**1. Introduction**

ITP 30002-02 Operating System, Spring 2020

**Homework 1**

name : EunJong Lee, Student ID : 21700556, Email address : 21700556@handong.edu

The first requirement is that implement function that block file open, when file name contains specific file name and user name who asked file open and specific user name is same.(specific file and specific user name is given from user)

The second requirement is that implement function that protect process from kill instruction, when process’s user and specific user is same.(specific user is given from user)

To implement this two functions, I have to access to kernel and change specific handler code in kernel space. To access kernel space I used loadable kernel module(mousehole). Also, I used proc file system to communicate between lkm(mousehole) and user program(jerry.c).

To implements two functions, I changed handler about open and handler about kill those are in sys\_call\_table which is in kernel space.

To distinguish several data which is given from user program, I used prefix concept.

**2. Approach**

I used loadable kernel module and proc file system. To implement HW1’s function(both block file and block kill), i have to change kernel code in kernel space, and i have to send data from user program(jerry.c) to kernel space. By using loadable kernel module I can add module to kernel space, and change kernel code. By using proc file system, I can send data from user program to lkm.

I used prefix concept to distinguish data. To implement several functions, Ikm need different kind of data. For example, user name and file name. however, proc file system can just read and write, so when data is given to lkm, the method that distinguish data is needed. To solve this problem I used prefix concept. When data is given to lkm, check first bit of data, and distinguish data. For example, if first bit is 2, it is data about specific file name.

To transfer kernel information to user program, I used only one buffer(printbuffer). Depending on prefix of input, the buffer contains other contents, and user program read the data.

To implement function 1(file open block), I changed NR\_OPEN handler in sys\_call\_table. my NR\_OPEN handler logic is that if file name contain specific file name and uid who is ask file open and specific user’s id is same, return -1. Else, call original NR\_OPEN handler. To find uid who is ask file open, I used cred information.

To impement function 2(kill block) I changed NR\_KILL handler in sys\_call\_table. In my NR\_KILL handler, I used for\_each\_process function to access the list of all processes. When there exist process that pid in process list == pid that user(who call kill instruction) want to kill, and user id who call kill instruction == specific user, return -1. Else, call original NR\_KILL handler. To find pid that user want to kill, I used cred information.

In jerry.c

jerry.c code logic is simple. jerry.c get inputs from user(for example, specific file name and specific user name). Then add prefix to inputs to distinguish its identity. open /proc/mousehole and write user inputs to mousehole module. And then read kernel information from mousehole module.

The special point is that specific user name input given as string rather than int(uid). To change user name(string) to uid(int), I needed api that is executed in jerry.c, but execute command(id -u ~) in another process(not in jerry.c) and jerry can read the result value. The api that I used is pipe. I give command(id -u ~) as parameter to popen function and just read result in pipe.

**3. Evaluation**

**Function 1(file open block) evaluation**

The criteria is that check file is open or not open in two cases.

Case1. File open block is enabled(specific file name : mouse, specific user name : tom)

Case2. File open block is disabled.

To check

1. change user by command “su tom”.
2. open file which name is mouseInfo.
3. check file is open normally.

When open block is disabled, file is open normally.

If open block is enabled and file is not open, first function is accomplished successfully.

In my case, in case1, file is not opened.

In my case, in case2, file is opened.

So, I accomplished function 2 successfully.

**Function2(kill block) evaluation**

The criteria is that check process is killed or not killed in two cases.

Case 1. Kill block is enabled(specific user name : tom)

Case 2. Kill block is disabled.

To check

1. change user by command “su tom”.
2. Tom execute program that wait input(process is not terminated until user give input).
3. Open another terminal that user is bell.
4. Get pid of tom’s program which is wait input.
5. Bell execute Kill-9 command about tom’s process.

When kill block is disabled, process is killed.

If kill block is enabled and process is not killed, function2 is accomplished successfully.

In my case, in case1, process is not killed.

In my case, in case2, process is killed.

So, I accomplished function 2 successfully.

**4. Discussion**

One of the definite signs of having an effective knowledge is to apply it to the topics beyond the given material. This section is to present your ideas, thoughts, lessons learned, suggestions, opinions that you have conceived while working on the given tasks.

I experienced terminal is stopped many time, while doing homework. I learned many things from this situation. I learned that if lkm change kernel code and lkm removed without return original kernel code, error is occurred. This made me wonder how kernel protect itself. Also, I learned why separating kernel space and user space is needed.

When I check file open block works well, I think cd(change directory) is also related with open handler, but file open block is not applied when I change directory. I know that directory is similar with file, so it is strange to me. I wondered how to block change directory.

**5. Conclusion**

In this homework, I used proc file system to communicate with kernel space and user space. By using loadable kernel module, I can change code in kernel space. And I changed syscall handler code to implement homework’s function. I also used pipe to get uid. By using pipe I understand more about pipe concept.

While I do homework, I understand a lot about system call mechanism that when user program asked the kernel to do things that should be handled in the kernel space, how it was processed and come to user program.