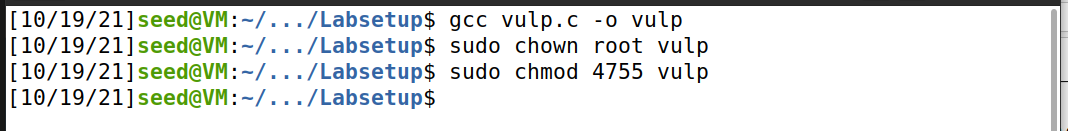
Assignment 6:

Graphical user interface, text, application

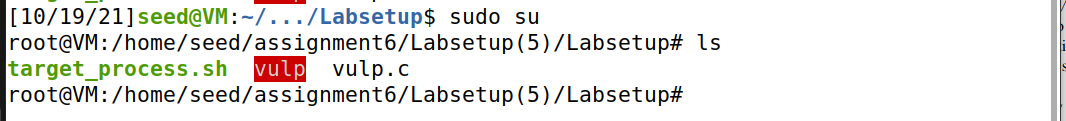
Description automatically generated

Turning of bult-in counter measures



Compiling the vulnerable c program that is given to us

Task1:



Open the terminal in root mode

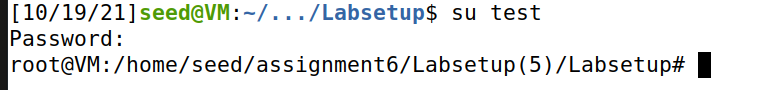


And access the /etc/passwd file using the vi editor

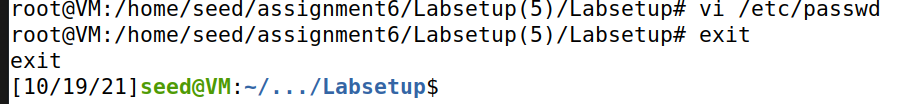
Text

Description automatically generated

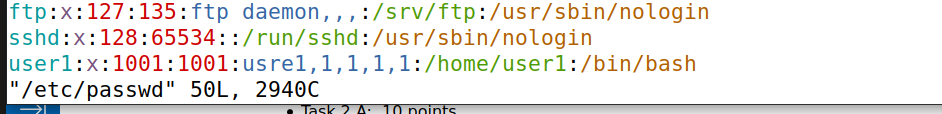
Adding the test user at the end of the file and saving it .



We can see that we are able to acces the test user terminal and this test user is a root user as the terminal open in root mode



Then we open the /etc/passwd file and clear the last line which we added for new user so that we can continue with the next task



Task2

2.a

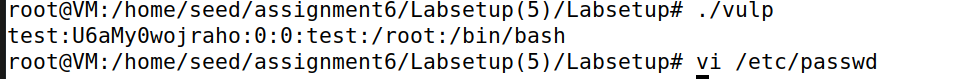


Making a symbolic link for /etc/passwd to /tmp/XYZ

A picture containing table

Description automatically generated

Sleeping the vulp.c program for 10 seconds using the sleep command in line 16 and the re compiling it



Executing the vulp.c program in root mode and giving the new user info in the sleep time when the program yields control to the system (in its sleep time)

Text

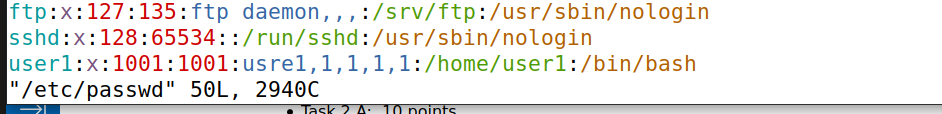
Description automatically generated with low confidence

We can see that the /etc/passwd file gets updated with the new user info



Here can see that the new user is created and has no password and this new user is a root user

2.b

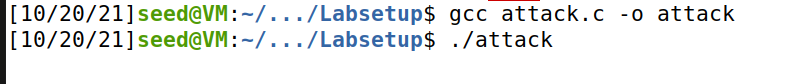


/etc/passwd file before task

Text

Description automatically generated

Createing the attack process code (c.program)



