CHAPTER

3

Network Hosts Scanning

This chapter is your first step into the penetration testing workflow. Whether you're advanced or a novice, this chapter will help you conduct your network scan with success. In the beginning, we will walk through the basics you need to know before you start scanning a network. Afterward, we will delve deeper to see how to scan a network target.

This chapter covers the following:

- The basics of networking
- Identifying live hosts
- Port scanning
- Services enumeration
- Operating system fingerprinting
- Nmap scripting engine
- Scanning for subdomains

Basics of Networking

Before you start scanning and identifying hosts, you need to understand the basics of networking first. For example, why do we use 10.0.0.1/16? Or what is a TCP handshake? Let's start!

Kali Linux Penetration Testing Bible. Gus Khawaja

Networking Protocols

The following are the two main networking protocols you need to be aware of to scan a network successfully.

TCP

The Transmission Control Protocol (TCP) is the main one used in network infrastructure. Every application server (HTTP, FTP, SMTP, etc.) uses this protocol to properly connect the client with the server.

TCP uses a concept called a *three-way handshake* to establish a network connection. First, to start a TCP session, the client sends a SYN packet (synchronize) to the server. The server receives the SYN and replies to the client with a synchronize/acknowledge (SYN/ACK) packet. Finally, the client completes the conversation by sending an ACK packet to the server.

For example, Figure 3.1 shows a scenario of Mr. Bob surfing the internet and searching on Google (the web server) using his browser (client) by visiting `www.google.com`.

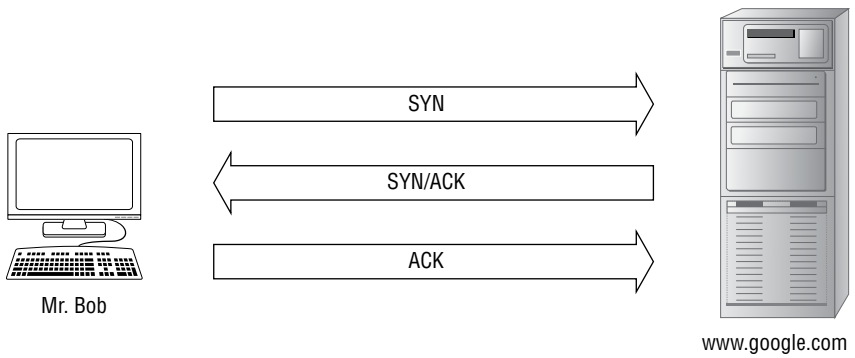

Figure 3.1: TCP Handshake

It's essential to understand the concept of the TCP handshake. Network scanners like Nmap use it to identify live hosts, open ports, and much more (you will learn more about this in the upcoming sections).

A network sniffer like Wireshark is a good tool to learn how computer networks work. Why? Because a network sniffer will listen to all incoming and outgoing traffic through the network card.

To start Wireshark, just type its name (`$wireshark`) in the terminal window. Next, you will need to select the network interface; it's either an Ethernet `eth0` or WiFi `wlan0`. In this case, we're using `eth0`, and then we will click the Start button in the top-left corner of the screen (see Figure 3.2).

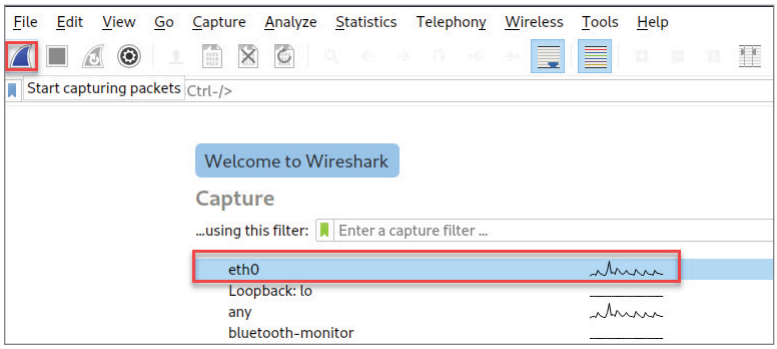

Figure 3.2: Wireshark Network Interface Selection

Figure 3.3 is taken from Wireshark on my Kali (10.0.0.20), which appears when I open my browser and go to `Google.com`. If you look closely, you can see the [SYN], [SYN ACK], and [ACK] packets.

