攻防世界 一（进阶篇）（含简单手动脱壳）

原创

chan3301 于 2019-06-02 21:45:58 发布 1311 收藏 1

分类专栏： 逆向题目练习 文章标签： 攻防世界 逆向入门

本文链接：https://blog.csdn.net/sjt670994562/article/details/90744023

逆向题目练习 专栏收录该内容

16 篇文章 1 订阅

订阅专栏

既然打算好好学逆向了，就应该全身心的投入，做了题以后才会发现自己有太多的不会，希望能在后面的学习更沉稳一点，更专注一点，更效率一点

dmd

放到ida进行分析，看到这些数据

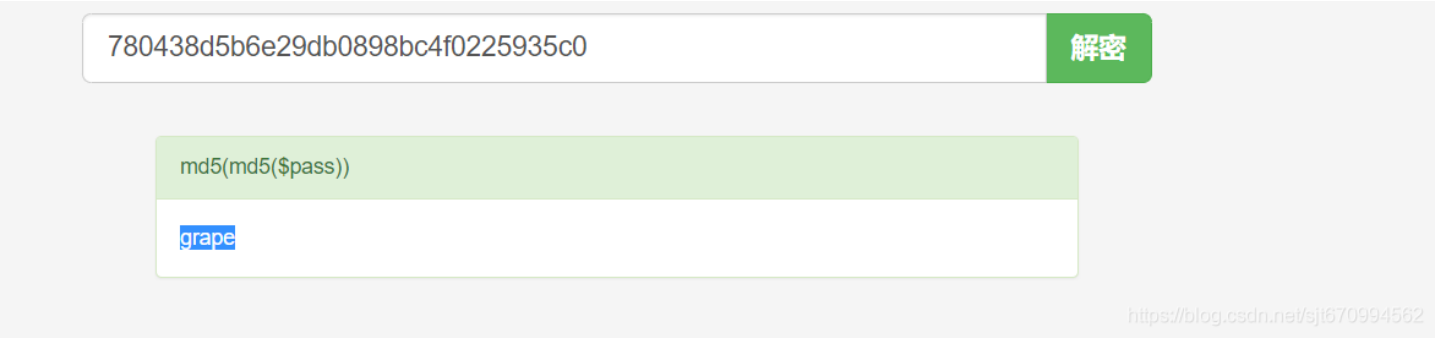
```
if ( *v41 != 55
  || v41[1] != 56
  || v41[2] != 48
  || v41[3] != 52
  || v41[4] != 51
  || v41[5] != 56
  || v41[6] != 100
  || v41[7] != 53
  || v41[8] != 98
  || v41[9] != 54
  || v41[10] != 101
  || v41[11] != 50
  || v41[12] != 57
  || v41[13] != 100
  || v41[14] != 98
  || v41[15] != 48
  || v41[16] != 56
  || v41[17] != 57
  || v41[18] != 56
  || v41[19] != 98
  || v41[20] != 99
  || v41[21] != 52
  || v41[22] != 102
  || v41[23] != 48
  || v41[24] != 50
  || v41[25] != 50
  || v41[26] != 53
  || v41[27] != 57
  || v41[28] != 51
  || v41[29] != 53
  || v41[30] != 99
  || v41[31] != 48 )
{
```


化简出来得到756d2fc84d07a356c8f68f71fe76e189
又因为题目标题，加上数据特征，再加上如下图，都可以想到MD5解密

```
md5(&v40, &v39);
```

```
MD5::MD5((MD5 *)&v3, a2);
MD5::hexdigest(a1);
```

