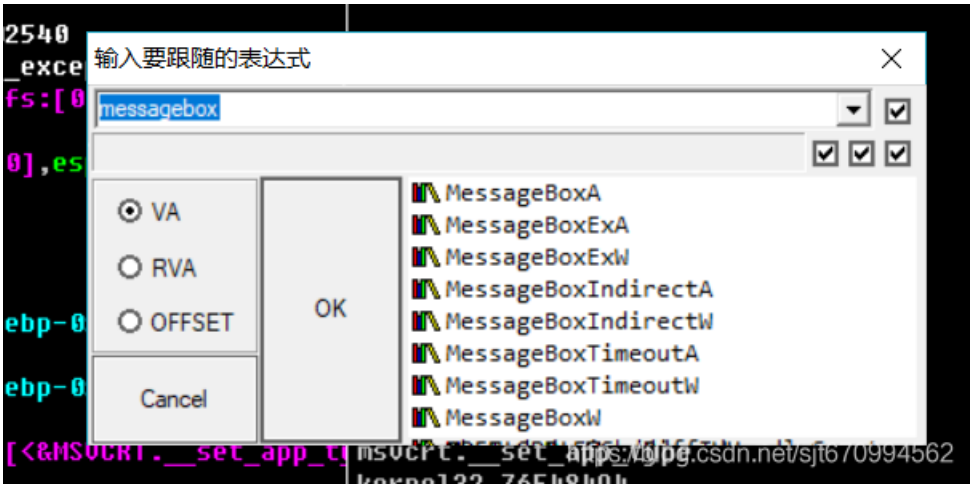
逆向题目练习 专栏收录该内容

16 篇文章 1 订阅

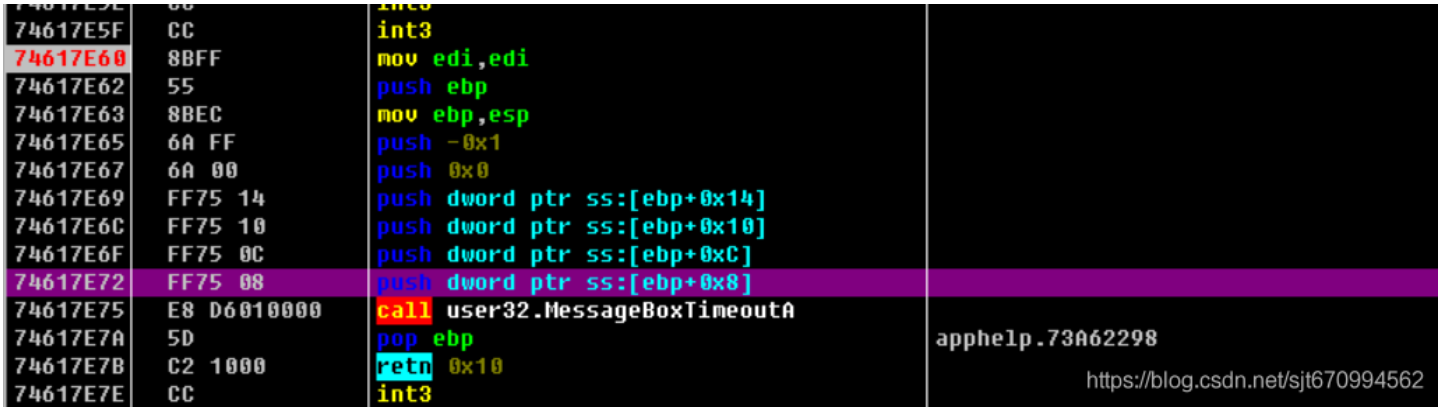
订阅专栏

CRACKME

这一道题对OD的使用进行了一定的训练，虽然相比真真正的使用还是差很多，首先是一个注册问题的exe文件，有窗口，打开OD寻找messagebox。



尝试设置断点



然后尝试打开软件点击注册，发现到了断点，说明断成功了。然后这时候看栈口，会发现之前压进来的数据有失败的提示字面，说明我们没有到判断正误的界面找到最下面的提示字面的地址，跳转，发现函数，设置断点

```
0019F23C  00401759  返回到 0dc17c69.00401759 来自 <jmp.&MFC42.#CWnd::MessageBoxA_4224>
0019F240  0019F258  ASCII "哎，注册码错了，你得换个新的呦 !"
0019F244  00000000
```

```
00401751   51              push ecx                                 user32.MessageBoxA
00401752   8BCB            mov ecx,ebx
00401754   E8 37020000      call <jmp.&MFC42.#CWnd::MessageBoxA_422
00401759   5F              pop edi                                  apphelp.73A62298
0040175A   5E              pop esi                                  apphelp.73A62298
0040175B   5B              pop ebx                                  apphelp.73A62298
0040175C   83C4 24         add esp,0x24
0040175F   C3              retn
00401760   8B41 20         mov eax,dword ptr ds:[ecx+0x20]
00401763   6A 00           push 0x0
00401765   50              push eax
00401766   FF15 E8214000   call dword ptr ds:[<&USER32.EnableWindo  user32.EnableWindow
0040176C   C3              retn
0040176D   90              nop
```


再次运行，然后发现不行，还是没到选择判断的地方继续如刚才的方法进行跳转，直接跳到一个比较前的地方，这时候看到一个updata说明是判断之前的汇编，所以单步往后走就可以了

```
004015DE   90              nop
004015DF   90              nop
004015E0   51              push ecx
004015E1   56              push esi
004015E2   8BF1            mov esi,ecx
004015E4   6A 01           push 0x1
004015E6   E8 9F030000      call <jmp.&MFC42.#CWnd::UpdateData_6334
004015EB   8B8E 94020000   mov ecx,dword ptr ds:[esi+0x294]
004015F1   8D86 94020000   lea eax,dword ptr ds:[esi+0x294]
004015F7   8379 F8 21      cmp dword ptr ds:[ecx-0x8],0x21
