Homework 3s Report

CS444 Fall17

Zach Lerew, Rohan Barve

Group 26

November 14, 2017

Abstract

Homework 3 involves building an encrypted block device driver using the Linux crypto API and a reference from LDD3. The module's task is to encrypt and decrypt data as it is written and read from physical storage. The module will be designed, developed, transferred to a running VM using SCP, inserted into the module list, and then tested.

DESIGN

For this assignment, the team was given a reference driver through the LDD3 manual online. Additionally, our research has lead us to the blog of Pat Patterson, who has updated the LDD3 reference driver for the Linux Kernel v2.6.31. [1] The reference code was run in the yocto kernel and its output was examined. We determined that the transfer function is the place where data should be encrypted and decrypted. The transfer function is called by the request function periodically. The transfer function takes a flag to determine if it should read or write. When reading, the function reads from the buffer, decrypts the data, and writes it back to the buffer. When writing, the function reads from the buffer, splits the data into blocks, encrypts them, and writes to device storage.

QUESTIONS

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

The main point of this assignment was to write a device driver for the Linux yocto kernel. Understand the block device interface, utilize the kernel's crypto API and practice kernel coding skills.

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

The way our team approached this problem was to first use the LDD3 reference to read up on block device drivers and understand how the interface works. We then utilized an example device driver source file found on Pat Patterson's blog and tested it with our current kernel and VM setup.

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

1) Source environment

```
source <environment-file>
```

2) Clean and build kernel

```
cd linux-yocto-3.19
make clean && make -j4 all
```

3) Run qemu using this command, which enables networking and file transfer with Description

```
qemu-system-i386 -redir tcp:<PORT>::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 rwconsole=ttyS0 debug''
```

4) Build module

```
cd <path_to_module>
make
```

5) Transfer module to VM with SCP

```
cd <path_to_module>
scp -P <PORT> <module.ko> root@localhost:~
```

6) Insert module

```
cd ~
insmod <module.ko>
shred -z /dev/sbd0
mkfs.ext2 /dev/sbd0
mount /dev/sbd0 /mnt/
lsmod
```

7) Test module to confirm that it works

```
        echo "Create test file"
        touch /mnt/testfile
        ls -la /mnt/

        echo "Insert test data"
        echo "Test Data" > /mnt/testfile
        cat /mnt/testfile

        echo "Search for test data in module"
        grep -a "Test Data" /dev/sbd0

        echo "Display contents of module"
```

```
cat /dev/sbd0

echo "Display contents of test file"
cat /mnt/testfile

echo "Delete test file"
rm /mnt/testfile
```

8) Remove module

```
umount /mnt
rmmod sbd.ko
lsmod
```

4.What did you learn?

In this assignment, our team learned how the interactions between kernel software and hardware come together to read and write from device memory. Different kernels have different requirements for module development, which lead to lots of confusing when working with the original LDD3 reference. Kernel level errors are usually less helpful than we are used to, but they get even less helpful at the driver level. We used the Linux cryptography library to encrypt and decrypt data as it was being read and written to device storage. We learned how modules are loaded in the kernel, as well as how to transfer files to a running VM using SCP.

WORK LOG

Date	Author	Description
yyyy-mm-dd	First Last	Text

REFERENCES

[1] P. Patterson, "A simple block driver for linux kernel 2.6.31," 2010. [Online]. Available: http://blog.superpat.com/2010/05/04/a-simple-block-driver-for-linux-kernel-2-6-31/