CPE453 Laboratory Assignment #5
Using Modules to write Psuedo-Device Drivers

Michael Haungs, Spring 2011

1 Objective

In this assignment, you will create a psuedo-device and write a device driver for it. The psuedo-device provides a "backdoor" for gaining root access for a particular user. Instead of compiling the device driver into the kernel, we will create a module. Modules are object binaries that can be dynamically loaded into the kernel. They are similar to a DLL in MS Windows.

2 Resources

You should read the following Linux manual pages:

- mknod: attempts to create a filesystem node (file, device special file or named pipe).
- insmod: install loadable kernel module.
- rmmod: unload loadable modules.
- lsmod: list loaded modules.
- getpid: get process identification.

The following functions defined in the linux kernel will prove to be useful:

- printk: printer messages to the log file /VAR/LOG/MESSAGES.
- find_get_pid: returns the pid structure for the process with matching PID.
- get_pid_task: returns the "task_struct" for the given "struct pid"
- copy_from_user/get_user: gets bytes from user space and copies it to kernel space.

3 Assignment

The first section gives you a tutorial to look at that describes modules and a device driver similar to the one you need to write. Use what you learned in the tutorial to create the device, `/dev/backdoor,` and its associated device driver.

3.1 Module Tutorial

To start, you need to look at "The Linux Kernel Module Programming Guide" located at `http://tldp.org/LDP/lkmpg/2.6/html/index.html`. You need to read the first four chapters and should do all of the examples shown. This guide teaches you how to make a kernel module and gives an example, *chardev*, that is similar to the module you need to write.

3.2 Backdoor device driver

Below I give you a description of the device you are creating and the functionality your device driver is required to implement. I also give you examples of creating the device, loading/unloading the device driver module, and using the device.

3.2.1 Device description

This device gives root privileges to the process whos *pid* is written to the device. When the module is installed, you supply, as an argument, the *userid* of the user who can use this device. Use this *userid*, in your device driver, to protect access to the device. Any other user, including *root*, that tries to use this device will get an "access denied" error.

When the correct user opens the device, using *open()*, and writes, using *write()*, a process id to the device, the device driver finds the task associated with the process id and gives that task root priviliges.

3.2.2 Device API

In the module code for your device driver, you need to implement the open, write, and release functions. In other words, you should have the function definitions of the aforementioned functions in your module code, as well as the following struct assignment:

```
static struct file_operations fops = {
        .write = backdoor_write,
        .open = backdoor_open,
        .release = backdoor_release
};
```

3.2.3 Example loading, using, and unloading the device driver

Below I show a session using the device. I start by inserting the module and creating the psuedo-device. I then give an example of running a program that sends it's pid to the device to gain root access. Once the process gains root access, it exec's a shell. Last, I remove the module and device file. You should do this cleanup everytime after testing or debugging your modules.

```
INSTALLING THE MODULE AS ROOT:
[root@backus backdoor_MODULE]# insmod backdoor_dev.o approvedPid=500
[root@backus backdoor_MODULE]# mknod /dev/backdoor c 254 0
[root@backus backdoor_MODULE]# chmod a+w /dev/backdoor
```

```
USING THE MODULE AS USERID=500:
[mhaungs@backus testing]$ whoami
mhaungs
[mhaungs@backus testing]$ ./getRootShell
[root@backus testing]# whoami
root
[root@backus testing]# ./getRootShell
open: Permission denied
Error in getRootPrivileges
[root@backus testing]# exit
exit
[mhaungs@backus testing]$ whoami
mhaungs

REMOVING THE MODULE AS ROOT:
[root@backus backdoor_MODULE]# rmmod backdoor_dev
[root@backus backdoor_MODULE]# rm /dev/backdoor
rm: remove character special file ‘/dev/backdoor’? y
```

Deliverables

You will submit your work to me using Blackboard's Digital Dropbox on June 3 before midnight. You need to submit the following:

1. All your ".c" and ".h" files that contain your C implementation of your backdoor module. You also need to supply a properly constructed Makefile I'll use to compile your code. Construct your Makefile so that your kernel module will compile when I type "make".
2. A program, called *mytest*, that uses your backdoor module to *exec()* a bash shell with root privileges. Construct your Makefile so that this program will compile when I type "make test".
3. A README file that: (1) lists your team members (1) describes what you got working, (2) gives a brief description of your implementations of open, write, and release, and (3) lists any bugs in your code.

Use the *Digital DropBox* tool in Blackboard to submit your files:

1. Create a zip archive of your work.
   (a) Rename the zipped folder in the following way:
   Student1Lastname_Student2LastName_Lab2.zip.
2. When using the Digital Dropbox tool, be sure to use the *Send File* button and not the *Add File* button to send me your compressed folder.
   (a) Only select me, Michael Haungs, as the recipient of your send.

GRADING NOTE 1: You must name all your files exactly as I specified above. Your output must exactly match the examples I have given above.

GRADING NOTE 2: Remember, you must have a different partner than the one you had in Lab#1.