v4

Penetration Testing Student

Footprinting and Scanning

Section 03 | Module 02

© Caendra Inc. 2019
All Rights Reserved

Table of Contents

MODULE 02 | Footprinting and Scanning

Learning Objectives

By the end of this module, you should have a better understanding of:

✓ Discovering and mapping network environments

✓ Tools, methods, and techniques to perform proper reconnaissance on the network level

Disclaimer

2.1 Disclaimer

In this module, you will see examples of tools and techniques used on realistic IP addresses and hosts.

Never run any of these tools and techniques on those addresses, or any machine and network without proper authorization!

2.2

Mapping a Network

2.2 Mapping a Network

How does this support my pentesting career?

- Ability to perform efficient penetration tests
- Knowledge of your in-scope targets
- Ability to create a technological map

2.2 Mapping a Network

After collecting information about the target organization during the information gathering stage, a penetration tester proceeds to the **fingerprinting** and **enumeration** of the nodes running on the client's network; this is the infrastructure part of information gathering.

The techniques you are going to see work both on **local** and **remote** networks.

2.2 Mapping a Network

As you know, every host connected to the Internet or a private network must have a unique IP address which identifies it.

How can a penetration tester determine what hosts, sitting in an in-scope network, are up and running?

EXAMPLE

2.2.1 Why Map a (Remote) Network?

A company asks for a penetration test, and the following address block is considered in scope:

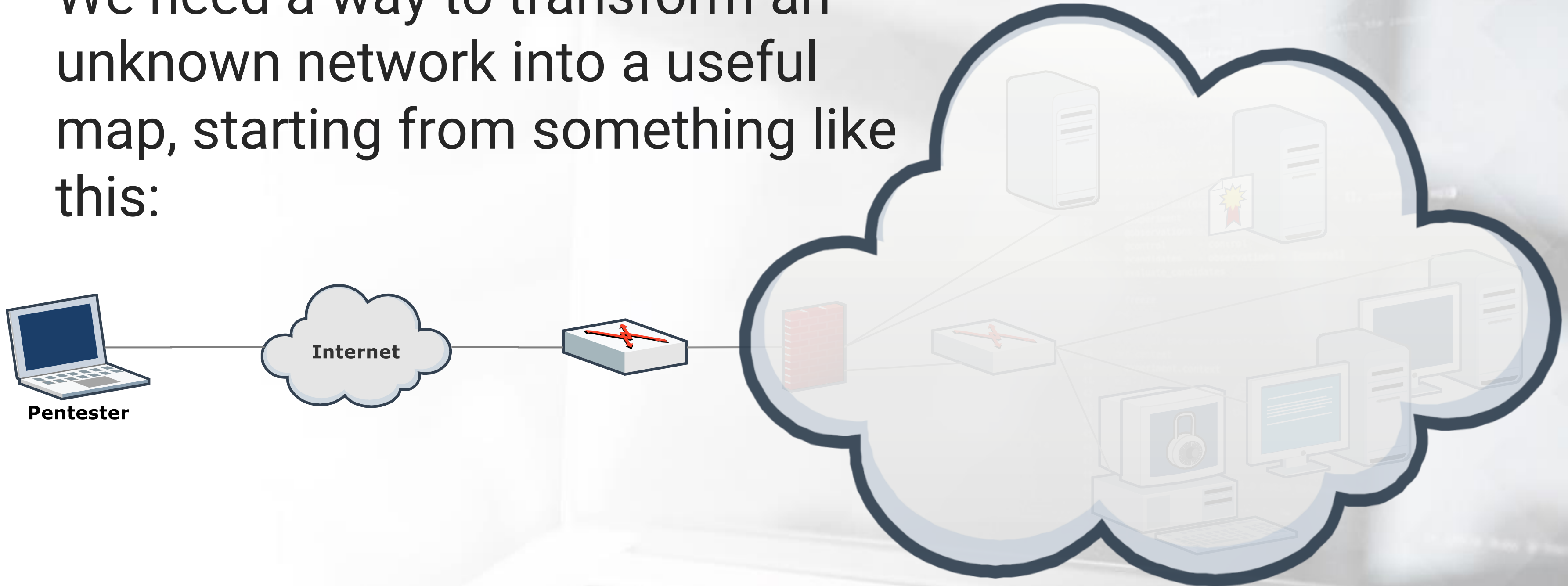
`200.200.0.0/16`.

A sixteen-bit long netmask means the network could contain up to 2^{16} (65536) hosts with IP addresses in the `200.200.0.0` - `200.200.255.255` range.

The penetration tester needs a way to find which of the 65536 IP addresses are **assigned to a node**.

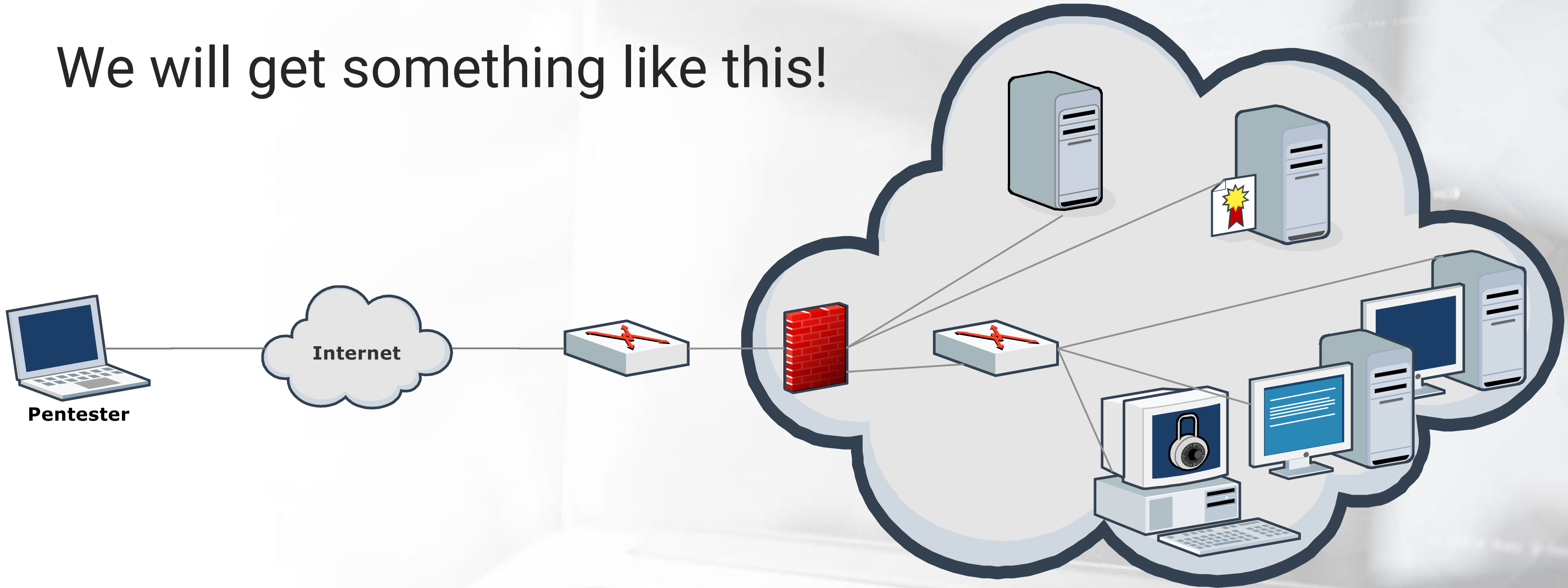
2.2.1 Why Map a (Remote) Network?

We need a way to transform an unknown network into a useful map, starting from something like this:



2.2.1 Why Map a (Remote) Network?

We will get something like this!



2.2.2 Ping Sweeping

The previous example shows how mapping the remote network completely changes your understanding of it.

Trying to **blindly exploit 65536 hosts** would be a massive waste of time and resources!

Q

How do you determine which IP addresses in scope are assigned to a host?

