**CYBER SECURITY LAB**

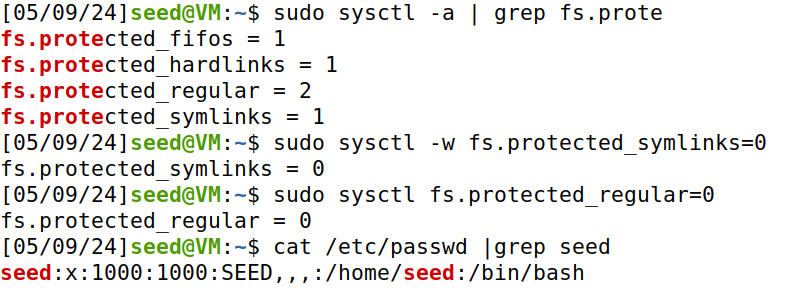
**RACE CONDITION**

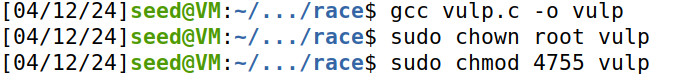
**(CMPG769)**

**ASTHA PATEL**

**Environment setup:**

To setup an environment we will follows following steps shown in following image.

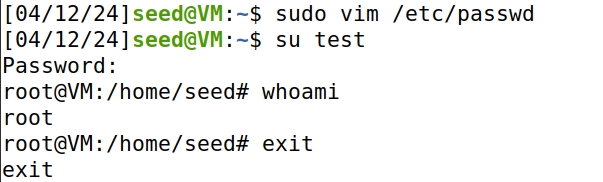


****

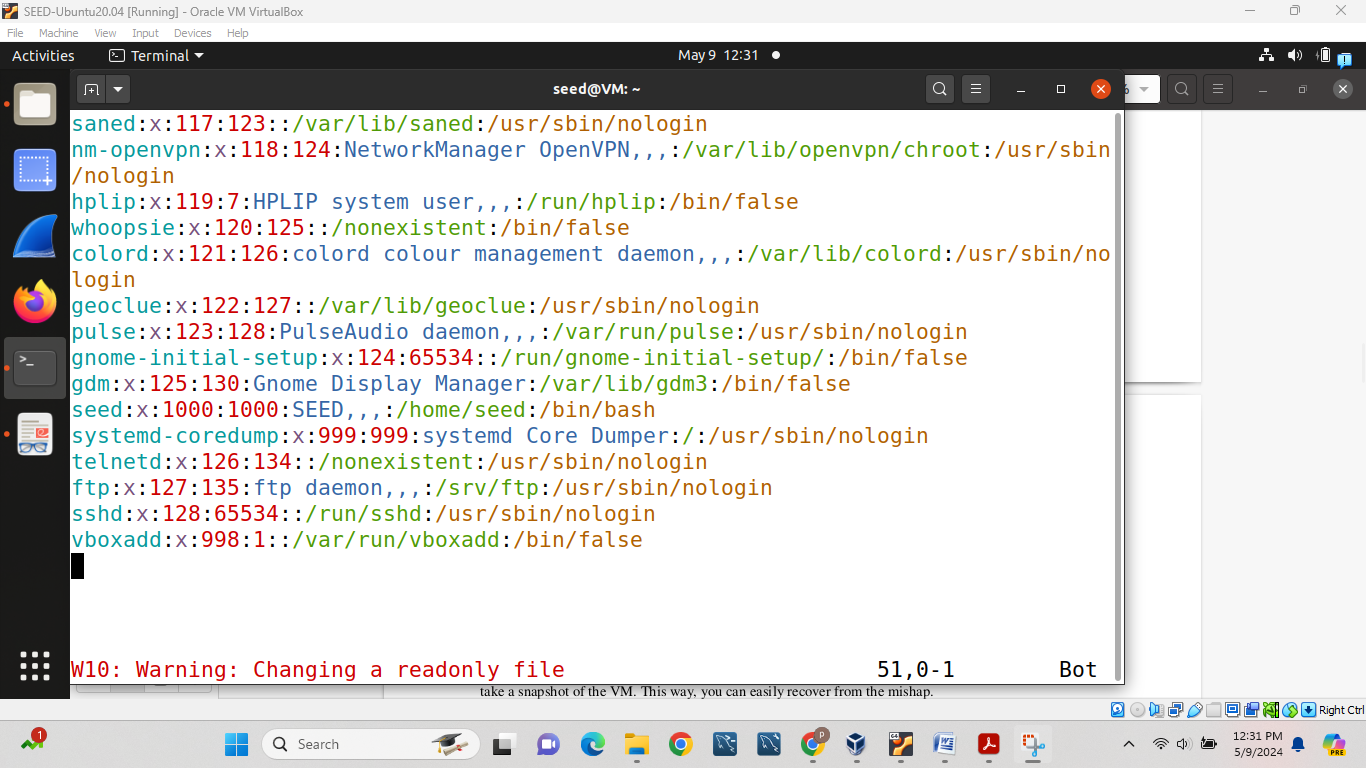
**Task 1: Choosing Our Target**

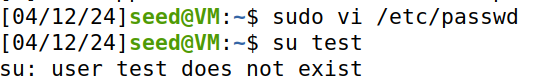
To verify whether the magic password works or not, we manually (as a superuser) add the following entry to the end of the /etc/passwd file. Please report whether you can log into the test account without typing a password, and check whether you have the root privilege.

With password it works and this can be seen as follows(When we add following command: test:U6aMy0wojraho:0:0:test:/root:/bin/bash):



Log into the test account without typing a password, and check whether you have the root privilege(When we remove following code: test:U6aMy0wojraho:0:0:test:/root:/bin/bash):





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

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

Using the time window that the vulnerable program's delay creates, demonstrate how to obtain root access to the system.

Steps:

1st modify the program to introduce a 10-second delay between **access()** and **fopen()** using **sleep(10).**

****

Create a symbolic link from **/tmp/XYZ** to **/dev/null** to redirect file operations.

Query: ln –sf /dev/null /tmp/XYZ

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

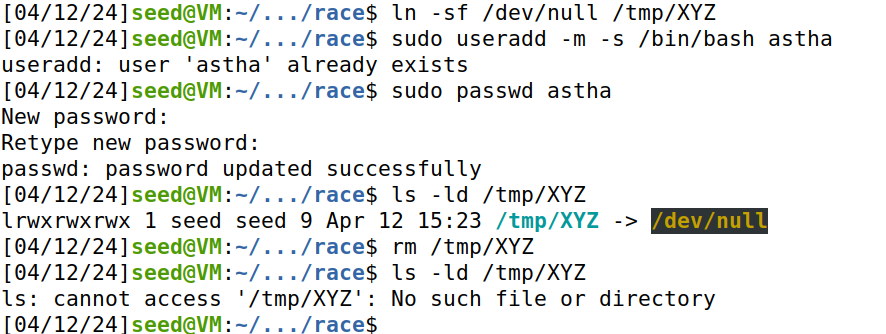
During the time window, manually add a root account to the system named astha and set password for it.

