Dirty COW Attack Lab

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

### Create a Dummy File zzz

sudo touch /zzz

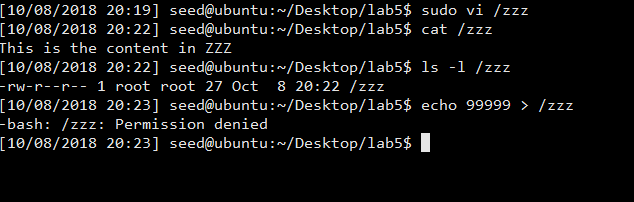
sudo chmod 644 /zzz

sudo vi /zzz

cat /zzz

ls -l /zzz

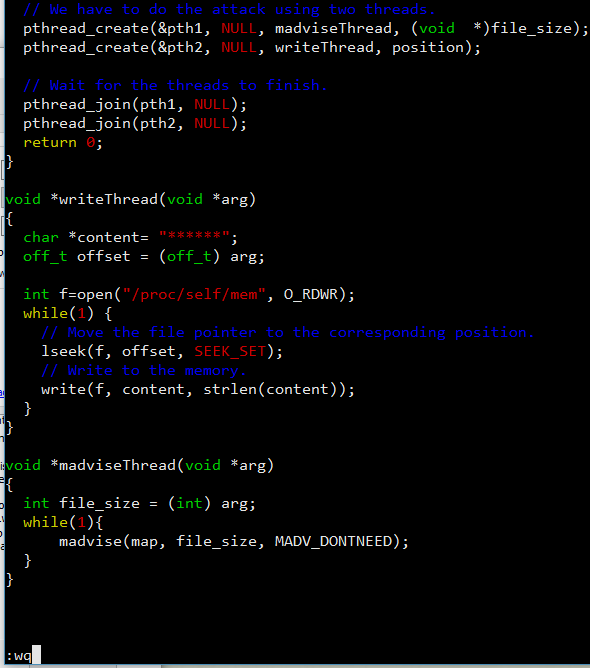
echo 99999 > /zzz



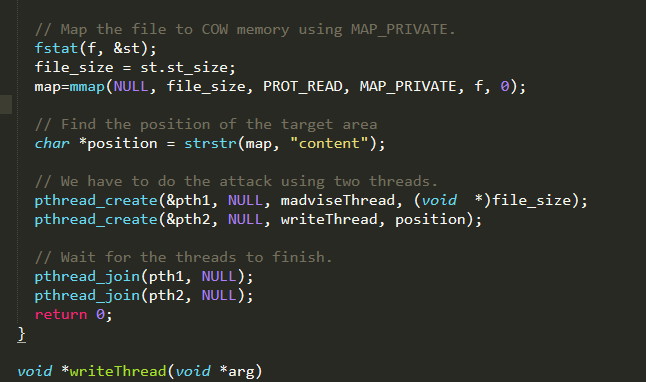
Observation and Explanation:

From above, we can see that the file we just created is only readable to normal users.

### Set Up the Memory Mapping Thread; Set Up the write Thread; The madvise Thread



Copy the file cow\_attack.c into the virtual machine. Since we change the content of zzz to “This is the content in ZZZ”, we can find the substring “content” instead of 22222.

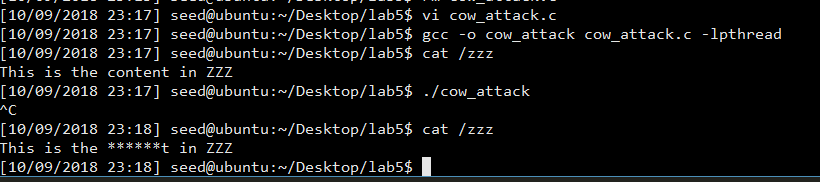


### Launch the Attack

gcc -o cow\_attack cow\_attack.c -lpthread

./cow\_attack (wait a few seconds)

Cat /zzz



**Observation and explanation:**

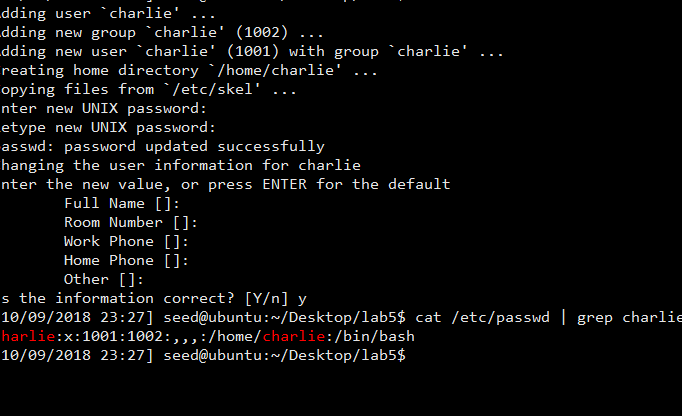
As we can see, the dirty cow attack is being executed successfully. The content in file /zzz is changed to “This is the \*\*\*\*\*\*t in ZZZ”. And we can see that we just use a normal user to modify the file that only readable to normal user “seed”.

