v4

Penetration Testing Student

Networking

Section 01 | Module 02

Table of Contents

Module 02 | Networking

Learning Objectives

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

- Modern network protocols
- Ways computers talk to each other
- Sniffing and capturing the network traffic

Protocols

2.1 Protocols

How does this support my pentesting career?

- Protocols are used in every computer network communication
- You need to know how things work to exploit them

2.1 Protocols

In a computer network, machines talk to each other by means of **protocols**.

These protocols ensure that different computers, using different hardware and software, can communicate.

2.1 Protocols

There is a large variety of networking protocols on the Internet, each one with its own purpose.

We are going to discuss a few of them in detail and point you towards free online resources for others.

2.1.1 Packets

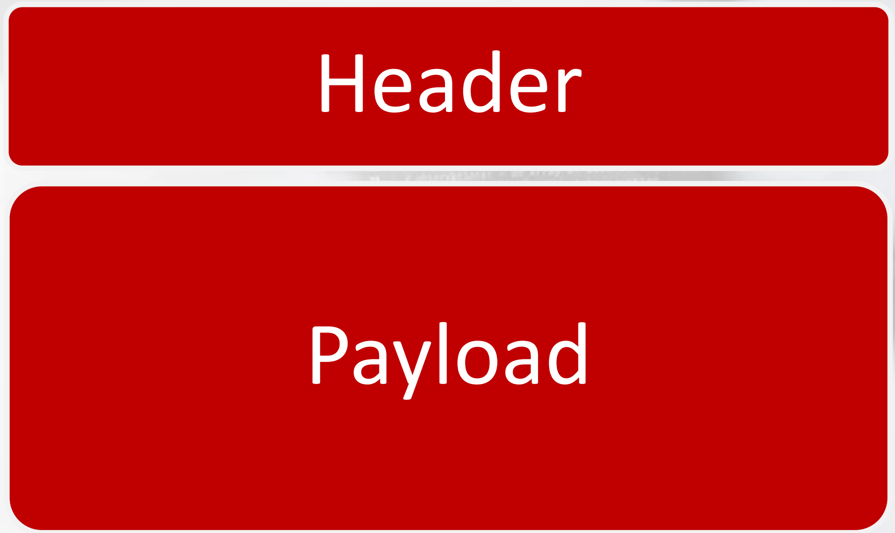
The primary goal of networking is to exchange information between networked computers; this information is carried by **packets**.

Packets are nothing but streams of bits running as electric signals on physical media used for data transmission. Such media can be a **wire** in a LAN or **the air** in a WiFi network.

These electrical signals are then interpreted as bits (zeros and ones) that make up the information.

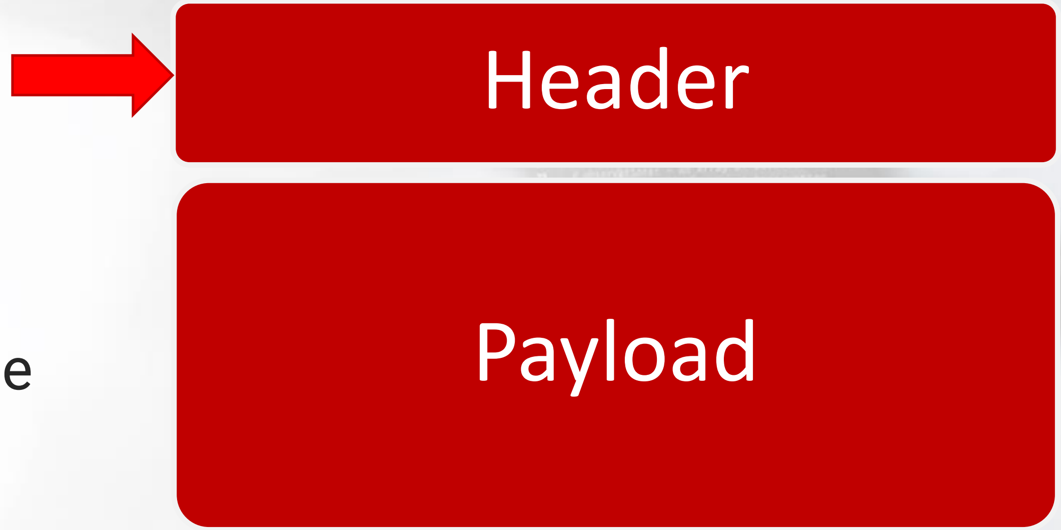
2.1.1 Packets

Every packet in every protocol has the following structure.



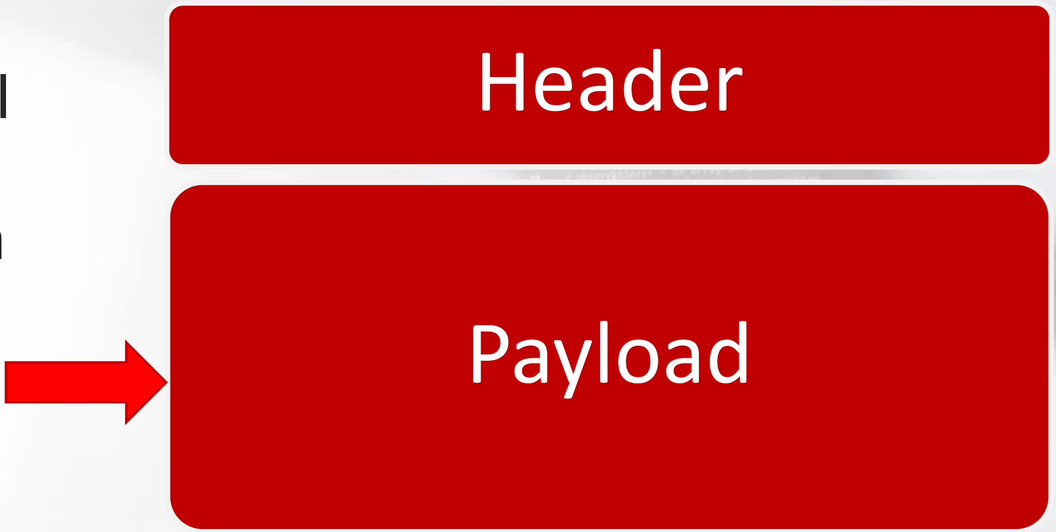
2.1.1 Packets

The **header** has a protocol-specific structure: this ensures that the receiving host can correctly interpret the payload and handle the overall communication.



2.1.1 Packets

The **payload** is the actual information. It could be something like part of an email message or the content of a file during a download.



2.1.1.1 Example – The IP Header

For example, the IP protocol header is at least 160 bits (20 bytes) long, and it includes information to interpret the content of the IP packet.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.1.1.1 Example – The IP Header

The first four bits identify the **IP version**. Today they can be used to represent IP version 4 or 6.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.1.1.1 Example – The IP Header

The 32 bits starting at position 96 represent the **source address**.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.1.1.1 Example – The IP Header

The following four bytes represent the **destination address**.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.1.1.1 Example – The IP Header

Using the information in the header, the nodes involved in the communication can understand and use IP packets.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Version|  IHL  |Type of Service|          Total Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Identification        |Flags|      Fragment Offset    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time to Live |    Protocol   |         Header Checksum       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Source Address                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Destination Address                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.1.2 Protocol Layers

In the previous example, we saw the header of the IP (**Internet Protocol**). There are many protocols out there, each one for a specific purpose. Purposes such as:

- Exchanging emails, files or performing VoIP calls
- Establishing a communication between a server and a client
- Identifying computers on a network
- Transmitting data

2.1.2 Protocol Layers

Instead of using specific examples, let's rewrite the previous list focusing on the features that a protocol provides:

- Make an application (email client, FTP, browser, …) work
- Transport data between processes (the server and the client programs)
- Identify hosts
- Use the physical media to send packets

2.1.2 Protocol Layers

Moreover, we can rewrite the list again as:

- Application layer
- Transport layer
- Network layer
- Physical layer

These **layers** work on top of one another, and every layer has its own **protocol**.

EXAMPLE

2.1.2 Protocol Layers

Each layer serves the one above it.

The **application layer** does not need to know how to **identify a process** on a host, how to **reach it** and how to **use the copper wire** to establish a communication.

It just uses its underlying layers.

2.1.3 ISO/OSI

In 1984, the International Organization for Standardization (ISO) published a theoretical model for network systems communication: the Open System Interconnection (OSI) model.

The **ISO/OSI** model was never implemented, but it is widely used in literature or when talking about IT networks.

2.1.3 ISO/OSI

ISO/OSI consists of seven layers and is used as a reference for the implementation of actual protocols.

You can find more information about ISO/OSI here.

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

http://support.microsoft.com/kb/103884

2.1.4 Encapsulation

But how do protocols work together? If every protocol has a header and a payload, how can a protocol use the one on its lower layer?

The idea is simple. The **entire upper protocol packet** (header plus payload) is the **payload** of the lower one; this is called **encapsulation**.

2.1.4 Encapsulation

In the following slides, you will see how encapsulation is used by the IP protocol suite, or TCP/IP.

TCP/IP is a real-world implementation of a networking stack and is the protocol stack used on the Internet.

2.1.4 Encapsulation

TCP/IP has four layers:

Application

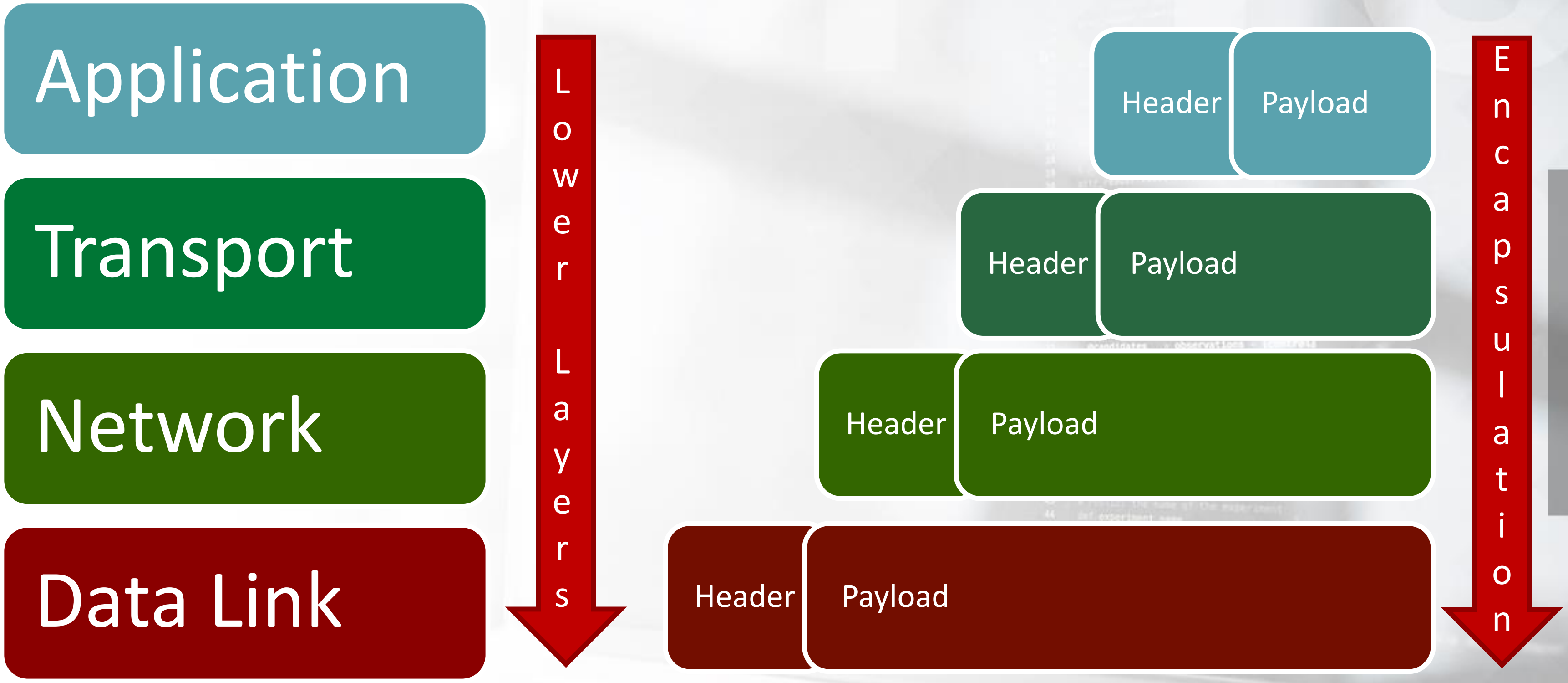
Transport

Network

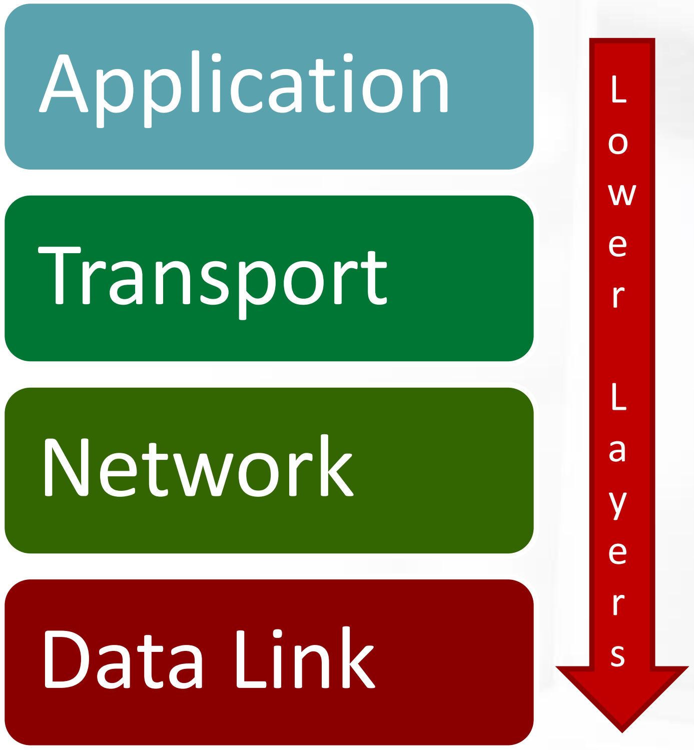
Data Link

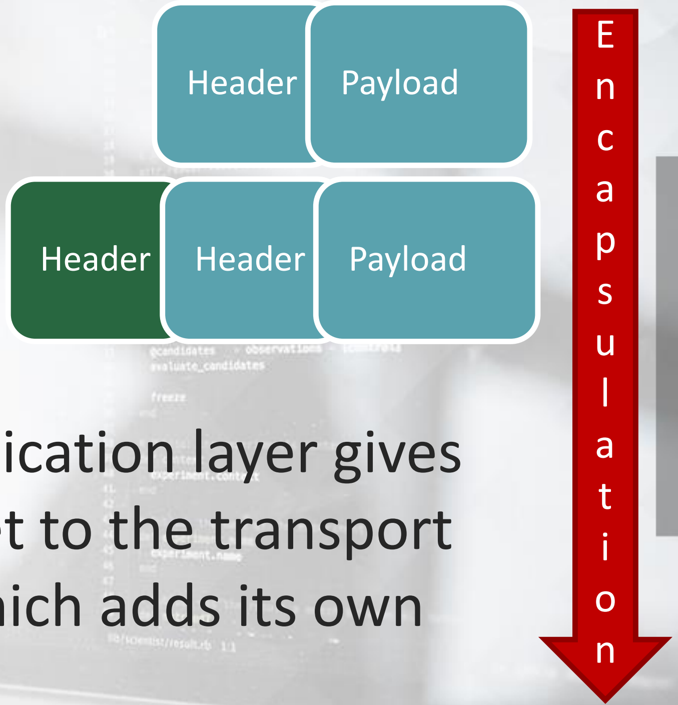
You will learn how TCP/IP works in the remainder of this module.

2.1.4 Encapsulation



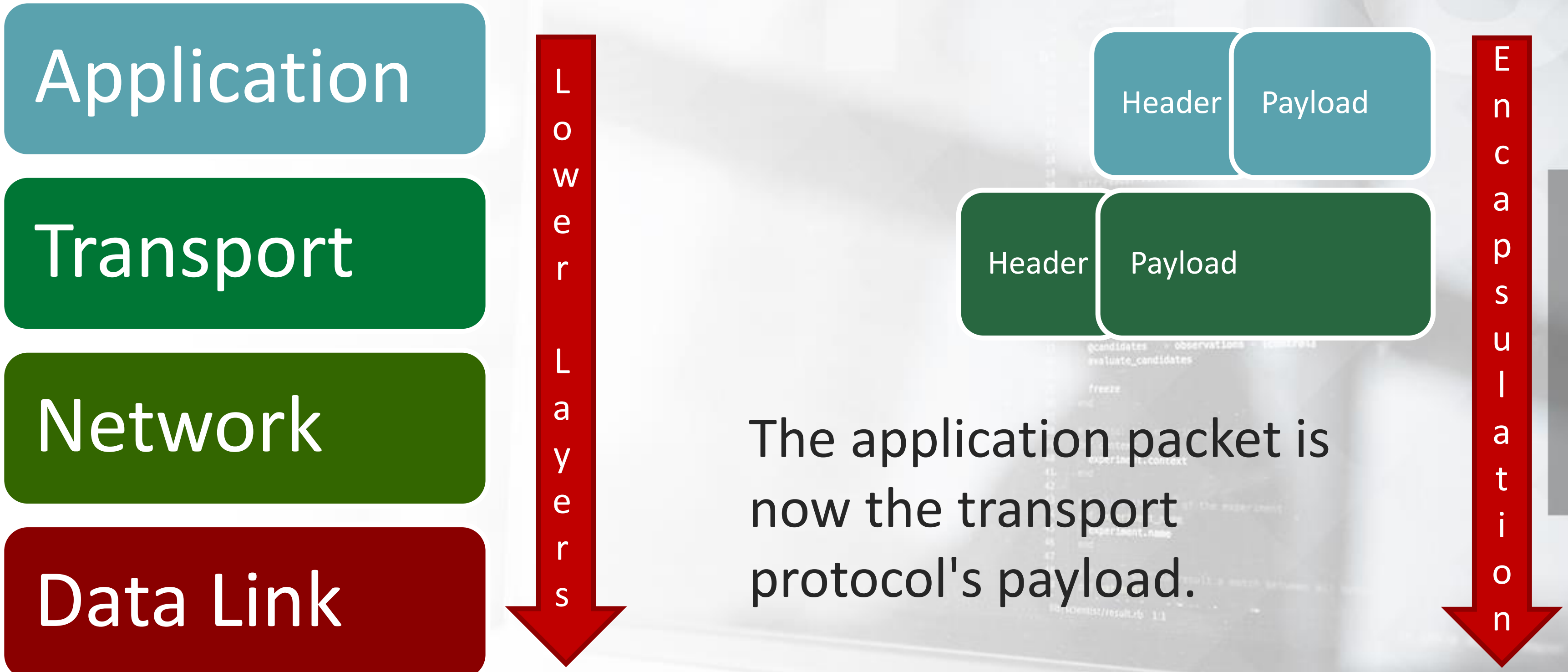
2.1.4 Encapsulation





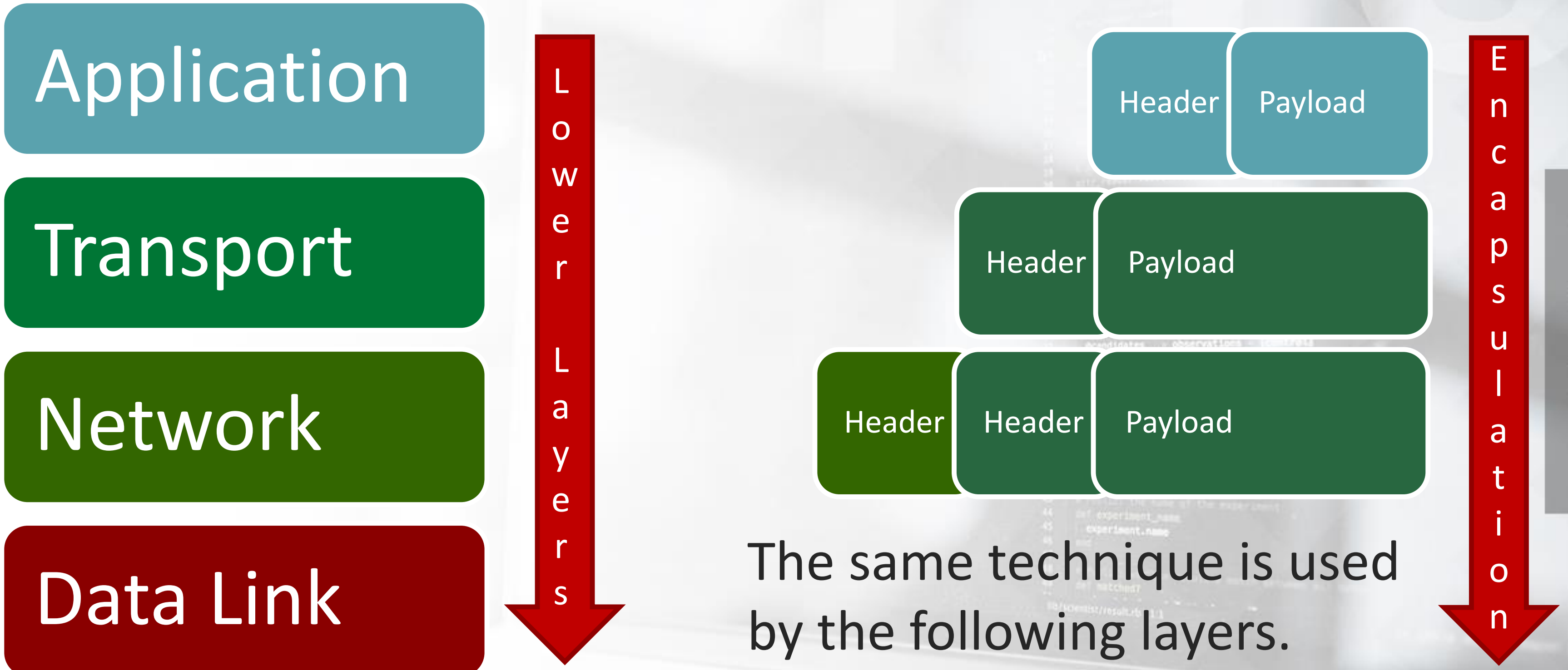
The application layer gives its packet to the transport layer, which adds its own header.