第一次解出来的

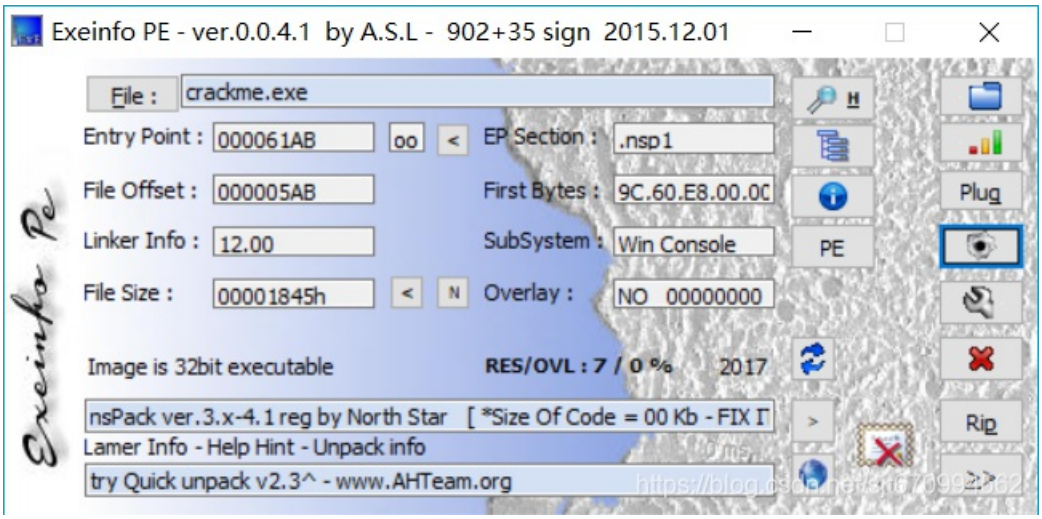


有两个MD5，且第一次的MD5属于嵌套式，所以想到，得到密文之后，再进行一次MD5加密，得到flag
b781cbb29054db12f88f08c6e161c199

