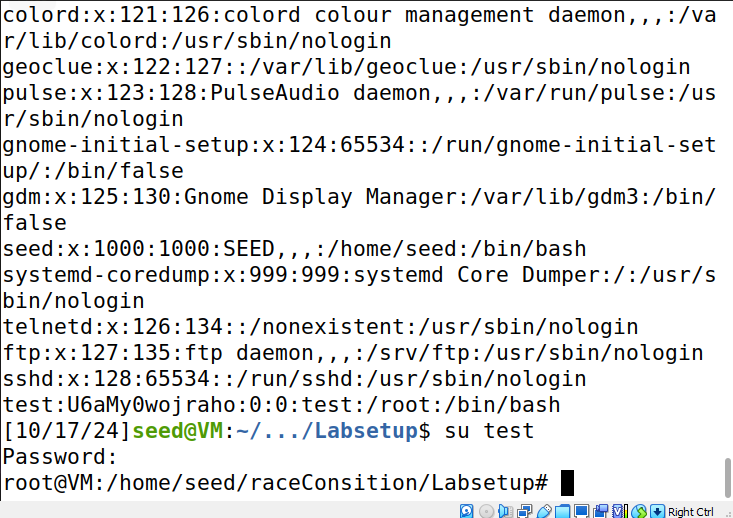
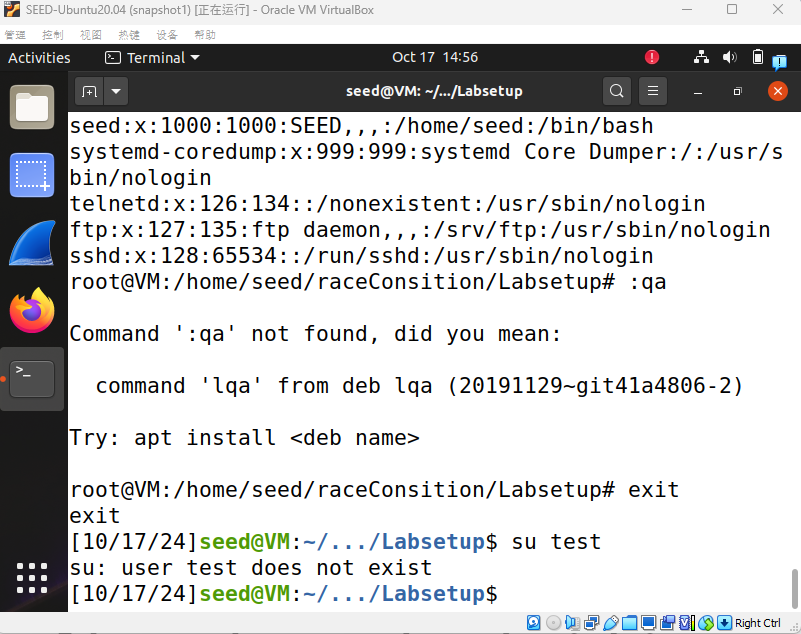
Name: Donghao Li

SU email: [dli106@syr.edu](mailto:dli106@syr.edu)

