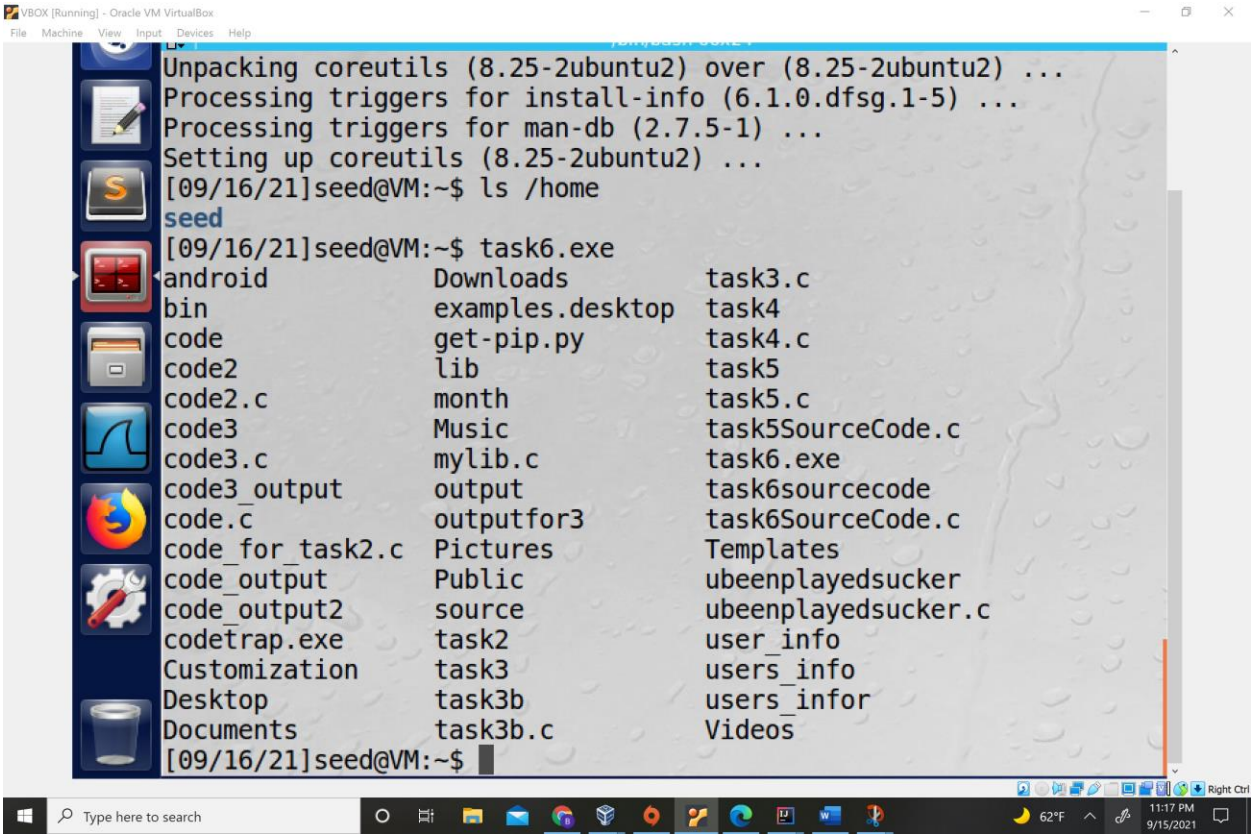
Capability Leaking Attacks Tasks 6 Through 9