004015FB   75 22           jnz short 0dc17c69.0040161F
004015FD   51              push ecx
004015FE   8BCC            mov ecx,esp
00401600   896424 08       mov dword ptr ss:[esp+0x8],esp
```


遇到跳转点，猜测是后面又都有函数，猜测应该是这里，attach掉这个jnz

```
004015F1   8D86 94020000   lea eax,dword ptr ds:[esi+0x294]
004015F7   8379 F8 21      cmp dword ptr ds:[ecx-0x8],0x21
004015FB   75 22           jnz short 0dc17c69.0040161F
004015FD   51              push ecx
004015FE   8BCC            mov ecx,esp
00401600   896424 08       mov dword ptr ss:[esp+0x8],esp
00401604   50              push eax
00401605   E8 7A030000      call <jmp.&MFC42.#CString::CString_535>
0040160A   8BCE            mov ecx,esi
0040160C   E8 1F000000      call 0dc17c69.00401630
00401611   84C0            test al,al
00401613   74 0A           je short 0dc17c69.0040161F
00401615   8BCE            mov ecx,esi
00401617   E8 C4000000      call 0dc17c69.004016E0
0040161C   5E              pop esi                                  0dc17c69.004022F8
0040161D   59              pop ecx                                  0dc17c69.004022F8
0040161E   C3              retn
0040161F   8BCE            mov ecx,esi
00401621   E8 FA000000      call 0dc17c69.00401720
```


单步进入函数0dc17c69.00401630，看到cmp就有希望，走到cmp果然，有flag的字符，然后循环走flag就好，这里可以直接用F4，更快捷一点，或者用ida动态可以直接显示所有字符更方便（因为作者还不懂怎么用OD显示所有寄存器字符，所以用的比较笨的办法）

```
00401663   8A0C0F          mov cl,byte ptr ds:[edi+ecx]
00401666   8D0416          lea eax,dword ptr ds:[esi+edx]
00401669   3A4C28 60       cmp cl,byte ptr ds:[eax+ebp+0x60]
0040166D   74 05           je short 0dc17c69.00401674
0040166F   C64424 13 00    mov byte ptr ss:[esp+0x13],0x0
```

最后的到flag

flag{The-Y3ll0w-turb4ns-Upri$ing}

ReverseMe-120

这里如果逻辑清晰一点就会很简单，扔到ida,看到比较函数，v13是关注点

```
43  }
44  v9 = strcmp(&v13, "you_know_how_to_remove_junk_code");
45  if ( v9 )
```

