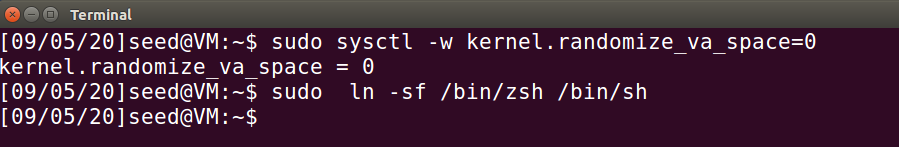
**Buffer Overﬂow Vulnerability Lab**

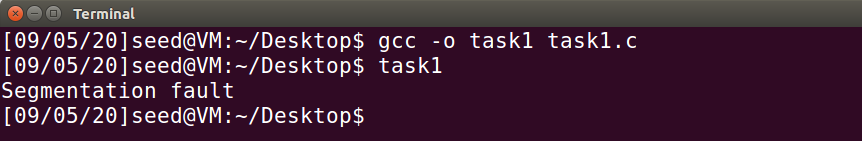
**57118132 吴嘉琪**

**Turning Off Countermeasures**

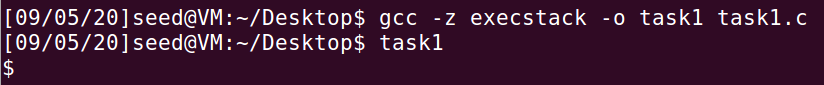


**Task 1: Running Shellcode**

使用 gcc -o task1 task1.c 编译程序并运行，发现系统会提示段错误

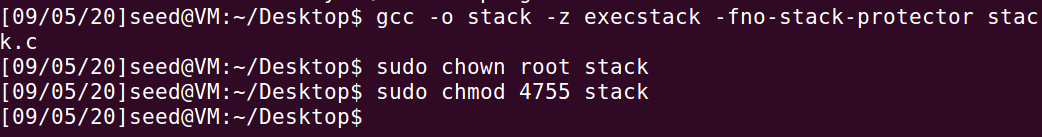


使用 gcc -z execstack -o task1 task1.c 编译程序并执行，发现系统进入到了shell中。

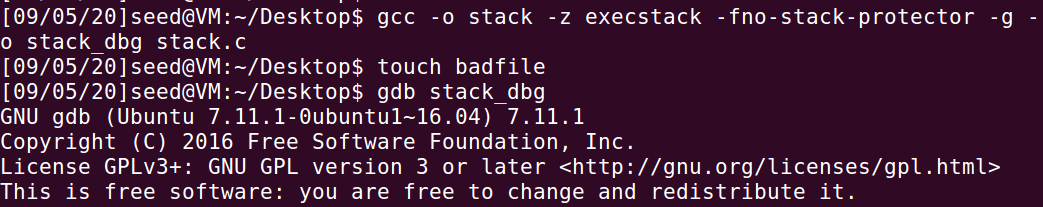


**Task2: ExploitingtheVulnerability**

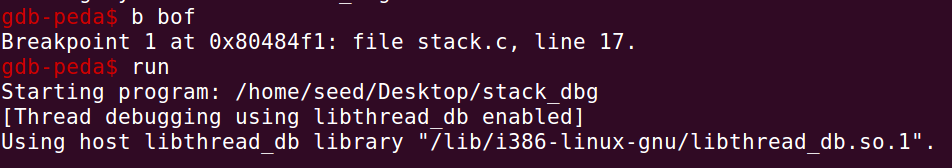
将代码复制到 stack.c 中，使用 gcc 编译，buf\_size 不变，使其成为 root 所属得 setuid 程序



