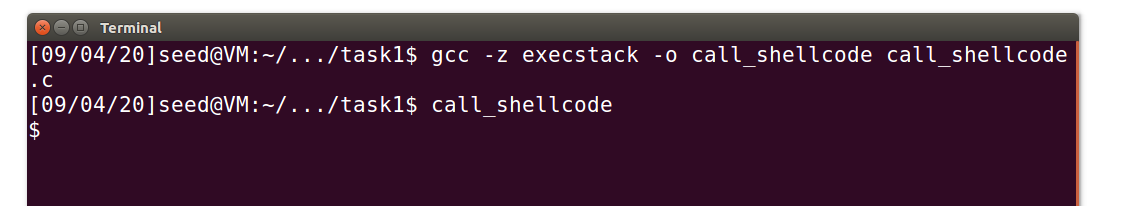
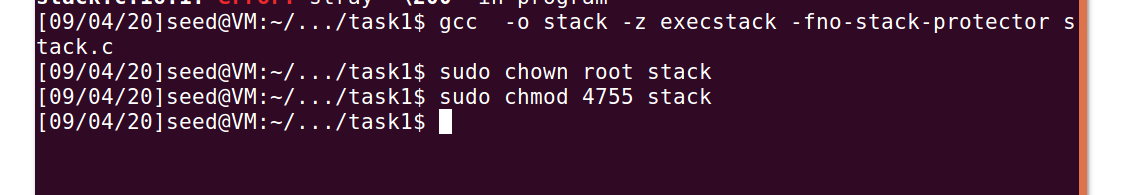
Task1:

