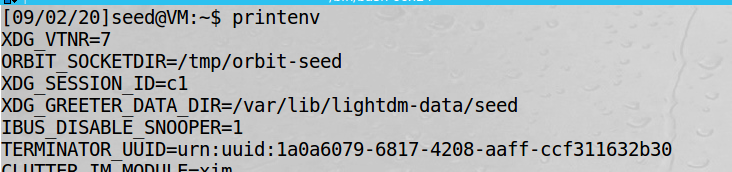
Lab1

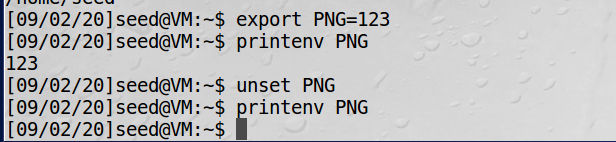
Task1:

• Use printenv or env command to print out the environment variables. If you are interested in some particular environment variables, such as PWD, you can use "printenv PWD" or "env | grep PWD".



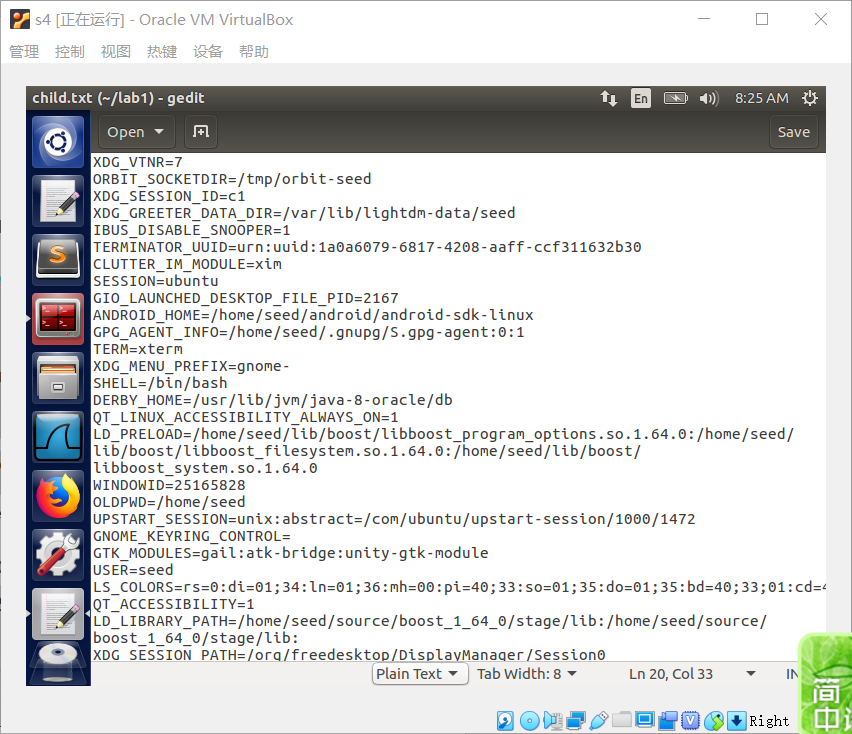


• Use export and unset to set or unset environment variables. It should be noted that these two commands are not seperate programs; they are two of the Bash’s internal commands (you will not be able to ﬁnd them outside of Bash).



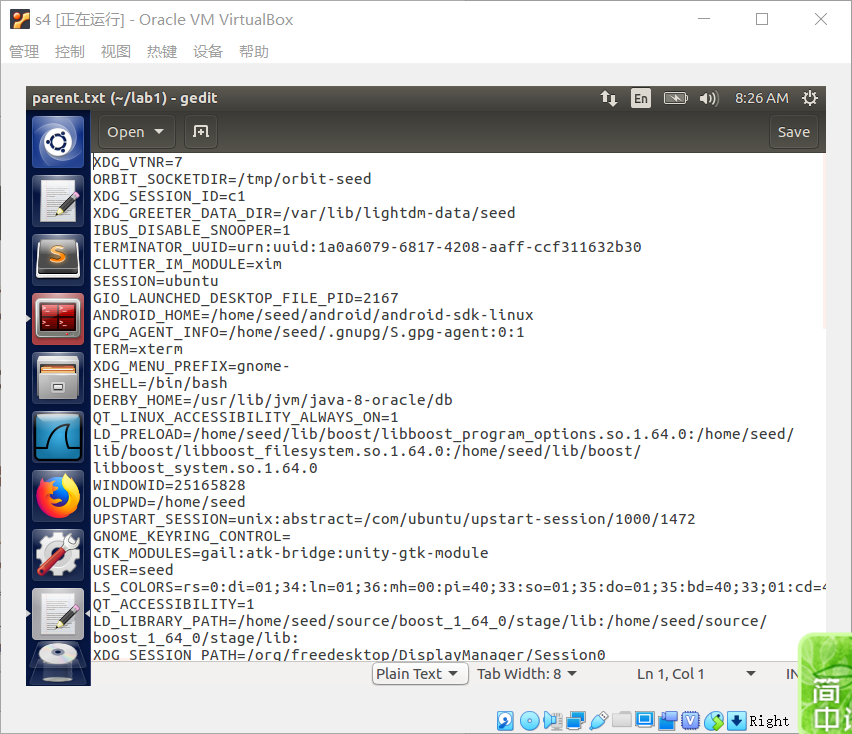
Task2:

