

学到许多的re题目

```
IDA View-A x Pseudocode-B x Pseudocode-A x Hex View-1 x Structures x Enums x
0
1 v10 = __readfsqword(0x28u);
2 printf("input flag:");
3 scanf("%s", &v9[6]);
4 strcpy(v9, "actf{");
5 v5 = 1;
6 for ( i = 0; i <= 4; ++i )
7 {
8     if ( v9[i] != v9[i + 6] )
9     {
10         v5 = 0;
11         goto LABEL_6;
12     }
13 }
14 if ( !v5 )
15     goto LABEL_16;
16 LABEL_6:
17 for ( j = 0; j <= 11; ++j )
18     v8[j] = v9[j + 11];
19 if ( (unsigned __int8)sub_83A(v8) && v9[23] == '}' )
20 {
21     printf("That's true! flag is %s", &v9[6]);
22     result = 0LL;
23 }
24 else
25 {
26 LABEL_16:
27 }
```

首先看这个

逻辑很清晰

很快就能明确到sub_83A是关键函数。

一开始因为sub_83A函数太大了，需要进入ida目录下的cfg目录中的MAX_FUNC_SIZE从64改为1024

```
MAX_FUNC_SIZE = 1024 // Functions over 64K are not decompiled
```

```
MAX_FUNC_ARGS = 64 // Max number of function arguments
```

然后可以编译了

```

3003 a1[7] ^= 0x3Cu;
3004 a1[8] ^= 0x6Bu;
3005 a1[9] ^= 0x70u;
3006 a1[10] ^= 0x29u;
3007 a1[11] ^= 0x3Bu;
3008 v3[0] = 126;
3009 v3[1] = 50;
3010 v3[2] = 37;
3011 v3[3] = 88;
3012 v3[4] = 89;
3013 v3[5] = 107;
3014 v3[6] = 53;
3015 v3[7] = 110;
3016 v3[8] = 0;
3017 v3[9] = 19;
3018 v3[10] = 30;
3019 v3[11] = 56;
3020 for ( i = 0; i <= 11; ++i )
3021 {
3022     if ( v3[i] != a1[i] )
3023     {
3024         printf("wrong on #%d\n", (unsigned int)i);
3025         return 0LL;
3026     }
3027 }
3028 return 1LL;
3029 }

```

可以看到，第几个字符错误，就会告诉你第几个字符出错了，这时候爆破就很有效

```

from pwn import *
import re
flag = 'actf{'
for i in range(12):
    for j in range(33,127):
        test_flag = flag
        io = process('./Soullike')
        test_flag += chr(j)
        if i == 11:
            test_flag += '}'
        print(test_flag)
        io.sendline(test_flag)
        msg = io.recvuntil('\n')
        r = re.findall('wrong on #(.*)\n',msg)
        r = r[0]
        r = int(r)
        io.close()
        if r == i:
            continue
        else:
            flag += chr(j)
            break

```

原来pwntools还能这样用，实在是涨姿势了啊