CYBR 3130 Secure Programming – Fall 2022

Andrew Koenig

Lab 4 – Race Condition Attack

**Background:**

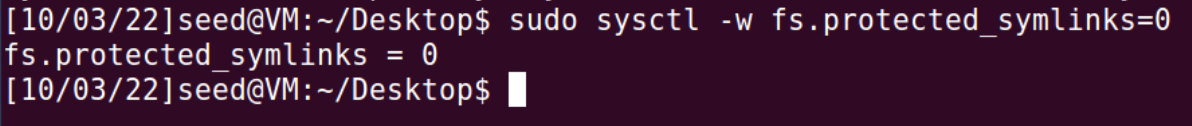
The purpose of this lab is to help you get familiar with the Race Condition Attack. Supervised section of Lab 4 will be in 10/3/2022. The deadline to submit the report is 10/10/2022.

**Task 1 Exploiting the Race Condition Vulnerability**

**Step 1. Turn off countermeasure.**

Ubuntu 10.10 and later come with a built-in protection against race condition attacks. In this lab, we need to disable this protection. You can achieve this using the following commands:

$ sudo sysctl -w fs.protected\_symlinks=0

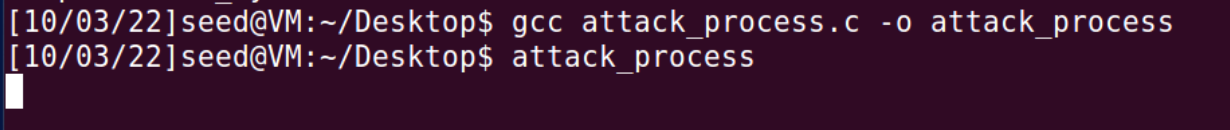


**Step 2. Compile and run the attack processes**

Compile and run the attack program **attack\_process.c**. The attack process will keep changing the symbolic link from **tmp -> null** to **tmp -> /etc/passwd**, and vice versa.

$ gcc attack\_process.c -o attack\_process

$ attack\_process



**Step 3**. **Check if the attack process is successful**

Up to this point, you suppose to run the attack process which will keep changing the symbolic link. You should check whether the symbolic change is successful or not before you can proceed.

Open **another terminal** and use $ **ls -l /tmp/** to check how does the symbolic change.

$ ls -l /tmp/

Text

Description automatically generated

**Step 4. Compile and run the vulnerable process**

Compile the vulnerable program **vulp.c** and turn it into a **root-owned Set-UID** program. The vulnerable process will check the permission of the process to the tmp file using access() and open and write into the tmp using open().

$ gcc vulp.c -o vulp

$ sudo chown root vulp

$ sudo chmod 4755 vulp

Text

Description automatically generated

In the new terminal, run the **target\_process.sh** in parallel with the attack program. This shell script will keep running the vulnerable process until the attack is successful. At this moment, you are running two processes – the attack process and the vulnerable process – at the same time and generate different orders of instructions. Let wait for the undesired one to launch the attack.

(If after 2 minutes, your attack is still not successful, you can stop the attack, and double check step 1 to step 4 or ask professor for help)

$ sudo chmod 777 target\_process.sh

$ target\_process.sh

Text

Description automatically generated

**Step 5. Check the experimental results**

If your attack is successful, the attack should stop automatically at step 4 and the /etc/passwd file should be changed. Check the **/etc/passwd** file and see if you have successfully added a new root user (test) with a normal user’s privilege.

$ gedit /etc/passwd

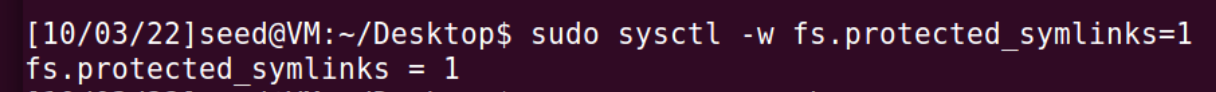
Text

Description automatically generated

**Task 2 Exploiting the countermeasure**

**Step 1**. Turn-on the built-in countermeasure.

$ sudo sysctl -w fs.protected\_symlinks=1



**Step 2.**  Try to launch the attack again. Can you successfully launch the attack this time? (1 point)

$ target\_process.sh

Text

Description automatically generated

You need to submit a detailed lab report for Task 1 to describe what you have done and what you have observed, including screenshots and code snippets. If you have any questions regarding the lab, please watch the lecture videos or contact Dr. Zhang [lzhang@ucmo.edu](mailto:lzhang@ucmo.edu). This lab will be worth 10 points.

In this lay we got practical experience running a race attack in linux. We ran a program that would change the pointer between a file we can edit and one we shouldn’t be able to edit. Then we ran another program at the same time that was vulnerable and would let us access the /etc/passwd file when we don’t have the correct permissions to do that.