Operating Systems II HW3 Writeup

May 27, 2018

Prepared for

Dr. Kevin McGrath

Prepared by

Max Moulds
Monica Sek
Nicholas Wong

Abstract

This document is the writeup for homework 3 of Operating Systems II Spring term 2018, written by Group 20.

Contents

1 Design Plan

For this assignment we were tasked with writing a RAM Disk device driver for the Linux Kernel which should allocate a chunk of memory and present it as a block device. This requires the use of the Linux Kernel's Crypto API to add encryption to the created block device such that the device driver encrypts and decrypts data when it is written and read.

Having created a filesystem in memory before we began from that shared knowledge by researching the difference between what was done before and what is being asked in this assignment [1]. Using the logical progression from creating an unecrypted block device residing entirely in memory first and then testing to then adding a level of encryption to that device. The last element, encryption, is where the majority of the work and research was needed.

We used the example block device given in the assignment description and extended encryption to operations that use that device. We included the crypto header, set a 16 digit base secret key and compiled our block device a module in a similar manner to assignment 2.

Next we had to implement a cipher at a low level. In the simple block device code when a write or read was required we changed the unencrypted function logic to support our encryption scheme using the crypto api. This created a transformation of the data being written or read. We used a call to the crypto library function crypto_cipher_encrypt_one for writes crypto_cipher_decrypt_one for reads.

This was built out and tested using printk statements to check the changes in situ. Further explanation of the testing and verification processes used is described in the following sections.

2 Version Control Log

Author	Date	Message
nickywongdong	2018-05-23	uploading assignment description
nickywongdong	2018-05-23	downloading clean linux image
nickywongdong	2018-05-23	uploading initial related files
nickywongdong	2018-05-23	updating readme unordered list
nickywongdong	2018-05-23	updating ordered sublist items
nickywongdong	2018-05-23	updating readme again
nickywongdong	2018-05-23	once again updating readme
nickywongdong	2018-05-23	adding initial simple block device from example online source
nickywongdong	2018-05-23	testing initial simple block device without adding crypto
nickywongdong	2018-05-23	adding crypto parts to encrypt/decrypt data
nickywongdong	2018-05-23	testing some more crypto gegeneration
nickywongdong	2018-05-24	adding specific qemu command used for easy access
nickywongdong	2018-05-24	updating sbd to one hopefully compatible with kernel
nickywongdong	2018-05-24	testing with same crypto from last version of sbd.c
nickywongdong	2018-05-24	adding req-¿buffer to use bio_data() found in swim.c – Must be update for kernel version
nickywongdong	2018-05-24	successful compilation, plan to run with qemu to test later

nickywongdong	2018-05-24	updating printk statements for correctness
nickywongdong	2018-05-24	changing module author, and running ac1_open script for linux kernel, and hw3 directory for teammates
nickywongdong	2018-05-24	adding command used for scp, and organized previous commands
nickywongdong	2018-05-24	adding commands used within qemu vitual machine to test kernel module
nickywongdong	2018-05-25	adding patch file
nickywongdong	2018-05-25	updating patch
nickywongdong	2018-05-25	adding block device in Kconfig
nickywongdong	2018-05-25	updating patch
nickywongdong	2018-05-25	updating patch
nickywongdong	2018-05-25	finalizing

3 WORK LOG

May 23	Began assignment by researching from the assignment description, and pulling examples from online sources
May 24	Began development of crypto portion of the assigment
May 25	Finalizing assignment with patch file
May 27	Write up and review

4 ADDITIONAL QUESTIONS

1) What do you think the main point of the assignment is?

   In the context of operating systems and the components that provide upper level functionality this assignment requires familiarization with Linux block devices and crypto apis within a kernel. Within the context of development of a low level system component this assignment teaches one to deal with external documentation and using other external libraries to create a set of required functionalities. In the context of software development this is useful because it teaches one to develop in a way that others can use it. While this last point may be a stretch it is undoubtedly important if one plans to develop software used by others.

2) How did you personally approach the problem? Design the decisions, algorithm, etc.

   As mentioned previously the assignment was approached by building off previous knowledge gained from both previous assignments and experience outside of the class. We also relied on advice from the professor and from discussion with others familiar with the assignment goals.

3) How did you ensure your solution was correct? Testing details, for instance.

   To ensure correctness of our solution, we first made sure that the module was compiled and the .ko file existed after the make command. This file should be located in driver/block/ directory within the Yocto root directory. We then started up the virtual machine with the command similar to the one in the previous assignment, but this time removing the flag -net:none. This will allow us to scp the .ko file from our engr group directory into localhost. We then loaded the mod with insmod. The block device then showed up under the /dev directory and it was capable of being mounted to the system with mkfs.ext2. This caused the initialization to run within the module and demonstrate writing/reading with an additional crypto encryption/encryption process.

4) What did you learn?

   While continuing to cement our understanding and comfort with developing Yocto kernel components and procedures we extended our general understanding of low level system components and their interaction with other kernel subsystems. This is most evident in the interaction between the kernel module we implemented to provide the block device and the corresponding functions used within that code that in turn used the already existing API to provide our needed cryptographic functionality.
   Implementation alone continued lessons begun in past assignments, this assignment brought about another element that forced us to learn. Testing and verifying our solution was not as straightforward as using the implemented changes and inspecting the output and led to greater understanding of development. Sometimes verification of a solution is tougher processes than the finding the initial solution itself. Additionally, cryptography has a higher degree of required verification since ramifications from a failure in a cryptographic implementation carry different ramifications from those in the past two assignments which mainly impact system performance relating to physical device such as a hard drive and the physical or operational damages from improper implementation. The impacts from improperly verified cryptographic functionality is complete loss of the functionality and intended service. That is a crytpographic integrity is either complete or non-existant whereas a poorly implemented seek algorithm for a hard drive device can still provide some level of service.

5) How should the TA evaluate your work? Provide detailed steps to prove correctness

   Start with the 3.19.2 Yocto kernel source and our project submission and enter into that submission directory. Use the patch file given with this write up or located at

```
https://github.com/nickywongdong/CS444_GROUP20/blob/master/hw3/patch/hw3.patchc file in
    / the block directory within the Yocto source.
```

Applying the patch and compiling the kernel image

First, please apply the patch supplied to the v3.19.2 branch of the Yocto Kernel, and while in that directory check that patch applied by looking for the simple device block driver.
Once compilation has completed, there should be a kernel image file in the x86 directory and subdirectory boot. Please start the VM using the following command changed to reflect booting from that image:

```
> qemu-system-i386 -gdb tcp::5520 -S -nographic -kernel linux-yocto-3.19.2/arch/x86/boot/
    / bzImage -drive file=core-image-lsb-sdk-qemux86.ext4,if=virtio -enable-kvm -usb
    / -localtime --no-reboot --append "root=/dev/vda rw console=ttyS0 debug"
```

Note that if the filesytem container **core-image-lsb-sdk-qemux86.ext4** is not located in the parent directory of the Yocto kernel, change either the command to reference the filesystem container file or move that file to the parent directory.

And that concludes our writeup for HW3. Thank you.

5 REFERENCES

REFERENCES

[1] J. Coyle. (2017) The Difference Between a tmpfs and ramfs RAM Disk. [Online]. Available: https://www.jamescoyle.net/knowledge/951-the-difference-between-a-tmpfs-and-ramfs-ram-disk