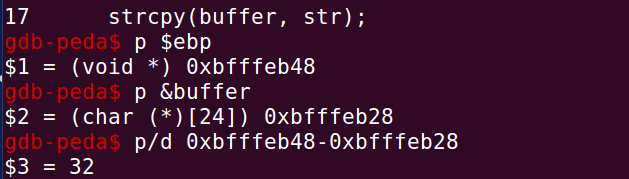
使用 gdb 对 stack 进行调试



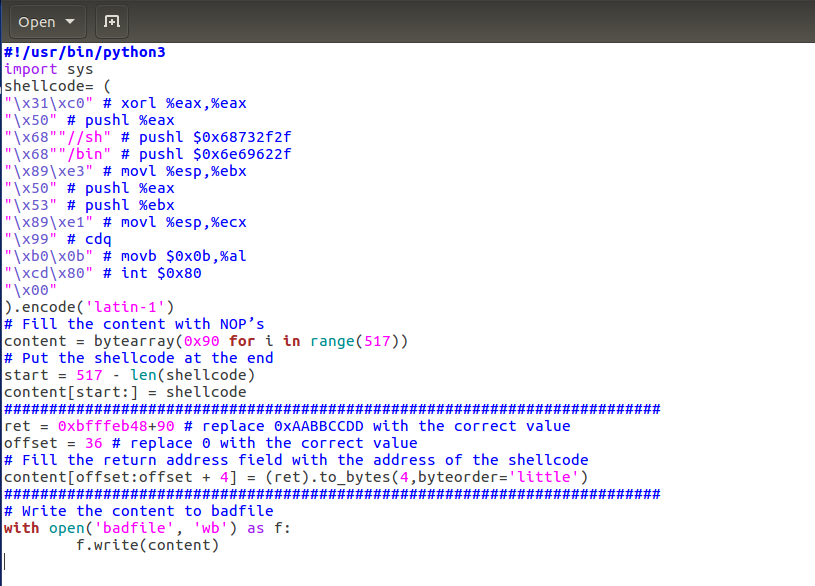
在函数位置确定一个断点并运行



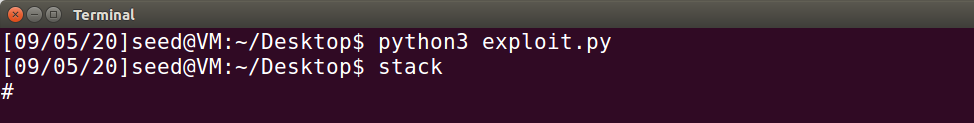
找到 ebp 寄存器和 buffer 的值，并计算出二者之间的差值：32



因此，return address 和 buffer 起点的差值为 36，在此基础上更改 exploit.py(这里选择了 python)，更改后代码如下，只需要更改 ret 和 offset 的值即可

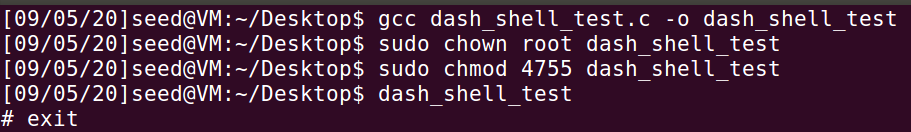


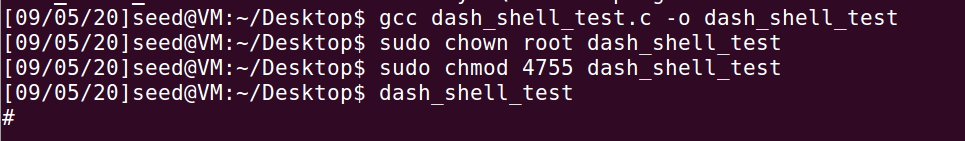
执行代码，运行 stack，发现获取到了 root 权限



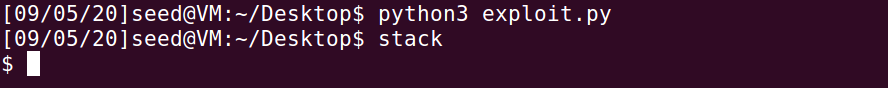
**Task 3: Defeating dash’s Countermeasure**

复制代码，在注释 setuid 的条件下编译并使之成为 root 拥有的 setuid 程序，运行，发现没 有提权，系统将权限降低，取消注释后再次重复编译等操作，发现成功获得 root 权限。