Queries:

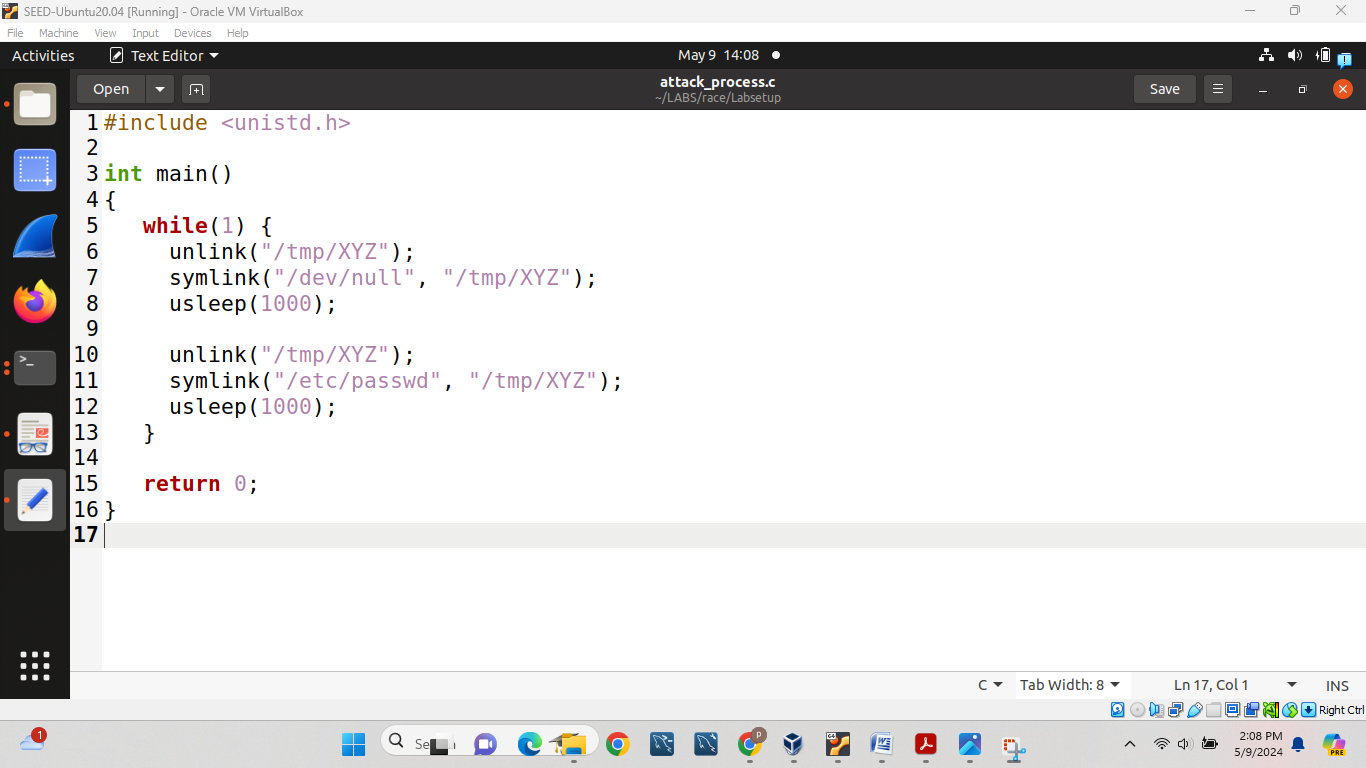
sudo useradd -m -s /bin/bash USERNAME

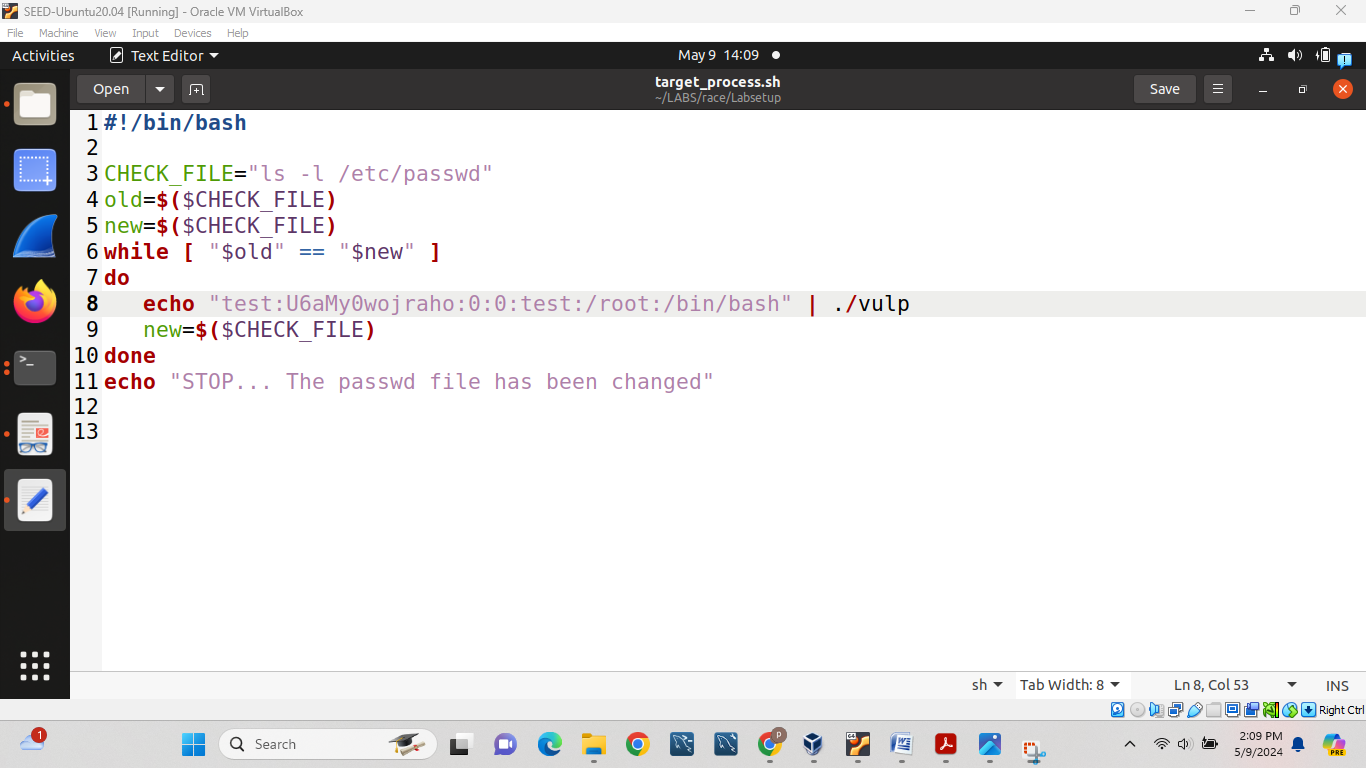
sudo passwd USERNAME PASSWORD

Then we will run following command: ls –ld /tmp/XYZ and check its working or not and now, if we want to remove the symbolic link when no longer needed by using following command: rm /tmp/XYZ and then again we try to run ls command it would show file not exist.