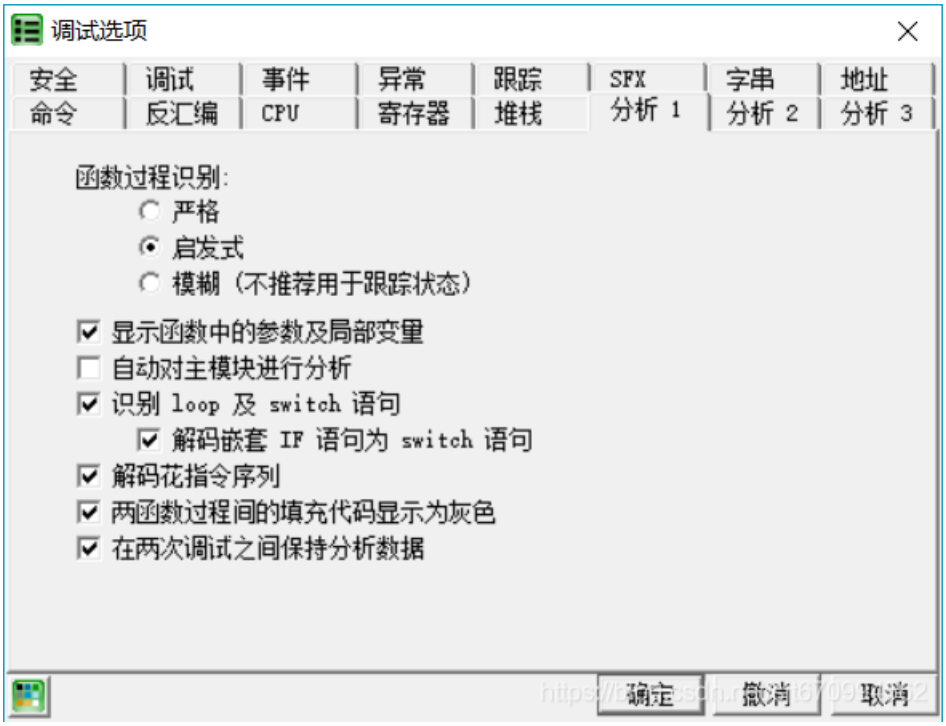
crackme

此题含有壳，所以扔到exeinfope里面去，发现是一个upack的压缩壳，所以手动脱壳还是比较简单的，下面说一下大致流程

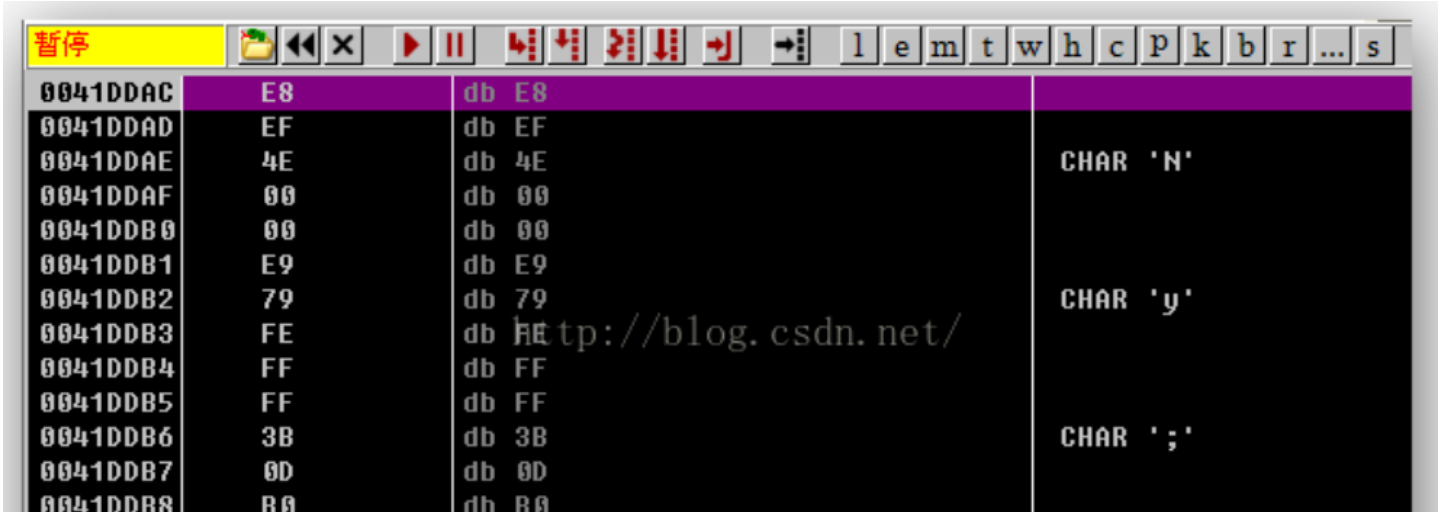
此题含有壳，所以扔到exeinfope里面去，发现是一个upack的压缩壳，所以手动脱壳是比较简单的，下面就一个大致流程。