2.1.4 Encapsulation



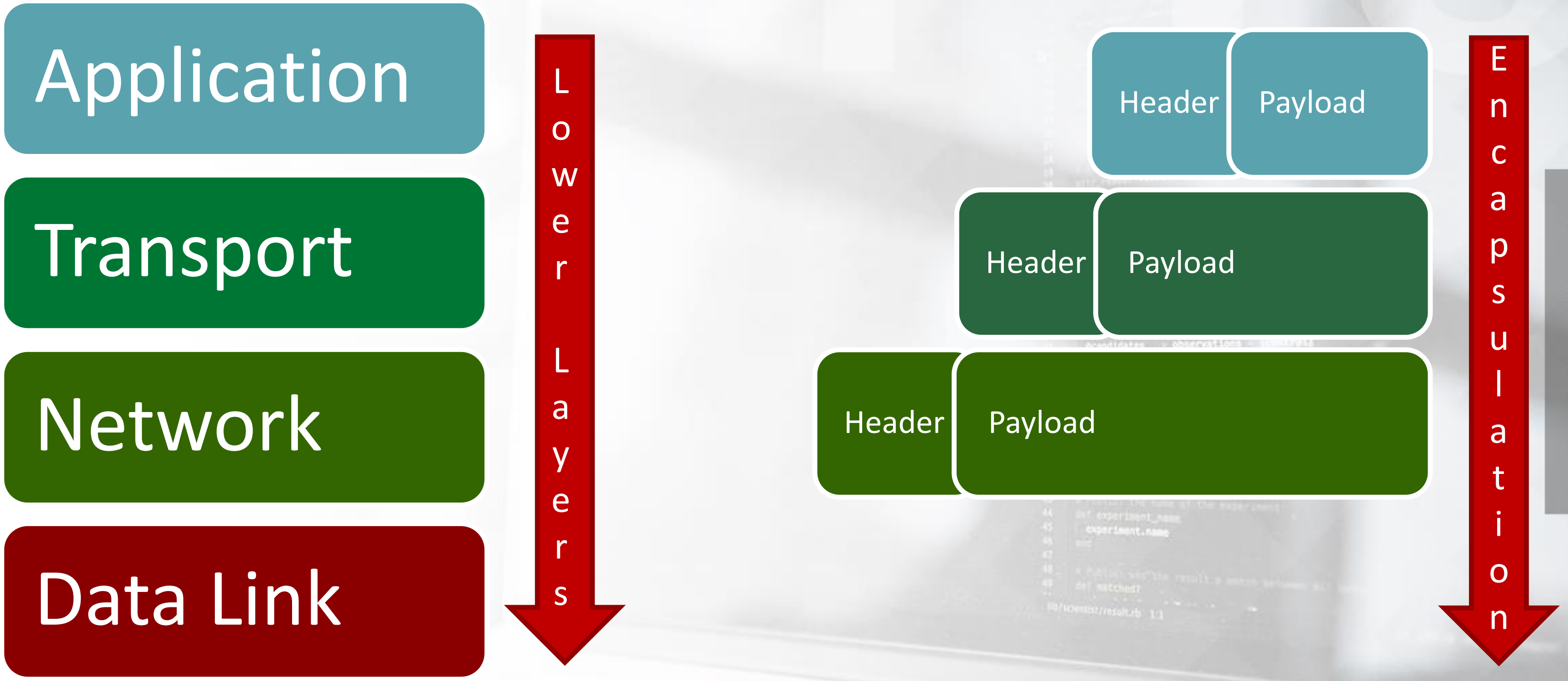
The application packet is now the transport protocol's payload.

2.1.4 Encapsulation

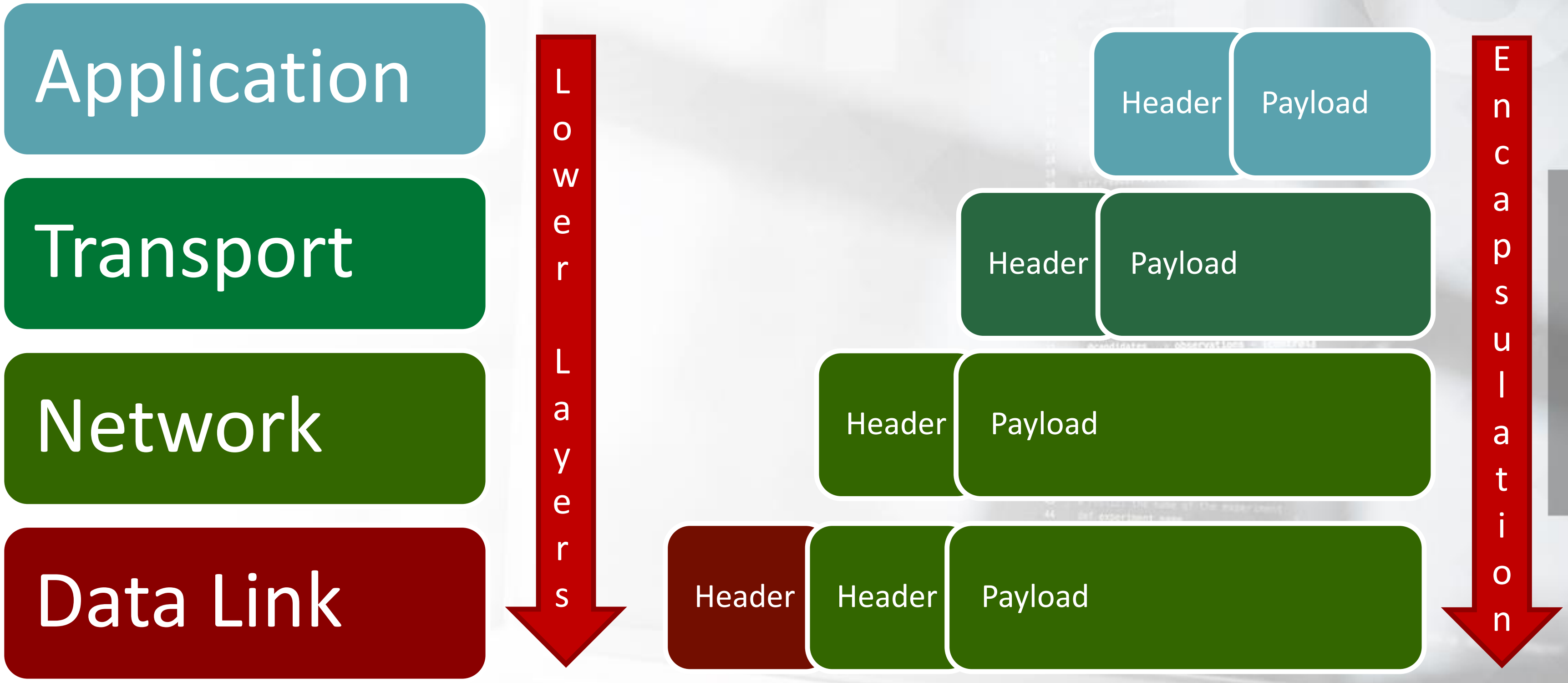


The same technique is used by the following layers.

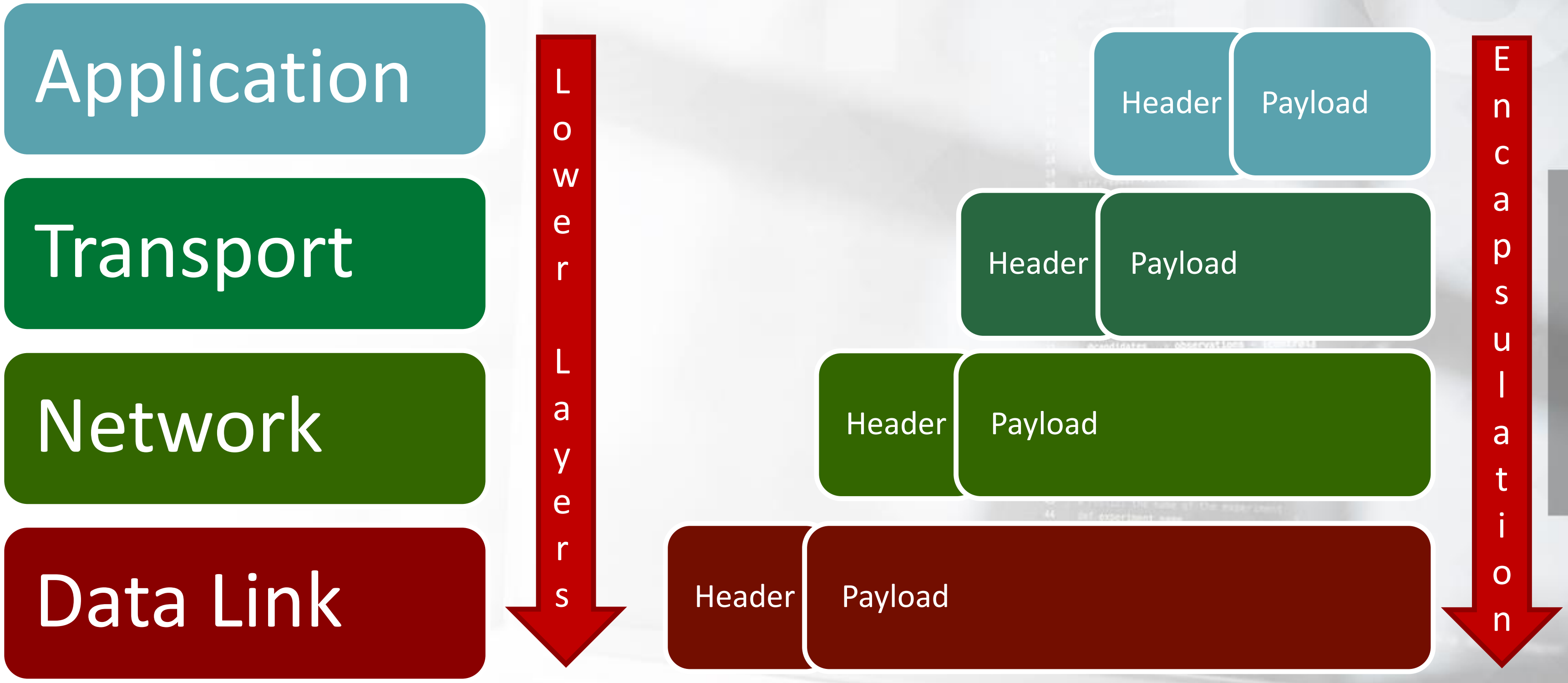
2.1.4 Encapsulation



2.1.4 Encapsulation



2.1.4 Encapsulation

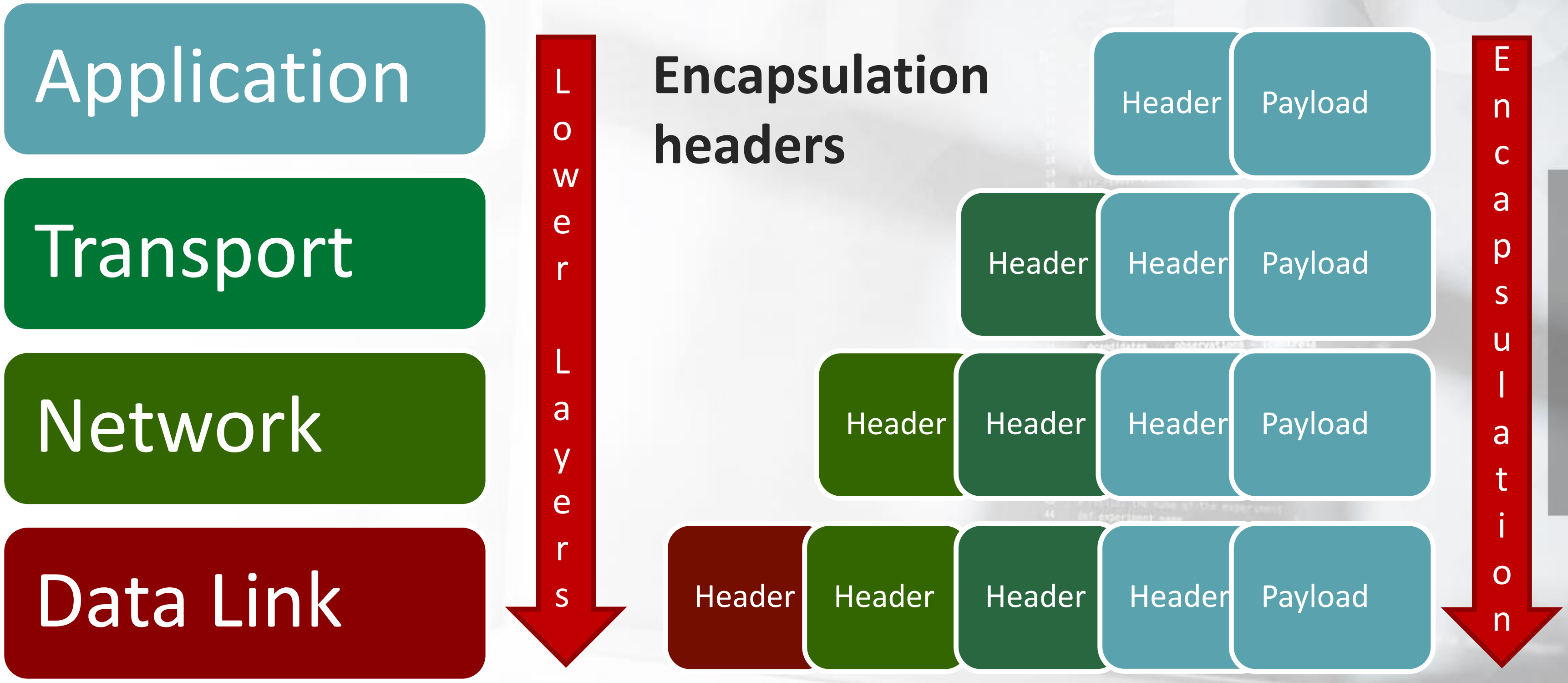


2.1.4 Encapsulation

During encapsulation every protocol adds its own header to the packet, treating it as a payload.

This happens to **every packet** sent by a host.

2.1.4 Encapsulation



2.1.4 Encapsulation

The receiving host does the same operation in reverse order. Using this method, the application does not need to worry about how the transport, network and link layers work. It just hands in the packet to the transport layer.

You will see encapsulation in practice later, during the Wireshark section.

2.2

Internet Protocol (IP)

2.2 IP

How does this support my pentesting career?

- Understanding network attacks
- Using network attack tools at their maximum
- Studying other networking protocols

2.2 IP

The **Internet Protocol** (IP) is the protocol that runs on the **Internet** layer of the Internet Protocol suite, also known as TCP/IP.

IP is in charge of delivering the **datagrams** (IP packets are called datagrams) to the hosts involved in a communication, and it uses **IP addresses** to identify a host.

2.2.1 IPv4 Addresses

When you write a letter, you have to specify the recipient's **address** on the envelope before sending it. Similarly, the Internet uses its addressing scheme to deliver packets to the right destination.

Any host on a computer network, be it a private network or the Internet, is identified by a **unique IP address**.

EXAMPLE

2.2.1 IPv4 Addresses

The vast majority of networks run IP **version 4** (IPv4).

An IPv4 address consists of four bytes, or octets; a byte consists of 8 bits.

```
73.5.12.132
```

2.2.1 IPv4 Addresses

A dot delimits every octet in the address.

$$\underbrace{73.}_{\text{First}}\underbrace{5.}_{\text{Second}}\underbrace{12.}_{\text{Third}}\underbrace{132}_{\text{Fourth}}$$

2.2.1 IPv4 Addresses

As you may recall from the introduction module, with 8 bits, you can represent **up to 2^8 different values** from 0 to 255.

This does not mean that you can **assign** any address starting from 0.0.0.0 to 255.255.255.255 to a host. Some addresses are **reserved** for special purposes.

2.2.2 Reserved IPv4 Addresses

For example, some reserved intervals are:

- **0.0.0.0 – 0.255.255.255** representing "this" network.
- **127.0.0.0 – 127.255.255.255** representing the local host (e.g., your computer).
- **192.168.0.0 – 192.168.255.255** is reserved for private networks.

You can find the details about the special use of IPv4 addresses in RFC5735.

http://tools.ietf.org/html/rfc5735

EXAMPLE

2.2.3 IP/Mask

To fully identify a host, you also need to know its **network**. To do that, you will need an IP address and a **netmask**, or subnet mask.

With an IP/netmask pair, you can identify the network part and the host part of an IP address.

IP address:	192.168.5.100
Subnet mask:	255.255.255.0

2.2.3 IP/Mask

To find the network part you have to perform a **bitwise *AND* operation** between the netmask and the IP address.

In the following example, we are going to see how to find the network part of this IP address/mask pair:

```
192.168.33.12/255.255.224.0
```

2.2.3.1 IP/Mask CIDR Example

1

Convert the octets in binary form:

```
192.168.33.12

11000000.10101000.00100001.00001100

255.255.224.0

11111111.11111111.11100000.00000000
```

2.2.3.1 IP/Mask CIDR Example

2 Perform the *bitwise AND*:

IP: 11000000.10101000.00100001.00001100

&

Mask: **11111111.11111111.111**00000.00000000

=

Network: 11000000.10101000.00100000.00000000

Network prefix in decimal notation: 192.168.32.0

2.2.3.1 IP/Mask CIDR Example

192.168.32.0 is the **network prefix**. You can identify the network by using the following notation:

```
192.168.32.0/255.255.224.0
```

Or, as the netmask is made by 19 consecutive "1" bits:

```
192.168.32.0/19
```

The latter is the **Classless Inter-Domain Routing (CIDR)** notation.

2.2.3.2 IP/Mask Host Example

The address part not covered by the netmask is the **host part** of the IP address. You can find it by performing a bitwise *AND* with the **inverse of the netmask**.

Let’s look at an example with the same IP/mask.

2.2.3.2 IP/Mask Host Example

1

Convert the octets in binary form:

192.168.33.12

11000000.10101000.00100001.00001100

255.255.224.0

11111111.11111111.11100000.00000000

2.2.3.2 IP/Mask Host Example

2

Invert the netmask by performing a *bitwise NOT*:

$$\neg(11111111.11111111.11100000.00000000)$$

$$=$$

$$00000000.00000000.00011111.11111111$$

2.2.3.2 IP/Mask Host Example

3

Perform the final *bitwise AND*:

IP: 11000000.10101000.00100001.00001100

&

¬Mask: 00000000.00000000.000**11111**.**11111111**

=

Host: 00000000.00000000.00000001.00001100

Host part in decimal notation: 0.0.1.12

2.2.3.2 IP/Mask Host Example

Moreover, the inverse of the netmask lets you know how many hosts a network can contain.

In our example, we have 13 bits to represent the hosts; this means that the network can contain 2^{13} = **8192 different addresses**.

2.2.4 Network and Broadcast Addresses

There are two special addresses:

- One with the host part made by all zeros.
- Another with the host part made by all ones.

These special addresses **were** used as the **network** and **broadcast** addresses, thus reducing by 2 the number of hosts on a given network. This technical limitation should be extinct (RFC1878) but is still used to keep compatibility with old equipment.

http://tools.ietf.org/html/rfc1878

2.2.5 IP Examples

Let’s recap by going over some IP examples.

2.2.5 IP Examples

10.54.12.0/24 (10.54.12.0/255.255.255.0)

- Contains $2^8 = 256$ addresses
- 10.54.12.0 is the network address according to the pre-CIDR standard
- 10.54.12.255 is the broadcast address according to the pre-CIDR standard

2.2.5 IP Examples

192.168.114.32/27 (192.168.114.32/255.255.255.224)

- Contains $2^5 = 32$ addresses
- 192.168.114.32 is the pre-CIDR network address
- 192.168.114.63 is the pre-CIDR broadcast address

2.2.5 IP Examples

Given the network 172.16.2.0/23

- 172.16.3.12 and 172.16.2.66 **are** in the same network
- 172.16.3.240 and 172.16.4.2 **are not** in the same network

The network 192.168.1.0/16

- Does not make sense; a bitwise *AND* between 192.168.1.0 and 255.255.0.0 leads to 192.168.0.0 as network address
- Could be a valid IP address in the 192.168.1.0/16 network

2.2.6 Subnet Calculators

You can practice more on this topic by using a subnet calculator.

Here are two subnet calculators you can check out:

- A classful calculator
- A CIDR calculator

http://www.subnet-calculator.com/
http://www.subnet-calculator.com/cidr.php

2.2.7 IPv6

IPv4 addresses are being consumed rapidly due to a large number of new devices connecting to the internet every day.

One day IPv4 addresses might be exhausted.

2.2.7 IPv6

As a 32-bit address, **IPv4** has 2^32 = **4.294.967.296** possible addresses.

While a 128-bit **IPv6** address has **2^128 = 2^32 * 2^96** possible addresses.

2^96 is equal to **79 octillion addresses**

2.2.7 IPv6

An **IPv6** address consists of **16-bit hexadecimal numbers** separated by a **colon (:)**. Hexadecimal numbers are case insensitive. In case zeros occur, they can be skipped.

Let’s check out some IPv6 examples on the next slide.

EXAMPLE

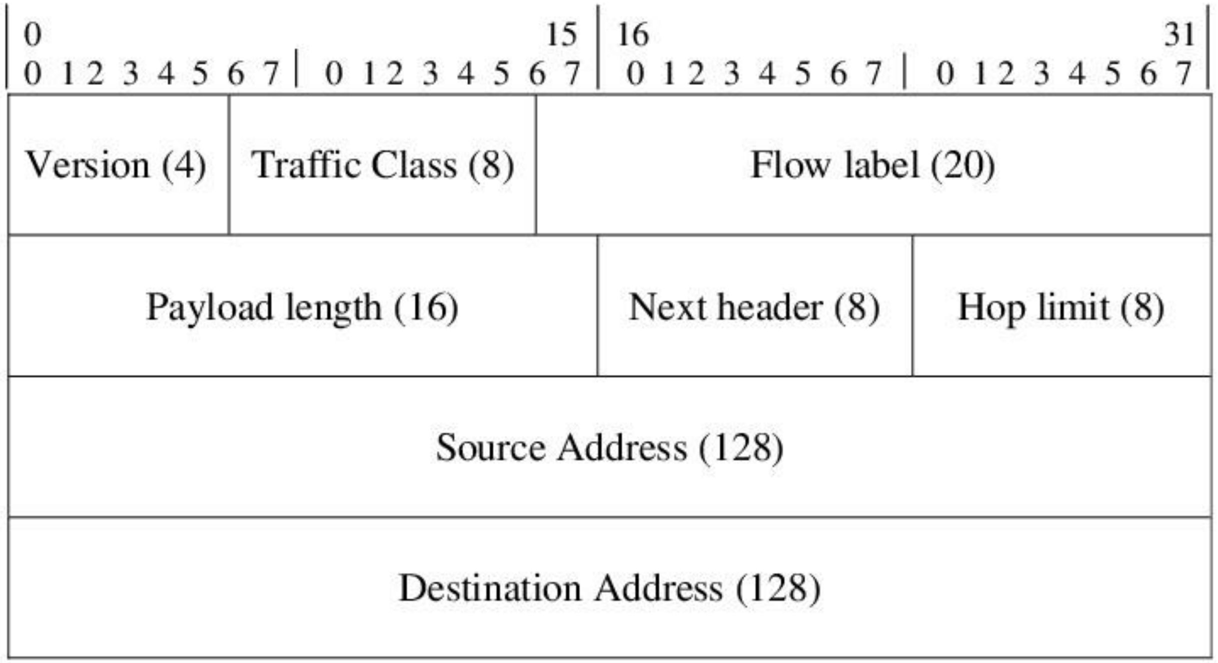
2.2.7 IPv6

IPv6 addresses examples:

2001:0db8:0020:130F:0000:0000:087C:140B

2001:0db8:0:160F::850C:140B

2.2.7.1 IPv6 header



https://www.researchgate.net/profile/Dragos_Truscan/publication/31596630/figure/download/fig5/AS:340685715722244@1458237212506/IPv6-Header-Format.png

2.2.7.2 IPv6 forms

IPv6 can be presented in following text representations:

- Regular form: **1080:0:FF:0:8:800:200C:417A**
- Compressed form: **FF01:0:0:0:0:0:0:43** becomes **FF01::43** as a result of skipping zeros
- IPv4-compatible: **0:0:0:0:0:0:13.1.68.3** or **::13.1.68.3** after skipping zeros

2.2.7.3 IPv6 Reserved Addresses

IPv6 also has reserved addresses, which cannot be used like the reserved IPv4 ones.

For example:

- **::1/128 is a loopback address**
- **::FFFF:0:0/96 are IPv4 mapped addresses**

For more information, you can check RFC3513

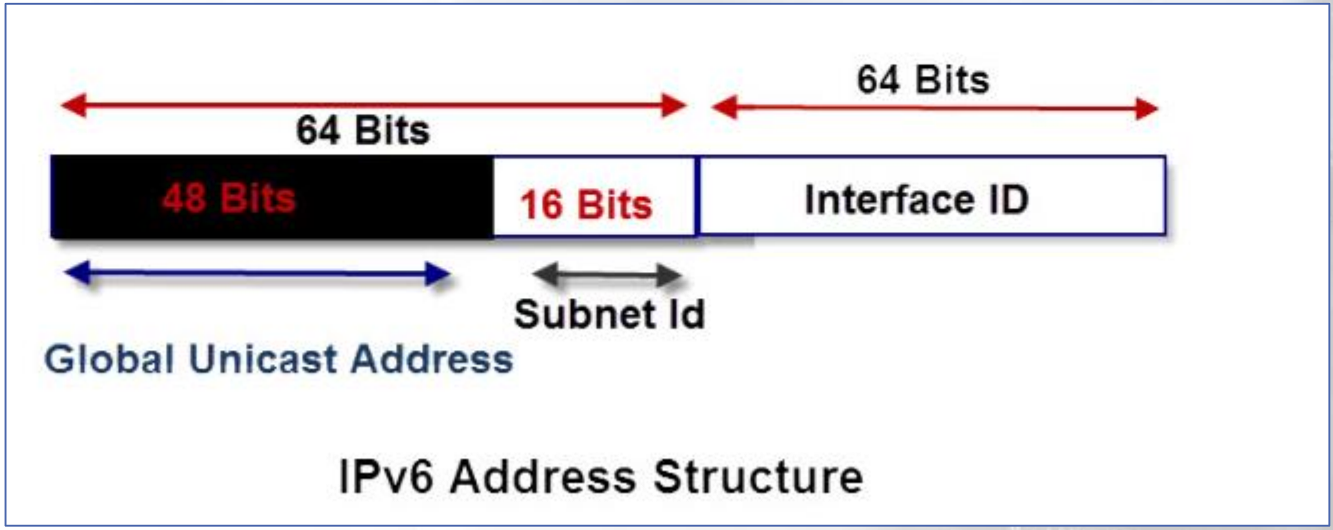
https://tools.ietf.org/html/rfc3513

2.2.7.4 IPv6 Structure

An **IPv6 address** can be split in half (64 bits each) into a **network part** and a **device part**.

Furthermore, the **first 64 bits** ends with a **dedicated 16-bits space** (one hex word) that can be used only for **specifying a subnet.**

2.2.7.4 IPv6 Structure



2.2.7.5 IPv6 Scope

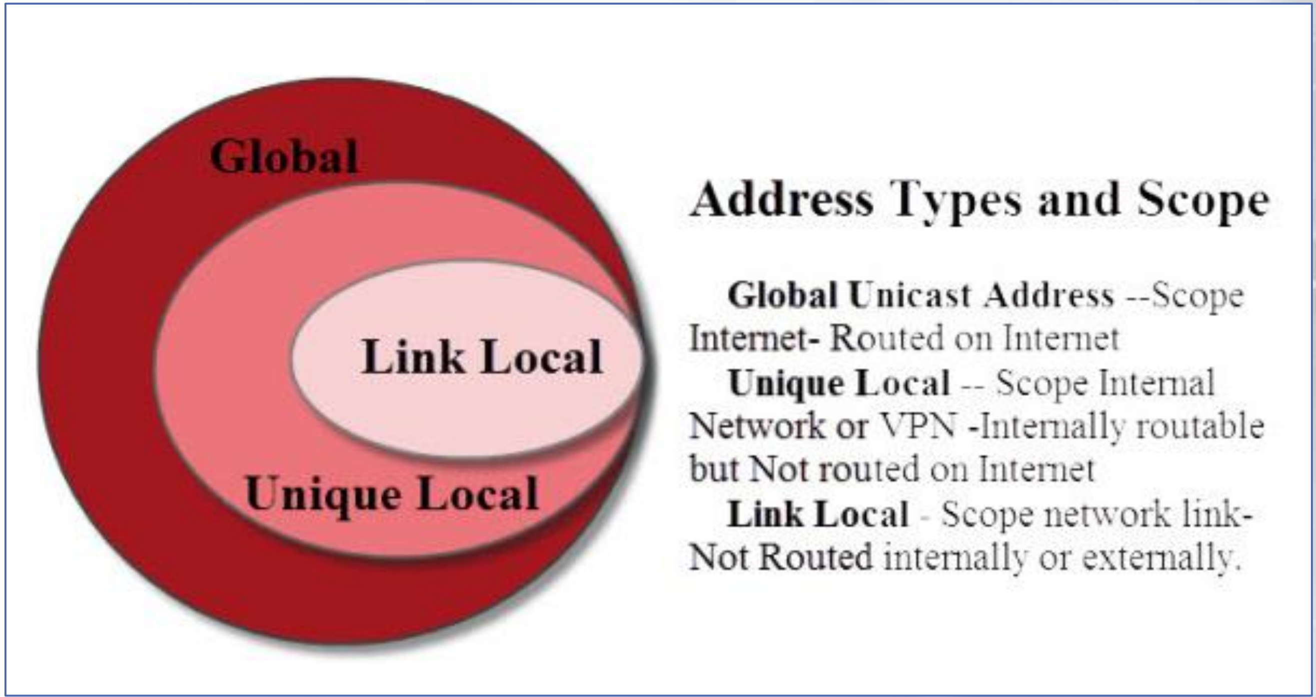
Address Types and Scope

IPv6 addresses have three types:

- **Global Unicast Address** – These addresses are global ones, and reside in global internet.

- **Unique Local and Link Local** – reside only in Internal Networks.

2.2.7.5 IPv6 Scope



Address Types and Scope

Global Unicast Address --Scope Internet- Routed on Internet

Unique Local -- Scope Internal Network or VPN -Internally routable but Not routed on Internet

Link Local - Scope network link- Not Routed internally or externally.

http://www.steves-internet-guide.com/ipv6-guide/

2.2.7.6 IPv6 Translation

IPv6 addresses can also be translated to **binary**.

One 4-digit hex word represents **16 binary digits**; we can see this demonstrated in the following way:

- Bin **0000000000000000** = Hex 0000 (or just 0)
- Bin **1111111111111111** = Hex FFFF
- Bin **1101010011011011** = Hex D4DB

2.2.7.6 IPv6 Translation

Thus, 128-bit binary address looks like:

1111111111111111.1111111111111111.1111111111111111.1111111111111111.1111111111111111.1111111111111111.1111111111111111.1111111111111111

And, the above can be represented by 8 hex words, separated by colons:

FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF

2.2.7.7 IPv6 Subnets

Like IPv4, an IPv6 address has a network portion and a device portion.

Unlike IPv4, an IPv6 address has a dedicated subnetting portion. On the next few slides, we'll show how the ranges are divided in IPv6.

2.2.7.7 IPv6 Subnets

Network Address Range

In IPv6, the first 48 bits are for Internet global addressing.

1111111111111111.1111111111111111.11111111111111
11.0000000000000000.0000000000000000.00000000000
00000.0000000000000000.0000000000000000

2.2.7.7 IPv6 Subnets

Subnetting Range

The 16 bits from the 49th to the 64th are for defining subnets.

0000000000000000.0000000000000000.00000000000000
00.1111111111111111.0000000000000000.00000000000
00000.0000000000000000.0000000000000000

2.2.7.7 IPv6 Subnets

Device (Interface) Range

The last 64 bits are for device (interface) ID's:

0000000000000000.0000000000000000.00000000000000
00.0000000000000000.1111111111111111.11111111111
11111.1111111111111111.1111111111111111

2.2.7.8 IPv6 Subnetting

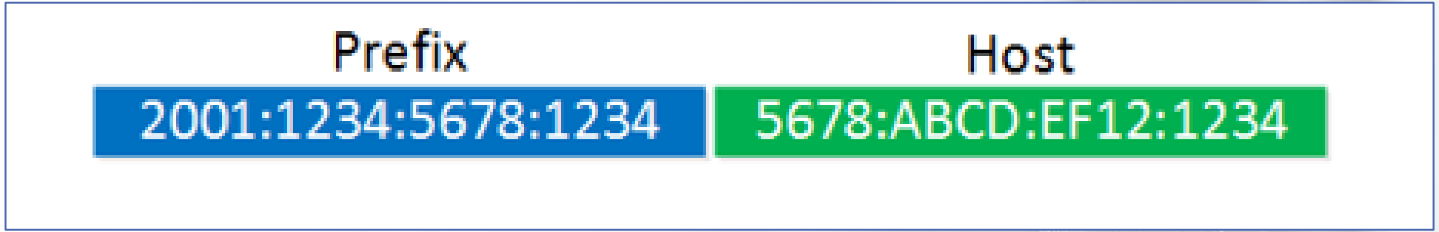
In **IPv6**, there are **prefixes** instead of subnets blocks. For example:

2001:1111:1234:1234::/64

In the above IPv6 address, the number after the slash (64) is the **number of bits that is used for a prefix**. Everything behind it can be used for **hosts** of the **subnet**.

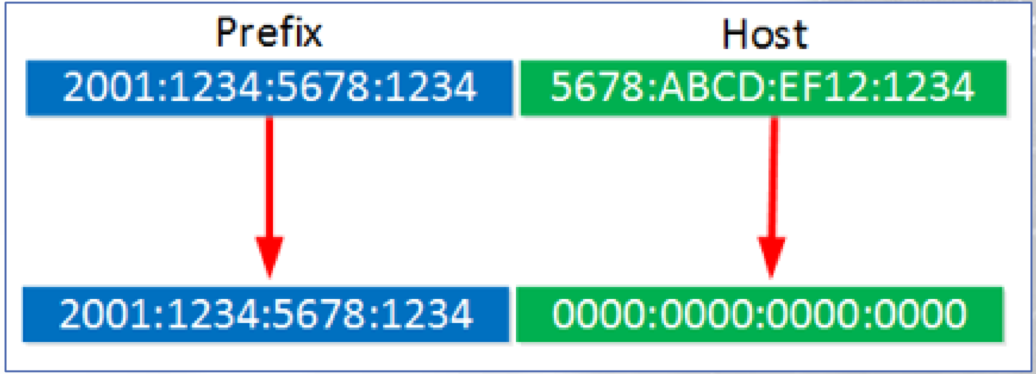
2.2.7.8 IPv6 Subnetting

As you may have noticed, /64 means that the first 64 bits are a prefix. And, as previously mentioned earlier, each 4-digit hex word is 16 bits, thus in following IPv6 address we can divide it as such:



2.2.7.8 IPv6 Subnetting

We confirmed that **2001:1234:5678:1234** is the prefix, but let's now focus on writing down a correctly formatted IPv6 address.



2.2.7.8 IPv6 Subnetting

2001:1234:5678:1234:0000:0000:0000:0000 is a valid prefix, but it can be shortened by omitting zeros, into following form:

2001:1234:5678:1234::/64

https://networklessons.com/ipv6/how-to-find-ipv6-prefix/

2.2.7.8 IPv6 Subnetting

You can practice more on this topic by using a subnet calculator.

Here is a calculator you can check out:

- IPv6 Calculator

2.3

Routing

2.3 Routing

How does this support my pentesting career?

- Understanding routing protocol attacks
- Performing network traffic inspection

2.3 Routing

Addressing devices is just half of the work needed to reach a host. Your packets need to follow a valid **path** to reach it.

Routers are devices connected to different networks at the same time. They are able to forward IP datagrams from one network to another. The forwarding policy is based on **routing protocols.**

2.3 Routing

Routing protocols are used to determine the best path to reach a network. They behave like a postman who tries to use the shortest path possible to deliver a letter.

A router inspects the destination address of every incoming packet and then forwards it through one of its interfaces.

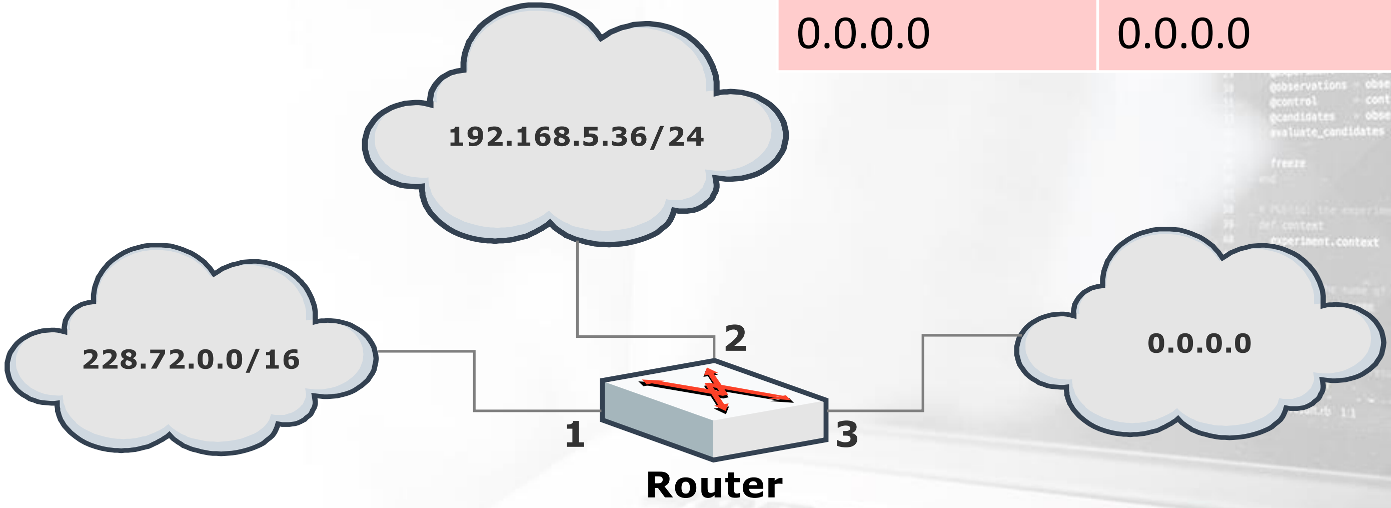
2.3.1 Routing Table

To choose the right forwarding interface, a router performs a lookup in the **routing table**, where it finds an IP-to-interface binding.

The table can also contain an entry with the **default address** (0.0.0.0). This entry is used when the router receives a packet whose destination is an *unknown network*.

2.3.1.1 Routing Table Example

IP	Netmask	Interface
228.72.0.0	255.255.0.0	1
192.168.5.0	255.255.255.0	2
0.0.0.0	0.0.0.0	3



2.3.1.1 Routing Table Example

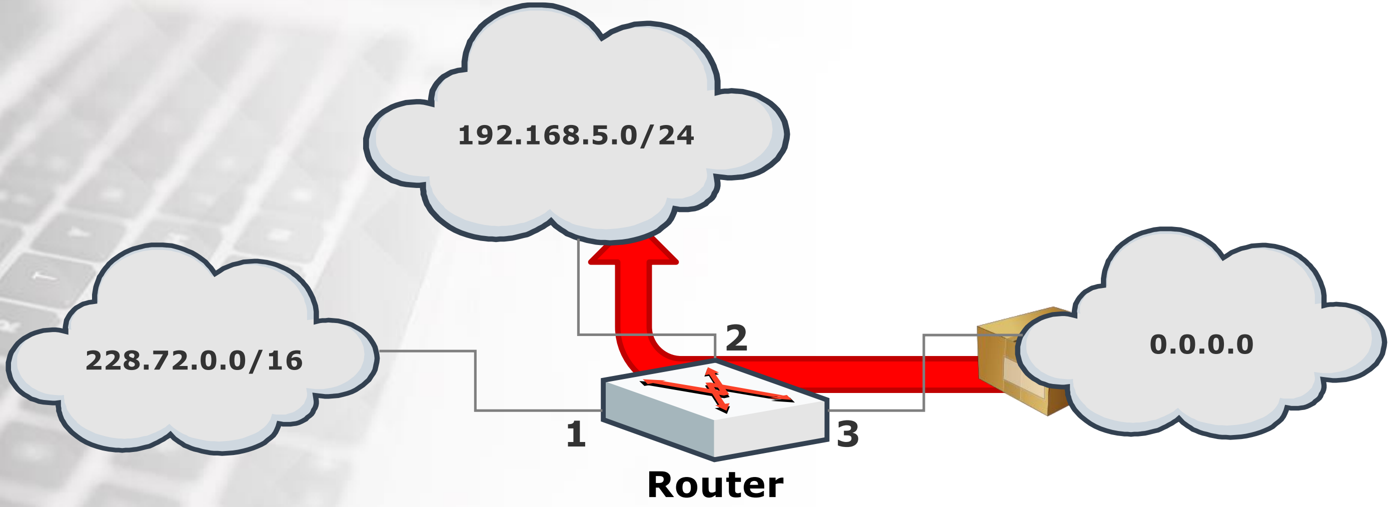
In this example, the routing table is made of three entries.

- Interface 1 is used to forward the packets to 228.72.0.0/16.
- Interface 2 is used to forward the packets to 192.168.5.0/24.
- Interface 3 is used as the default route for packets whose destination does not match any other entry in the table.

EXAMPLE

2.3.1.1 Routing Table Example

A packet arriving on interface 3 for 192.168.5.3 is forwarded on interface 2.



2.3.1.1 Routing Table Example

In fact, the first entry in the routing table does not match the destination network.

IP	Netmask	Interface
228.72.0.0	255.255.0.0	1 ✗
192.168.5.0	255.255.255.0	2
0.0.0.0	0.0.0.0	3

2.3.1.1 Routing Table Example

While the second does: `192.168.5.3` sits in the `192.168.5.0/24` network.

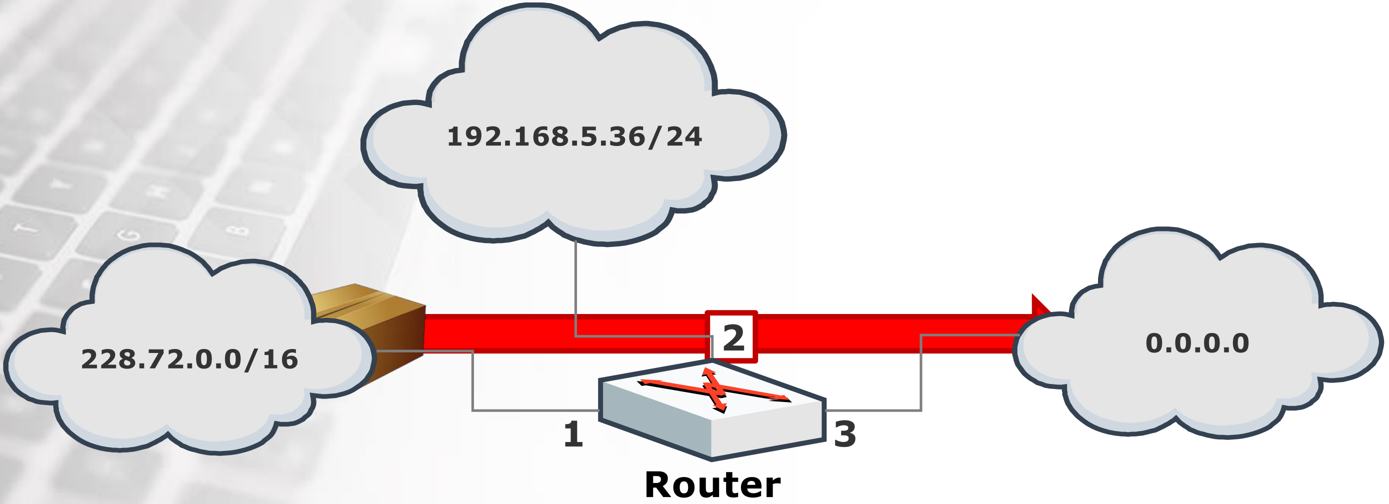
To: 192.168.5.3

IP	Netmask	Interface
228.72.0.0	255.255.0.0	1
192.168.5.0	255.255.255.0	2 ✓
0.0.0.0	0.0.0.0	3

EXAMPLE

2.3.1.2 Default Route Example

A packet arriving on interface 1 for 72.13.37.2 is routed through interface 3, the default route.



2.3.1.2 Default Route Example

There is no matching entry, so the router forwards the packet through interface 3.

To: 72.13.37.2

IP	Netmask	Interface
228.72.0.0	255.255.0.0	1
192.168.5.0	255.255.255.0	2
0.0.0.0	0.0.0.0	3 ✓

2.3.2 Routing Metrics

As in the real world, there could be more than a way to reach a destination.

So, during path discovery, routing protocols also assign a **metric to each link**.

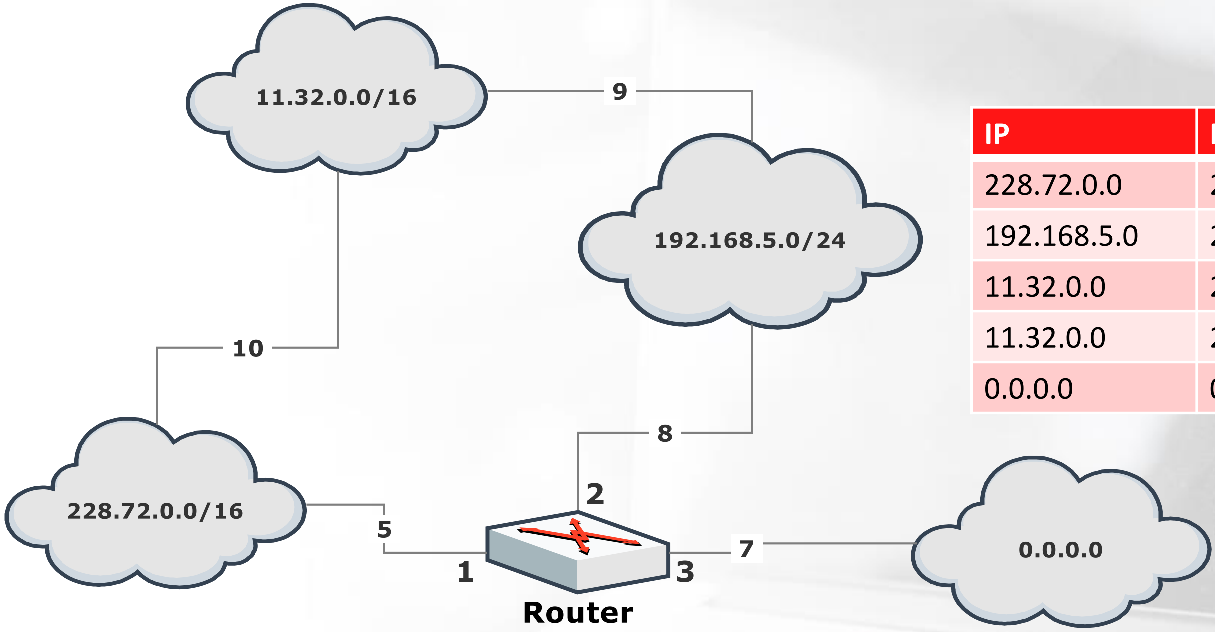
2.3.2 Routing Metrics

This ensures that, if two paths have the same number of hops, the fastest route is selected.

The metric is selected according to the channel's estimated bandwidth and congestion.

2.3.2.1 Routing Metrics Example

Let's look at how routing decisions are made according to metrics.

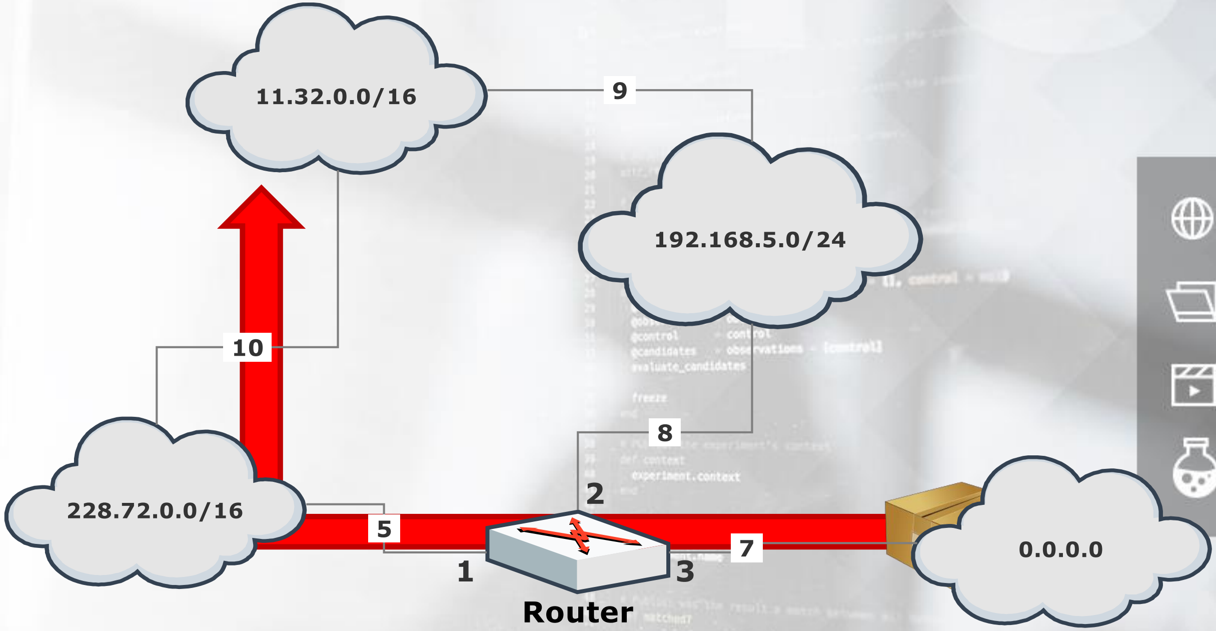


IP	Netmask	Interface	Metric
228.72.0.0	255.255.0.0	1	5
192.168.5.0	255.255.255.0	2	8
11.32.0.0	255.255.0.0	2	17
11.32.0.0	255.255.0.0	1	15
0.0.0.0	0.0.0.0	3	7

2.3.2.1 Routing Metrics Example

A packet arriving on interface 3 for 11.32.3.118 is routed through interface 1, as the metric for that route is 15.

Routing through interface 2 would have a metric of 17.



2.3.3 Checking the Routing Table

Routing tables are not only kept by routers; every host stores its own table.

To check what they look like, you can use:

- `ip route` on Linux
- `route print` on Windows
- `netstat -r` on OSX

EXAMPLE

2.3.3 Checking the Routing Table

Checking the routing table on a Linux box:

```
root@host:~# ip route
default via 192.168.51.1 dev eth0  proto static
192.168.51.0/24 dev wlan0  proto kernel  scope link  src 192.168.51.123
```

EXAMPLE

2.3.3 Checking the Routing Table

Checking the routing table on Microsoft Windows:

```
C:\Users\User>route print
===========================================================================
Interface List
 11...08 00 27 bf ac c8 ......Intel(R) PRO/1000 MT Desktop Adapter
  1...........................Software Loopback Interface 1
===========================================================================


IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.0.2.2       10.0.2.15     10
         10.0.2.0    255.255.255.0         On-link        10.0.2.15    266
```

EXAMPLE

2.3.3 Checking the Routing Table

Checking the routing table on Mac OSX:

```
User:~ user$ netstat -r
Routing tables

Internet:
Destination        Gateway            Flags        Refs      Use   Netif Expire
default            192.168.51.1       UGSc           13        0     en1
127                127.0.0.1          UCS             0        0     lo0
127.0.0.1          127.0.0.1          UH              1       16     lo0
169.254            link#4             UCS             0        0     en1
192.168.51         link#4             UCS             4        0     en1
192.168.51.1       58:6d:8f:e5:e:d2   UHLWIir        14       24     en1   1200
192.168.51.109     2:f:b5:4b:76:cf    UHLWIi          0        0     en1   1148
```

Link Layer Devices and Protocols

2.4 Link Layer Devices and Protocols

How does this support my pentesting career?

- MAC spoofing
- Testing switches security
- Sniffing techniques
- Man in the middle attacks

2.4 Link Layer Devices and Protocols

Packet forwarding also happens in the **lowest layer** of the TCP/IP stack: the **link layer**.

While routers are aware of the best overall path to the destination, link layer devices and protocols deal only with the next hop.

2.4 Link Layer Devices and Protocols

In this section you will see:

- How switches work
- Network card's MAC addresses
- The Address Resolution Protocol (ARP)

2.4.1 Link Layer Devices

Hubs and switches are network devices that forward **frames** (layer 2 packets) on a local network.

They work with link layer network addresses: **MAC addresses**.

2.4.2 Mac Addresses

IP addresses are the Layer 3 (Network layer) addressing scheme used to identify a host in a network, while **MAC addresses** uniquely identify a network card (Layer 2).

A MAC (Media Access Control) address is also known as the **physical address**.

2.4.2 Mac Addresses

MAC addresses are 48 bit (6 bytes) long and are expressed in hexadecimal form (HEX).

```
00:11:AA:22:EE:FF
```

2.4.2 Mac Addresses

To discover the MAC address of the network cards installed on your computer, you can use:

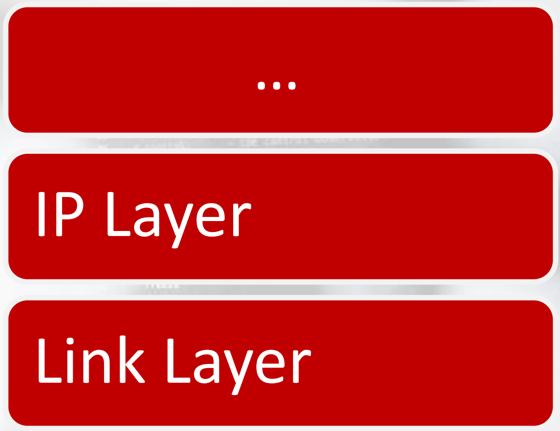
- `ipconfig /all` on Windows
- `ifconfig` on *nix operating systems, like MacOS
- `ip addr` on Linux

2.4.2 Mac Addresses

Every host on a network has both a MAC and an IP address.

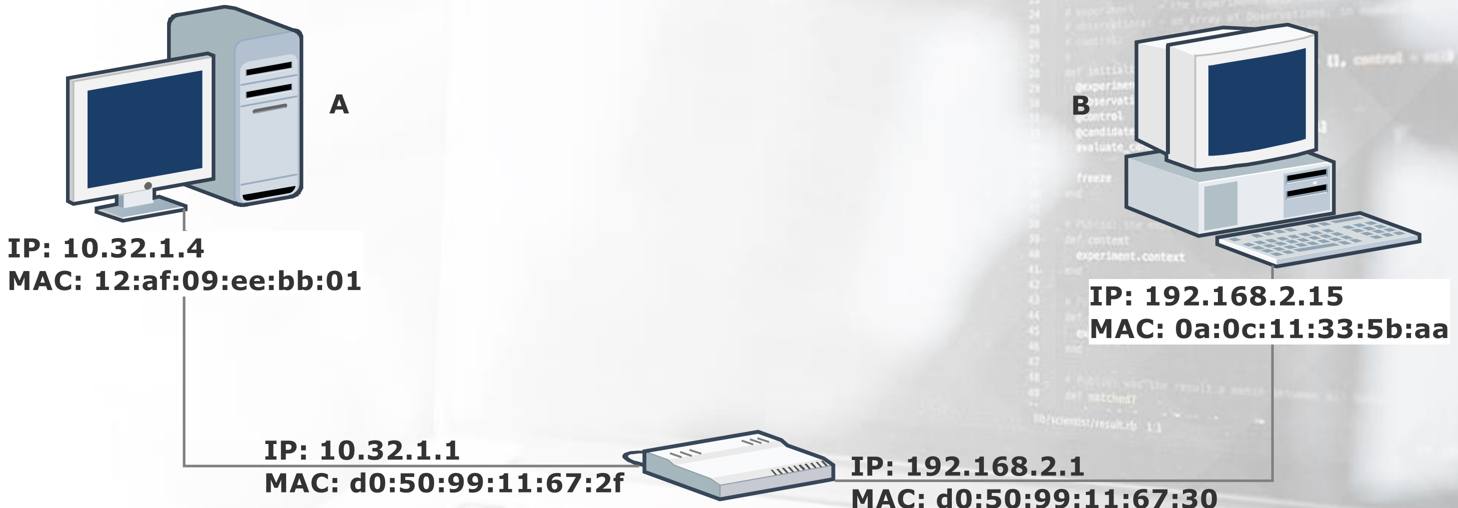
Let us see how they are used together to send packets.

Remember: the lower layer serves the layer above.



2.4.3 IP and MAC Addresses

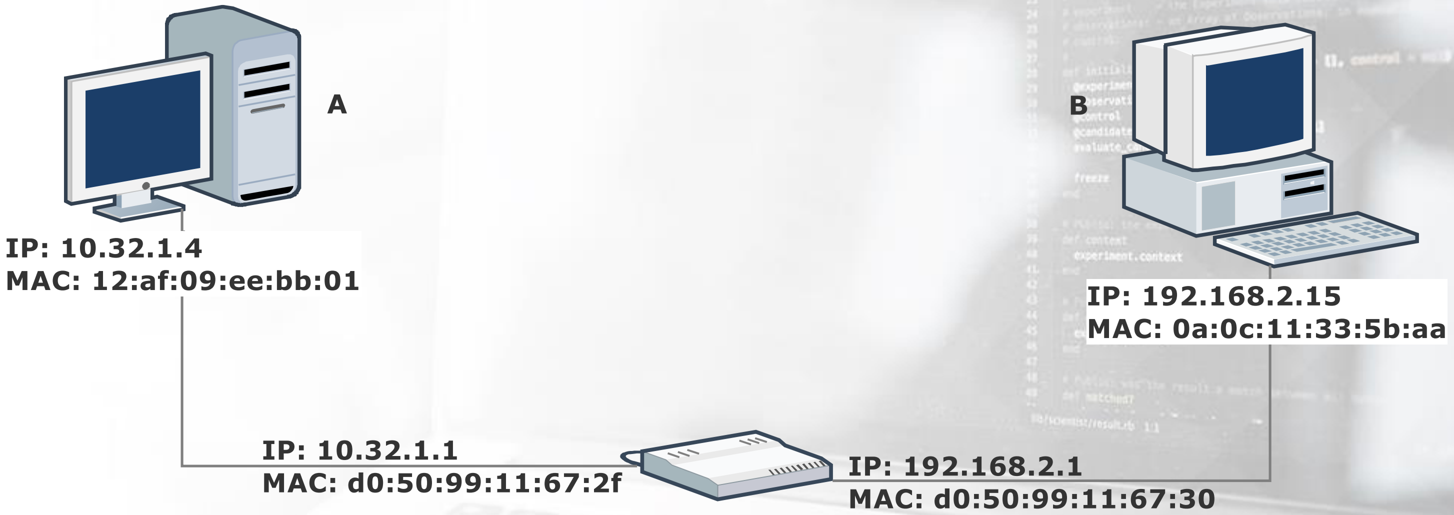
Let's take a look at an example to see how MAC addresses are used.



2.4.3 IP and MAC Addresses

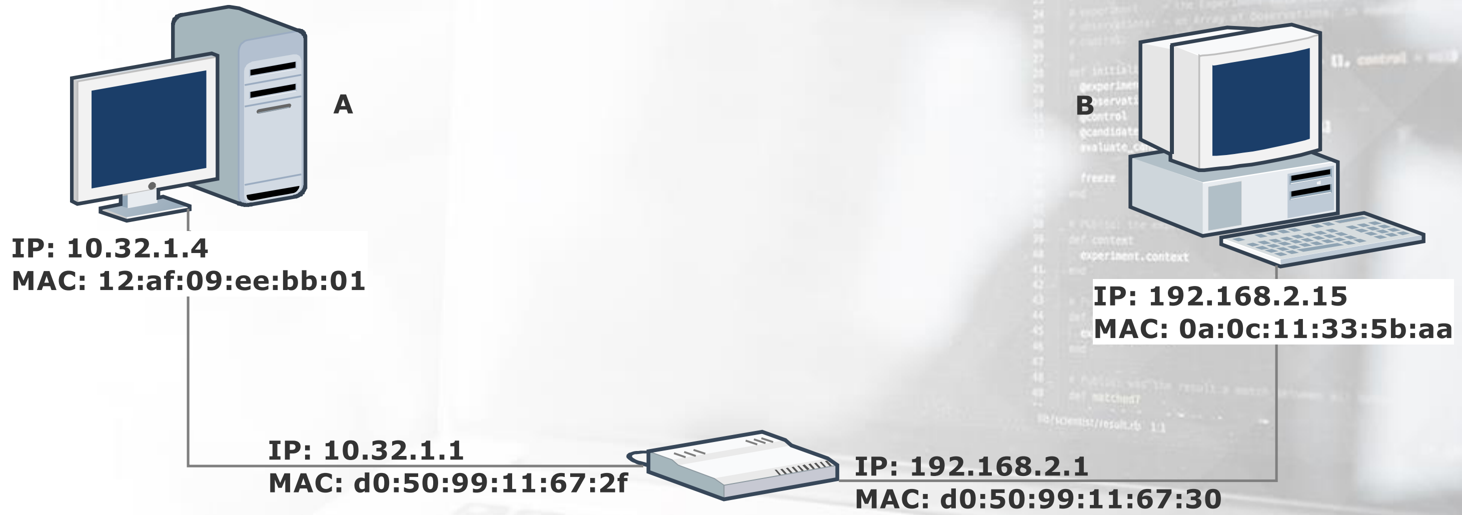
Two different networks are connected together by a router:

- 10.32.1.0/24
- 192.168.2.0/24



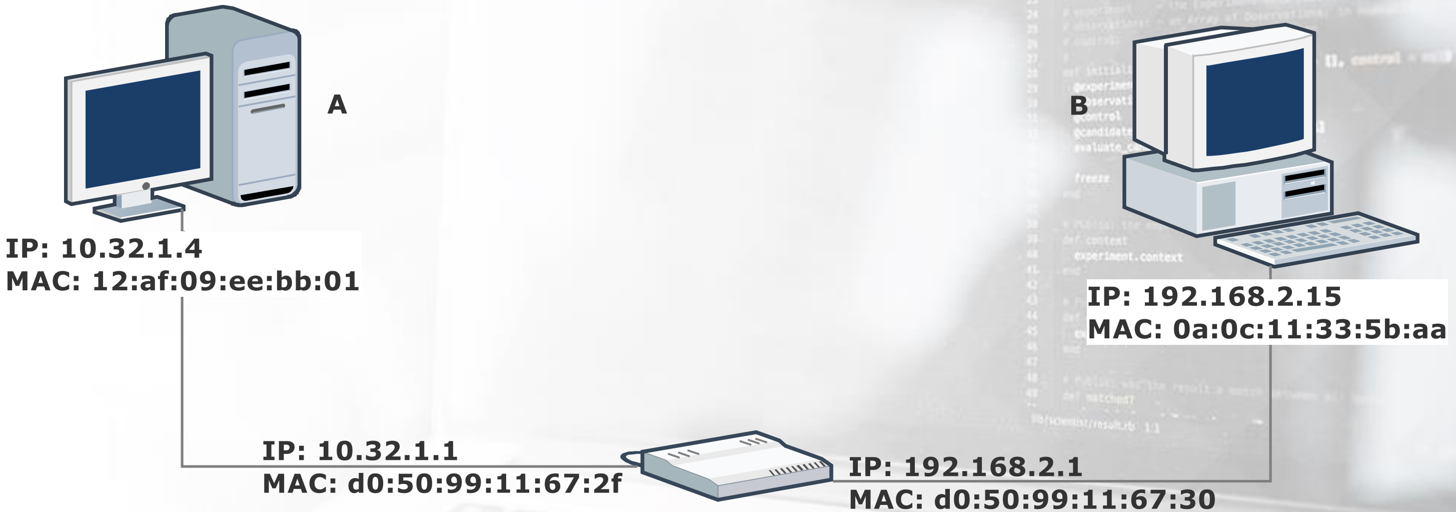
2.4.3 IP and MAC Addresses

Every host on the network has both an IP and a MAC address. The router has two interfaces, each with its own addresses.



2.4.3 IP and MAC Addresses

If workstation A wants to send a packet to workstation B, which IP and MAC addresses will it use?



2.4.3 IP and MAC Addresses

Workstation A will create a packet with:

- The **destination IP address of workstation B** in the IP header of the datagram.
- The **destination MAC address of the router** in the link layer header of the frame.
- The **source IP address of workstation A**
- The **source MAC address of workstation A**

2.4.3 IP and MAC Addresses

The router will then take the packet and forward it to B's network, **rewriting the packet's MAC addresses**:

- The **destination MAC address** will be B's
- The **source MAC address** will be the router's

The router will not change the source and destination IP addresses.

2.4.3 IP and MAC Addresses

When a device sends a packet:

- The destination MAC address is the MAC address of the **next hop**; this ensures that, locally, the network knows where to forward the packet.

- The destination IP address is the address of the **destination host**; this is global information and remains the same along the packet trip.

2.4.3 IP and MAC Addresses

This method, in a way, recalls how you send a letter to a friend.

You need to know his or her home address (IP address) and the address of the nearest post office (MAC address) where you can drop the letter.

2.4.4 Broadcast MAC Address

There is also a special MAC address

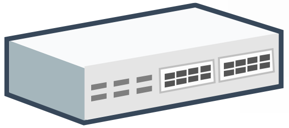
```
FF:FF:FF:FF:FF:FF
```

…which is the **broadcast** MAC address.

A frame (the name of the packets at Layer 2) with this address is delivered to all the hosts in the local network (within the same broadcast domain).

2.4.5 Switches

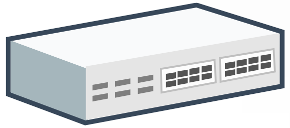
While routers work with IP addresses, switches work with MAC addresses. Switches also have multiple interfaces, so they need to keep a **forwarding table** that binds one or more MAC addresses to an interface.



MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5 Switches

The forwarding table is called Content Addressable Memory (CAM) table. Many hosts can connect to a switch. Let's see how.



MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5 Switches

The smallest switches you can encounter are home switches, usually integrated into a DSL home router. They usually have 4 ports.

Corporate switches may have up to 64 ports, and system administrators can connect multiple switches together to accommodate more hosts.

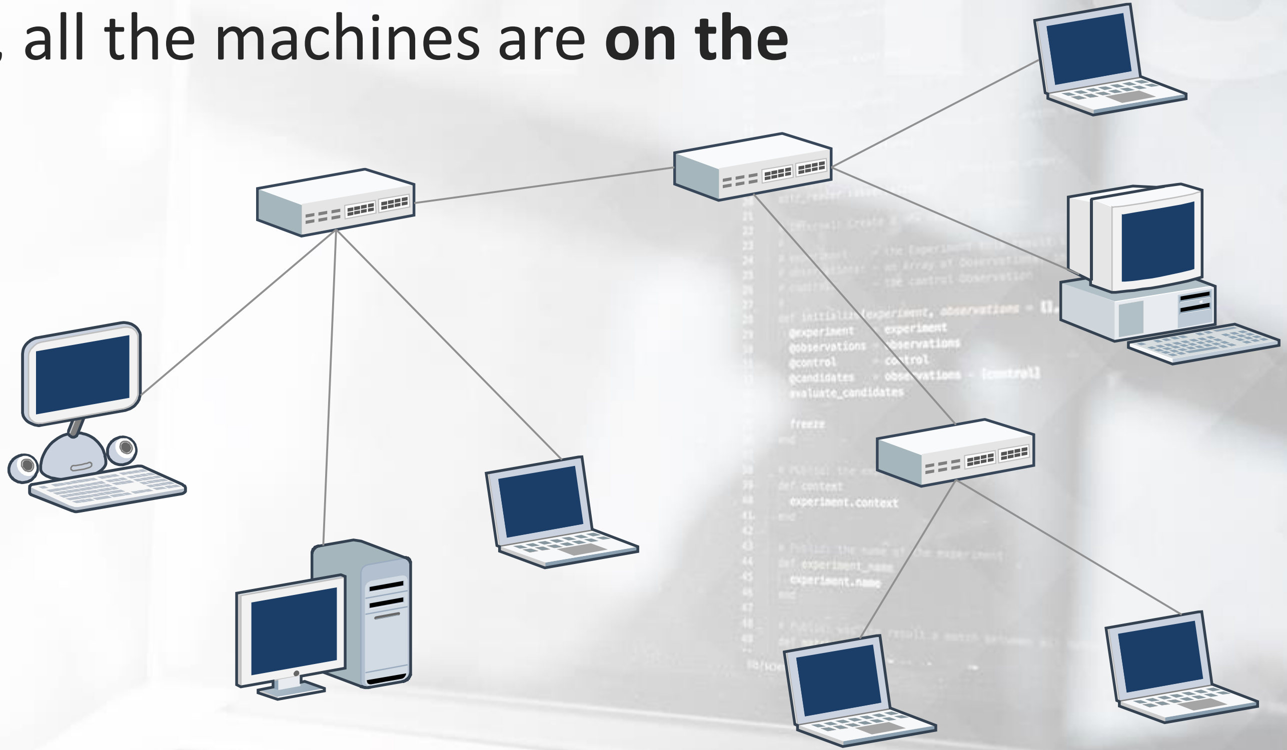
2.4.5 Switches

The main difference between one switch and another is the packet forwarding speed.

The speed of a switch varies from 10Mbps (megabits per second) to 10Gbps (gigabits per second). Nowadays, 1Gbps is the most common forwarding speed in commercial switches.

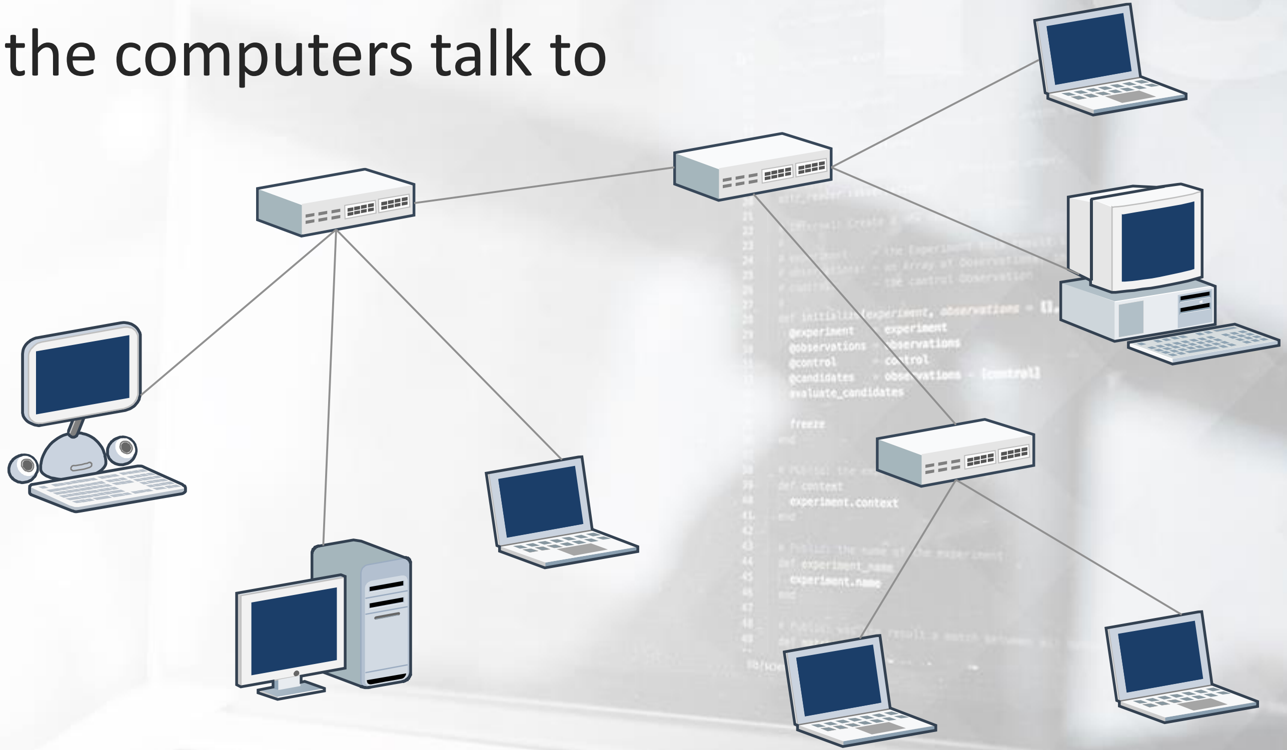
2.4.5.1 Multi-switch Network

In this diagram, all the machines are **on the same network**.



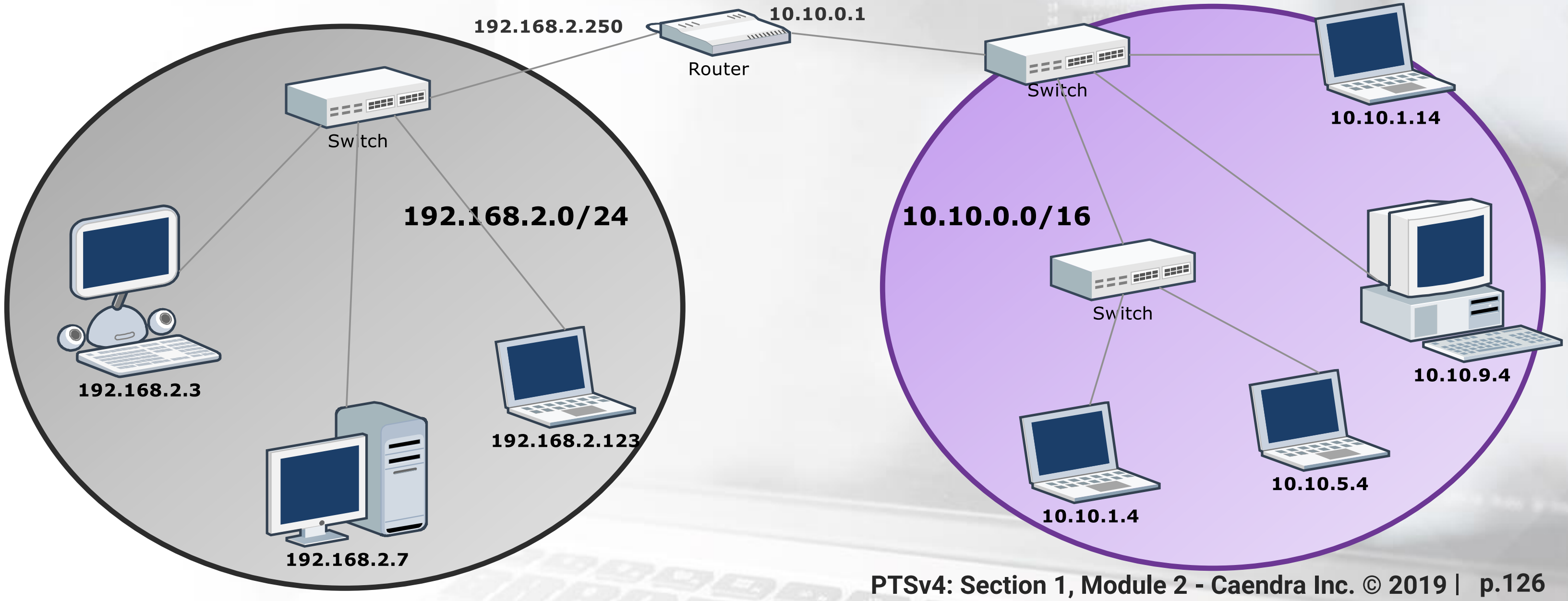
2.4.5.1 Multi-switch Network

Switches let all the computers talk to each other.



2.4.5.2 Segmentation

Switches, without VLANs, do not **segment** networks. Routers do.



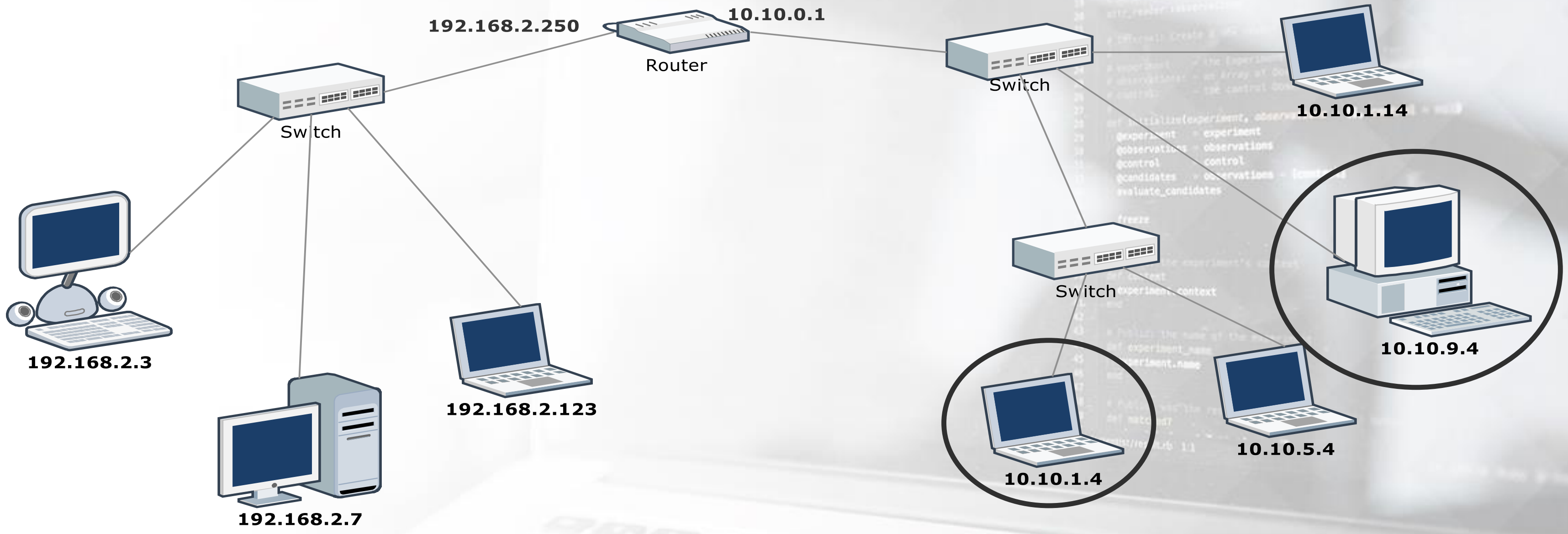
2.4.5.2 Segmentation

Usually, every interface of a router is attached to a different subnet with a different network address.

Also, routers do not forward packets coming from one interface if they have a ff:ff:ff:ff:ff:ff broadcast MAC address (imagine if they did!).

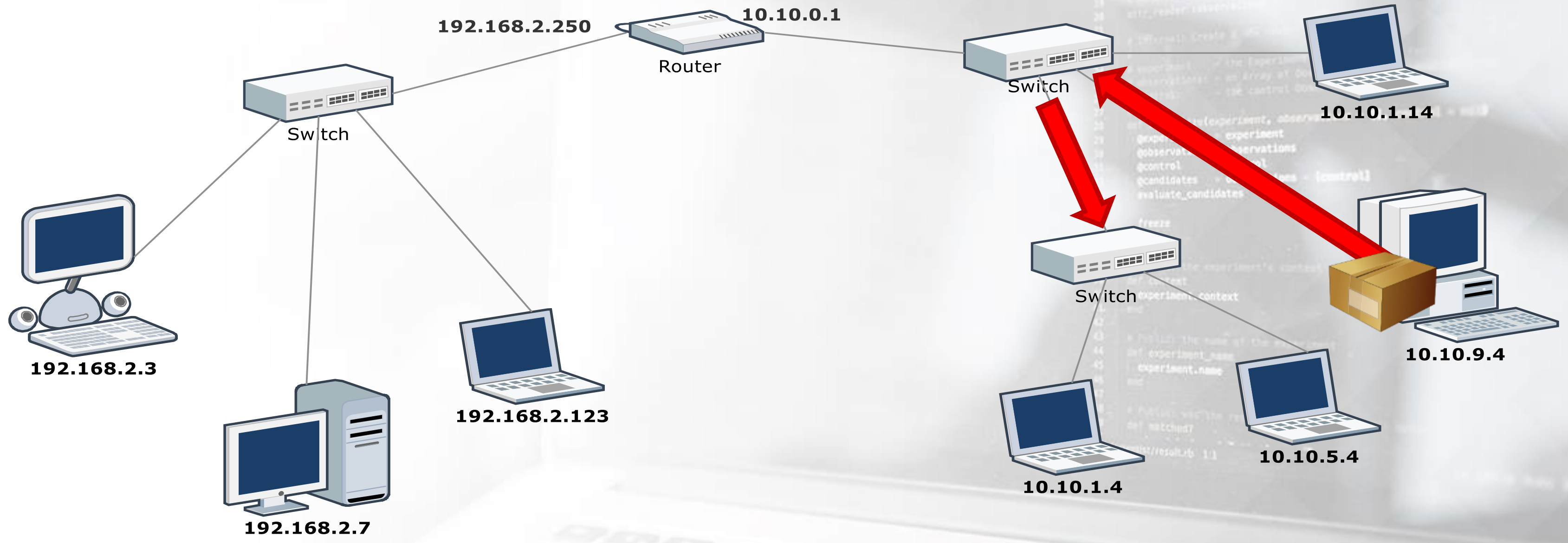
2.4.5.3 Multi-switch Example

What happens if 10.10.9.4 wants to send a packet to 10.10.1.4?



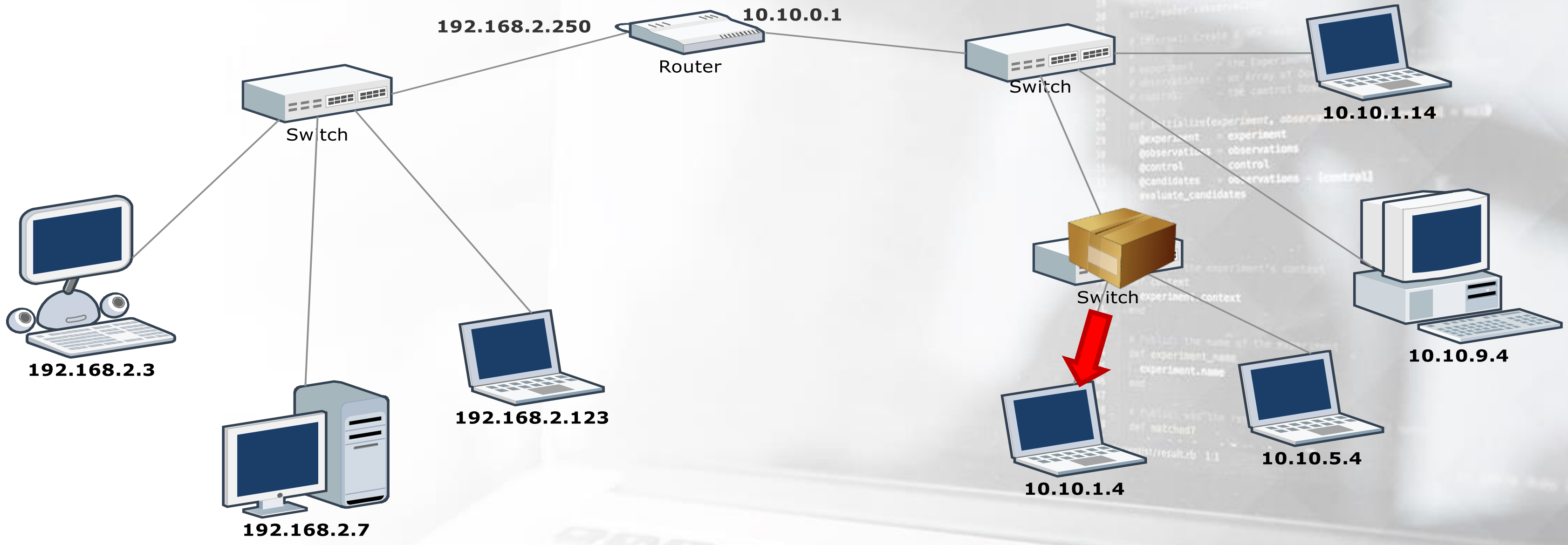
2.4.5.3 Multi-switch Example

The first switch receives the packet, performs a look-up in the CAM table and forwards it to the next switch.



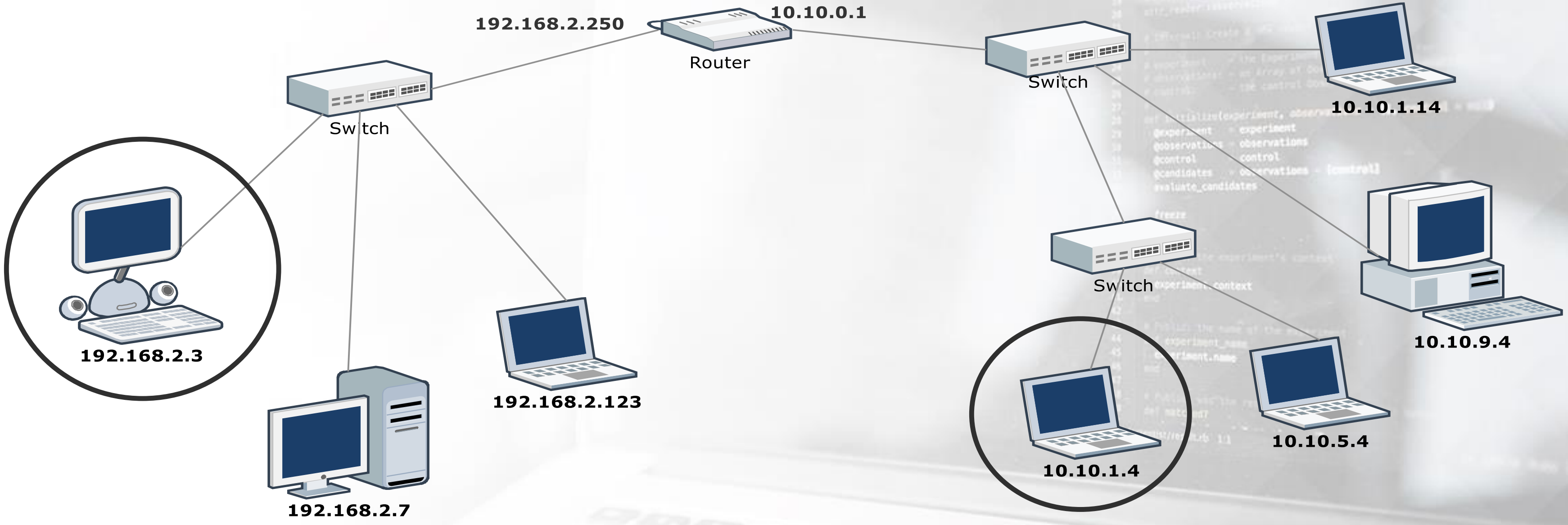
2.4.5.3 Multi-switch Example

The second switch forwards the packet to 10.10.1.4.



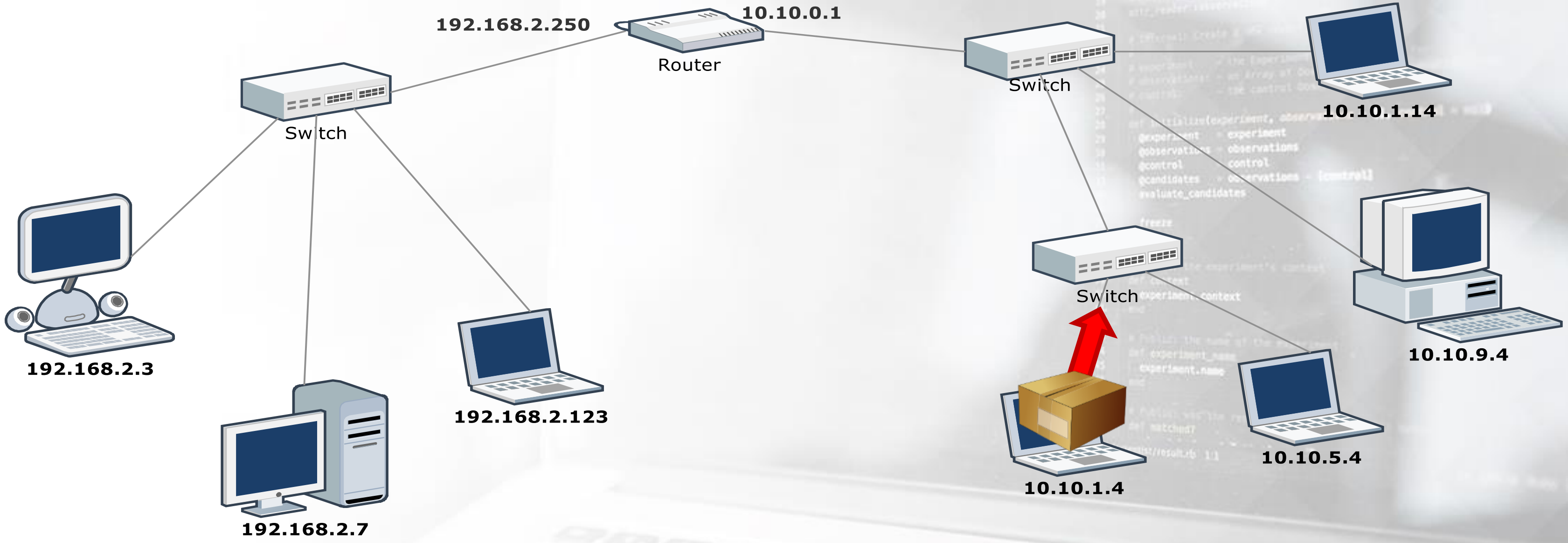
2.4.5.4 Multi-switch and Router Example

What happens if 10.10.1.4 wants to send a packet to 192.168.2.3?



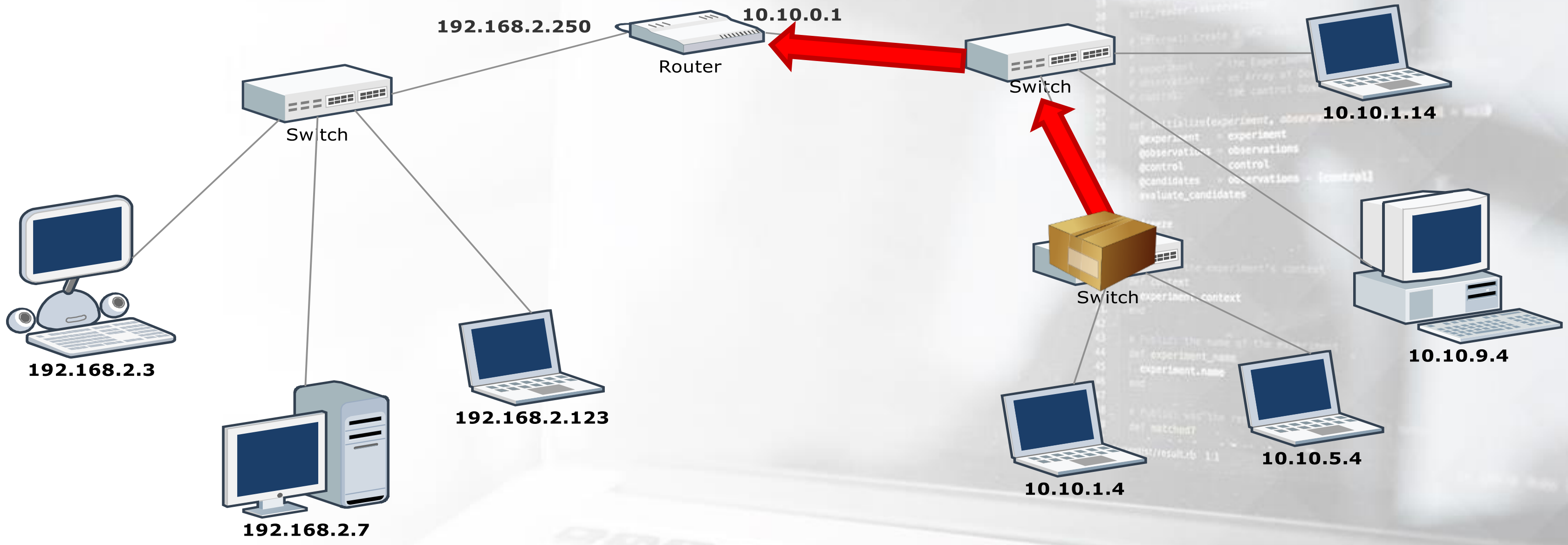
2.4.5.4 Multi-switch and Router Example

10.10.1.4 needs to send the packet to the router so that the first switch will forward the packet to the next one.



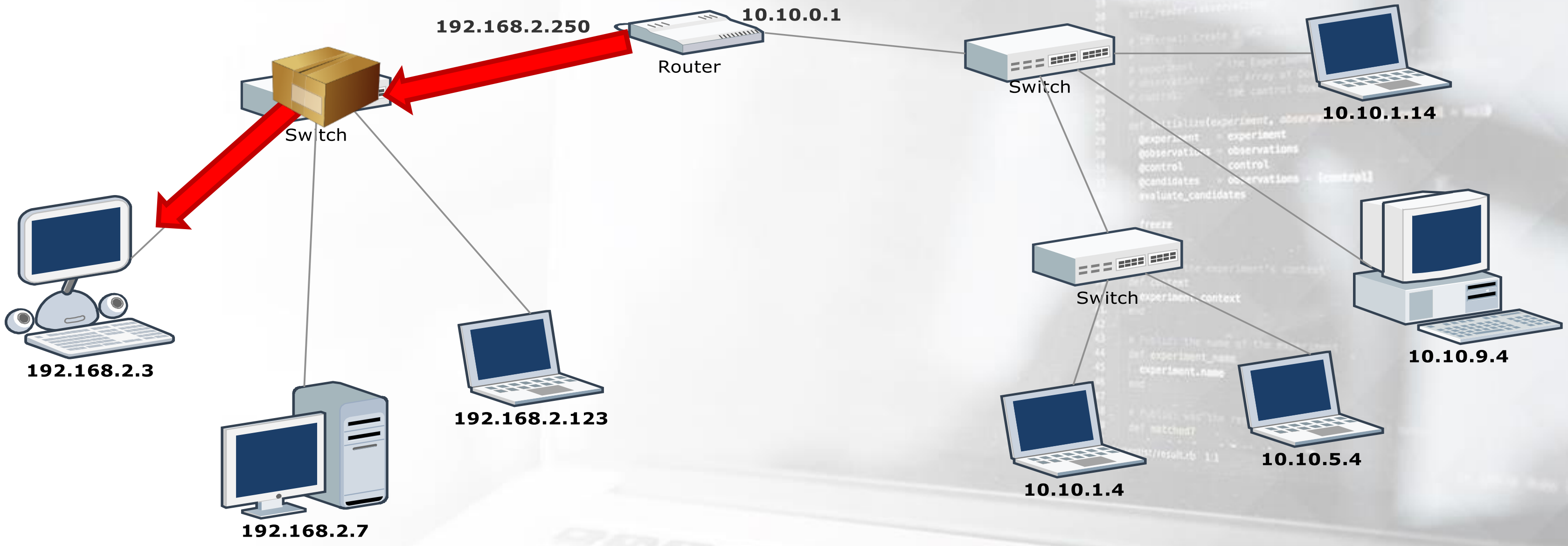
2.4.5.4 Multi-switch and Router Example

The packet then arrives at the router that, after a look up in the routing table, forwards it to the 192.168.2.0/24 network.



2.4.5.4 Multi-switch and Router Example

Finally, the packet is delivered.



2.4.5.5 Forwarding Tables

A forwarding table binds MAC addresses to interfaces.

In the following slides you will see:

- The structure of the table
- The way a switch constructs the MAC address – interface binding
- How forwarding works

2.4.5.5 Forwarding Tables

A typical forwarding table contains:

- The MAC address
- The interfaces the switch can use to deliver packets to a specific MAC address
- A time to live (TTL)

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5.5 Forwarding Tables

The forwarding table, or Content Addressable Memory table (CAM table), is stored in the device's RAM and is constantly refreshed with new information.

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5.5 Forwarding Tables

Looking at the table you can tell that:

- A single host is attached to Interface 1 and 3 respectively
- Two hosts are attached to interface 2 (probably via another switch).

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5.5 Forwarding Tables

There might be multiple hosts on the same interface and interfaces without any host attached.

In our example interface, 4 has no hosts attached.

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5.5 Forwarding Tables

The TTL determines how long an entry will stay in the table. This is important because the **CAM table has a finite size**.

So, as soon as an entry expires it is removed from the table.

MAC	Interface	TTL
00:11:22:33:44:55	1	30
AA:BB:CC:DD:EE:01	2	5
AA:CC:FF:0A:0C:12	2	5
11:22:33:1D:CC:0A	3	7

2.4.5.6 CAM Table Population

Switches learn new MAC addresses dynamically; they inspect the header of every packet they receive, thus identifying new hosts.

While routers use complex routing protocols to update their routing rules, switches just use the source MAC address of the packets they process to decide which interface to use when forwarding a packet.

2.4.5.6 CAM Table Population

The source MAC address is compared to the CAM table:

- If the MAC address is not in the table, the switch will add a new MAC-Interface binding to the table
- If the MAC-Interface binding is already in the table, its TTL gets updated
- If the MAC is in the table but bound to another interface the switch updates the table

2.4.5.7 Forwarding

To forward a packet:

1. The switch reads the destination MAC address of the frame.
2. It performs a look-up in the CAM table.
3. It forwards the packet to the corresponding interface.
4. If there is no entry with that MAC address, the switch will forward the frame to all its interfaces.

2.4.6 ARP

When a host wants to send a packet to another host, it needs to know the IP and the MAC address of the destination in order to build a proper packet.

You wouldn’t be able to send your friend a letter if you don’t know his/her address, right? What happens if the source host knows the IP address, but not the MAC address of the destination host?

2.4.6 ARP

This situation occurs in many circumstances, for example at every power up.

- A PC in an office knows a bunch of IP addresses, like the fileserver, the printers, and the webserver, but not their corresponding MAC addresses.

The host needs to know the MAC addresses of the other network nodes, and it can learn them by using the **Address Resolution Protocol (ARP)**.

2.4.6 ARP

With ARP a host can build the correct IP Address – MAC address binding.

This is one of the most fundamental protocols any modern network uses, so make sure to fully understand it.

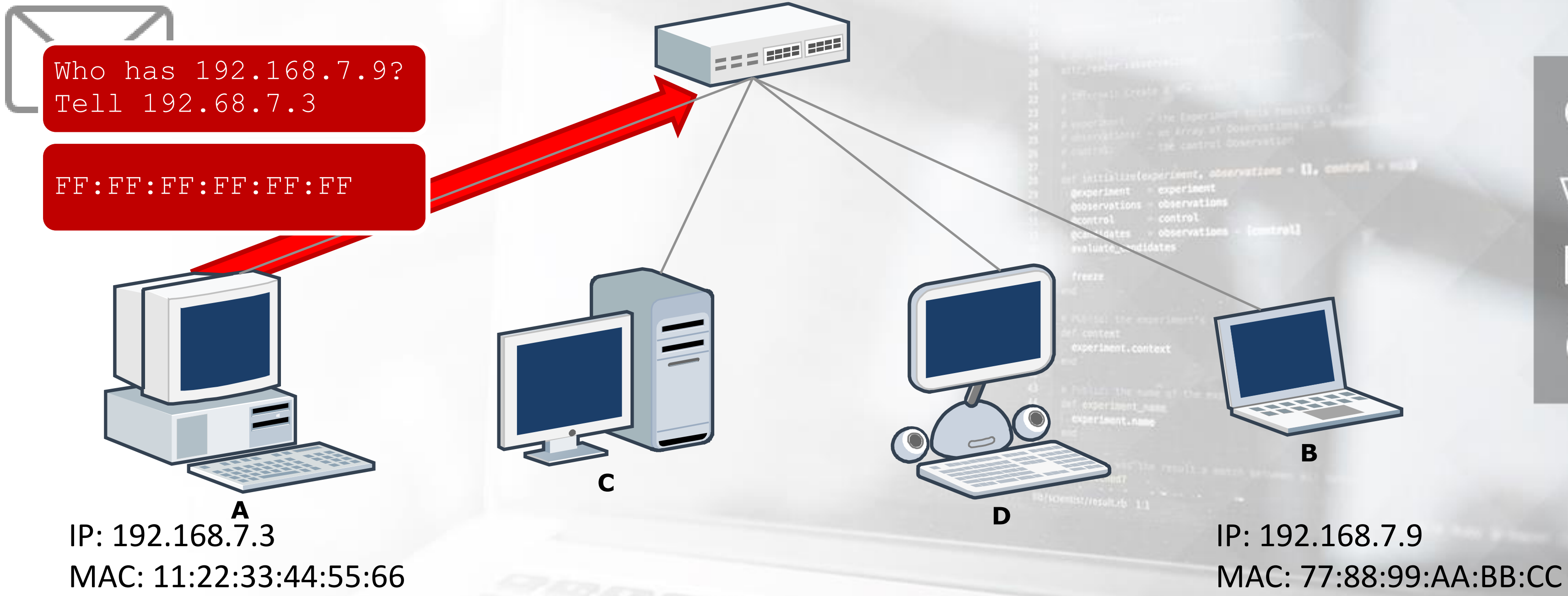
2.4.6 ARP

When a host (*A*) wants to send traffic to another (*B*), and it only knows the IP address of *B*:

1. *A* builds an **ARP request** containing the IP address of *B* and FF:FF:FF:FF:FF:FF as destination MAC address. This is fundamental because the switches will forward the packet to every host.
2. Every host on the network will receive the request.
3. B replies with an **ARP reply**, telling *A* its MAC address.

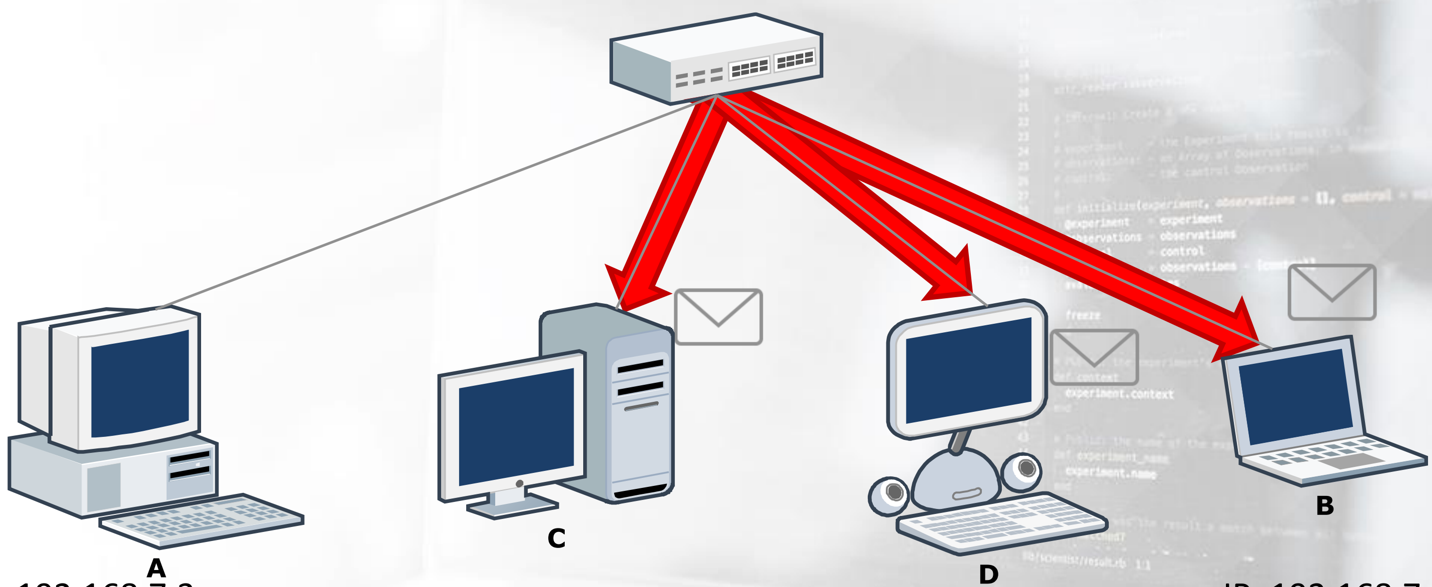
2.4.6 ARP

1 'A' sends a packet to the broadcast MAC address, asking for the MAC address of B.



2.4.6 ARP

2 The switch forwards the packet to all its ports.

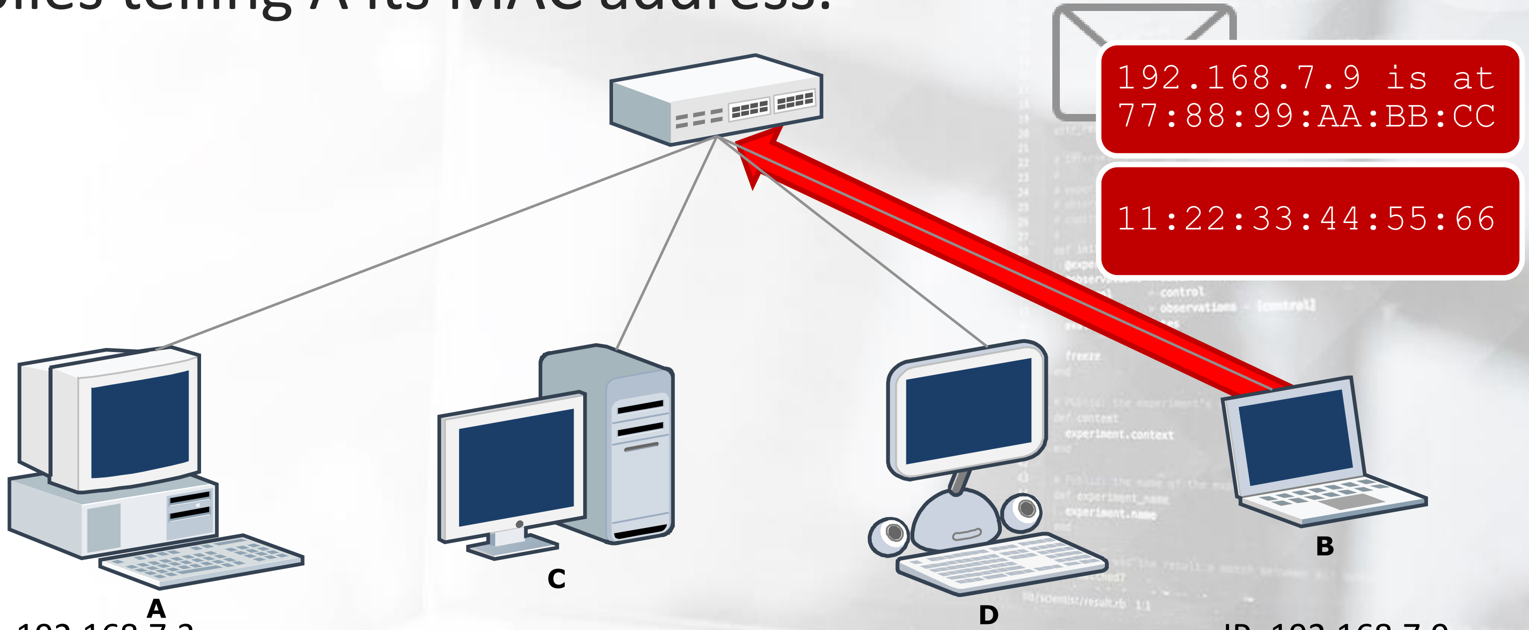


IP: 192.168.7.3
MAC: 11:22:33:44:55:66

IP: 192.168.7.9
MAC: 77:88:99:AA:BB:CC

2.4.6 ARP

3 B replies telling A its MAC address.

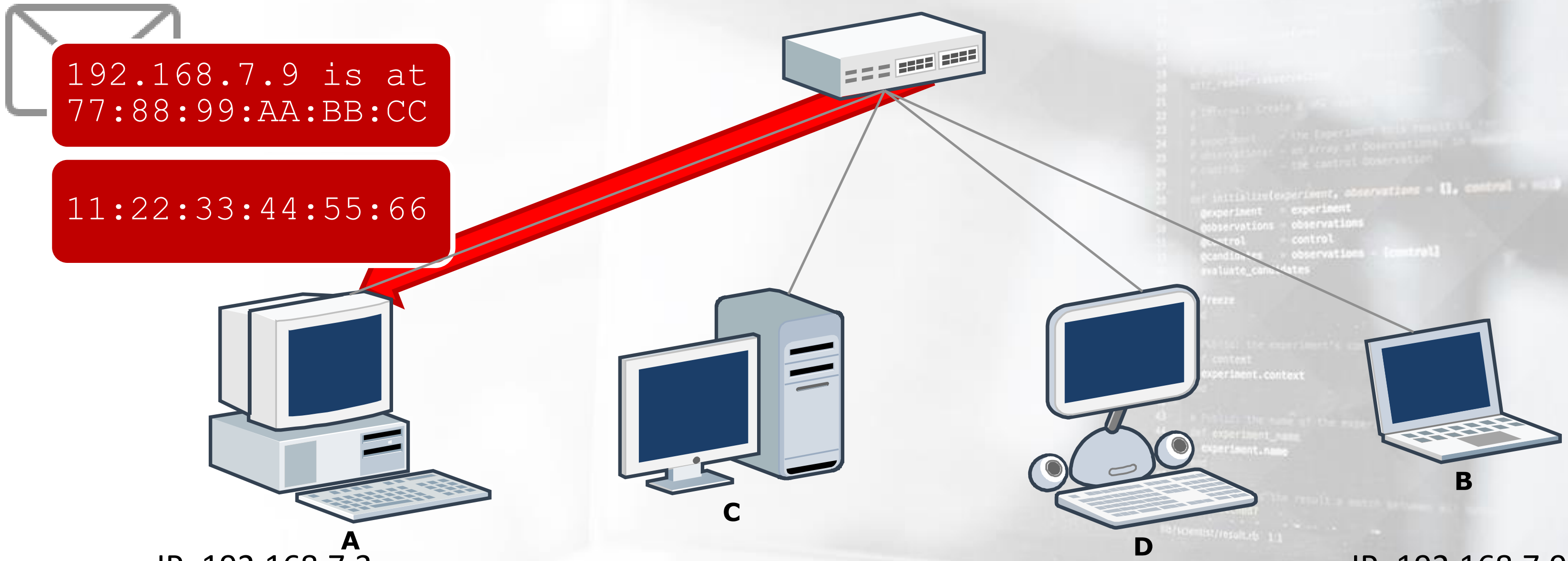


IP: 192.168.7.3
MAC: 11:22:33:44:55:66

IP: 192.168.7.9
MAC: 77:88:99:AA:BB:CC

2.4.6 ARP

Finally, the switch forwards the reply to A.



IP: 192.168.7.3
MAC: 11:22:33:44:55:66

IP: 192.168.7.9
MAC: 77:88:99:AA:BB:CC

2.4.6 ARP

'A' will save the IP – MAC binding in its ARP cache. Further traffic to *'B'* will not need a new ARP resolution protocol round.

ARP cache entries have a TTL too, as the size of the device RAM is finite. A host discards an entry at the power off or when the entry's TTL expires.

2.4.6.1 Checking the ARP Cache

You can check the ARP cache of your host by typing:

- `arp -a` on Windows.
- `arp` on *nix operating systems
- `ip neighbour` on Linux

```
$ ip neighbour
192.168.17.202 dev eth0 lladdr d0:d4:12:e1:ef:5a STALE
192.168.17.1 dev eth0 lladdr 00:50:7f:78:fc:40 STALE
192.168.17.99 dev eth0 lladdr 00:d0:4b:92:2d:89 STALE
192.168.17.14 dev eth0 lladdr 60:a4:4c:a8:be:5b STALE
192.168.17.18 dev eth0 lladdr 20:cf:30:c7:ad:ae STALE
192.168.17.30 dev eth0 lladdr 20:cf:30:ea:22:13 STALE
192.168.17.66 dev eth0 lladdr a4:ee:57:e8:2e:0b STALE
192.168.17.254 dev eth0 lladdr c8:4c:75:a4:79:a6 REACHABLE
192.168.17.12 dev eth0 lladdr 60:a4:4c:a8:bd:1a STALE
192.168.17.19 dev eth0 lladdr 54:04:a6:a0:6e:ad STALE
192.168.17.24 dev eth0 lladdr bc:5f:f4:ef:63:51 STALE
```

2.4.7 Hubs

Hubs were used in computer networks before switches. They have the same **purpose** but not the same **functionality**.

Hubs are simple repeaters that do not perform any kind of header check and simply forward packets by just repeating electric signals. They receive electric signals on a port and repeat the same signals on all the other ports.

2.4.7 Hubs

This means that every node on a hub-based network receives the same electric signals, thus the same packets.

Nowadays, hubs are very rare as they have mostly been replaced by switches.

TCP & UDP

2.5 TCP and UDP

How does this support my pentesting career?

- TCP Session Attacks
- Advanced DoS attacks
- Network scanning

2.5 TCP and UDP

In this section, you will see how the **transport layer** works, and how the application layer uses its services to identify server and client processes.

The **Transmission Control Protocol** (TCP) and the **User Datagram Protocol** (UDP) are the most common transport protocols used on the Internet.

2.5 TCP and UDP

Before checking out the different services that a transport layer protocol can offer to the application layer, let’s consider something important about networks.

Computer networks can be **unreliable**. This means that some **packets can be lost** during their trip from source to destination. A packet can be lost because of network congestion, temporary loss of connection and other technical issues.

2.5 TCP and UDP

When designing a transport layer protocol, the designer must choose how to deal with these limitations. For example, TCP:

- **Guarantees packet delivery**. Because of that, an application that needs a guaranteed delivery will use TCP as the transport protocol.
- Is also **connection oriented**. It must establish a connection before transferring data.

Keep in mind these facts during your study!

2.5 TCP and UDP

TCP is the most used transport protocol on the Internet. The vast majority of applications use it, and the IP protocol suite is often called **TCP/IP**.

Email clients, web browsers and FTP clients are some common applications using TCP.

2.5 TCP and UDP

On the other hand, UDP is much more simple than TCP:

- It does **not guarantee** packet delivery.
- It is **connectionless**.

2.5 TCP and UDP

UDP is faster than TCP, as it provides a **better throughput** (number of packets per second); in fact, it is used by **multimedia applications** that can tolerate packet loss but are throughput intensive.

For example, UDP is used for VoIP and video streaming: applications where you can tolerate a little glitch in the audio or video.

2.5 TCP and UDP

Here we can see a comparison table between TCP and UDP.

TCP	UDP
Lower throughput	Better throughput
Connection-oriented	Connectionless
Guarantees delivery	Does not guarantee packet delivery

2.5.1 Ports

Applications and their processes use TCP and UDP to send and receive data over the network. When an IP datagram reaches a host, how can the transport layer **know what the destination process is**?

We'll now introduce **ports**.

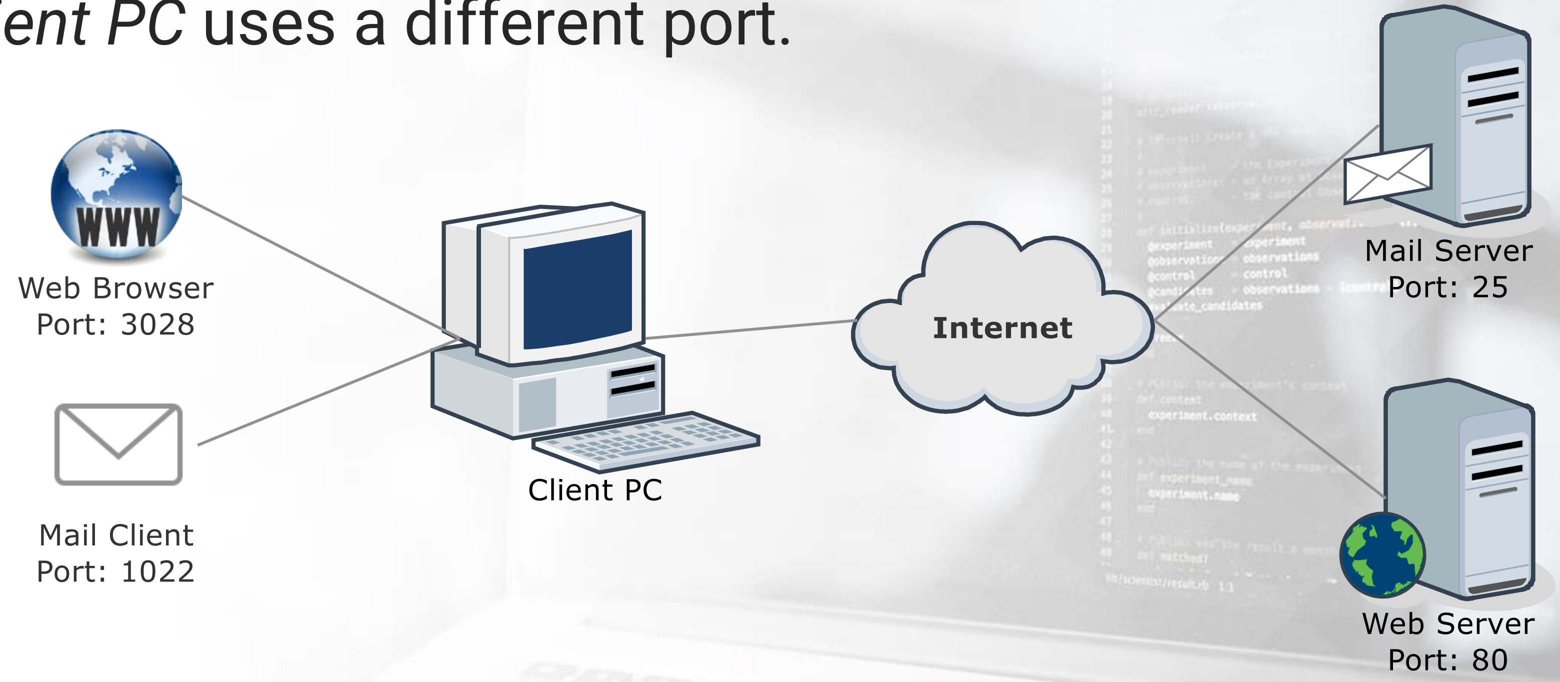
2.5.1 Ports

Ports are used to identify a single network **process** on a machine. If you want to unequivocally identify a process on a network, you need to know the **`<IP>:<Port>` pair**.

As an example, you can compare the port to the recipient's name on a letter; the street address (IP) identifies the building, while the person name identifies the final recipient of the letter.

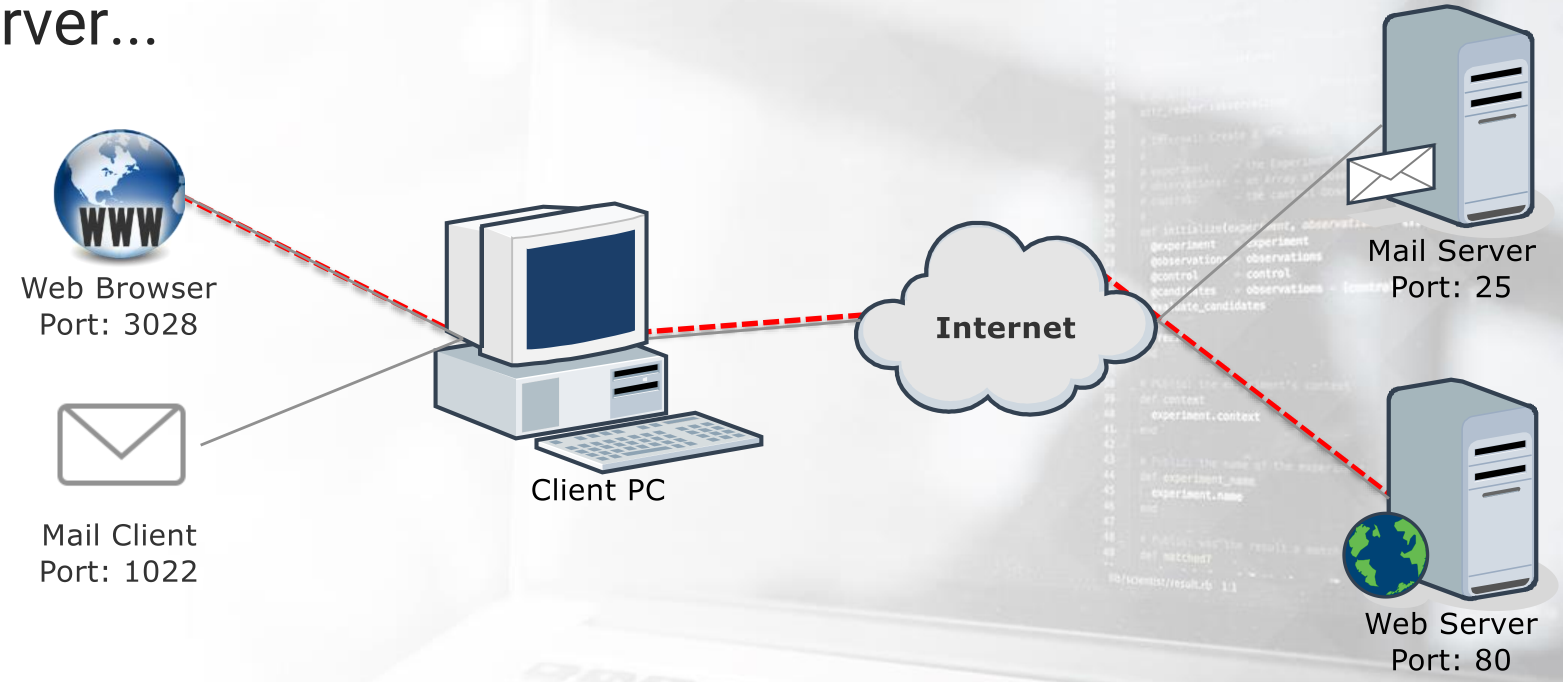
2.5.1.1 Ports Examples

In this image, you can see how every client application on *Client PC* uses a different port.



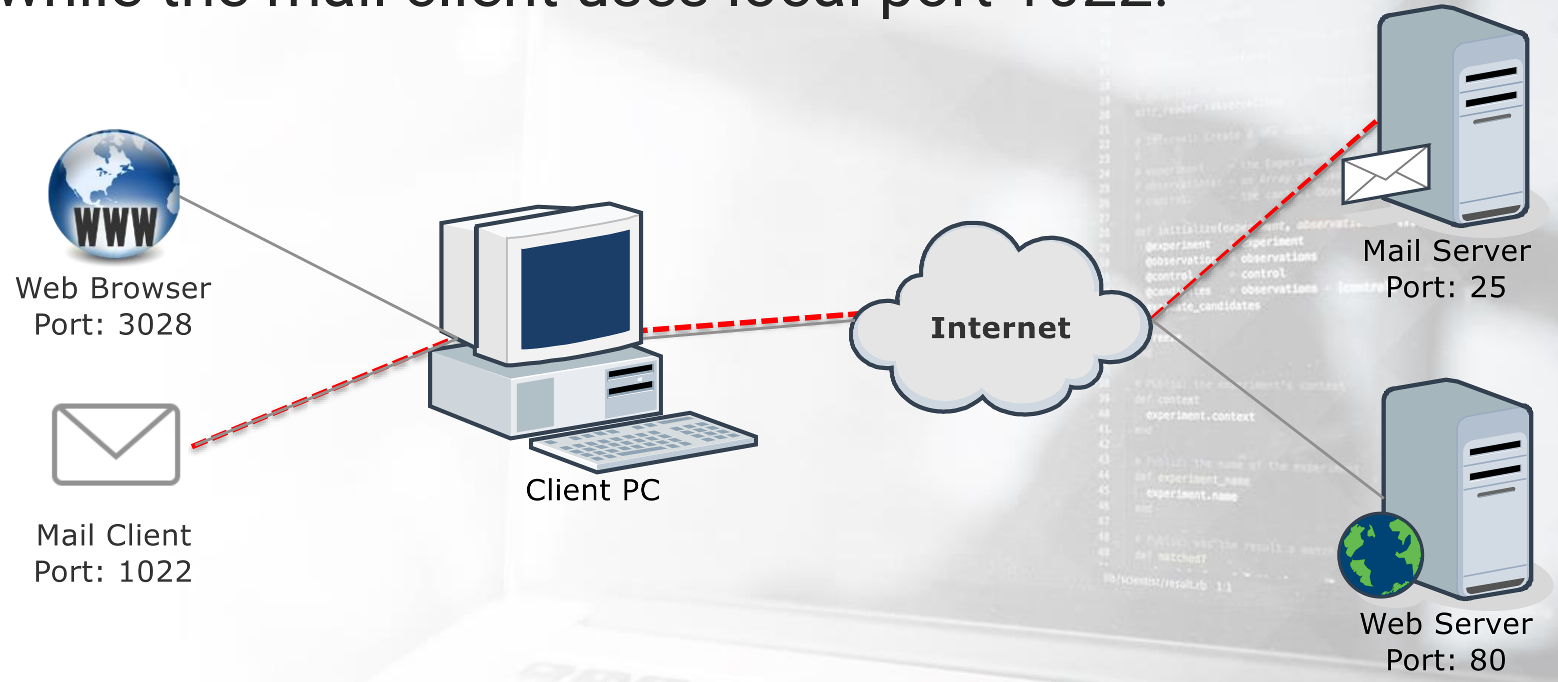
2.5.1.1 Ports Examples

The browser uses local port 3028 to connect to the web server...



2.5.1.1 Ports Examples

... while the mail client uses local port 1022.



2.5.1.1 Ports Examples

In the previous example:

- All the communication from the web browser to the web server will have 3028 as the source port and 80 as the destination port.
- All the communication back from the web server to the browser will have 80 as the source port and 3028 as the destination port.

2.5.1.1 Ports Examples

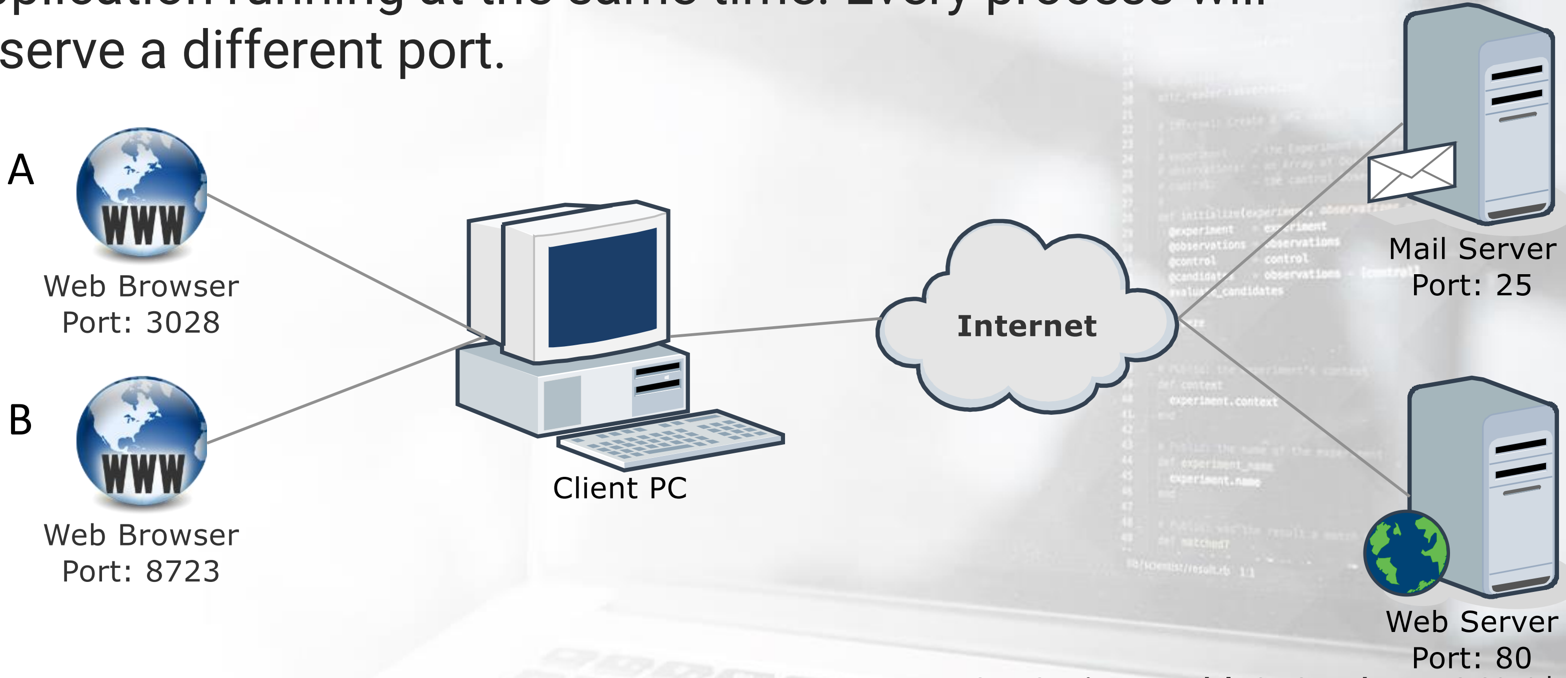
Similarly, for the mail client and server:

- All the communication from the mail client to the server will have 1022 as the source port and 25 as the destination port.

- All the communication back from the mail server to the mail client will have 25 as the source port and 1022 as the destination port.

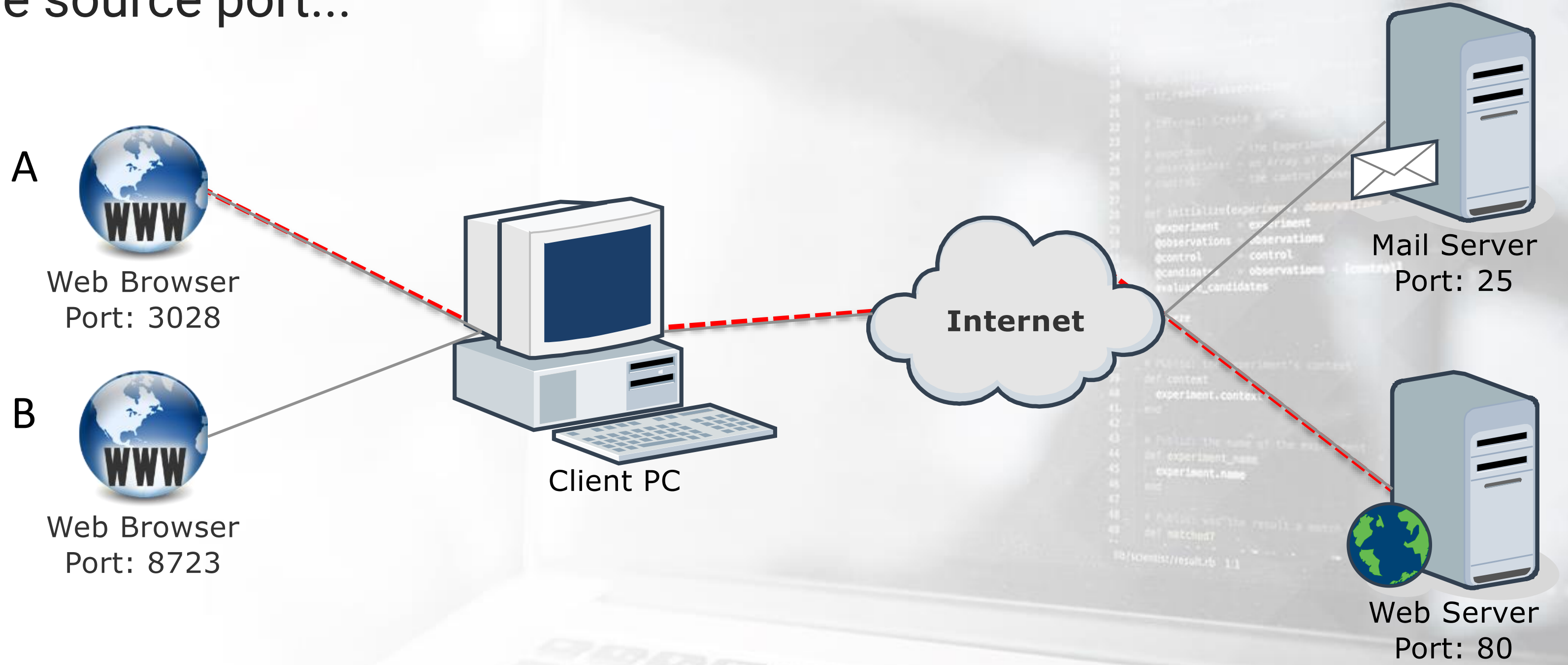
2.5.1.1 Ports Examples

Furthermore, you may also have multiple instances of the same application running at the same time. Every process will reserve a different port.



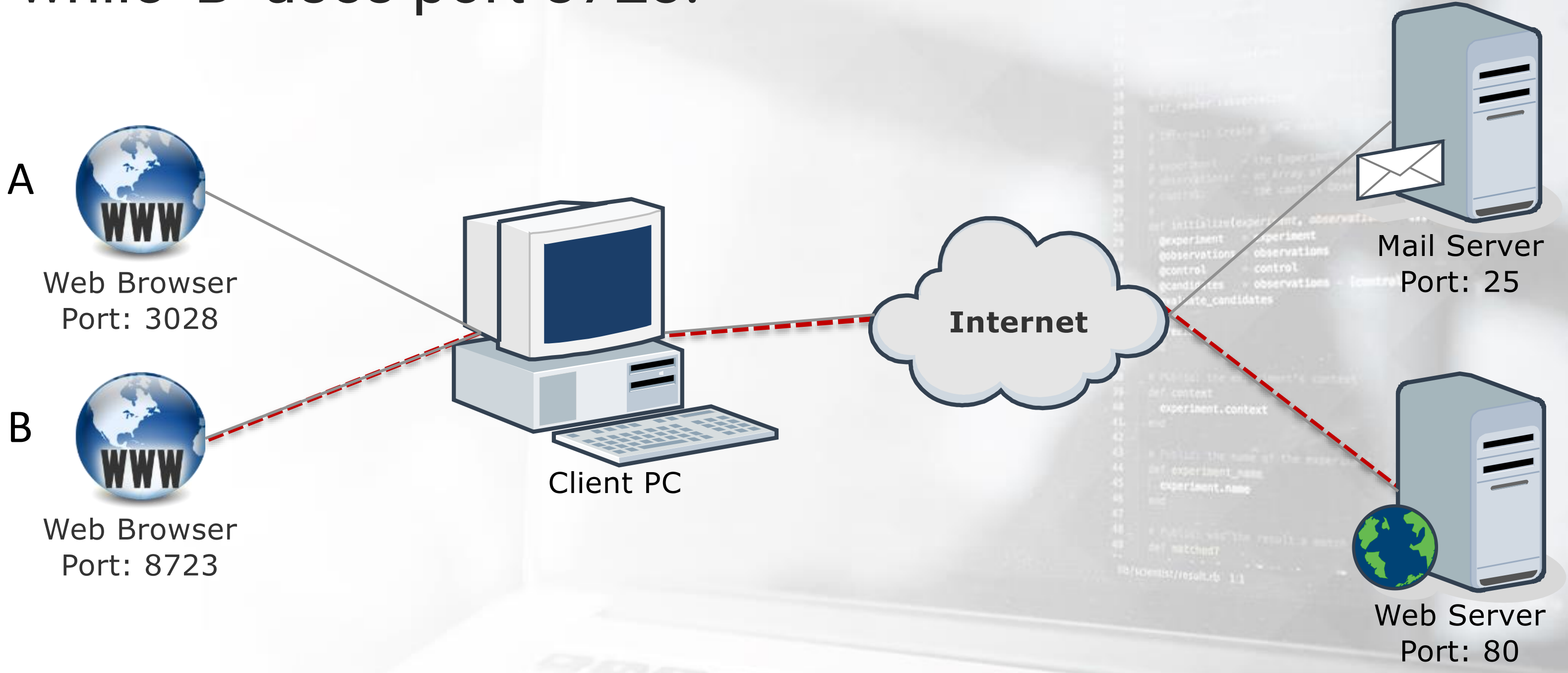
2.5.1.1 Ports Examples

In this example, ‘A’ communicates with the web server using 3028 as the source port...



2.5.1.1 Ports Examples

... while 'B' uses port 8723.



2.5.1 Ports

To correctly address a process on a network, you have to refer to the `<IP>:<Port>` pair. For example:

- `192.168.5.3:80`
- `10.11.12.1:443`
- `172.16.8.9:22`

But, how can you know the right port for a common service?

2.5.2 Well-known Ports

Ports in the ranging from **0-1023**, the first 1024 that is, are called **well-known ports** and are used by servers for the most common services.

For example, when a web browser connects to a server via HTTPS, the user does not have to manually specify 443 as the destination port.

2.5.2 Well-known Ports

Each common protocol has a well-known port in the 0-1023 range. Common server processes, or daemons, use well-known ports most of the time.

Ports are assigned by IANA and are referenced in this document.

2.5.2 Well-known Ports

You do not need to know all the service port assignments, but you should at least remember the most common, such as:

- SMTP (25)
- SSH (22)
- POP3 (110)
- IMAP (143)
- HTTP (80)
- HTTPS (443)
- NETBIOS (137, 138, 139)
- SFTP (115)
- Telnet (23)
- FTP (21)
- RDP (3389)
- MySQL (3306)
- MS SQL Server (1433)

2.5.2 Well-known Ports

As briefly introduced before, a **daemon** is a program that runs a service. System administrators can change the daemon configuration, **changing the port** the service listens to for connection. They do that to make services recognition a little bit harder for hackers.

For example, you could find an FTP daemon listening on port 4982 instead of 21 or SSH listening on port 8821.

2.5.3 TCP and UDP headers

Let’s now see how ports are used by applications.

How can server and client applications know which port to use? They use two fields in the TCP or UDP header: the **source** and **destination** ports.

2.5.3.1 TCP Header

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |          Source Port          |       Destination Port        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                         Sequence Number                         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                     Acknowledgment Number                       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |  Data |           |U|A|P|R|S|F|                               |
 | Offset| Reserved  |R|C|S|S|Y|I|            Window             |
 |       |           |G|K|H|T|N|N|                               |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           Checksum            |         Urgent Pointer        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                    Options                    |    Padding    |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                             data                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The source and destination port are included in the transport layer protocol header. Like the TCP header...

2.5.3.1 TCP Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.5.3.2 UDP Header

…or the UDP header.

```
0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.5.3.2 UDP Header

```
 0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Length             |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.5.4 Netstat Command

To check the listening ports and the current (TCP) connections on a host you can use:

- `netstat -ano` on Windows
- `netstat -tunp` on Linux
- `netstat -p tcp -p udp` together with `lsof -n -i4TCP -i4UDP` on MacOS

Use these commands to show information about the processes listening on the machine and processes connecting to remote servers.

2.5.4 Netstat Command

Another great tool for Windows is TCPView from Sysinternals.

