Group 15 Assignment 3 Report

Ryan Howerton, Jinfeng Li, Vincenzo Piscitello
CS444 Spring 2018

Abstract

In the third assignment for Operating Systems II, we created a RAM Disk Driver which encrypts data as it is written and decrypts data when it is read. We accomplished this using the Linux Kernel Crypto API, and a sample simple block driver (SBD) found online, provided by the author of the book Linux Device Drivers (third edition). Our group studied both the Crypto API and the SBD to fully understand how to use them. For the cryptography portion, we decided to use the AES scheme with a randomly generated key (which we deemed usable considering that this driver should only ever be used on an expendable virtual machine). We wrote out our design, and created a detailed version control log and work log that keep track of our code edits and work time, respectively. Finally, we answer a few reflection questions on the work done.

I. Design

We used a simple block driver file found online as a base for our project, and dissected it to determine where to insert kernel cryptography commands to encrypt data as it is written, and decrypt data as it is read. To do this, we needed to create a crypto object, define a key, and initialize the object with a scheme. Finally, we decided that since this would be operating entirely on a virtual machine which is crushed and recreated each time, we would randomly generate a key using get_random_bytes() and storing it in a char array.

II. Version Control Log

Commits
commit c2baed612cfa51aeca64677eda5bb3fd8e7b21a4 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 24 17:30:00 2018 -0700 Changed driver filename and fixed buffer bug
commit fc1ffc0a6491c531d285196c3e91ba44ee04b09d **Author:** Ryan Howerton <howertor@oregonstate.edu > **Date:** Sun May 17 18:00:00 2018 -0700 Finished sbd.c
commit a29b80f7b56515b550e841414cdc3efb725fc258 **Author:** Ryan Howerton <howertor@oregonstate.edu > **Date:** Sun May 17 16:30:00 2018 -0700 Updated sbd.c
commit f8c90132c9c83186997f1663b843389521cb5e52 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 16 16:26:00 2018 -0700 Added Makefile alterations
commit 448eb57c7712b69e92a70109e3b1c22bbcd20775 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 16 15:45:00 2018 -0700 Added the crypto library include
commit 388d73054f44d3da3028baa01e16fabd087b7ca2 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sun May 16 15:12:00 2018 -0700 Added base SDB.c
commit 1222d7e6c7804d9a7232654df3906500f653c2b8 **Author:** Vincenzo Piscitello <piscitev@os2.engr.oregonstate.edu > **Date:** Sat Apr 14 15:36:44 2018 -0700 Initial Commit
commit 660613d1a4e94144490850b6c3d350331860fac4 **Author:** Greg Kroah-Hartman <gregkh@linuxfoundation.org > **Date:** Wed Mar 18 14:11:52 2015 +0100 Linux 3.19.2

III. Work Log

Wednesday, May 16th	Downloaded the simple block device source code sbd.c. Added initialization code of crypto for memory encryption. Also, found out the algrothim used to encryption. Uploaded sbd.c to Github.
Thursday, May 24th	Finished the encryption & description parts and compiled sbd.c as a module in the VM. Found buffer bugs and fixed it.
Friday, May 25th	Recompiled sbd.c and tested it in the VM make sure no error generated. Started writing parts.
Sunday, May 27th	Finished writing work. Packaged everything up and submitted.

IV. Questions

1) What do you think the main point of this assignment is?
   - The main point of the assignment is to practice using a poorly documented API and implement it in an established project.
2) How did you personally approach the problem? Design decisions, algorithm, etc.
   - We began by looking into anything related to the provided search terms (SBD and 3.X). We found the simple block driver provided by the author of LDD3, and cross-referenced the Linux Kernel API to understand its operation. We then researched into the Kernel Crypto API to find applicable commands. Finally, we put everything together, added to the sbd, and tested it to make sure it worked.
3) How did you ensure your solution was correct? Testing details, for instance. Ensure this is written in a way that the TAs can follow to ensure correctness.
   - We started by opening two terminal windows and sourcing the file environment-setup-i586-poky-linux in one of the windows. We ran make menuconfig to change the scheduler back to the standard one. We then ran make -j4 all to compile the kernel. After that completed, we ran the qemu virtual machine using the command qemu-system-i386 -redir tcp:5515::22 -nographic -kernel linux-yocto-3.19/arch/x86/boot/bzImage -drive file=core-image-lsb-sdk-qemux86.ext4 -enable-kvm -usb -localtime –no-reboot –append ”root=/dev/hda rw console=ttyS0 debug”, and logged in with the root account. Once the machine was running, in the other terminal window, we navigated to linux-yocto-3.19/drivers/block and ran make to build the sbd_enc.ko file, and then used scp -P 5515 sbd_enc.ko root@localhost: to copy the driver to the virtual machine. Finally, we just need to insert the driver by running insmod sbd_enc.ko, and make sure that it properly installed by running lsmod.
4) What did you learn?
   - We learned a good deal about the APIs both for the kernel and for kernel crypto. To use secure copy, we needed to learn a little about port redirection (as evidenced by the option -redir tcp:5515::22, which forces our port on the server to also listen on the ssh port, so as to let the secure copy communicate with the VM).