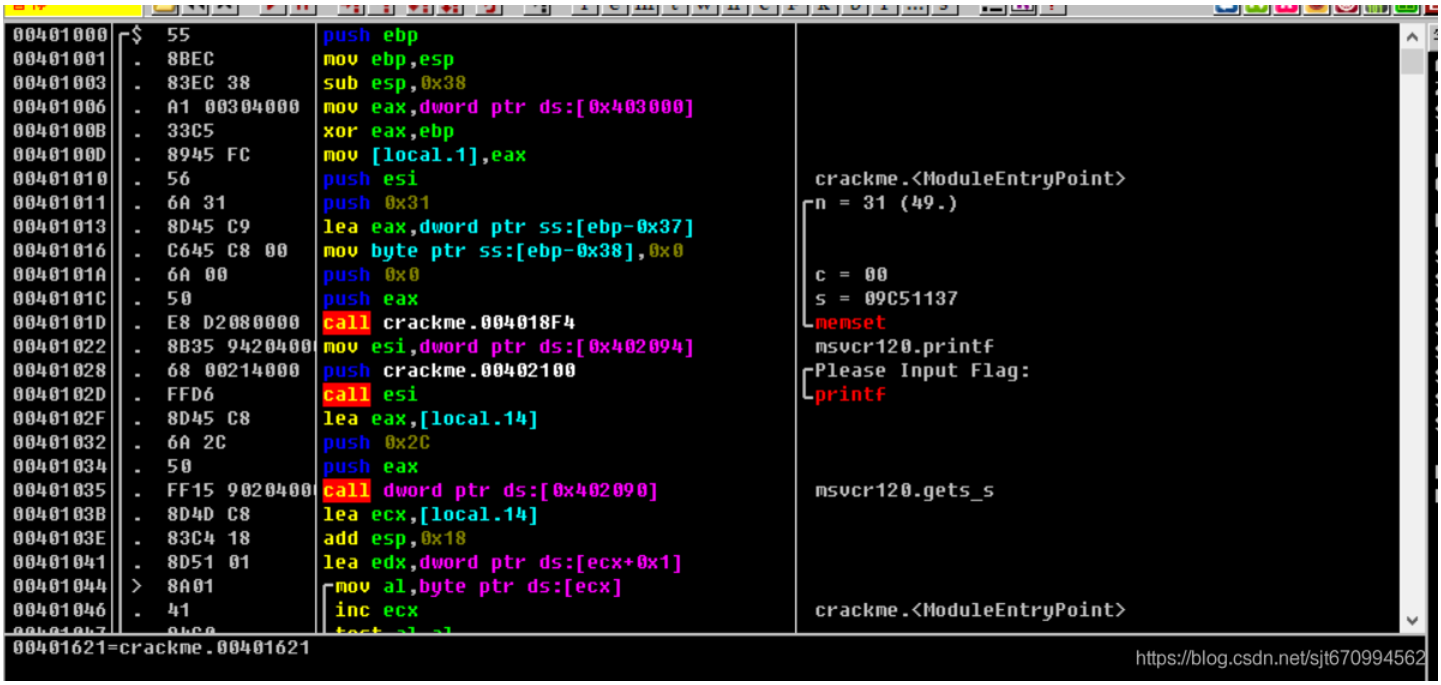
用OD打开文件，但之前要把分析1里面的自动对主模块进行分析勾掉



进入以后就直接无脑F8，遇到循环就F4一下循环下边的命令，让他不循环，然后一直F8，最后你就会进入一个全都是黑的区域，类似于下图（非本题）



我们这时候手动的将内存数据转换成汇编指令显示出来(选中没有没有正确显示的内存数据–>右键–>分析–>分析代码或者使用快捷键Ctrl + A)，就进入真正的程序入口啦



然后用OD的插件OllyDump或者Load PE结合RECImport 工具，进行程序的脱壳了。我用的是OllyDump

最后得到一个新的exe文件，后面就比较简单了，扔到ida发现是一个异或问题，数据就在text里面找就ok

```
v4 = 0;
while ( (*(&Buf + v4) ^ byte_402130[v4 % 16]) == dword_402150[v4] )
{
```

异或数据（这里有个小插曲74h也是其中的数据之一）

异或数据（这里有个小坑注意74h也是其中的数据之一）

```
30 byte_402130     db 74h                  ; DATA XREF: _main:loc_40107F↑r
31 aHisIsNotFlag   db 'his_is_not_flag',0
41                 align 10h
```

源数据

```
50 dword_402150    dd 12h                  ; DATA XREF: _main+8D↑r
54                 dd 4, 8, 14h, 24h, 5Ch, 4Ah, 3Dh, 56h, 0Ah, 10h, 67h, 0
34                 dd 41h, 0
3C                 dd 1, 46h, 5Ah, 44h, 42h, 6Eh, 0Ch, 44h, 72h, 0Ch, 0Dh
3C                 dd 40h, 3Eh, 4Bh, 5Fh, 2, 1, 4Ch, 5Eh, 5Bh, 17h, 6Eh, 0Ch
3C                 dd 16h, 68h, 5Bh, 12h, 2 dup(0)
90                 dd 48h, 0Eh dup(0)
3C                 dd offset dword 403000
```