**Task 1: Choosing Our Target**



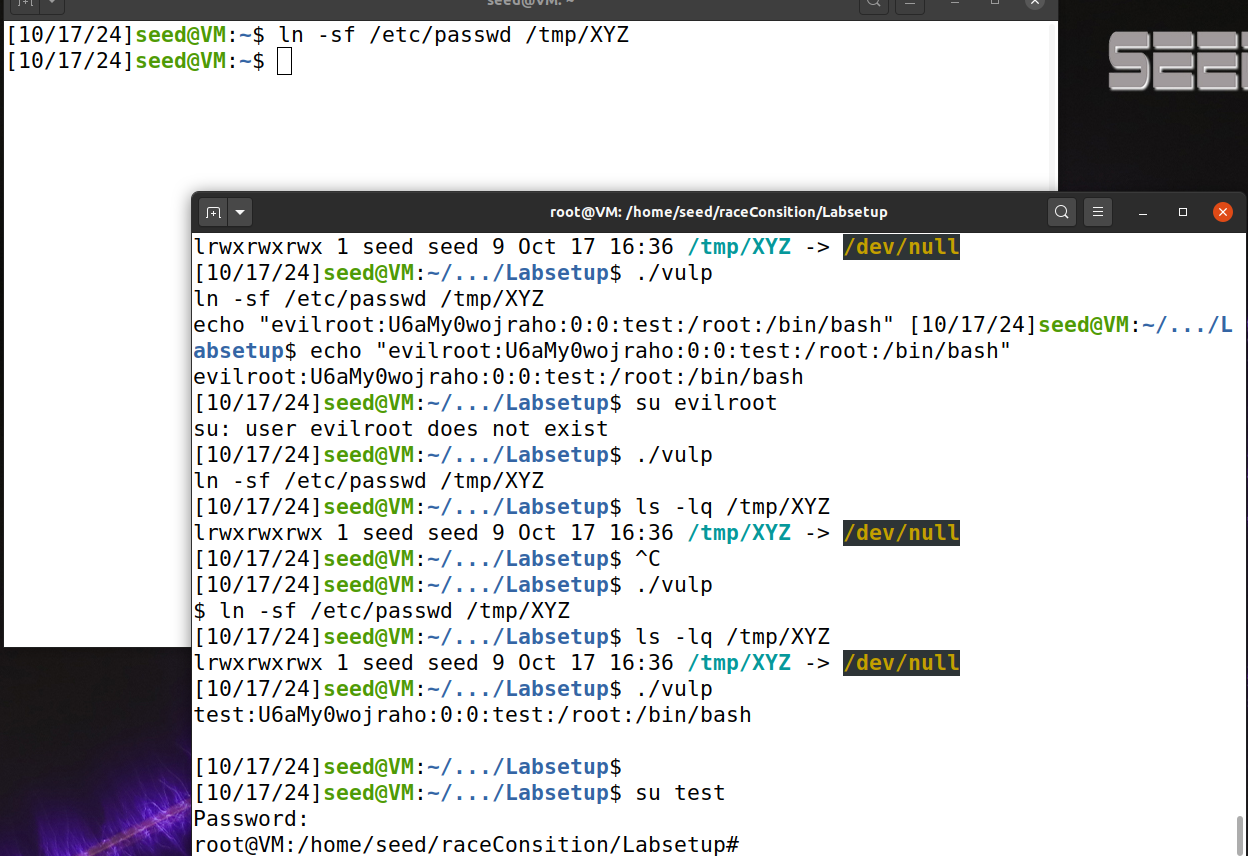
As you can see I first change to the root mode and add test into passwd file, then go back to the seed mode to see if the magic function in test works.



Then i remove test in passed as the instruction says.