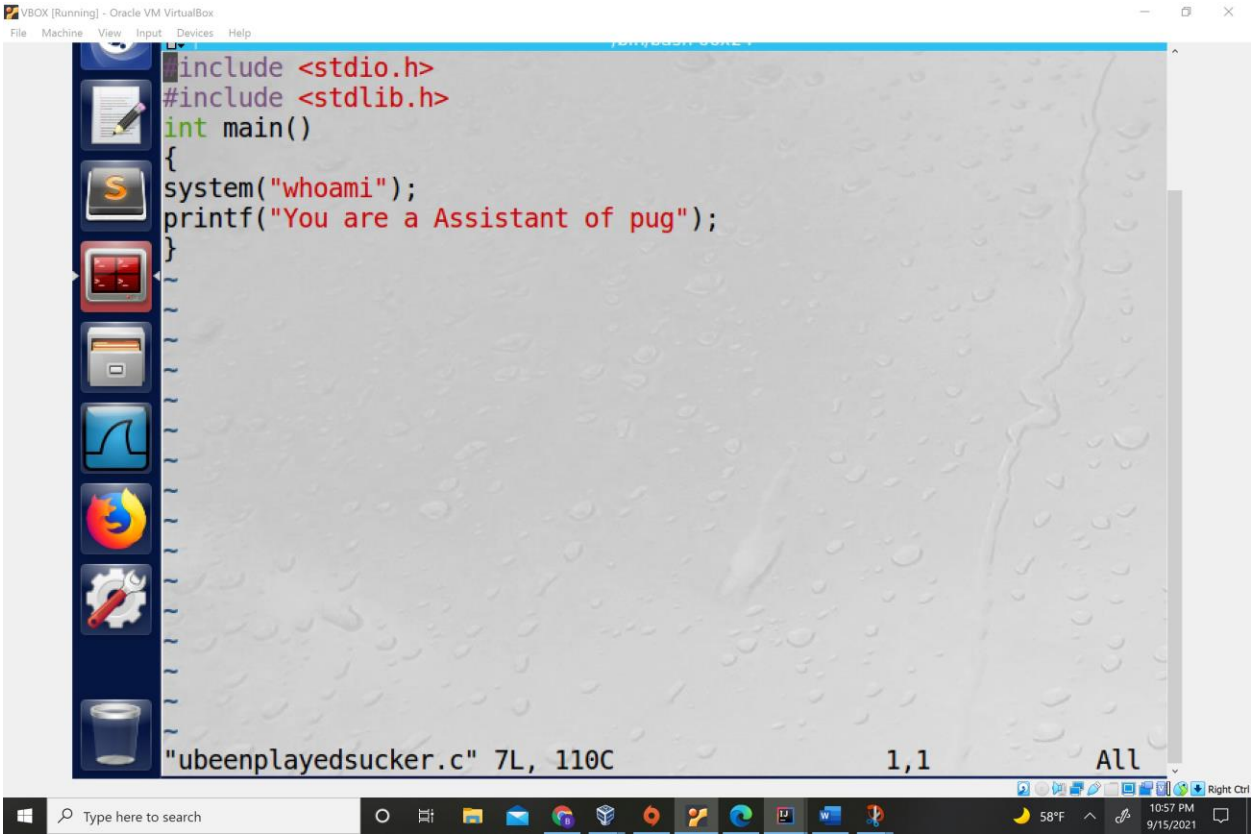
Task 6- Using System as an attack vector

This is the original source code which invokes the original ls command via a new shell.



This is the source code for the file ubeenplayedsucker.c which is compiled into the a file called ls. I got the program given to run my code with root privileges(evidenced from the output root from the who am I call) after I removed the original ls function(this later required reinstalling using command (sudo apt-get install --reinstall coreutils))