编译call\_shellcode程序并运行，结果如图

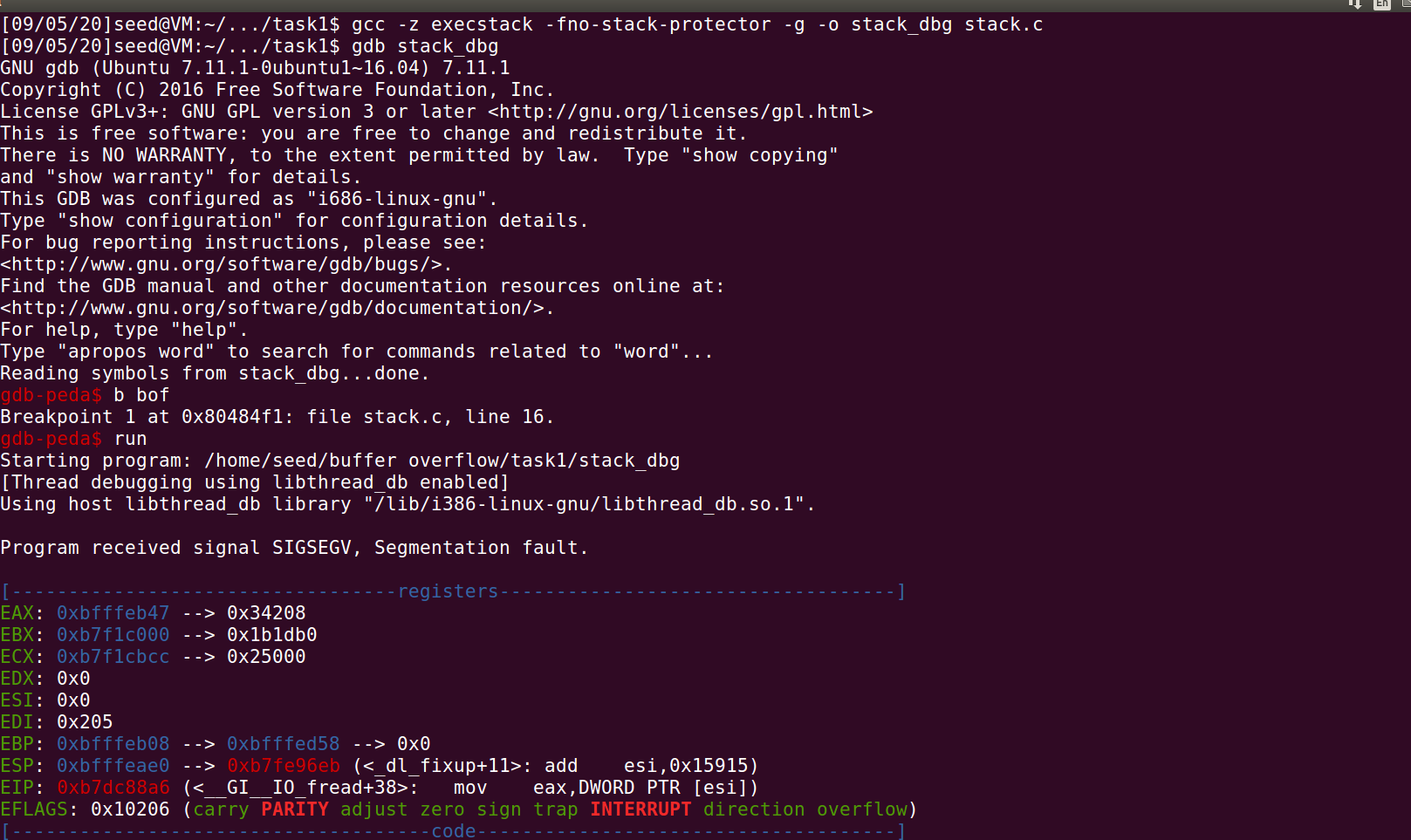


编译stack程序并更改为root的set-uid程序

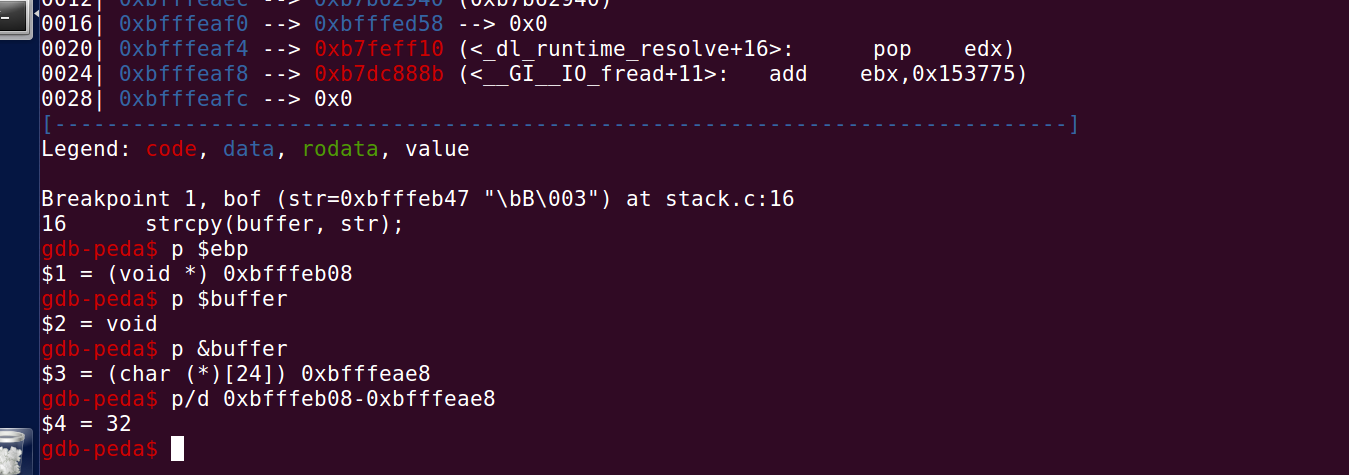


Task2:

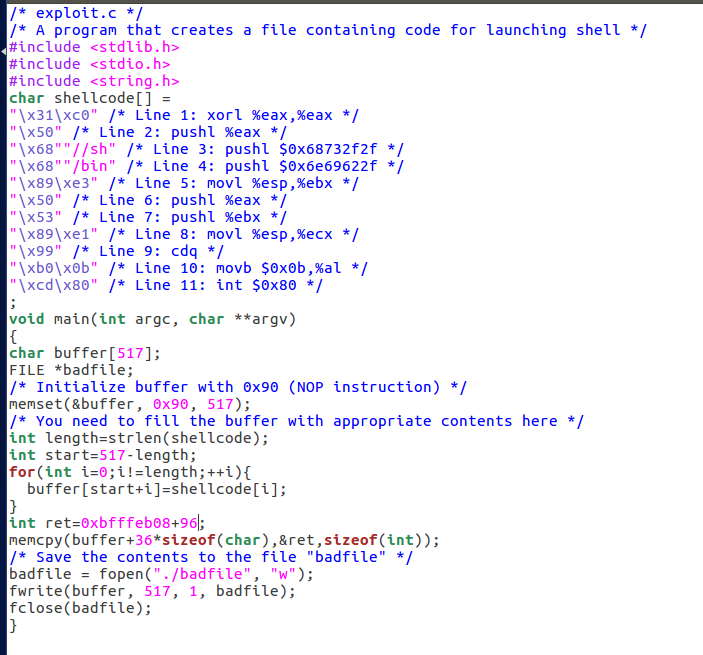
使用gdb调试stack程序，插入断点并运行



如下图所示，找到ebp的地址和buffer的地址，得出offset为32+4=36



下图为c语言版exploit程序，malicious code的地址0xbfffeb08+96通过多次尝试得出



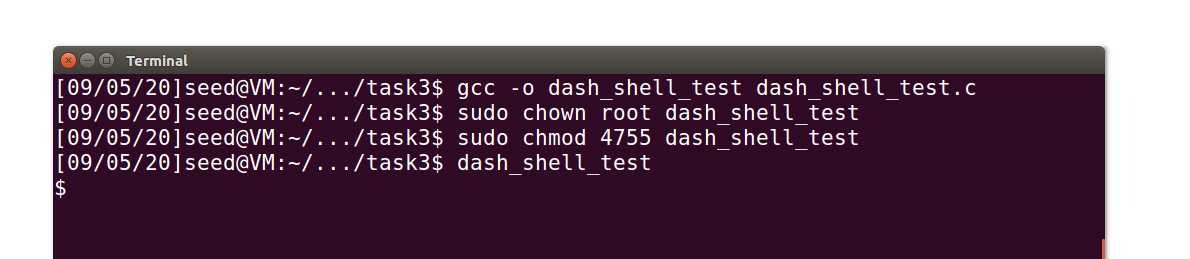
下图为生成正确的badfile后执行stack的结果，可见成功进入了shell的root模式



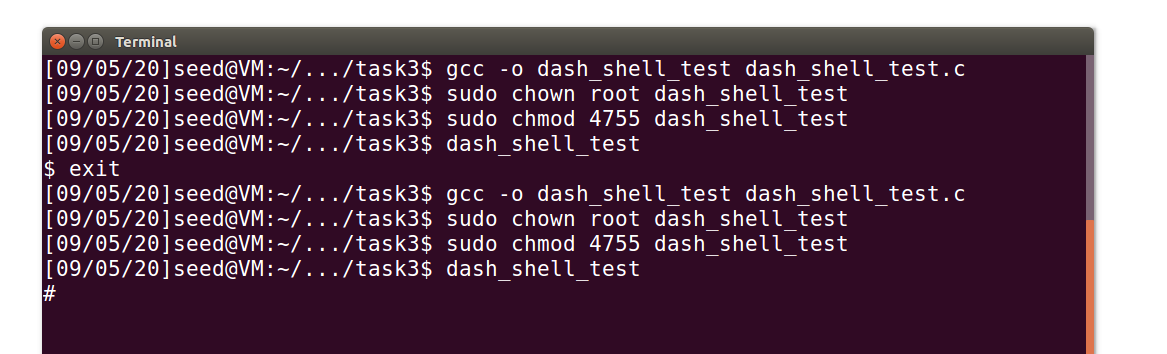
Task3:

首先将bash替换为dash

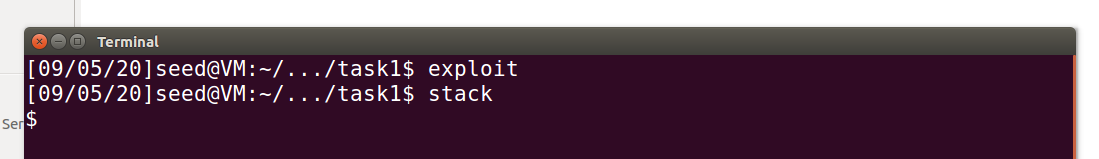
编译dash\_shell\_test并改为root dash\_shell\_test，运行结果如下，可见打开了非root的dash



