**1. Introduction**

ITP 30002-XX Operating System, Spring 2020

**Homework 1**

Park Yegyeom / 21600277 / 21600277@handong.edu

To carry out this task, the mousehole LKM root kit and the user application jerry.c had to be implemented. To implement the Mousehole, LKM had to study how LKM was placed on the kernel and learned about the file system called /proc/mousehole provided by the mousehole and how to intercept the system call as my own system call. It was necessary to learn how the information entered through jerry into the mousehole. And learning and thinking about how to handle the entered values was also needed. After learning these things, the mousehole intercepted two system calls (sys\_open, sys\_kill) and defined the corresponding system calls as the mousehole's own system calls. And for the file operation of the mousehole, the mousehole was implemented to take appropriate action against orders and information from Jerry using the mousehole\_write and mousehole\_read. Next, jerry decided how users would use jerry in the form of API. And user thought about what form of order and information to enter in jerry and implemented it so that it works well with the mousehole.

**2. Approach**

2.1 Ovarall Structure

As mentioned in the introduction, the mousehole.c and jerry.c must be implemented to carry out this task. mousehole.c is the LKM Root kit to be put on the kernel and jerry is the user program. Users who use this program can enter uname and fname into /proc/mousehole file system through jerry or read its value and control the mousehole function through command. And the mousehole intercepts sys\_open and sys\_kill system call so that its function works normally.

2.1 Proc Interface

I think one of the first things to organize in order to accomplish this task is proc interface, so I will explain this first. The Proc Virtual file system is files act as agents of a kernel data structure or kernel module to interact with user-level program. It provides the internal kernel information to users as form of file. When the module is loaded(insmod), the proc interface is created as a file system (/proc). This can be red and written by users and the kernel. It can work as a communication bridge between jerry and mousehole.

2.2 How to communicate with the mousehole through jerry

Jerry is a user program that users use directly. When a user enters username in jerry, jerry finds uid in username and enters the uid in /proc/mousehole. The task aims to implement two functions: the first function is to prevent users from accessing the files that contain the file names when they are entered through jerry. This feature can be used by the user as follows.

./jerry “1 username” “filename”

When the user enters it, the mousehole recognizes it as the first function and replaces the values passed through jerry in the ban\_uid and ban\_fname. If this function is enabled, the user will not be able to access the file containing the filename. For example, if a user enters

./jerry "1 host" "temp,"

a user named ‘host’ is blocked when accessing a file containing a filename called ‘temp’.

To restore this function, enter

./jerry "R-1"

The second function is to receive the user name through jerry, and the process of the user is protected by another process killing the user's process. This feature can be used by the user as follows.

./jerry “2 username”

If the user enters this way, the mousehole recognizes the second function and stores the corresponding uid in the protect\_uid. The process of the uid is protected until this function is released.

To restore this function, enter

./jerry "R-2"

Finally, I'll write down a brief description of the other jerry’s command.

./jerry S : Currently, ban\_id, ban\_fname, protect\_uid can be checked.

./jerry , ./jerry -h : show jerry usage

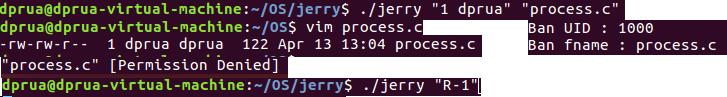
2.3 Operation of the mousehole LKM Root Kit

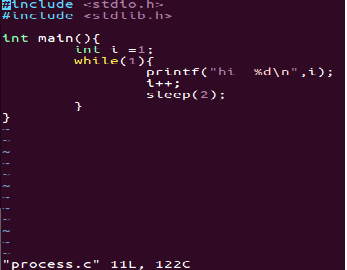
The mousehole plays a practical role in realizing the function of the task. First, the existing sys\_open should be intercepted as mousehole\_sys\_open. In addition, every time sys\_open is called, the mousehole\_sys\_open can be called to implement function1 by blocking this system call if the value of current-cred-uid.val and bau\_uid are the same.

To implement the following functions, the existing sys\_kill must be intercepted as mousehole\_sys\_kill.

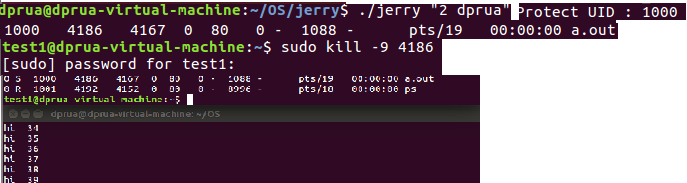
Thus every time sys\_kill is called, the mousehole\_sys\_kill be called to implement function2 by preventing the kill of the process if the current task is equal to the protect\_uid value and the t-cred-uid.val. (struct task\_struct \* t)

**3. Evaluation**

After loading the mousehole into the kernel with "sudo insmod mousehole.ko", enter uname and fname through jerry and determine whether a particular user blocks access to a specific file.



The above results confirm that function1 is operating normally and restore is also operating normally.

Function2 identifies the operation of the function based on the determination of whether the user's process is killed or not.

As a result of the above, we can confirm that the function2 is operating normally and the restore is also operating normally.

**4. Discussion**

Idea: The final result is implemented by receiving input from the user and storing string-type variables in the proc file system through file descriptor and open and write functions. However, the initial idea was to save the structure in the form of a structure with the value input from the user, but failed. If there is a source code implemented in the form of a structure, I would like to refer to it.

Lesson learned: I haven't had a chance to experience Linux. Thus, I spent a lot of time doing this assignment to get familiar with Linux. As a result, I got to know a lot of Linux command lines, and it became a chance to become familiar with Linux by compiling and executing C source files with CLI. And I was able to experience a lot of searching for what I needed through elixir.bootlin.com and Googleing, which was really rewarding.

**5. Conclusion**

This task was to create a mousehole.ko, which is toy rootkit, load my own LKM into the kernel, and implement two functions through communication with user-level program jerry.c and CLI. To implement these two functions, it was necessary to understand the proc-file system, file descriptor, function(open,read,write) and place system call handler, and through the process of learning and organizing these concepts, the two functions were successfully implemented..