PROJECT 3

Lance Umagat

Project 3 Write Up

Prepare a document containing the design you plan to use to implement the necessary algorithms. Since the sequence of function calls goes from init which calls request and request calls transfer, it made the process pretty simple. I decided to initialize the device and crypto in the initialization. The init function would call the request to get requests. The request would call the transfer function with the the request data stored in the buffer, whether it is a read or write, the starting offset sector, the amount of bytes sectors needed to be read/write and the device itself. The transfer would encrypt or decrypt depending on the read/write parameter and that's where I set the key and using that key to encrypt/decrypt the data. It would encrypt/decrypt every sector in the data until it is completely encrypt/decrypt as well as transfering the data from buffer to device driver and vice versa. Then the transfer function waits to be called by request again and then it should be request/transfer calls.

Answer the following questions in sufficient detail

1. What do you think the main point of this assignment is? To learn how to implement a device driver as a module within the Linux kernel. To learn how to use different commands from the Linux kernel and understand the kernel's structure a little better. As an example I didn't realize my device was inside the dev directory until I went to go search for it with the find command. To learn there are multiple ways of implementing encryption. There are multiple algorithms for this task as well as multiple variation on what function calls to use and when such as using a plain cipher to get each individual block or use another cipher that gets a continguous block all at once, but you must implement a scatterlist which was confusing to me. To learn how to transfer data from a device disk to a buffer with crypto cipher function calls and vice versa.

2. How did you personally approach the problem? Design decisions, alogorithm, etc. This was a tough assignment to me and I had to mess around with Pat Patterson's code to figure out what was going on with print statements. Then I searched up all of the function calls and where they were coming from. I wrote all of these function calls down and then I figured out how to manipulate the transfer function as well as see the sequence of execution of functions in the Linux kernel. There was no algorithms for this as this was bare bones programming. I did plenty of research on stackoverflow to figure out encryption and had many failure and kernel panics. Then I found out how to read and write from device drivers which utiliizes the buffer. It was just a simple transfer as if they were variables where you can transfer with memcpy, but I founded out that a crypto call does all that for me and it was great to both simplify my code as well as seeing it working. Then I learned how to use file I/O to read and write data out to a file to see if it was doing anything which it was and the file was a binary file which is to

be expected. After hours of research I managed to initialize the device which is by registering which I used from the base code then use the device driver to transfer data to the buffer for read and vice versa for writing.

3. How did you ensure your solution was correct? Testing details, for instance. I used plenty of print statements to see my process and it came out as I expected which is initialize one time then get requests then transfer. The cycle goes on and on. If there's no pending requests then the request just waits for more requests. I tested the encryption by passing in my key and key length and seeing if my encryption sets the key or not. If it doesn't with a key and key length that I know would not work then it's correct which I can verify using print statements to see if my set key crypto function call fails. The next part is to see if there's anything on the device driver. There should be data on the device driver, so I used file I/O to input data into a file and then I check that file inside my Linux kernel to see if it was empty or not. I had to use less to even look at the file and since it was a binary file I saw plenty of data which I believe was good since I did initialize the device driver to all zeros.

4. What did you learn? I learned plenty of useful information and more than plenty unuseful information that I may or may not use in the future. I learned how to boot my linux kernel and scp my device driver kernel object into my virtual machine. I also learned how to run and remove my module from reading some documentation provided by Kevin Mcgrath. I learned how to use a few different methods for encryption which only one worked for me and the others gave me a kernel panic. All of which I lived on stackoverflow for. I learned how to use basic file I/O from stackoverflow as well to get data from the device driver. I learned how to make a file system using a command. I also learned how to manipulate data with device drivers. I learned how to use file I/O in within the kernel which was not too bad.

Version control log (formatted as a table) – there are any number of tools for generating a TeX table from repo logs

acronym	meaning
V	`version`
tag	`git tag`
MF	Number of `modified files`.
AL	Number of `added lines`.
DL	Number of `deleted lines`.

V	tag	date	commit message	MF	AL	DL
1		2016-5-16	Added the sbd files and modifed the drivers	2	374	0

Work log. What was done when?

1) 05/12/2016 Started playing around with simple block code
2) 05/12/2016 Added encryption to code

3) 05/12/2016 Finished encryption
4) 05/15/2016 Testing device and learned to make filesystem
5) 05/16/2016 I add, commit, and pushed all work to the repository.