

## RE3

题目是用ollvm来混淆的 所以得用IDA去混淆插件D-801去除混淆

先用D-801去除混淆

可以看到程序主要有两个加密函数 一个是cry1

一个是cry2

function name	segment	start
<a href="#">f</a> _init_proc	.init	0000
<a href="#">f</a> sub_401020	.plt	0000
<a href="#">f</a> _strlen	.plt	0000
<a href="#">f</a> _printf	.plt	0000
<a href="#">f</a> _memset	.plt	0000
<a href="#">f</a> _memcpy	.plt	0000
<a href="#">f</a> _memcpy	.plt	0000
<a href="#">f</a> _scanf	.plt	0000
<a href="#">f</a> _start	.text	0000
<a href="#">f</a> _dl_relocate_static_pie	.text	0000
<a href="#">f</a> deregister_tm_clones	.text	0000
<a href="#">f</a> register_tm_clones	.text	0000
<a href="#">f</a> __do_global_ctors_aux	.text	0000
<a href="#">f</a> frame_dummy	.text	0000
<a href="#">f</a> cry1(uchar *,uchar *,ulong)	.text	0000
<a href="#">f</a> cry2(uchar *,uchar *,ulong)	.text	0000
<a href="#">f</a> main	.text	0000
<a href="#">f</a> __libc_csu_init	.text	0000
<a href="#">f</a> __libc_csu_fini	.text	0000
<a href="#">f</a> _term_proc	.fini	0000
<a href="#">f</a> strlen	extern	0000
<a href="#">f</a> printf	extern	0000
<a href="#">f</a> memset	extern	0000
<a href="#">f</a> __libc_start_main	extern	0000
<a href="#">f</a> memcpy	extern	0000
<a href="#">f</a> memcpy	extern	0000
<a href="#">f</a> scanf	extern	0000
<a href="#">f</a> __gmon_start__	extern	0000

但是都带有混淆

```

20 v17 = a1;
21 v16 = a2;
22 v15 = a3;
23 v14 = 0;
24 v13 = 0;
25 memset(s, 0, 0x100uLL);
26 v14 = 0;
27 v10 = 1135530334;
28 while ( 1 )
29 {
30     while ( 1 )
31     {
32         while ( 1 )
33         {
34             while ( 1 )
35             {
36                 while ( 1 )
37                 {
38                     while ( 1 )
39                     {
40                         while ( 1 )
41                         {
42                             while ( 1 )
43                             {
44                                 while ( v10 == -2026262691 )
45                                 {
46                                     ++v14;
47                                     v10 = 219113649;
48                                 }
49                                 if ( v10 != -1800435896 )
50                                 break;
51                                 v10 = -1154024163;
52                             }
53                             if ( v10 != -1154024163 )
54                             break;
55                             v5 = -1800435896;
56                             v18 = v14 < 256;
57                             if ( (y < 10) ^ (((((_BYTE)x - 1) * (_BYTE)x) & 1) == 0) | (y < 10
58                                     && (((_BYTE)x - 1) * (_BYTE)x) & 1) == 0) )
59                             v5 = 741148234;
60                             v10 = v5;
61                         }
62                         if ( v10 != -872009407 )
63                         break;
64                         v13 = ((unsigned __int8)s[v14] + v13 + v17[v14]) % 256;
65                         v11 = v17[v14];
66                         v17[v14] = v17[v13];
67                         v17[v13] = v11;
68                         v10 = -2026262691;
69                     }
70                     if ( v10 != 219113649 )
71                     break;
72                     v7 = 1468176970;
73                     if ( v14 < 256 )
74                     v7 = -872009407;

