2021年hgameWeek1WP

Web

Hitchhiking_in_the_Galaxy

这题拿到手打开是这样

404

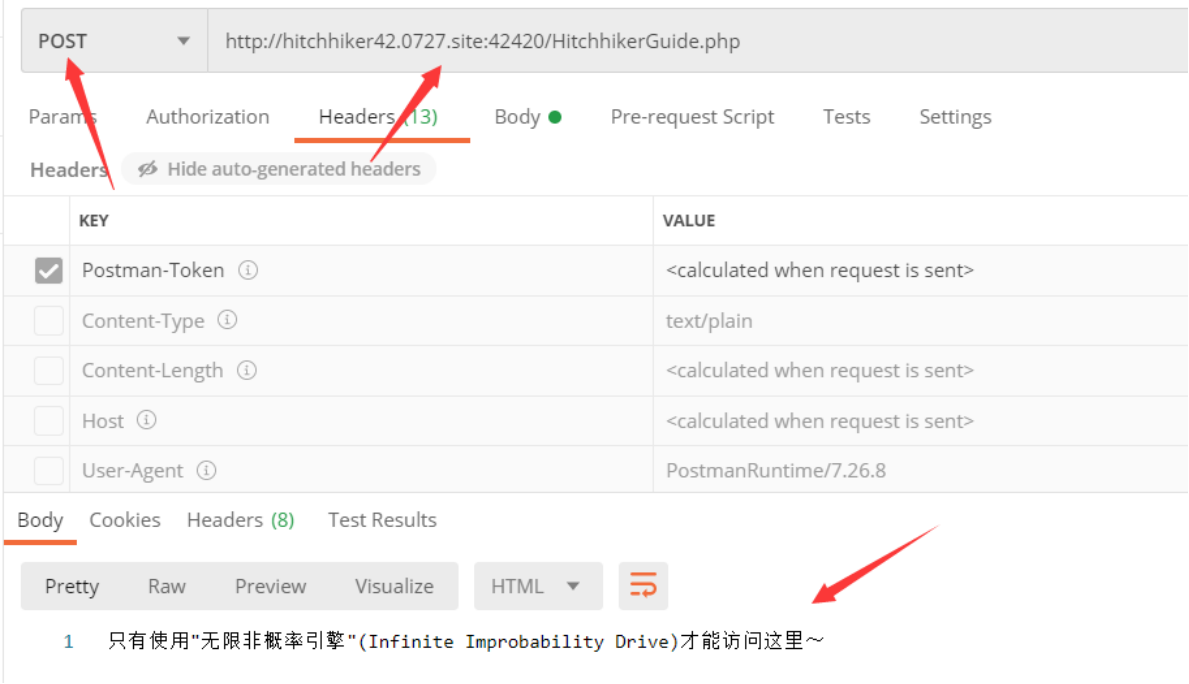
你来晚了，地球已经被沃贡人摧毁了。原因是地球挡住了它们的超空间快速通道。

我要搭顺风车！

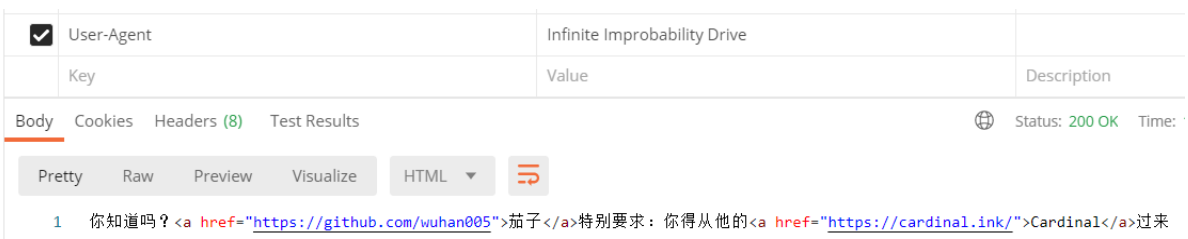
那必然是直接冲去搭顺风车，url：http://hitchhiker42.0727.site:42420/HitchhikerGuide.php

然后 302 跳转回原来页面（这里记不太清了，写 wp 的时间已经是做题完3天后了）

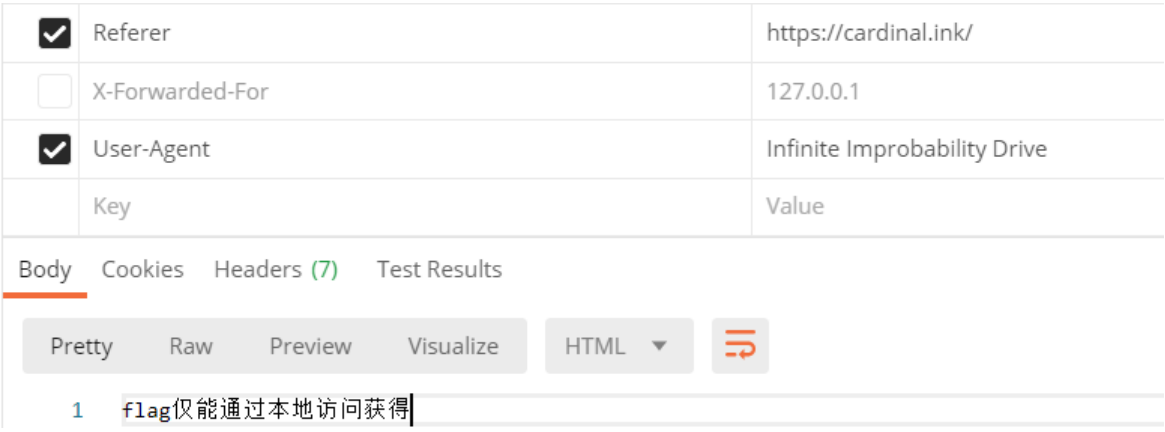
直接把链接拿到 Postman 里头尝试不同的请求方式（这里第一时间没想到，是后来其他题目卡住了回来试这个）



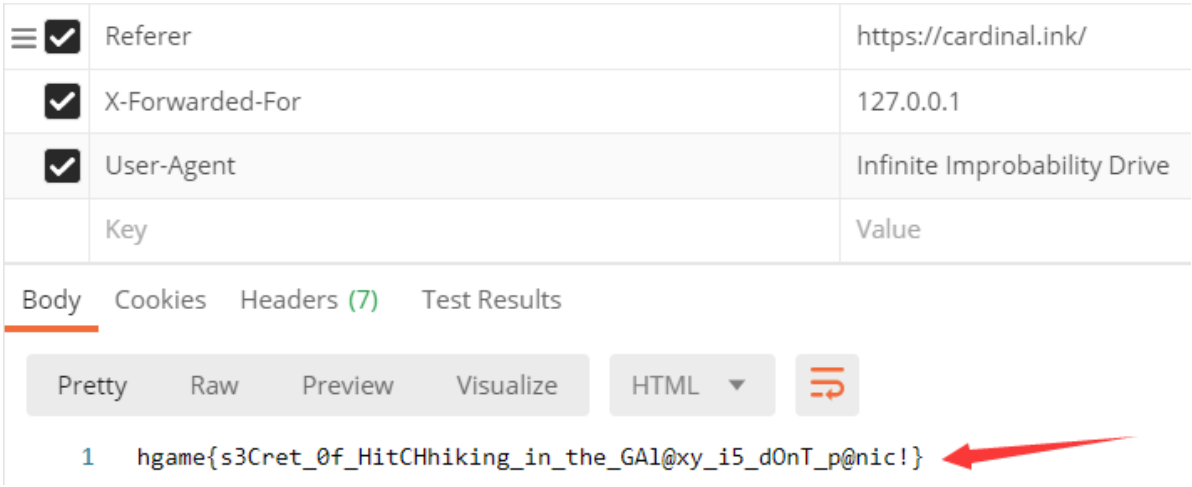
发现使用 Post 请求回返回要求使用“无限非概率引擎”访问，将 UA 改为“Infinite Improbability Drive”得到：



将 Referer 改为“https://cardinal.ink/”，请求得到：

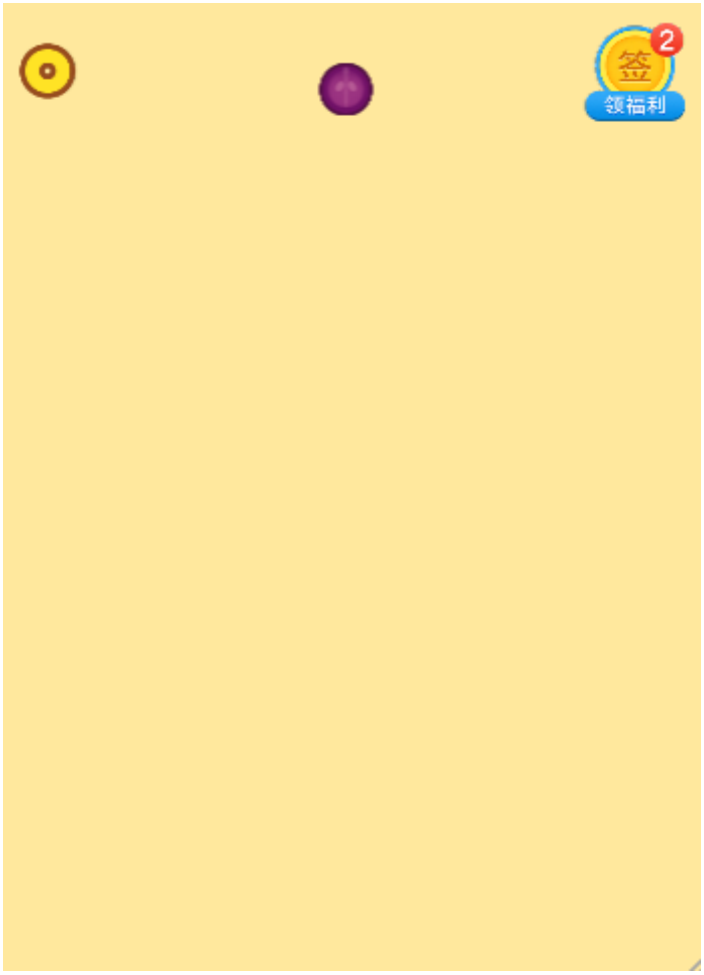


更改 X-Forwarded-For 为 “127.0.0.1”，请求得到：



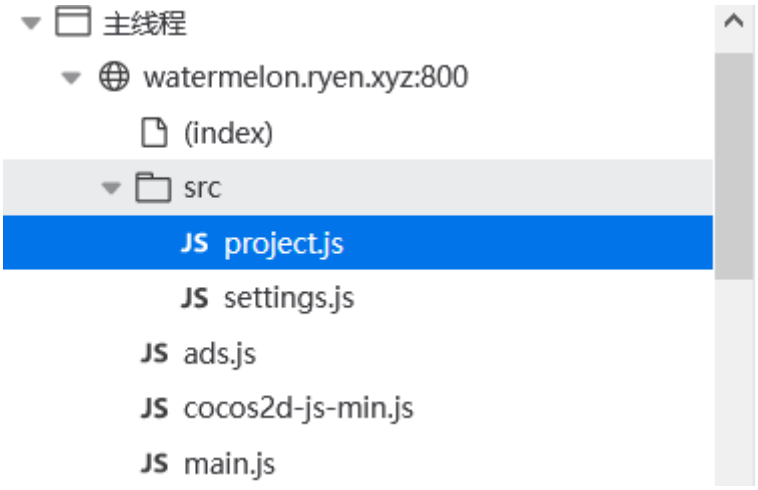
hgame{s3Cret_0f_HitCHhiking_in_the_GAl@xy_i5_dOnT_p@nic!}

watermelon



拿到这个网页，先玩了亿把，然后开始分析网页源码。

先快速结束一局，发现结束后并网页没有向服务器发送请求，推测 js 中存在判断分数的语句，有可能 flag 就在js中



经过分析，project.js 是游戏程序

因为题目要求的是分数超过 2000 分给 flag，所以尝试直接搜索 2000，但是没有找到

然后杀去分析了一下游戏结束后回触发哪些函数（挺蠢的，实际上再搜索一下 1999 就直接出答案了）

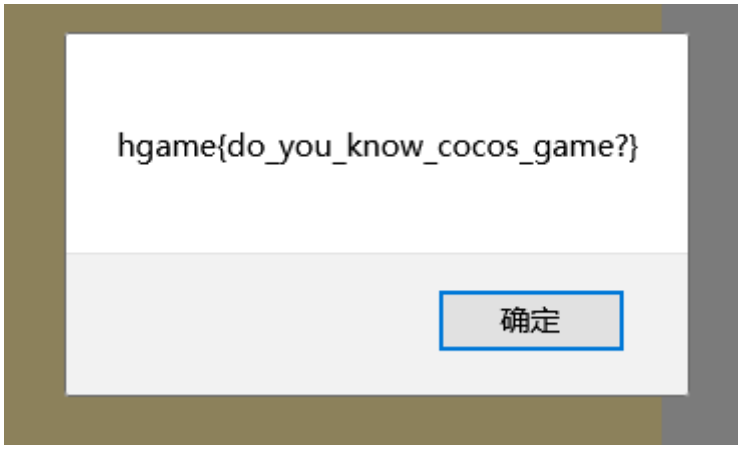
然后在 2087 行找到这个函数：

```
2087                gameOverShowText: function (e, t) {
2088                    if(e > 1999){
2089                        alert(window.atob("aGdhbWV7ZG9feW91X2tub3dfY29jb3NfZ2FtZT99"))
2090                    }
2091                    // this.ajaxLoad("http://www.wesane.com/admin.php/Gamescore/saveGam
2092                },
```

