## 1.关闭ASLR

开启栈不可执行

关闭StackGuard

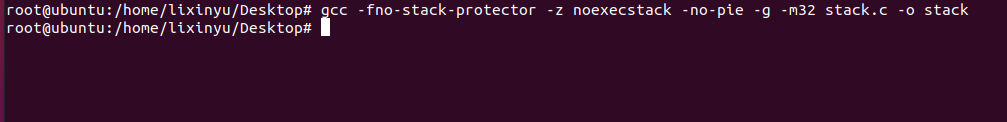
关闭ASLR

IMG_256

编译stack.c

-fno-stack-protector 关闭stackguard

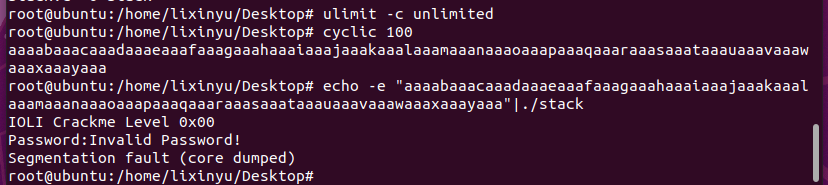
-z noexecstack 栈不可执行



构造输入串

IMG_256

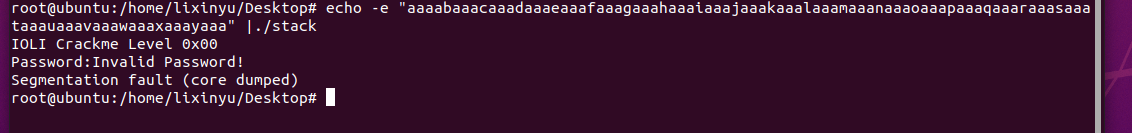
开启核心转储



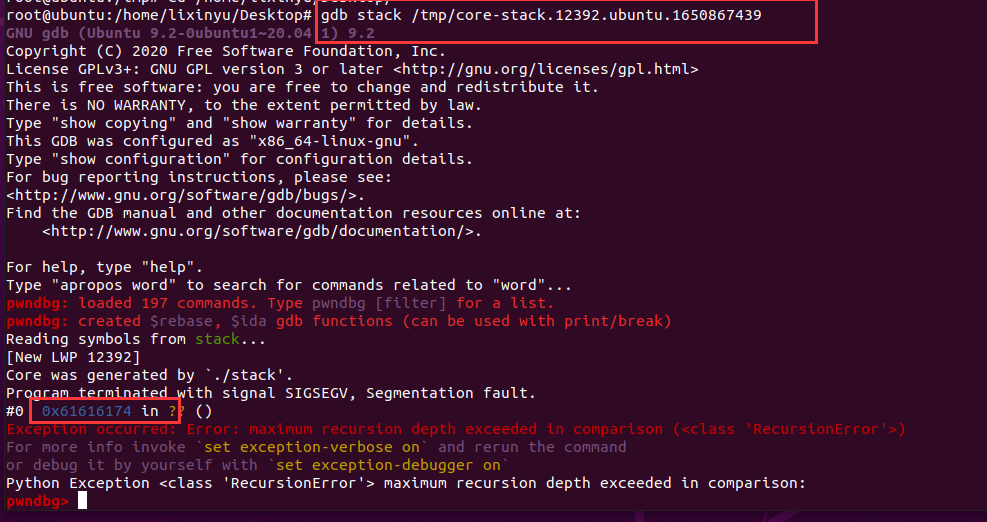
修改core文件生成位置和名称

IMG_256