Step1:



输出了一堆环境变量

Step2:



输出了一堆环境变量

Step3:



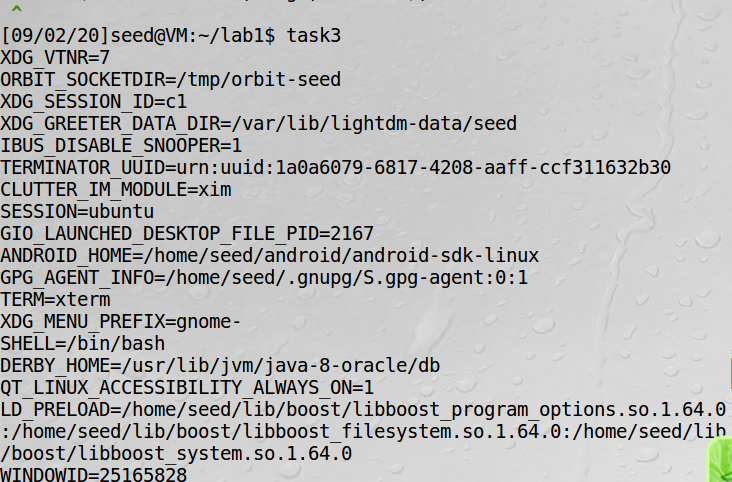
父进程和子进程的环境变量没有区别，子进程完全继承了父进程的环境变量

Task3:

Step1:

No output

Step2:

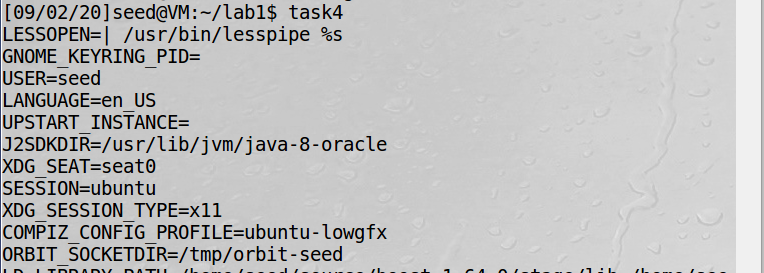


输出了这个进程的环境变量

Step3:

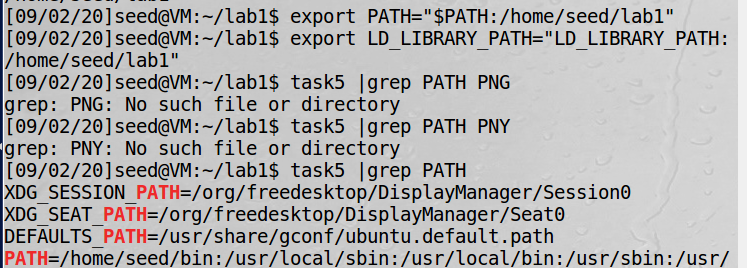
Execve()在执行命令时，会根据传入的参数改变它的环境变量

Task4:



环境变量被传递过来

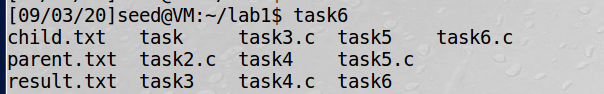
Task5:



Shell进程中设置的PATH与Set-UID进程的PATH并不一致，说明Set-UID并没有完全继承父进程的环境变量

Task6:

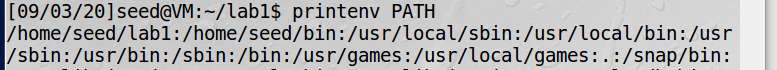
编译并执行程序：

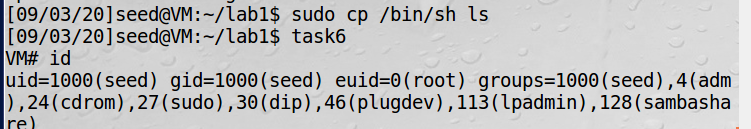


在PATH环境变量前面加上当前路径，并把/bin/sh拷贝到当前目录并改名为ls当成自己的ls，并对shell进行一定处理







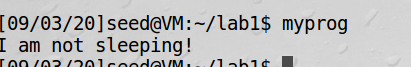


可以看到，再运行task6，执行的是当前目录的ls 而不是bin/ls；

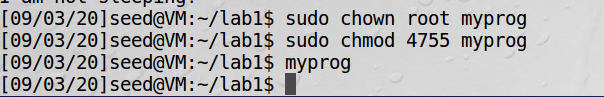
我们获得了一个权限为root的shell

Task7:

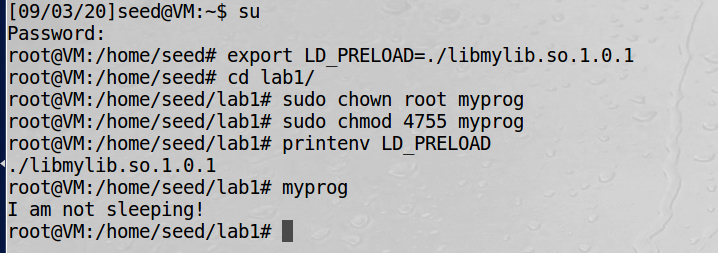
• Make myprog a regular program, and run it as a normal user.



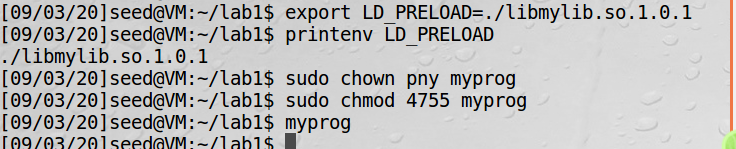
• Make myprog a Set-UID root program, and run it as a normal user.



• Make myprog a Set-UID root program, export the LD PRELOAD environment variable again in the root account and run it.



• Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), export the LD PRELOAD environment variable again in a different user’s account (not-root user) and run it.

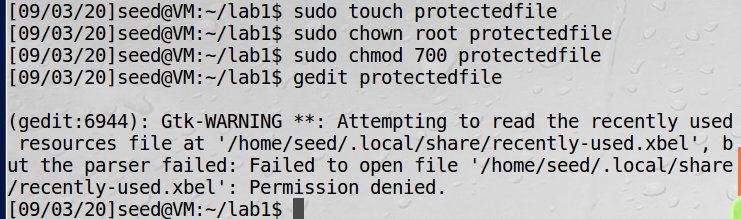


结论：只有在ruid和euid相同时，自己修改的LD\_PRELOAD环境变量才有效

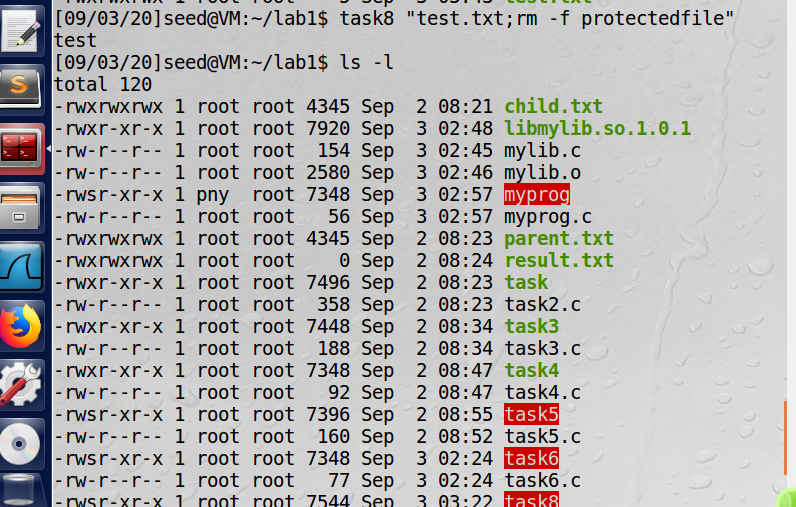
Task8:

创建owner为root且权限为700的protectedfile的文件

正常情况下没有权限修改这个文件

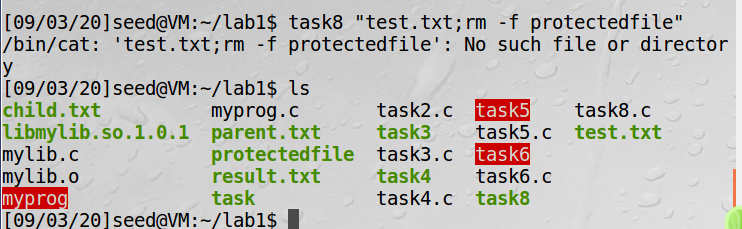


利用上面的程序删除文件



文件被删除

使用execve()后



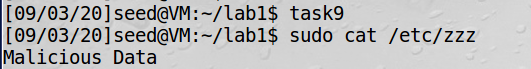
攻击不再奏效

结论：system()是新开一个shell去执行命令，可以通过分号把多条命令通过参数传递，从而进行攻击；execve()只是当成参数去执行命令，更加安全。

Task9:



普通用户无权修改



普通用户执行这个程序，zzz文件被修改

执行完特定任务后fd没有被及时关闭，特权还在，所以可以修改文件