Jennifer Finaldi

CSC415.02

Assignment6

12/13/2020

# Device Driver: A Dongle Simulator

## Introduction

This program is intended to simulate the behavior of a 2-factor authentication dongle, similar to keychain dongle hardware that generates a random login key to be entered within a limited time frame. For this particular driver, since this is a software-based character device driver, only the behaviors of using the dongle will be implemented, rather than actually using hardware input/output. To describe what it does, the user program first outputs instructions to the user, then prompts the user to enter code '668' to generate a random key. This key is then generated by the module, which subsequently starts a three-minute timer in which the key will be valid. If the user enters the wrong key, the program will simply keep asking for the correct key. If the user enters the correct key but after the timer is up, they will be notified of the key expiration and prompted to generate another key. If the user enters the correct key within the correct time frame, they will be "logged in" and the program ends.

## Solving the Problem

In order to implement these features, the driver module needed to be created first. Some sample code provided from Derek Molloy helped as a starter template to which I adapted to suit this program. This code can be found at: http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device/ . The init function of this module is mostly left the same, however I did add some initiation of module variables, and also initialized the timer. File operations

functions were created for open, release, read, write, and ioctl. The open and release functions do nothing more than output to kernel logs that they were indeed used. The read function is responsible for taking a value stored in the "key" variable in the module, and writing it to the user's buffer, if there is one. The write function takes a user passed key and writes it to the "userEnteredKey" variable stored in the module. This function has error handling that will prevent an invalid value from being assigned to this variable. The ioctl function handles behaviors related to processing. As such, it contains a switch statement to analyze which code was input, calling helper functions to execute the code it pertains to. For example, if the user-program passes '668,' the switch will call the helper function to generate a random key. If '667' is passed, then the module will compare the key with the user entered key (if they exist), and then return a code for if the comparison was a match or not a match.

Some helper functions were also added to the module. A function to generate a random key operates by using the get_random_bytes() function from the kernel, that returns random bytes (as the name suggests). This raw key generated is then analyzed to find out if it is too large of a number or if it is a negative value. Since we don't want negative values for a key, the sign is changed if it is negative. If the number is larger than the max value allowed, it will be divided to make it smaller.

Another function to start the timer operates by first setting (or resetting) to zero, a boolean variable that flags if the timer is up. It then calls a timer function that starts a timer, which will call another callback helper function once the timer gets to 180,000 milliseconds, aka 3 minutes. When this callback function is executed it flags the timer boolean flag to true and then exits. Another function to validate a key operates by checking if the timer flag is set to true. If false, it checks to make sure we have valid variables for key and userEnteredKey, then compares

them. It returns an integer result back to the caller: 1 for correct key entered, 2 for time up, 0 for incorrect key. The value 2 is so that if it is sent back to ioctl, then it can pass the 2 back to the user space, so that the appropriate error handling can be performed.

The user program is merely a simple program that loads the device driver, gets user input, and then passes this input to and from the linux kernel module (LKM) to be processed. A main loop exists that will prompt a user to enter '668' to generate a key. And inner loop will operate to get a valid code (in this case, which is only 668), which won't let the program progress until it is entered. Then the program will call ioctl() to generate the code. Once the code is generated without errors, the user program then calls read(), to read the newly generated key into the user-space. This key is then output to the user in the console, followed by prompting the user to enter that same key within a 3 minute time frame. Another loop begins that repeats until either the user key expires, or they enter the correct key. If the user's key expires, the program prompts the user if they would like to generate another. If the user chooses anything other than 'Y', the program will escape the loop and end. If they choose yes, the main loop will repeat, asking for the user to enter '668' to generate a code.