****

After that we copy attack\_process.sh and target\_process.sh and attack\_process.c and paste it in Labsetup file then we will that by using following command line by line:





gcc vulp.c –o vulp

sudo chown root vulp

sudo chmod 4755 vulp

ll

gcc attack\_process.c –o attack\_process

./attack\_process &

Top

Kill 83291

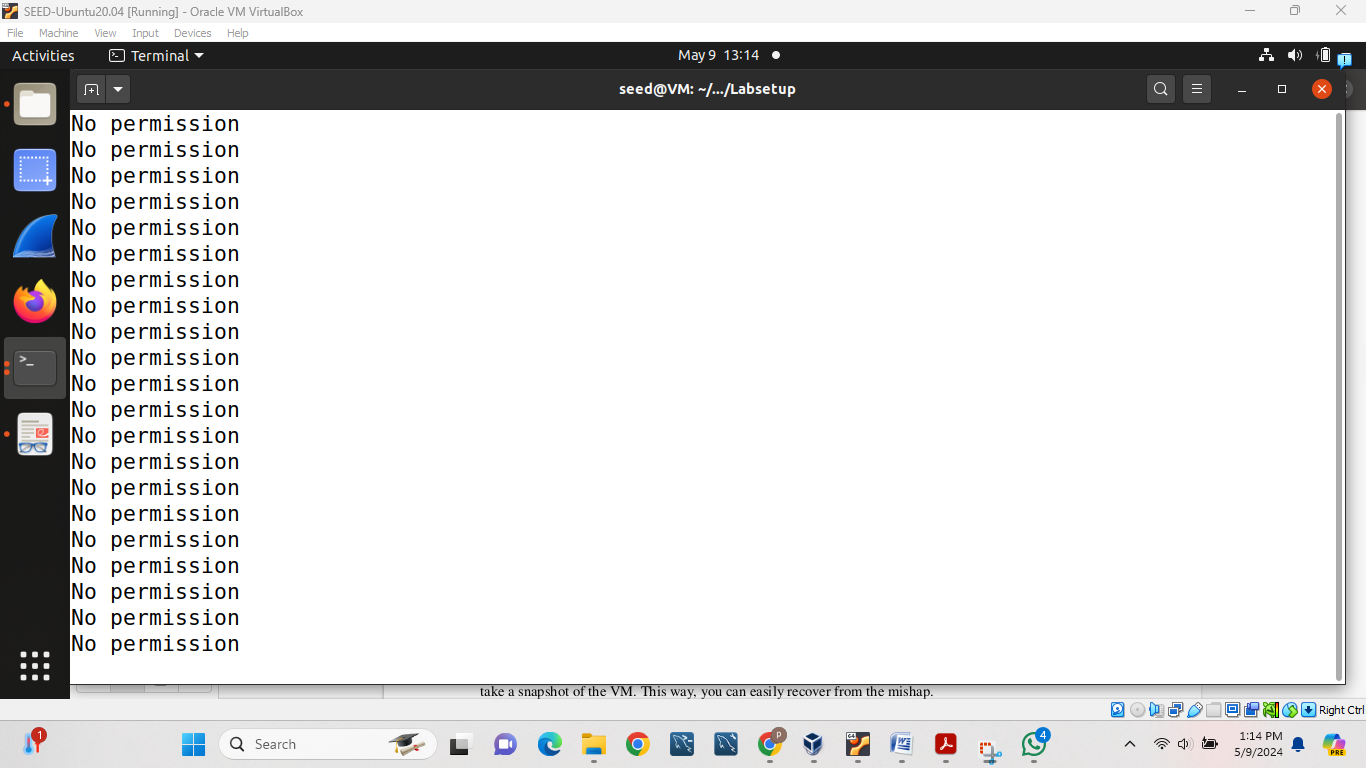
./attck\_process &

./target\_process.sh

And at the same time we will open tmp file by going home folder and typing cd tmp and then we will following code:

ll XYZ

****

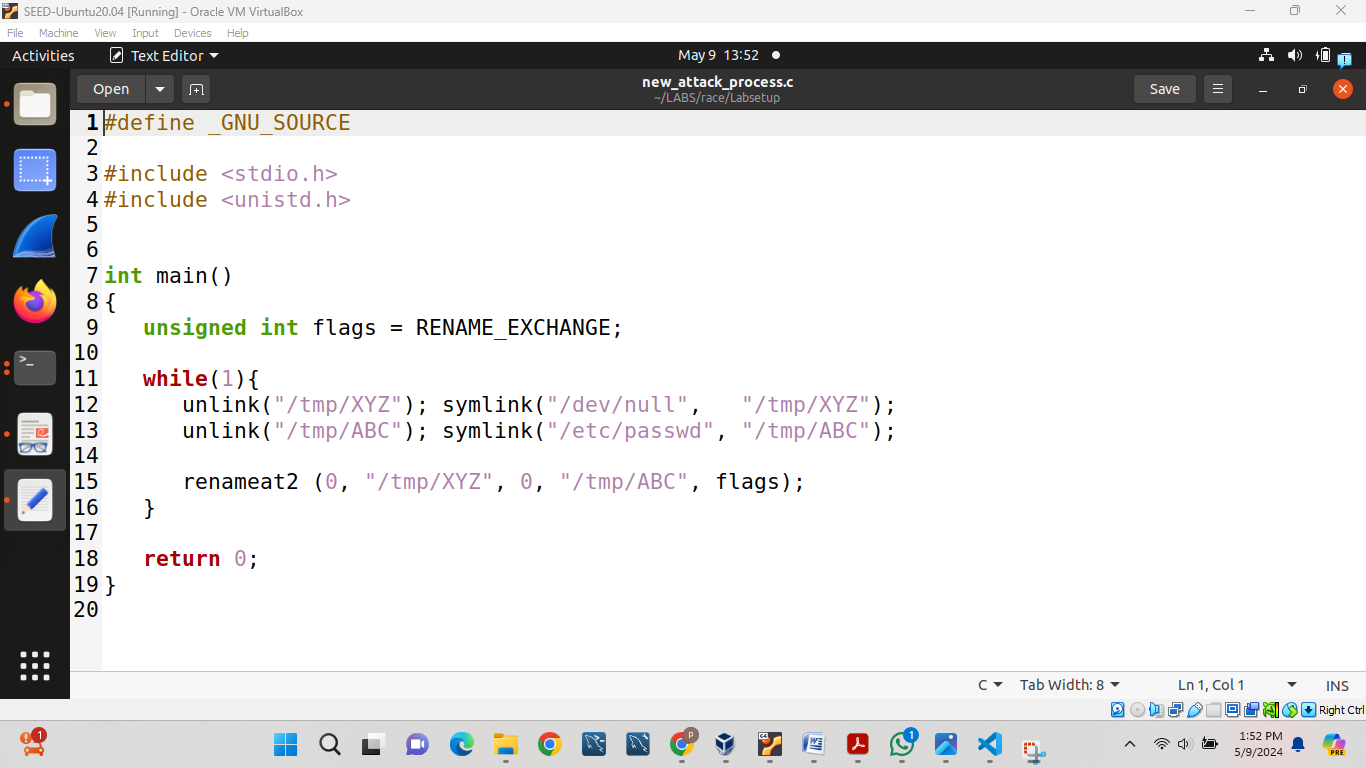
****

Then up to the time we get message like following we will wait for some time.