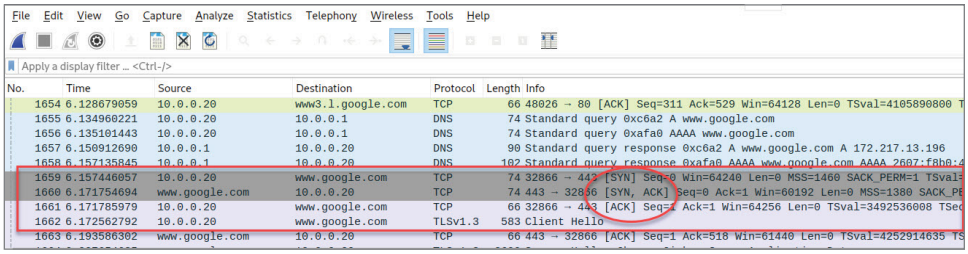

Figure 3.3: Wireshark Capture

UDP

The User Datagram Protocol (UDP) is a *connectionless* network connection. Contrary to the TCP connection, the UDP client and server will not guarantee a packet transmission, so there is no three-way handshake in UDP. Examples of applications that use UDP are audio and video streaming—you're looking for performance in these kinds of connections. Later in this chapter, Table 3.3 shows the most popular applications with their appropriate protocol, either TCP or UDP.

Other Networking Protocols

TCP and UPD are the most popular network protocols, but other types of protocols exist as well. In this section, we will cover the rest of them.

ICMP

Internet Control Message Protocol (ICMP) is used for testing connectivity. The Ping tool uses this protocol to test whether a host network is up and running (`traceroute` uses it as well by default). In the following example, we will ping the IP address 10.0.20.1 and check the ICMP connection from Wireshark:

```
root@kali:~# ping 10.0.0.1 -c 3
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.706 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=64 time=0.725 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=64 time=0.506 ms

--- 10.0.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2033ms
rtt min/avg/max/mdev = 0.506/0.645/0.725/0.099 ms
```

Figure 3.4 shows Wireshark with the `icmp` filter applied:

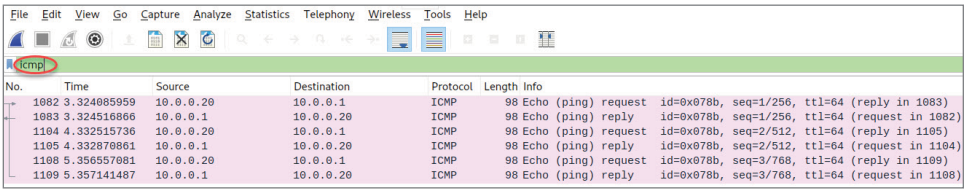

Figure 3.4: Wireshark ICMP Filter

ARP

Address Resolution Protocol (ARP) is a mechanism that maps IPv4 addresses to MAC addresses. This concept is essential for an internal network to work. Routers connect with each other over the internet through IP addresses (layer 3), but once the packet is inside your network, then an ARP table will be used using MAC addresses (layer 2). You can use the command `arp -a` to get the list of items inside the ARP table (saved locally on your localhost):

```
root@kali:~# arp -a
? (10.0.0.10) at 70:5a:0f:f6:fc:3a [ether] on eth0
USGPRO (10.0.0.1) at b4:fb:e4:2f:04:3d [ether] on eth0
```

The layers are referring to the OSI layers shown earlier (see Table 3.1). The Open System Interconnection (OSI) separates a network connection into different layers.

Table 3.1: OSI Layers

NUMBER	NAME	PROTOCOL EXAMPLES	DEVICE EXAMPLES
1	Physical	Ethernet etc.	Cables
2	Data Link	MAC, VLAN, etc.	Switches
3	Network	IPv4/v6 ICMP etc.	Routers
4	Transport	TCP UDP	NA
5	Session	NA	NA
6	Presentation	NA	NA
7	Application	FTP, HTTP, Telnet, etc.	Firewalls, proxies, etc.

IP Addressing

The Internet Protocol (IP) is one of the main pillars in networking so that computers can communicate with one another. IP addresses are divided into two versions: IPv4 and IPv6.

IPv4

IPv4 is 32-bit but always presented in a decimal format such as 192.168.0.1, which is equal to 11000000.10101000.00000000.00000001. It's simpler to write it in a decimal format instead of in binary, right?

IP addresses are divided into public, which are used on the internet, and private, which are used in an intranet. Your public IP address is probably supplied to you automatically by your internet service provider (ISP) unless you bought a static public IP address.

Here are the private IPv4 address ranges:

- 10.0.0.0 to 10.255.255.255 (10.x.x.x), with about 16 million addresses
- 172.16.0.0 to 172.31.255.255 (172.16.x.x to 172.31.x.x), with about 1 million addresses
- 192.168.0.0 to 192.168.255.255 (192.168.x.x), with about 65,000 addresses

Subnets and CIDR

The subnet's role is to divide a network into smaller ranges (network segmentation). A subnet will identify the number of hosts inside an IP range. For example, 192.168.0.1 could have a subnet mask 255.255.255.0, which means that we can

use 254 hosts inside this IP range. Classless Interdomain Routing (CIDR) was created to simplify the subnet masks. If we take the prior example, we can write the subnet /24 (the CIDR equivalent) instead of a long one. Table 3.2 lists subnets and netmasks that you can use as a reference.

