Name: Bharath Karumudi

Lab: Race Condition and Dirty COW attack

Lab Setup:

A screenshot of a computer

Description automatically generated

Observation: disabled the sticky symlinks protection and also created a root owned set-UID program: vulp with the given code in lab manual.

Explanation: The sticky symlinks protection disabled to perform the attack, as it is one of the countermeasures in Ubuntu, to perform the further lab exercises, we need to disable.

**Task1: Choosing our target**

A screenshot of a computer

Description automatically generated

A close up of a sign

Description automatically generated

**Observation:**

By editing the /etc/passwd file by adding a new line with the given content for the test user, the new user id was created and the password is set to blank with the magic password. When switched to the user “test”, just hit the enter and I was able to enter into “test” user account with root shell. This was verified with id and then removed the entry from /etc/passwd file.

**Explanation:**

The /etc/passwd file will have all the users and the second part is hashed password and third part is UID. As the new user test was added to the file with the magic password which is equivalent to blank and the third part is set to 0, a root privileged account was created and able to login to the test user account without password with root access.

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

A screenshot of text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Observation**: Created a new program called “attack\_process” with normal privileges that can alter the symlink between “myfile” and “/etc/passwd” file. Also, created a program “target\_process.sh” which can run the “vulp” in loop until the “/etc/passwd” file is changed. The input for the vulp was given through the file called “passwd\_input” which has the “test” user details as shown in the screenshot. By running the program “attack\_process” and “target\_process” after sometime, the /etc/passwd file was modified at a new user was added to the passwd file. Verified by checking the file and also by logging in and we can see we have a root shell.

**Explanation**: The /tmp/XYZ was initially pointed to user file “myfile” and then in vulp program the access was verified and after context switching to attack\_process and again back to vulp which is a root owned set-UID program which is capable of writing the content to the /etc/passwd file. Here, the access() verifies the real UID permissions on the file, where as, the open() checks on effective Uid. The window between the check and use was used to attack by brute-forcing and context switching. Thus, the program was able to make a new entry to the “/etc/passwd” file with a new user details and the attack was successful.

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

A screenshot of text

Description automatically generated

A screenshot of text

Description automatically generated

A screenshot of a computer

Description automatically generated

**Observation**: Modified the vulp, by adding the Principle of least privilege policy, by setting the real user id for effective user id, before open() system call. When executed the attack again, it was never success.

**Explanation**: This is due to limiting the root privilege as part of Principle of Least Privilege. By updating the effective user id before system call, now the privilege was dropped, and the open system call will fail to open the file. Thus, the attack will never be success and applied the countermeasure for race condition attacks.

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

A screen shot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

**Observation**: The Ubuntu countermeasure sticky symlinks was turned on back with ﻿“sysctl -w fs.protected\_symlinks=1” and ran the attack with original vulp program **without** having the principle of least privilege changes. When ran the program, observed the Segmentation fault error and the attack was failed.

**Explanation**: The attack failure was due to countermeasure by Ubuntu – sticky symlinks enabled. (1) The protection works, even if attackers can win the race condition, they cannot cause damages. The protection only applies to world-wide sticky directories, such as /temp. This is enabled by default in Ubuntu. When it is enabled, symbolic links inside a sticky world-writable directory can only be followed when the owner of the symlink matches either the follower or the directory owner. (2) The limitation with this is, it applies to only word-writable sticky directories and under certain conditions.

**Part 2: Dirty COW attack**

**Task 1: Modify a Dummy Read-Only File**

* 1. Create a Dummy file

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Observation:

Explanation: