

6

Active Information Gathering

Active information gathering can be used to provide very useful results during the reconnaissance phase of a penetration test. With this active approach, the penetration tester makes a direct connection to the actual target to gather specific details that **Open Source Intelligence (OSINT)** is unable to provide. Using active information gathering, the penetration tester is able to create a very detailed profile of the target, gathering information such as the type of operating system and running services. This information helps to research and identify vulnerabilities in relation to the target, thereby narrowing the scope in choosing specific exploits to unleash against it.

For this entire chapter, we will focus on directly engaging the target to gather specific details about it in order to help us profile any running services. Understanding how to perform active reconnaissance will provide us with vital assistance for the exploitation phase. During the information-gathering phase, you'll be able to identify vulnerabilities and determine suitable exploits to break into a system and network. You will also be able to retrieve sensitive information from network devices and systems.

During the course of this chapter, we will cover the following topics:

- Understanding active information gathering
- DNS interrogation
- Scanning
- Nmap
- Hping3
- SMB, LDAP enumeration, and null sessions
- Web footprints and enumeration with EyeWitness
- Metasploit auxiliary modules

Technical requirements

The following are the technical requirements for this chapter:

- Kali Linux: www.kali.org
- Wireshark: www.wireshark.org
- JXplorer: <https://github.com/pegacat/jxplorer>
- EyeWitness: <https://github.com/FortyNorthSecurity/EyeWitness>

Understanding active information gathering

Active information gathering uses a direct approach to engage with our target; it involves actually making a connection between our machine and the target network and systems. By performing active information gathering, we are able to gather specific and detailed data such as live hosts, running services and application versions, network file shares, and user account information.



Performing active information gathering does pose a risk of detection.

Determining live hosts will give us an idea of the number of devices that are online. It doesn't make sense to target an offline device as it would be unresponsive. Knowing the operating system and running services on a target helps us to understand the role of that device in the network and the resources it provides to its clients.

For example, if we were to find lots of file shares on the target system during active information gathering, this could mean that the target may be a file server that has a lot of important data on its shared drive. When performing active information gathering, the attacker machine (in our case, a Kali Linux-based machine) sends special queries to the potential victim in the hope that the victim machine will respond by providing some sort of confidential information (such as network shares and service versions) in return.

Now that you have a better understanding of what active information gathering is, let's dive deep into its practices in the following sections.

DNS interrogation

As a future cybersecurity professional, understanding the purpose of various applications and network protocols is very important. In this section, we are going to focus on a particular protocol: **Domain Name System (DNS)**.

Let's begin by further understanding the role of DNS and how we can obtain information as a penetration tester.

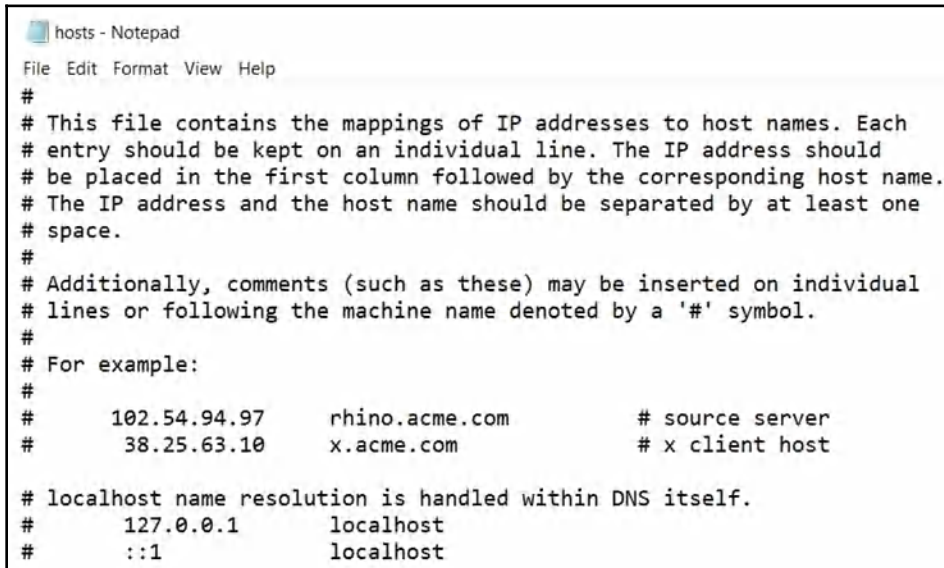
What is DNS and why do we need it on a network?

DNS is like a telephone directory containing names, addresses, and telephone numbers. DNS is used on networks—both the internal networks of organizations and external networks across the internet. The DNS protocol is used to resolve hostnames (domain names) to IP addresses.

Before DNS, each computer contained a `hosts` file located in the `C:\Windows\System32\drivers\etc` directory. This file needed to be updated frequently to ensure that users were able to reach various websites or servers by specifying their hostnames or domain names. If the `hosts` file was not present, a user needed to specify the IP address of the server they would like to visit.

All devices on a network have an assigned IP address. Remembering all of the IP addresses for each server or website you want to visit would be quite challenging. If the `hosts` file doesn't contain the most up-to-date records of new servers and websites, the user would have difficulty in reaching their destination.

The following screenshot shows current entries within the `hosts` file of a Windows operating system:



```
hosts - Notepad
File Edit Format View Help
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com       # source server
#       38.25.63.10       x.acme.com          # x client host
#
# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1              localhost
```

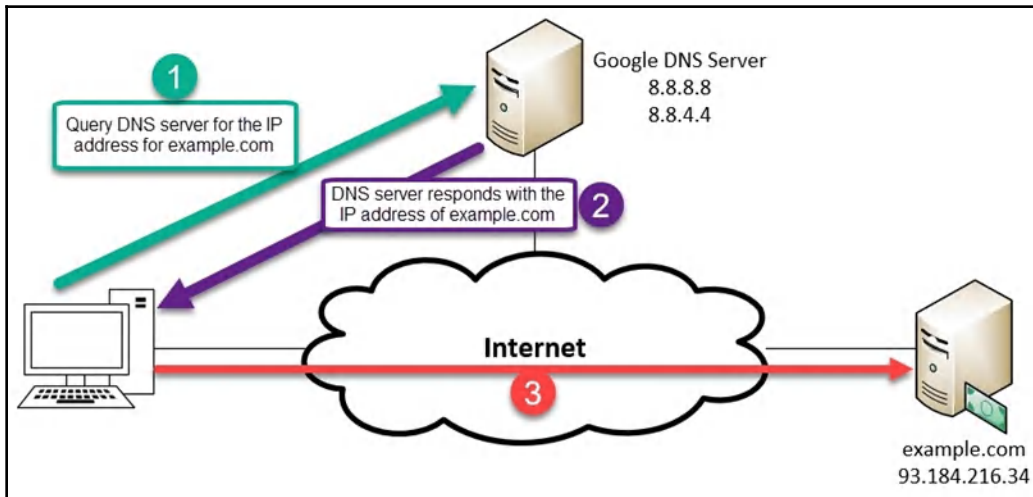
Windows hosts file record

DNS helps us to avoid depending on the `hosts` file. Many popular internet companies, such as Cisco, Google, and Cloudflare, have established public DNS servers that contain records of almost every domain name on the internet. To elaborate further, let's use a simple example to help you to understand how DNS works.

Imagine you would like to visit a website, such as `www.example.com`:

1. Whenever a computer or device needs to resolve a hostname to an IP address, it sends a DNS query message to its DNS server, as indicated in *Step 1* in the following screenshot.
2. The DNS server will check its records and respond with a DNS reply providing the client computer with the IP address of the domain, as displayed in *Step 2* in the following screenshot.

3. Finally, the client receives the IP address and establishes a session between itself and the `www.example.com` domain, as shown in the following screenshot:



DNS transactions

There are many public DNS servers on the internet; some are malicious in nature, capturing your DNS information and redirecting you to harmful websites and domains. As a result, I recommend using a trusted DNS provider on all of your networking devices and computers to improve your online safety. The following are some popular DNS servers on the internet:

- Cloudflare DNS: <https://1.1.1.1/>
- Google Public DNS: <https://developers.google.com/speed/public-dns/>
- Cisco OpenDNS: <https://www.opendns.com/>

Additionally, DNS servers not only resolve a hostname to an IP address, they also contain various records that are used for various types of resolution.

The following are the different record types:

Record Type	Description
A	Maps hostname to IPv4 address
AAAA	Maps hostname to IPv6 address
MX	Maps domain to mail server
NS	Points to domain's nameserver
CNAME	Canonical naming used for aliases of a domain
SOA	Authority for a domain
SRV	Service records
PTR	Maps an IP address to a hostname
RP	Responsible person
HINFO	Host information
TXT	Text record

DNS record types

An example of the **A** record type would be mapping the hostname of `www.example.com` to the IPv4 address `93.184.216.34`; the **AAAA** record of the same hostname would contain the IPv6 address `2606:2800:220:1:248:1893:25c8:1946`, and so on.



The `nslookup` utility is a very useful tool for validating DNS information. `nslookup` can perform various tasks, such as resolving each type of DNS record for a given domain, and it has the ability to query specific DNS servers.

DNS enumeration is the technique of probing specific DNS records for a specific organization's domain. In other words, we ask a DNS server about the IP addresses and server names for a target organization. Additionally, we attempt to perform a DNS zone transfer. A **DNS zone transfer** would allow the zone file to be copied from a master DNS server to another DNS server, such as a secondary DNS server.

However, DNS server administrators sometimes forget to apply security controls to prevent the copying of zone files to unauthorized servers. A successful DNS zone transfer can lead to a penetration tester obtaining the corporate network layout. In a worst-case scenario (for a targeted organization, that is), an organization may not separate the internal and external namespaces on their DNS servers. Such misconfigurations can lead to someone obtaining such information for malicious purposes.

In the following exercises, we are going to attempt the extraction of various DNS records for a given domain:

- DNS enumeration
- DNS zone transfer
- Using the `host` utility to perform DNS analysis
- DNS interrogation using **Fierce**

Let's dive in and have some fun with DNS and Kali Linux!

Performing DNS enumeration and zone transfer using `dnsenum`

`dnsenum` is a very simple and easy-to-use tool for enumerating and resolving DNS information for a given target. Additionally, it has the ability to automatically perform DNS zone transfers using the **nameserver** details:

1. Open a new Terminal window and execute the `dnsenum` command. The Help menu appears, providing detailed descriptions of various operators/parameters and their use.

2. Use the `dnsenum zonetransfer.me` command to perform DNS enumeration on the `zonetransfer.me` domain, as shown in the following screenshot:

```
root@kali:~# dnsenum zonetransfer.me
Smartmatch is experimental at /usr/bin/dnsenum line 698.
Smartmatch is experimental at /usr/bin/dnsenum line 698.
dnsenum VERSION:1.2.4

-----  zonetransfer.me  -----

Host's addresses:
-----
zonetransfer.me.                7199      IN      A       5.196.105.14

Name Servers:
-----
nsztml.digi.ninja.              10799     IN      A       81.4.108.41
nsztml.digi.ninja.              10799     IN      A       34.225.33.2

Mail (MX) Servers:
-----
ASPMX.L.GOOGLE.COM.            292       IN      A       173.194.68.27
ALT1.ASPMX.L.GOOGLE.COM.       292       IN      A       172.217.192.27
ASPMX2.GOOGLEMAIL.COM.         292       IN      A       172.217.192.27
ALT2.ASPMX.L.GOOGLE.COM.       292       IN      A       209.85.202.27
ASPMX3.GOOGLEMAIL.COM.         292       IN      A       209.85.202.26
ASPMX4.GOOGLEMAIL.COM.         292       IN      A       173.194.76.26
ASPMX5.GOOGLEMAIL.COM.         292       IN      A       74.125.128.26
```

dnsenum



dnsenum will attempt to obtain all of the servers and hostnames for the given domain. We are able to obtain the nameservers, mail servers (used for email exchange), and IP addresses for each server and hostname found.

3. `dnsenum` will attempt to perform a DNS zone transfer by querying the specific nameservers found during the enumeration process, as shown in the following screenshot:

```
Trying Zone Transfers and getting Bind Versions:

Trying Zone Transfer for zonetransfer.me on nsztml.digi.ninja ...
zonetransfer.me.      7200    IN      SOA      (
zonetransfer.me.      300     IN      HINFO     "Casio
zonetransfer.me.      301     IN      TXT      (
zonetransfer.me.      7200    IN      MX        0
zonetransfer.me.      7200    IN      MX        10
zonetransfer.me.      7200    IN      MX        10
zonetransfer.me.      7200    IN      MX        20
zonetransfer.me.      7200    IN      MX        20
zonetransfer.me.      7200    IN      MX        20
zonetransfer.me.      7200    IN      MX        20
zonetransfer.me.      7200    IN      A         5.196.105.14
zonetransfer.me.      7200    IN      NS        nsztml.digi.ninja.
zonetransfer.me.      7200    IN      NS        nsztml2.digi.ninja.
_sip._tcp.zonetransfer.me. 14000   IN      SRV       0
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me. 7200    IN      PTR       www.zonetransfer.me.
asfdbauthdns.zonetransfer.me. 7900    IN      AFSDB     1
asfdbbbs.zonetransfer.me. 7200    IN      A         127.0.0.1
asfdbvolume.zonetransfer.me. 7800    IN      AFSDB     1
canberra-office.zonetransfer.me. 7200    IN      A         202.14.81.230
cmdexec.zonetransfer.me. 300     IN      TXT       ";"
contact.zonetransfer.me. 2592000 IN      TXT       (
dc-office.zonetransfer.me. 7200    IN      A         143.228.181.132
deadbeef.zonetransfer.me. 7201    IN      AAAA      dead:beaf::
dr.zonetransfer.me. 300     IN      LOC       53
```

DNS zone transfer

As in the preceding snippet, the `dnsenum` tool was able to successfully extract/replicate the **master zone records** from the `nsztml.digi.ninja` nameserver. Using the information found, a penetration tester will have better insights into the target organization's (`zonetransfer.me`) internal and external network devices.

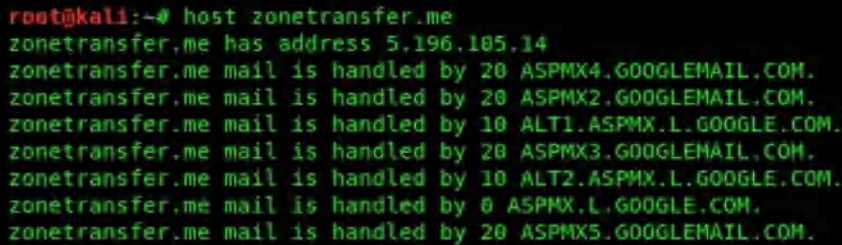
Access to sensitive information such as what we have found can potentially lead to a successful network breach in the target organization.

Up next, we will attempt to perform DNS analysis using a native Linux tool.

Using the host utility to perform DNS analysis

The `host` utility is native to the Linux operating system and can help us to obtain various DNS information about a target domain:

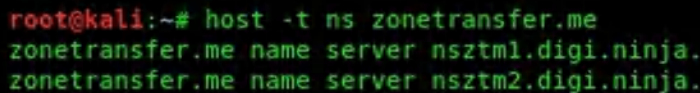
1. Open a new Terminal on Kali Linux and execute the `host zonetransfer.me` command; the `host` tool will attempt to obtain the DNS records, such as the **A** and **MX** records, for the domain:

A terminal window showing the output of the 'host zonetransfer.me' command. The output lists the IP address and several MX records for the domain.

```
root@kali:~# host zonetransfer.me
zonetransfer.me has address 5.196.105.14
zonetransfer.me mail is handled by 20 ASPMX4.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 20 ASPMX2.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 10 ALT1.ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ASPMX3.GOOGLEMAIL.COM.
zonetransfer.me mail is handled by 10 ALT2.ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 0 ASPMX.L.GOOGLE.COM.
zonetransfer.me mail is handled by 20 ASPMX5.GOOGLEMAIL.COM.
```

Retrieving DNS records using `host`

2. Use the `host -t ns zonetransfer.me` command to attempt enumeration by obtaining the nameservers for the domain. The `-t` operator allows you to specify the DNS record:

A terminal window showing the output of the 'host -t ns zonetransfer.me' command. The output lists the nameservers for the domain.

```
root@kali:~# host -t ns zonetransfer.me
zonetransfer.me name server nsztml.digi.ninja.
zonetransfer.me name server nsztm2.digi.ninja.
```

Nameserver records

3. Now that we have obtained the nameservers for the domain, let's use the information we have gathered so far. Let's attempt to perform a DNS zone transfer by querying nameservers for the domain by using the `host -l zonetransfer.me nsztml.digi.ninja` command, as shown in the following screenshot:

```
root@kali:~# host -l zonetransfer.me nsztml.digi.ninja
Using domain server:
Name: nsztml.digi.ninja
Address: 81.4.108.41#53
Aliases:

zonetransfer.me has address 5.196.105.14
zonetransfer.me name server nsztml.digi.ninja.
zonetransfer.me name server nsztml2.digi.ninja.
14.105.196.5.IN-ADDR.ARPA.zonetransfer.me domain name pointer www.zonetransfer.me.
asfdbbox.zonetransfer.me has address 127.0.0.1
canberra-office.zonetransfer.me has address 202.14.81.230
dc-office.zonetransfer.me has address 143.228.181.132
deadbeef.zonetransfer.me has IPv6 address dead:beaf::
email.zonetransfer.me has address 74.125.206.26
home.zonetransfer.me has address 127.0.0.1
internal.zonetransfer.me name server intns1.zonetransfer.me.
```

DNS zone transfer with host

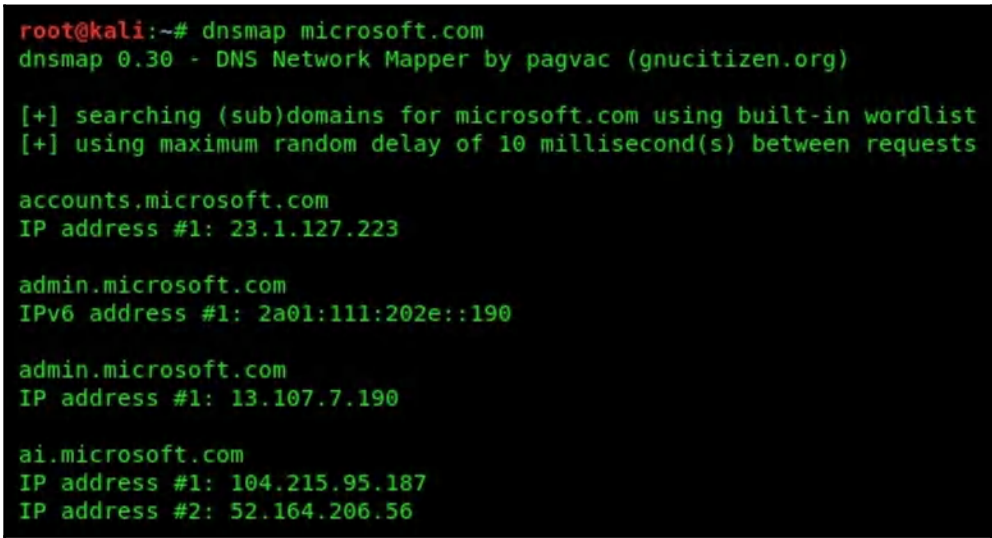
Be sure to query all nameservers for a given domain—sometimes, one server may be misconfigured even though the others are secured.

Now that you have the skills to perform DNS enumeration and zone transfers, let's attempt to discover subdomains using DNS.

Finding subdomains with dnsmap

dnsmap works a bit differently from the tools we looked at in the previous examples. dnsmap attempts to enumerate the subdomains of an organization's domain name by querying a built-in wordlist on the Kali Linux operating system. Once a subdomain has been found, dnsmap will attempt to resolve the IP address.

Using the `dnsmap microsoft.com` command, we are able to find subdomains for the organization and their corresponding IP addresses:



```
root@kali:~# dnsmap microsoft.com
dnsmap 0.30 - DNS Network Mapper by pagvac (gnucitizen.org)

[+] searching (sub)domains for microsoft.com using built-in wordlist
[+] using maximum random delay of 10 millisecond(s) between requests

accounts.microsoft.com
IP address #1: 23.1.127.223

admin.microsoft.com
IPv6 address #1: 2a01:111:202e::190

admin.microsoft.com
IP address #1: 13.107.7.190

ai.microsoft.com
IP address #1: 104.215.95.187
IP address #2: 52.164.206.56
```

dnsmap results

As mentioned in a previous section, discovering the subdomains of an organization can lead to finding hidden and sensitive portals and directories in a domain.

As you may have noticed, each tool we have used so far gives us a bit more detail. In the next section, we will use a more aggressive tool to help us to extract more details about a target domain.

DNS interrogation using Fierce

Fierce is considered a semi-lightweight DNS interrogation tool. It performs extensive lookups on IP spaces and hostnames for a given target domain. To use Fierce, we can execute the `fierce -dns example.com` command, as shown in the following screenshot:

```
root@kali:~# fierce -dns microsoft.com
DNS Servers for microsoft.com:
    ns3.msft.net
    ns4.msft.net
    ns1.msft.net
    ns2.msft.net

Trying zone transfer first...
Testing ns3.msft.net
    Request timed out or transfer not allowed.
Testing ns4.msft.net
    Request timed out or transfer not allowed.
Testing ns1.msft.net
    Request timed out or transfer not allowed.
Testing ns2.msft.net
    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)...
134.170.188.221 agent.microsoft.com
134.170.185.46 agent.microsoft.com
104.215.95.187 ai.microsoft.com
52.164.206.56 ai.microsoft.com
13.77.161.179 asia.microsoft.com
104.215.148.63 asia.microsoft.com
```

Fierce DNS interrogation

Fierce will attempt to obtain all of the DNS records for a given domain and discover any subdomains with their corresponding IP addresses. This tool may take some time to complete its interrogation, as it implements an in-depth analysis of the target domain.

We have now completed the exercises in this section. Next, we will directly engage the target to gather more specific details using various scanning techniques.

Scanning

Let's take our information gathering phase a bit further than we have done before. In this section, we are going to perform various scan types on a target. These will include the following:

- Ping sweep
- Operating system and service version detection

- Scanning for host devices that have ICMP disabled
- Performing stealth scanning
- Scanning UDP ports using Nmap
- Performing evasion scanning techniques using Nmap

The objective of scanning is to identify live hosts on a network, determine open and closed ports on a system, identify running services on a target, and create a network diagram of the target's network infrastructure. The information obtained during the network-scanning phase is key in creating a profile of a target organization.



Scanning a target without permission is illegal in many countries. For this reason, we will be scanning devices within our lab.

Within a packet, there are many types of TCP flag that are used during communication between two or more hosts on a network. As a penetration tester, we can leverage certain vulnerabilities within the TCP/IP stack while performing our network scans. In other words, we are going to send specially crafted flags to a target to determine their port status, operating system, the services running, and their versions; we'll also to determine whether a firewall is monitoring inbound or outbound traffic, and so on.

The following TCP flags are within a packet:

- **URG: (Urgent)** Indicates this packet should be processed immediately
- **PSH: (Push)** Sends buffered data immediately
- **FIN: (Finish)** Indicates there are no more transmissions to be sent
- **ACK: (Acknowledgement)** Confirms receipt of a message
- **RST: (Reset)** Resets a network connection
- **SYN: (Synchronization)** Used to initialize a connection between host devices

By using a tool such as Wireshark (www.wireshark.org), you can observe every detail within packets on a network.

The following snippet is a capture of a network packet where the ACK flag is set:

```
> Frame 10: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: Vmware_b6:b5:48 (00:0c:29:b6:b5:48), Dst: Vmware_f3:f2:f6 (00:50:56:f3:f2:f6)
> Internet Protocol Version 4, Src: 172.16.254.128 (172.16.254.128), Dst: gstaticadssl.l.google.com (216.58.208.227)
▼ Transmission Control Protocol, Src Port: 52182 (52182), Dst Port: https (443), Seq: 1, Ack: 1, Len: 0
  Source Port: 52182 (52182)
  Destination Port: https (443)
  [Stream index: 2]
  [TCP Segment Len: 0]
  Sequence number: 1 (relative sequence number)
  [Next sequence number: 1 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  ▼ Flags: 0x010 (ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    ....0... = Congestion Window Reduced (CWR): Not set
    ....0... = ECN-Echo: Not set
    ....0... = Urgent: Not set
    ....1... = Acknowledgment: Set
    ....0... = Push: Not set
    ....0... = Reset: Not set
    ....0... = Syn: Not set
    ....0... = Fin: Not set
    [TCP Flags: .....A....]
  Window size value: 64240
  [Calculated window size: 64240]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x53ca [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

ACK Flag is used in this packet

A packet with the ACK flag enabled

Additionally, by observing the details in the packet, you can see the source and destination MAC addresses, IP addresses, ports, and other important characteristics. Wireshark is considered to be one of the best network protocol analyzers and sniffers among network and cybersecurity professionals alike.

Now that we understand the importance of scanning, let's learn about one of the most popular scanning tools in the industry, Nmap.

Nmap

Nmap is free and is one of the most powerful network scanning tools available for both Windows and Linux platforms. Nmap can help both network administrators and cybersecurity professionals in many ways.

Nmap features include the following:

- Creating a network inventory
- Checking for live hosts
- Determining operating systems
- Determining running services and their version
- Identifying vulnerabilities on a host
- Detecting sniffers
- Determining whether a firewall is present on a network

We will go over, to begin with, the basics of Nmap and move gradually on to advanced scanning techniques. As penetration testers, we must ensure that we have an arsenal of tools that will help us to perform our jobs efficiently. However, as professionals, we must also ensure that we are very familiar with, and know how to use, each tool available to us.

So, we are going to start by performing a basic scan on a target:

1. Let's begin by opening a new Terminal and using the following syntax: `nmap <target IP or hostname>`.
2. We are going to scan a website that has given us legal permission to perform a scan. Let's use the `nmap scanme.nmap.org` command:

```
root@kali:~# nmap scanme.nmap.org
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-22 12:27 AST
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.12s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
9929/tcp  open  nping-echo
31337/tcp open  Elite
Nmap done: 1 IP address (1 host up) scanned in 9.99 seconds
```

Nmap scan 1



By performing a regular scan on a target or network, Nmap checks the 1,000 most commonly used TCP/IP ports on the target.

3. Observing the output, Nmap was able to identify the open ports, determine whether the open ports are TCP or UDP, identify the application layer protocols, and find out the IP addresses (IPv4 and IPv6) of the target.

Identifying open ports on a target is like discovering an open door into the system, and identifying services can help us to narrow our scope in searching for, and exploiting, vulnerabilities.



To perform a scan on an IPv6 address, you can include the `-6` operator, as in: `nmap -6 2600:3c01::f03c:91ff:fe18:bb2f`.

Nmap isn't that difficult, right? Let's take a few more steps with Nmap in the upcoming sections.

Performing a ping sweep with Nmap

At times, you may need to identify all live hosts on a network during a penetration test. Nmap is able to perform a ping sweep across multiple targets, whether specifying a range or an entire subnet. Using the `-sn` operator will allow you to perform a ping scan only on the target:

```
root@kali:~# nmap -sn 10.10.10.0/24
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-22 13:03 AST
Nmap scan report for 10.10.10.1
Host is up (0.00034s latency).
MAC Address: 00:0C:29:2B:29:7F (VMware)
Nmap scan report for 10.10.10.14
Host is up (0.00027s latency).
MAC Address: 00:0C:29:A0:80:6A (VMware)
Nmap scan report for 10.10.10.15
Host is up (0.00019s latency).
MAC Address: 00:0C:29:53:2A:EB (VMware)
Nmap scan report for 10.10.10.19
Host is up (0.00033s latency).
MAC Address: 00:0C:29:24:BE:4F (VMware)
Nmap scan report for 10.10.10.100
Host is up (0.00024s latency).
MAC Address: 00:0C:29:28:78:DB (VMware)
Nmap scan report for 10.10.10.16
Host is up.
Nmap done: 256 IP addresses (6 hosts up) scanned in 34.91 seconds
```

Ping sweep with Nmap

In the preceding snippet, Nmap has presented only the hosts that it thinks are alive on the network segment and was able to look up the MAC addresses of each host to determine the vendor.



- If you would like to perform a range scan, you can use the following syntax: `nmap start ip addr - end ip addr`.
- If you would like to scan specific IP devices on a network, use the following syntax: `nmap host1 host2 host3`.
- Nmap also has support for scanning hosts that are listed within a text file by using the following syntax: `nmap -iL file.txt`.

Let's now take things up a notch and learn more about how to use Nmap in the following section.

Obtaining operating system and service versions using Nmap

So far, we have been able to gather basic details about a target. We can use Nmap to help users determine the operating system, the operating system version, and the service versions of any running applications on a target.

Using the `-A` operator will initiate an aggressive scan, `-O` will profile the operating system, and `-sV` will identify service versions.



Performing an aggressive type of scan can potentially be flagged by an **Intrusive Detection System/Intrusive Prevention System (IDS/IPS)** or a firewall appliance. Be wary of this, as a big part of penetration testing is being as silent as possible to avoid detection.


Using the `nmap -A -O -sV target` command on our Metasploitable VM as our target system, we will be able to obtain much more meaningful information about the target.

As you can see in the following snippet, for each port that is open, Nmap has identified a particular service operating on the port, and we were able to retrieve the application service version details as well:

```

root@kali:~# nmap -A -O -sV 10.10.10.100
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-22 15:57 AST
Nmap scan report for 10.10.10.100
Host is up (0.00013s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (code 230)
|_ftp-syst:
|   STAT:
|   FTP server status:
|   | Connected to 10.10.10.16
|   | Logged in as ftp
|   | TYPE: ASCII
|   | No session bandwidth limit
|   | Session timeout in seconds is 300
|   | Control connection is plain text
|   | Data connections will be plain text
|   | vsFTPd 2.3.4 - secure, fast, stable
|_End of status
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)

```



Operating system and service version


Scrolling down a bit more on the output, we can see that, by using the `-O` parameter, Nmap was able to determine the type of operating system:

```

Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
Network Distance: 1 hop
Service Info: Hosts: metasploitable.localdomain, localhost, irc.Metasploitable.LAN; OSs: Unix, Linux;

Host script results:
|_clock-skew: mean: 2h00m01s, deviation: 2h49m53s, median: -6s
|_nbstat: NetBIOS name: METASPLOITABLE, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)
|_smb-os-discovery:
|   OS: Unix (Samba 3.0.20-Debian)
|   NetBIOS computer name:
|   Workgroup: WORKGROUP\x00
|   System time: 2019-04-22T15:59:22-04:00
|_smb2-time: Protocol negotiation failed (SMB2)

```



Detecting the kernel version

At this point, we have a much better idea of our target, the Metasploitable VM. We know all of the open ports, services, and service versions that are currently running, as well as the operating system.

Nmap is awesome, isn't it? Let's learn how to use Nmap to scan a device that has ICMP disabled.

Scanning host devices with ICMP disabled

When Nmap is about to perform a scan on a host, it sends a ping packet to the host to determine whether the target is alive. If the target does not respond, Nmap will not attempt to execute the scan. However, system administration and cybersecurity professionals usually disable **Internet Control Message Protocol (ICMP)** responses on servers. Not receiving an ICMP echo reply from a target would indicate that the target device is down/offline; however, this technique sets out to basically trick a novice hacker into thinking the host is simply not available. Using the `-Pn` operator during an Nmap scan will skip the host discovery phase and treat the target as online.

The following is an example:

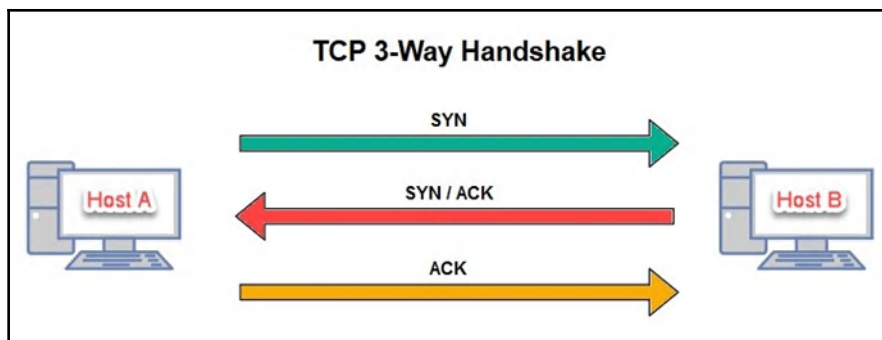
```
nmap -Pn 10.10.10.100
```

During a penetration test, if you are not able to discover live hosts on the network, don't be overly concerned as network security professionals tend to apply security controls to their end devices and networks. Nmap can detect hidden systems, bypassing firewalls and network sniffers to detect security vulnerabilities on a host.

When performing a scan, there's a high possibility that the target will know a port scan is being done by an attacker or a penetration tester. In the next section, we will describe how to perform a stealth scan using Nmap.

Performing a stealth scan using Nmap

By default, Nmap establishes a **TCP three-way handshake** on any open TCP ports found. After the handshake has been established, the messages are exchanged. The following snippet displays the handshake process, where **Host A** wants to communicate with **Host B**:

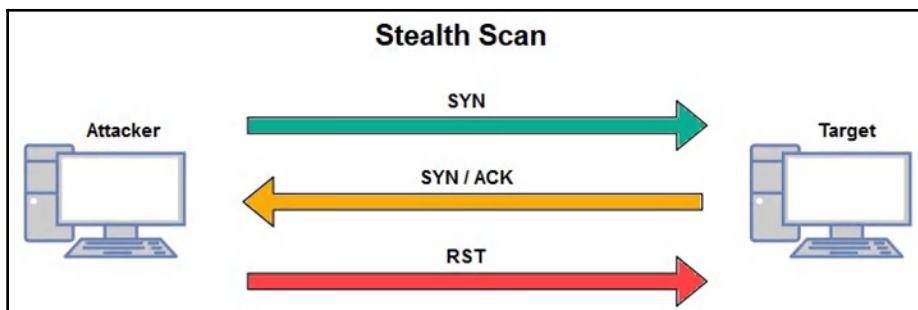


TCP three-way handshake

During a penetration test, we need to remain as stealthy as possible on the network. This creates the effect of an actual hacker attempting to compromise the system/network without being caught by the organization's security controls and systems. By establishing a TCP three-way handshake with our target devices, we are making ourselves known to the target.

Therefore, we are going to perform a stealth scan (half-open) using Nmap. A stealth scan does not establish a full TCP handshake with the target:

1. The attacker machine tricks the target by sending a TCP **SYN** packet to a particular port on the target if the port is open on the target.
2. A TCP **SYN/ACK** packet is returned to the attacker machine.
3. Lastly, the attacker sends a TCP **RST** packet to reset the connection state on the target:



Stealth scan

In our exercise, we are going to probe port 80 on our Metasploitable VM using stealth scanning with Nmap. Using the `-sS` operator to indicate a stealth scan, and with the `-p` operator scanning (probing) a particular port, we can execute the `nmap -sS -p 80 10.10.10.100` command on our Kali Linux machine:

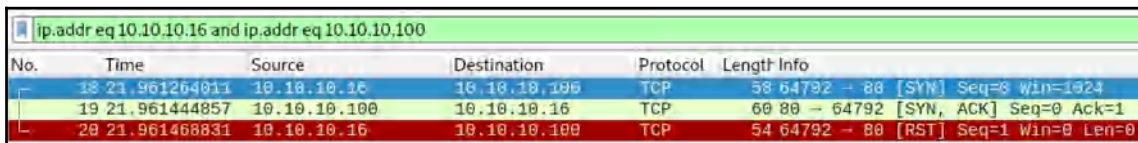
```
root@kali:~# nmap -sS -p 80 10.10.10.100
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-23 10:58 AST
Nmap scan report for 10.10.10.100
Host is up (0.00023s latency).

PORT      STATE SERVICE
80/tcp    open  http
MAC Address: 00:0C:29:28:78:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 16.69 seconds
```

A stealth scan using Nmap

Using Wireshark, we are able to see the flow of packets between our Kali Linux machine and the target. Packet number **18** indicates that an **[SYN]** packet was sent to the Metasploitable VM, packet number **19** indicates that an **[SYN, ACK]** packet was returned to the Kali Linux machine, and finally, packet number **20** indicates that our Kali Linux machine sent an **[RST]** packet to reset the connection:



No.	Time	Source	Destination	Protocol	Length	Info
18	21.961264811	10.10.10.16	10.10.10.100	TCP	58	64792 → 80 [SYN] Seq=8 win=1324
19	21.961444857	10.10.10.100	10.10.10.16	TCP	60	80 → 64792 [SYN, ACK] Seq=0 Ack=1
20	21.961468831	10.10.10.16	10.10.10.100	TCP	54	64792 → 80 [RST] Seq=1 Win=0 Len=0

Stealth scan detected in Wireshark

The final result is that we were able to successfully probe a given port on a target system and did not establish a network session between our machine and the target.

There are many services and protocols that use UDP as a preferred transportation method. UDP applications do not respond to a typical port scan by default. Whenever you perform a network/port scan using Nmap, the scanning engine searches for open TCP ports by default; this means UDP ports are usually missed in the results. In the next section, we'll take a look at performing a UDP port scan.

Scanning UDP ports using Nmap

There are many application layer protocols that use **User Datagram Protocol (UDP)** as their preferred transport protocol. Using the `-sU` operator will indicate the need to perform a UDP port scan on a given target. Using the following command, we can achieve this task:

```
nmap -sU target
```

We have now acquired the skills to perform a UDP scan on a target device or network. In the next section, we will take a look at evading security appliance and detection using Nmap.

Evading detection using Nmap

Whenever a packet is sent from one device to another, the source IP address is included within the header of the packet. This is the default behavior of the TCP/IP stack; all address details must be included within all packets that need to traverse a network. In performing a network scan on a target, our source IP address is included within all packets that our machine, Kali Linux, sends to the target.

Nmap has the capability of using decoys to trick the target into believing that the network scans are originating from multiple sources rather than a single source IP address. The `-D` operator is followed by random IP addresses, which are the decoys. Let's assume we want to scan an IP address, `10.10.10.100`, and set three decoys: `10.10.10.14`, `10.10.10.15`, and `10.10.10.19`. We can use the following command:

```
nmap -sS 10.10.10.100 -D 10.10.10.14, 10.10.10.15, 10.10.10.19
```

Observing the following Wireshark capture, we can see that packets containing both our source IP address and the decoys' IP addresses were used during the port scan on our target:

No.	Time	Source	Destination	Protocol	Length	Info
15	16.8193233...	10.10.10.16	10.10.10.100	TCP	58	48518 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
16	16.8193487...	10.10.10.14	10.10.10.100	TCP	58	48518 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
17	16.8193836...	10.10.10.15	10.10.10.100	TCP	58	48518 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
18	16.8194016...	10.10.10.19	10.10.10.100	TCP	58	48518 → 80 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
19	16.8195510...	10.10.10.100	10.10.10.16	TCP	60	80 → 48518 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
20	16.8195749...	10.10.10.16	10.10.10.100	TCP	54	48518 → 80 [RST] Seq=1 Win=0 Len=0

Detecting decoys in Wireshark

However, an **RST** packet is sent from the actual source address. Additionally, we can use other operators such as `--spoof-mac` to spoof the source MAC address.

In the next section, we will learn how to evade firewall detection while performing a network scan using Nmap.

Evading firewalls with Nmap

During your career as a cybersecurity professional, penetration tester, or ethical hacker, you'll often encounter organizations—be they small, medium, or large enterprises—that have some sort of firewall appliance or software on their network infrastructure.

Firewalls can prevent network scans and create a challenge for us as penetration testers. The following are various operators that can be used in Nmap to evade firewalls:

Operator	Description
-f	Used to fragment probes by creating 8-byte packets
-D RND: 10	Generates 10 random decoys
--source-port	Allows you to spoof your source port. Eg. <code>--source-port 123</code>
--source-mac	Allows you to spoof your source MAC address
--mtu	Used to specify a custom Maximum Transmission Unit (MTU)

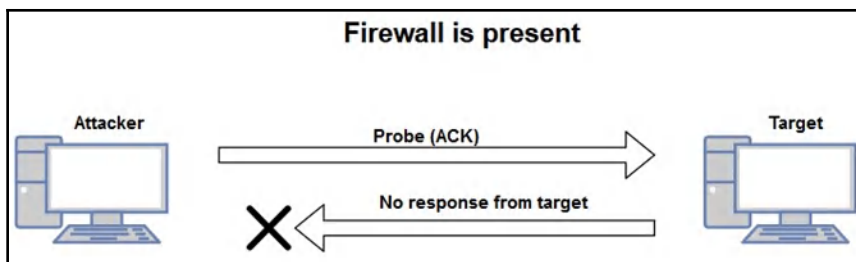
Nmap's firewall evasion operators

Additionally, we can send custom probes with specific flags to a target and analyze the responses.

In the following sections, we'll take a look at how to determine whether a stateful firewall is present on a network.

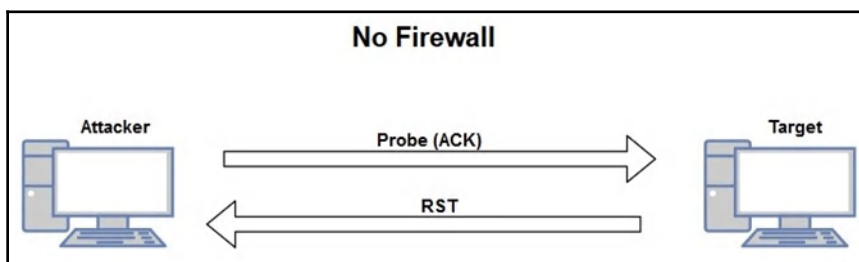
Checking for a stateful firewall

In checking for a stateful firewall, we can send a probe to the target with the **ACK** flag enabled. If no response is provided from the target, this would indicate that a firewall is present:



Stateful firewall present

However, if a packet is returned with the **RST** flag set, this would indicate that there is no firewall on the target system:



Stateful firewall is not present

We can use the `-sA` operator on Nmap to perform an ACK scan on a target. Let's perform a scan on our Metasploitable VM to determine whether port 80 is open, and whether the system has a firewall present:

1. Use the `nmap -sA -p 80 <target>` command:

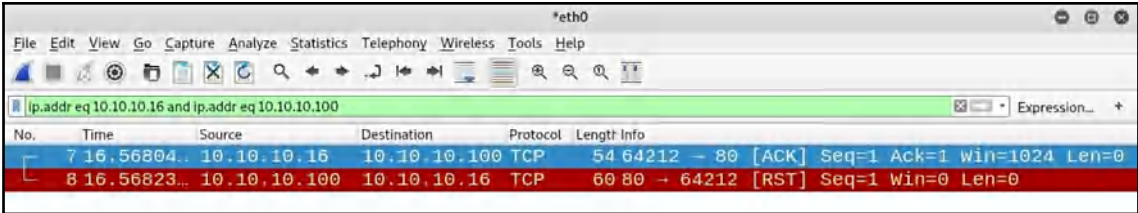
```
root@kali:~# nmap -sA -p 80 10.10.10.100
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-23 12:50 AST
Nmap scan report for 10.10.10.100
Host is up (0.00021s latency).

PORT      STATE      SERVICE
80/tcp    unfiltered http
MAC Address: 00:0C:29:28:78:DB (VMware)

Nmap done: 1 IP address (1 host up) scanned in 16.69 seconds
```

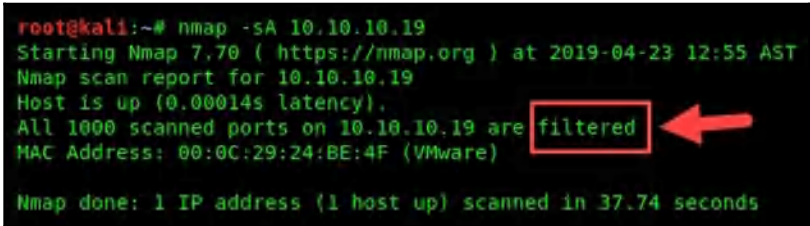
ACK scan using Nmap

- 2. We were able to identify port 80 as opened and unfiltered (no firewall) on the target. Additionally, by observing the packets, we saw that an **RST** packet was returned to our Kali Linux (attacker) machine:



The port scan shown in Wireshark

Whenever you run a scan on a target and the results indicate *filtered*, this means there is a firewall present and that it's actively monitoring the port, as shown in the following screenshot:



Detecting a filtered port using Nmap

Additionally, the following operators can be used to determine whether a firewall is present on a system:

Operator	Description
-sX	Performs an XMAS scan. The URG, FIN, and PSH flags are all set.
-sF	Performs a FIN scan. Only the FIN flag is set.
-sN	Performs a Null scan. No flags are set.

Additional Nmap operators

Nmap will interpret the responses and determine whether the ports on a target are filtered or unfiltered.

Having completed this section, you are now able to use Nmap to profile a target. In the next section, we will learn about the **Nmap Scripting Engine (NSE)**.

NSE scripts

NSE is one of the most powerful features within Nmap. It allows users to create and automate scripting to perform customized scans on a target device. By performing scans using various Nmap scripts, you can quickly detect whether your target is susceptible to a known vulnerability, malware, open backdoors, and so on.

The following are the main categories of NSE scripts:

Category	Description
All	Run all NSE Scripts
Auth	Authentication Scripts
Default	Executes basic scripts during NMap scan
Discovery	Run scripts which will help to obtain indepth information about a target
External	Run scripts which communicate with OSINT sources
Intrusive	Run intrusive scripts against the target
Malware	Run scripts which will check for backdoors and malware
Safe	Run basic scripts which are not intrusive
Vuln	Check for common vulnerabilities in a target

NSE categories

To execute an entire category of scripts, we can use the `--script category` command. The following snippet is an example of using the `vuln` category of scripts during an Nmap scan:

```

root@kali:~# nmap --script vuln 10.10.10.100
Starting Nmap 7.70 ( https://nmap.org ) at 2019-04-24 12:26 AEST
Pre-scan script results:
| broadcast-avahi-dos:
|   Discovered hosts:
|   | 224.0.0.251
|   | After NULL UDP avahi packet DoS (CVE-2011-1002).
|   | Hosts are all up (not vulnerable).
|_
Nmap scan report for 10.10.10.100
Host is up (0.000097s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
| ftp-vsftpd-backdoor:
|   VULNERABLE:
|   | vsFTPD version 2.3.4 backdoor
|   | State: VULNERABLE (Exploitable)
|   | IDs: OSVDB:73573 CVE:CVE-2011-2523
|   | vsFTPD version 2.3.4 backdoor, this was reported on 2011-07-04.
|   | Disclosure date: 2011-07-03

```

Vulnerability found using NSE

Running an entire category of scripts may not always be suitable for various situations. If you are performing a scan to search for systems that contain a specific vulnerability, such as **vsFTPD 2.3.4 backdoor (CVE-2011-2523)**, you can use the following command:

```
nmap --script ftp-proftpd-backdoor target
```

Each NSE script is stored locally on Kali Linux in the `/usr/share/nmap/scripts` directory. However, you should become familiar with using NSE scripts, as this will help you to save time and find specific information about a target much more quickly. To help you to further understand NSE scripts, please visit the official NSE documentation website at <https://nmap.org/nsedoc/>. The repository contains a detailed description of each NSE script available.

Having completed this section on Nmap and NSE, let's now learn about Zenmap, the GUI version of Nmap.

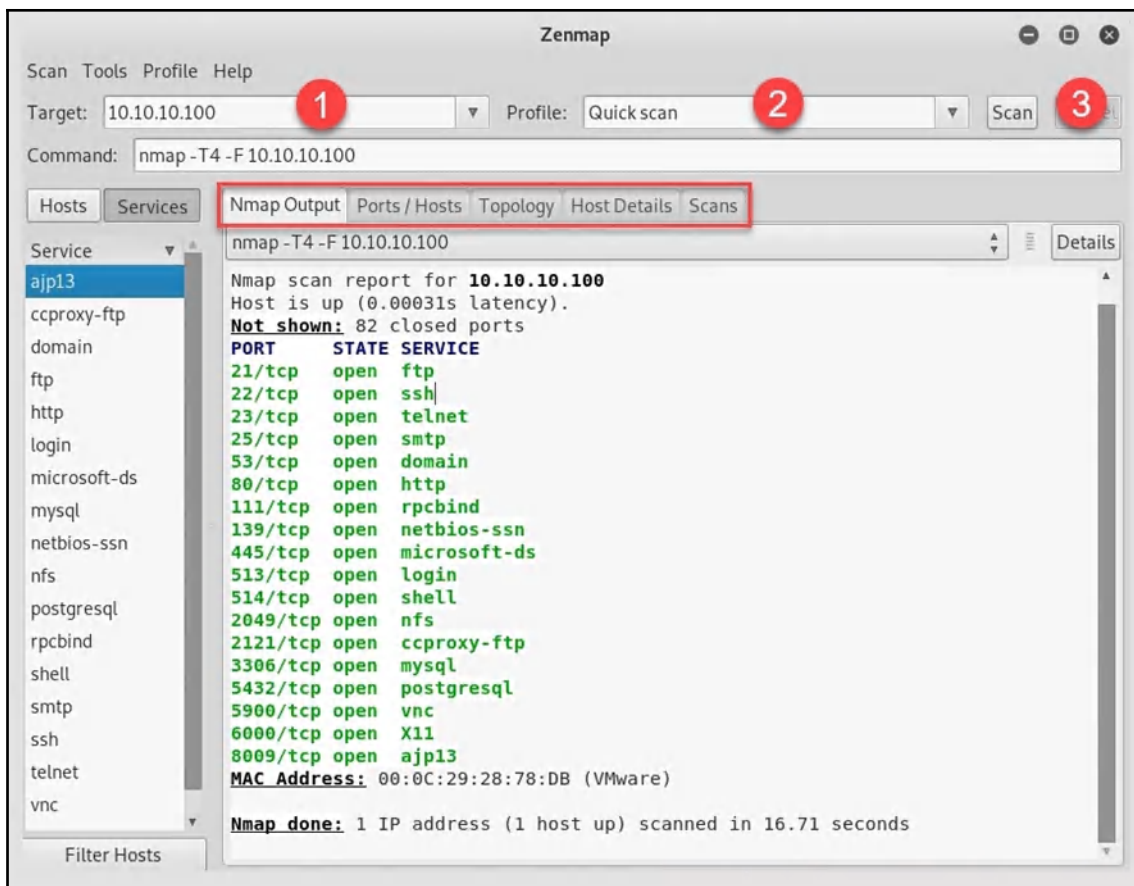
Zenmap

Zenmap is the **graphical user interface (GUI)** version of Nmap and is supported on multiple platforms, such as Windows, Linux, and macOS. The creation of Zenmap was geared toward beginners as it's easier to use than the traditional command-line interface of Nmap. To download Zenmap on your system, please visit <https://nmap.org/zenmap/>.

The following shows the Zenmap interface. It's quite simple to use: simply enter the target and select the type of scan you would like to perform. Depending on the type of scan you select, the necessary Nmap operators will be set in the command field.

To demonstrate, let's perform a quick scan on our Metasploitable VM by observing the following steps:

1. Enter the IP address of our target.
2. Select the **Quick scan** option from the **Profile** menu.
3. Click on **Scan** to begin, as shown in the following screenshot:

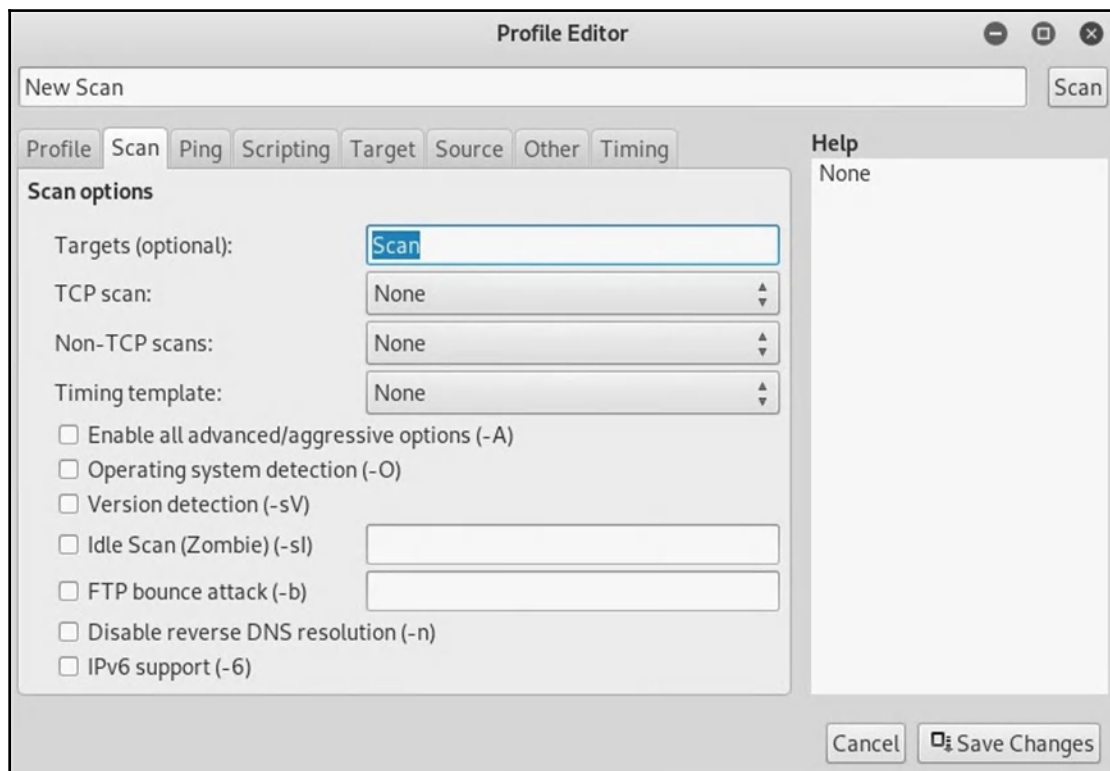


The Zenmap interface

Once the scan has completed, click on each tab to get further details about the target. If you're performing a scan on an entire network, the **Topology** tab will help you create a network diagram of the target network.

Customized scanning profiles can be created on Zenmap by performing the following steps:

1. To create a new scanning profile, click on **Profile | New Profile or Command**.
2. The **Profile Editor** will open, providing you with all of the options available for scanning in Nmap, as shown in the following screenshot:



Zenmap Profile Editor

Be sure to visit each tab and familiarize yourself with the various options available, since they will be useful in the future.

As you saw, Zenmap is very easy to use and user-friendly. In the next section, we will learn about Hping3, another tool with which to perform scanning.

Hping3

Hping3 is a command-line tool that allows a user to analyze TCP/IP messages on a network. Additionally, Hping3 allows use to assemble network packets, which can be beneficial to a penetration tester in performing device and service discovery and offensive actions, such as a **Denial-of-Service (DoS)** attack.

Hping3 is a tool that can perform the following tasks:

- Host discovery on a network
- Fingerprinting host devices to determine services
- Sniffing network traffic
- Flooding packets (DoS)
- File transfer

As mentioned in the previous section, there are many servers and devices that have ICMP responses disabled as a security precaution. We can use Hping3 to probe a port on a target system to force an ICMP response back to our attacker machine.

To get started using Hping3, let's use the following steps to perform a port scan on port 80:

1. We use the `ping` utility to send four ICMP echo request messages to our Windows Server machine (firewall enabled and ICMP disabled):

```
root@kali:~# ping 10.10.10.14 -c 4
PING 10.10.10.14 (10.10.10.14) 56(84) bytes of data.

--- 10.10.10.14 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 58ms
```

Pinging a target

- Our attacker machine (Kali Linux) did not receive any responses from the target. A novice hacker would assume the target is offline and would probably move on. However, using Hping3 to probe a specific port by sending SYN packets will force the target to reveal itself. Using the `hping3 -S target ip addr -p port -c 2` syntax, we get the following responses:

```
root@kali:~# hping3 -S 10.10.10.14 -p 80 -c 2
HPING 10.10.10.14 (eth0 10.10.10.14): S set, 40 headers + 0 data bytes
len=46 ip=10.10.10.14 ttl=128 DF id=2493 sport=80 flags=5A seq=0 win=65392 rtt=7.8 ms
len=46 ip=10.10.10.14 ttl=128 DF id=2495 sport=80 flags=5A seq=1 win=65392 rtt=7.8 ms

--- 10.10.10.14 hping statistic ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 7.8/7.8/7.8 ms
```

Port scan using Hping3

By looking at our results, we can see we have received successful responses from our target. This means that the `10.10.10.14` device is online and that port 80 is open.



The `-s` operator indicates the sending of SYN packets, `-p` allows you to specify destination port numbers, and `-c` specifies the number of packets to be sent.

- Additionally, we can take this step a bit further by performing port scanning on a range of network ports on a target device. Using the `hping3 -s 20-1000 -S 10.10.10.14` command, we are able to perform an SYN scan on a range of ports from 20-1000 on our target. The following snippet indicates that ports 80, 135, 139, 445, 902, and 912 are open on our target:

```
root@kali:~# hping3 -s 20-1000 -S 10.10.10.14
Scanning 10.10.10.14 (10.10.10.14), port 20-1000
981 ports to scan, use -V to see all the replies
+-----+-----+-----+-----+-----+
|port| serv name | flags | ttl | id | win | len |
+-----+-----+-----+-----+-----+
| 80 | http      | :S..A... | 128 | 17675 | 65392 | 46 |
| 135 | epmap     | :S..A... | 128 | 28683 | 65392 | 46 |
| 139 | netbios-smb | :S..A... | 128 | 29963 | 8192 | 46 |
| 445 | microsoft-ds | :S..A... | 128 | 49163 | 65392 | 46 |
| 902 |           | :S..A... | 128 | 49419 | 65392 | 46 |
| 912 |           | :S..A... | 128 | 49675 | 65392 | 46 |
All replies received. Done.
```

Stealth scan using Hping3

There are many more operators that can be combined when using Hping3; please be sure to check out the Help menu using the `hping3 -h` command on the Terminal.

Now that you are familiar with using Hping3 as a scanner, let's take a deep dive into performing enumeration on a target device.

SMB, LDAP enumeration, and null sessions

In this section, we are going to take a look at using various application protocols to help us extract sensitive data and records from a target system.

SMBmap and SMBclient

SMBmap is a popular and easy-to-use tool that is used to help us discover any SMB shares on a device and detect permissions on any shares found:

1. Using the `smbmap -H target syntax`, we can attempt to perform a port scan, looking for ports that are used by the SMB service; in our target, it's 445 and it's open:

```
root@kali:~# smbmap -H 10.10.10.100
[+] Finding open SMB ports....
[+] User SMB session established on 10.10.10.100...
[+] IP: 10.10.10.100:445      Name: 10.10.10.100
    Disk
    ----
    print$  <----- Permissions
    tmp      READ, WRITE
    opt      NO ACCESS
    IPC$     NO ACCESS
    ADMIN$   NO ACCESS
```

SMB shares

2. SMBmap will attempt to establish a session between the attacker machine and the target on port 445 to enumerate any share drives and folders. On our target (Metasploitable), there's the `tmp` folder, which gives us read and write permissions.
3. Using the `smbmap -H 10.10.10.100 -r tmp` command, we will be able to list the contents of the directory specified. In our example, we are listing the content of the `tmp` folder, as shown in the following screenshot:

```
root@kali:~# smbmap -H 10.10.10.100 -r tmp
[+] Finding open SMB ports....
[+] User SMB session establishd on 10.10.10.100...
[+] IP: 10.10.10.100:445      Name: 10.10.10.100
    Disk
    ----
    tmp                                Permissions
    ./                                READ, WRITE
    dr--r--r--      0 Fri Apr 19 15:11:31 2019  .
    dw--w--w--      0 Sun May 20 14:36:11 2012  ..
    dr--r--r--      0 Fri Apr 19 15:01:04 2019  .ICE-unix
    fw--w--w--      0 Fri Apr 19 15:02:06 2019  5117.jsvc_up
    dr--r--r--      0 Fri Apr 19 15:01:30 2019  .X11-unix
    fw--w--w--     11 Fri Apr 19 15:01:30 2019  .X0-lock
root@kali:~#
```

SMBmap enumeration

SMBmap is an excellent tool for enumerating SMB shares on target devices; however, it's always good to have another tool available in your arsenal. Other tools include SMBlookup, SMBclient, and Nmap.



Further information about SMBmap can be found at: <https://tools.kali.org/information-gathering/smbmap>.

SMBclient is another handy tool and works in a similar fashion to SMBmap. To enumerate SMB services on a target, we can use the `smbclient -L //target` command:

```
root@kali:~# smbclient -L //10.10.10.100
WARNING: The "syslog" option is deprecated
Enter WORKGROUP\root's password:
Anonymous login successful
```

Sharename	Type	Comment
print\$	Disk	Printer Drivers
tmp	Disk	oh noes!
opt	Disk	
IPC\$	IPC	IPC Service
ADMIN\$	IPC	IPC Service

```
Reconnecting with SMB1 for workgroup listing.
Anonymous login successful
```

Server	Comment
Workgroup	Master
WORKGROUP	METASPLOITABLE

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

SMBclient enumeration

SMBclient will attempt to extract any shares on the target device, as seen in the previous screenshot. Further information on SMBclient can be found at: <https://www.samba.org/samba/docs/current/man-html/smbclient.1.html>.

Having completed this section, you have gained the skills to use both SMBmap and SMBclient to perform SMB enumeration on a target. In the next section, we will briefly discuss another popular tool for SMB enumeration, Enum4linux.

Enum4linux

Enum4linux is an enumeration tool capable of detecting and extracting data from Windows and Linux operating systems, including those that are **Samba** (SMB) hosts on a network. Enum4linux is capable of discovering the following:

- Password policies on a target
- The operating system of a remote target

- Shares on a device (drives and folders)
- Domain and group membership
- User listings

To scan a target, use the following command: `enum4linux target`. The tool will perform all the checks and enumeration that it can perform. The output can be a bit overwhelming at first; be sure to check the details carefully as they will contain meaningful information about your target.

Enum4linux comes in handy at times for performing a scan on the network to discover any shared resources. In the next section, we will take a deep dive into LDAP enumeration on a Windows network.

LDAP enumeration

The **Lightweight Directory Access Protocol (LDAP)** is used to query a database or directory type of service. A common example is a corporate environment with an **Active Directory (AD)** server that manages the user accounts of the entire organization. End devices such as desktop computers need to query the AD server each time a user is attempting to log in to that desktop computer.

LDAP uses port 389 by default; however, packets are sent across the network in plaintext. Additionally, using **LDAPS (LDAP Secure)** ensures that the information sent between a client and the LDAP server is encrypted by default; LDAPS uses port 636 by default. We can use Nmap to scan for devices on a network that has ports 389 and 636 open.

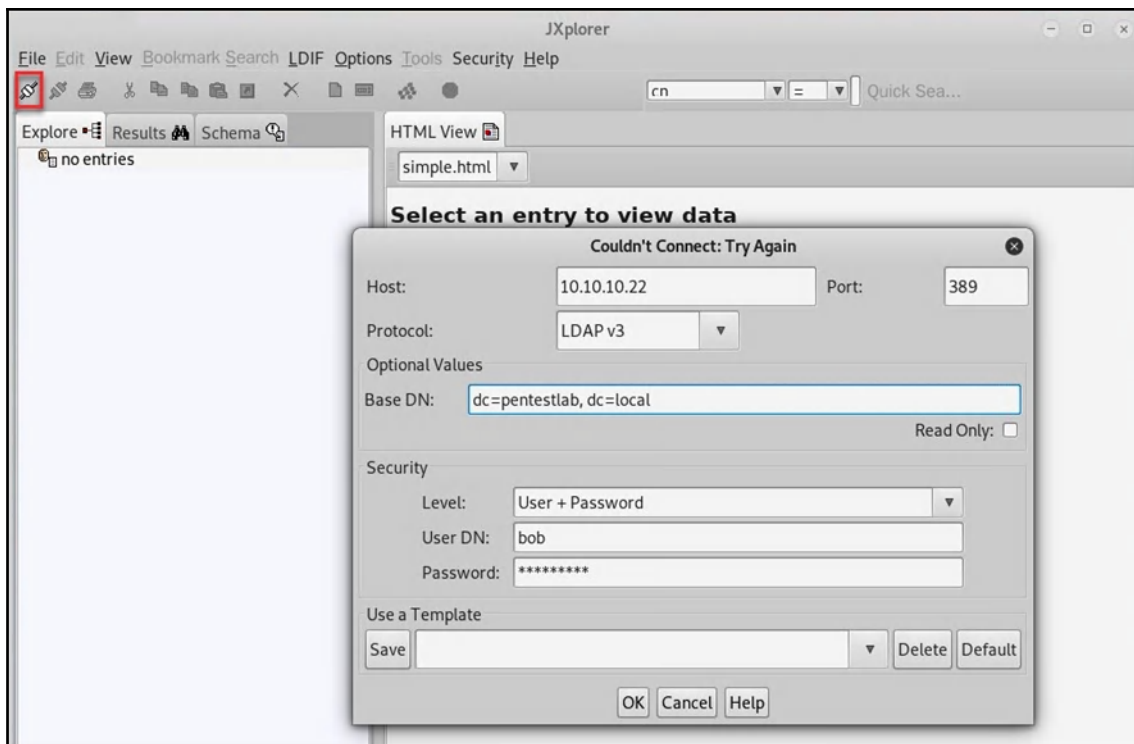
We can use a tool called JXplorer (<http://jxplorer.org>) to perform LDAP enumeration. This tool is not natively installed in Kali Linux; therefore, we'll need to download it from its GitHub repository and run it.

To get started with LDAP enumeration, let's use the following steps:

1. Use the following command to download and execute the tool:

```
git clone https://github.com/pegacat/jxplorer.git
cd jxplorer
chmod +x jxplorer.sh
./jxplorer.sh
```

2. Once you successfully execute the `./jxplorer.sh` script, the user interface will open. Click the Connect icon (located under **File**) to insert the details of your target:



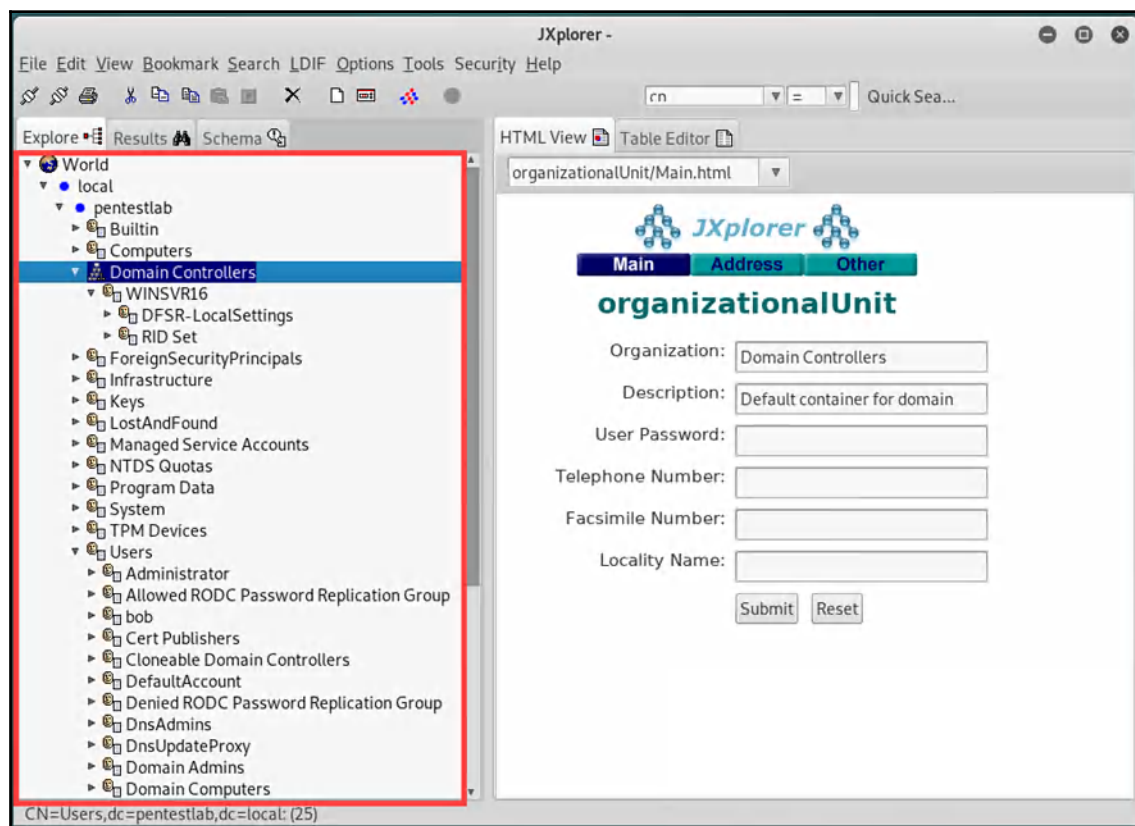
The JXplorer interface

In our lab, we have a Windows Server machine with the following configurations:

- Active Directory Domain Service installed
- Active Directory Lightweight Directory Services installed
- Domain: `pentestlab.local`
- The user account created: `bob` (belongs to the domain admin user group)

Assuming that, by using a packet sniffing tool such as Wireshark during a penetration test, you are able to capture user credentials while they are attempting to authenticate to the AD server, you can use these user accounts in the **Security** field in the preceding screenshot.

Using an administrator user account will provide the necessary privileges to extract information in JXplorer; you'll be able to enumerate sensitive information from the Active Directory server, as shown in the following screenshot:



LDAP enumeration with JXplorer

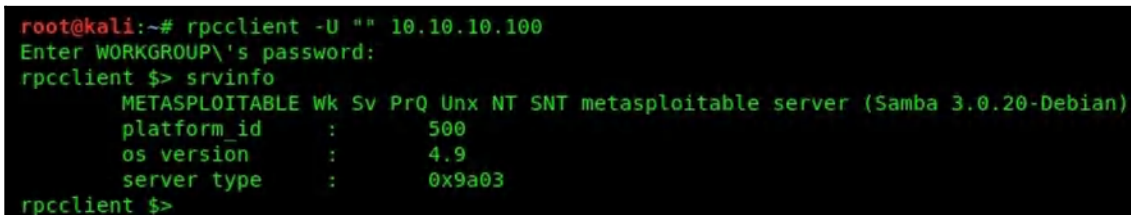
You'll be able to view the entire directory from your attacker machine and extract sensitive information. If the service only uses LDAPS, this will be a challenge as the user credentials will be concealed.

Having completed this exercise, let's use the **rpcclient** tool to perform a null session attack in the next section.

Null sessions

In a null session, an attacker is able to log in to a target using a null account. A null account is an account that does not actually exist. How is this possible? Some systems are vulnerable to allowing anonymous login. Once a user is able to log in anonymously, the null user is able to retrieve sensitive information stored on the target.

We can attempt a null session enumeration from our Kali Linux machine (attacker) on to the target, Metasploitable, by using the `rpcclient -U "" 10.10.10.100` command, as shown in the following screenshot:



```
root@kali:~# rpcclient -U "" 10.10.10.100
Enter WORKGROUP\'s password:
rpcclient $> srvinfo
      METASPLOITABLE Wk Sv PrQ Unx NT SNT metasploitable server (Samba 3.0.20-Debian)
platform_id       :          500
os version        :           4.9
server type       :        0x9a03
rpcclient $>
```

A null session attack

Using the `srvinfo` command, the target will return its operating system type to us. For a full listing of query commands, you can use the `rpcclient --help` command.

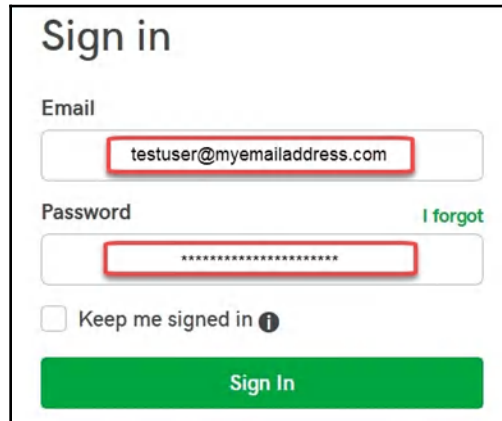
Additionally, you can visit <https://www.samba.org/samba/docs/current/man-html/rpcclient.1.html>.

Keep in mind that not all machines are vulnerable to this type of attack, but it's still worth performing during a penetration test. In the next section, we will discuss user enumeration through noisy authentication controls.

User enumeration through noisy authentication controls

Enumeration is the technique in which a hacker or a penetration tester attempts to perform a brute-force attack to either guess or confirm valid users on a target system. A simple example is where a malicious user or a penetration tester performs a password-guessing or brute-force attack on an email portal.

The following is an example of a typical login portal. The credentials shown in the following screenshot are an example and are not real:



Sign in

Email

testuser@myemailaddress.com

Password [I forgot](#)

☐ Keep me signed in ⓘ

Sign In

An attacker can attempt various combinations of possible usernames and passwords until a valid user is found. However, such attacks are considered noisy rather than stealthy (quiet). As a comparison, imagine you are playing an online first-person shooter game, and your task is to invade the enemy base and steal a trophy without alerting the guards. If you are not careful enough and make any loud noises, the guards will be alerted and the mission will fail. In this analogy, the guards are the security controls, and the sensors are the firewalls, IDS/IPS, and anti-malware protection. Hence, this technique is not quiet on a network; however, this method can still get you access to a system, provided that the security controls do not perform a lockout action before you can gain access.

A lot of times, when a user enters an incorrect username on a login portal, an error message is returned, usually stating that an incorrect username has been entered. This clearly tells an attacker that the username provided does not exist in the database. Additionally, if the incorrect password was entered, the system usually returns a message stating that an incorrect password was entered for the username. So, from an attacker's point of view, the system is telling us that the username exists in the database, but we have not provided the correct password for it.

Web developers and security professionals now include generic responses when either a username or password is incorrect, with a similar message to this: *The username/password is incorrect*. This message does not state exactly which value is correct or incorrect.

Now that you have a better understanding of noisy authentication controls, let's attempt to perform web enumeration in the following section.

Web footprints and enumeration with EyeWitness

EyeWitness is a tool that allows a penetration tester to capture screenshots of a website without leaving the Terminal—the tool does all of the work in the background. Imagine having to visually profile multiple websites, open **Virtual Network Computing (VNC)** servers, and use **Remote Desktop Protocols (RDPs)**. This can be a time-consuming task. EyeWitness takes the screenshots, stores them offline, and provides an HTML report:

1. To begin, you'll need to download EyeWitness from its GitHub repository using `git clone https://github.com/FortyNorthSecurity/EyeWitness.git`.
2. Once the download has completed, access the `root/EyeWitness/setup` directory and run the `setup.sh` script using the following sequence of commands:

```
root@kali:~# cd EyeWitness/setup/  
root@kali:~/EyeWitness/setup# chmod +x setup.sh  
root@kali:~/EyeWitness/setup# ./setup.sh
```

EyeWitness setup screen

3. Once the setup process is complete, use the `cd ..` command to go one directory up to the `root/EyeWitness` directory. To screenshot a single website, use the following command:

```
./EyeWitness.py --web --single example.com
```

You can try this tool on one of the web applications on Metasploitable or OWASP BWA virtual machines.



EyeWitness allows you to specify various protocols using operators such as: `--web`, `--rdp`, `--vnc`, and `--all-protocols`.

- 4. Once the task completes, EyeWitness will indicate whether it was successful in capturing screenshots of the target(s) and provide you with the location of the offline report, as seen in the following screenshot:

```
#####
#                               EyeWitness                               #
#####
#   FortyNorth Security - https://www.fortynorthsecurity.com           #
#####

Attempting to screenshot http://example.com

[+] Done! Report written in the /root/EyeWitness/04242019_225410 folder!
Would you like to open the report now? [Y/n] Y
```

EyeWitness reporting wizard

- 5. Upon opening the HTML report, the left-hand column contains information about the web request, while the right-hand column contains the screenshots:

Report Generated on 04/24/2019 at 22:54:10	
Web Request Info	Web Screenshot
http://example.com Page Title: Example Domain content-length: 1270 x-cache: HIT accept-ranges: bytes expires: Thu, 02 May 2019 02:54:23 GMT vary: Accept-Encoding server: ECS (phd/FD6F) last-modified: Fri, 09 Aug 2013 23:54:35 GMT connection: close etag: "1541025663" cache-control: max-age=604800 date: Thu, 25 Apr 2019 02:54:23 GMT Response Code: 200 content-type: text/html; charset=UTF-8 Source Code	<div>Example Domain</div> <p>This domain is established to be used for illustrative examples in documents. You may use this domain in examples without prior coordination or asking for permission.</p> <p>More information...</p>

Report from EyeWitness

This tool is very handy when profiling multiple services and websites at once.



Further information on EyeWitness can be found at <https://tools.kali.org/information-gathering/eyewitness>.

Now that you have completed this section, you are able to perform web enumeration using the EyeWitness tool.

Metasploit auxiliary modules

Metasploit is an exploitation development framework created by Rapid7 (www.rapid7.com). Metasploit contains many features and functions for penetration testing. There are many modules, such as exploits, payloads, encoders, and auxiliary. The auxiliary module contains port scanners, network sniffers, fuzzers, and a lot more to facilitate the information-gathering phase of a penetration test:

1. To access the Metasploit interface, open a new Terminal and execute the following commands:

```
service postgresql start  
msfconsole
```

2. Once the user interface loads, the `show auxiliary` command will provide a list of all of the auxiliary modules within Metasploit. Let's use a simple example to demonstrate how to use a module: let's imagine you would like to perform a stealth (SYN) scan on a target. You can begin by selecting a module.
3. Use the `use auxiliary/scanner/portscan/syn` command.
4. Check the description and requirements by using the `show options` command.
5. This module requires that a remote host is configured; use the `set RHOSTS target` command.
6. To execute the module, use the `run` command.

7. The following screenshot is a demonstration of a stealth scan on our Windows Server machine (10.10.10.14) and the results provided at the bottom indicate that various ports were found open:

```
msf5 > use auxiliary/scanner/portscan/syn
msf5 auxiliary(scanner/portscan/syn) > show options 1

Module options (auxiliary/scanner/portscan/syn):

  Name      Current Setting  Required  Description
  ----
  BATCHSIZE 256             yes       The number of hosts to scan per set
  DELAY     0               yes       The delay between connections, per thread, in mil
  INTERFACE           no       The name of the interface
  JITTER    0               yes       The delay jitter factor (maximum value by which
  PORTS     1-10000         yes       Ports to scan (e.g. 22-25,80,110-900)
  RHOSTS           yes       The target address range or CIDR identifier
  SNAPLEN   65535           yes       The number of bytes to capture
  THREADS    1               yes       The number of concurrent threads
  TIMEOUT   500             yes       The reply read timeout in milliseconds

msf5 auxiliary(scanner/portscan/syn) > set RHOSTS 10.10.10.14 2
RHOSTS => 10.10.10.14
msf5 auxiliary(scanner/portscan/syn) > run 3

[+] TCP OPEN 10.10.10.14:80
[+] TCP OPEN 10.10.10.14:135
[+] TCP OPEN 10.10.10.14:139
```

Open ports on the target

8. Additionally, to search for a module within Metasploit, using the `search` keyword syntax can be very useful as there are a lot of different modules in the framework, and learning them all can be very challenging and overwhelming.

In the later chapters in this book, we will dive deeper into using Metasploit to perform exploitation on target devices in our lab.

Summary

During this chapter, we covered various DNS interrogation techniques using a variety of tools to discover important servers, subdomains, and IP addresses, and were able to successfully extract the zone files from a DNS server (zone transfer) due to a misconfiguration on the target's DNS server.

Then, we used Nmap to perform various types of port scanning to determine the port status, running services and their versions, and the target's operating system; we also gained an indication of whether there's a firewall on the target. Finally, to close this chapter, we performed SMB and LDAP enumeration to gather user shares and directory records on our network devices.

Now that you have completed this chapter, you'll be able to successfully perform DNS zone transfers on vulnerable DNS servers; profile a system to discover its operating system, running services, and security vulnerabilities; evade detection while performing network scans; and perform LDAP and system enumeration on a target. You also obtained the skills to visually profile multiple websites at once. I hope this chapter has been helpful to your journey in learning penetration testing.

In Chapter 7, *Working with Vulnerability Scanners*, we will cover the importance of using vulnerability scanners to find security weaknesses and flaws on a target.

Questions

The following are some questions based on the topics we have covered in this chapter:

1. What is the primary purpose of using DNS?
2. What is meant by a DNS zone transfer?
3. What tool allows us to perform a scan on a target system and determine its running services and operating system?
4. What method is used to evade a firewall during a scan?
5. What tool can be used to enumerate Active Directory?

Further reading

- **Information Gathering and Vulnerability Assessment:** <https://hub.packtpub.com/information-gathering-and-vulnerability-assessment-0/>
- **Open Source Intelligence:** <https://hub.packtpub.com/open-source-intelligence/>
- **Gather Intel and Plan Attack Strategies:** <https://hub.packtpub.com/gather-intel-and-plan-attack-strategies/>

3

Section 3: Vulnerability Assessment and Penetration Testing with Kali Linux 2019

This section exposes the reader to various vulnerability scanners and their purpose and functionality, as well as penetration testing, and assists in identifying security weaknesses in a system or network and how to exploit them.

Furthermore, readers will acquire experience of hands-on penetration testing techniques and methodologies by using various tools in Kali Linux 2019. The reader will be taken through all the pertinent stages, from discovering vulnerabilities on a target to exploiting various operating systems and web applications.

This section comprises the following chapters:

- Chapter 7, *Working with Vulnerability Scanners*
- Chapter 8, *Understanding Network Penetration Testing*
- Chapter 9, *Network Penetration Testing - Pre-Connection Attacks*
- Chapter 10, *Network Penetration Testing - Gaining Access*
- Chapter 11, *Network Penetration Testing - Post-Connection Attacks*
- Chapter 12, *Network Penetration Testing - Detection and Security*
- Chapter 13, *Client-Side Attacks - Social Engineering*
- Chapter 14, *Performing Website Penetration Testing*
- Chapter 15, *Website Penetration Testing - Gaining Access*
- Chapter 16, *Best Practices*