往上看，v13这一段数组做了异或

```
for ( ; v4 < v3; ++v4 )
  *(&v13 + v4) ^= 0x25u;
}
```

在往上看，就有一个v13给v7赋值，忽略，还有一个sub_401000函数，把函数目的得出来，逆回去就是我们输入的flah啦，然后分析函数，里面有一个byte_414E40的字符表，发现是乱序的base64表，但是此函数和base64不同，而且表是乱序，估计形式结果还是和base64一样，所以先异或在base64加密，得到flag

```
58        return -44;
59      if ( v10 > 0x7Fu )
60        return -44;
61      v11 = byte_414E40[v10];
62      if ( v11 == 127 || (unsigned __int8)v11 < 0x40u && v4 )
63        return -44;
64      v6 = v20++ + 1;
65    }
66    ++v5;
67  }
```

zorropub（重要！！）

这几天一直死在这道题上，难度其实不大，但主要考研的还是编程水平，同时菜鸡作者又学会了一种可以用python进行程序运行爆破的方法，废话不多说，上解题流程。
程序运行，ida分析，开始以为是一个很长的流程题，但是后面注意到了，前面这一段部分，第一个输入酒的数量，第二次输入每个酒的id，并判断是否符合条件，但我们发现前半部分和后面的关系只是由一个小小的seed联系起来的，所以困难减少一半

```
v15 = __readfsqword(0x28u);
seed = 0;
puts("Welcome to Pub Zorro!!");
printf("Straight to the point. How many drinks you want?", a2);
__isoc99_scanf("%d", &v5);
if ( v5 <= 0 )
{
  printf("You are too drunk!! Get Out!!", &v5);
  exit(-1);
}
printf("OK. I need details of all the drinks. Give me %d drink ids:", (unsigned int)v5);
for ( i = 0; i < v5; ++i )
{
  __isoc99_scanf("%d", &v6);
  if ( v6 <= 16 || v6 > 0xFFFF )
  {
    puts("Invalid Drink Id.");
    printf("Get Out!!", &v6);
    exit(-1);
  }
  seed ^= v6;
}
```


这里有一个seed知识，它是一个rand()随机数的一个取值范围，emmmmmm，虽然很简单,但我也是后来才明白，所以记下来吧。然后分析后面就是对这个seed值运用各种MD5，估计公式逆向不太可能，所以直接爆破，这里就用到了python进行程序运行爆破的方法。

根据第一步的代码我们将所有可能的数输出到一个数组中去，这里可以在py编辑器里运行一下，代码如下：

```
c=[]
for i in range(16,0xffff):
    a=0
    j=i
    while(j):
        a=a+1
        j = j & (j - 1)
    if(a==10):
        c.append(i)
```

运行linux，这一部分作者的kali出了点问题，这也是拖延了很长时间的原因之一，因为opensell版本的问题，所以程序里的MD5函数运行不起来，导致kali一直运行不了这个程序，最后换了Ubuntu才解决了问题。
linux打开python，将刚才的代码段打进去，然后输出后面的代码段，这里是看了大佬的wp才明白的，这里我解析一下，大佬的具体地址在这里
https://www.cnblogs.com/whitehawk/p/10933552.html
主要是用了一个subprocess函数

```
for i in c:
    proc = subprocess.Popen(['./zorro_bin'], stdin=subprocess.PIPE, stdout=subprocess.PIPE)
    #这里是用Popen进行一个子程序的打开，建立输入输出流的通道（注意大小写）
    out = proc.communicate(('1\n%s\n' % i).encode('utf-8'))[0]
    #这里是Popen的communicate函数，1是酒的数量，i是酒的id，既我们逆出来的数组里的值，后面encode是进行了转码，[0]记住就好了具体用处我也不太清楚
    if "nullcon".encode('utf-8') in out:
    #判断最后输出匹配
      print(out)
    #输出
```

最终得到flag

```
...
b'Welcome to Pub Zorro!!\nStraight to the point. How many drinks you want?OK. I
 need details of all the drinks. Give me 1 drink ids:\nYou choose right mix and
 here is your reward: The flag is nullcon{nu11c0n_s4yz_x0r1n6_1s_4m4z1ng}\n'
```