Table 3.2: Subnets and CIDR

CIDR	NETMASK	# OF HOSTS
/30	255.255.255.252	2
/29	255.255.255.248	6
/28	255.255.255.240	14
/27	255.255.255.224	30
/26	255.255.255.192	62
/25	255.255.255.128	126
/24	255.255.255.0	254
/23	255.255.254.0	510
/22	255.255.252.0	1,022
/21	255.255.248.0	2,046
/20	255.255.240.0	4,094
/19	255.255.224.0	8,190
/18	255.255.192.0	16,382
/17	255.255.128.0	32,766
/16	255.255.0.0	65,534
/15	255.254.0.0	131,070
/14	255.252.0.0	262,142
/13	255.248.0.0	524,286
/12	255.240.0.0	1,048,574
/11	255.224.0.0	2,097,150
/10	255.192.0.0	4,194,302
/9	255.128.0.0	8,288,606
/8	255.0.0.0	16,777,216

IPv6

So far, we're still using IPv4 heavily to operationalize the network infrastructure. There have been a few attempts to change from IPv4 to IPv6 since the world

will run out of IPv4 addresses one day. You need to understand at least how IPv6 operates in practice.

The IPv6 format is 128 bits of hexadecimal characters. Here is an example of IPv6 worth a trillion words:

fff0:0000:eeee:0000:0000:0000:fe77:03aa

We'll use this example to see how the IPv6 format works.

1. To follow the IPv6 specifics first, we need to remove the leading zeros.

   Before: fff0:**0000**:eeee:**0000**:**0000**:**0000**:fe77:**0**3aa
   After: fff0:0:eeee:0:0:0:fe77:3aa

2. Compress the series of zeros (in our case, there are three series of zeros) and replace them with ::.

   Before: fff0:0:eeee:**0:0:0**:fe77:3aa
   After: fff0:0:eeee::fe77:3aa

Take note that in IPv6, you can compress a series of zeros only once.

Port Numbers

Port numbers and IP addresses are like brothers and sisters. Without a port number, a network packet will never be able to reach its destination. A port number is like a civic address. The street's name (IP address) is not enough to get to a certain property; you will need a civic number (port number) to have a full and complete address.

Imagine that you're using your browser to reach `www.google.com`. Your packet will need the IP address of the web server host and the port number, which by default is 443 for HTTPS. On the same server (with the same IP address), Google could be hosting other services like FTP, for example; then the packet will use port 21 to reach it.

Table 3.3 lists the most common default port numbers that you'll need to know while scanning a network. Take note that port numbers range from 1 to 65,535.

Table 3.3: Common Port Numbers

PROTOCOL NAME	PORT #	PROTOCOL NAME	PORT #
FTP	TCP 21	LDAP over SSL	TCP 636
SSH/SCP	TCP 22	FTP over SSL	TCP 989–990
Telnet	TCP 23	IMAP over SSL	TCP 993
SMTP	TCP 25	POP3 over SSL	TCP 995
DNS Query	UDP 53	MS-SQL	TCP 1433
DNS Zone Transfer	TCP 53	NFS	TCP 2049

Continues

Table 3.3 (*continued*)

PROTOCOL NAME	PORT #	PROTOCOL NAME	PORT #
DHCP	UDP 67	Docker Daemon	TCP 2375
	UDP 68		
TFTP	UDP 69	Oracle DB	TCP 2483–2484
HTTP	TCP 80	MySQL	TCP 3306
Kerberos	UDP 88	RDP	TCP 3389
POP3	TCP 110	VNC	TCP 5500
SNMP	UDP 161	PCAnywhere	TCP 5631
	UDP 162		
NetBIOS	TCP/UDP 137	IRC	TCP 6665–6669
	TCP/UDP 138		
	TCP/UDP 139		
IMAP	TCP 143	IRC SSL	TCP 6679
			TCP 6697
LDAP	TCP 389	BitTorrent	TCP 6881–6999
HTTPS (TLS)	TCP 443	Printers	TCP 9100
SMTP over SSL	TCP 465	WebDAV	TCP 9800
rlogin	TCP 513	Webmin	10000

This table enumerates the most popular port numbers that I personally think you must know. For a full list, please check it out on Wikipedia at `en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`.

Network Scanning

Now that you understand the basics of networking, it's time to start the action. In the upcoming sections, you will learn how to identify target hosts on the network.

Identifying Live Hosts

There are multiple ways to determine whether a host is up and running on the network.