A

*The simplest way is through **ping sweeping!***

2.2.2 Ping Sweeping

You probably already know the **ping** command; it is a utility designed to test if a machine is alive on the network. You can run a ping in every major operating system by using the command line:

```
> ping www.site.test

Pinging www.site.test [12.34.56.78] with 32 bytes of data:
Reply from 12.34.56.78: bytes=32 time=57ms TTL=127
Reply from 12.34.56.78: bytes=32 time=43ms TTL=127
Reply from 12.34.56.78: bytes=32 time=44ms TTL=127
```

2.2.2 Ping Sweeping

Ping works by sending one or more special ICMP packets (Type 8 – **echo request**) to a host. If the destination host replies with ICMP **echo reply** packets, then the host is alive.

ICMP (RFC792) is a protocol used to carry diagnostic messages. ICMP uses the services provided by IP, but it is actually a part of the Internet Protocol.

http://tools.ietf.org/html/rfc792

2.2.2 Ping Sweeping

Ping sweeping tools **automatically** perform the same operation to **every host** in a subnet or IP range, saving you from typing hundreds or thousands ping of commands.

In the following slides, you will see two utilities to map a network.

2.2.2.1 fping

Fping is a Linux tool which is an improved version of the standard *ping* utility. You can use *Fping* to perform ping sweeps.

It is installed by default on Kali Linux, and you can run it from the command line using the following syntax:

```
# fping -a -g IPRANGE
```

2.2.2.1 fping

The `-a` option forces the tool to show only alive hosts, while the `-g` option tells the tool that we want to perform a ping sweep instead of a standard ping.

You can define an IP range by using the **CIDR notation** or by specifying the **start and the ending addresses** of the sweep.

Examples:

```
# fping -a -g 10.54.12.0/24
# fping -a -g 10.54.12.0 10.54.12.255
```

2.2.2.1 fping

When running *Fping* on a **LAN** you are directly attached to, even if you use the `-a` option, you will get some warning messages (`ICMP Host Unreachable`) about offline hosts.

2.2.2.1 fping

To suppress those messages, you can redirect the process standard error to `/dev/null`.

Example:

```
# fping -a -g 192.168.82.0 192.168.82.255 2>/dev/null
192.168.82.1
192.168.82.11
192.168.82.112
192.168.82.171
192.168.82.202
```

2.2.2.1 fping

Pinging a host is just one way to see if it is alive. In the next slides, you will see a very powerful tool able to perform advanced host discovery and much more: ***nmap***!

http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-3.html
http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO.html
http://tldp.org/

2.2.2.2 Nmap Ping Scan

Nmap (Network Mapper) is an open source tool for network exploration and security auditing. It is installed on Kali by default, but you can also install it on nearly every operating system.

We will see some of *Nmap's* features during this course, one of them being **ping scanning**.

http://nmap.org/download.html

2.2.2.2 Nmap Ping Scan

You can perform a ping scan by using the `-sn` command line switch. You can specify your targets on the command line in CIDR format as a range and by using wildcard notation.

Examples:

```
# nmap -sn 200.200.0.0/16
# nmap -sn 200.200.123.1-12
# nmap -sn 172.16.12.*
# nmap -sn 200.200.12-13.*
```

2.2.2.2 Nmap Ping Scan

Moreover, you can save your host list in a file and use the *input list* `-iL` command line switch.

Example:

You can achieve the same results of the previous example by creating a file containing:

```
200.200.0.0/16
200.200.123.1-12
172.16.12.*
200.200.12-13.*
```

2.2.2.2 Nmap Ping Scan

You can save your host list and invoke it with Nmap by using the input command line switch in conjunction with -sn. This way you will conduct a ping scan against each host that is in the given list.

Please note, that in this example we are naming our saved file hostslist.txt.

```
# nmap -sn -iL hostslist.txt
```

2.2.2.2 Nmap Ping Scan

Nmap has many host discovery features; we strongly suggest you test the tool as much as you can at home or in **Hera Lab**.

To check what host discovery methods you can use, you can refer to the *nmap* manual page (on Linux you can use the `man nmap` command) or the brief output of the `nmap` command without options.

2.2.2.2 Nmap Ping Scan

In fact, the scanner does not just use ping packets to find live hosts. Here is an extract of the output of the `nmap` command.

```
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given
ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
```

2.2.3 OS Fingerprinting

Once you have mapped the network, you end up with a **list of live hosts**. These are the hosts that **responded to pings** or to *nmap* probes, but you still don't know what software the hosts are running and what role they have in the network.

Are they routers, servers or clients? At this stage, you can only say that they are up and running hosts connected to the network.

2.2.3 OS Fingerprinting

To transform a list of live IP addresses into something more useful, we need to know what software (i.e., what operating system) responded to our probes.

OS fingerprinting is the process of determining the operating system used by a host on a network.

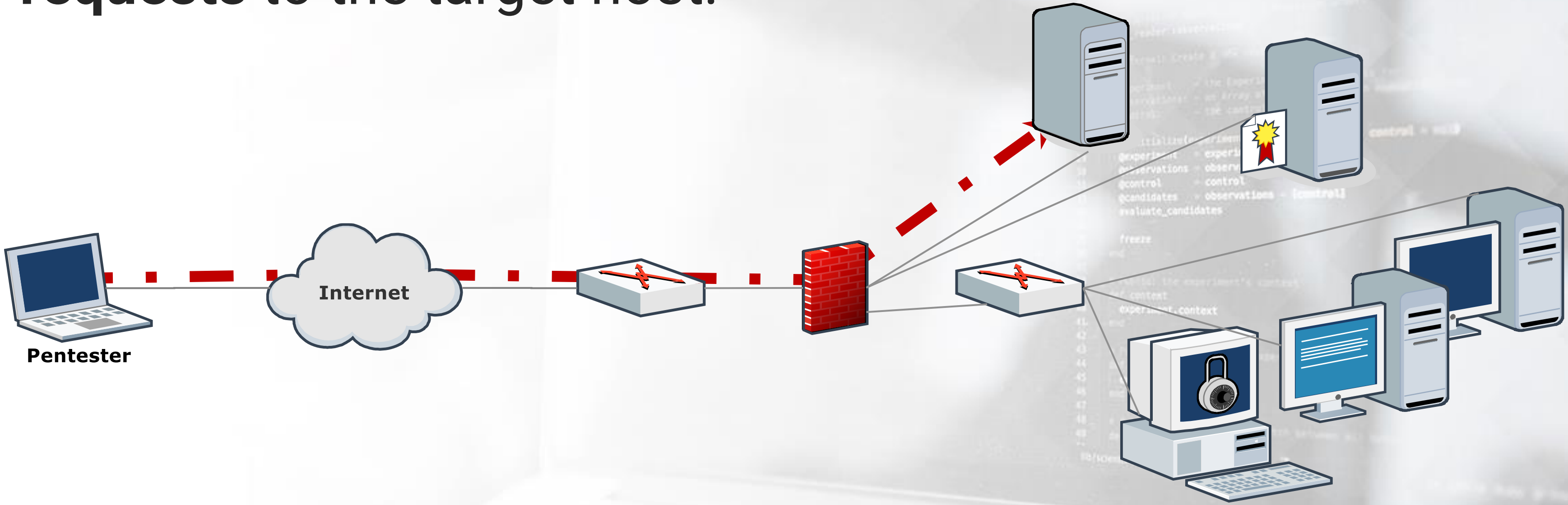
2.2.3 OS Fingerprinting

To fingerprint an operating system you have to **send network requests** to the host and then **analyze the responses** you get back.

This is possible because of some tiny differences in the **network stack implementation** of the various operating systems.