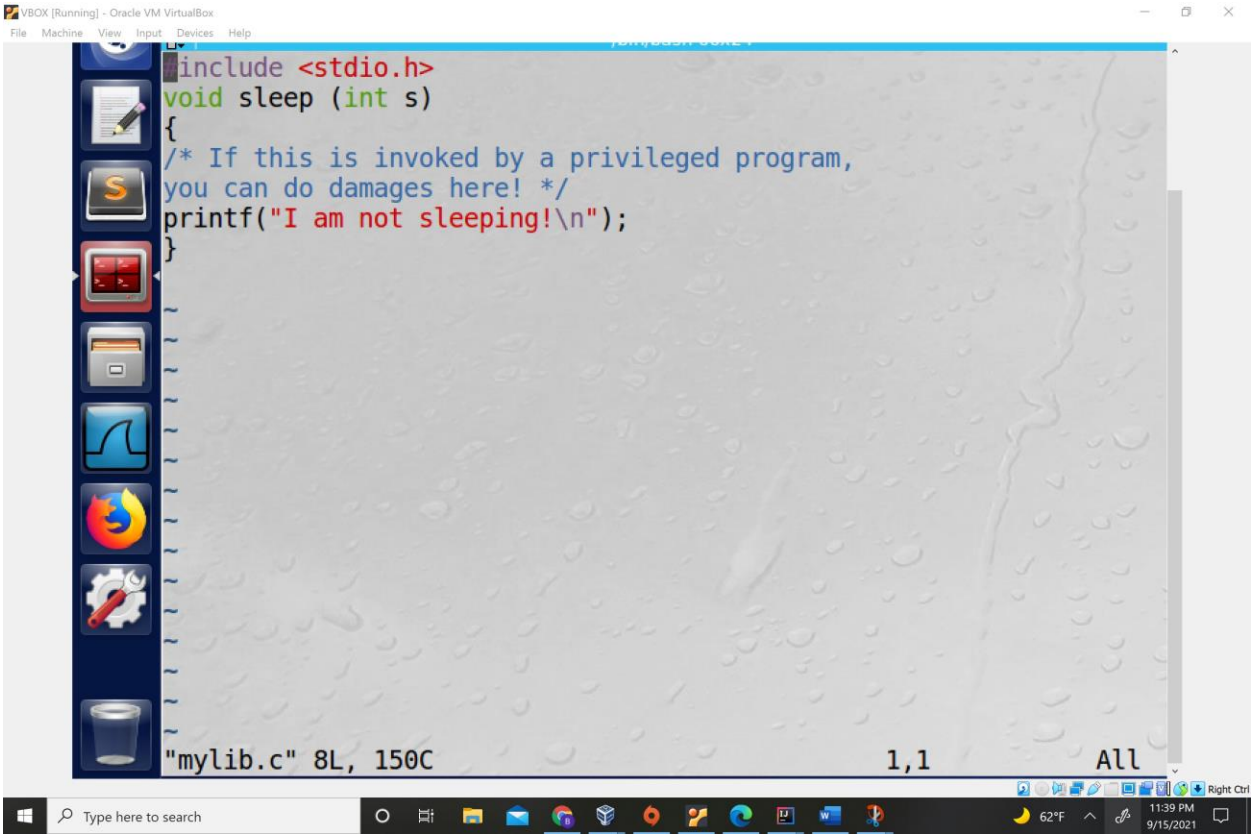
```
sudo apt install coreutils
[09/16/21]seed@VM:~$ sudo rm /bin/ls
rm: cannot remove '/bin/ls': No such file or directory
[09/16/21]seed@VM:~$ /bim
bash: /bim: No such file or directory
[09/16/21]seed@VM:~$ /bin
bash: /bin: Is a directory
[09/16/21]seed@VM:~$ gcc -Wall ubeenplayedsucker.c -o ls
[09/16/21]seed@VM:~$ ls
seed
You are a Assistant of pug[09/16/21]seed@VM:~$ gcc -Wall ubeenplay
edsucker.c -o ls     ls
seed
You are a Assistant of pug[09/16/21]seed@VM:~$ gcc -Wall ubeenplay
edsucker.c -o ls     task6
No command 'task6' found, did you mean:
 Command 'task0' from package 'pvm-examples' (universe)
 Command 'taskd' from package 'taskd' (universe)
 Command 'task' from package 'taskwarrior' (universe)
 Command 'task1' from package 'pvm-examples' (universe)
task6: command not found
[09/16/21]seed@VM:~$ task6.exe
root
You are a Assistant of pug[09/16/21]seed@VM:~$
```