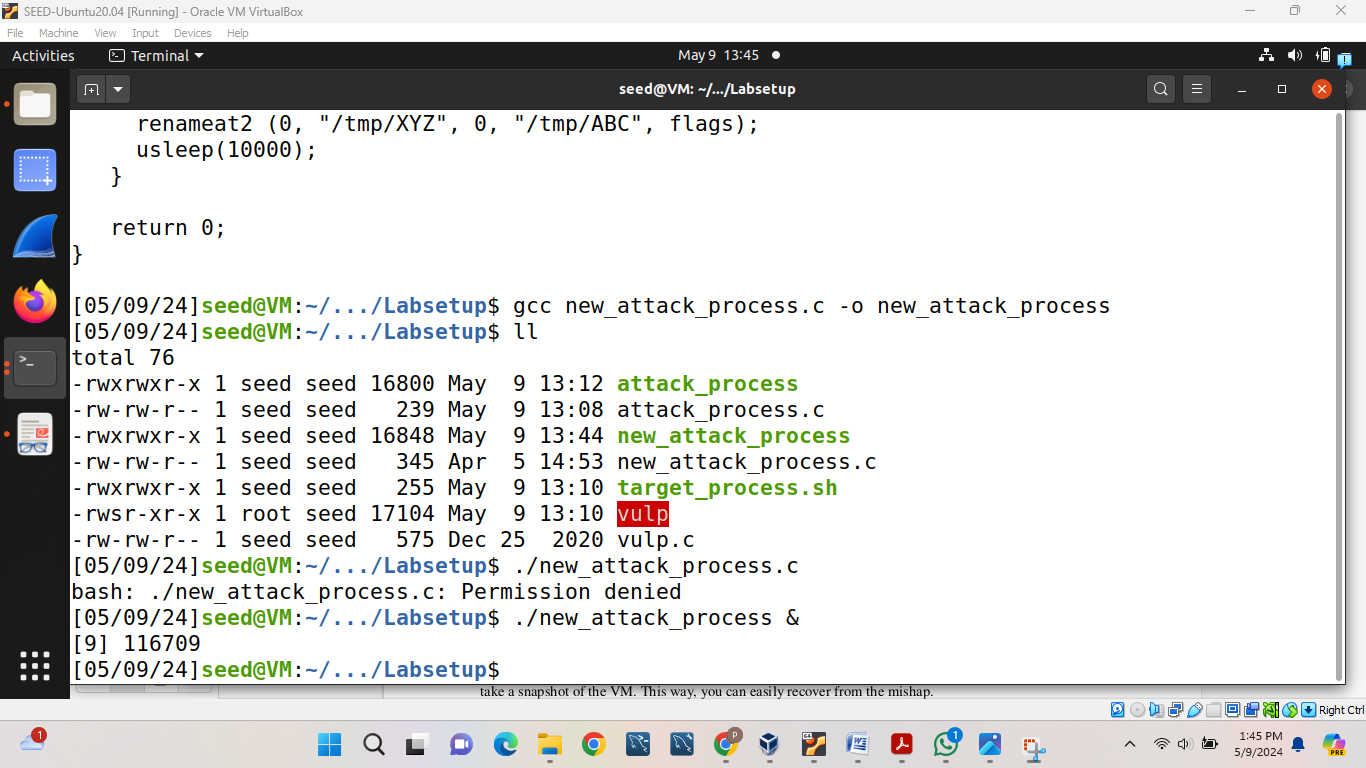
****

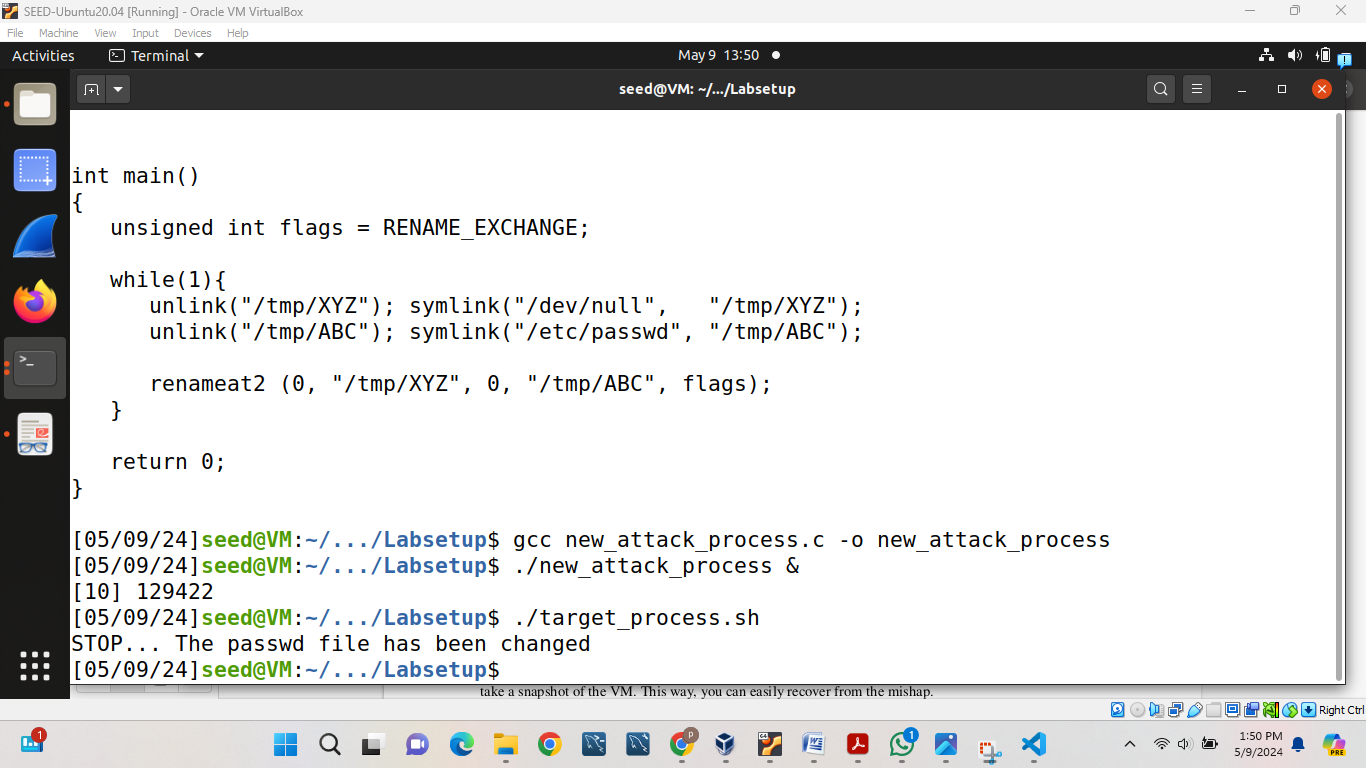
**Task 2.C: An Improved Attack Method**

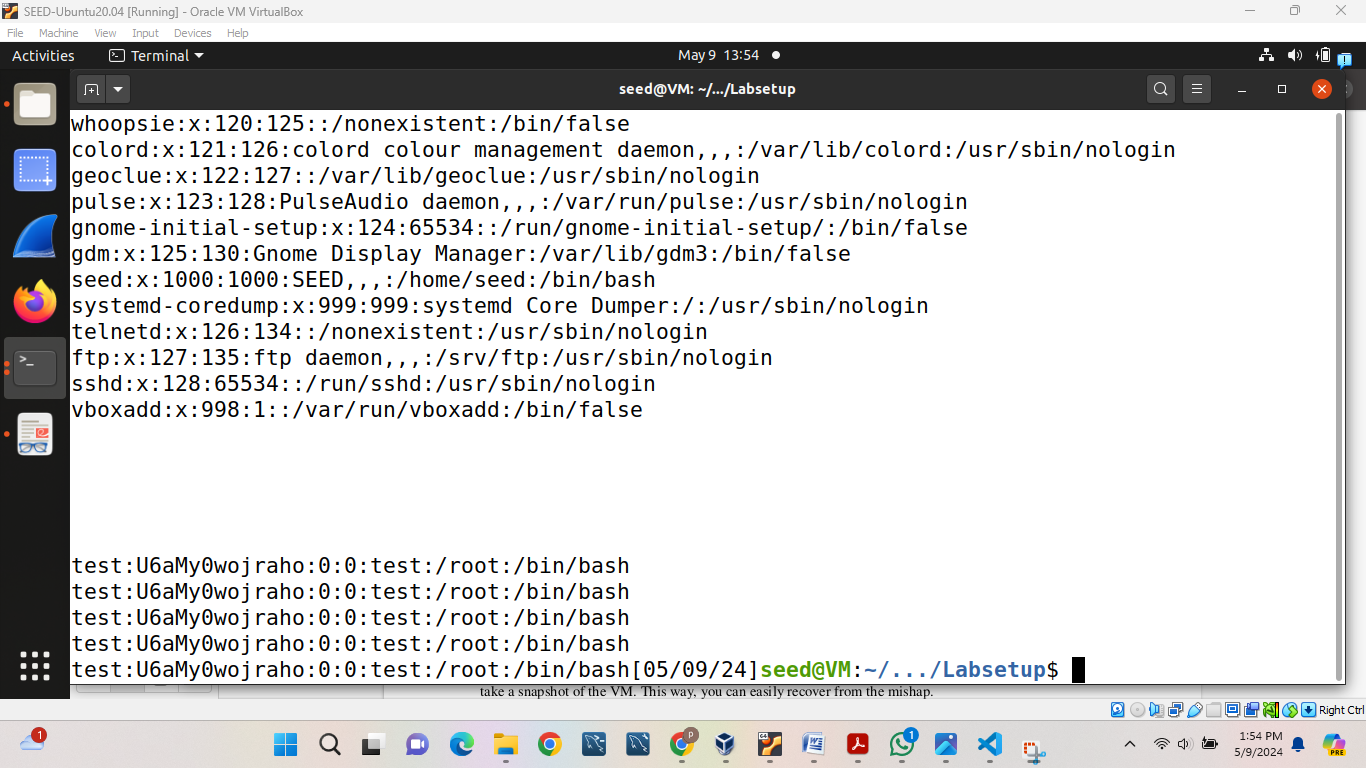
Then we will use new\_attack\_process.c as follows we will get result instantly.



We will follow same steps as we followed for getting task2(b) and as a result, we will get following result which shows we are able to change password easily:ss