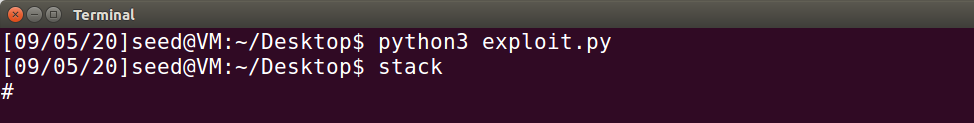




未更改 shellcode 之前，栈溢出攻击失败，因为系统自动判断 euid 和 ruid 不相等，进而降权， 导致无法获得权限

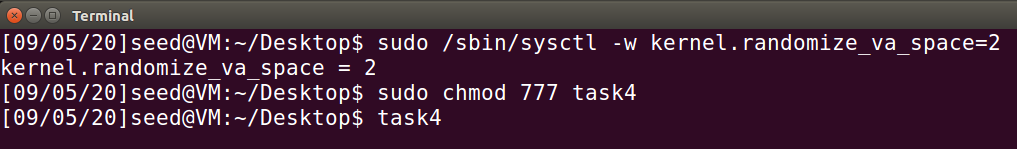


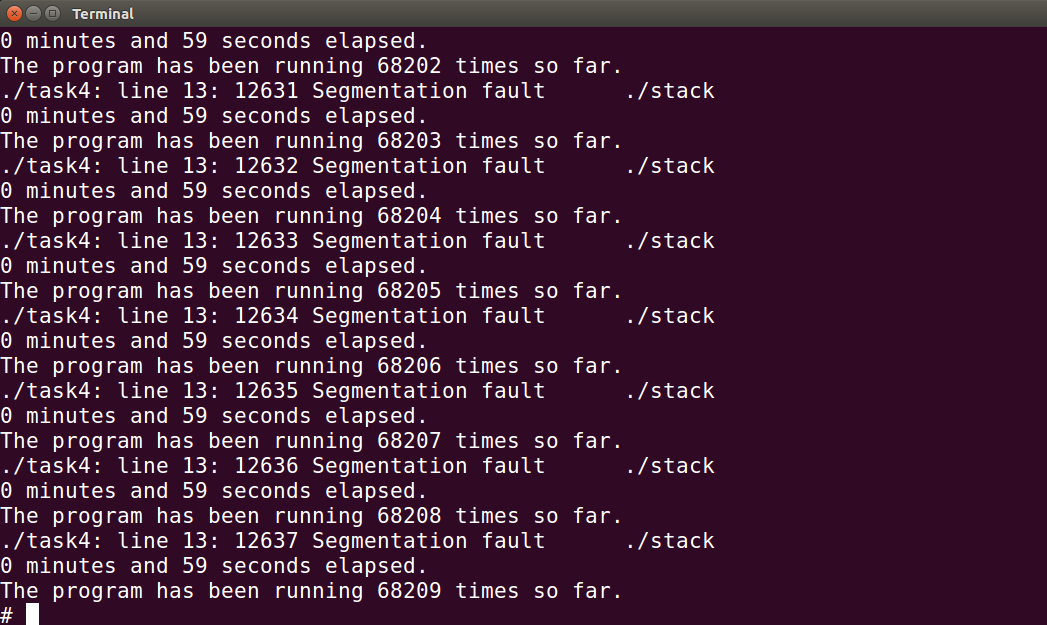
更改 shellcode 后运行程序并进行栈溢出攻击，发现可以获取权限，是因为在调用 execve 之 前 setuid 使 ruid=0（root），使系统误判为 root 用户的使用，所以权限不会降级，获取到 root 权限



**Task 4: Defeating Address Randomization**

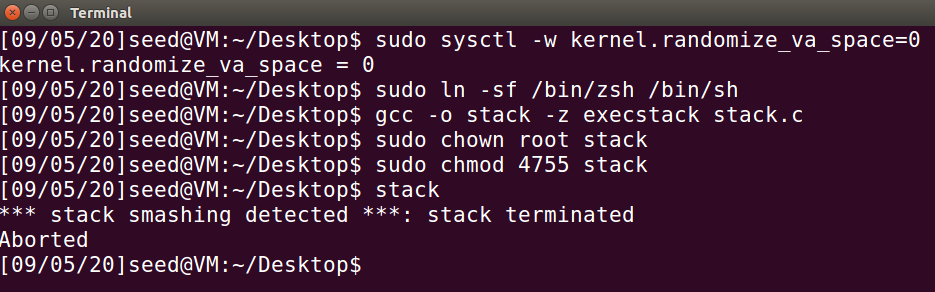
使用 sudo /sbin/sysctl -w kernel.randomize\_va\_space=2 命令启用地址随机化，复制代码并运 行，进行暴力破解来获取 shell，如下为运行成功的截图，共运行 68290 万次





**Task 5: Turn on the StackGuard Protection**

开启栈保护（去除-fno-stack-protector 自动启用）的条件下重新编译运行，会发现系统会检 测到栈被破坏，程序异常退出



**Task 6: Turn on the Non-executable Stack Protection**

启用 Non-executable Stack，编译运行程序发现出现了段错误，由于启用了栈不可执行使得 shellcode 不能在栈上运行导致失败。

