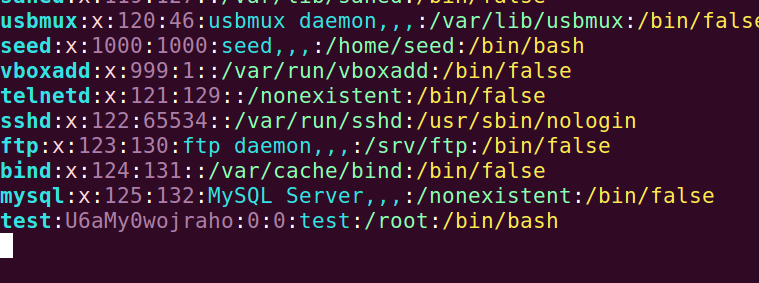
Lab 6

Race Condition Vulnerability Lab

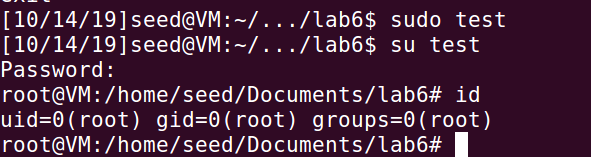
By Marco

**Task1: Targeting /etc/passwd**

Step 1: I modied the /etc/passwd file. Add the line on the bottom like following:

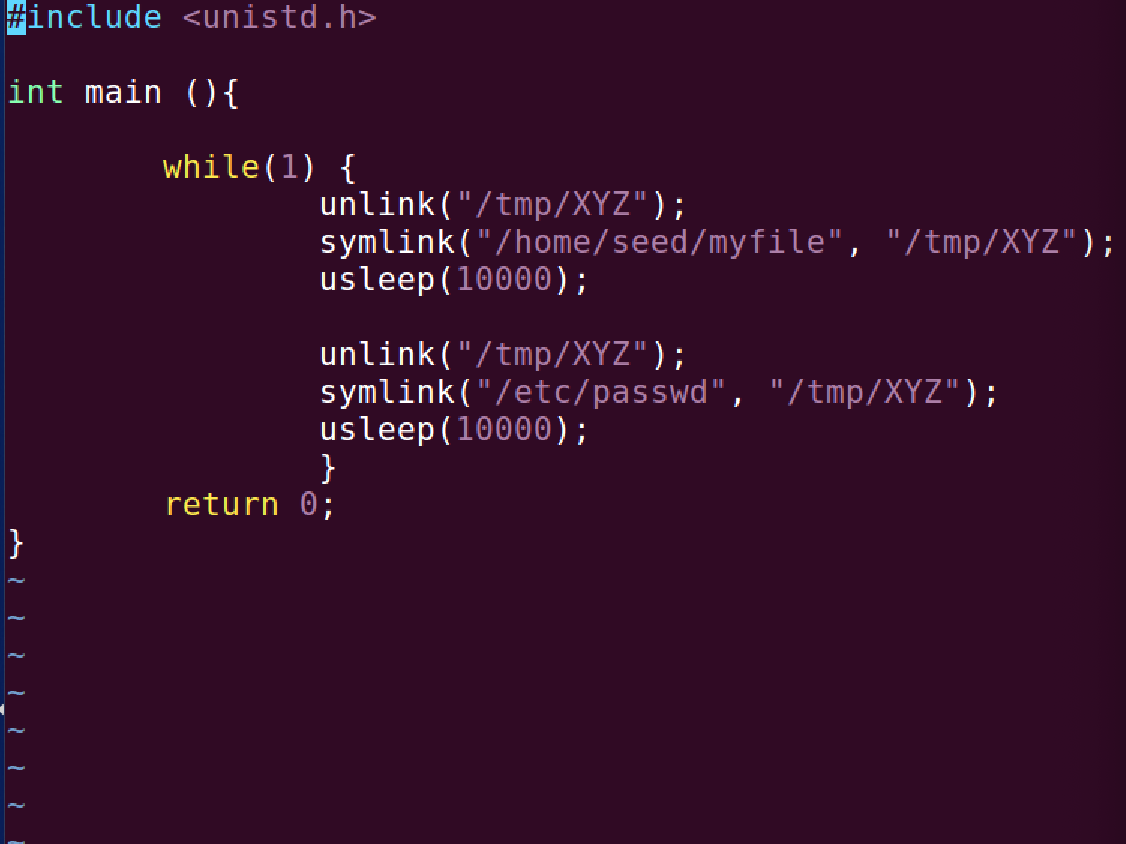


Step 2: We can get root by logging test account.