一个脚本，最后出flag

```
a=[0x12, 0x4, 0x8, 0x14, 0x24, 0x5C, 0x4A, 0x3D, 0x56, 0x0A, 0x10, 0x67, 0x0, 0x41, 0x0, 0x1, 0x46, 0x5A, 0x44, 0x42, 0x6E, 0x0C, 0x44, 0x72, 0x0C, 0x0D, 0x40, 0x3E, 0x4B, 0x5F, 0x2, 0x1, 0x4C, 0x5E, 0x5B, 0x17, 0x6E, 0x0C, 0x16, 0x68, 0x5B, 0x12, 0x2, 0x48, 0x0E]
b="this_is_not_flag"
for i in range(0,43):
    print(chr(a[i] ^ ord(b[i%16])),end='')
```

```
flag{59b8ed8f-af22-11e7-bb4a-3cf862d1ee75}
```

re2-cpp-is-awesome

这一道题不难，只是有一个小坑，同样，扔进ida，看起来很复杂，但不要慌，抓住关键特别简单

```
if ( a1 != 2 )
{
  v3 = *a2;
  v4 = std::operator<<<std::char_traits<char>>((__int64)&std::cout, (__int64)"Usage: ", (__int64)a3);
  v6 = std::operator<<<std::char_traits<char>>(v4, (__int64)v3, v5);
  std::operator<<<std::char_traits<char>>(v6, (__int64)" flag\n", v7);
  exit(0);
}
std::allocator<char>::allocator((__int64)&v13, (__int64)a2, (__int64)a3);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::basic_string(&v12, a2[1], &v13);
std::allocator<char>::~allocator(&v13);
v15 = 0;
for ( i = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::begin((__int64)&v12); ; inc_i(&i) )
{
  v14 = std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::end(&v12);
  if ( !sub_400D3D((__int64)&i, (__int64)&v14) )
    break;
  v9 = *(unsigned __int8 *)::i((__int64)&i);
  if ( (_BYTE)v9 != off_6020A0[dword_6020C0[v15]] )
    sub_400B56((__int64)&i, (__int64)&v14, v9);
  ++v15;
}
sub_400B73((__int64)&i, (__int64)&v14, v8);
std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<char>>::~basic_string((__int64)&v12);
return 0LL;
}
```


关键点：

```
if ( (_BYTE)v9 != off_6020A0[dword_6020C0[v15]] )
  sub_400B56((__int64)&i, (__int64)&v14, v9);
++v15;
```

大意就是在下面这段字符串里面排序，用dword_6020c0做下标，再次拼接

```
db    0
db 'L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t'
                        ; DATA XREF: .data:off_6020A0↓o
db '_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t',0
db 'Better luck next time',0Ah,0
                        ; DATA XREF: sub_400B56+4↑o
```

这边注意align8，这边还有一个0x0，千万不要忘了

```
; int dword_6020C0[]
dword_6020C0    dd 24h                  ; DATA XREF:
                align 8
```

然后python脚本得flag

```
a=("L3t_ME_T3ll_Y0u_S0m3th1ng_1mp0rtant_A_{FL4G}_W0nt_b3_3X4ctly_th4t_345y_t0_c4ptur3_H0wev3r_1T_w1ll_b3_C00l_1F_Y0u_g0t_1t")
t=[0x24, 0x0, 0x5, 0x36, 0x65, 0x7, 0x27, 0x26, 0x2D, 0x1, 0x3, 0x0, 0x0D, 0x56, 0x1, 0x3, 0x65, 0x3, 0x2D, 0x16, 0x2, 0x15, 0x3, 0x65, 0x0, 0x29, 0x44, 0x44, 0x1, 0x44, 0x2B]
for i in range(0,31):
    s=t[i]
    print(a[s],end='')
```

```
ALEXCTF{W3_L0v3_C_W1th_CL45535}
```