This chapter introduces you to some of the security features of the Linux operating system. We will also cover aspects of Linux that differ from other UNIX-like operating

They Want Your Data Hackers may want your business' trade secrets for personal use or to sell. Or they may want your bank records. Or they may want your credit card numbers. Or they may want to make you look like a hacker when they launch from your machine.

Or they may just want to wreak havoc on you. The sad fact is that there are people in the world who like to sabotage other people's computer systems for no other reason than that they can. And maybe they think it is cool. And maybe they have destructive personalities. And maybe it brings them some sort of bizarre pleasure. And maybe they want to impress their hacker friends. And maybe they are bored and have nothing better to do with their lives. Who knows why they want to hack your machine? But the fact is: they do want to hack your machine. My machine. Our machines.

paper, Mr. Raymond makes many very good points about the benefits of open source

There are several shells available for Linux, including the following:

`/bin/sh` The Bourne shell, named after Steven Bourne, its creator

`/bin/ksh` The Korn shell, named after creator David Korn. It adds a number

The record has a number of fields that are colon separated. The fields are as follows:

`users`	The unique name of the group
`x`	The encrypted group password; if this field is empty, no password is needed, and if it is `x`, use the group shadowing file `/etc/gshadow`
`100`	The unique group ID number
`jdoe,student`	A comma-separated list of the group member usernames

Therefore, the group `users` is a collection of normal users on the system, in this case the users `jdoe` and `student`.

How to Place Controls on Users

If we put this idea into practice for owner / group / world permission, then the permissions

```
rwxr-x--x
```

in binary format are

```
111101001
```

and if we treat this as a series of three groups of octal numbers, the value is 751.

Changing File Permissions The chmod command changes file permissions. Its format is

```
chmod mode file [file ...]
```

To see how to use chmod, let's look at a file on our system:

```
jdoe@server1$ ls -l a.txt
-rw-rw-r--   1 jdoe   users          10 Nov
jdoe@server1$ chmod 751 a.txt
jdoe@server1$ ls -l a.txt
-rwxr-x--x   1 jdoe   users          10 Nov
```

751 translate to rwxr-x--x. No

```
jdo
-rw
```

Here, 640 translates to rw-r-----.

You can also use the chmod command in symbolic mode as follows:

```
jdoe@server1$ ls -l a.txt
-rw-r-----   1 jdoe    jdoe          10 Nov 15 12:24
jdoe@server1$ chmod +x a.txt
jdoe@server1$ ls -l a.txt
-rwxr-x--x   1 jdoe    jdoe          10 Nov 15 12:24
```

Here, chmod is used with +x″ which means "add executable permission." When the + character is used, it means to add the permission, whereas the – character means to subtract or remove the permission. Here, +x means to add executable permissions for the owner, group, and world. The chmod command can also be used to change permissions for a specific group:

```
jdoe@server1$ chmod g-r a.txt
jdoe@server1$ ls -l a.txt
-rwx--x--x   1 jdoe    jdoe          10 Nov 15 12:24 a.txt
```

Notice how a `umask` value of `077` gave `jdoe` read / write permissions for the file `d` and read / write / execute permissions for `directory_e`, but no permissions to the group

```
jdoe@server1$ ulimit -n 512
jdoe@server1$ ulimit –a
core file size (blocks)  1000000
data seg size (kbytes)   unlimited
file size (blocks)       unlimited
max memory size (kbytes) unlimited
stack size (kbytes)      8192
cpu time (seconds)       unlimited
max user processes       2048
pipe size (512 bytes)    8
open files               512
virtual memory (kbytes)  2105343
```

After ten years of failing to get the capability-based security model (POSIX 1003.1e) spec'd out, the committee in charge dropped the draft. Though Linux and other systems are implementing capabilities, do not expect them to be handled in exactly the same way between different UNIX-like operating systems.

A process can be given full control of the set capabilities, such that it can pass them onto other programs the process runs, or you can restrict these capabilities to this program only and not any of its children. This means you can offer permissions for a process

Other operating systems do not have this compartmentalization. This means that all the system memory may be available to all of the processes on the machine.

System Logging

Linux has a standard logging facility that is very easy to use and can be plugged into es-