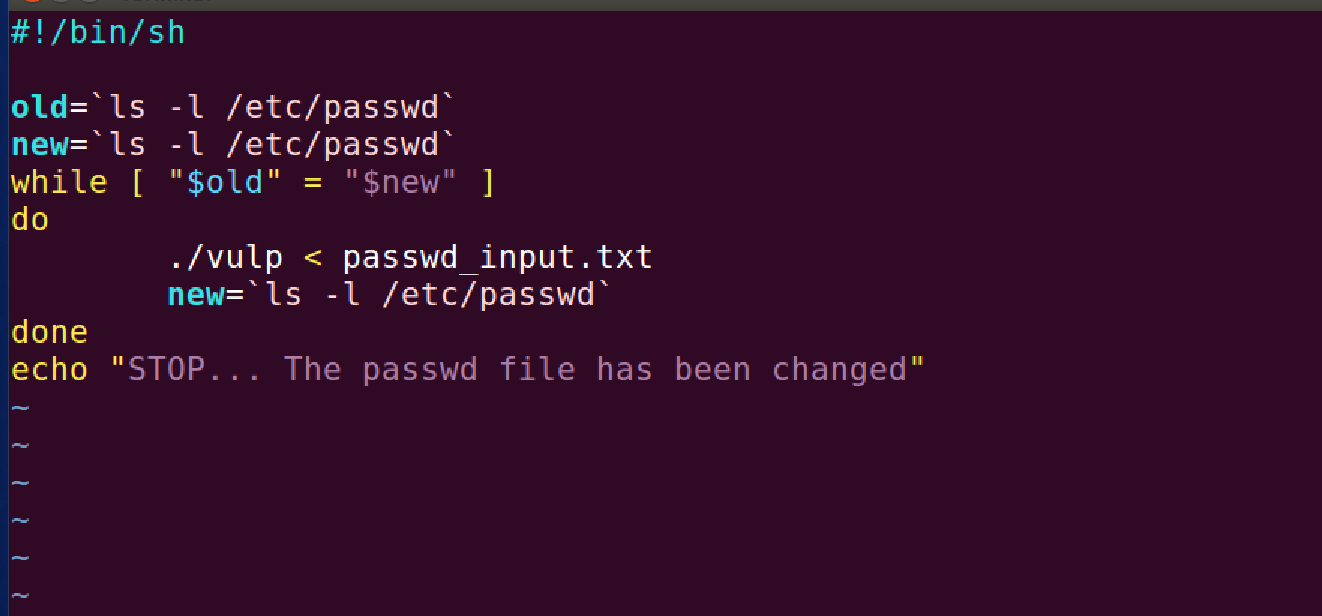


**Task2: Exploit the Race Condition Vulnerabilities**

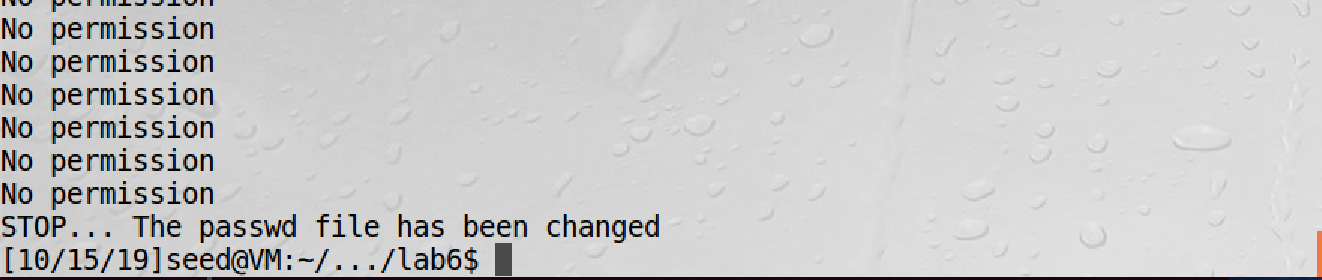
Step 1: We made the file call sym\_link\_pass.c. It will keep linking and unlinking the files:



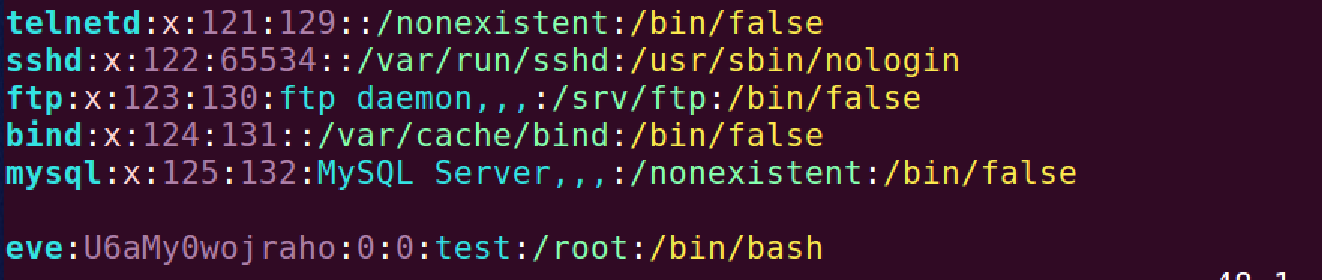
Step 2: The check.sh will keep executing to input the passwd\_input.txt to vulp.