可以很明显看出这个函数负责 flag

直接把alert复制到控制台运行就行：

```
» alert(window.atob("aGdhbWV7ZG9feW91X2tub3dfY29jb3NfZ2FtZT99"))
```



hgame{do_you_know_cocos_game?}

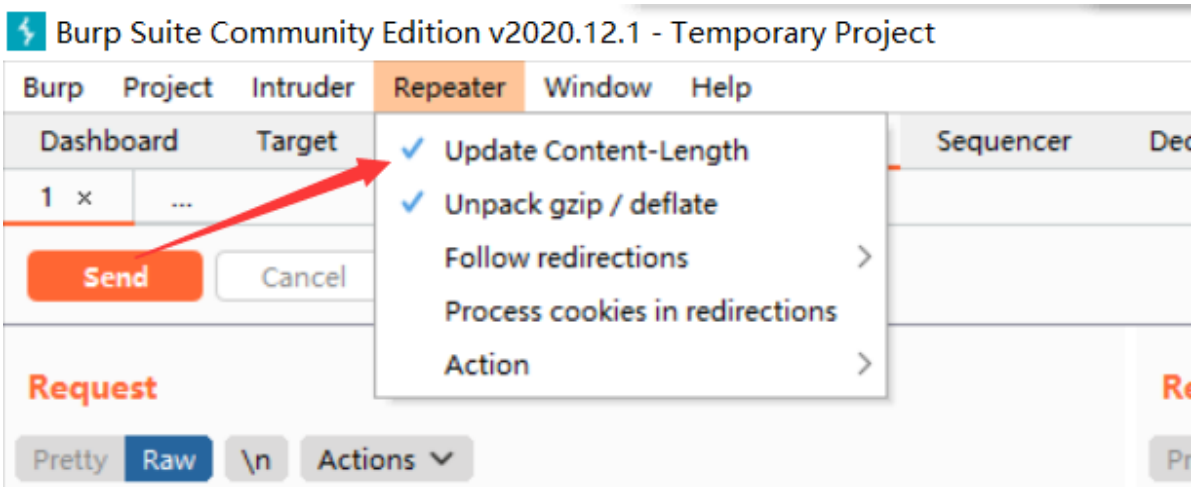
宝藏走私者

这题卡了一天

刚开始请求要求说要本地访问（请求头无 client-ip），但是在请求头添加 client-ip 后发现还是不行，服务器获取到的 client-ip 是我的公网 ip

然后了解到http走私这种操作

用 burpsuite 构造一个 http 请求即可

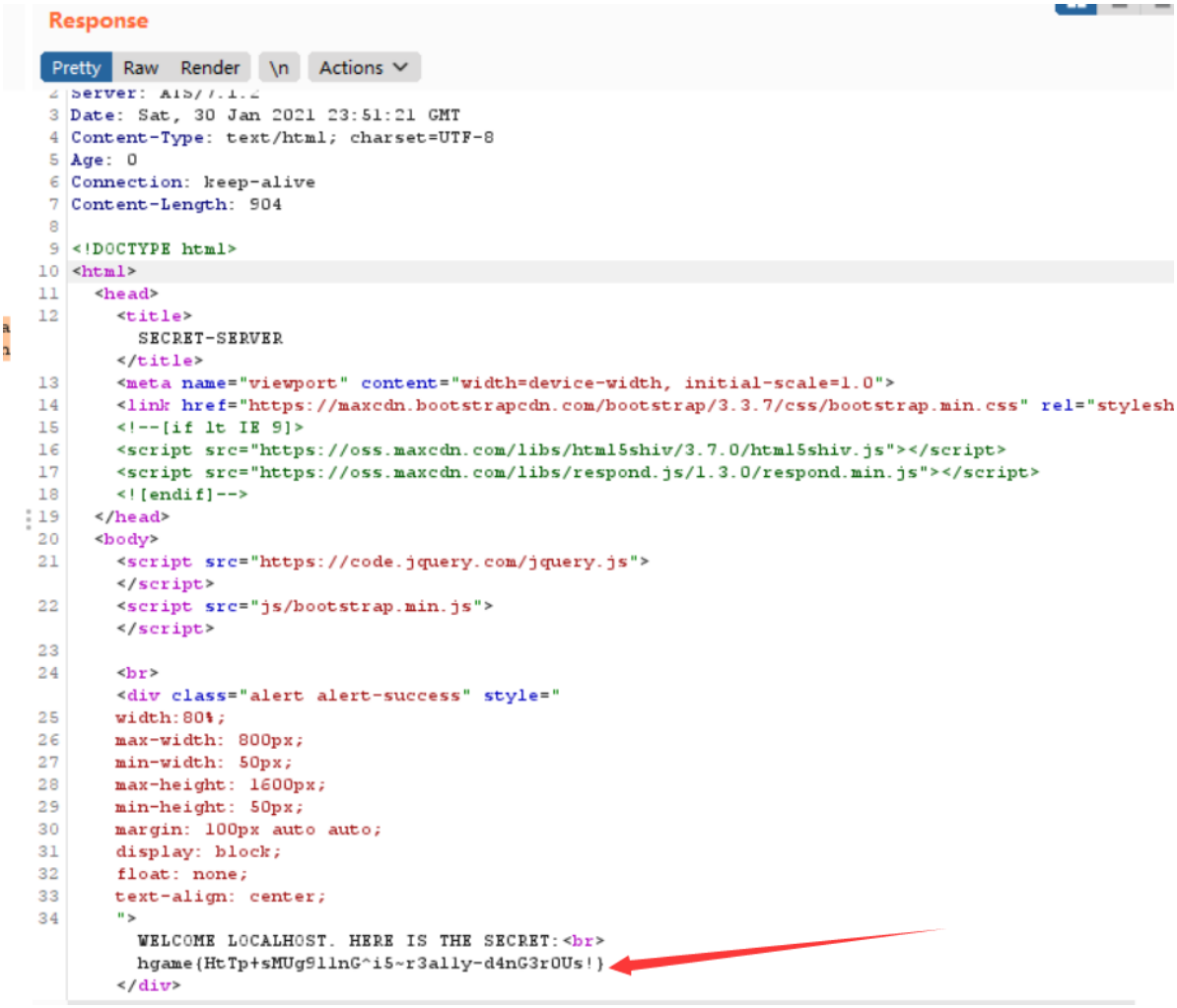


在发送请求之前记得把自动更新内容长度的勾取消掉

连续多次请求

```
GET /secret HTTP/1.1
Host: 951ec7e5e2.thief.0727.site
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
```

```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.88 Safari/537.36
DNT:1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Content-Length: 5
Transfer-Encoding: chunked

0


GET /secret HTTP/1.1
Client-IP: 127.0.0.1
Host: 951ec7e5e2.thief.0727.site
Connection: close
```



hgame{HtTp+sMUg9l1nG^i5~r3al1y-d4nG3r0Us!}

智商检测鸡

这道题挺。。检测智商的。

做了两题，然后分析了一下cookies，发现其中带有当前解决的题目的数量，再往下发现有一定的分析难度，索性直接写了个小脚本搞定

```
import requests
import json
from bs4 import BeautifulSoup
session = requests.session()
r = session.get('http://r4u.top:5000/api/getQuestion')
soup = BeautifulSoup(r.content, "html.parser")
x2 = int(soup.find_all('mn')[0].get_text()) * -1
x1 = int(soup.find_all('mn')[1].get_text())
a = int(soup.find_all('mn')[2].get_text())
b = int(soup.find_all('mn')[3].get_text())
# print(soup.find_all('mn')[0].get_text())
answer = a/2*x1*x1+b*x1-a/2*x2*x2-b*x2
answer = round(answer, 2)
data = {"answer": answer}
r = session.post('http://r4u.top:5000/api/verify', json=data)
print(r.content)
status = json.loads(session.get('http://r4u.top:5000/api/getStatus').text)

while(status.get('solving') != 100):
    r = session.get('http://r4u.top:5000/api/getQuestion')
    soup = BeautifulSoup(r.content, "html.parser")
    x2 = int(soup.find_all('mn')[0].get_text()) * -1
    x1 = int(soup.find_all('mn')[1].get_text())
    a = int(soup.find_all('mn')[2].get_text())
    b = int(soup.find_all('mn')[3].get_text())
    answer = a/2*x1*x1+b*x1-a/2*x2*x2-b*x2
    answer = round(answer, 2)
    data = {"answer": answer}
    r = session.post('http://r4u.top:5000/api/verify', json=data)
    print(r.content)
    status = json.loads(session.get('http://r4u.top:5000/api/getStatus').text)
    print(status.get('solving'))
    print(session.cookies)
```

拿到cookies，用cookies访问网页即可

hgame{3very0ne_H4tes_Math}

走私者的愤怒

这一题的思路和宝藏走私者的思路应该是一样的

但是不知道为什么，用宝藏请求者的头去请求一直都拿不到东西

最后无奈问 Liki 姐姐，说是改成 POST 试一试

然后在第二个 GET 请求中加了一点点小改动拿到了 flag

```
GET /secret HTTP/1.1
Host: police.liki.link
Pragma: no-cache
```

```
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/87.0.4280.88 Safari/537.36
DNT:1
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Content-Length: 5
Transfer-Encoding: chunked

0


GET /secret HTTP/1.1
Client-IP: 127.0.0.1
Host: police.liki.link
Content-Length: 7

12345
```

和宝藏走私者就差了三行，终于拿到了flag

hgame{Fe3l^tHe~4N9eR+oF_5mu9gl3r!!}

Reverse

apacha

拿到程序，丢进ida

找到主程序位置，按f5转换一下

```
__int64 __fastcall main(int a1, char **a2, char **a3)
{
  _DWORD *v3; // rbx
  __int64 i; // rax
  int v6[4]; // [rsp+0h] [rbp-48h] BYREF
  char v7[40]; // [rsp+10h] [rbp-38h] BYREF
  unsigned __int64 v8; // [rsp+38h] [rbp-10h]

  v8 = __readfsqword(0x28u);
  v6[0] = 1;
  v6[1] = 2;
  v6[2] = 3;
  v6[3] = 4;
  sub_11AA();
  __printf_chk(1LL, "Please input: ");
  __isoc99_scanf("%35s", v7);
  if ( (unsigned int)strlen(v7) != 35 )
  {
    puts("wrong length!");
    exit(0);
  }
  v3 = malloc(0x8CuLL);
  for ( i = 0LL; i != 35; ++i )
    v3[i] = v7[i];
  sub_1447(v3, 35, (__int64)v6);
  if ( (unsigned int)sub_1550(v3, 35) )
    puts("    :)  Flag is your input.");
  else
    puts("    :(  Try again.");
  return 0LL;
}
```

分析函数可以大概看出要输入一个长度为 35 的字符串，并且根据返回的信息可以看出flag的长度就是35

25 行那里调用了一个函数，判断大概率是将输入的f lag 加密的程序

双击跟过去看一眼

```
do
{
  v6 -= 1640531527;
  v7 = v6 >> 2;
  if ( a2 == 1 )
  {
    v9 = 0;
  }
  else
  {
    v8 = 0LL;
    do
    {
      v5 = a1[v8]
         + ((((v5 >> 5) ^ (4 * a1[v8 + 1])) + ((16 * v5) ^ (a1[v8 + 1] >> 3))) ^ ((*(_DWORD *)(a3
                                                                                              + 4LL
                                                                                              * (((unsigned __int8)v8 ^ (unsigned __int8)v7) & 3)) ^ v5)
                                                                                + (a1[v8 + 1] ^ v6)));
      a1[v8++] = v5;
    }
    while ( v8 != (unsigned int)(a2 - 2) + 1LL );
    v9 = a2 - 1;
  }
  result = (16 * v5) ^ (*a1 >> 3);
  v5 = *v4
     + (((*(_DWORD *)(a3 + 4LL * ((v9 ^ (unsigned __int8)v7) & 3)) ^ v5) + (*a1 ^ v6)) ^ (((4 * *a1) ^ (v5 >> 5))
                                                                                         + result));
  *v4 = v5;
}
while ( v6 != -1640531527 * (52 / a2) - 1253254570 );
return result;
}
```

看到熟悉的异或，实锤了实锤了

回到主程序，第 26 行判断语句中调用的函数应该就是判断加密后的输入的字符串是否和加密后的flag相同

同样跟进去看

```
26  if ( (unsigned int)sub_1550(v3, 35LL) )
27    puts("    :)  Flag is your input.");
28  else
29    puts("    :(  Try again.");
30  return 0LL;
31}
```

```
1 __int64 __fastcall sub_1550(_DWORD *a1, int a2)
2 {
3   __int64 v2; // rax
4   int v3; // edx
5
6   if ( a2 <= 0 )
7     return 1LL;
8   if ( *a1 != dword_5020 )
9     return 0LL;
10  v2 = 4LL;
11  while ( v2 != 4LL * (unsigned int)(a2 - 1) + 4 )
12  {
13    v3 = a1[(unsigned __int64)v2 / 4];
14    v2 += 4LL;
15    if ( v3 != *(_DWORD *)((char *)&unk_501C + v2) )
16      return 0LL;
17  }
18  return 1LL;
19}
```

这堆 4 看起来挺难受的，索性自己重新写一下这个函数

```
int verify(_DWORD* word, int length) {
    if (*word != 0x23)
        return 0;
    v2 = 1;
    while (v2 != 1 * (unsigned int)(length - 1) + 1) {
        v3 = word[v2];
        v2 += 1;
        if (v3 != *(_DWORD*)((char*)&unk_501C + v2 * 4))
            return 0;
        // word[1] = *(&unk_501C + 8)
    }
    return 1;
}
```

简化后的这个函数大概长这样，可以看出是从word[1]开始，和 unk_501C + 8 的地址上的值开始比较

同样在ida中跟过去看 unk_501C 地址之后的值

```
.data:000000000000501C unk_501C         db    0
.data:000000000000501D                  db    0
.data:000000000000501E                  db    0
.data:000000000000501F                  db    0
.data:0000000000005020 unk_5020         db  23h ; #
.data:0000000000005021                  db 0B3h
.data:0000000000005022                  db  4Eh ; N
.data:0000000000005023                  db 0E7h
.data:0000000000005024                  db  36h ; 6
.data:0000000000005025                  db  28h ; (
.data:0000000000005026                  db 0A7h
.data:0000000000005027                  db 0B7h
.data:0000000000005028                  db 0E2h
.data:0000000000005029                  db  6Fh ; o
.data:000000000000502A                  db 0CAh
.data:000000000000502B                  db  59h ; Y
.data:000000000000502C                  db 0C1h
.data:000000000000502D                  db 0C5h
.data:000000000000502E                  db  7Ch ; |
.data:000000000000502F                  db  96h
.data:0000000000005030                  db  74h ; t
.data:0000000000005031                  db  26h ; &
.data:0000000000005032                  db  80h
.data:0000000000005033                  db 0E7h
.data:0000000000005034                  db 0E6h
.data:0000000000005035                  db  54h ; T
.data:0000000000005036                  db  2Dh ; -
.data:0000000000005037                  db  3Dh ; =
.data:0000000000005038                  db  56h ; V
.data:0000000000005039                  db    3
.data:000000000000503A                  db  9Dh
.data:000000000000503B                  db  8Ah
.data:000000000000503C                  db  9Ch
.data:000000000000503D                  db 0C3h
.data:000000000000503E                  db 0DCh
.data:000000000000503F                  db  99h
```

这个比较可以看出密文第一位的值

```
8  if ( *a1 != dword_5020 )
9    return 0LL;
```

那么这样就能拼凑出一个完整的密文字符串：

```
0x0E74EB323, 0x0B7A72836, 0x59CA6FE2,  0x967CC5C1,  0x0E7802674,
0x3D2D54E6,  0x8A9D0356,  0x99DCC39C,  0x7026D8ED,  0x6A33FDAD,
0x0F496550A, 0x5C9C6F9E,  0x1BE5D04C,  0x6723AE17,  0x5270A5C2,
0x0AC42130A, 0x84BE67B2,  0x705CC779,  0x5C513D98,  0x0FB36DA2D,
0x22179645,  0x5CE3529D,  0x0D189E1FB, 0x0E85BD489, 0x73C8D11F,
0x54B5C196,  0x0B67CB490, 0x2117E4CA,  0x9DE3F994,  0x2F5AA1AA,
0x0A7E801FD, 0x0C30D6EAB, 0x1BADDC9C,  0x3453B04A,  0x92A406F9
```

再去分析加密的函数，发现很特殊的 1640531527

经过学长点拨，去查了了一下，发现是 XXTEA 加密，然后上 Github 找了一份现成的解密库，把解密的函数单独拉出来，写了个程序

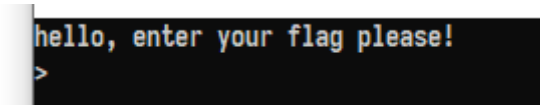
```
#include <stdio.h>
#include <string.h>
#define DELTA 0x9e3779b9
#define MX                                                \
    (((z >> 5) ^ (y << 2)) + ((y >> 3) ^ (z << 4))) ^ \
        ((sum ^ y) + (key[(p & 3) ^ e] ^ z))
static unsigned int* xxtea_uint_decrypt(unsigned int* data,
                                        size_t len,
                                        unsigned int* key) {
    unsigned int n = (unsigned int)len - 1;
    unsigned int z, y = data[0], p, q = 6 + 52 / (n + 1), sum = q * DELTA, e;

    if (n < 1)
        return data;

    while (sum != 0) {
        e = sum >> 2 & 3;

        for (p = n; p > 0; p--) {
            z = data[p - 1];
            y = data[p] -= MX;
        }

        z = data[n];
        y = data[0] -= MX;
        sum -= DELTA;
    }

    return data;
}
int main() {
    unsigned int data2[35] = {
        0x0E74EB323, 0x0B7A72836, 0x59CA6FE2,  0x967CC5C1,  0x0E7802674,
        0x3D2D54E6,  0x8A9D0356,  0x99DCC39C,  0x7026D8ED,  0x6A33FDAD,
        0x0F496550A, 0x5C9C6F9E,  0x1BE5D04C,  0x6723AE17,  0x5270A5C2,
        0x0AC42130A, 0x84BE67B2,  0x705CC779,  0x5C513D98,  0x0FB36DA2D,
        0x22179645,  0x5CE3529D,  0x0D189E1FB, 0x0E85BD489, 0x73C8D11F,
        0x54B5C196,  0x0B67CB490, 0x2117E4CA,  0x9DE3F994,  0x2F5AA1AA,
        0x0A7E801FD, 0x0C30D6EAB, 0x1BADDC9C,  0x3453B04A,  0x92A406F9};
    unsigned int key[4] = {1, 2, 3, 4};
    xxtea_uint_decrypt(data2, 35, key);
    for (int i = 0; i < 35; i++) {
        printf("%c", (char)data2[i]);
    }
    return 0;
}
```

运行程序，输出flag，冲了。

hgame{l00ks_1ike_y0u_f0Und_th3_t34}

helloRe

拿到一个exe文件



拿到ida中分析一下

找到主程序

```
.text:00000001400014C5                mov     [rsp+arg_8], rbp
.text:00000001400014CA                mov     [rsp+arg_10], rsi
.text:00000001400014CF                mov     [rsp+arg_18], rdi
.text:00000001400014D4                push    r14
.text:00000001400014D6                sub     rsp, 50h
.text:00000001400014DA                mov     rax, cs:__security_cookie
.text:00000001400014E1                xor     rax, rsp
.text:00000001400014E4                mov     [rsp+58h+var_18], rax
.text:00000001400014E9                xor     ebx, ebx
.text:00000001400014EB                mov     [rsp+58h+var_28], rbx
.text:00000001400014F0                mov     [rsp+58h+var_20], 0Fh
.text:00000001400014F9                mov     byte ptr [rsp+58h+Block], bl
.text:00000001400014FD                lea     rdx, aHelloEnterYour ; "hello, enter your flag please!"
.text:0000000140001504                mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::ostream std::cout
.text:000000014000150B                call    sub_1400017C0
.text:0000000140001510                lea     rdx, sub_140001990
.text:0000000140001517                mov     rcx, rax
.text:000000014000151A                call    cs:??6?$basic_ostream@DU?$char_traits@D@std@@@std@@QEAAAEAV01@P6AAEAV01@AEAV01@@Z@Z ; std::ostream
.text:0000000140001520                mov     rcx, rax
.text:0000000140001523                lea     rdx, asc_140003438 ; "> "
.text:000000014000152A                call    sub_1400017C0
.text:000000014000152F                lea     rdx, [rsp+58h+Block]
.text:0000000140001534                mov     rcx, cs:?cin@std@@3V?$basic_istream@DU?$char_traits@D@std@@@1@A ; std::istream std::cin
.text:000000014000153B                call    sub_140001BD0
.text:0000000140001540                lea     rdx, aCheckingFlag ; "checking flag "
.text:0000000140001547                mov     rcx, cs:?cout@std@@3V?$basic_ostream@DU?$char_traits@D@std@@@1@A ; std::ostream std::cout
.text:000000014000154E                call    sub_1400017C0
.text:0000000140001553                mov     ecx, 0C8h
.text:0000000140001558                call    sub_140001290
.text:000000014000155D                cmp     [rsp+58h+var_28], 16h
.text:0000000140001563                jnz     loc_140001640
```