Ping

You can use Ping to quickly check the network connection. So, how does Ping work under the hood?

When you execute the `ping` command, your Kali host will send an ICMP echo request to the destination, and afterward, the target will respond with an ICMP echo reply packet. Thus, you can say that the target is alive. The `ping` command is useful for system administrators, but we're the elite, right? Later, you will see why you must use Nmap to scan for live hosts. Finally, you need to be aware that some system admins will close the ICMP echo on the firewall level to block hackers from checking the connectivity of some servers.

ARP

Address Resolution Protocol is a fantastic utility that maps IP addresses into physical MAC addresses in a local network.

Now, we can take advantage of the ARP table contents to list all the hosts on the same network using the `arp-scan` command on Kali:

```
root@kali:~# arp-scan 10.0.0.1/24
Interface: eth0, type: EN10MB, MAC: 00:0c:29:40:e7:a6, IPv4: 10.0.0.20
WARNING: host part of 10.0.0.1/24 is non-zero
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/
arp-scan)
10.0.0.1        b4:fb:e4:2f:04:3d       Ubiquiti Networks Inc.
10.0.0.2        fc:ec:da:d4:d5:99       Ubiquiti Networks Inc.
10.0.0.5        b4:fb:e4:1b:c4:d2       Ubiquiti Networks Inc.
10.0.0.10       70:5a:0f:f6:fc:3a       Hewlett Packard
10.0.0.50       00:11:32:94:25:4c       Synology Incorporated
10.0.0.75       fc:ec:da:d8:24:07       Ubiquiti Networks Inc.
10.0.0.102      d0:2b:20:95:3b:96       Apple, Inc.
```

Nmap

It's now time to show you my favorite tool that I use to identify live hosts: Nmap. You will need to use the following command options to get the job done:

```
$nmap -sn [IP Address / Range]
```

To help you memorize it, think of the option `-s` as Sam and `N` is Nanny. The real meaning of these options are as follows:

- `n` is for No.
- `s` is for Scan.

That's why the name of this option is No Port Scan. Some people call it Ping Scan, but don't mix it with the ICMP Ping tool that we talked about earlier in this chapter. That being said, let's see why this option is magical. To identify live hosts, Nmap will attempt to do the following:

1. It will send an ICMP echo request, and Nmap will not give up if ICMP is blocked.
2. Also, it will send an ICMP timestamp request.
3. It will send an ACK packet to port 80 and send a SYN packet to port 443.
4. Finally, it will send an ARP request.

It's so powerful, right? It's important to understand that you will need to be root (or a member of the `sudo` group) on your Kali box, or else your options will be limited, and you won't be able to execute all these functionalities. Let's put Sam and Nanny into action:

```
root@kali:~# nmap -sn 10.0.0.1/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:25 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00036s latency).
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Nmap scan report for unifi (10.0.0.2)
Host is up (0.00027s latency).
MAC Address: FC:EC:DA:D4:D5:99 (Ubiquiti Networks)
Nmap scan report for 10.0.0.5
Host is up (0.0024s latency).
MAC Address: B4:FB:E4:1B:C4:D2 (Ubiquiti Networks)
Nmap scan report for 10.0.0.10
Host is up (0.0081s latency).
MAC Address: 70:5A:0F:F6:FC:3A (Hewlett Packard)
Nmap scan report for 10.0.0.50
Host is up (0.00066s latency).
MAC Address: 00:11:32:94:25:4C (Synology Incorporated)
```

Port Scanning and Services Enumeration

One of the tasks that you will be asked for during network scanning is to look for the open ports on each host. Why? Let's say you want to know all the web servers on the local area network (LAN); the port scan will allow you to get this information easily. Let's see how Nmap handles this task like a boss.

TCP Port SYN Scan

There are so many options in Nmap to execute a port scan, but the one that I always use for TCP is the SYN scan. In fact, Nmap will execute this type of port scan by default:

```
$nmap -sS [IP address / Range]
```

To memorize it, you can correlate the option `-sS` to Sam and Samantha. Always think about Sam and Samantha when you want to execute a port scan. You're a lucky guy if your name is Sam, but the `sS` option stands for a *SYN scan*, and some people call it *stealth scan*; I made up the SAM -Samantha terms so you can memorize it easily.

Let me explain to you how the SYN scan works in Nmap. The scanner, when supplied with the `sS` option, will send a SYN request to the server, and if a SYN/ACK is received in the response, then it will show that the port is open. And if the scanner did not receive a SYN/ACK, it's either closed or filtered. For the record, *filtered* means a firewall is protecting it:

```
root@kali:~# nmap -sS 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:27 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00051s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Nmap done: 1 IP address (1 host up) scanned in 0.25 seconds
```

UDP

Now, what about the UDP port scan? To use the UDP port scan in Nmap, you have to add the `sU` option:

```
$nmap -sU [IP Address / Range]
```

It is important to note that UDP is slow due to its connectionless nature. We can always use the T5 timing option to tweak it and make it faster. Firewalls can easily block T5 on an internet address. The T2 option is your friend when scanning internet IP addresses, so you can bypass the radar (firewalls, etc.):

```
$nmap -sU -T5 [IP Address / Range]
```

So, for the UDP scanner to identify whether the port is open or closed, it will send a UDP packet and wait for a response from the destination. If Nmap got a response or not, probably the port is open.

On the other hand, if the scanner received an ICMP error, then the port is either closed or filtered:

```
root@kali:~# nmap -sU -T5 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:28 EDT
Warning: 10.0.0.1 giving up on port because retransmission cap hit (2).
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.0014s latency).
Not shown: 875 open|filtered ports, 123 closed ports
PORT    STATE SERVICE
53/udp  open  domain
123/udp open  ntp
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Nmap done: 1 IP address (1 host up) scanned in 119.25 seconds
```

Basics of Using Nmap Scans

Let's discuss a few basics in Nmap. If you run Nmap without any options, the tool will use three significant functionalities by default:

- It will set the speed to T3.
- It will scan the top TCP 1000 ports.
- Assuming you're root on your Kali box, it will set the SYN TCP scan by default.

In other words, all this is happening in the back end, which means you don't need to specify the T3 speed, because it's there by default. It's the same as for the port numbers (you don't need to add `--top-ports 1000`) or the TCP SYN Scan (you don't need to add the `-sS` option). In the previous examples, we specified the `sS` option for the SYN scan, but we don't need to do that here because Nmap will set it by default, right?

For the tweaking part, always remember to choose the speed wisely. For example, don't use a fast speed like T5 on a production IP address; instead, stick with the default one (T3). Also, make sure you choose the number of ports that suits your needs, either the top 100 or the default option 1,000. Let's look at a practical example; let's say you want to scan only the top 100 ports using the TCP scan:

```
#A quicker TCP scan
$nmap --top-ports 100 -T5 [IP Address / Range]
```

If you're targeting a specific port, then use the `-p` option, followed by the port number, range, or list:

```
#To scan for the HTTP port 80 on the network
$nmap -p 80 [IP Address / Range]
```

Finally, if you want to include all the ports, then use the `-p-` (also you can use `-p 1-65535`) to scan every single port (scanning all the port numbers will reveal any hidden applications). I never use this option for UDP (because it's too slow), but I use it a lot for TCP scanning (in the following command, we didn't specify the `-sS` option, because it's there by default):

```
root@kali:~# nmap -p- -T5 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:35 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00097s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Nmap done: 1 IP address (1 host up) scanned in 9.91 seconds
```

Services Enumeration

It's time to see how to execute a service version scan using the best scanner: Nmap. The option for doing a version scan in Nmap is `-sV`. Generally, a good way to memorize the command options in Nmap is to apply a meaning to every letter. For example, the `s` stands for *scan*, and the `V` stands for *version*. Easy, right? That's why it's called a *version scan*. Take note that a version scan will take longer than a normal port scan since it will try to identify the service type. In the following example , we will scan the same host that we used earlier, but this time we will add the `-sV` option (check the version column):

```
root@kali:~# nmap -p- -T5 -sV 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:36 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00097s latency).
Not shown: 65531 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
```

Continues

(continued)

```
80/tcp  open  http     lighttpd
443/tcp open  ssl/http Ubiquiti Edge router httpd
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 25.88 seconds
```

Here is a good tip to make the version scan even faster. Ready? Since we already scanned for the open ports previously, we don't need to scan for all the ports again using `-p-`. Instead, we can only specify the port numbers that we identified in the port scan previously (compare the speed time to the previous version scan; it's half the time!):

```
root@kali:~# nmap -p 22,53,80,443 -T5 -sV 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:39 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00092s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
80/tcp  open  http     lighttpd
443/tcp open  ssl/http Ubiquiti Edge router httpd
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.63 seconds
```

Finally, you can change the version scan aggressivity. In other words, how deep do you want Nmap to scan the version of each service? The higher the intensity, the more time it will take to scan the target host. Table 3.4 summarizes the version intensity options:

Table 3.4: Nmap Version Intensity

NMAP OPTION	DESCRIPTION
`--version-intensity [0-9]`	9 is the highest intensity, and the default is 7.
`--version-light`	Is equivalent to `--version-intensity 2`.
`--version-all`	Is equivalent to `--version-intensity 9`.

TIP **You can always use the `nmap -h` command to list all the options inside Nmap in case you forget any of them. It will take you years of practice to know them all by heart.**

In the following example, we will use the `--version-light` option to make it even faster. Note that we were able to make it one second quicker than before without sacrificing the version column information (maybe one second is not a big deal for one host, but for a network range, it will make a big difference):

```
root@kali:~# nmap -p 22,53,80,443 -T5 -sV --version-light 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:41 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00099s latency).

PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
80/tcp  open  http     lighttpd
443/tcp open  ssl/http Ubiquiti Edge router httpd
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.60 seconds
```

Operating System Fingerprinting

Imagine the following scenario: your manager or your client comes to you and says, "We want to know if anyone in our network is using Windows XP." You smile and say, "Of course, I know how to do it like a pro." In this section, you will learn how to identify the operating system of a host on a LAN using Nmap.

To get the job done, you will need to add the `-O` option to detect the operating system of the target host. Nmap will attempt to identify the target operating system by inspecting the packets received from the host. Next, Nmap will try to match the fingerprint to a saved list. Take note that in the following example, we are adding the version scan to help identify the operating system:

```
root@kali:~# nmap -sV -O -T4 10.0.0.187
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:45 EDT
Nmap scan report for Win7Lab.ksec.local (10.0.0.187)
Host is up (0.00035s latency).
Not shown: 990 closed ports
```

Continues

(continued)

```
PORT      STATE SERVICE      VERSION
135/tcp   open  msrpc        Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
5357/tcp  open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
49152/tcp open  msrpc        Microsoft Windows RPC
49153/tcp open  msrpc        Microsoft Windows RPC
49154/tcp open  msrpc        Microsoft Windows RPC
49155/tcp open  msrpc        Microsoft Windows RPC
49156/tcp open  msrpc        Microsoft Windows RPC
49157/tcp open  msrpc        Microsoft Windows RPC
MAC Address: 00:0C:29:1C:0E:EE (VMware)
Device type: general purpose
Running: Microsoft Windows 7|2008|8.1
OS CPE: cpe:/o:microsoft:windows_7::- cpe:/o:microsoft:windows_7::sp1
cpe:/o:microsoft:windows_server_2008::sp1
cpe:/o:microsoft:windows_server_2008:r2
cpe:/o:microsoft:windows_8 cpe:/o:microsoft:windows_8.1
OS details: Microsoft Windows 7 SP0 - SP1, Windows Server 2008 SP1,
Windows Server 2008 R2, Windows 8, or Windows 8.1 Update 1
Network Distance: 1 hop
Service Info: Host: WIN7LAB; OS: Windows; CPE: cpe:/o:microsoft:windows

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 66.00 seconds
```

Nmap Scripting Engine

The Nmap Scripting Engine (NSE) contains a set of additional functionalities (like brute force, DNS enumeration, HTTP enumeration, etc.) that make Nmap work like a boss. The Nmap team categorized all these functionalities into different groups, shown here:

- Auth
- Broadcast
- Default
- Discovery
- DOS
- Exploit
- External
- Fuzzer
- Intrusive

- Malware
- Safe
- Version
- Vuln

To get the list of all these NSE scripts, just list the contents of the directory `/usr/share/nmap/scripts`:

```
root@kali:~# ls /usr/share/nmap/scripts/
acarsd-info.nse                         http-hp-ilo-info.nse
nping-brute.nse
address-info.nse                        http-huawei-hg5xx-vuln.nse
nrpe-enum.nse
afp-brute.nse                           http-icloud-findmyiphone.nse
ntp-info.nse
afp-ls.nse                              http-icloud-sendmsg.nse
ntp-monlist.nse
afp-path-vuln.nse                       http-iis-short-name-brute.nse
omp2-brute.nse
afp-serverinfo.nse                      http-iis-webdav-vuln.nse
omp2-enum-targets.nse
afp-showmount.nse                       http-internal-ip-disclosure.nse
omron-info.nse
ajp-auth.nse                            http-joomla-brute.nse
openlookup-info.nse
ajp-brute.nse                           http-jsonp-detection.nse
openvas-otp-brute.nse
ajp-headers.nse                         http-litespeed-sourcecode-
download.nse  openwebnet-discovery.nse
ajp-methods.nse                         http-ls.nse
oracle-brute.nse
ajp-request.nse                         http-majordomo2-dir-traversal.nse
oracle-brute-stealth.nse
allseeingeye-info.nse                   http-malware-host.nse
oracle-enum-users.nse
amqp-info.nse                           http-mcmp.nse
oracle-sid-brute.nse
[...]
```

A common example is HTTP enumeration, an NSE script exists for this purpose. To get the job done, we need to specify the port number that we're targeting first, and also, we will add the version scan (which is optional, but I recommend it) to get juicier information:

```
$nmap -p [port number] -sV –script [NSE script name] [IP address / range]
root@kali:~# nmap -sV -p 80 --script http-enum 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:51 EDT
```

Continues

(continued)

```
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00082s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    lighttpd
|_http-server-header: Server
|_https-redirect: ERROR: Script execution failed (use -d to debug)
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.32 seconds
```

NSE Category Scan

When you use a category scan, you don't specify the port number since that a category will target multiple port numbers (all the TCP open ports). The most common one used in penetration testing is the default script scan `-sC`. Before we proceed with an example, you need to know why it's popular. The default script scan has a low number of false positives (*false positive* means a false vulnerability) and is less intrusive on the target system compared to other categories (some categories could bring your target host down like the DOS category):

```
root@kali:~# nmap -sV -sC 10.0.0.1
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:52 EDT
Nmap scan report for USGPRO (10.0.0.1)
Host is up (0.00059s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 6.6.1p1 Debian 4~bpo70+1 (protocol 2.0)
| ssh-hostkey:
|   1024 40:a1:21:7f:53:fe:71:41:bb:54:5d:83:1d:44:dd:65 (DSA)
|   2048 fa:08:a3:16:7c:3a:48:e3:7e:d6:ea:2c:6a:5d:15:93 (RSA)
|   256 36:d5:77:3f:f8:6f:a0:36:07:30:7a:43:1f:4d:ac:b5 (ECDSA)
|_  256 88:5a:3c:60:df:0a:dd:b2:2b:4e:a8:af:19:d7:f5:9e (ED25519)
53/tcp  open  domain   dnsmasq 2.78-23-g9e09429
| dns-nsid:
|_  bind.version: dnsmasq-2.78-23-g9e09429
80/tcp  open  http     lighttpd
|_http-server-header: Server
|_http-title: Did not follow redirect to https://usgpro/
|_https-redirect: ERROR: Script execution failed (use -d to debug)
443/tcp open  ssl/http Ubiquiti Edge router httpd
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Server
```

```
|_http-title: UniFi Security Gateway
| ssl-cert: Subject: commonName=UbiquitiRouterUI/
organizationName=Ubiquiti Inc./stateOrProvinceName=New York/
countryName=US
| Subject Alternative Name: DNS:UbiquitiRouterUI
| Not valid before: 2020-03-11T01:02:25
|_Not valid after:  2022-06-13T01:02:25
|_ssl-date: TLS randomness does not represent time
MAC Address: B4:FB:E4:2F:04:3D (Ubiquiti Networks)
Service Info: OS: Linux; Device: router; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.20 seconds
```

Also, you can use the `-A` switch to execute the default script scan, but be aware that this option will use the following:

- Version scan
- Syn TCP scan
- Default NSE script scan
- Operating system scan
- Network traceroute

```
$nmap -A [IP address / range]
```

Another way to target multiple scripts in the same category is to use the wildcard `*`. For example, to target all the Samba vulnerabilities-related scripts, you will need to use the following command. (Take note that this command will take a lot of time to end. At any point in time during the execution, you can press the Enter key to get the progress percentage value.)

```
root@kali:~# nmap -sV -p 135,445 --script smb-vuln* 10.0.0.187
Starting Nmap 7.80 ( https://nmap.org ) at 2020-10-05 09:54 EDT
Nmap scan report for Win7Lab.ksec.local (10.0.0.187)
Host is up (0.00027s latency).

PORT    STATE SERVICE      VERSION
135/tcp open  msrpc        Microsoft Windows RPC
445/tcp open  microsoft-ds Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
MAC Address: 00:0C:29:1C:0E:EE (VMware)
Service Info: Host: WIN7LAB; 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:
```

Continues

(continued)

```
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers
(ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in
Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-
guidance-for-wannacrypt-attacks/
|       https://technet.microsoft.com/en-us/library/security/ms17-010
.aspx
|_      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 12.60 seconds
```

NSE Arguments

Some NSE scripts will require you to enter some additional arguments. A good example is the brute-force attack. Let's say you want to brute-force the SSH service on the target host, so you will need to add the; `--script-args` option to identify the files for the usernames/passwords:

```
$nmap -p 22 -sV –script ssh-brute –script-args userdb=users.txt,
passdb=passwords.txt
```

The secret to knowing all these options is to use the Nmap official NSE reference: `https://nmap.org/nsedoc/`

DNS Enumeration

Why is this section in the network scanning chapter? DNS enumeration may allow us to identify the nature of the target host that we want to scan. In addition, DNS enumeration will use the public search engines to search for hidden domain names that we were not aware of at the beginning of our engagement. For example, the `router.ethicalhackingblog.com` target is probably a router host (`ethicalhackingblog.com` is my domain; I'm just giving an example here).

DNS Brute-Force

There are many tools that will search for DNS domain names using the brute-force methodology. To begin with, let's try to understand how a DNS brute-force works by developing our own Bash script. First, let's save a list of potential subdomains that could exist:

- www
- vpn
- prod
- api
- dev
- ftp
- staging
- mail

TIP **In practice, I use the following subdomain dictionary files from the list on GitHub:**

https://github.com/danielmiessler/SecLists/tree/master/Discovery/DNS

Now that we have defined our subdomain's dictionary file, it's time to develop the Bash script:

```
#! /bin/bash
#This script will brute force domain names

#Prompt to user to enter the domain name
read -p "Enter the domain name that you want to brute-force: " DOMAIN_
NAME


function check_domain(){
        #Execute the host command and extract the live subdomains
        results=$(host $SUB_DOMAIN | grep 'has address')
        #if not empty results
        if [[ -n $results ]]
        then
                printf "Found $SUB_DOMAIN\n"
        fi
}
```

Continues

(continued)

```
#read the dictionary file
for sub in $(cat sub-d.txt)
do
        SUB_DOMAIN=$sub.$DOMAIN_NAME
        check_domain
done
```

Let's test it on my blog website:

```
root@kali:/opt# ./dns-brute.sh
Enter the domain name that you want to brute-force: ethicalhackingblog
.com
Found www.ethicalhackingblog.com
Found ftp.ethicalhackingblog.com
```

DNS Zone Transfer

Copying the DNS master server records to another DNS slave server is called a *DNS transfer*. We can take advantage of this process to get the list of all the subdomains by asking the master DNS (referred to as an NS server, for example `ns1.ethicalhackingblog.com`) to give us the list of the target domain name. This query rarely works, but if it's successful, then you'll be handed with a gift containing all the subdomains and their associated IP addresses.

First, let's get the list of the `ns` servers for a random domain name:

```
$host -t ns [domain name] | cut -d " " -f 4
root@kali:/opt# host -t ns google.com | cut -d " " -f 4
ns4.google.com.
ns3.google.com.
ns1.google.com.
ns2.google.com.
```

Now that we know the list of the DNS servers, we can ask the server for the DNS records. Let's try to ask the `ns1` server using the `host` command:

```
$host -l [domain name] [ns server domain name]

root@kali:/opt# host -l google.com ns1.google.com
Using domain server:
Name: ns1.google.com
Address: 216.239.32.10#53
Aliases:

; Transfer failed.
```

Now that you know how Bash scripting works, you can create your own scripts that will extract the list of `ns` servers and then loop to each one, and then you can ask for the list of DNS records using the zone transfer technique specified earlier. Now you understand the usage and power of Bash scripting.

DNS Subdomains Tools

There are a lot of tools out there that scan for subdomains. What matters is to understand what they do, so you need to make sure that your chosen one will do the following:

1. Quickly brute-force subdomains based on a good quality dictionary file.
2. Check for DNS transfer.
3. Automate a subdomain lookup on internet search engines like Google.

Fierce

Fierce is a great tool for searching subdomains. Fierce will execute multiple DNS tests, including zone transfers and brute-forcing as well. It's fast and contains a good dictionary file:

```
$fierce -dns [domain name]

root@kali:/opt# fierce -dns ethicalhackingblog.com
DNS Servers for ethicalhackingblog.com:
        ns66.domaincontrol.com
        ns65.domaincontrol.com

Trying zone transfer first...
        Testing ns66.domaincontrol.com
                Request timed out or transfer not allowed.
        Testing ns65.domaincontrol.com
                Request timed out or transfer not allowed.

Unsuccessful in zone transfer (it was worth a shot)
Okay, trying the good old fashioned way... brute force

Checking for wildcard DNS...
Nope. Good.
Now performing 2280 test(s)...
45.40.155.145   ftp.ethicalhackingblog.com
45.40.155.145   www.ethicalhackingblog.com
```

Continues

(continued)

```
Subnets found (may want to probe here using nmap or unicornscan):
        45.40.155.0-255 : 2 hostnames found.

Done with Fierce scan: http://ha.ckers.org/fierce/
Found 2 entries.

Have a nice day.
```

Do not rely only on one tool to get the job done; each tool has its pros and cons. Other good DNS tools are also to be considered in the search of subdomains:

```
#sublist3r is on Github
$python sublist3r.py [domain name]
#subbrute is on Github
$python subbrute.py [domain name]
```

Summary

You just learned the first step in the ladder of penetration testing. Ideally, you enjoyed this chapter and learned something new. In the upcoming chapters, we will take advantage of the information in network scans to exploit target hosts. The fun has just begun, folks!