Compile and execute the attack process

Text

Description automatically generated

Vulp.c program without sleep

Graphical user interface, text, application

Description automatically generated

Target\_process.sh file that is already given to us

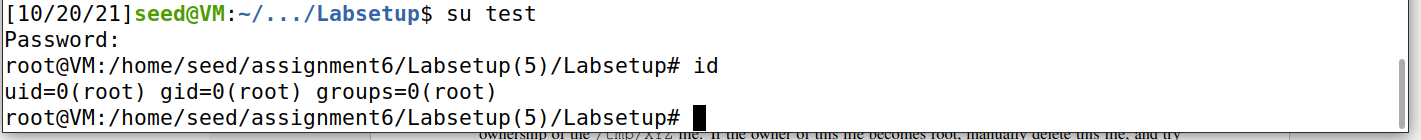


Execute the target process so that vulp.v vulnerable program gets executed multiple times until the target file gets updated

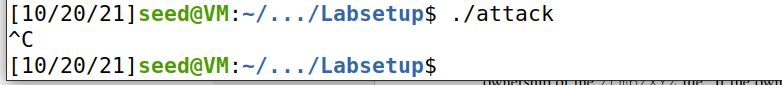
A picture containing timeline

Description automatically generated

We can see that the execution stops when the target file get updated



An then we test and see that we have added a new root user to the passwd file by exploiting the vulnerable program that has race condition vulnerability



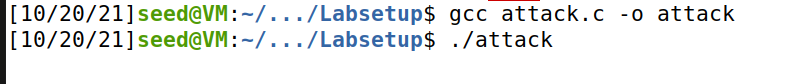
Terminating the attack process

2.c

Graphical user interface, text

Description automatically generated

Edit the attack.c program atomic using renameat2 and modifying the rest of the code accordingly



Compile and execute the attack process

Graphical user interface

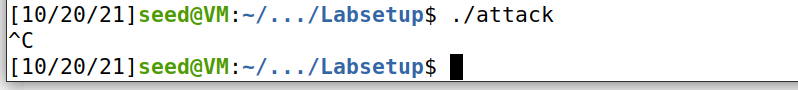
Description automatically generated with low confidence

Execute the target\_process.sh that already exists which executes vulp.c multiples times until the target file is updated. In this case the target file is /etc/passwd file

Graphical user interface, text

Description automatically generated

We can see that we are able to create a new root user called test by exploiting the race condition vulnerability and targeting the /etc/passwd file



Terminating the attack process

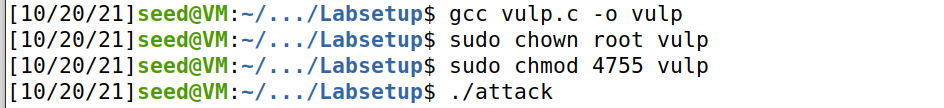
Task3:

3a:

Graphical user interface, text, application

Description automatically generated

Adding seteuid commands at line 16 and line 25 in the vulnerable program to drop privileges before opening the file and then restoring the privileges after file is closed (according to code)



Compiling the vulp.c program after the modifications and executing the attack code

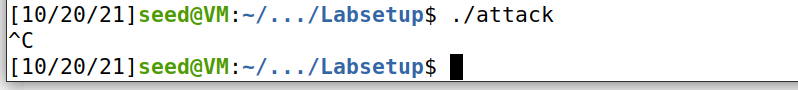


Executing the target\_process.sh for it to execute vulp.c program multiple times until it updates the /etc/passwd file

Graphical user interface, text, application

Description automatically generated

The program can’t open the target file as the privileges are dropped right before opening the file in the vulnerability program as we have done seteuid(1000) meaning the program is not going to have root privileges anymore cant access everything as it pleases which helps as a counter measure for the race condition vulnerability that exists in the vulnerable program



Terminating the attack process

3b:

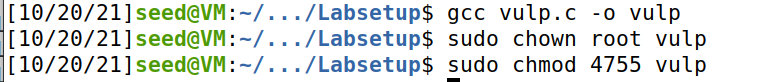


Enabling the ubuntu default symlink protection against race condition attacks

Text

Description automatically generated

Program changed back to how it was



Compiling the vulp program and making it a setuid program



Running the attack process

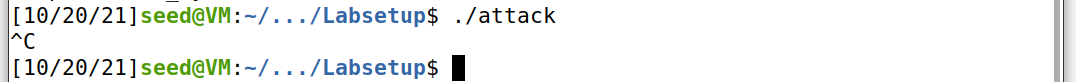


Running the target process to execute vulp.c multiple times

A picture containing shape

Description automatically generated

The ubuntu built-in counter measure against race condition also doesn’t allow the vulnerable program to open the target file and the attack fails



Terminating attack process

Question1:

The symlink protection only allows fopen() function when the owner of the symlink matches wither the follower (eid of the process) or the directory owner only

Question2:

It only works on directories that have its sticky bit enabled like the /tmp file or are world-writable