Assignment 3

Shawn Cross, Ryan Crane

October 13, 2017

1 Plan for Implementation

For this project we plan on first finding a simple block device similar to the one in the Linux Device Drivers book that will work with our kernel. Once we have found one that works with our kernel we will then add the cryptography to it.

2 Version Control Log

3 Work Log

4 Project Questions

4.1 Main point of the assignment

The main point of this assignment was to learn how linux modules and device drivers work, as well as gaining some experience with linux's crypto api.

4.2 How did we approach the problem

We began by finding a working block device driver to use as a base. Then we looked for examples where linux's cryptographic tools were used to determine how to add encryption to the device. We chose to use AES as our block cipher with a default key of all zeros when the module parameter is blank.

4.3 Ensure correctness and details of how to preform

5 TA Evaluation

Steps for applying patch to clean linux-yocto-3.19 tree.

1. step 1: clone new repo from github using the command

   git clone git://git.yoctoproject.org/linux-yocto-3.19

2. step 2: switch to the 3.19.2 by using the command

   git checkout -b linux-yocto-3.19.2

3. step 3: copy the file /scratch/fall2017/files/config-3.19.2-yocto-qemu into a file named .config into the source root of the repo that you just cloned.

4. step 4: Now source the environment configuration. If you are using bash you will need to source /scratch/fall2017/files/environment-setup-i586-poky-linux and if you are using tcsh you will need to source /scratch/fall2017/files/environment-setup-i586-poky-linux.csh

5. step 5: You need to apply the patch we provided to the repo by using the command

   git apply [path to patch file] where path to patch file is the path you need to the patch file that we have provided.

6. step 6: You now need to make the kernel by running the command make -j4

7. step 7: Once the kernel is created you will need to run the vm using the command

   qemu-system-i386 -gdb tcp::5550 -S -nographic -kernel [path to bzImage] -drive file=[path to core image],if=virtio -enable-kvm -usb -localtime –no-reboot –append "root=/dev/vda rw console=ttyS0 debug"

   Where [path to core image] is the file found in

   /scratch/fall2017/files/core-image-lsb-sdk-qemux86.ext4

   and the [path to bzImage] for me would be the bzImage found in

   linux-yocto-3.19/arch/x86/boot/bzImage

   and the ???? are the port number that you want to connect gdb to remotely.

   After this command is run the terminal will halt.

8. step 8: you will now need to run $GDB in a new terminal window. You may need soucre the enviroment again as you did in step 4 previously if you did start a new termial session and the $GDB command does not work. Once gdb is running you will use the command

   target remote :????

   where the ???? are the same as the port that you had specified in step 7.

   then type continue and the first terminal should start.

9. step 9: You need to then scp the sbull.ko file in and then scp the provided script test.sh.

10. step 10: Type echo 'hello world' ¿ tempmount/hello
    Then type cat tempmount/hello This shows that the data being entered is different than what is acutally being stored and that once the data is decrypted the data is back to it's original form.

5.1 What did we learn

We learned that all linux documentation is terrible. We also learned about manipulating block ciphers in linux and setting module parameters, and loading them into linux. we also had to mount a block device and create a filesystem for it.