按F5，方便分析

```
25  sub_140001290(200i64);
26  if ( v14 != 22 )
27LABEL_13:
28    sub_140001480();
29  v6 = v15;
30  v7 = (void **)Block[0];
31  do
32  {
33    v8 = Block;
34    if ( v6 >= 0x10 )
35      v8 = v7;
36    if ( (*((_BYTE *)v8 + v3) ^ (unsigned __int8)sub_140001430()) != byte_140003480[v3] )
37      goto LABEL_13;
38    ++v3;
39  }
40  while ( v3 < 22 );
41  v9 = (__int64 *)std::ostream::operator<<(std::cout, sub_140001990);
42  v10 = sub_1400017C0(v9, (__int64)&unk_140003470);
43  std::ostream::operator<<(v10, sub_140001990);
44  if ( v6 >= 0x10 )
45  {
46    v11 = v7;
47    if ( v6 + 1 >= 0x1000 )
48    {
49      v7 = (void **)*(v7 - 1);
50      if ( (unsigned __int64)((char *)v11 - (char *)v7 - 8) > 0x1F )
51        invalid_parameter_noinfo_noreturn();
52    }
53    j_j_free(v7);
54  }
55  return 0;
56}
```

发现在输出结果之前这一波操作似曾相识

盲猜是加密，然后看看其中用到了哪些变量

逆到最后发现就是下面这一坨被 xor 加密了

```
.rdata:0000000140003480 ; _BYTE unk_140003480[23]
.rdata:0000000140003480 unk_140003480   db  97h                 ; DATA XREF: main+A9↑o
.rdata:0000000140003481                 db  99h
.rdata:0000000140003482                 db  9Ch
.rdata:0000000140003483                 db  91h
.rdata:0000000140003484                 db  9Eh
.rdata:0000000140003485                 db  81h
.rdata:0000000140003486                 db  91h
.rdata:0000000140003487                 db  9Dh
.rdata:0000000140003488                 db  9Bh
.rdata:0000000140003489                 db  9Ah
.rdata:000000014000348A                 db  9Ah
.rdata:000000014000348B                 db 0ABh
.rdata:000000014000348C                 db  81h
.rdata:000000014000348D                 db  97h
.rdata:000000014000348E                 db 0AEh
.rdata:000000014000348F                 db  80h ; €
.rdata:0000000140003490                 db  83h
.rdata:0000000140003491                 db  8Fh
.rdata:0000000140003492                 db  94h
.rdata:0000000140003493                 db  89h
.rdata:0000000140003494                 db  99h
.rdata:0000000140003495                 db  97h
.rdata:0000000140003496                 db    0
.rdata:0000000140003497                 align 8
```

xor的另外一部分就是个函数

看了下前面一堆没用的东西，重点就在最后一句，是变量--，直接杀去看变量，得出变量原始值是ff

```
1 __int64 sub_140001430()
2 {
3   CloseHandle((HANDLE)0xC001CAFEi64);
4   sub_140001290(50i64);
5   return (unsigned __int8)byte_140005044--;
6 }
```

```
.data:0000000140005044 byte_140005044  db 0FFh
```

然后就是写个python程序获取flag就好了

```
key = '\xff'
words =
'\x97\x99\x9C\x91\x9E\x81\x91\x9D\x9B\x9A\x9A\xAB\x81\x97\xAE\x80\x83\x8F\x94\x8
9\x99\x97'
for word in words:
    print(chr(ord(word) ^ ord(key)), end='')
    key = chr(ord(key) - 1)
```

hgame{hello_re_player}

pypy

```
4           0 LOAD_GLOBAL              0 (input)
            2 LOAD_CONST               1 ('give me your flag:\n')
            4 CALL_FUNCTION            1
            6 STORE_FAST               0 (raw_flag)

5           8 LOAD_GLOBAL              1 (list)
           10 LOAD_FAST                0 (raw_flag)
           12 LOAD_CONST               2 (6)
           14 LOAD_CONST               3 (-1)
           16 BUILD_SLICE              2
           18 BINARY_SUBSCR
           20 CALL_FUNCTION            1
           22 STORE_FAST               1 (cipher)

6          24 LOAD_GLOBAL              2 (len)
           26 LOAD_FAST                1 (cipher)
           28 CALL_FUNCTION            1
           30 STORE_FAST               2 (length)

8          32 LOAD_GLOBAL              3 (range)
           34 LOAD_FAST                2 (length)
           36 LOAD_CONST               4 (2)
           38 BINARY_FLOOR_DIVIDE
           40 CALL_FUNCTION            1
           42 GET_ITER
      >>   44 FOR_ITER                54 (to 100)
           46 STORE_FAST               3 (i)

9          48 LOAD_FAST                1 (cipher)
           50 LOAD_CONST               4 (2)
```

```
# your flag: 30466633346f59213b4139794520572b45514d61583151576638643a
```

pypy这题和python的dis有关，本来想直接分析的，无奈脑子不够用，只好逆回去写了原程序

```
raw_flag = input('give me your flag:')
cipher = raw_flag[6:-1]
length = len(cipher)
for i in range(length/2):
    cipher[2*i], cipher[2*i+1] = cipher[2*i+1], cipher[2*i]
res = []
for i in range(length):
    res.append(ord(cipher[i]) ^ i)
res = bytes(hex(res))
return 0
```

然后就把步骤反过来就好了

这里我把密文先用hex解密解了再进行解密

```
cipher = "OFf34oY!;A9yE W+EQMaX1QWf8d:"
length = len(cipher)
words = []
for i in range(length):
    words.append(ord(cipher[i]) ^ i)
for i in range(int(length/2)):
    i = int(length/2)-i-1
    words[2*i], words[2*i+1] = words[2*i+1], words[2*i]
print(words)
for i in words:
    print(chr(i), end='')
```

> G00dj0&_H3r3-I$Y@Ur_$L@G!~!~

得到的不能直接用，从原程序可以看出是掐头去尾了，所以补充头尾得到：

hgame{G00dj0&_H3r3-I$Y@Ur_$L@G!~!~}

PWN

whitegive

这里拿到的是编译好的程序和c语言文件

先看一下源码：

```c
#include <stdio.h>
#include <unistd.h>

void init_io()
{
    setbuf(stdin, NULL);
    setbuf(stdout, NULL);
    setbuf(stderr, NULL);
}

int main()
{
    unsigned long long num;

    init_io();

    printf("password:");
    scanf("%ld", &num);

    if (num == "paSsw0rd") { //Do you know strcmp?
        printf("you are right!\n");
        system("/bin/sh");
    } else {
        printf("sorry, you are wrong.\n");
    }


    return 0;
}
```

重点就在这两句话，所以只要让num变量的值等于“paSsw0rd”的首地址即可。因为这个“paSsw0rd”是常量，所以地址已经固定好了。

把程序拖入ida分析一波，直接Alt+T查找

```
.rodata:0000000000402012 aPassw0rd        db 'paSsw0rd',0
```

得到“paSsw0rd”的首地址402012，注意这个是16进制，得转为10进制输入

打开计算器转换得到4202514

打开wsl：输入nc 182.92.108.71 30210回车

```
atom@LAPTOP                              $ nc 182.92.108.71 30210
password:4202514
you are right!
```

然后就可以输linux指令了，直接ls+cat：

hgame{W3lCOme_t0_Hg4m3_2222Z222zO2l}

SteinsGate2

letter

once

Crypto

まひと

这题提供了一坨莫斯电码

---../-..../-..-./.----/-----/----./-..-./.----/-----/---../-..-./.----/.----/-----/-..-./----./-----/-..-./---../--.../-..-./.....//...--/-..-./.---/-----/---../-..-./.----/.----/-..-./.----/-----/----./-..-./---../.....-/-..-./.----/.----/-..../-..-./---../....-/-..-./--...../.----/-..-./.---/-----/---../-..-./.----/.----/....-/-..-./----./--.../-..-./---../....-/-..-./.----/.----/..---/-..-./.....-/--.../-..-./---../-..../-..-./.----/---.----/-----./-..-./.----/.----/-..../-..-./.----/-----/-----/-..-./.----/-----/--.../-..-./.----/.----/..---/-..-./.----/---.../.....-/-..-./--.../...--/-..-./---../....-/-..-./--.../-----/-..-./---../----./-..-./.----/-----/-----/-..-./-..../----./-..-./--.../----

-/-..-./...../..---/-..-./-----/-----/-..-./---../...--/-..-./--.../-----/-..-./.----/.----/.----/-..-./----./----./-..-./-....----./-..-./....-/---../-..-./.----/..---/-----/-..-./.----/-----/.----/-..-./....-/---../-..-./....-/---../-..-./.----/.----/....-/-..-./--.../----./-..-./---../---../-..-./.----/-----/....-/-..-./.----/..---/-----/-..-./.----/-----/.----/-..-./.----/.----/-----/-..-./--.../....-/-..-./---../.....-/-..-./---../....-/-..-./---../-....-/-..-./.....-/--.../-..-./--.../----./-..-./----./--.../-..-./.----/.----/-----/-..-./.....-/...--/-..-./.----/----
-/-....-/-..-./---../.....-/-..-./.----/-----/----./-..-./----./----./-..-./....-/---../-..-./.----/-----/.----/-..-./....-/....-/---
-/-..-./-....-/.----

```
from __future__ import print_function
import sys
import hexdump
import math
import os
import random
from simanneal import Annealer

# requires https://github.com/perrygeo/simanneal

KEYLEN = 16


def xor_strings(s, t):
    # https://en.wikipedia.org/wiki/XOR_cipher#Example_implementation
    """xor two strings together"""
    return "".join(chr(ord(a) ^ ord(b)) for a, b in zip(s, t))


def read_file(fname):
    file = open(fname, mode='rb')
    content = file.read()
    file.close()
    return content


def chunks(l, n):
    """divide input buffer by n-len chunks"""
    n = max(1, n)
    return [l[i:i + n] for i in range(0, len(l), n)]


trigrams = ["the", "and", "tha", "ent", "ing", "ion", "tio",
            "for", "nde", "has", "nce", "edt", "tis", "oft", "sth", "men"]
digrams = ["th", "he", "in", "er"]

# additional tuning may be needed:
BONUS_FOR_PRINTABLE = 1
BONUS_FOR_LOWERCASE = 3
BONUS_FOR_SPACE = 8
BONUS_FOR_DIGRAM = 8
BONUS_FOR_TRIGRAM = 16
PENALTY_FOR_DIGIT = -10


def fitness(state):
    fitness = 0
    state_as_string = "".join(map(chr, state))
    tmp = chunks(cipher_file, KEYLEN)
    plain = "".join(map(lambda x: xor_strings(x, state_as_string), tmp))
    for p in plain:
        if p in "qwertyuiopasdfghjklzxcvbnm":
            fitness = fitness+BONUS_FOR_LOWERCASE
        if p in "0123456789":
            fitness = fitness+PENALTY_FOR_DIGIT
        p = ord(p)
        if (p >= 0x20 and p <= 0x7E) or p == 0xA or p == 0xD:
```

```
            fitness = fitness+BONUS_FOR_PRINTABLE

    for digram in digrams:
        fitness = fitness + plain.count(digram)*BONUS_FOR_DIGRAM

    for trigram in trigrams:
        fitness = fitness + plain.count(trigram)*BONUS_FOR_TRIGRAM

    fitness = fitness + plain.count(" ")*BONUS_FOR_SPACE

    return fitness


class TestProblem(Annealer):

    def __init__(self, state):
        super(TestProblem, self).__init__(state)  # important!

    def move(self):
        # set random byte in state to random byte
        i = random.randint(0, len(self.state) - 1)
        self.state[i] = random.randint(0, 255)

    def energy(self):
        return -fitness(self.state)


# cipher_file=read_file("cipher1.txt")
cipher = b'H;<Z\x1a*=G \x14$\xO6fT:Ab\xO16\x17\xOO\',
[y@%\rfP$P/\r+C\xO4o&Sy>,H6T![6\xO1+PW ;\x15=F,\x1f/[/\x15
\t)V\x19,,\x15<U.\xOOfZ<]\'\x1ae=\x18:=\x1by`%\xO15\x15+Z7\xO4!\x17\x15*iA1Qm\xO
7$_-V6\x1beC\x1f*$F<X;\r5\x19h?
\x1d1\x17\x1e;iV8Zm\t*F\'\x15O\r)V\xO3*iA6\x14.\xO7*Z:Fb\t+SWE&A1Q?H%Z%E-
\x1b,C\x1e \'T5\x149\r%]&\\3\x1d DYE\x1OZ,\x14 \t?\x15&Z6H7R\x16#
O<\x14$\x1cj\x15*@6H<X\xO2=iW+U$\xO6f?!Fb\nOD\xOeo>Z+_$\xO6!\x15*P*\xO1+SW;!PyG.
\r(P;\x156\xO7eD\x12*"\x15S[8\x1cfF1X/\r1E\xOeo>]<Zm\x11)@hY-
\xO7.\x17\x16;iTyD,\xO1(A!
[%Fe=#\',G<\x14,\x1a#\x15;P4\r7V\x1bo;P8G"\xO65\x15.ZOH1_\x1e<g\x15\r\\
(HLS!G1\x1ce^\xO4o=]8@m\x1f#\x12:Pb\xOO$E\x13b>\\+Q)H2ZhY-\xO7.\x17\x11
;\x15S]9Ffz=Gb\t+T\x1e*\'AyU#\xOb#F<ZO\x1beZ\x166i[6@m\xOO\'C-
\x15*\t!\x17}.i[8Y(H Z:\x15+\x1ci\x17\x15:=\x15-\\(\x11f^&P5H1_\x16;iA1Q$\x1af?
\'B,H\'X\x13&,FyC(\x1a#\x15*T1\xO1&V\x1b#0\x15*M
\xO5#A:\\!\t)\x1bW.:\x15SC(\x1a#\x15<]-\x1b
\x17\x18)iE6@(\xO62\\)Yb\x187R\x13.=Z+Gm\xO74\x158G\'\x11k\x17}\x1b!P+Q+\xO74Pd\
x156\xOO,DW,(X<\x14$\xO6f])[&\x11e@\x1f*=]
<Fmb%]\'Z1\xO1+PW.iX8@(DfV)A!\xOO,Y\x1Oo-\\7Z(\x1afZ:\x15H\t3X\x1e+
[>\x14/\r/[/\x15-\xO6eC\x1f*iX<Z8H)ShTb\x1b+V\xO5#
[>\x18mb.@&RO\x11eG\x16,"\x156Rm\x1f)Y>P1H*EW-,T+Glb\x12T#Pb\te[\x18
"\x158@m\x11)@:\x15$\t&RW&\'\x15-\\(H+\\:G-\x1ae=\x16!-\x15OY,\xOf/[-
\x15#H)^\x19*iF-F,\xO1!]<\x15&\xO72YW;!Py>
\xO1"Q$PlH\x1cX\xO2h%YyG(\rfW\'A*H6^\x13*:\x156Rm\x11)@:\x15H\xOe$T\x12o(G<\x14=
\x1a#A<Lb\x1b<Z\x1a*=GOW,\xO4h\x15\x1c]+\x1be^\xO4oC^7[:\xO6fT;\x15
\xO1)V\xO3*;T5\x14>\x11+X-AO\x11eV\x19+i\\-\x13>HLB PO\reU\x18;!\x15*])\r5\x15-
\\6\xOO EW< Q<\x14"\xOefA \\1HOS\x1e9 QOZ*H*\\&Pb\t5G\x12.;\x154[?
\rfZ:\x15.\r6DW;!PyG,\xO5#\x1bBf-H-
R\xO5*i\\*\x149\xOO#\x15.Y#\xOf\x7f\x17}\'.T4Q6OvG\x17\\wE$h\xO2\x1czS\xOc\xO5f\\
(QlS\x17&+N(\xOcxE\x11\xO7?\x15L'
# print(len(cipher))
```

```
cipher_file = str(cipher, encoding="utf-8")

init_state = [0 for i in range(KEYLEN)]

test = TestProblem(init_state)
# increase if you're not satisfied with result:
test.steps = 200000
# since our state is just a list, slice is the fastest way to copy
test.copy_strategy = "slice"
#test.copy_strategy = "deepcopy"
state, e = test.anneal()

possible_key = "".join(map(chr, state))
print("state/key:")
# hexdump.hexdump(possible_key)

print("decrypted:")
tmp = chunks(cipher_file, KEYLEN)
plain = "".join(map(lambda x: xor_strings(x, possible_key), tmp))
print(plain)
# hexdump.hexdump(plain)
```