Step3: After waiting for a while, we won the race to get the permission to modify the passwd file.

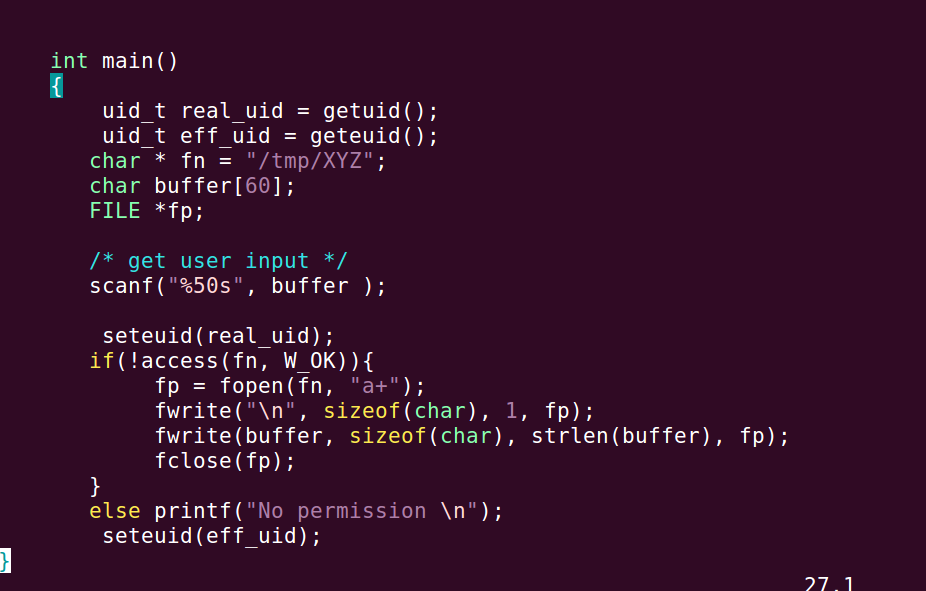


/etc/passwd file looks like the following: we added “eve:….” At the bottom of file successfully.

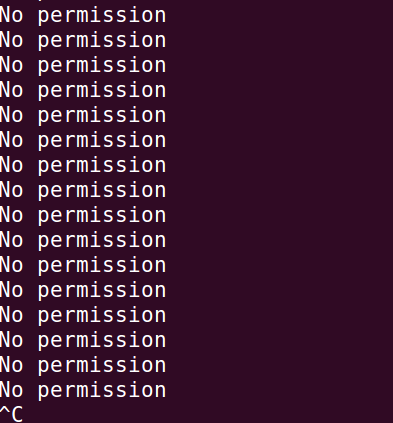


**Task3: Protection Mechanism B: Principle of Least Privilege**

The attack was not successful. This is because we can use seteuid to currently limit the privilege. I modify the vulp.c like the following:

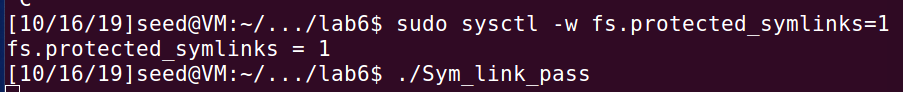


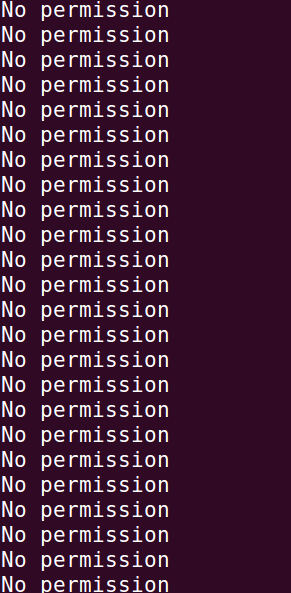
Output:



Task4: Protection Mechanism C: Ubuntu’s Built-in Scheme:

We can turn on the built-in protection to avoid such vulnability:





A long-standing class of security issues is the symlink-based time-of-check-time-of-use race, most commonly seen in world-writable directories like /tmp. The common method of exploitation of this flaw is to cross privilege boundaries when following a given symlink. When set to “0”, symlink following behavior is unrestricted. When set to “1” symlinks are permitted to be followed only when outside a sticky world-writable directory, or when the uid of the symlink and follower match, or when the directory owner matches the symlink’s owner. It depends on situation because there is still some limitation of this method. In addition, the limitation of this method is that If the symlink is just in a directory owned by the symlink’s owner, this protection will make no effect.