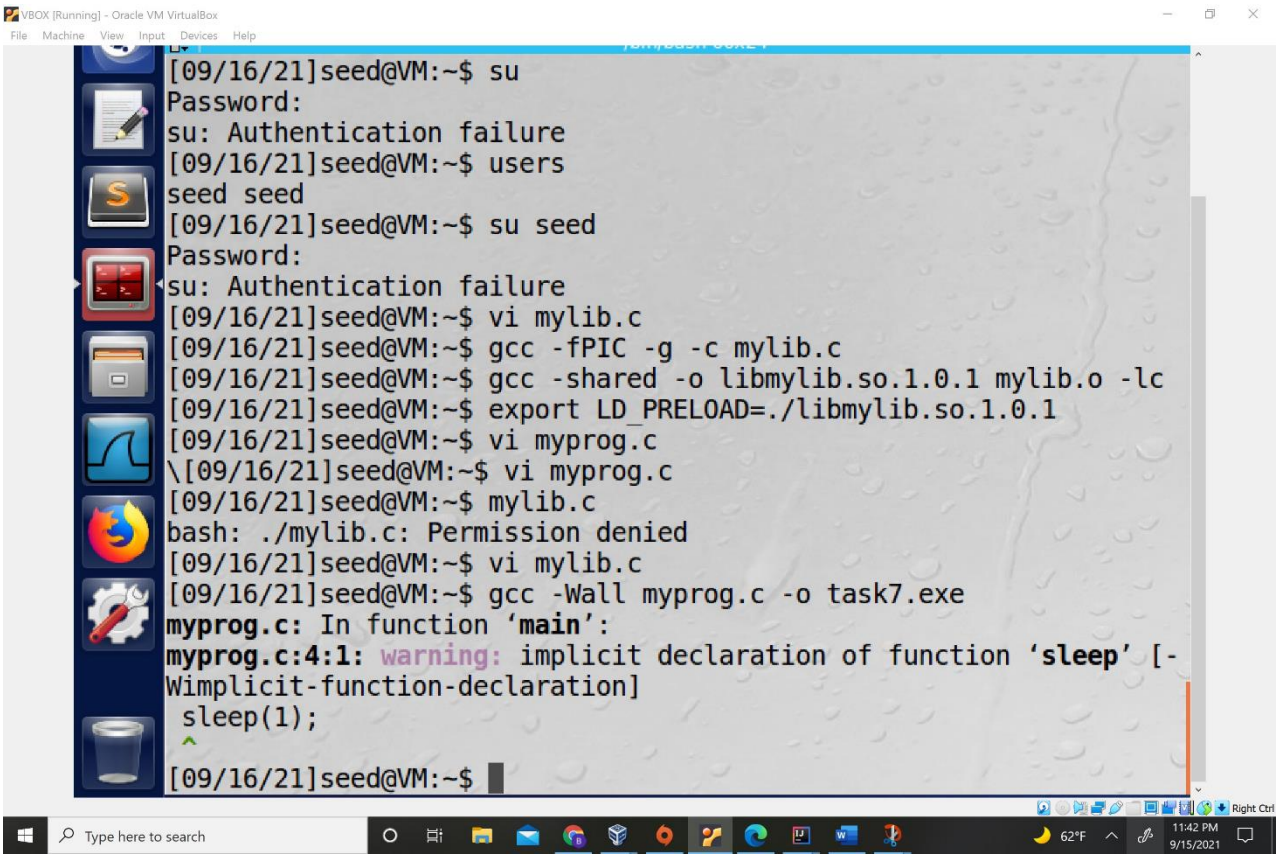
Task 7

Source Code for myprog.c

Source code for Mylib.c

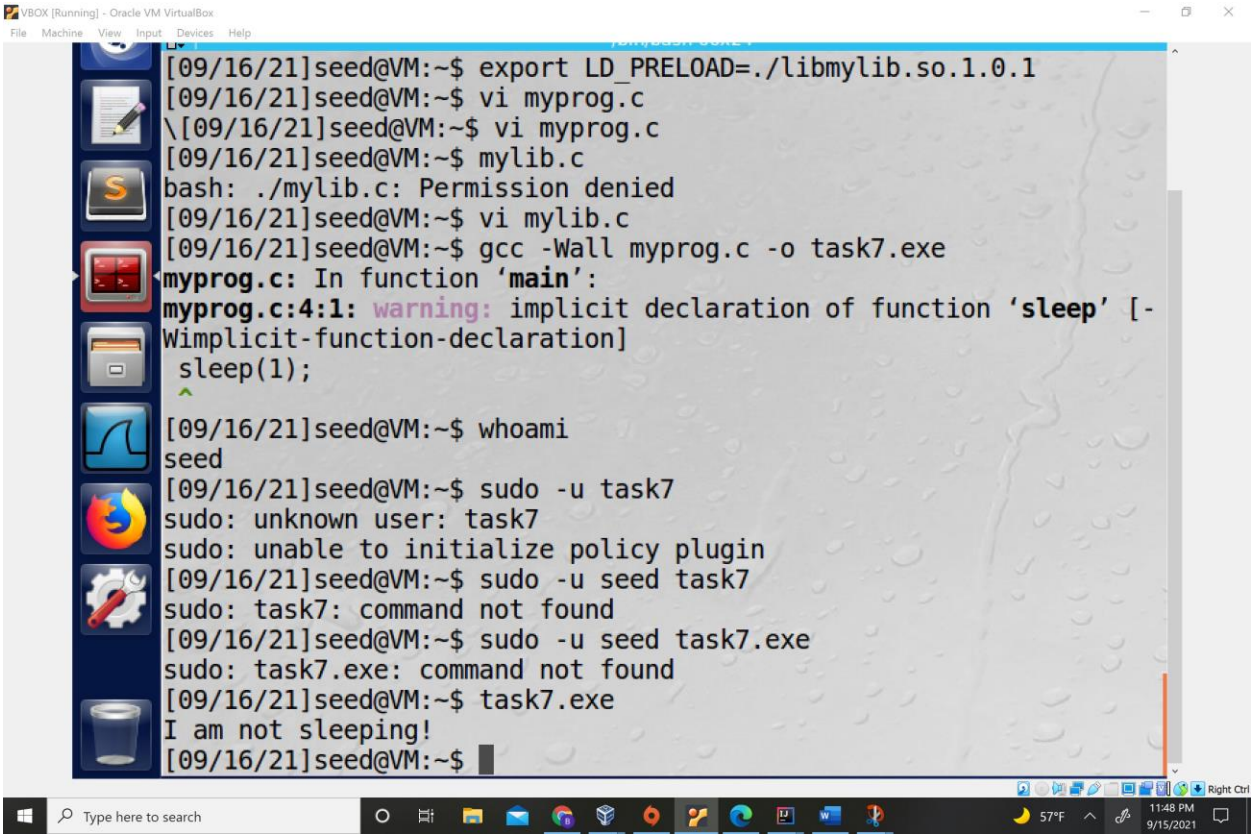


