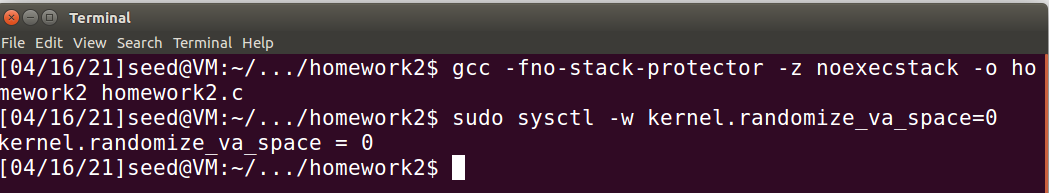
## return2libc攻击

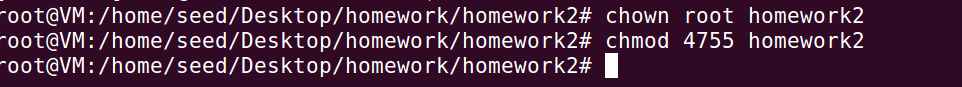
1. **设置环境背景**

打开栈保护，即栈内数据禁止作为代码执行，关闭stackguard（地址随机化），使用以下指令：

|  |
| --- |
| gcc -fno-stack-protector -z noexecstack -o homework2 homework2.c  sudo sysctl -w kernel.randomize\_va\_space=0 |



而后给予homework2程序权限并设置suid位，使用chown和chmod指令来实现：

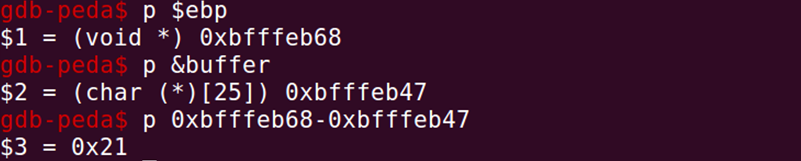


1. **调试程序计算缓冲区基址和返回地址间的距离**

计算缓冲区基址和返回地址间的距离是为了通过缓冲区溢出实现将system()函数地址覆盖到返回地址，从而实现return2libc攻击。

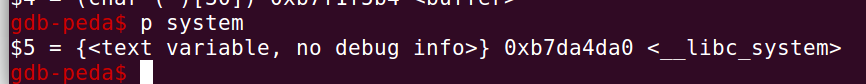
使用gdb对homework2进行调试：

1. 首先使用 b vulfunc在vulfunc函数处设置断点
2. 而后run执行
3. 使用p $ebp和p &buffer来查看这两个对应的地址，并将两者相减得出偏移距离为0x21，即33



1. 那么在33的基础上加上4就是ret的地址了，后续的步骤就是获取system()函数的地址、exit()函数的地址以及“\bin\sh”的地址。
2. **获取相关参数地址**
   1. 获取system()函数地址：