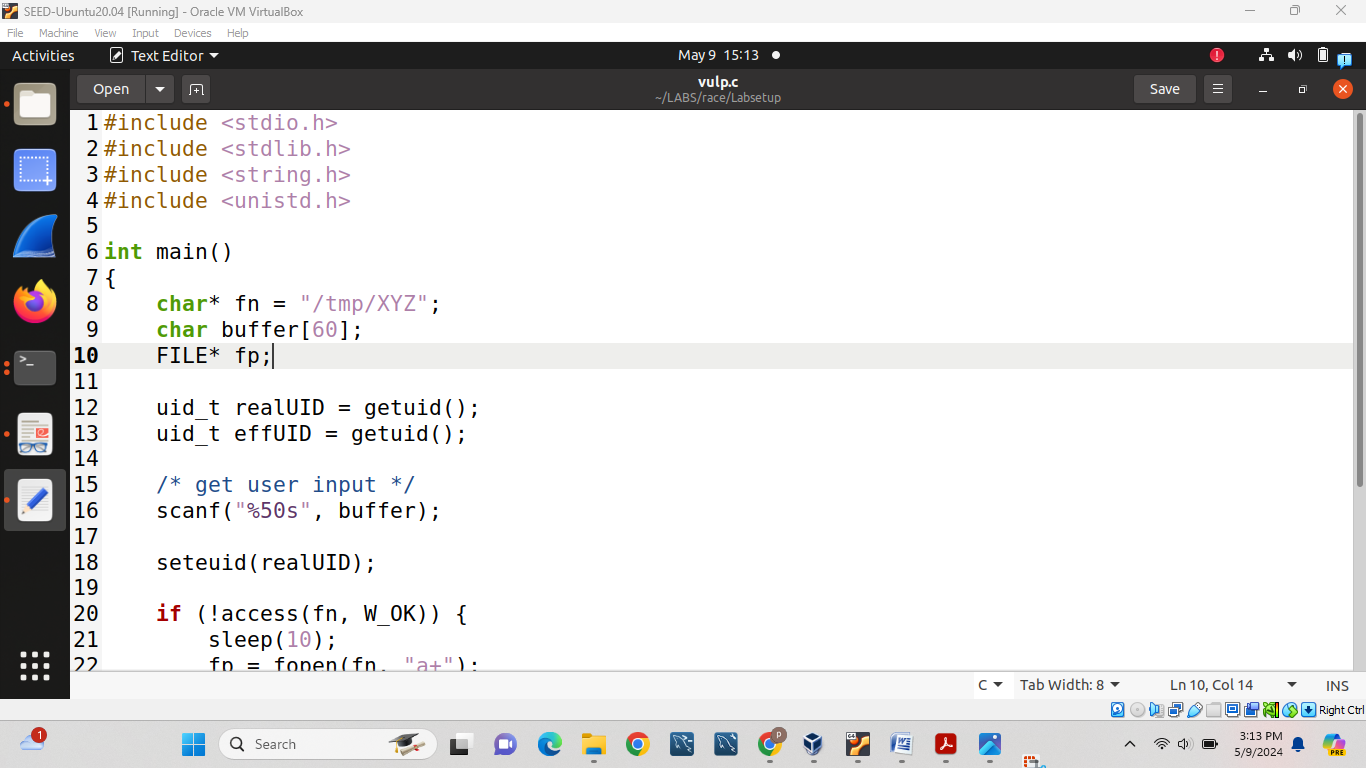
**Task 3: Countermeasures**

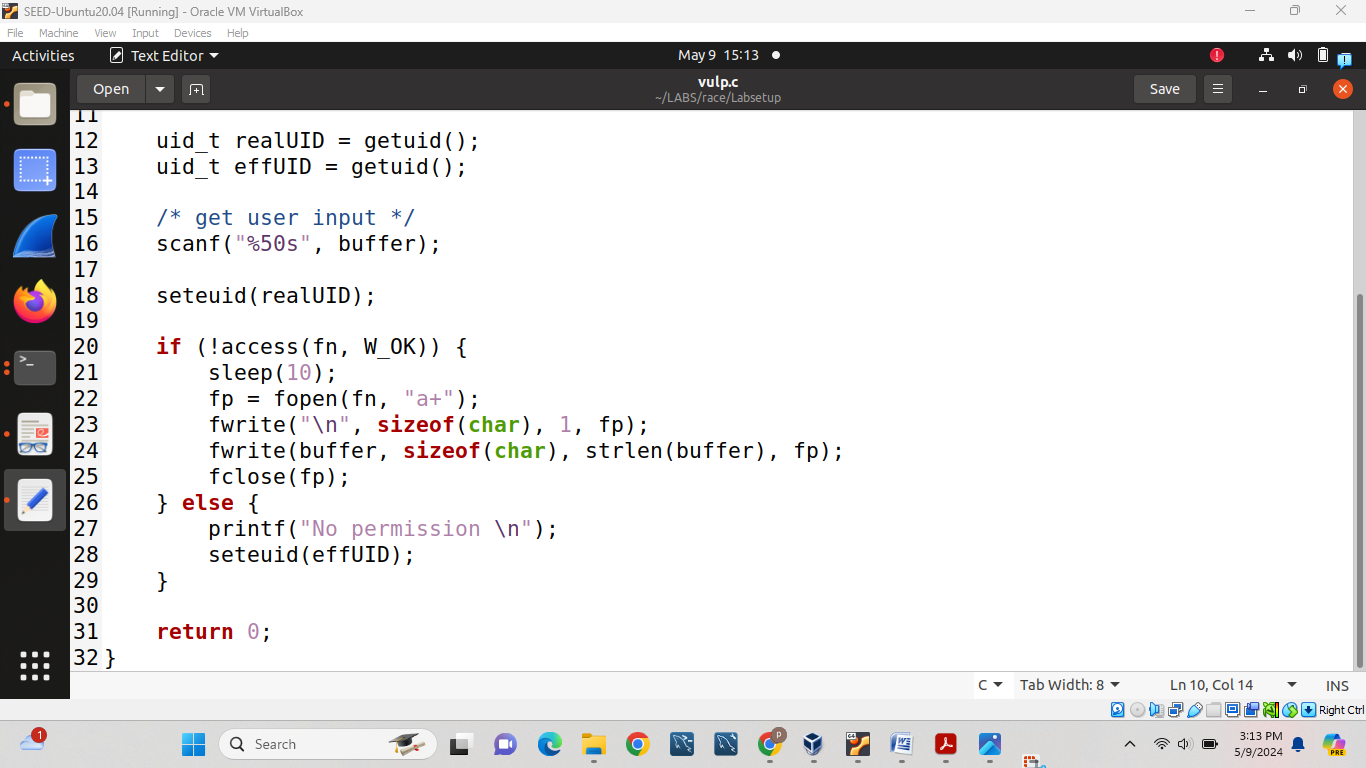
**Task 3.A: Applying the Principle of Least Privilege**

Will you be able to succeed?

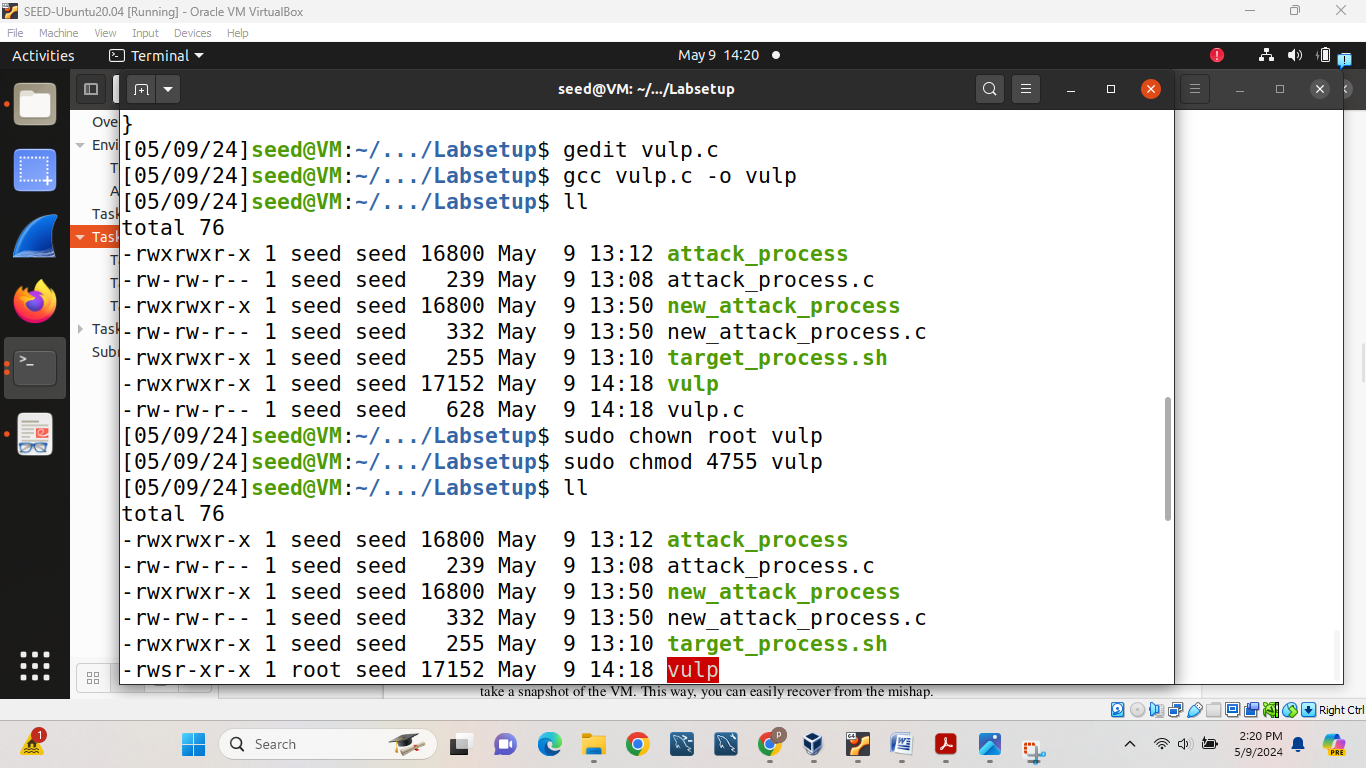
No, I do not able to succeed in this task.