**Task 2: Launching the Race Condition Attack**

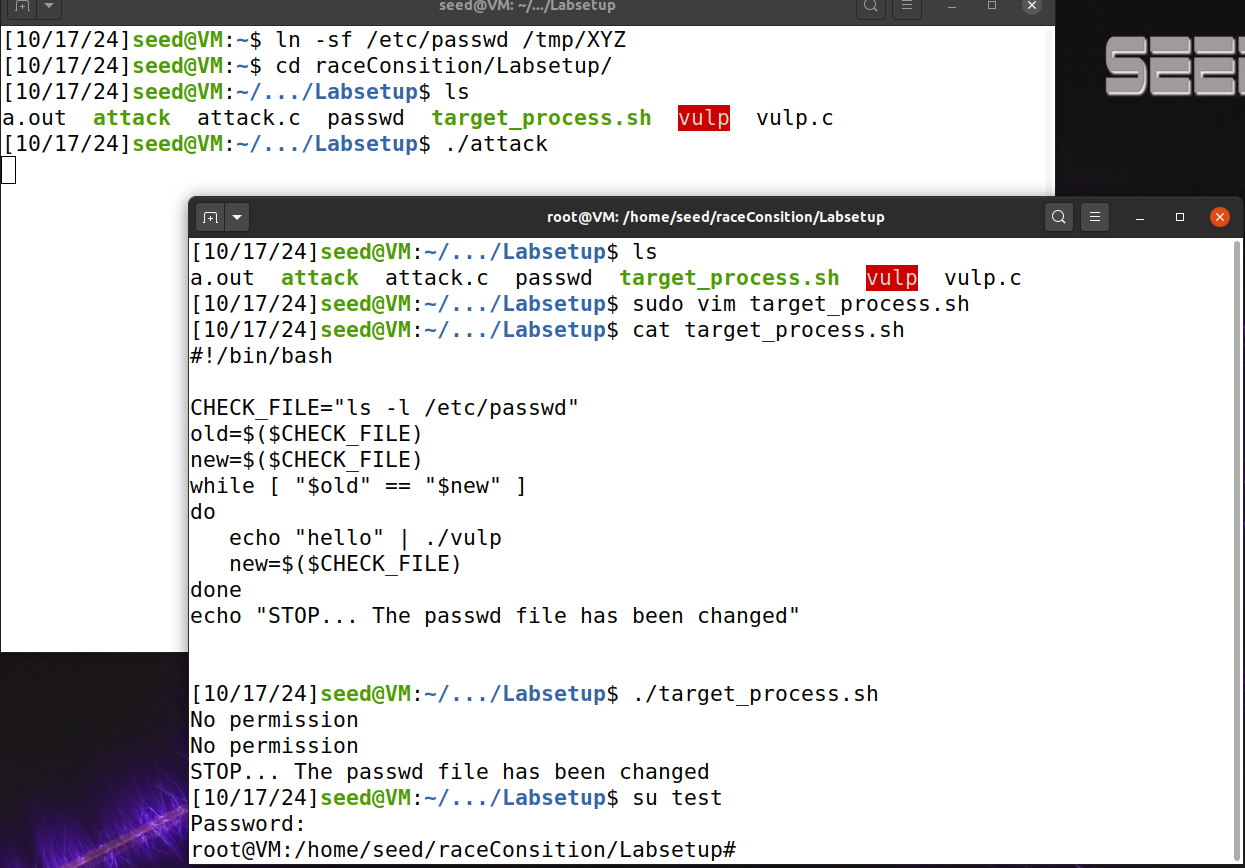
**Task 2.A: Simulating a Slow Machine**



I run the vulp first and input the information i need, while the sleep time is not finish up, i link the /tmp/XYZ file with the /etc/passwd file to make the command i imputed in vulp append in the /etc/passwd and you can see i add test and it can be run after vulp is finished.

**Task 2.B: The Real Attack**

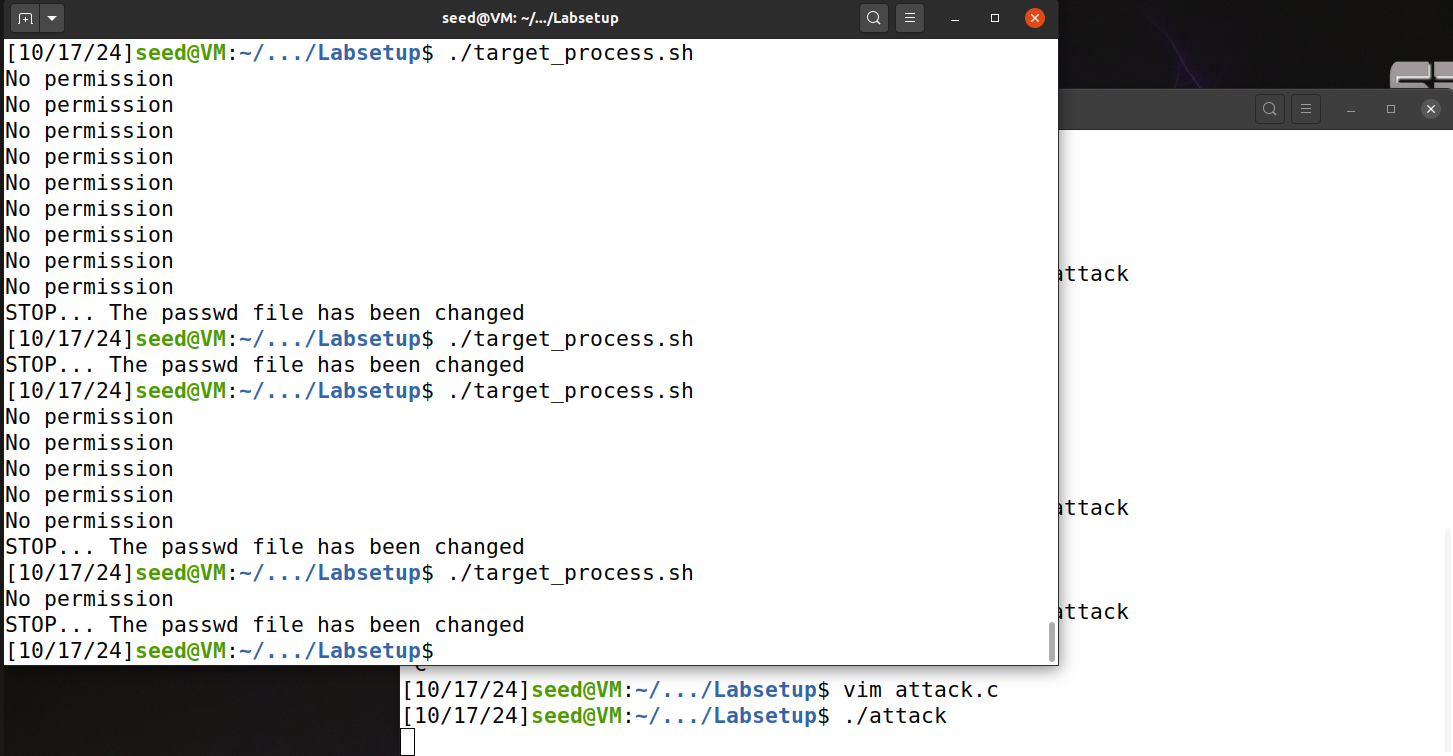
****



The Figure 1 shows how my attack.c looks like basically it is a infinite loop that kept unlink /tmp/XYZ link it to /dev/null usleep 10 then unlink and link /tmp/XYZ to /etc/passwd. in the target\_process.sh i kept compare the new and old password if it changed, we need to stop the loop. This process basically make the machine believe that /tmp/XYZ is currently link with /dev/null which is not dangerous, but when the machine try to input something into XYZ we change the link to /etc/passwd which will append the malicious code into the file we are not permeated to edit. This process make take several terms to success so we need to keep the attack loop infinitely.

**Task 2.C: An Improved Attack Method**

****



By infinitely looping the rename part, This allows us to change what /tmp/XYZ points to without introducing any race condition.

**Task 3: Countermeasures**

**Task 3.A: Applying the Principle of Least Privilege**





I use seteuid system call to disable the root privilege this causes the ./target\_process to become endless no permission. As you can see I got a lot of open failed while the race condition attack success because of the prevent. By using seteuid system call, we hardly prevent the use of root privilege which prevent the malicious code running with a root privilege.

**Task 3.B: Using Ubuntu’s Built-in Scheme**

****

I also got a lot of open failed when race condition attack success. from our attack, the /tmp/XYZ is created by the attacker which has SEED privilege, but in this prevention, it prevent any /tmp file created by user other than root privilege, this will efficiently prevent attack with race condition.

**limitation:**

This kind of prevention can only prevent attack using write-in function like /tmp i do not know if other functions with race conditions will be prevented by the countermeasure.