使用 p system获取system地址

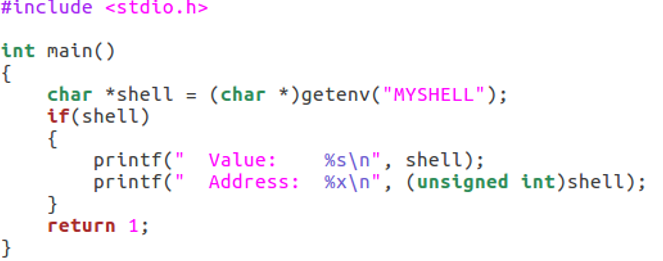


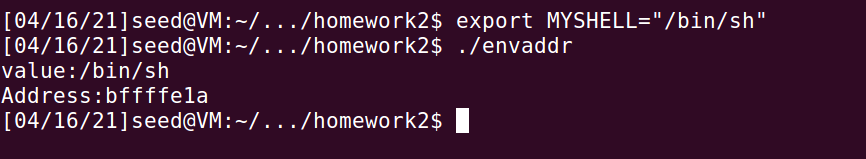
* 1. 获取exit()函数地址：

使用p exit获取exit地址



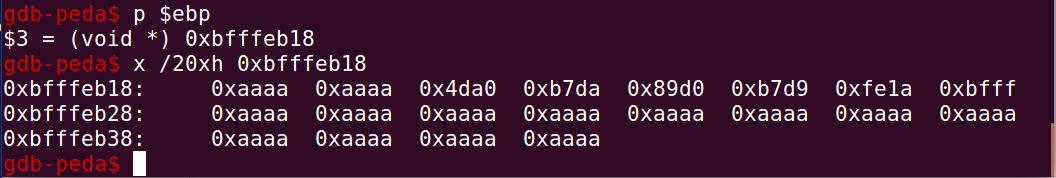
* 1. 编写程序查看envaddr查看system参数，环境变量/bin/sh的地址



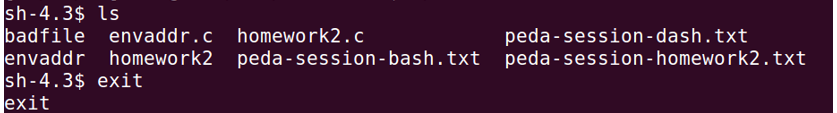


1. **创建badfile**

根据上述获得的/bin/sh地址，以及buffer和ebp的地址差，写一个创建badfile的脚本，其中在ebp+33+4的位置放入system的地址，在ebp+33+4+4的位置放入exit的地址，在ebp+33+4+4+4的位置放入参数”\bin\sh”的地址。重新执行后，查看ebp地址而后查看对应地址的数据：



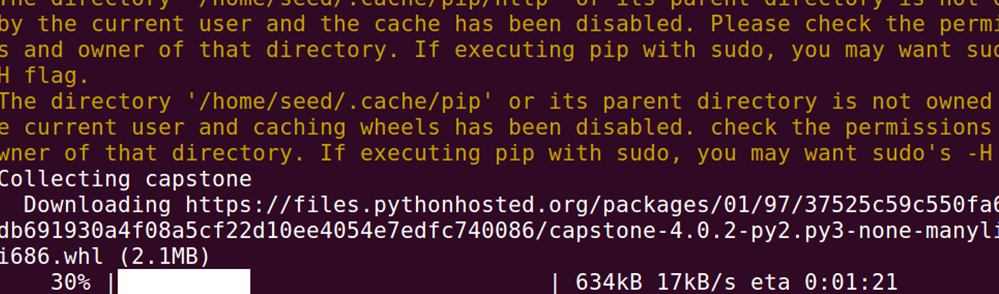
数据显然是满足条件的，那么执行homework2就可以绕过栈执行保护获取系统的shell权限。



## ROP攻击

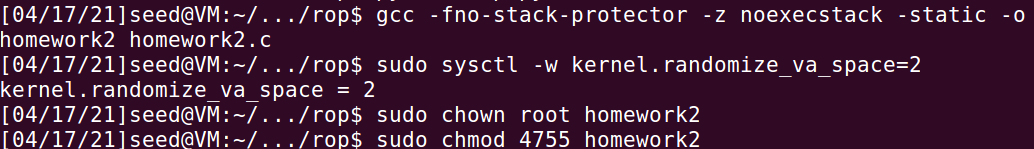
1. 计算返回地址与buffer的偏移地址（与return2libc一致）
2. 使用ROPgadget工具生成badfile文件
3. 安装ROPgadget，使用如下指令：

|  |
| --- |
| 使用pip安装ROPgadget，首先安装pip  sudo apt-get install python-pip  安装ROPgadget的依赖capstone  sudo pip install capstone  使用pip安装ROPgadget  sudo pip install ROPgadget |