## Issues that Came Up

There were a few significant issues that came up. The first was getting used to kernel space syntax, and what libraries could or could not be used while writing an LKM. At first I tinkered with the parameters of ioctl and write, which produced many errors, as the parameters for these functions have to exactly match the file operations declarations. The original template code did not include the ioctl function, so adding that in also proved to be a challenge, because it was difficult to figure out exactly what features I should put in there, rather than inside read/write. When attempting my first compiling of the module, a lot of errors popped up, many

of which were related to the random number generation method I was trying to use, and the timer method. For random number generation I thought I was going to be able to use the standard template library that includes rand() and random() but those were not able to be used in kernel space. After doing extensive research I decided to go with the library linux/random.h, so I could make use of the kernel function called get_random_bytes(). As for the timer, I was trying to use time.h, with its timestamp features that would subtract two time stamps to see if three minutes had passed. This, too, was unable to be used in kernel space, so I had to resort to the linux/timer.h library. A function in this library allows you to initialize a timer that seems to run asynchronously during program execution, issuing a callback once the specified time interval has expired. This ended up being even more convenient to use, despite a lot of research required to find out why set_timer() initializing was failing. Apparently in newer versions of Linux, this init function has been changed to timer_set().

Another silly issue that came up was the module license. I mistakenly thought that we could use any made-up code for a module license and it took me some moments of confusion before I found out that we need to use "GPL" in order for it to compile properly. I also had compiler errors that came up whenever I tried to declare and initialize a variable inside a helper function, or fops function. This was remedied by declaring them all global along with other variables. One more significant setback I had was figuring out why my open function was not being called after installing the module successfully and running the user program successfully. In the user program, the open() function was returning -1 and I could not figure out why at first. Then I realized that I was using the "make run" command to build and run my user program, when I should have been using "sudo ./dongle_user".

Screenshots

This demonstrates the compilation and installation of the LKM


(Scroll to next page for more output)

Now we compile and run the user program

```
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignm
ent-6-device-driver-jfinaldi$ cd User
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignm
ent-6-device-driver-jfinaldi/User$ make
gcc -c -o dongle_user.o dongle_user.c -g -I.
gcc -o dongle_user dongle_user.o -g -I.
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignm
ent-6-device-driver-jfinaldi/User$ sudo ./dongle_user
Opening dongle module
dongle module opened
-----------------------------------------
Welcome to the Dongle App

This program simulates 2factor authentication
by generating a random 7-digit key for the
user to enter within 3 minutes of being
generated, in order to 'log-in' to a service.

To use,enter the following code when prompted
-----------------------------------------

Enter 668 to generate a code: 667

Sorry, you did not enter the correct code
Enter 668 to generate a code: 668
Your key: 20121507
You have less than 3 minutes to enter this key.

Enter your key: 34
Error, write failed. Invalid Input Format

Sorry, you did not enter the correct key
Enter your key: 20121507

Sorry, this key has expired.
Would you like to generate another?[Y/N]: y

Enter 668 to generate a code: 668
Your key: 10074727
You have less than 3 minutes to enter this key.
```

```
Enter 668 to generate a code: 668
Your key: 10074727
You have less than 3 minutes to enter this key.

Enter your key: 10074727
Congratulations, you are now logged in

Thank you for using this dongle tool. Goodbye!

student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignm
-device-driver-jfinaldi/User$
```

