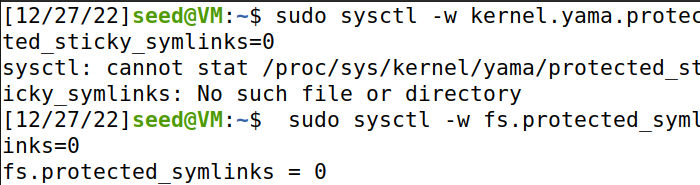
**21CY682 – Secure Coding lab - I**

**Name:Ramya Ajay**

**Roll No:CB.EN.P2CYS22004**

SETUP :

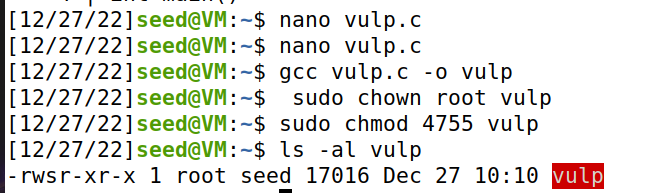


VULNERABLE PROGRAM

The following program is a seemingly harmless program. It contains a race-condition vulnerability.



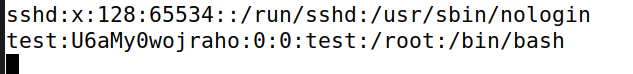
Set up the Set-UID program. We first compile the above code, and turn its binary into a Set-UID program that is owned by the root. The following commands achieve this goal

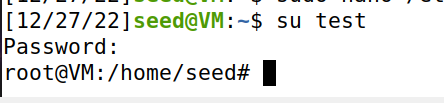


**Task 1: Choosing our target**

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.

test:U6aMy0wojraho:0:0:test:/root:/bin/bash





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

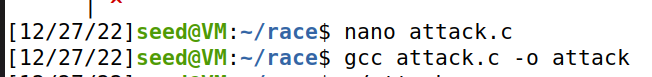
The goal of this task is to exploit the race condition vulnerability in the vulnerable Set-UID program listed earlier. The ultimate goal is to gain the root privilege.

After you have implemented the attack program, you should run it in the background, and then run the vulnerable program in parallel. If the attack fails, the vulnerable program will crash, so you need to run the vulnerable program repeatedly until the attack is successful.

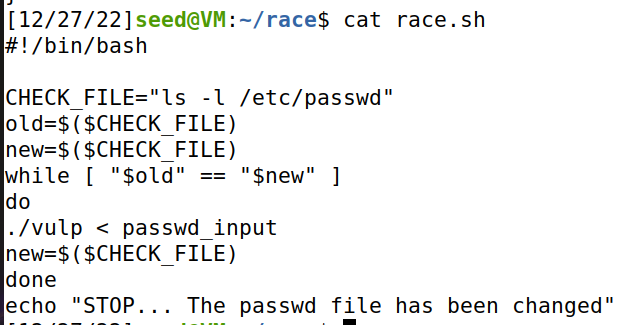
The shell script runs the vulnerable program (vulp) in a loop, using passwd input as the input. If the attack is successful, i.e., the passwd is modified, the shell script will stop

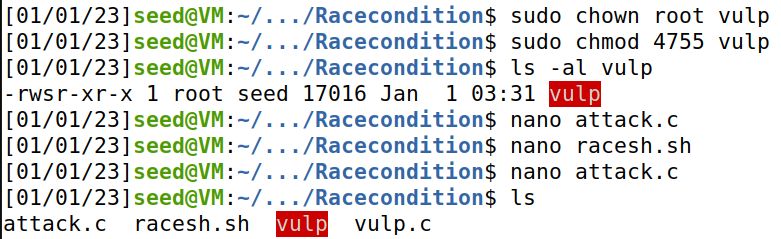
**Attack.c**



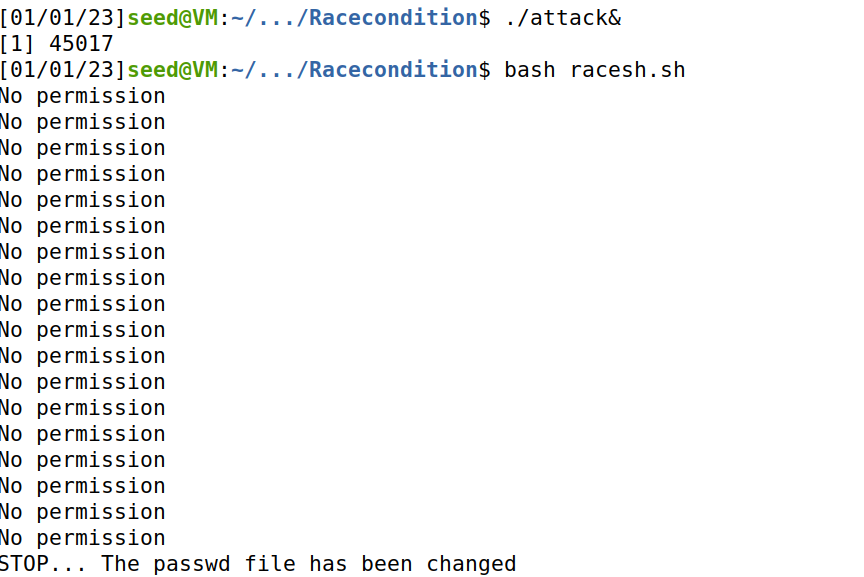


**Race.sh**

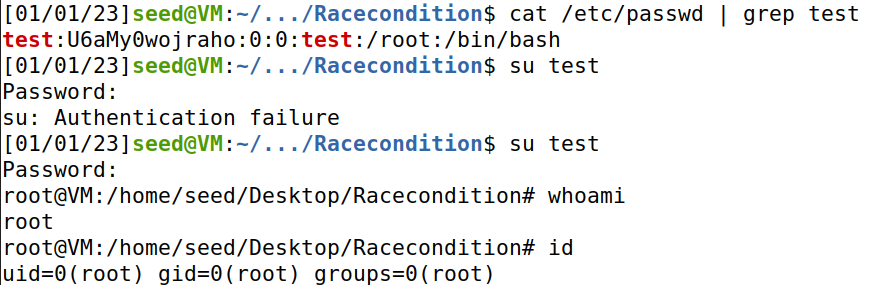




On running the attack program in background and race.sh bash script , after some time we get a STOP message which means the password file has been changed

