1 Overview

The Dirty COW vulnerability is an interesting case of the race condition vulnerability. It existed in the Linux kernel since September 2007, and was discovered and exploited in October 2016. The vulnerability affects all Linux-based operating systems, including Android, and its consequence is very severe: attackers can gain the root privilege by exploiting the vulnerability. The vulnerability resides in the code of copy-onwrite inside Linux kernel. By exploiting this vulnerability, attackers can modify any protected file, even though these files are only readable to them.

The objective of this project is for students to gain the hands-on experience on the Dirty COW attack, understand the race condition vulnerability exploited by the attack, and gain a deeper understanding of the general race condition security problems.

Project environment. Please complete the environment setup as soon as possible. This project will be conducted on SeedLab's Ubuntu 12.04 VM, which can be downloaded from

https://seed.nyc3.cdn.digitaloceanspaces.com/SEEDUbuntu12.04.zip

The download should start autamatically. To estabish the Ubuntu 12.04 VM, you may need VirtualBox, which can be downloaded from

https://www.virtualbox.org/

To configure VM on VirtualBox, you can use the manual

https://github.com/seed-labs/seed-labs/blob/master/manuals/vm/seedvm-manual.md

Make sure that the VM-host and VM-internet connections are broken.

Important Notice: Please take screen shots for every step **after** configuring and starting the VM. For saving fast and easy frequent screen shots, you may utilize programs such as Dropbox. During the Project report, you will need these screenshots to prove each of your actions. Show everything you did to complete each task, but you do not need to mention the failure attempts on the report.

2 Task 1: Modify a Dummy Read-Only File

The objective of this task is to write to a read-only file using the Dirty COW vulnerability.

2.1 Create a Dummy File

You first need to select a target file. Although this file can be any read-only file in the system, you will use a dummy file in this task, so you do not corrupt an important system file in case you make a mistake. Please create a file called zzz in the root directory, change its permission to read-only for normal users, and put your first name (if you have more than one use the first one) into the file using an editor such as gedit.

```
$ sudo touch /zzz
$ sudo chmod 644 /zzz
$ sudo gedit /zzz
$ cat /zzz
```

[YourVeryFirstName]

$ echo [YourSurname] > /zzz

bash: /zzz: Permission denied

From the above experiment, you can see that if you try to write to this file as a normal user, you will fail, because the file is only readable to normal users. However, because of the Dirty COW vulnerability in the system, you can find a way to write to this file. Your objective is to replace the pattern [YourVeryFirstName] with [YourSurname] (again if you have more than one please use the first one).

2.2 Set Up the Threads

You can download the program cow_attack.c from

https://seedsecuritylabs.org/Labs_20.04/Files/Dirty_COW/Labsetup.zip

The program has three threads: the main thread, the write thread, and the madvise thread. You will slightly modify this code for your task. The main thread maps /zzz to memory, finds where the pattern "222222" is, and then creates two threads to exploit the Dirty COW race condition vulnerability in the OS kernel. This code uses a string function called strstr() to find where "222222" is in the mapped memory. It then start two threads: madviseThread and writeThread.

The job of the write thread listed in the following is to replace the string "222222" in the memory with "******". Since the mapped memory is of COW type, this thread alone will only be able to modify the contents in a copy of the mapped memory, which will not cause any change to the underlying /zzz file.

The madvise thread does only one thing: discarding the private copy of the mapped memory, so the page table can point back to the original mapped memory.

2.3 Launch the Attack

If the write() and the madvise() system calls are invoked alternatively, i.e., one is invoked only after the other is finished, the write operation will always be performed on the private copy, and you will never be able to modify the target file. The only way for the attack to succeed is to perform the madvise() system call while the write() system call is still running. You cannot always achieve that, so you need to try many times. As long as the probability is not extremely low, you have a chance. That is why in the threads, you run the two system calls in an infinite loop. Compile the cow attack.c and run it for a few seconds. If your attack is successful, you should be able to see a modified /zzz file. Report your results in the Project report and explain how you are able to achieve that.

$ gcc cow_attack.c -lpthread

$ a.out

... press Ctrl-C after a few seconds ...

3 Task 2: Modify the Password File to Gain the Root Privilege

Now, let's launch the attack on a real system file, so you can gain the root privilege. You choose the /etc/passwd file as your target file. This file is world-readable, but non-root users cannot modify it.

The file contains the user account information, one record for each user. Assume that your user name is seed . The following lines show the records for root and seed:

```
root:x:0:0:root:/root:/bin/bash

seed:x:1000:1000:Seed,123,,:/home/seed:/bin/bash
```

Each of the above record contains seven colon-separated fields. Your interest is on the third field, which specifies the user ID (UID) value assigned to a user. UID is the primary basis for access control in Linux, so this value is critical to security. The root user's UID field contains a special value 0; that is what makes it the superuser, not its name. Any user with UID 0 is treated by the system as root, regardless of what user name he or she has. The seed user's ID is only 1000, so it does not have the root privilege. However, if you can change the value to 0, we can turn it into root.

You will exploit the Dirty COW vulnerability to achieve this goal. In your experiment, you will not use the seed account, because this account is used for most of the experiments in this book; if you forget to change the UID back after the experiment, other experiments will be affected. Instead, you create a new account called charlie, and you will turn this normal user into root using the Dirty COW attack. Adding a new account can be achieved using the adduser command. After the account is created, a new record will be added to /etc/passwd. See the following:

```
$ sudo adduser charlie

 $ cat /etc/passwd | grep Charlie

charlie:x:1001:1001:,,,:/home/charlie:/bin/bash
```

We suggest that you save a copy of the /etc/passwd file, just in case you make a mistake and corrupt this file. An alternative is to take a snapshot of your VM before working on this project, so you can always roll back if the VM got corrupted.

Task: You need to modify the charlie's entry in /etc/passwd, so the third field is changed from 1001 to 0000, essentially turning charlie into a root account. The file is not writable to charlie, but you can use the Dirty COW attack to write to this file. You can modify the cow attack.c program from Task 1 to achieve this goal.

After your attack is successful, if you switch user to charlie, you should be able to see the # sign at the shell prompt, which is an indicator of the root shell. If you run the id command, you should be able to see that you have gained the root privilege.

```
seed@ubuntu$ su Charlie

Passwd:

root@ubuntu# id

uid=0(root) gid=1001(charlie) groups=0(root),1001(charlie)
```

4 Submission

You need to submit a detailed project report until 28th March 23:59, with screenshots, to describe what you have done and what you have observed. You also need to provide explanation to the observations that are interesting or surprising. Please also list the important code snippets followed

by explanation. Simply attaching code without any explanation will not receive credits. There will be a PS session with TA during the week, so that you can ask your questions regarding the project.