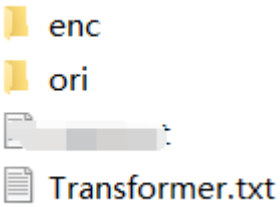
运行后结果马上就出了：

> Symmetry in art is when the elements of
> a painting or drawing balance each other
> out. This could be the objects themselves,
> but it can also relate to colors and
> other compositional techniques.
> You may not realize it, but your brain
> is busy working behind the scenes to seek
> out symmetry when you look at a painting.
> There are several reasons for this. The
> first is that we're hard-wired to look for
> it. Our ancient ancestors may not have had
> a name for it, but they knew that their
> own bodies were basically symmetrical, as
> were those of potential predators or prey.
> Therefore, this came in handy whether
> choosing a mate, catching dinner or
> avoiding being on the menu of a snarling,
> hungry pack of wolves or bears!
> Take a look at your face in the mirror
> and imagine a line straight down the
> middle. You'll see both sides of your
> face are pretty symmetrical. This is
> known as bilateral symmetry and it's
> where both sides either side of this
> dividing line appear more or less the same.
> So here is the flag:
> hgame{X0r_i5-a_uS3fU1+4nd$fUNny_C1pH3r}

hgame{X0r_i5-a_uS3fU1+4nd$fUNny_C1pH3r}

Transformer

> 所有人都已做好准备,月黑之时即将来临,为了击毁最后的主控能量柱,打开通往芝加哥的升降桥迫在眉睫 看守升降桥的控制员已经失踪,唯有在控制台的小房间留下来的小纸条,似乎是控制员防止自己老了把密码忘记而写下的,但似乎都是奇怪的字母组合,唯一有价值的线索是垃圾桶里的两堆被碎纸机粉碎的碎纸,随便查看几张,似乎是两份文件,并且其中一份和小纸条上的字母规律有点相像



密文：Tqh ufso mnfcyh eaikauh kdkoht qpk aiud zkhc xpkkranc uayfi kfieh 2003, oqh xpkkranc fk "qypth{hp5d_s0n_szi^3ic&qh11a_}",Dai'o sanyho oa pcc oqh dhpn po oqh hic.

这题根据题意“其中一份和小纸条上的字母规律有点相像”

将enc和ori中文件中的字符对应起来（我用文件大小排序，相同文件大小的文件往往就是对应的明文和密文）

少量对不上的文件也可以利用已知的子母对应关系找到未知字母的映射关系

a-p
b-m
c-e
d-c
e-h
f-s
g-y
h-q
i-f
j-w
k-j
l-u
m-t
n-i
o-a
p-x
q-l
r-n
s-k
t-o
u-z
v-v
w-r
x-g
y-d
z-b

最后得出这么一个对应关系，左边为明文，右边为密文

写了个小程序将密文翻译过来：

```
#include <stdio.h>
int main(){
    char map[26] =
{'o','z','d','y','c','i','x','e','n','k','s','q','b','r','t','a','h','w','f','m'
,'l','v','j','p','g','u'};
    char temp;
    while ((temp = getchar())!=0){
        if (temp>='a'&&temp<='z') printf("%c", map[temp-'a']);
        else printf("%c",temp);
    }
    return 0;
}
```

翻译得到：

The lift bridge console system has only used password login since 2003, the password is "hgame{ea5y_f0r_fun^3nd&he11o_}"
,Don't forget to add the year at the end.

hgame{ea5y_f0r_fun^3nd&he11o_}

MISC

Base全家福

这题没什么可说的

拿到base64密文：
R1k0RE1OWldHRTNFSU5SVkc1QkRLTlpXR1VaVENOUlRHTVlETVJCV0dVMlVNTlpVR01ZREtSUlVIQTJET01aVUdSQ0RHTVpWSVlaVEVNWlFHTVpER01KWElRPT09PT09

和题目一样base全家桶

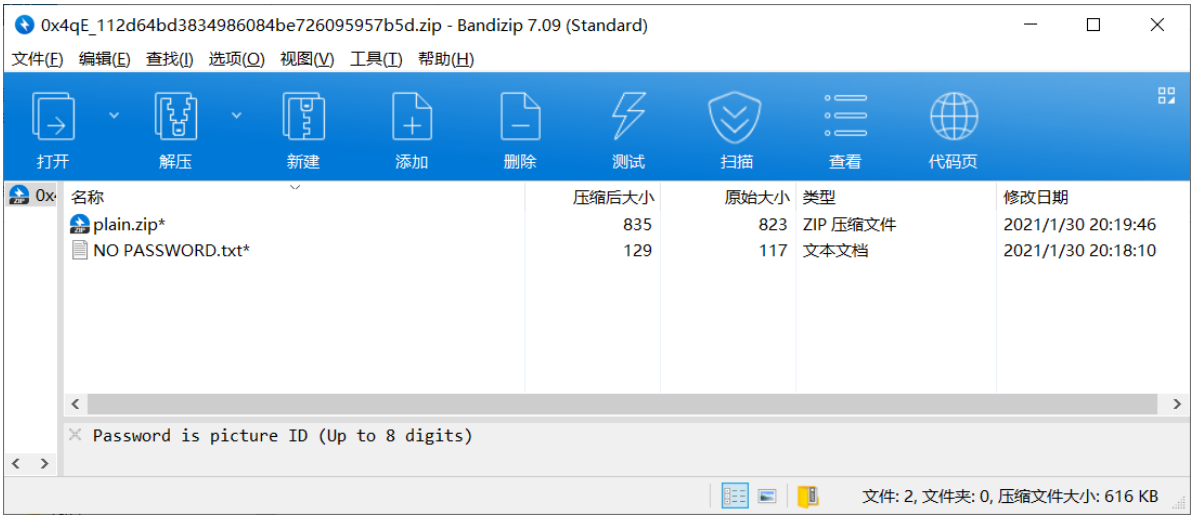
base64-》base32-》base16

得到：

hgame{We1c0me_t0_HG4M3_2021}

不起眼压缩包的养成的方法

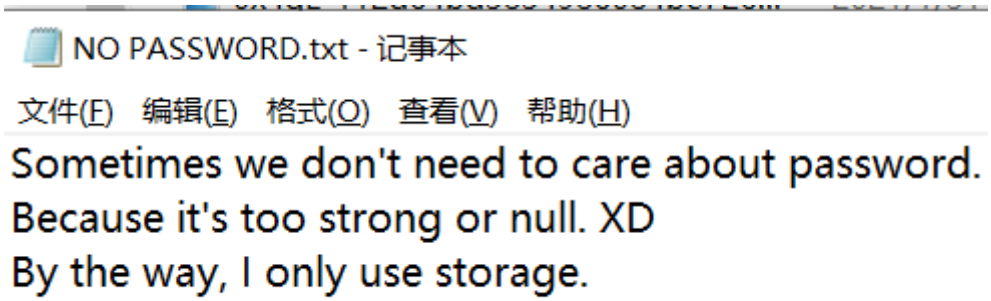
拿到的题目是一张图片，看题目很明显能想到将后缀改zip（用binwalk看一眼显然更严谨）



这边提到密码是八位的图片id，可以直接爆破出来，但我选择识图

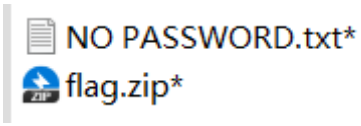
得到zip密码：70415155　解压

查看其中NO PASSWORD.txt的内容



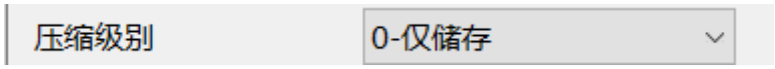
提到有时候密码太强或者完全没什么用，以及他只用存储模式打包zip

查看另外一个文件plain.zip

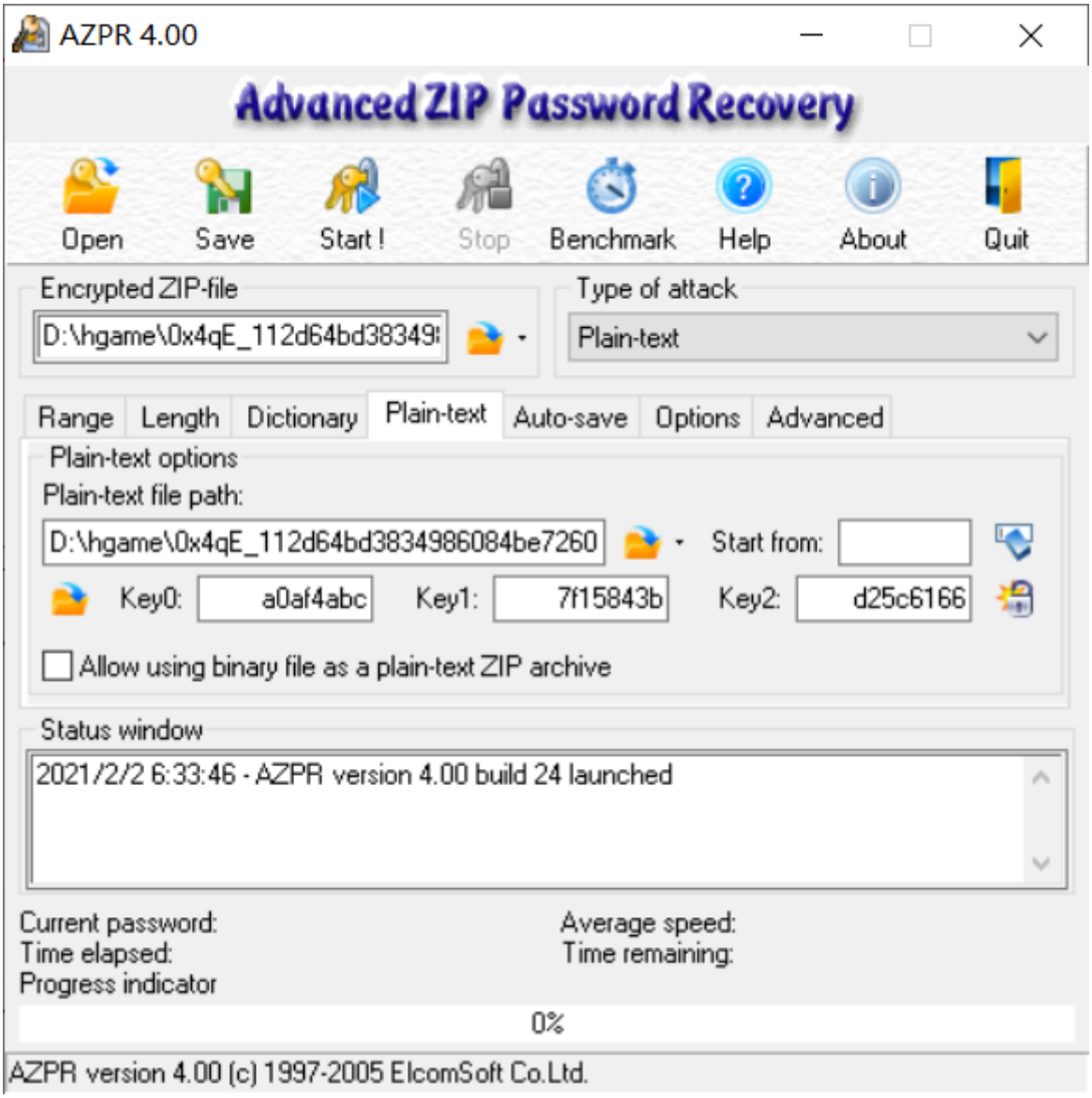


发现其中有一个同样的NO PASSWORD.txt（文件大小等完全相同），那么可以考虑使用明文攻击

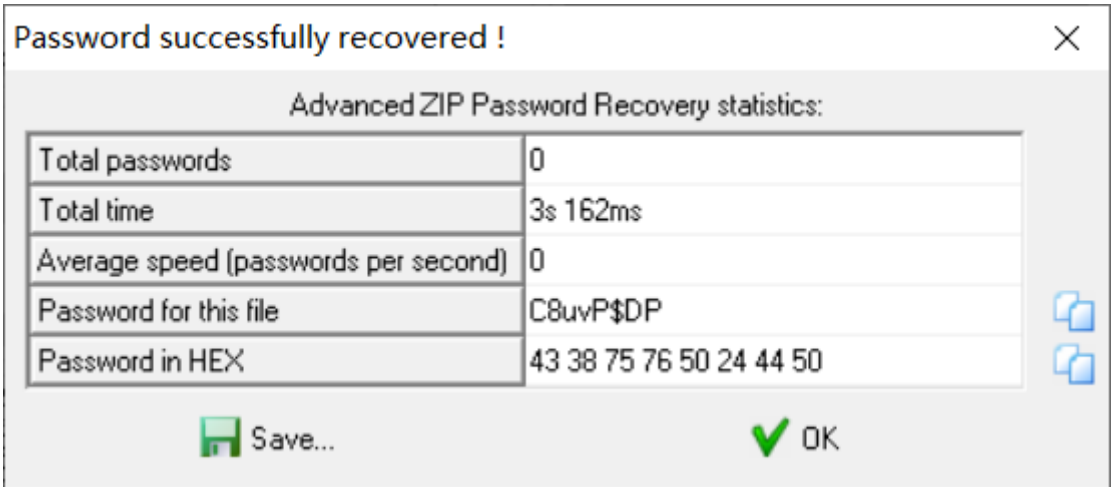
先把NO PASSWORD.txt打成zip，压缩级别设为仅储存



然后打开AZPR，设定明文攻击参数

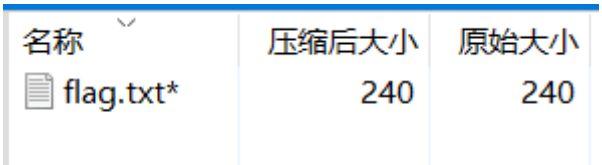


3秒出结果：C8uvP$DP

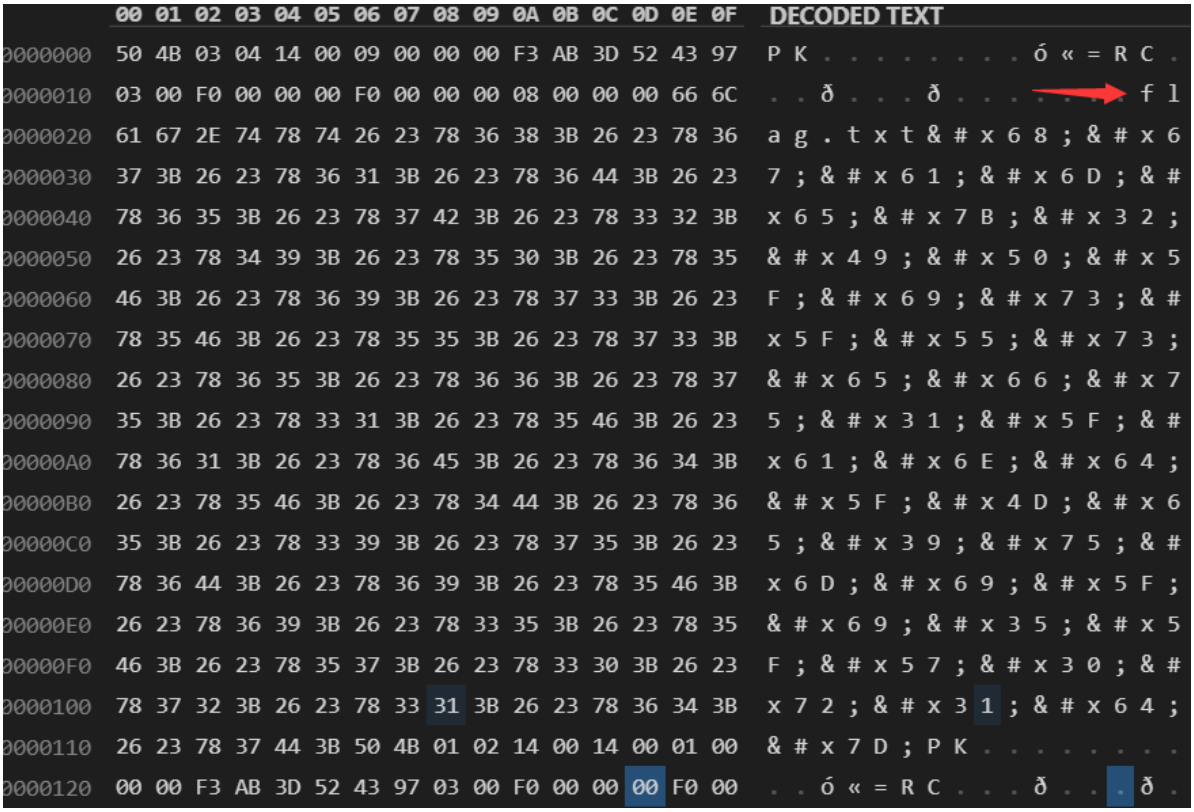


解压plain.zip得到flag.zip

查看flag.zip，只有一个加密文件



没有任何提示，杀去看HEX



```
          00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  DECODED TEXT
00000000  50 4B 03 04 14 00 09 00 00 00 F3 AB 3D 52 43 97  P K . . . . . . . . ó « = R C .
00000010  03 00 F0 00 00 00 F0 00 00 00 08 00 00 00 66 6C  . . ð . . . ð . . . . . . . f l
00000020  61 67 2E 74 78 74 26 23 78 36 38 3B 26 23 78 36  a g . t x t & # x 6 8 ; & # x 6
00000030  37 3B 26 23 78 36 31 3B 26 23 78 36 44 3B 26 23  7 ; & # x 6 1 ; & # x 6 D ; & #
00000040  78 36 35 3B 26 23 78 37 42 3B 26 23 78 33 32 3B  x 6 5 ; & # x 7 B ; & # x 3 2 ;
00000050  26 23 78 34 39 3B 26 23 78 35 30 3B 26 23 78 35  & # x 4 9 ; & # x 5 0 ; & # x 5
00000060  46 3B 26 23 78 36 39 3B 26 23 78 37 33 3B 26 23  F ; & # x 6 9 ; & # x 7 3 ; & #
00000070  78 35 46 3B 26 23 78 35 35 3B 26 23 78 37 33 3B  x 5 F ; & # x 5 5 ; & # x 7 3 ;
00000080  26 23 78 36 35 3B 26 23 78 36 36 3B 26 23 78 37  & # x 6 5 ; & # x 6 6 ; & # x 7
00000090  35 3B 26 23 78 33 31 3B 26 23 78 35 46 3B 26 23  5 ; & # x 3 1 ; & # x 5 F ; & #
000000A0  78 36 31 3B 26 23 78 36 45 3B 26 23 78 36 34 3B  x 6 1 ; & # x 6 E ; & # x 6 4 ;
000000B0  26 23 78 35 46 3B 26 23 78 34 44 3B 26 23 78 36  & # x 5 F ; & # x 4 D ; & # x 6
000000C0  35 3B 26 23 78 33 39 3B 26 23 78 37 35 3B 26 23  5 ; & # x 3 9 ; & # x 7 5 ; & #
000000D0  78 36 44 3B 26 23 78 36 39 3B 26 23 78 35 46 3B  x 6 D ; & # x 6 9 ; & # x 5 F ;
000000E0  26 23 78 36 39 3B 26 23 78 33 35 3B 26 23 78 35  & # x 6 9 ; & # x 3 5 ; & # x 5
000000F0  46 3B 26 23 78 35 37 3B 26 23 78 33 30 3B 26 23  F ; & # x 5 7 ; & # x 3 0 ; & #
00000100  78 37 32 3B 26 23 78 33 31 3B 26 23 78 36 34 3B  x 7 2 ; & # x 3 1 ; & # x 6 4 ;
00000110  26 23 78 37 44 3B 50 4B 01 02 14 00 14 00 01 00  & # x 7 D ; P K . . . . . . . .
00000120  00 00 F3 AB 3D 52 43 97 03 00 F0 00 00 00 F0 00  . . ó « = R C . . . ð . . . ð .
```

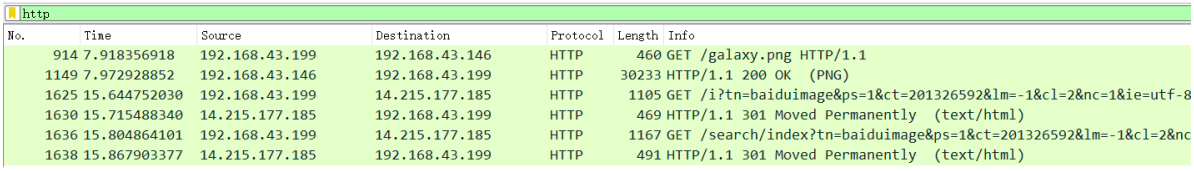
可以很明显看到flag.txt中的内容，取出，丢到unicode解码器，直接出flag

hgame{2IP_is_Usefu1_and_Me9umi_i5_W0r1d}

Galaxy

这题是拿到了一个wireshark的记录文件

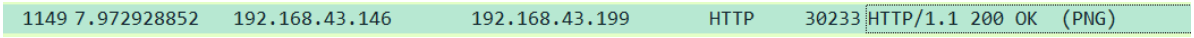
打开筛选器，直接把http请求筛选出来



```
http
No.     Time           Source          Destination     Protocol Length Info
    914 7.918356918    192.168.43.199  192.168.43.146  HTTP      460 GET /galaxy.png HTTP/1.1
   1149 7.972928852    192.168.43.146  192.168.43.199  HTTP    30233 HTTP/1.1 200 OK  (PNG)
   1625 15.644752030   192.168.43.199  14.215.177.185  HTTP     1105 GET /i?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc=1&ie=utf-8
   1630 15.715488340   14.215.177.185  192.168.43.199  HTTP      469 HTTP/1.1 301 Moved Permanently  (text/html)
   1636 15.804864101   192.168.43.199  14.215.177.185  HTTP     1167 GET /search/index?tn=baiduimage&ps=1&ct=201326592&lm=-1&cl=2&nc
   1638 15.867903377   14.215.177.185  192.168.43.199  HTTP      491 HTTP/1.1 301 Moved Permanently  (text/html)
```

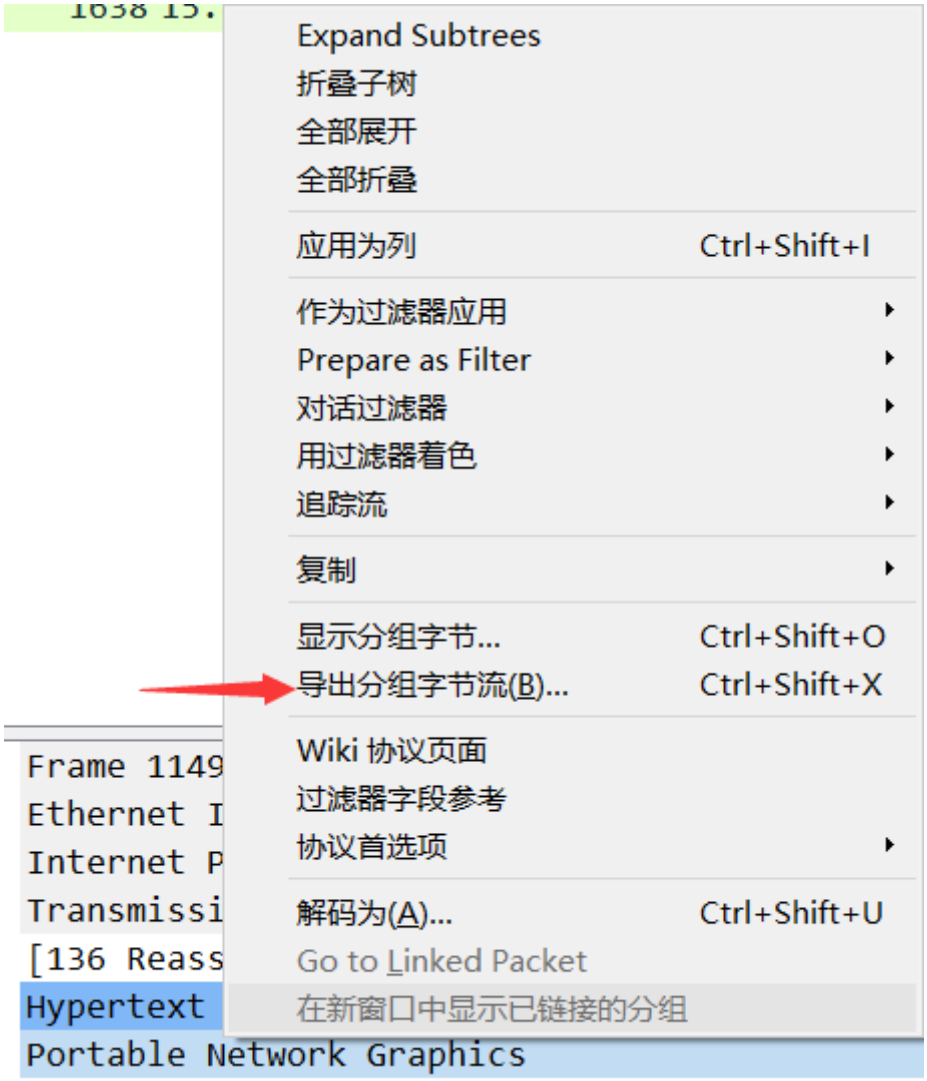
可以看到第一个请求/galaxy.png是200，而其他都是301，那么应该是从第一个请求入手

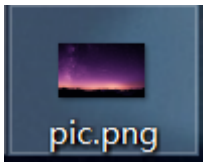
选中服务器返回的数据



```
1149 7.972928852   192.168.43.146   192.168.43.199   HTTP   30233 HTTP/1.1 200 OK  (PNG)
```

在下面选中“portable network .....”右键-》导出分组字节流，然后根据接收的文件类型，直接保存成png文件





导出后发现用honeyView打不开，怀疑照片被改过，使用PCRT（可在github下载）尝试修复文件，发现是crc校验码出错，这里我走了弯路，选择修复校验码，但是还是打不开图片，兜兜转转弄了一个下午。

crc校验码只和png文件头（文件头相关知识一定要认真看）有关，所以校验码出错就意味着文件头遭到了修改，ctf的一种图片隐写操作就是修改图片高度来隐藏flag

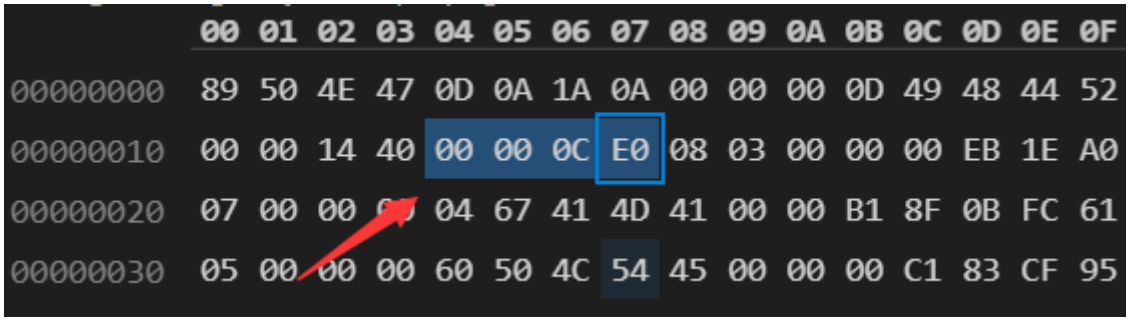
所以找了个现成的轮子来计算正确的图片高度信息

https://www.cnblogs.com/vict0r/p/13258286.html

```
import struct
import binascii
from Crypto.Util.number import bytes_to_long

img = open("galaxy.png", "rb").read()

for i in range(0xFFFF):
    stream = img[12:20] + struct.pack('>i', i) + img[24:29]
    crc = binascii.crc32(stream)
    if crc == bytes_to_long(img[29:33]):
        print(hex(i))
```

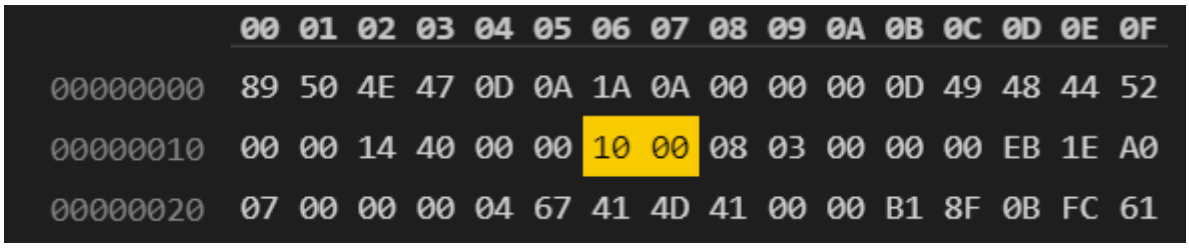
计算完得出图像高度数据块的值为0x1000，把图片用hex编辑器打开，修改高度部分的值

原始：



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52
00000010	00	00	14	40	00	00	0C	E0	08	03	00	00	00	EB	1E	A0
00000020	07	00	00	00	04	67	41	4D	41	00	00	B1	8F	0B	FC	61
00000030	05	00	00	00	60	50	4C	54	45	00	00	00	C1	83	CF	95

修改后：



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00000000	89	50	4E	47	0D	0A	1A	0A	00	00	00	0D	49	48	44	52
00000010	00	00	14	40	00	00	10	00	08	03	00	00	00	EB	1E	A0
00000020	07	00	00	00	04	67	41	4D	41	00	00	B1	8F	0B	FC	61

保存，图片可以打开

hgame{Wh4t_A_W0nderfu1_Wa11paper}

hgame{Wh4t_A_W0nderfu1_Wa11paper}

Word RE:MASTER

下载下来两个docx

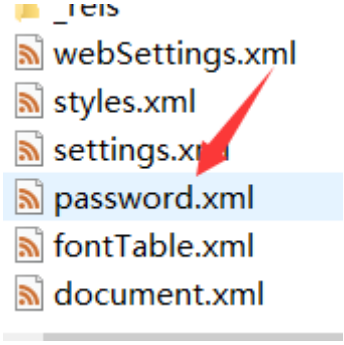
打开first.docx

Fuck！我的脑子好疼！这可能是音游癌发作最严重的一次,躺在床上很想打交互，嘴里念叨：O-ooooooooo·AAAAE-A-A-I-A-U·JO-ooooooooooo·AAE-O-A-A-U-U-A·E-eee-ee-eee·AAAAE-A-E-I-E-A·JO-ooo-oo-oo-oo·EEEEO-A-AAA-AAAA,不行我得在brainpower 耗尽前把密码记下来。