输入创造溢出



使用gdb查看溢出EIP内容

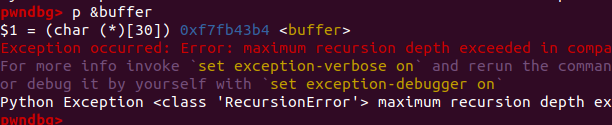


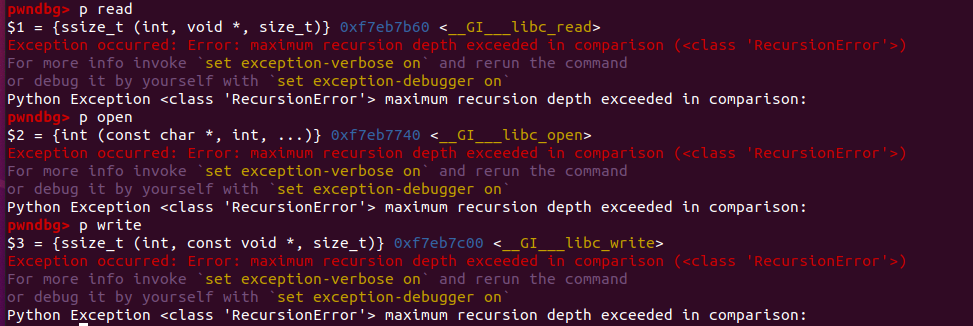
eip为0x61616174，ascii码为aaat，由于小端字节序，这里应该为taaa，使用cyclic指令判断偏移值，得到76

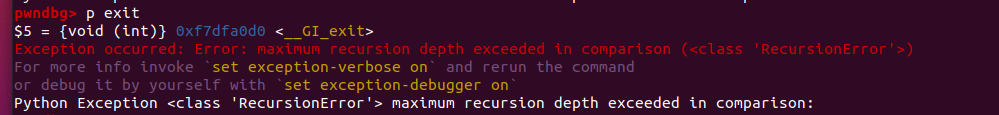
IMG_256

然后获得BUF地址、read、write、exit、open地址  
0xf7dfa0d0 exit  
0xf7eb7b60 read  
0xf7eb7740 open  
0xf7eb7c00 write