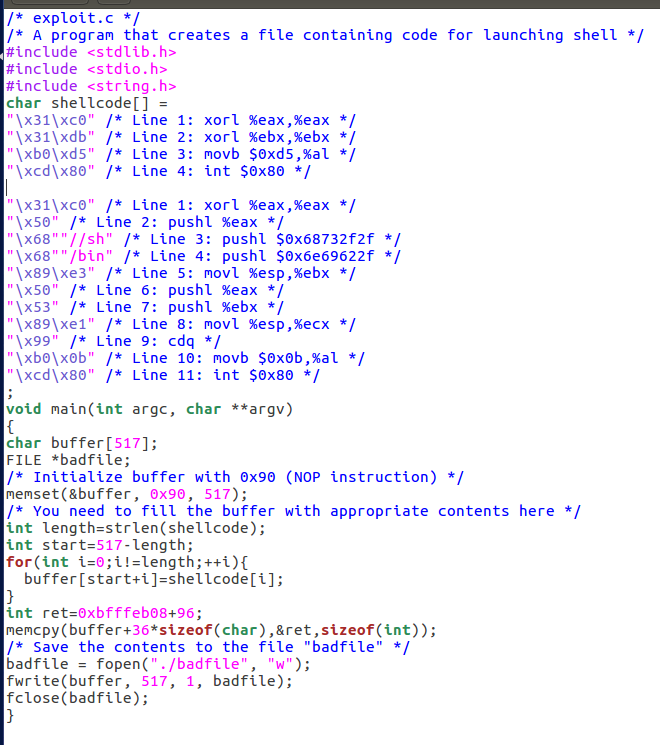
在dash\_shell\_test中将setuid(0)取消注释后重新编译运行，打开了root模式的dash，如图



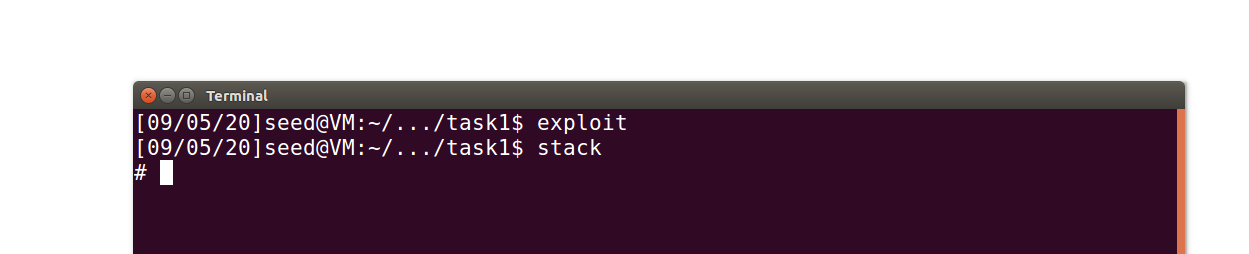
未添加setuid(0)机器指令的exploit生成badfile，stack执行后打开非root的dash



下图为添加setuid(0)机器指令的exploit程序

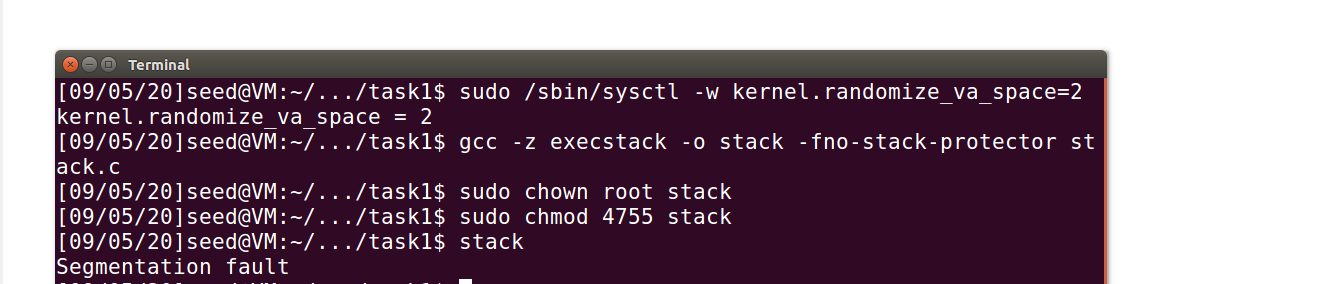


重新编译exploit并执行，再次执行stack，发现打开了root模式的dash，说明dash的反制措施已经被攻破了



Task4:

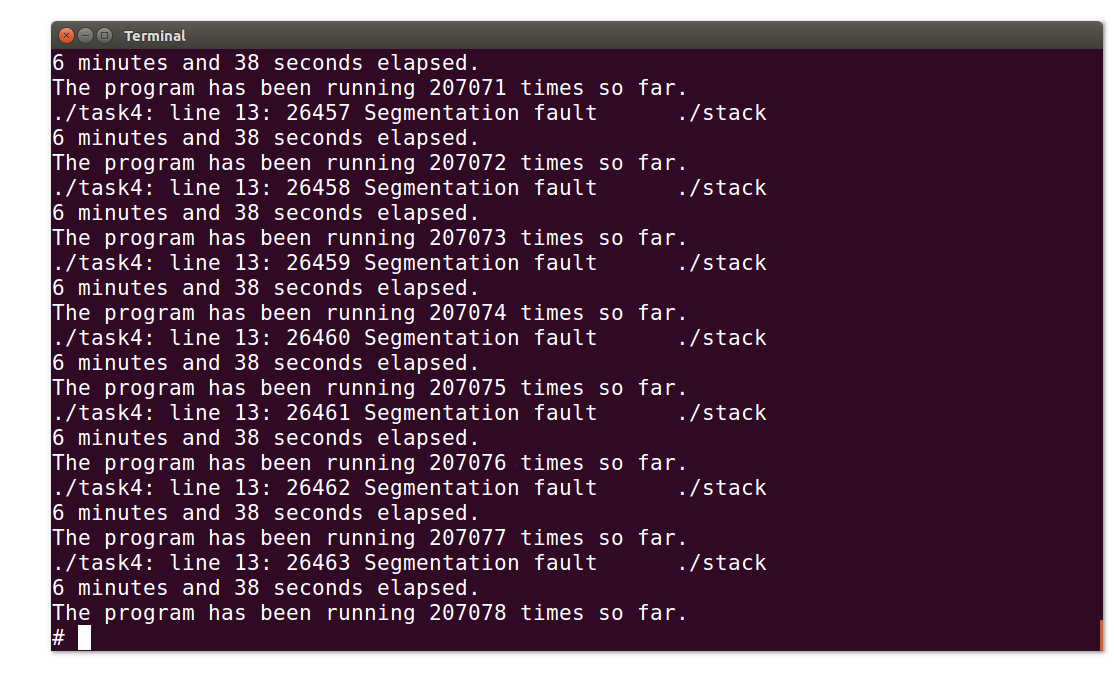
首先按照task2的步骤攻击，发现出现错误提示段错误，因为栈地址的随机化使exploit中的地址无效，无法生成正确的badfile



将暴力破解的代码写入task4



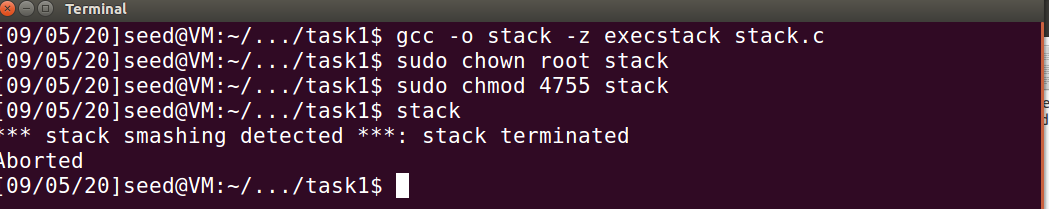
使用chmod更改task4的权限为777然后执行，结果如下



进入了root模式的shell，说明暴力攻破了地址随机保护

Task5:

如图所示，重新编译stack，不使用-fno-stack-protector，将可执行程序改为root的setuid程序后执行，发现报错：stack smashing detected，即检测到栈溢出故终止了程序，说明gcc的stackguard起到了作用



Task6:

如图所示，重新编译stack，使用-z noexecstack，将可执行程序改为root的setuid程序后执行，发现报错：段错误，无法打开shell。Non-execstack-proctection使栈不可执行，因此即使在栈中放入恶意程序也无法执行。

