CS 449 Assignment 3

Release Nov 5th, 2023; Due Nov 20th, 2023

Instructions

This assignment is due on Nov 20th 23:59:59 EST. The whole assignment has been tested on a 64-bit Ubuntu 22.04 virtual machine. Using a Linux virtual machine to set up the environments and complete this assignment is **highly recommended**. If you have trouble running a Linux virtual machine on your own computer, you can go to the UNIX/PC Lab (M-3-731), Web Lab (M-3-732), or IT Lab (M-3-730) of our department where you can find computers pre-installed with virtualization software, including VMware Workstation Pro 17 and Oracle VirtualBox 7.

Your submissions will have one folder. Place required files in this folder, using the exact names and conventions specified in the question text. Please zip this folder without encryption, rename the zip file as CS449A3.first_name.last_name.studentID.zip, and submit it on Blackboard.

Please start working on this assignment early. No extension will be given for assignments and this policy will be strictly enforced for all students.

Question 1 Race Condition (40 points)

A race condition can arise when multiple processes are trying to access and modify the same data at the same time. If the order in which the access takes place is different than expected, it can result in unexpected outcomes. During the class, we have discussed the race condition vulnerability for Set-UID programs. We also looked at an attack named `Meltdown`, which exploits race conditions inside the design of many modern CPUs.

In this question, you are given a program called `vprog.c`, which has a race condition vulnerability. `vprog.c` is a root-owned Set-UID program that appends a string of user input to the end of a temporary file `/tmp/XYZ`. Since `vprog.c` runs with the root privilege, it can overwrite any file. To prevent `vprog.c` from overwriting other users' files, the program first checks whether the real user ID has access to the file `/tmp/XYZ` using the `access()` function. If the user who runs this program has access to `/tmp/XYZ`, to program will continue to open `/tmp/XYZ` and append the user's input to it.

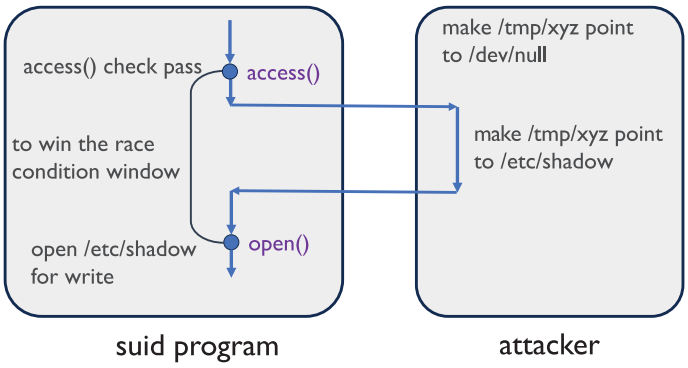

Figure 1: Race condition in `vprog.c`.

As illustrated in Figure 1, `vprog.c` has a race condition vulnerability. There is a time window between the execution of `access()` and `fopen()`. If an attacker can make `/tmp/XYZ` a symbolic link pointing

`/etc/passwd`, which is a root-owned protected file, inside this time window, the attacker can use `vprog.c` to appended anything to `/etc/passwd`.

Environment Setup

(a). Turn off protections. If using Ubuntu, it implements protection techniques to mitigate race condition attacks. In this assignment, we use the following commands to turn off the protections to make the attack easier:

```
sudo sysctl -w fs.protected_symlinks=0
sudo sysctl fs.protected_regular=0
```

(b). Compile the vulnerable program. Use the following commands to compile the vulnerable program and make it a root-owned Set-UID program:

```
gcc -o vprog vprog.c
sudo chown root vprog
sudo chmod 4755 vrpog
```

(c). Please make a copy of the `/etc/passwd` file in case you accidentally corrupt it during this assignment using the following command:

```
sudo cp /etc/passwd /etc/passwd.back
```

Task

You are given the vulnerable program `vprog.c` which **should not be modified**. In this question, your task is to exploit the race condition vulnerability in `vprog.c` and use it to modify the protected `/etc/passwd` file. Speficitlly, you want to add the following to `/etc/passwd`:

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

In Ubuntu, this would create a root user named "test" whose password is empty, i.e., we only need to press the return key when asking for the password. If not using Ubuntu, you can copy the hashed password of your own non-root account from `/etc/shadow` to replace `U6aMy0wojraho` here so that you can use the password of your account to log into `test`.

Task 1: As we discussed during the class, the success of race condition attacks is usually probabilistic. An attacker may need to run the attack multiple times until successfully modifying `/etc/passwd`. Your **first task** is to complete the partially filled shell script `race.sh` that runs `vprog` in a loop until you detect that `/etc/passwd` has been modified.

Task 1: Your **second task** is to complete the partially filled attack program `attack.c`. `attack.c` basically creates two symbolic links using `symlink()` and make them points to `dev/null` and `/etc/passwd` respectively. We need to create a loop to switch between `dev/null` and `/etc/passwd` that `/tmp/XYZ` points to. When `/tmp/XYZ` is linked to `dev/null`, which is writable to anyone, it will pass the `access()` check. When `/tmp/XYZ` is linked to `/etc/passwd`, you can write contents to `/etc/passwd`.

During the attack, you will run `race.sh` and the attack program in parallel. After certain tries, you should observe that `/etc/passwd` has been modified and `test:U6aMy0wojraho:0:0:test:/root:/bin/bash` has been appended to `/etc/passwd`.

Submission Instructions

Submit the completed `race.sh` **and** `attack.c`. Also, submit a report named `q1.pdf` describing the major steps and the reasons for the choices of filled-in values in `race.sh` and `attack.c`. `q1.pdf` should also **include the screenshots** of the successful attack and the contents in the modified `/etc/passwd` file. Place all of them under the Q1 folder of the submission.