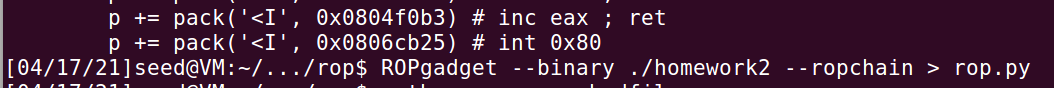


1. 环境设置，不可执行栈: 打开；StackGuard: 关闭；地址随机化: 打开

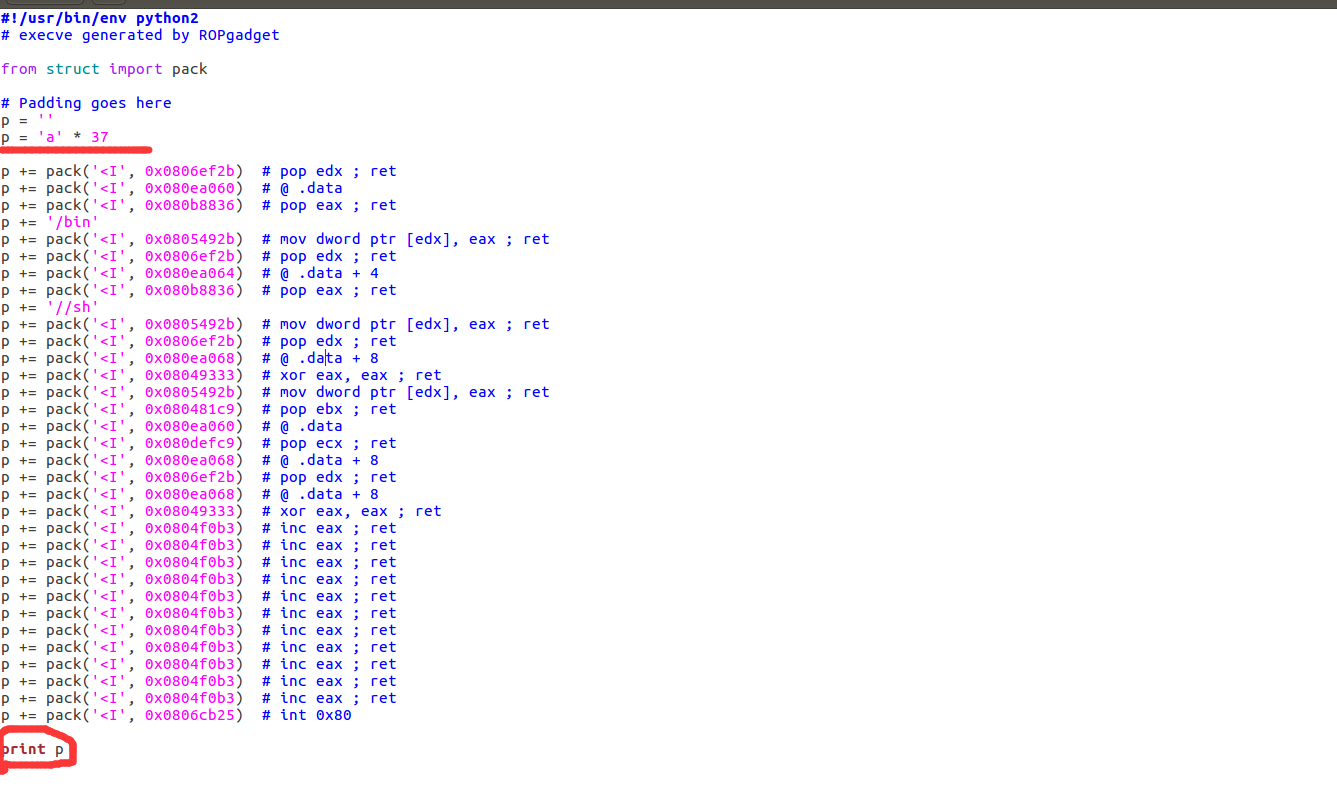
|  |
| --- |
| gcc -fno-stack-protector -z noexecstack -static -o homework2 homework2.c  sudo sysctl -w kernel.randomize\_va\_space=2  sudo chown root homework2  sudo chmod 4755 homework2 |



1. 使用ROPgadget生成gadget链，在生成的时候可以使用重定向导入在rop.py文件中，方便后续执行：



**注意：需要在py文件中添加print p语句以及对前37个字符进行填充**



1. 最后再执行rop.py生成相应的badfile文件，然后重新生成homework2并执行：