```

我们运行D-801插件

Name	Description	Configuration
1 AddKor_Rule_1	$((x_0 \wedge x_1) \vee (0x2 \wedge (x_0 \wedge (0x0 \wedge x_1)) \wedge (x_0 \wedge x_1) + val_2))$	0
2 AddKor_Rule_2	$((x_0 \wedge x_1) \vee (0x2 \wedge ((0x0 \wedge x_0 \wedge x_1))) \wedge (x_0 \wedge x_1) + val_2)$	0
3 Add_HackersDelightRule_1	$(x_0 \vee \neg(x_1) \vee (0x1)) \Rightarrow (x_0 \wedge x_1)$	0
4 Add_HackersDelightRule_2	$((x_0 \wedge x_1) \vee (0x2 \wedge (x_0 \wedge x_1))) \Rightarrow (x_0 \wedge x_1)$	0
5 Add_HackersDelightRule_3	$((x_0 \wedge x_1) \vee (x_0 \wedge x_1)) \Rightarrow (x_0 \wedge x_1)$	0
6 Add_HackersDelightRule_4	$((0x2 \wedge (x_0 \wedge x_1)) \vee (x_0 \wedge x_1)) \Rightarrow (x_0 \wedge x_1)$	0
7 Add_HackersDelightRule_5	$((0x2 \wedge ((x_0 \wedge x_1) \wedge x_2)) \vee (x_0 \wedge x_1 \wedge x_2)) \Rightarrow (x_0 \wedge x_1 \wedge x_2)$	0
8 Add_OllvmRule_1	$\neg((x_0 \wedge x_1) \vee (0x2 \wedge (x_1 \wedge x_0))) \Rightarrow ((x_0 \wedge x_1) \vee val_1)$	0
9 Add_OllvmRule_2	$\neg((x_0 \wedge x_1) \vee (val_1 \wedge (x_0 \wedge x_1))) \Rightarrow ((x_0 \wedge x_1) \vee val_1)$	0
10 Add_OllvmRule_3	$((x_0 \wedge x_1) \vee (0x2 \wedge (x_0 \wedge x_1))) \Rightarrow (x_0 \wedge x_1)$	0
11 Add_OllvmRule_4	$((x_0 \wedge x_1) \vee (val_1 \wedge (x_0 \wedge x_1))) \Rightarrow (x_0 \wedge x_1)$	0
12 Add_SpecialConstantRule_1	$((x_0 \wedge c_1) \vee (0x2 \wedge (x_0 \wedge c_2))) \Rightarrow (x_0 \wedge c_1)$	0
13 Add_SpecialConstantRule_2	$((x_0 \wedge 0x0) \wedge c_1) \vee (0x2 \wedge (x_0 \wedge c_2)) \Rightarrow (x_0 \wedge c_1)$	0
14 Add_SpecialConstantRule_3	$((x_0 \wedge c_1) \vee (0x2 \wedge (x_0 \wedge c_2))) \Rightarrow (x_0 \wedge val\_res)$	0

  

Name	Description	Configuration
1 Unflattener: Remove control flow flattening generated by LLVM		0
2 JumpFixer: No description available	{'enabled_rules': ['CompareConstantRule1', 'CompareConstantRule2', 'CompareConstantRule3', 'JaeRule1', 'JbRule1', 'JncRule1', 'JncRule2', 'JncRule3', 'JncRule4', 'JncRule5', 'JncRule6', 'JncRule7', 'JncRule8']}	

然后混淆基本去除完毕了

发现cry1和cry2只是RC4加密的初始化密钥和加密操作

这里key = 'keys\_LLVM'

```

v10 = 0;
printf("Do you know LLVM?\n");
v9 = 0;
scanf("%s", input);
memset(s, 0, 0x100uLL);
memcpy(v7, "keys_LLVM", sizeof(v7));
v3 = strlen((const char *)v7);
cry1(s, v7, v3);
v4 = strlen((const char *)(unsigned int)input);
cry2(s, (unsigned __int8 *)input, v4);
v11 = strlen((const char *)(unsigned int)input);
if ( v11 == 23 && memcmp(input, &enc, 0x17uLL) == 0 )
{
    if ( y_7 >= 10 && y_7 < 10 )
        goto LABEL_13;
    while ( 1 )
    {
        printf("Congratulations~\n");
        if ( y_7 < 10 || y_7 >= 10 )
            break;
LABEL_13:
        printf("Congratulations~\n");
    }
}
else
{
    if ( y_7 < 10 || y_7 >= 10 )
        goto LABEL_10;
    do
    {
        printf("Sorry try again.\n");
LABEL_10:
        printf("Sorry try again.\n");
    }
    while ( y_7 >= 10 && y_7 < 10 );
}
return v10;
}

```

然后提取出enc的数据就直接解密就好了

## exp

```

import binascii
from Crypto.Util.number import *
def rc4_crypt(text, key):
    textlen=len(text)
    keylen=len(key)
    ciper=[]
    count=0
    s=list(range(256))
    for i in range(256):
        count=(count+s[i]+key[i%keylen])%256
        s[i],s[count]=s[count],s[i]
    i=0
    j=0
    for m in range(textlen):
        i=(i+1)%256
        j=(j+s[i]+1)%256
        s[i],s[j]=s[j],s[i]
        k=s[(s[i]+s[j])%256]
        ciper.append(k^text[m])
    ciper_text=''.join("%02x"%i for i in ciper)
    return ciper_text.upper()

```

```
if __name__ == "__main__":
    s =
[0x54,0x89,0x61,0x10,0xd7,0xdf,0x33,0x11,0xc0,0x43,0xdf,0x76,0xdf,0x28,0xda,0xe6
,0x13,0x45,0x81,0x6d,0x79,0x18,0xc7]
    data = 'BE3924244CD030697D071E65BEBE5BF050EE84D2C94611'
    key = '6b6579735f4c4c564d'
    print("rc4 result:", rc4_crypt(binascii.a2b_hex(data),
binascii.a2b_hex(key.upper()))
    print(long_to_bytes(0x4D4F43534354467B6D315F4C4C564D5F7730726C64217D))
```