Now, the kernel logs

```
Dec 13 13:34:33 student-VirtualBox kernel: [ 6178.441005] Dongle: Initializing Dongle Linux Kernel Module
Dec 13 13:34:33 student-VirtualBox kernel: [ 6178.441008] Dongle major number successfuly registered with 240
Dec 13 13:34:33 student-VirtualBox kernel: [ 6178.441015] Dongle class successfully registered
Dec 13 13:34:33 student-VirtualBox kernel: [ 6178.441147] Dongle driver successfully registered
Dec 13 13:34:33 student-VirtualBox kernel: [ 6178.441148] Dongle device successfully initialized
Dec 13 13:36:39 student-VirtualBox kernel: [ 6304.742109] Dongle successfully opened
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875951] Inside dgl_ioctl
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875952] dgl_ioctl: case 668: about to call dgl_generate
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875952] dgl_generate: creating key..
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875957] dgl_generate: key 20121507 created
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875958] dgl_start_timer: timer
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875959] dgl_start_timer: timer successfully started
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875961] Inside dgl_read
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875962] dgl_read: key = 20121507
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875963] dgl_read: message = 20121507
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875964] dgl_read: sizeOfMessage = 8
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875965] dgl_read: Sent 8 characters to the user
Dec 13 13:37:05 student-VirtualBox kernel: [ 6330.875965] dgl_read: sizeOfMessage reset to 0
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314930] Inside dgl_write
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314932] dgl_write: userEnteredKey: 34
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314932] dgl_write: User entered invalid key format
```

```
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314932] dgl_write: User entered invalid key format
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314950] Inside dgl_ioctl
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314951] dgl_ioctl: case 667: validate user key
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314951] Trying to validate keys with one or both missing
Dec 13 13:37:18 student-VirtualBox kernel: [ 6343.314953] dgl_ioctl: case 667: res = 0
Dec 13 13:40:18 student-VirtualBox kernel: [ 6523.732012] dgl_time_up: the key has expired
Dec 13 13:40:18 student-VirtualBox kernel: [ 6523.732037] dgl_time_up: isTimerUp = 1
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124574] Inside dgl_write
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124575] dgl_write: userEnteredKey: 20121507
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124577] Inside dgl_ioctl
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124578] dgl_ioctl: case 667: validate user key
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124578] dgl_is_correct_key: key = 20121507
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124579] dgl_is_correct_key: userEnteredKey = 20121507
Dec 13 13:42:46 student-VirtualBox kernel: [ 6672.124579] dgl_ioctl: case 667: res = 2
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500392] Inside dgl_ioctl
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500393] dgl_ioctl: case 668: about to call dgl_generate
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500394] dgl_generate: creating key..
```

```
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500394] dgl_generate: creating key..
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500469] dgl_generate: key 10074727 created
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500469] dgl_start_timer: timer
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500470] dgl_start_timer: timer successfully started
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500473] Inside dgl_read
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500473] dgl_read: key = 10074727
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500475] dgl_read: message = 10074727
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500475] dgl_read: sizeOfMessage = 8
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500476] dgl_read: Sent 8 characters to the user
Dec 13 13:42:53 student-VirtualBox kernel: [ 6678.500476] dgl_read: sizeOfMessage reset to 0
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507174] Inside dgl_write
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507175] dgl_write: userEnteredKey: 10074727
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507177] Inside dgl_ioctl
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507178] dgl_ioctl: case 667: validate user key
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507178] dgl_is_correct_key: key = 10074727
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507179] dgl_is_correct_key: userEnteredKey = 10074727
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignment-6-device-driver-jfinaldi/User$
```

Now, removing the dongle module

```
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignment-6-device-driver-jfinaldi/User$ cd ..
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignment-6-device-driver-jfinaldi$ sudo rmmod dongle.ko
student@student-VirtualBox:~/Desktop/CSC415/asmt06/assignment-6-device-driver-jfinaldi$ cd /dev
student@student-VirtualBox:/dev$ ls -l don*
ls: cannot access 'don*': No such file or directory
student@student-VirtualBox:/dev$ sudo tail -f /var/log/kern.log
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507175] dgl_write: userEnteredKey: 10074727
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507177] Inside dgl_ioctl
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507178] dgl_ioctl: case 667: validate user key
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507178] dgl_is_correct_key: key = 10074727
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507179] dgl_is_correct_key: userEnteredKey = 10074727
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507179] dgl_ioctl: case 667: res = 1
Dec 13 13:43:03 student-VirtualBox kernel: [ 6688.507266] Dongle successfully closed
Dec 13 13:46:02 student-VirtualBox kernel: [ 6867.785771] dgl_time_up: the key has expired
Dec 13 13:46:02 student-VirtualBox kernel: [ 6867.785794] dgl_time_up: isTimerUp = 1
Dec 13 13:59:04 student-VirtualBox kernel: [ 7649.225356] Dongle successfully removed from the system
```