Compiling program and compiling library and exporting new LD_PRELOAD

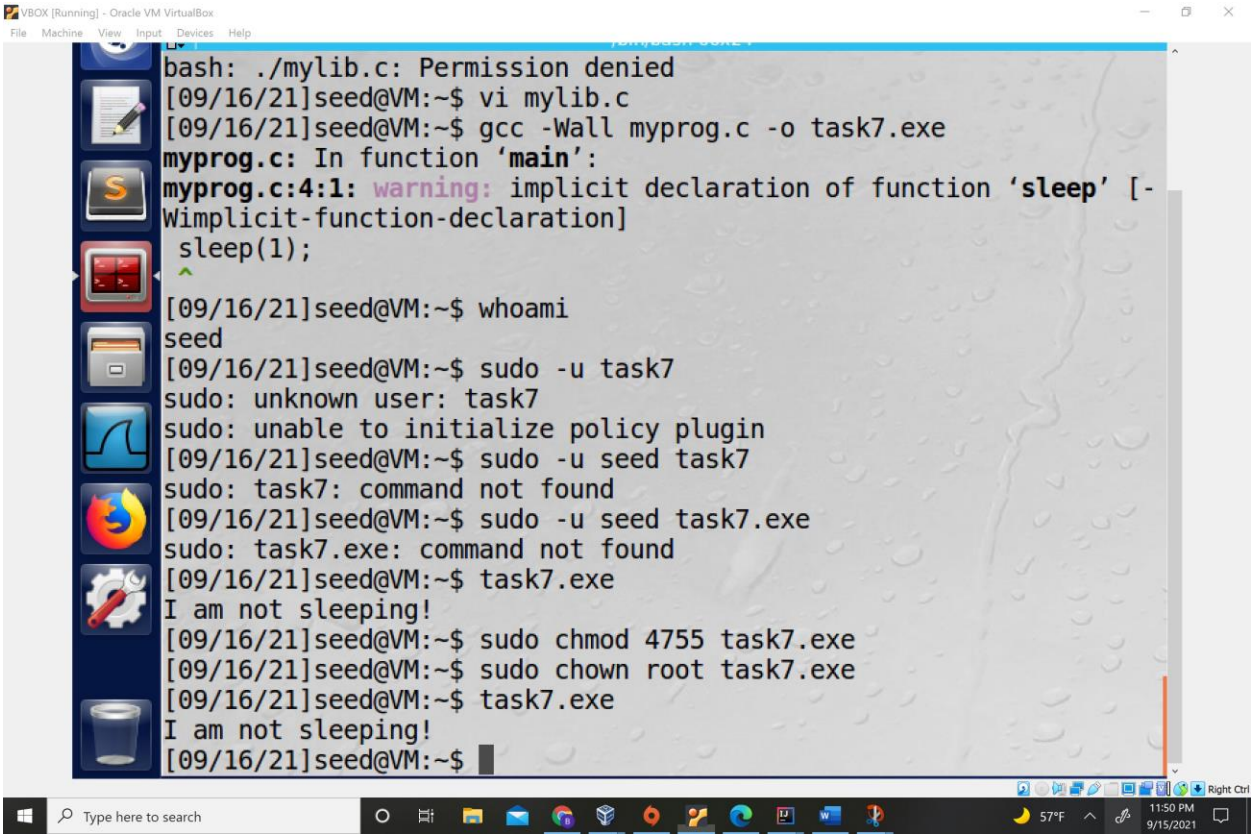


Trials

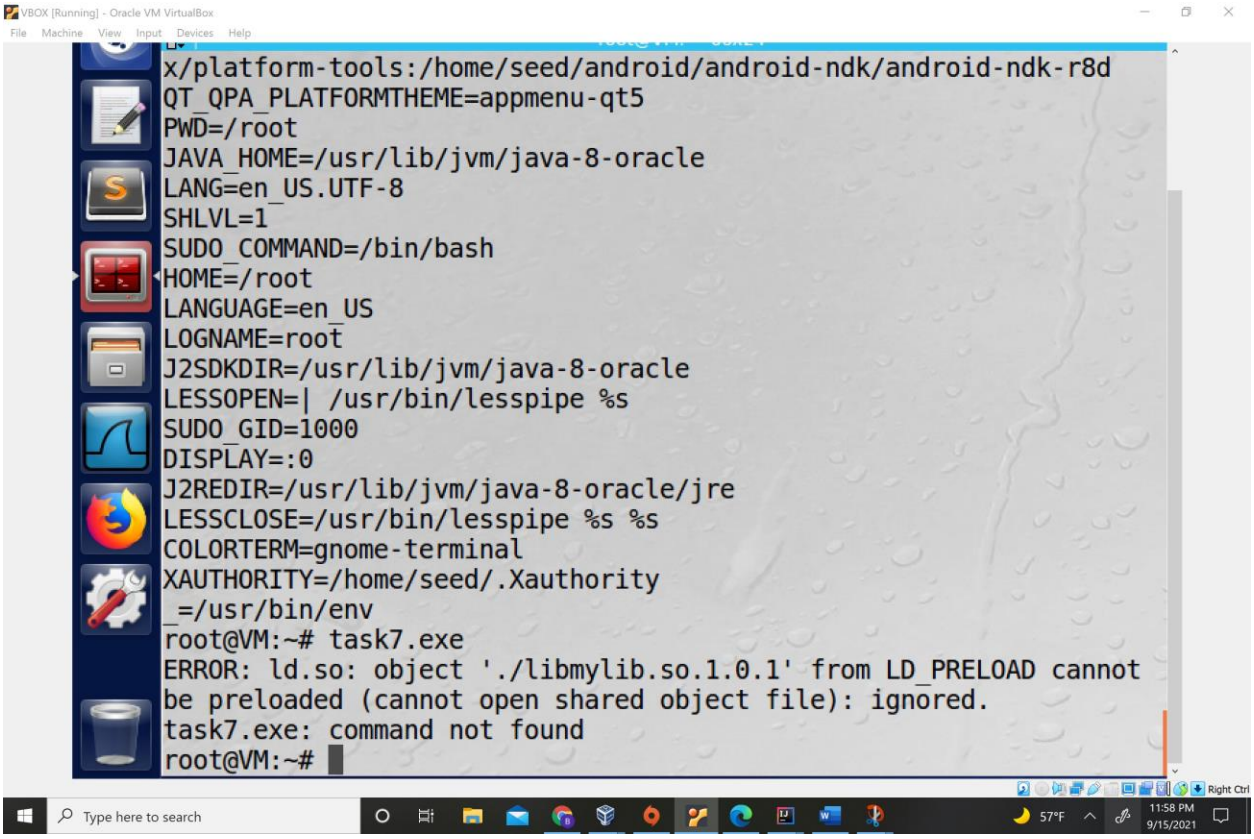