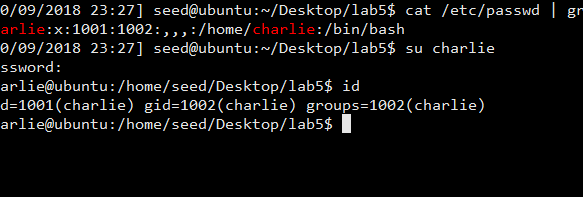
The reason that the attack is successful is that in the two threads we made together, the madvice function won the race condition. The right order should be: 1. Copy the file to another physical location and redirect the pointer to the new location; 2. Do the write operation in the new location; 3 Discard the new copy and remap back to the original location. However, after the madvice function winning the race condition, the order has been changed into 1 3 2. In this case, although it looks like we write to the physical copy of /zzz file, because the map back operation is before write thread, it will write on the original /zzz’s physical address. After finding the location of “content”, the “\*” mark replaced the original contents.

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

Firstly, we need to create an account named Charlie.

sudo adduser Charlie

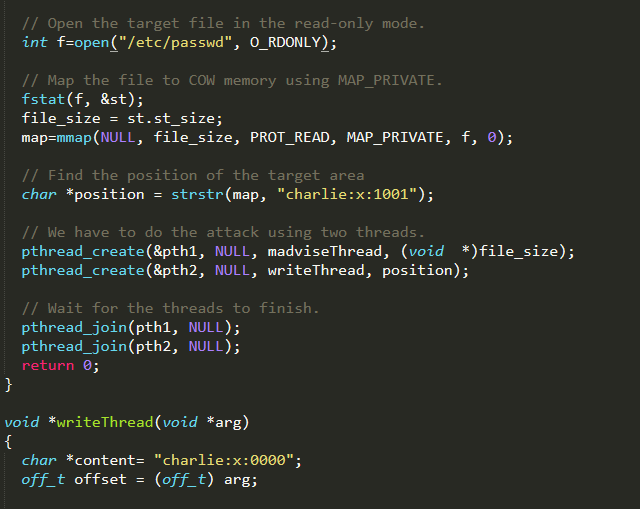




Observation:

As we can see, at this time, new user Charlie is still a normal user.

Then, modify the codes:

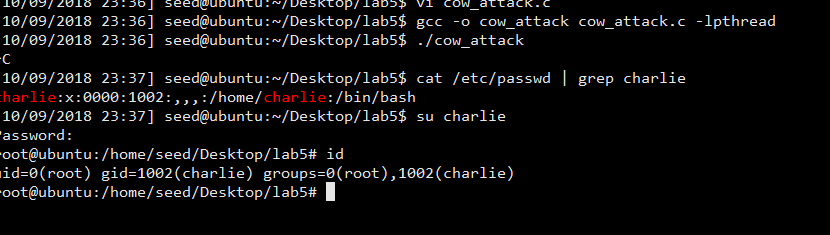


In the position field, we need to find the contents of user Charlie. From the previous screenshots, we can see that the record starts with charlie:x:1001, where 1001 is the user group of normal users. In this section, we need to change it to 0000, the root’s user group, so in the writeThread function we changed the content to 0000. Besides, we need to change the filename opens by the program to “/etc/passwd”.

Finally, do the attack.

*gcc -o cow\_attack cow\_attack.c -lpthread*

*./cow\_attack (wait a few seconds)*



Observation and Explanation:

From both passwd file and charlie’s id information, without using any root privilege operation, we can see that the user Charlie has gained the root privilege.

The reason that the attack is successful is that in the two threads we made together, the madvice function won the race condition. The right order should be: 1. Copy the file to another physical location and redirect the pointer to the new location; 2. Do the write operation in the new location; 3 Discard the new copy and remap back to the original location. However, after the madvice function winning the race condition, the order has been changed into 1 3 2. /etc/passwd is only writable to root. However, because the map back operation is before write thread, the write operation to change the content of user Charlie write not to the copy but to the original location of the file /etc/passwd. After finding the location of “Charlie:x:1001”, the “1001” is replaced by “0000”. As a result, the privilege of the account Charlie has become root.