0xf7fb43b4 BUF



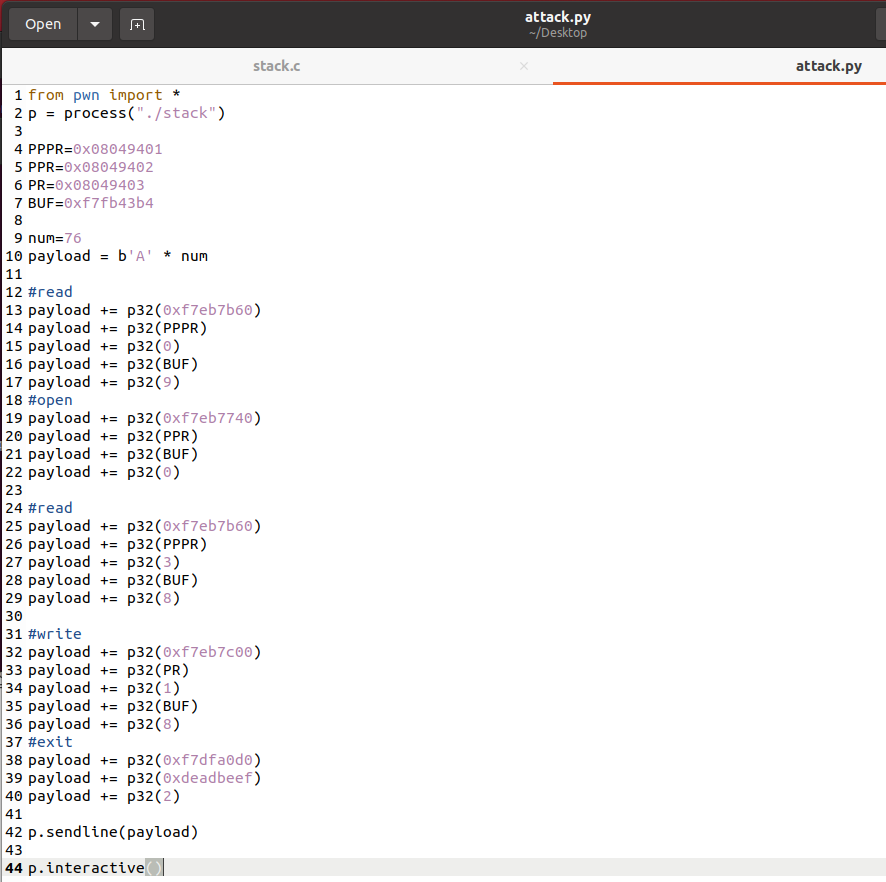




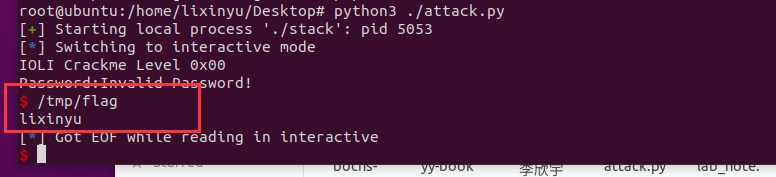
然后查看PPPR、PPR、PR的地址分别为0x08049401 0x08049402 0x08049403



根据这些地址构造攻击payload如下



进行攻击，成功查看到/tmp/flag中内容



## 2.关闭ASLR

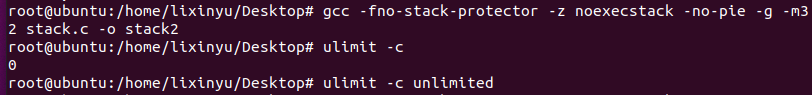
开启栈不可执行

关闭StackGuard

关闭ASLR

IMG_256

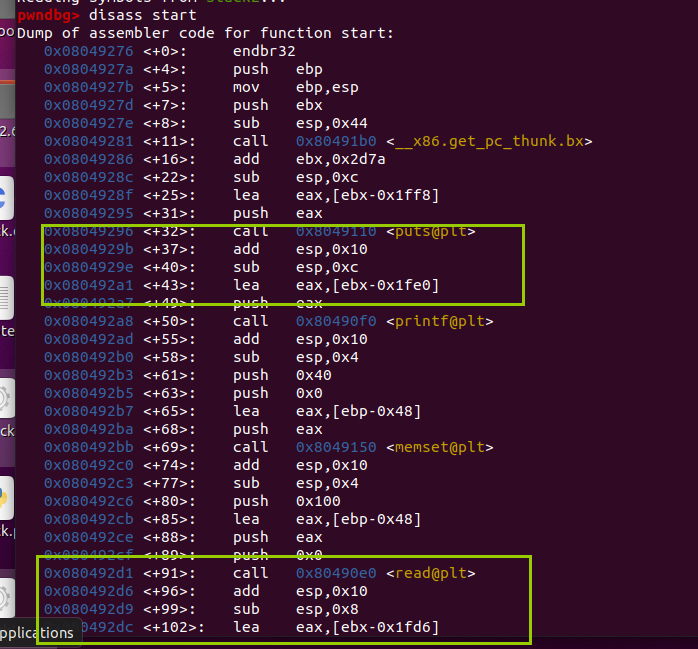
编译



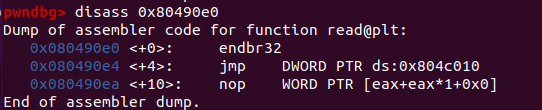
找到putplt和readgot，然后通过调用puts函数打印readgot指向的字符串，然后read的地址就封装在程序里了，然后再通过偏移求得其他函数地址，进行相关的操作。