Make myprog a regular program, and run it as a normal user.



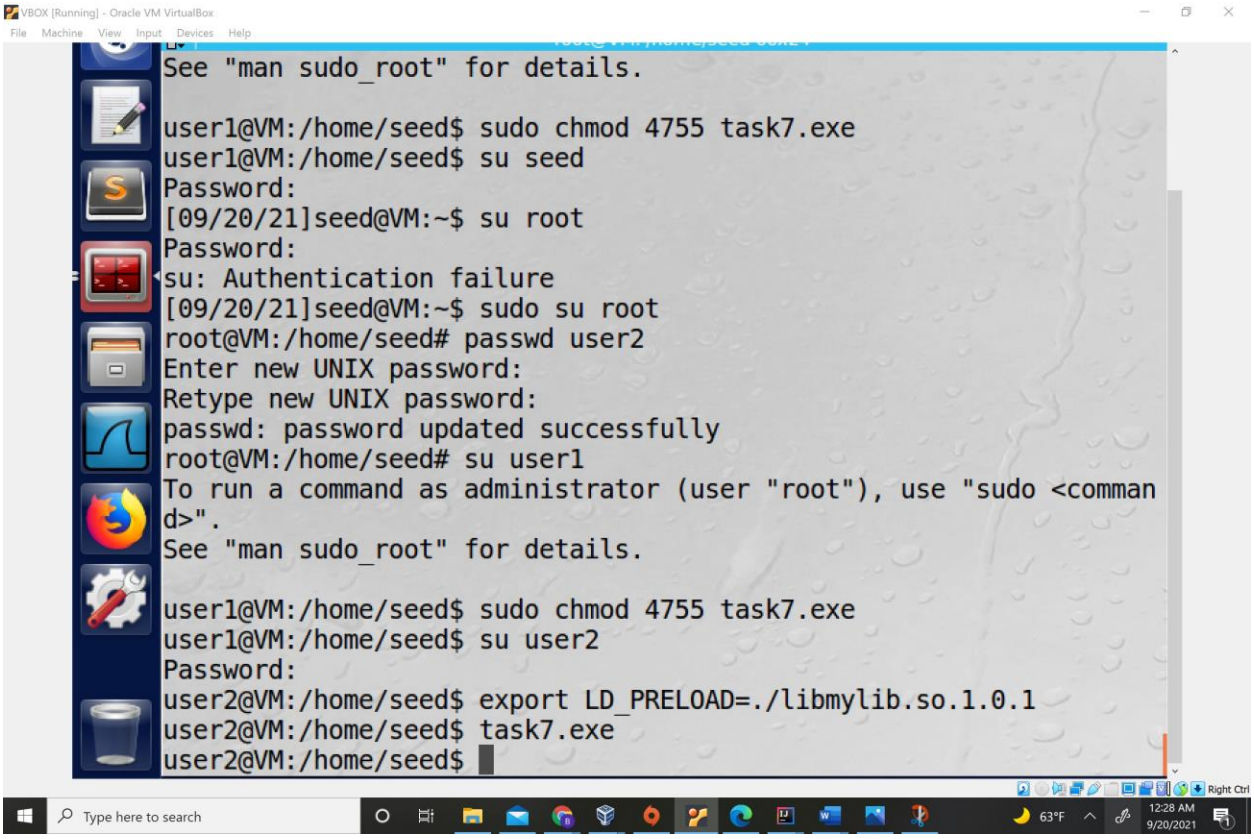
Make myprog a Set-UID root program, and run it as a normal user.