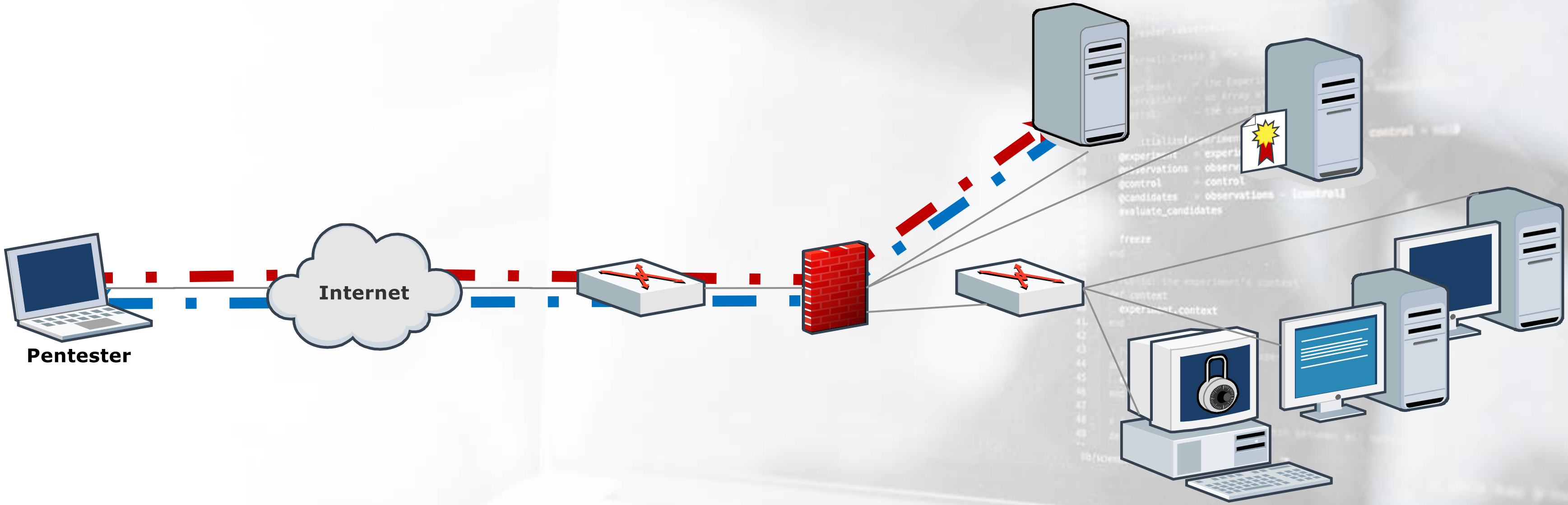
2.2.3 OS Fingerprinting

Fingerprinting tools send a **series** of specially crafted **requests** to the target host.



2.2.3 OS Fingerprinting

They then examine **every bit** in the responses, creating a **signature** of the host behavior.



2.2.3 OS Fingerprinting

Finally, the signature is compared against a **database** of known operating systems signatures.

This process exploits differences in the network stack implementation, so there is some **guesswork**; but, using the best tools and supporting them with experience, you can achieve pretty reliable results!

2.2.3 OS Fingerprinting

During a penetration test, you will have to perform this reconnaissance step on **every network node**, including:

- Routers
- Firewalls
- Hosts
- Servers
- Printers
- And so on...

2.2.3 OS Fingerprinting

The goal of this phase is to write a table like the following:

IP Address	OS	Confidence
200.200.3.1	PAN-OS	85%
200.200.3.10	Linux 3.7	100%
200.200.3.78	Linux 2.6.19 – 2.6.36	90%
200.200.4.12	Windows 7 SP1	100%
200.200.4.16	Windows 7 SP1	75%
200.200.4.18	FreeBSD	85%
200.200.4.19	HP-OS	78%

2.2.3 OS Fingerprinting

You can perform OS fingerprinting on a traffic capture you recorded (passive) or by using the technique we have just seen (active).

Offline OS fingerprinting can be done with *p0f*. In this course, we will concentrate on active fingerprinting with *nmap*.

2.2.3.1 OS Fingerprinting with Nmap

To perform OS fingerprinting with *nmap*, you have to use the `-O` command line option and specify your target(s).

You can also add the `-Pn` switch to skip the ping scan if you already know that the targets are alive.

```
# nmap -Pn -O <target(s)>
```

2.2.3.1 OS Fingerprinting with Nmap

You can fine-tune the OS fingerprinting process by using the following options:

```
OS DETECTION:
        -O: Enable OS detection
        --osscan-limit: Limit OS detection to promising targets
        --osscan-guess: Guess OS more aggressively
```

Choosing a lighter or more aggressive OS detection depends on the engagement.

EXAMPLE

2.2.3.1 OS Fingerprinting with Nmap

If you really need to detect the OS of a machine you know is alive, but is not responding to ping probes, you could run:

```
# nmap -Pn -O <target>
```

On the other hand, if you have to scan thousands of hosts you could at first limit OS reconnaissance to just the promising ones.

```
# nmap -O --osscan-limit <targets>
```

2.2.4 Video – OS Fingerprinting with Nmap

In the following video, you will see how to perform remote OS reconnaissance with *nmap*.

If you want to know even more about remote OS detection with *nmap*, check out this resource here.

http://nmap.org/book/osdetect.html

2.2.4 Video – OS Fingerprinting with Nmap

OS Fingerprinting with Nmap

You will learn basics of discovering information about your targets using nmap.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Port Scanning

2.3 Port Scanning

How does this support my pentesting career?

Ability to:

- Prepare for the vulnerability assessment phase
- Perform stealth reconnaissance
- Detect firewalls

2.3 Port Scanning

Once network mapping is performed, we know which nodes, and associated operating systems, are active on the target network.

We now need a way to discover the daemons and services running on those nodes: **port scanning!**

2.3 Port Scanning

Port scanning is a process used to determine what TCP and UDP ports are **open** on target hosts. Moreover, it lets you know which daemon, in terms of **software and version**, is **listening** on a specific port.

Using this technique, you can create a list of potential weaknesses and vulnerabilities to check during the following vulnerability assessment phase.

2.3 Port Scanning

As you can recall from the *Networking* module, a **daemon** is a piece of software running on a server to provide a given **service**. A daemon also listens on a specific **port**.

The ultimate **goal** of port scanning/service detection is to find the software name and version of the daemons running on each host.

2.3.1 Under the Hood of a Port Scanner

Port scanning, like OS fingerprinting, works by sending probes to the targets and analyzing the responses.

Port scanning utilities, or **port scanners**, automate probes request and response analysis. These are powerful tools that not only give you information about the targets but also let you detect if there is a firewall between you and the target.

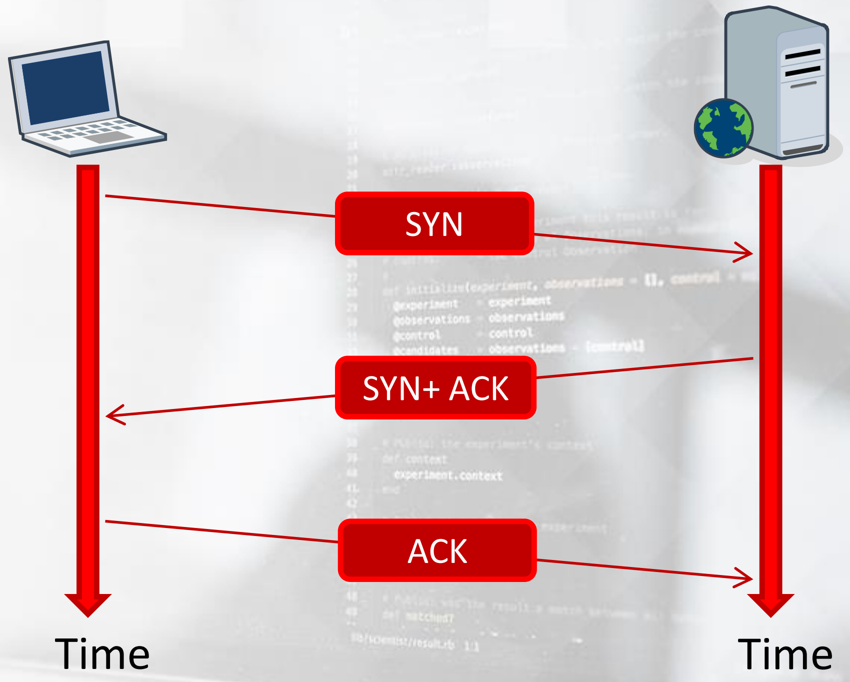
2.3.1 Under the Hood of a Port Scanner

To use those tools at their best, you have to know how they work!