无隐藏文字，图片中不存在隐写

打开maimai.docx，发现需要密码

尝试将first.docx改名为first.zip，打开后发现有password.xml



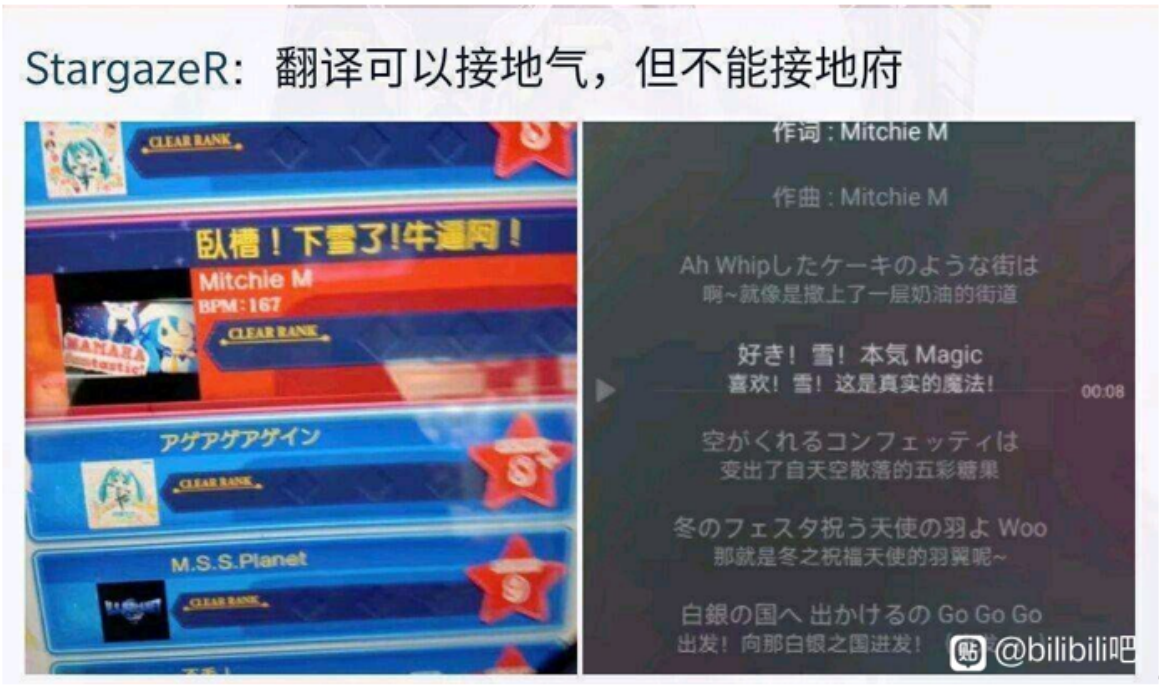
vscode打开得到密文：

+++++ +++[- >++++ ++++<]>+++ +.<++ +[->+ ++<]> ++.<+ ++[-> +++<] >+.<+ ++[-> ---<] >-.++ ++++. <+++[->--- <]>-. +++.+ .++++ ++++. <+++[->--- <]>-- ----. +.--- --..+ .++++ +++++ .<+++ [->-- -<]>- ----- .<

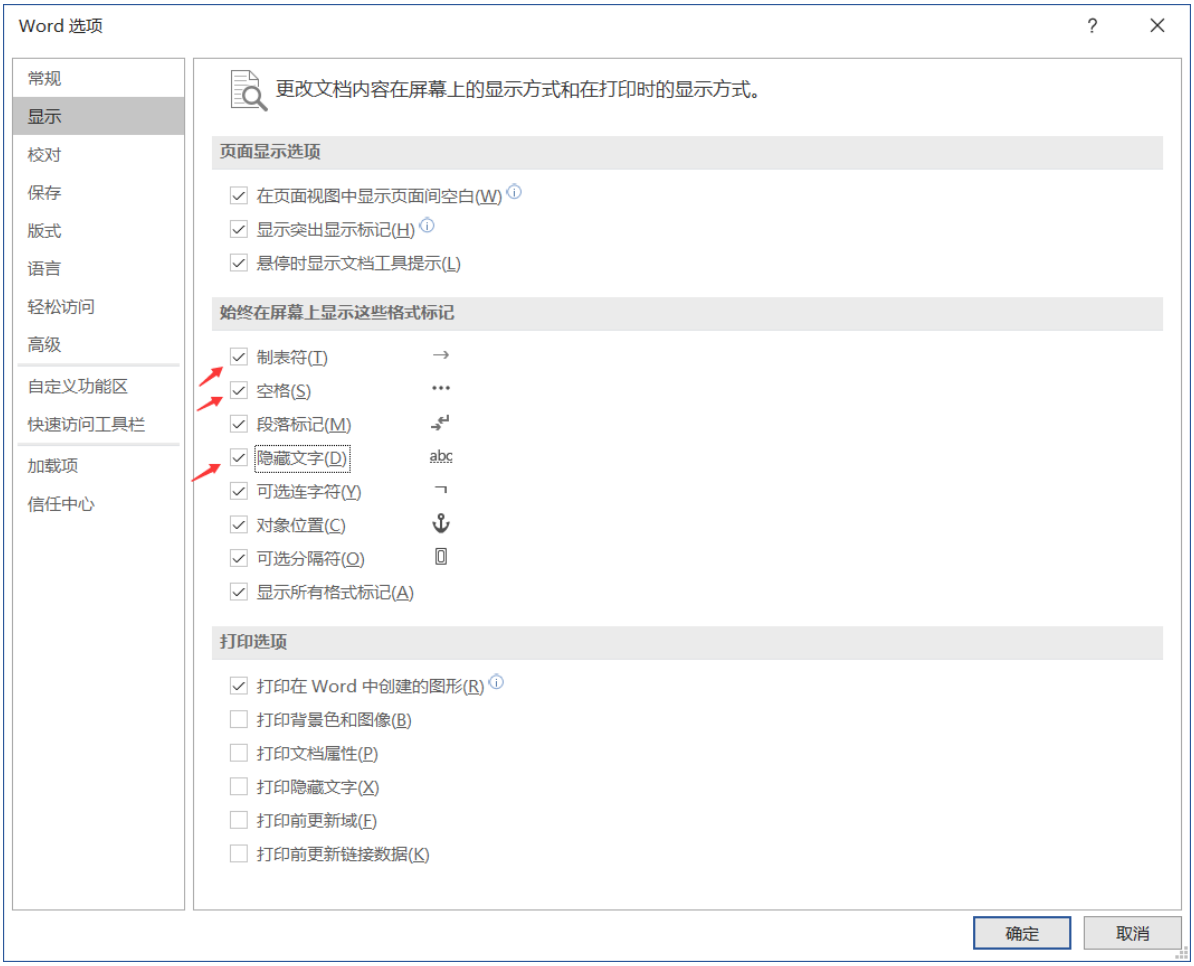
用brainfuck解码得到：DOYOUKNOWHIDDEN?

这里也给下面提示（隐藏字符）

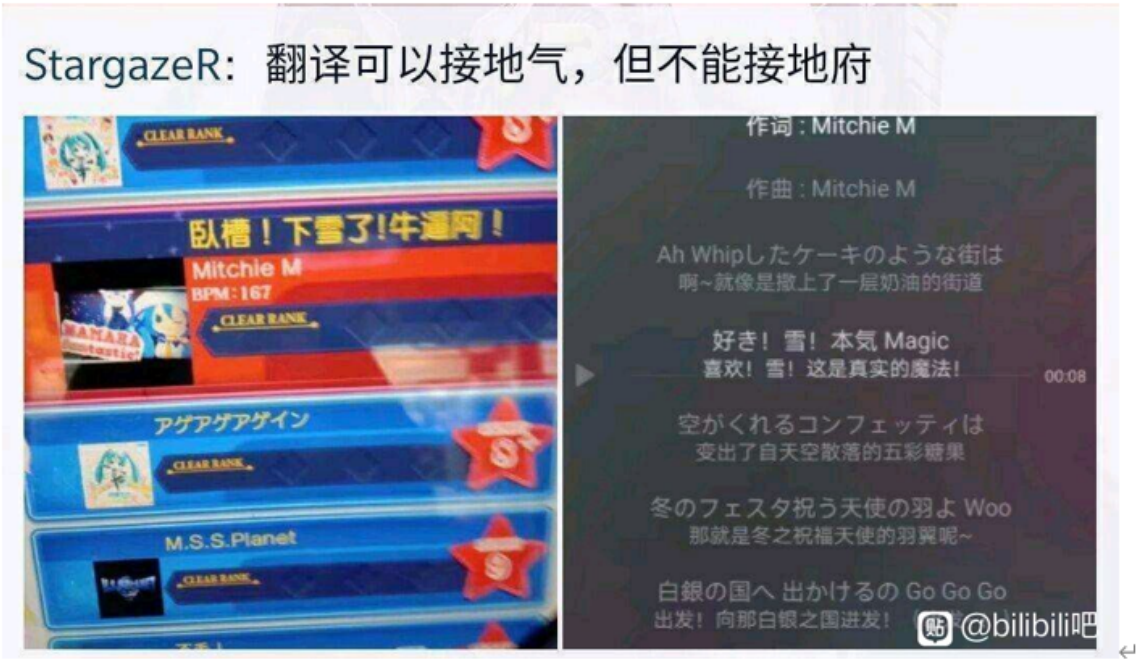
用密码打开docx，发现只有一张照片

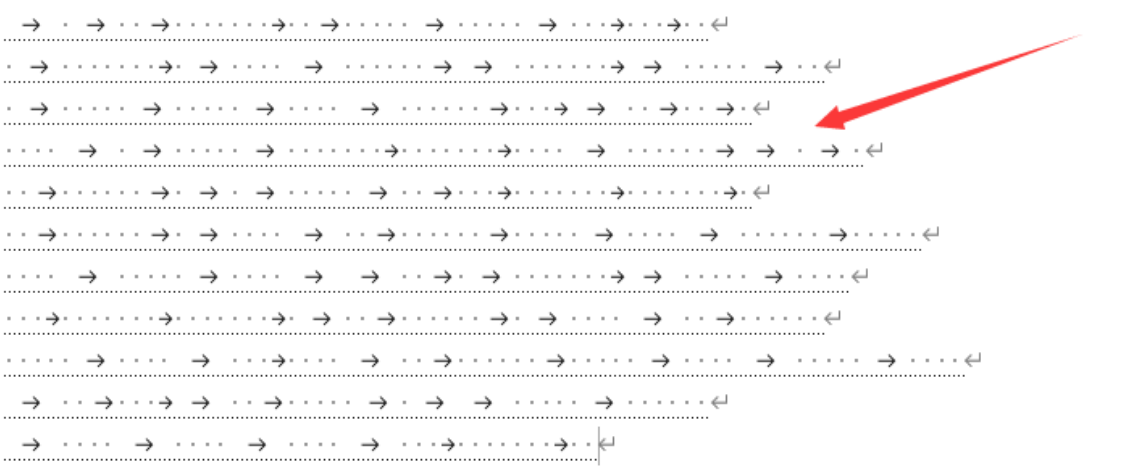


进入菜单，打开显示隐藏文字、制表符、空格



发现密文：





经过漫长的Google，终于明白这是snow隐写，图片中的“雪”这个提示我根本没有get到

https://joner11234.github.io/article/e9839e6f.html

http://darkside.com.au/snow/index.html

https://teamrocketist.github.io/2018/10/09/Forensics-InCTF-2018-Winter-Sport/

然后就是把word中的密文拿去解密，因为我没有办法直接将密文复制出来，所以写了个py来输出密文

```
import os
temp = '10100100000001001000001000001000100100\n\
01000000010100001000000110000000110000100\n\
01000001000001000010000001000110010010\n\
00001010000010000000010000000100001000000011010\n\
00100000010101000001001000010000001000000010\n\
00100000010100001001000000100000100001000000100000\n\
00001000001000011001010000000110000010000\n\
00010000000100000001010010000001010000010010000000\n\
00000100001000100001001000000100000100001000010000\n\
10010001100100000101100000010000000\n\
10000100001000010001000000100'
words = ''
for i in temp:
    if i == '1':
```

```
        words = words+'\t'
    if i == '0':
        words = words+' '
    if i == '\n':
        words = words+'\n'
with open('b.txt', 'w') as f:
    f.write(words)
```

然后下载了snow的win32位编解码程序

命令：.\SNOW.EXE -C .\b.txt

得到flag

hgame{Cha11en9e_Whit3_P4ND0R4_P4R4D0XXX}

The End