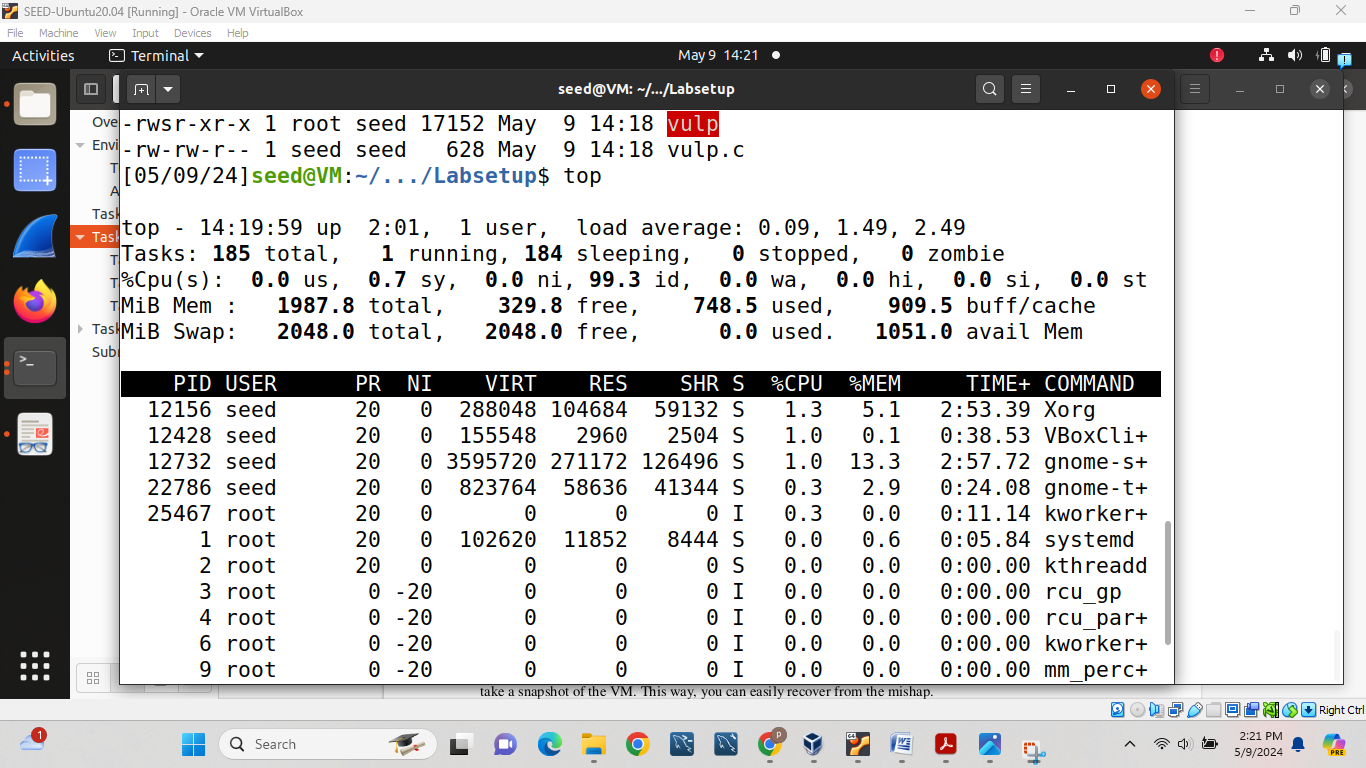
For achieving this task, we will change vulp.c code as follows:



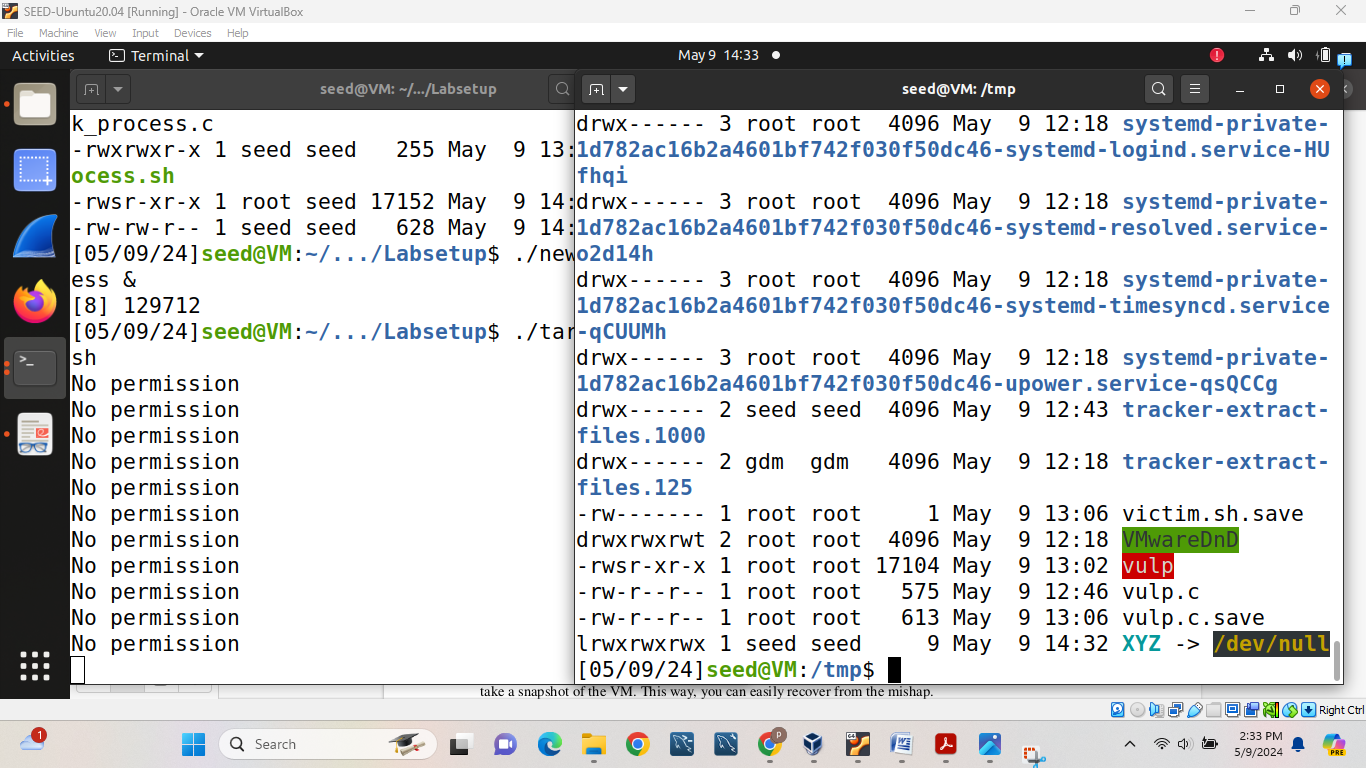


Then we will perform all the operations that we performed in task-2 but in this case we will not able to change password this can be clear after seeing following screenshot.