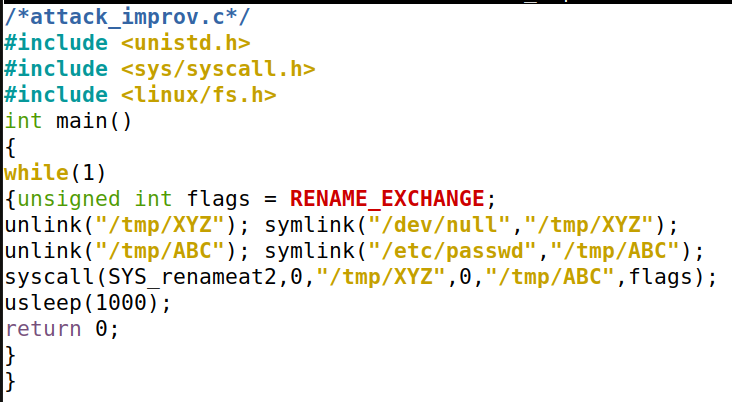


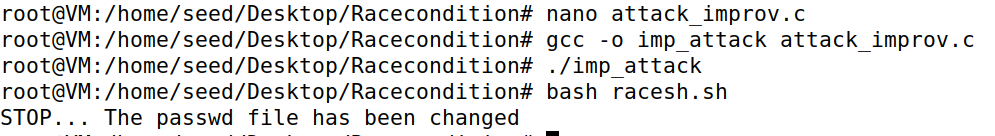
We can verify it by checking out the passwd file and su into the test user.



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

The following program first makes two symbolic links /tmp/XYZ and /tmp/ABC, and then using the SYS renameat2 system call to atomically switch them. This allows us to change what /tmp/XYZ points to without introducing any race condition

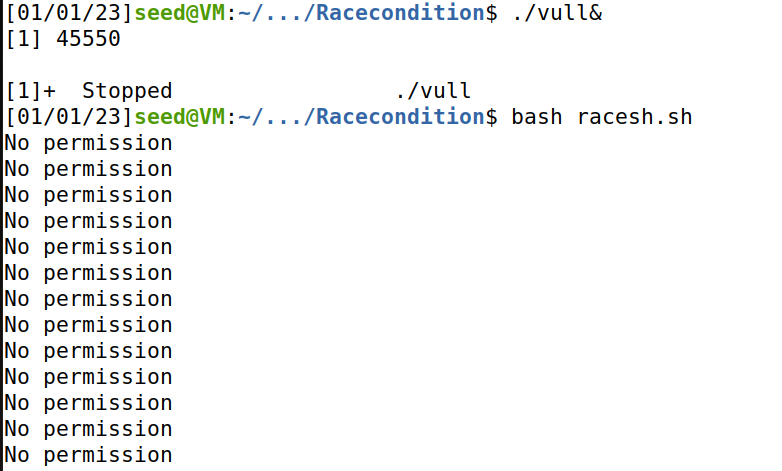




**Task 3: Countermeasure: Applying the Principle of Least Privilege**

The fundamental problem of the vulnerable program in this lab is the violation of the Principle of Least Privilege. The programmer does understand that the user who runs the program might be too powerful, so he/she introduced access() to limit the user’s power. However, this is not the proper approach. A better approach is to apply the Principle of Least Privilege; namely, if users do not need certain privilege, the privilege needs to be disabled. We can use seteuid system call to temporarily disable the root privilege, and later enable it if necessary.

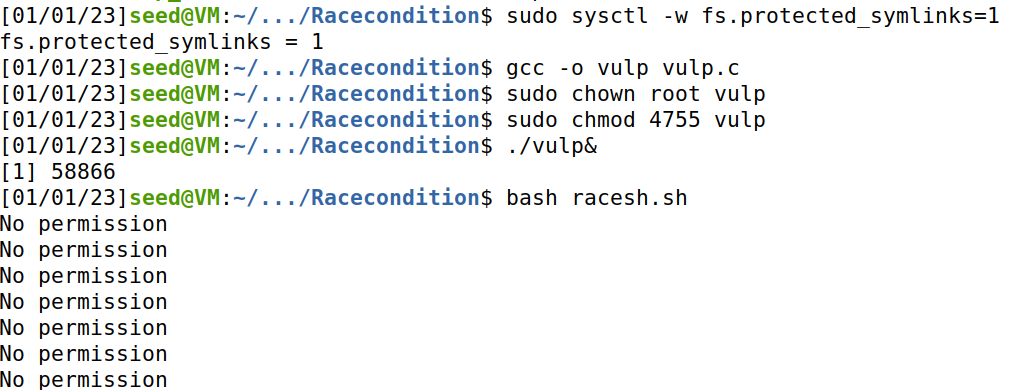




Here the Race Condition attack is unsuccessful because of the program using principle of least privilege.

**Task 4: Countermeasure: Using Ubuntu’s Built-in Scheme**

Turn the protection back on using the following commands:



Now also the attack is not successful, and we get No permission in the output .