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.

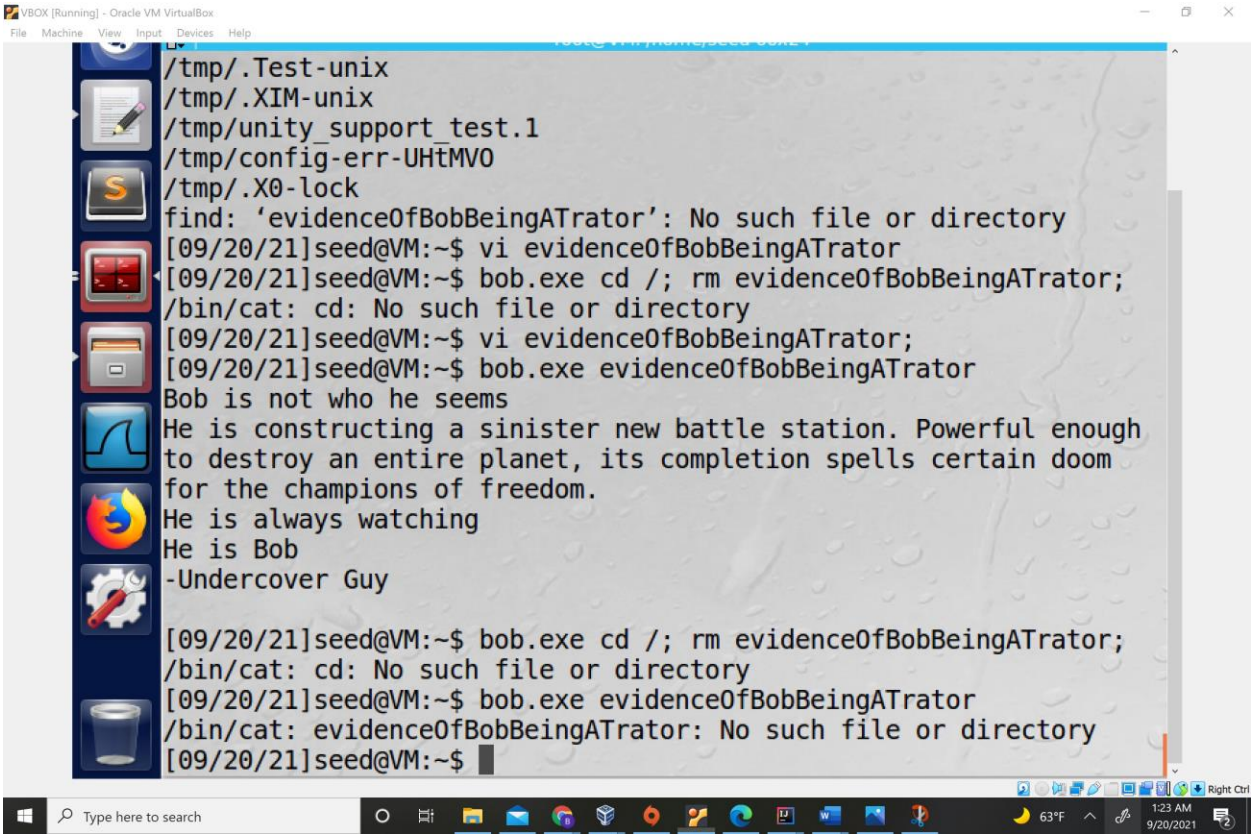


Task 8

Step 1

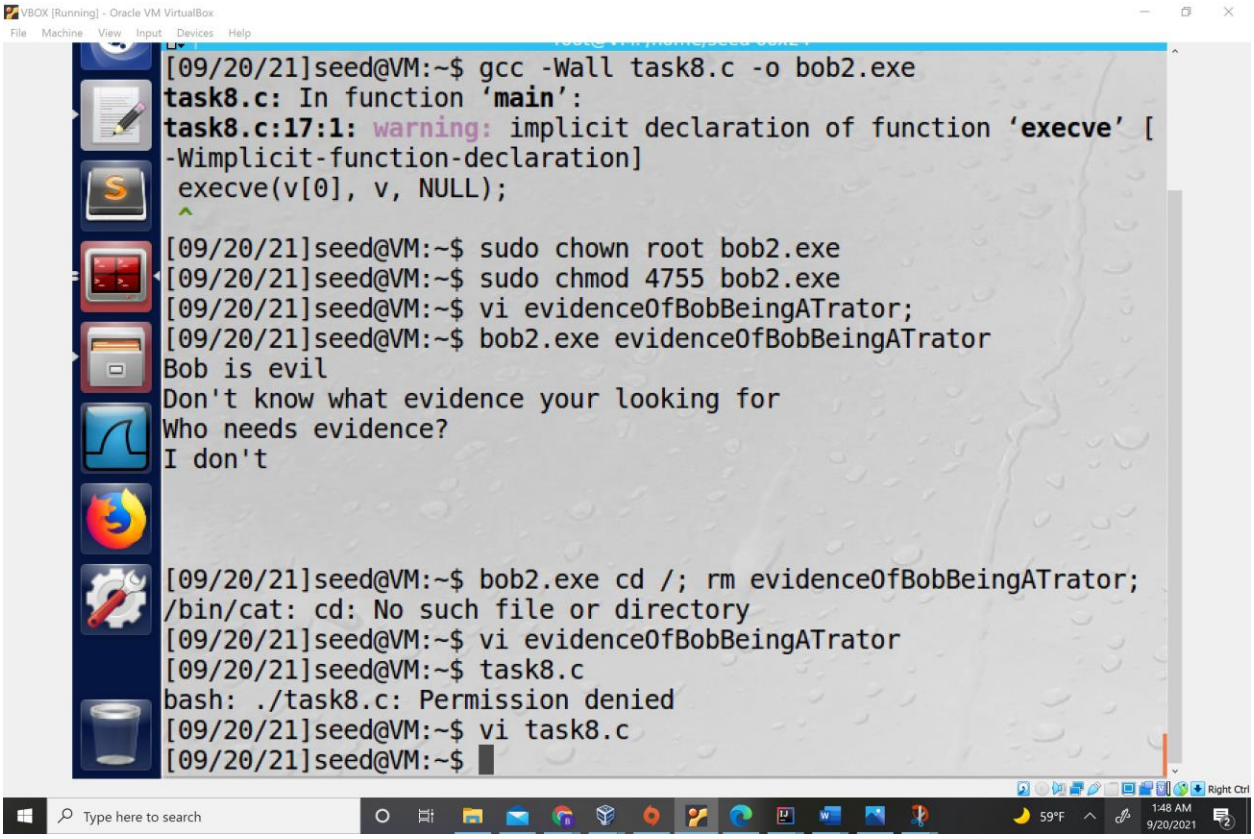
Compile the above program, make it a root-owned Set-UID program. The program will use system() to invoke the command. If you were Bob, can you compromise the integrity of the system? For example, can you remove a file that is not writable to you?

I made a file evidenceOfBobBeingATrator(I know Traitor is spelt wrong, honest mistake) went about trying to delete it. I found you can take advantage of the new system call and SetUID privileges by just passing commands into the arguments instead of a file.



Step 2

Comment out the system(command) statement, and uncomment the execve() statement; the program will use execve() to invoke the command. Compile the program, and make it a root-owned Set-UID. Do your attacks in Step 1 still work? Please describe and explain your observations.



Uhhhhhhhh, yeah it still worked.

Task 9

```
 if (fork()) { /* In the parent process */
     ^
task9.c:21:2: warning: implicit declaration of function ‘close’ [-
Wimplicit-function-declaration]
  close (fd);
  ^
task9.c:24:1: warning: implicit declaration of function ‘write’ [-
Wimplicit-function-declaration]
 write (fd, "Malicious Data\n", 15);
 ^
[09/20/21]seed@VM:~$ sudo chown root ex9.exe
[09/20/21]seed@VM:~$ sudo chmod 4755 ex9.exe
[09/20/21]seed@VM:~$ ex9.exe
[09/20/21]seed@VM:~$ su root
Password:
su: Authentication failure
[09/20/21]seed@VM:~$ sudo su root
root@VM:/home/seed# cd
root@VM:~# vi zzz
root@VM:~# ls
zzz
root@VM:~# cd /etc
root@VM:/etc# vi zzz
root@VM:/etc#
```

```
Cool stuff
Malicious Data
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"zzz" 2L, 26C                                   1,1           All
```