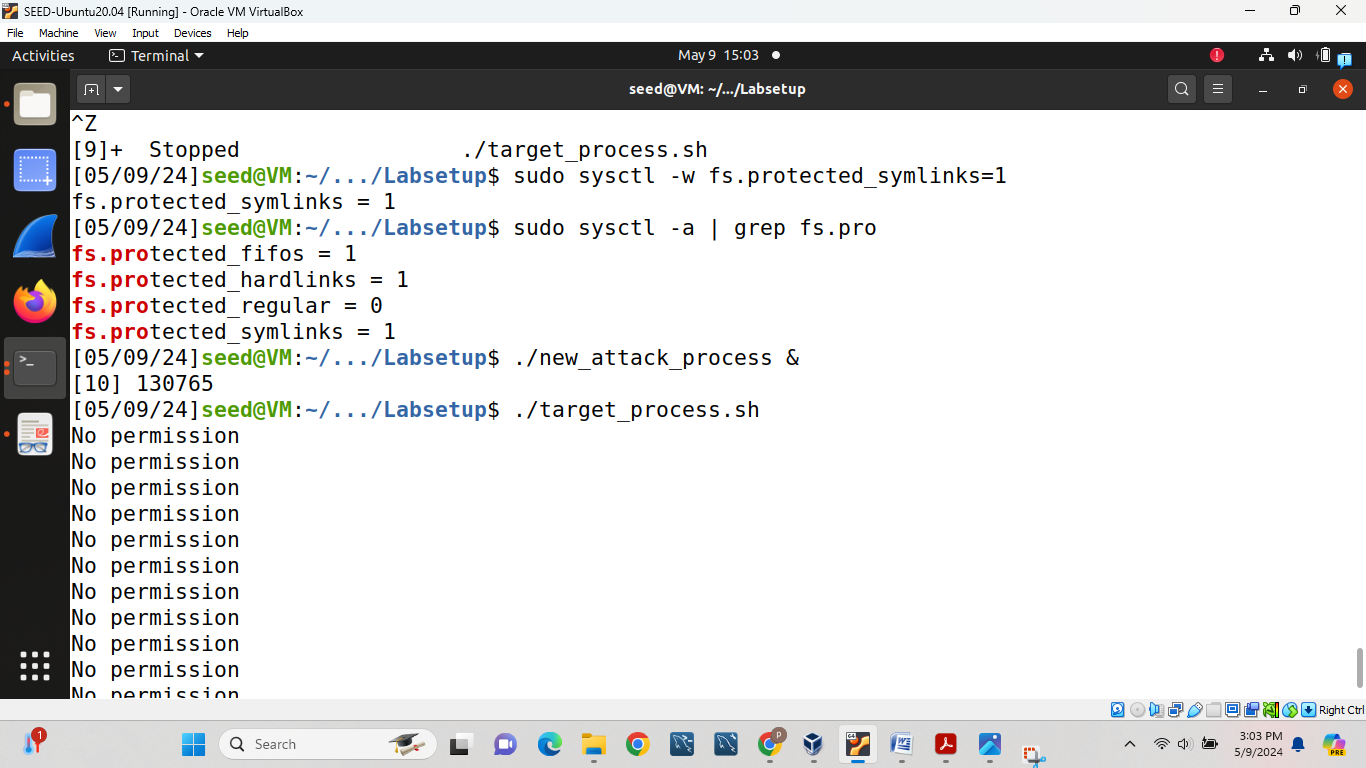


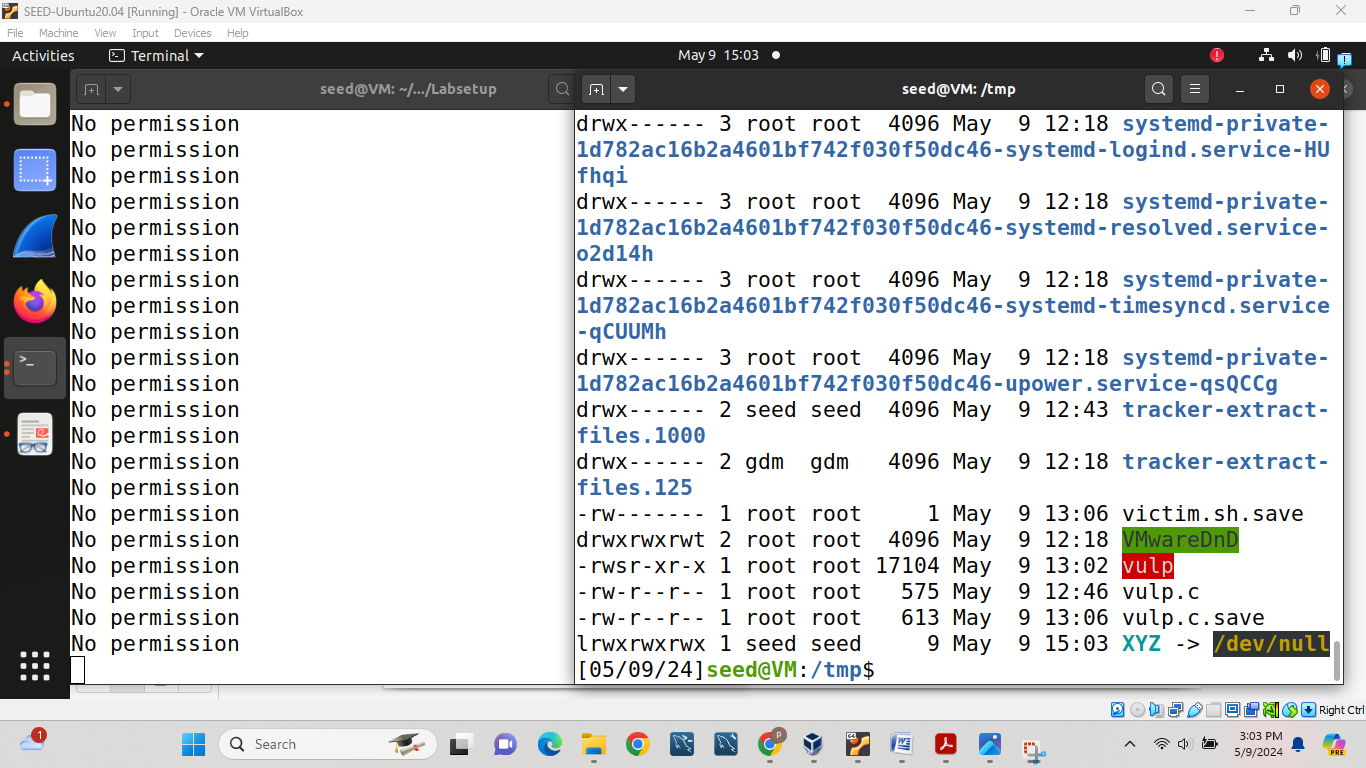
**Task 3.B: Using Ubuntu’s Built-in Scheme**

In this task first we will execute following command:

sudo sysctl -w fs.protected\_symlinks=1

Then we will follow all the operations to execute target\_process.sh as follows:

****

****

1. How does this protection scheme work?

Explanation:

1. To stop such attacks, this is an integrated defence system. Our attack was therefore unsuccessful. The protection scheme was effective because, in this instance, root is the follower, root is the owner of the /tmp directory, and seed is the owner of the symlink. One can notice that access will be refused from the screenshot above.   
2. Due to its limitations, this isn't a very effective protection method. Only sticky bit directories like/tmp and /var/tmp are supported by the mechanism. In this way, the attacker can obtain access by taking advantage of the race condition in other directories.

(2). What are the limitations of this scheme?

Restrictions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Follower(eUID)** | **Directory Owner** | **Symlink Owner** | **Decision(fopen())** |
| seed | seed | seed | Allowed |
| **seed** | **seed** | **root** | **Denied** |
| seed | root | seed | Allowed |
| seed | root | root | Allowed |
| root | seed | seed | Allowed |
| root | seed | root | Allowed |
| **root** | **root** | **seed** | **Denied** |
| root | root | root | Allowed |

* Only functional for directories that have the sticky bit turned on.
* As seen in the screenshot above, the protection mechanism only refuses access in a few instances. Both the directory's owner and the symlink's owner are seed in case 5, where the follower is root. Under such circumstances, the attack is successful. A root-owned file can be modified, and this can be exploited to make a race condition work in a directory owned by root.