gdb中查找puts plt和readgot

首先查找到putsplt是0x8049110 readplt是0x80490e0



进一步查找readgot得到0x804c010



然后获取main函数的地址

IMG_256

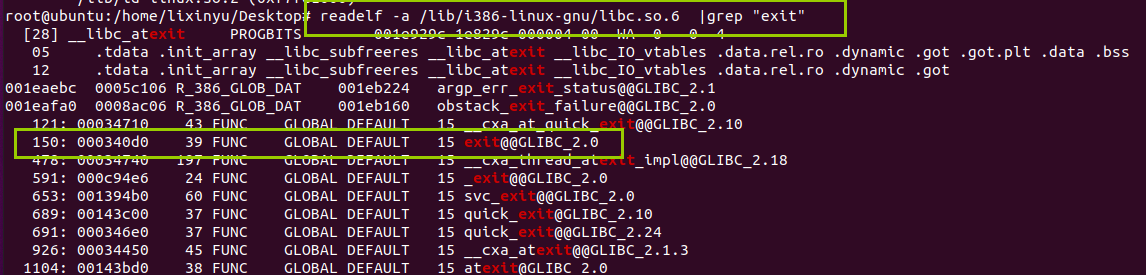
获取open、write、read、exit函数的偏移地址

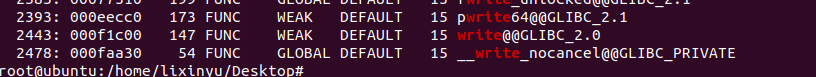
write 000f1c00

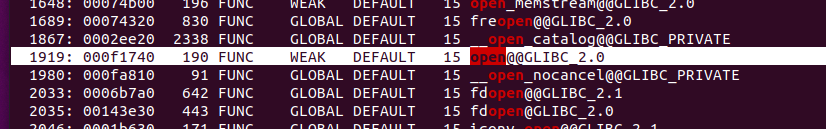
read 000f1b60

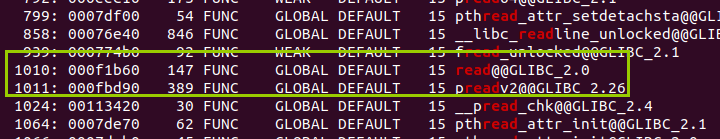
open 000f1740

exit 000340d0



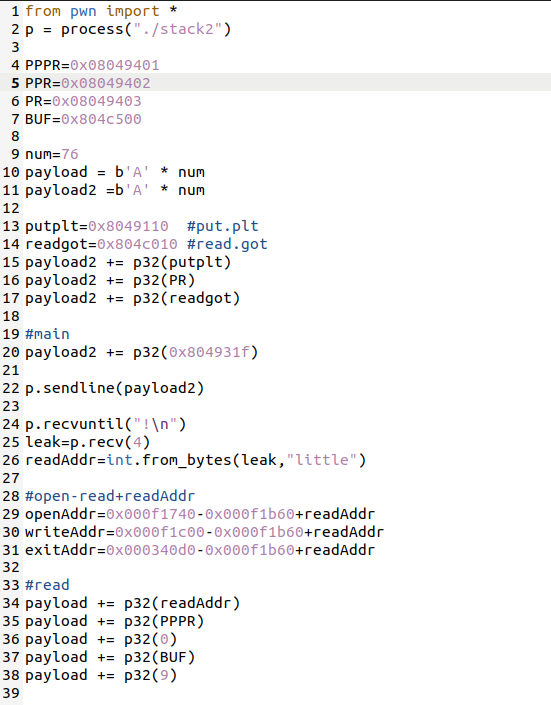


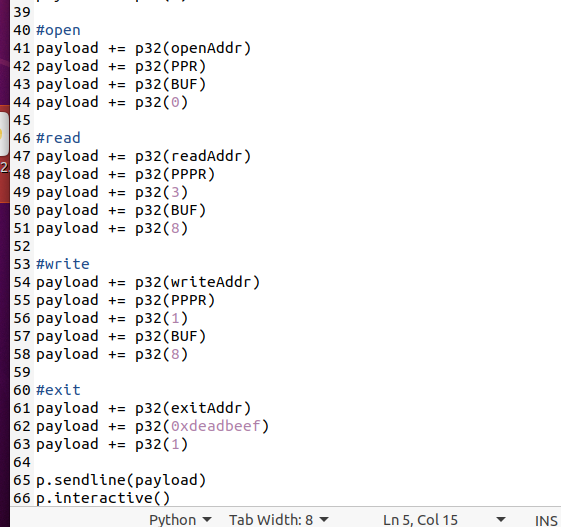




再获取PPR、PPPR、PR，这一部分地址与第一部分不开启ASLR的时候一样

将获得的地址构造payload





进行攻击，成功获得/tmp/flag内容