In the following slides, you will see how a **TCP port scanner** works.

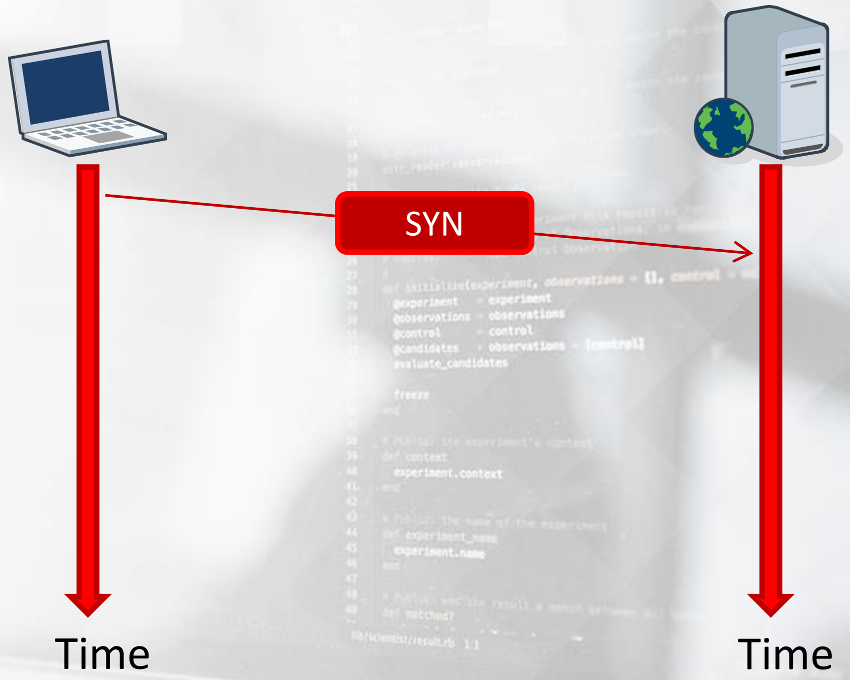
2.3.1.1 TCP Three Way Handshake

In the *Networking* module, you saw how the TCP 3-way handshake works.



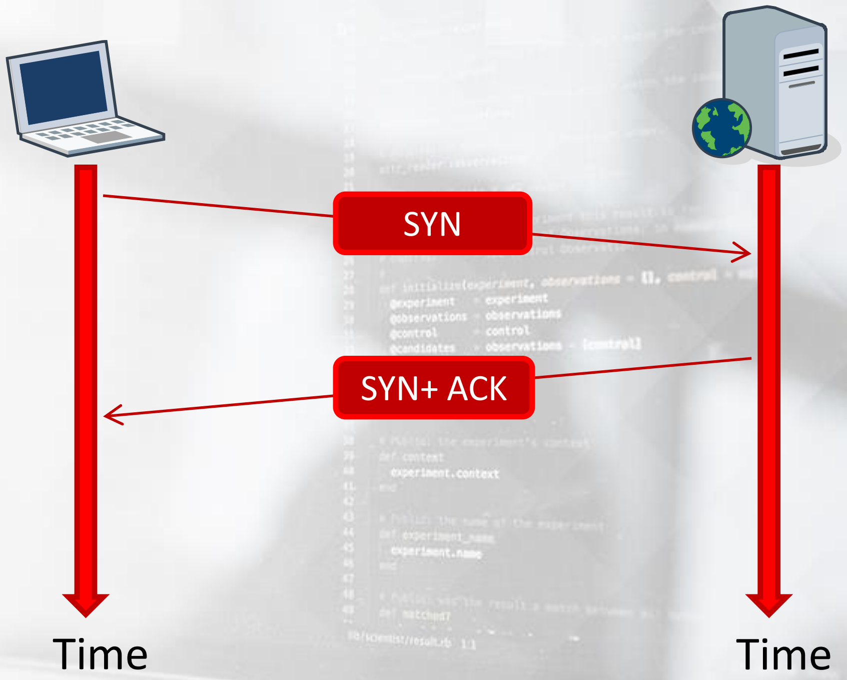
2.3.1.1 TCP Three Way Handshake

When a client wants to connect to a server, it first sends a packet with the SYN flag enabled.



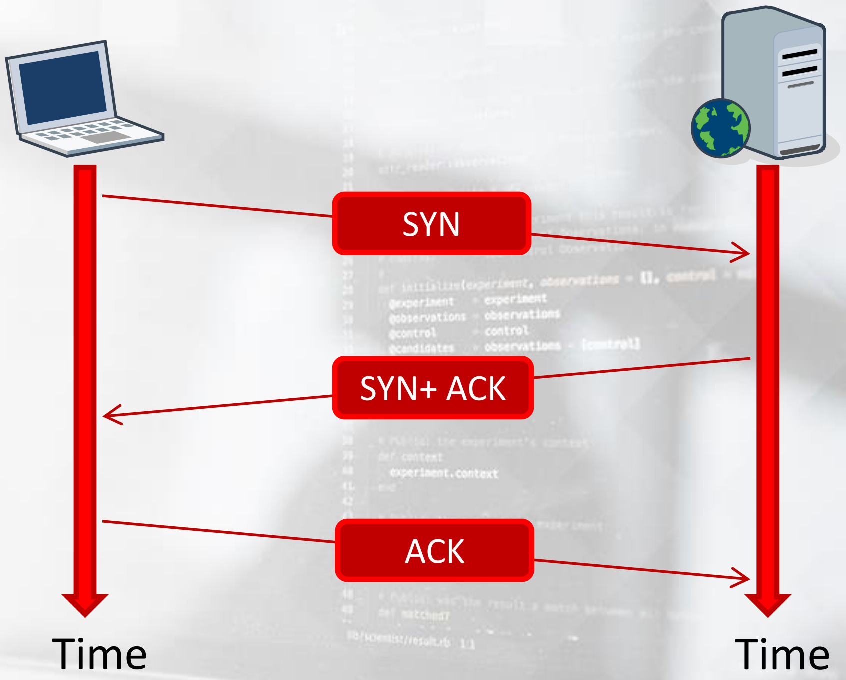
2.3.1.1 TCP Three Way Handshake

The server then responds by sending a packet with both `SYN` and `ACK` flags enabled.



2.3.1.1 TCP Three Way Handshake

Finally, the client replies back by sending a packet with the `ACK` flag enabled and the actual data transmission can start.



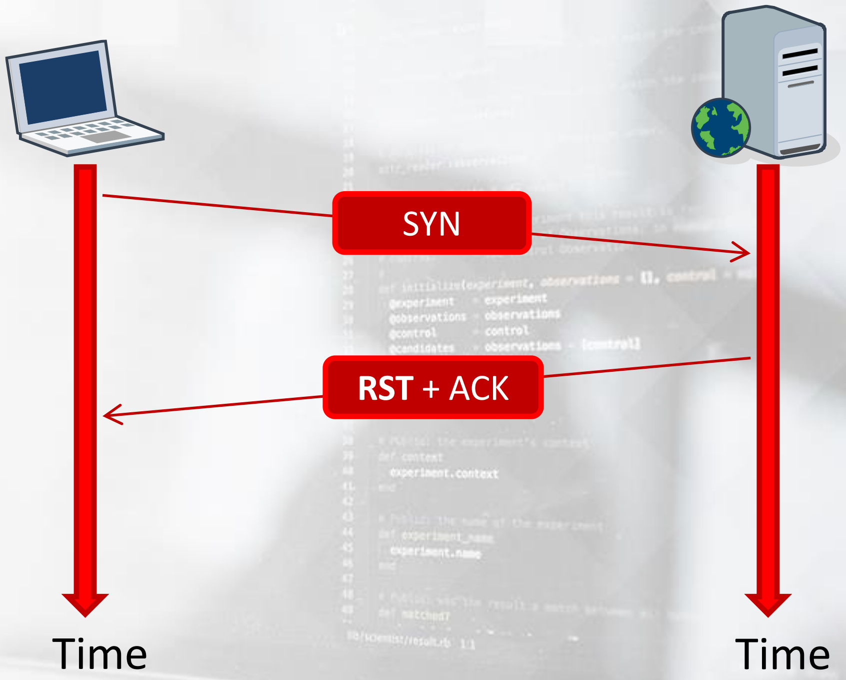
2.3.1.1 TCP Three Way Handshake

But, what happens if a client tries to connect to a **closed** port?

In other words, what happens if a client tries to contact a daemon that is **not running or is running on another port**?

2.3.1.1 TCP Three Way Handshake

The server will reply with a packet that has the *Reset* (`RST`) and `ACK` flags set. This behavior tells the client that the port is **closed**.



2.3.1.1 TCP Three Way Handshake

The techniques you will find in the following slides leverage these facts to perform a port scan.

You will also see why a particular scan method can be stealthier than another.

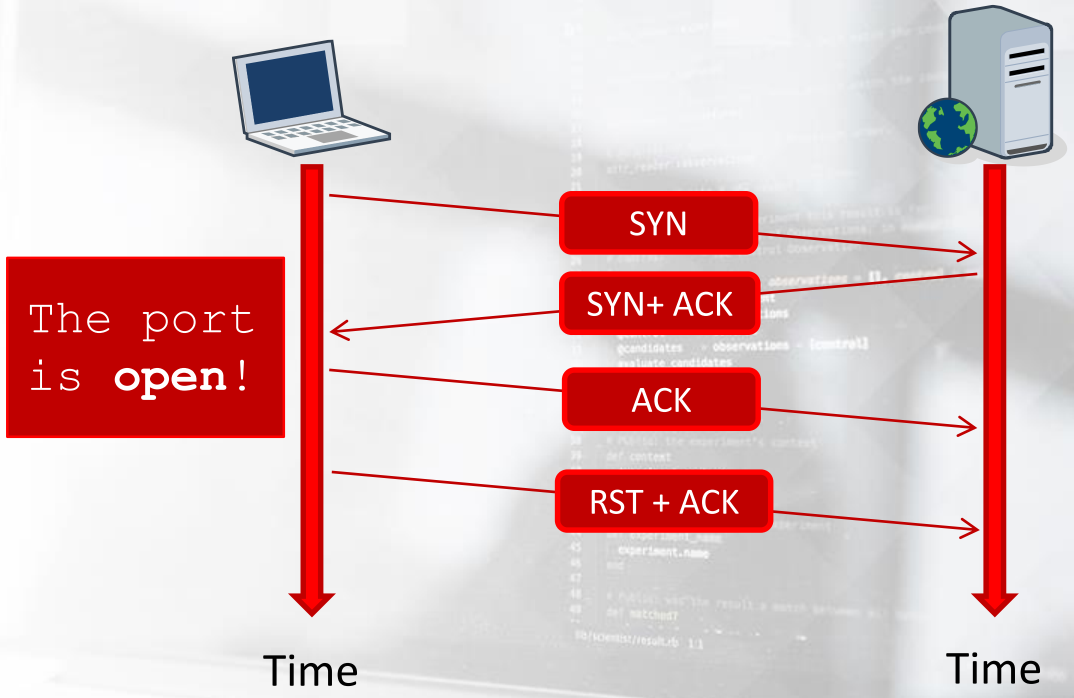
2.3.1.2 TCP Connect Scan

The simplest way to perform a port scan is trying to connect to every port.

- If the scanner receives a `RST` packet, then the port is **closed.**
- If the scanner can **complete the 3-way handshake**, then the port is **open**. After connecting, the scanner sends an `RST` packet to the target host to abruptly close the connection.

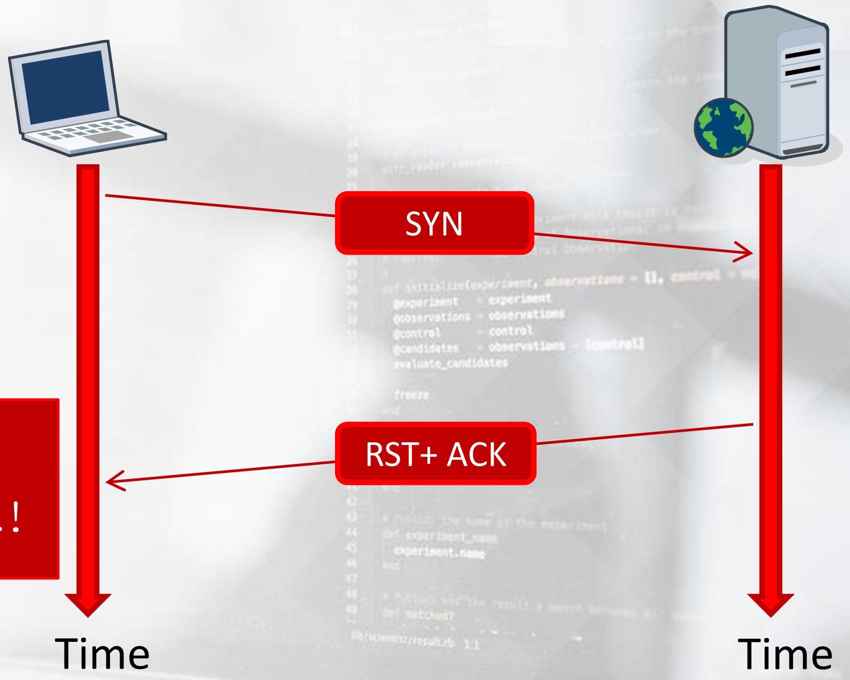
2.3.1.2 TCP Connect Scan

The following diagram summarizes a TCP Connect scan run for a single **open** port.



2.3.1.2 TCP Connect Scan

This diagram summarizes a TCP Connect scan run for a single **closed** port.



```
The port
is closed!
```

2.3.1.2 TCP Connect Scan

Every TCP connect scan probe gets recorded in the daemon logs because, from the application point of view, the probe looks like a legitimate connection.

System administrators can easily **detect** the scan as they will see a lot of connections to all the services running on a single machine; to prevent that, TCP SYN scans were invented.

2.3.1.3 TCP SYN Scan

TCP SYN scans were invented to be **stealthy** by design.

During a SYN scan, the scanner does not perform a full handshake it just **sends a SYN packet** and analyzes the response coming from the target machine.

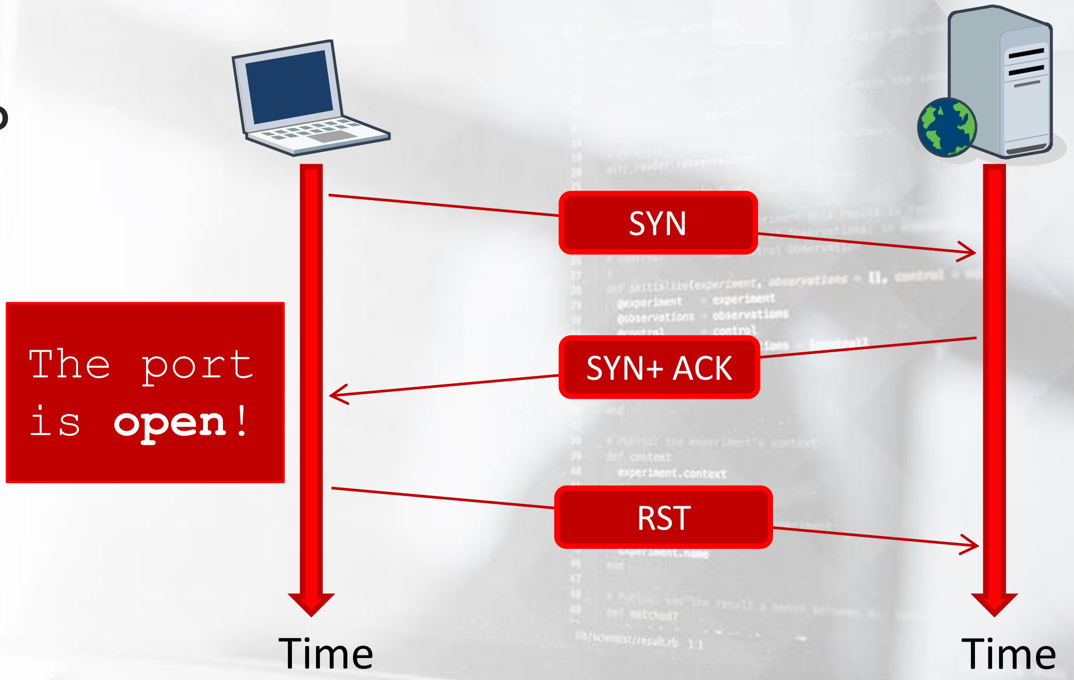
2.3.1.3 TCP SYN Scan

The scanner sends a TCP packet with the `SYN` flag enabled to the destination `<host>:<port>` pair and:

- If it receives a **`RST`** packet, then it marks the port as **closed.**
- If the scanner receives an **`ACK`** packet, then the port is **open**. After marking the port as open, the scanner sends an `RST` packet to the target host to stop the handshake.

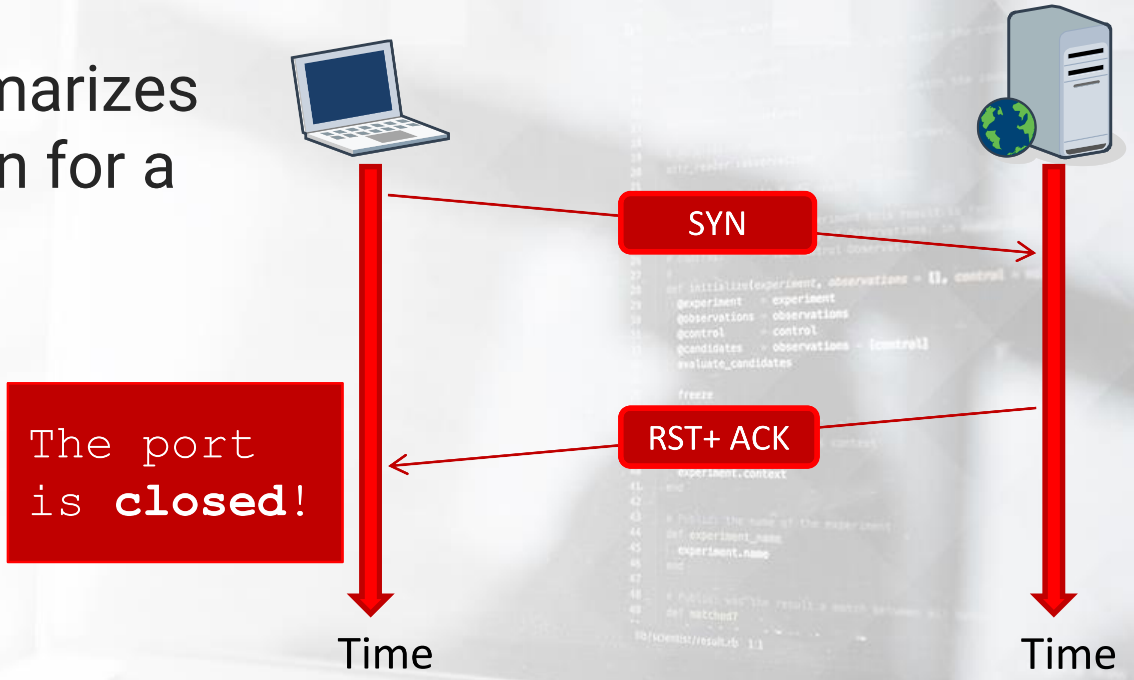
2.3.1.3 TCP SYN Scan

Here we see a diagram that summarizes a TCP SYN scan run for a single **open** port.



2.3.1.3 TCP SYN Scan

This diagram summarizes a TCP SYN scan run for a single **closed** port.



2.3.1.3 TCP SYN Scan

As there is no real connection to the destination daemon, a SYN scan **cannot be detected by looking at daemons logs**.

As the course progresses, you will see how to use *Nmap* to perform all these scan types!

2.3.2 Scanning with Nmap

You have already seen how to use *Nmap* to perform host discovery and OS fingerprinting.

Let's now take a look at how to use *Nmap* in general and how to perform scans with different methods.

2.3.2 Scanning with Nmap

Nmap syntax is very simple, yet powerful:

```
# nmap [Scan Type(s)] [Options] {target specification}
```

You have to specify one or more scan types, some options, and your target(s).

EXAMPLE

2.3.2 Scanning with Nmap

To pass a scan type or an option on the command line, you have to use a dash "**-**" followed by one or more letters to specify your options. In the following slides, you will see some common command line switches you can use to perform different scan activities.

```
# nmap –sS –sV –O --osscan-limit 192.168.1.0/24
```

2.3.2.1 Nmap Scan Types

The most used scan types are:

- `-sT` performs a **TCP connect scan**
- `-sS` performs a **SYN scan**
- `-sV` performs a **version detection scan**

While **TCP connect scans** and **SYN scans** works as you saw earlier, the version detection scan does **something more**.

Let's see how to use *Nmap* to perform these scans!

2.3.2.2 TCP Connect Scan with Nmap

To perform a **TCP connect scan**, you can use the command line switch **`-sT`**.

Keep in mind that this type of scan gets **recorded in the application logs** on the target systems, as every daemon will receive a connection from the scanning machine.

```
# nmap -sT <target>
```

2.3.2.3 TCP SYN Scan with Nmap

To perform a **TCP SYN scan**, you can use the command line switch `-sS`. This type of scan is also known as **stealth scan** because there is no (full) connection to the target daemons.

Note that a well-configured IDS will still detect the scan!

```
# nmap -sS <target>
```

2.3.2.4 Version Detection Scan with Nmap

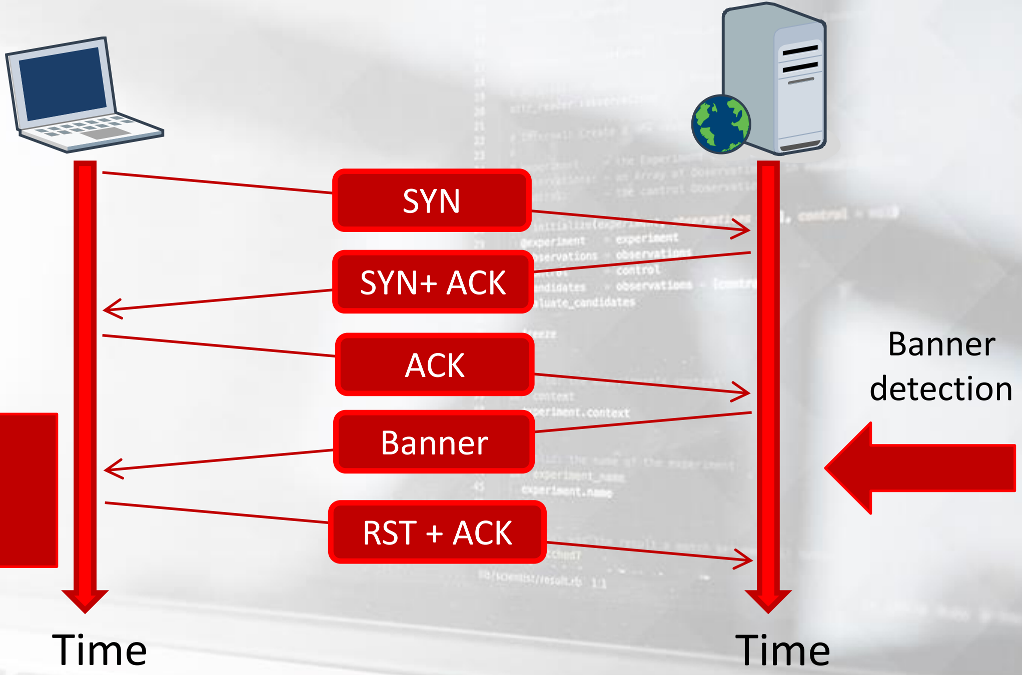
To perform a **version detection scan**, you can use the command line switch **`-sV`**.

This type of scan mixes a TCP connect scan with some probes, which are used to detect what application is listening on a particular port; this is not stealthy but very useful. Let's see why.

```
# nmap -sV <target>
```

2.3.2.4 Version Detection Scan with Nmap

During a version detection scan, *Nmap* performs a TCP connect scan and reads from the server the **banner** of the daemon listening on a port.



```
The application
listening is ...
```

2.3.2.4 Version Detection Scan with Nmap

If the daemon does not send a banner by itself, *Nmap* sends some probes to understand what the listening application is. The idea behind this is to guess the application and its version by studying its behavior.

If it looks like a dog, walks like a dog and barks like a dog, then it's a dog.

2.3.2.4 Version Detection Scan with Nmap

The knowledge you gather about the network, the Operating Systems and the daemons running on it, are of paramount importance to set up and carry out a successful vulnerability assessment; this will prevent you from performing vulnerability scans and manual investigations blindly.

2.3.3 Specifying the Targets

We’ve just seen how to use different scan methods, but choosing the right scan types without effectively setting your targets would not be very useful, right?

Nmap has a very flexible syntax to specify your targets. You can use **DNS names, IP address lists, CIDR notation, wildcards, ranges, octets lists**, and even **input files.**

2.3.3.1 By DNS Name

Specifying targets by their **DNS names** is just a matter of writing them on the command line.

Example:

```
# nmap <scan type> target1.domain.com target2.otherdomain.com
```

2.3.3.2 With an IP Addresses List

Similarly, you can write a **list of IP addresses** on the command line.

Example:

```
# nmap <scan type> 192.168.1.45 200.200.14.56 10.10.1.1 10.10.1.3
```

2.3.3.3 By Using CIDR Notation

You could also use the CIDR notation if you have to scan one or more networks.

Example:

```
# nmap <scan type> 192.168.1.0/24 200.200.1.0/16 10.0.0.0/8
```

2.3.3.4 By Using Wildcards

There are also some very useful features to combine target IP addresses in a very flexible way.

One of them is using **wildcards** where an asterisk * is converted to the 0-255 range.

Example:

```
# nmap <scan type> 192.168.1.*
```

2.3.3.4 By Using Wildcards

For instance, this is very useful if you have to map a network and you know that the last octet of a router address is a one, but you are working on a /16 network.

Example:

```
# nmap <scan type> 10.10.*.1
```

2.3.3.4 By Using Wildcards

If you want to scan 200.200.0.0/16, you could use the following syntax instead of the CIDR one.

Example:

```
# nmap <scan type> 200.200.*.*
```

2.3.3.5 Specifying Ranges

Continuing on, you can specify an interval for every octet. For example, if you want to scan just a part of a /16 network you could use:

Example:

```
# nmap <scan type> 200.200.6-12.*
```

2.3.3.5 Specifying Ranges

Or, you can leave out a host from your scan of a /24 network.

Example:

```
# nmap <scan type> 10.14.33.0-112 10.14.33.114-255
```

2.3.3.6 Octets Lists

If you need to scan a limited number of targets that are somehow related IP-wise, you can specify octet lists by using a comma.

2.3.3.6 Octets Lists

For instance, you can scan only the hosts with an IP address ending in 1, 3 and 17 by using:

Example:

```
# nmap <scan type> 10.14.33.1,3,17
```

2.3.3.6 Octets Lists

You can use this feature on every octet, even multiple times. In the following example, *Nmap* will scan eight hosts:

Example:

```
# nmap <scan type> 10.14,20.3.1,3,17,233
```

2.3.3.7 Combining the Previous Methods

You can combine the methods we have just seen to specify scan targets. For instance, you can mix a CIDR notation with an octet lists notation.

Example:

```
# nmap <scan type> 10.100.45.0/24 200.200.4.1,7,5,99
```

2.3.3.7 Combining the Previous Methods

But, if you have a very long list of targets, writing it on the command line would be impractical.

As we saw earlier, you can instead write them in a file, one per line, and then use the **`-iL`** command line switch.

Example:

```
# nmap <scan type> -iL HostToScan.txt
```

2.3.4 Choosing the Ports to Scan

When you specify one or more targets, by default, *Nmap* scans the **most common ports** used on the Internet.

If you want to specify custom ports, you can use the `-p` option.

2.3.4 Choosing the Ports to Scan

You can specify your ports as a comma-**separated list**, or as a **port interval**.

Examples:

```
# nmap -p 21,22,139,445,443,80 <target>
# nmap -p 100-1000 <target>
```

2.3.5 Nmap Examples

Let’s check out a *Nmap* invocations sample!

EXAMPLE

2.3.5 Nmap Examples

```
# nmap 10.11.12.0/24
```

- This performs a SYN scan (*nmap* default)
- The target is a /24 network

```
# nmap -sT 192.168.12.33,45
```

- This performs a TCP connect scan
- The targets are `192.168.12.33` and `192.168.12.45`

EXAMPLE

2.3.5 Nmap Examples

```
# nmap -sV 10.11.12.0/24 10.200.0.1
```

- This performs a service detection scan
- The target is a /24 network plus a single host

```
# nmap -sS 1.2.3.4 4.5.6-9.7
```

- This performs a SYN scan
- The targets are:
  - `1.2.3.4`
  - `4.5.6.7`
  - `4.5.7.7`
  - `4.5.8.7`
  - `4.5.9.7.`

EXAMPLE

2.3.5 Nmap Examples

```
# nmap -p 80 10.11.12.0/24
```

- This performs a SYN scan on port 80 only
- The target is a /24 network

```
# nmap -sT -p 1-100,443 192.168.12.33,45
```

- This performs a TCP connect scan on the first 100 ports and on port 443
- The targets are `192.168.12.33` and `192.168.12.45`

2.3.6 Video – Portscanning

Portscanning

In this video, you will see different *Nmap* scan types in action.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

2.3.7 Discovering Network with Port Scanning

In your penetration testing career, you might encounter networks that are protected by firewalls and where pings are blocked.

For such cases, you should be armed with at least one backup plan.

2.3.7 Discovering Network with Port Scanning

It’s not uncommon to come across a server that does not respond to pings but has many TCP or UDP ports open.

When using nmap, you can use the **-Pn** switch to force the scan on such a server. (skip ping scanning and treat it as alive)

But how does that help in discovering potential targets?

2.3.7 Discovering Network with Port Scanning

You should be aware that there are some ports that are almost always open on certain systems.

If you would like to find an „alive” host, you can scan typical ports instead of performing a ping sweep.

2.3.7 Discovering Network with Port Scanning

Typically, open ports are signs of the following services running:

Protocol	Port Numbers	Service
TCP	22	SSH
TCP	80,443	HTTP/HTTPs web server
TCP	445	Windows shares (SMB), also Linux equivalent – Samba service
TCP	25	SMTP (Simple Mail Transfer Protocol)
TCP	21	FTP (File Transfer Protocol)
TCP	137-139	Windows NetBIOS services
TCP	1433-1434	1433-1434 MSSQL Database
TCP	3306	MySQL Database
TCP	8080,8443	HTTP(s) web server, HTTP proxy
UDP	53	DNS

2.3.7 Discovering Network with Port Scanning

You can now use the aforementioned technique as an alternative to ping sweep, for identifying “live” hosts.

The four most basic TCP ports (**22, 445, 80, 443**) can be used as indicators of “live” hosts in the network.

2.3.8 Spotting a Firewall

When trying to discover alive hosts through port scanning, you should be prepared for potential false-positives and firewall presence.

2.3.8 Spotting a Firewall

During network reconnaissance, it is sometimes difficult to say if a firewall is in place.

You should keep in mind that the larger the network, the bigger the possibility some additional network protections and access control mechanisms are in place.

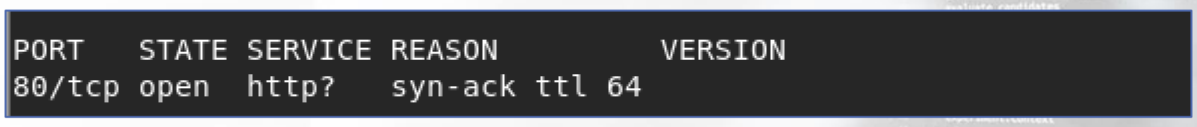
2.3.8 Spotting a Firewall

However, there are some ways to identify that a firewall is in place.

First, you pay attention to incomplete nmap results. On an open network, if a TCP scan succeeded against a well-known service, like a **web server**, nmap should not have any difficulty in trying to fingerprint it with **-sV** switch.

2.3.8 Spotting a Firewall

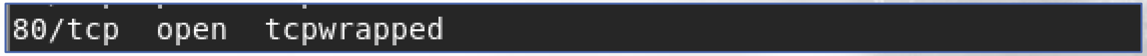
But, you might often see that a version was not recognized regardless of the open http port:



```
PORT   STATE SERVICE REASON         VERSION
80/tcp open  http?   syn-ack ttl 64
```

2.3.8 Spotting a Firewall

Or, that even the service type is not recognized:



```
80/tcp   open   tcpwrapped
```

„tcpwrapped” means that the **TCP handshake** was completed, but the remote host closed the connection without receiving any data.

2.3.8 Spotting a Firewall

This is an indication that something is blocking connectivity with the target host.

Moreover, you might want to use the nmap „**--reason**” switch that will show an explanation of why a port is marked open or closed.

2.3.8 Spotting a Firewall

For example, you may learn from it that the remote host sent an „**RST**” packet during the **TCP Handshake**, which means that something prevented the **TCP handshake** from being completed. Probably a firewall…

2.3.9 Masscan

Before concluding this module, another interesting tool that can help you to discover a network via probing TCP ports is Masscan.

2.3.9 Masscan

Masscan was designed to deal with large networks and to scan thousands of IP addresses at once.

It's like nmap, but a lot faster; however, it might be a bit less accurate. It's up to you which one you will use on a penetration testing assessment.

2.3.9 Masscan

You could perform host discovery using masscan, and then conduct a detailed scan with nmap against certain interesting hosts.

2.3.9.1 Video – Masscan

Basic Masscan Usage

In this video, you will see how masscan can be used to discover machines in a network.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

2.3.10 Hera Lab – Scanning and OS Fingerprinting

It is now time to practice what you learned in a hands-on lab!

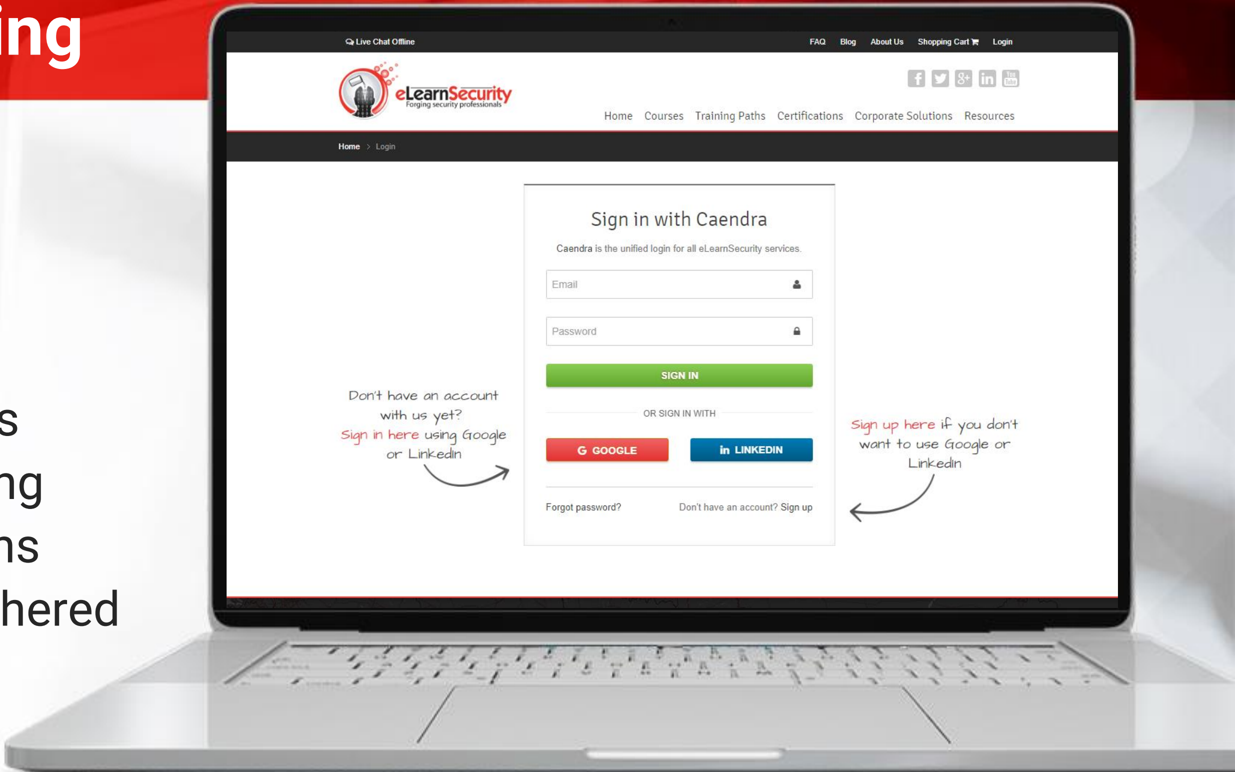
Try to do it by yourself, if you get stuck you can check the solutions in your members area.

2.3.10 Hera Lab – Scanning and OS Fingerprinting

Scanning and OS Fingerprinting

In this lab you will:

- Map computer networks
- Perform OS fingerprinting
- Perform many port scans
- Use the information gathered to... It's a secret!



Caendra is the unified login for all eLearnSecurity services.

Don't have an account with us yet? Sign in here using Google or LinkedIn

Sign up here if you don't want to use Google or LinkedIn

**Labs are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*

References

References

This concludes the module on footprinting and scanning. If you want to know more about *Nmap*, you can check out this book written by the founder of *Nmap* himself, Gordon "Fyodor" Lyon: "*Nmap Network Scanning*".

ICMP – RFC792

http://tools.ietf.org/html/rfc792

Bash Redirection

http://tldp.org/HOWTO/Bash-Prog-Intro-HOWTO-3.html

Nmap Network Scanning

http://nmap.org/book/

References

p0f

http://lcamtuf.coredump.cx/p0f3/

Nmap

http://nmap.org/

The Linux Documentation Project

http://tldp.org/

Remote OS Detection with Nmap

http://nmap.org/book/osdetect.html

Videos

OS Fingerprinting

You will learn basics of discovering information about your targets using nmap.

Port scanning

In this video, you will see different *Nmap* scan types in action.

****Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.***

Videos

Basic Masscan Usage

In this video, you will see how masscan can be used to discover machines in a network.

**Videos are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the resources drop-down in the appropriate module line.*

Labs

Scanning and OS Fingerprinting

In this lab you will:

- Map computer networks
- Perform OS fingerprinting
- Perform many port scans
- Use the information gathered to... It's a secret!

**Labs are only available in Full or Elite Editions of the course. To upgrade, click HERE. To access, go to the course in your members area and click the labs drop-down in the appropriate module line or to the virtual labs tabs on the left navigation.*