TCPView shows:

- Process name
- PID
- Protocol
- Local and remote addresses
- Local and remote ports
- State of the connection (if applicable)

http://technet.microsoft.com/en-us/sysinternals/bb897437

2.5.5 TCP Three Way Handshake

We have seen that TCP is **connection oriented**. Now, let's look at how TCP connections work, as well as highlight the most important factors involved, from the penetration tester's point of view, in a 3-way handshake.

To establish a connection between two hosts running TCP, they must perform three steps: the **three-way handshake**. They can then start the actual data transmission.

2.5.5 TCP Three Way Handshake

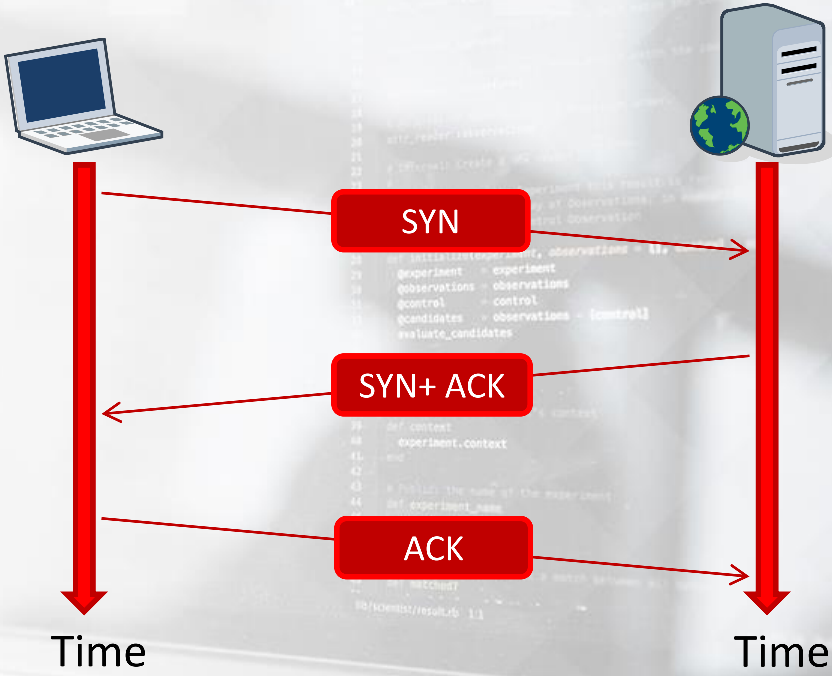
The header fields involved in the handshake are:

- Sequence number
- Acknowledgement numbers
- SYN and ACK flags

```
0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Sequence Number          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Acknowledgment Number       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|
| Offset| Reserved  |R|C|S|S|Y|I|
|       |           |G|K|H|T|N|N|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                 |
```

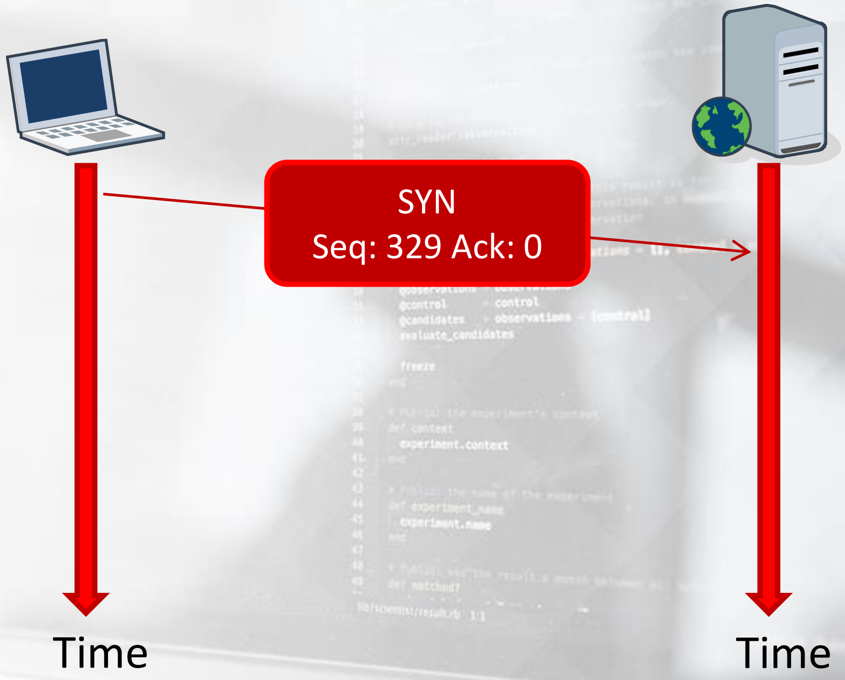
2.5.5 TCP Three Way Handshake

The steps in the handshake are used to synchronize the sequence and acknowledgment numbers between the server and the client.



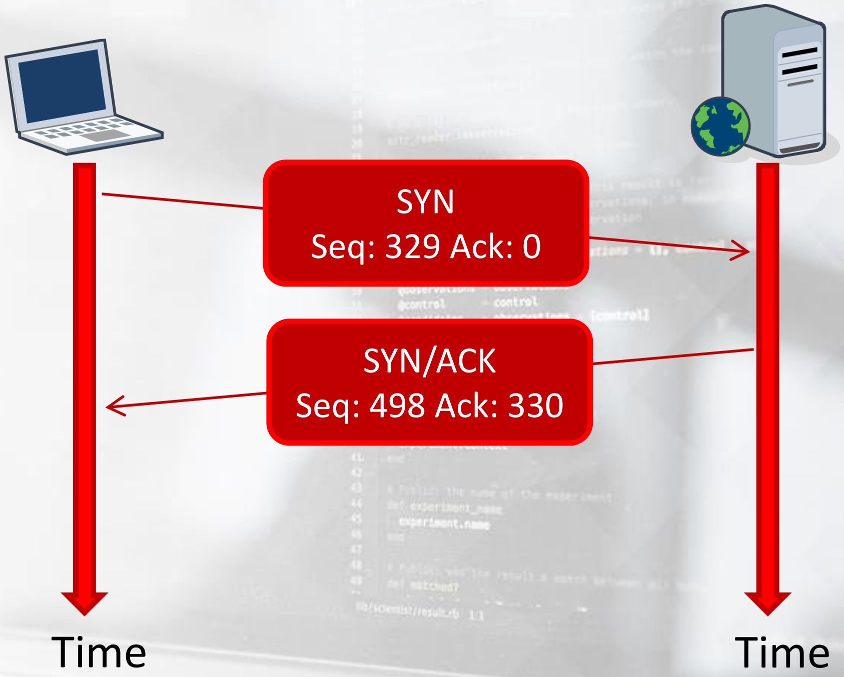
2.5.5 TCP Three Way Handshake

During the first step, the client sends a TCP packet to the server with the SYN flag enabled and a random sequence number.



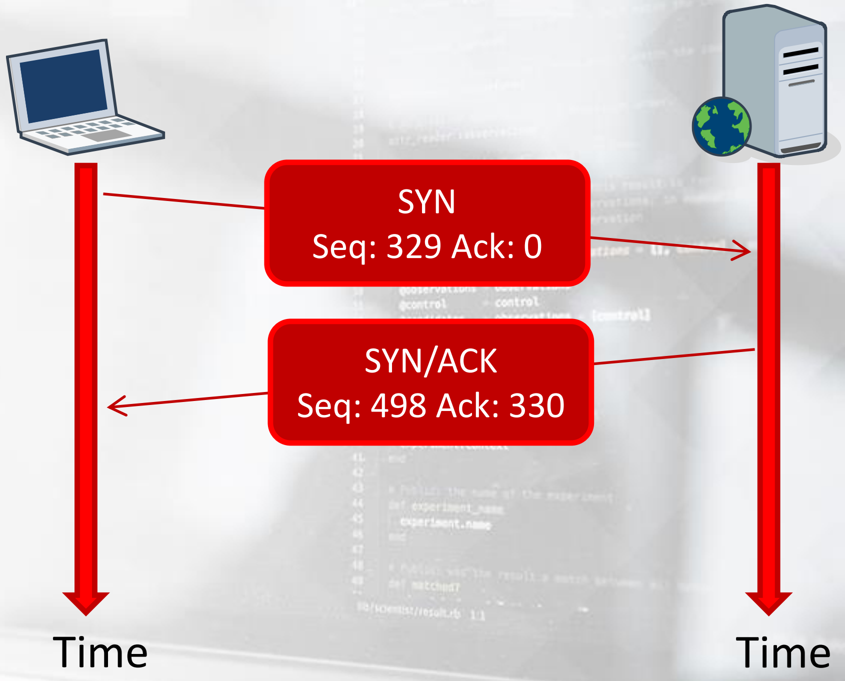
2.5.5 TCP Three Way Handshake

In the second step, the server replies by sending a packet with both the SYN and ACK flag set and another random sequence number.



2.5.5 TCP Three Way Handshake

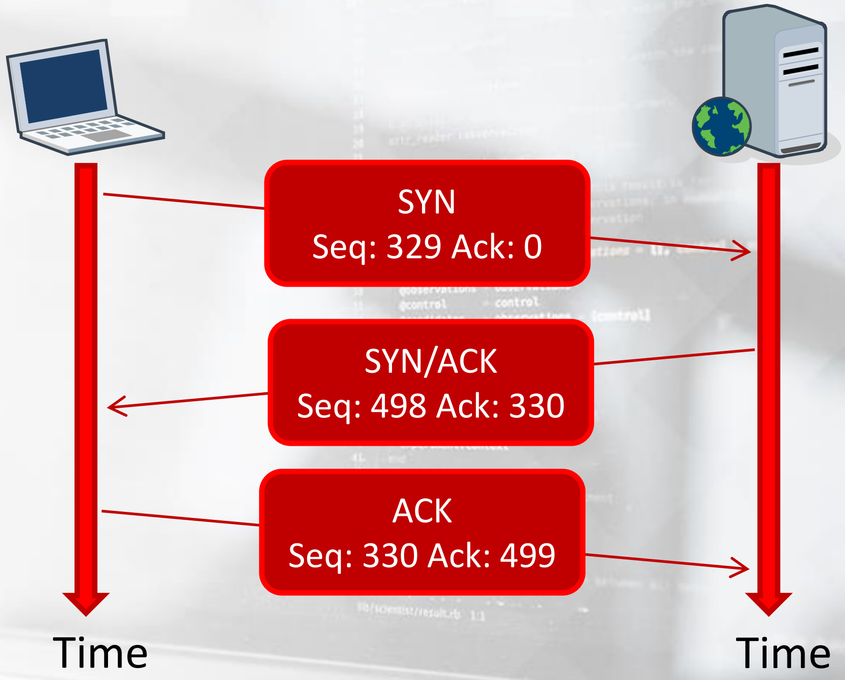
The ACK number is always a simple increment of the SYN number sent by the client.



2.5.5 TCP Three Way Handshake

Finally, the client completes the synchronization by sending an ACK packet.

Note that the client behaves just like the server when sending ACK packets.



2.5.6 References

For additional information, please check out the links below:

- IP Layer Network Administration with Linux.
- TCP/IP Tutorial and Technical Overview.
- Packet Analysis Reference Guide v3.0.

To access the above attachments, go to the course in your members area and click the resources drop-down in the appropriate module line.

Firewalls and Network Defense

2.6 Firewalls and Network Defense

There are many different appliances on the market that a system administrator can use to protect the network.

These devices use **different techniques** and work on **different layers** to perform **access control, attack detection and prevention**.

2.6 Firewalls and Network Defense

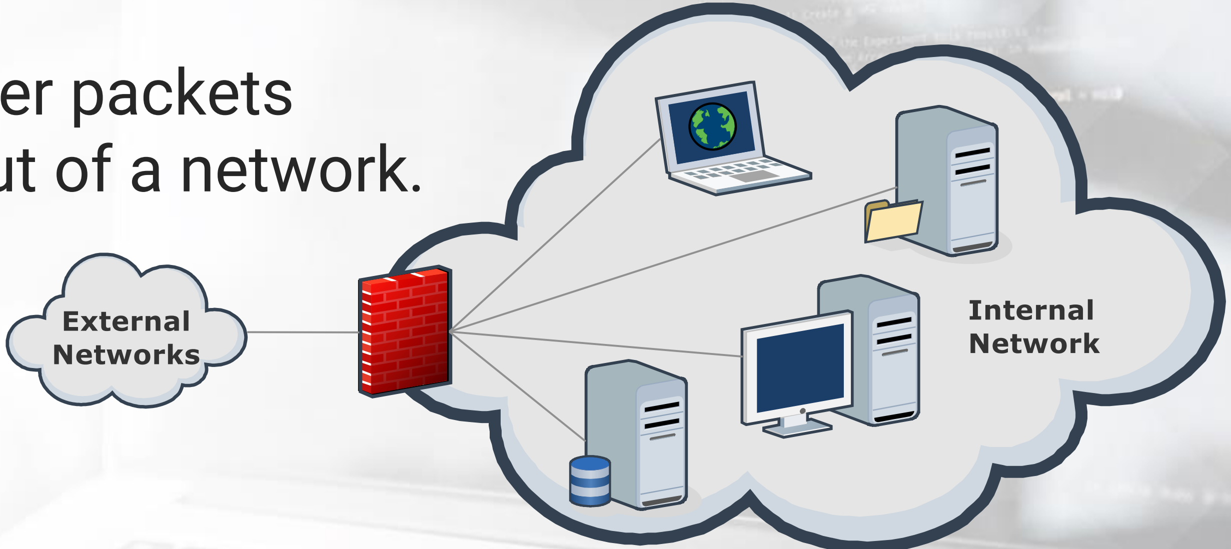
How does this support my pentesting career?

- Evading firewalls
- Advanced stealth scanning
- Filtering evasion

2.6.1 Firewalls

Firewalls are specialized software modules running on a computer or a dedicated network device.

They serve to filter packets coming in and out of a network.



2.6.1 Firewalls

Firewalls help system administrators and desktop users to control the access to network resources and services.

A firewall can **work on different layers** of the OSI model, thus providing different features and protections.

2.6.1 Firewalls

It is imperative that you understand how firewalls work and what kind of threats they prevent.

Many people believe that firewalls and antiviruses are all they need to stay secure, in the following slides you will see why this idea is wrong.

2.6.2 Packet Filtering Firewalls

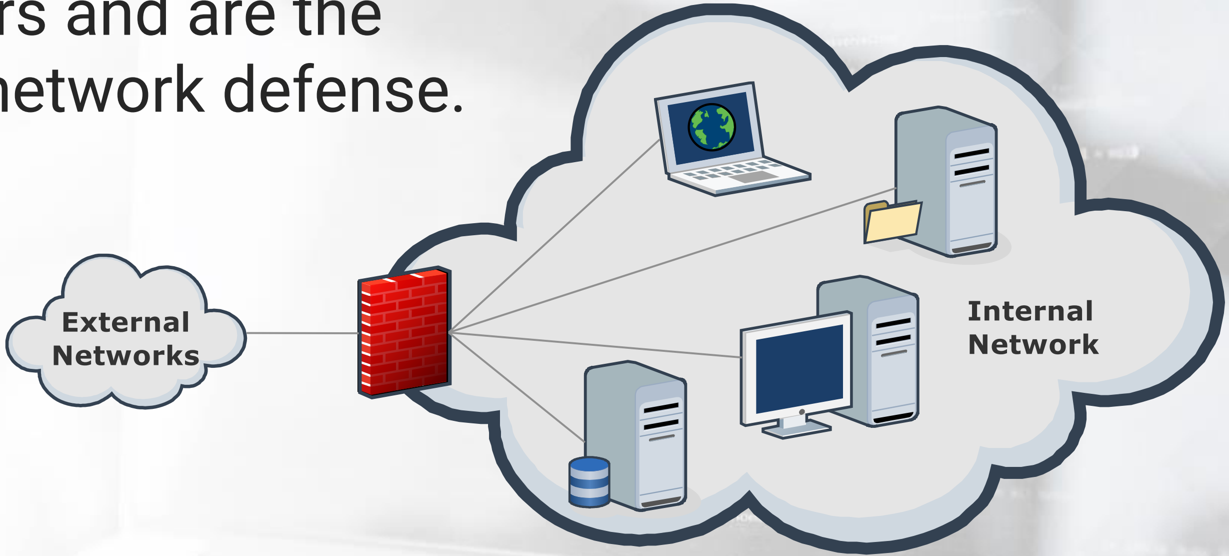
The most basic feature of a firewall is **packet filtering**.

With packet filtering, an administrator can create rules which will filter packets according to certain characteristics like:

- Source IP address
- Destination IP address
- Protocol
- Source port
- Destination Port

2.6.2 Packet Filtering Firewalls

Packet filters run on home DSL routers as well as high-end enterprise routers and are the cornerstone of network defense.



2.6.2 Packet Filtering Firewalls

Packet filters inspect the header of every packet to choose how to treat the packet. The more common actions are:

- **Allow**: allow the packet to pass
- **Drop**: drops the packet without any diagnostic message to the packet source host
- **Deny**: do not let the packet pass, but notify the source host

EXAMPLE

2.6.2 Packet Filtering Firewalls

Inspecting the header of a packet does not give you any information on the **actual packet content**.

In a company, network administrators configure a corporate firewall to allow web browsing from the internal network. They do that by allowing TCP traffic to have 80 or 443 as destination ports. What happens if an internal machine tries to connect via SSH to a server listening on port 80?

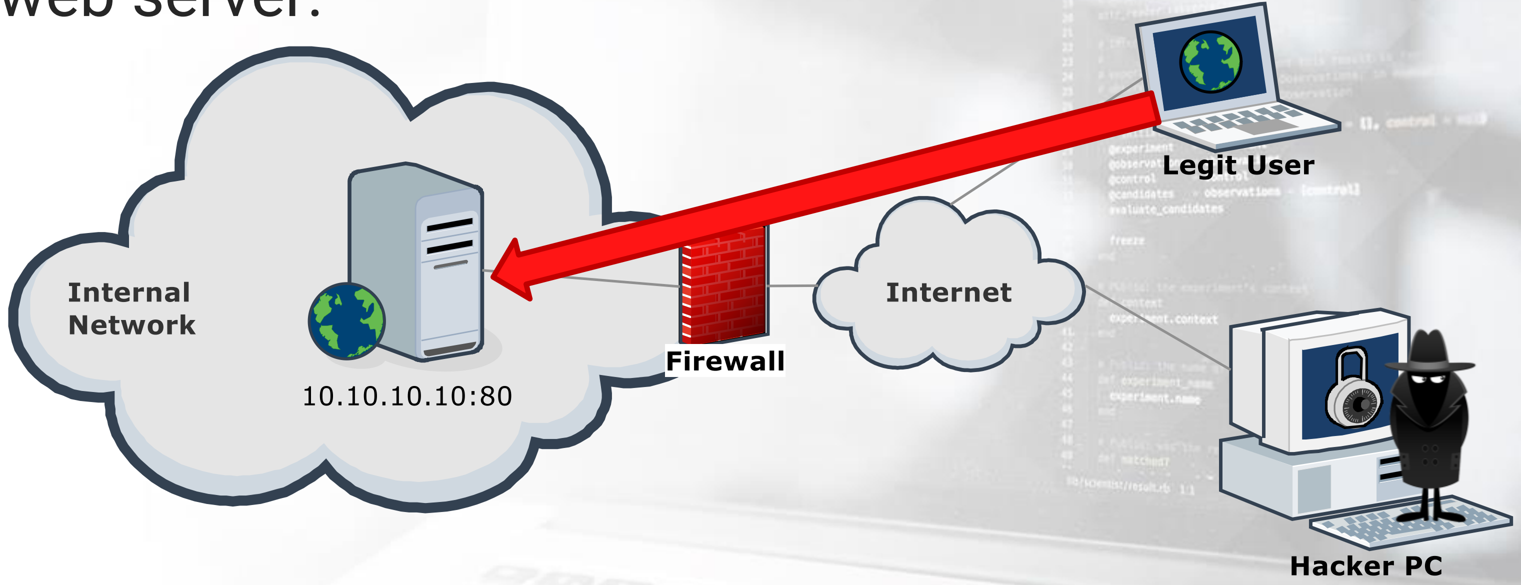
The traffic will pass even if it is not web traffic!

2.6.2.1 Packet Filtering vs. Application Attacks

Let's now look at a typical example where packet inspection does not stop a hacker from exploiting a server.

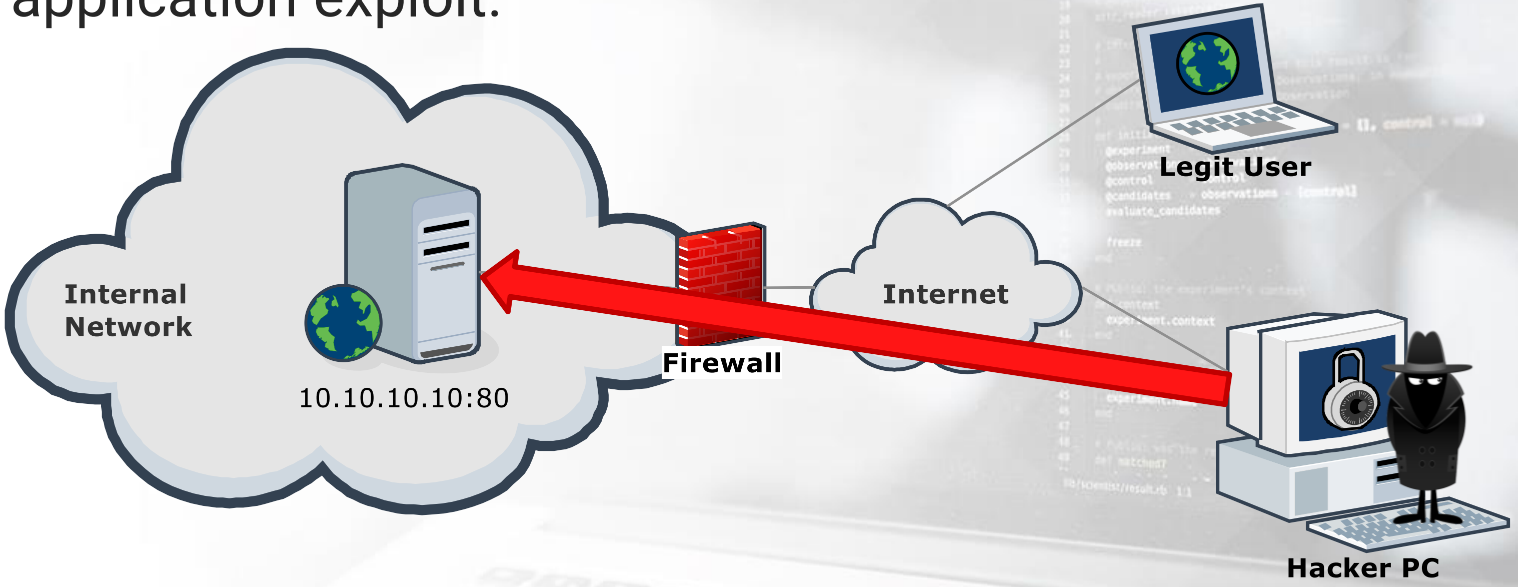
2.6.2.1 Packet Filtering vs. Application Attacks

A company hosts a web server. The firewall will allow all incoming traffic from the Internet and direct it to port 80 of the web server.



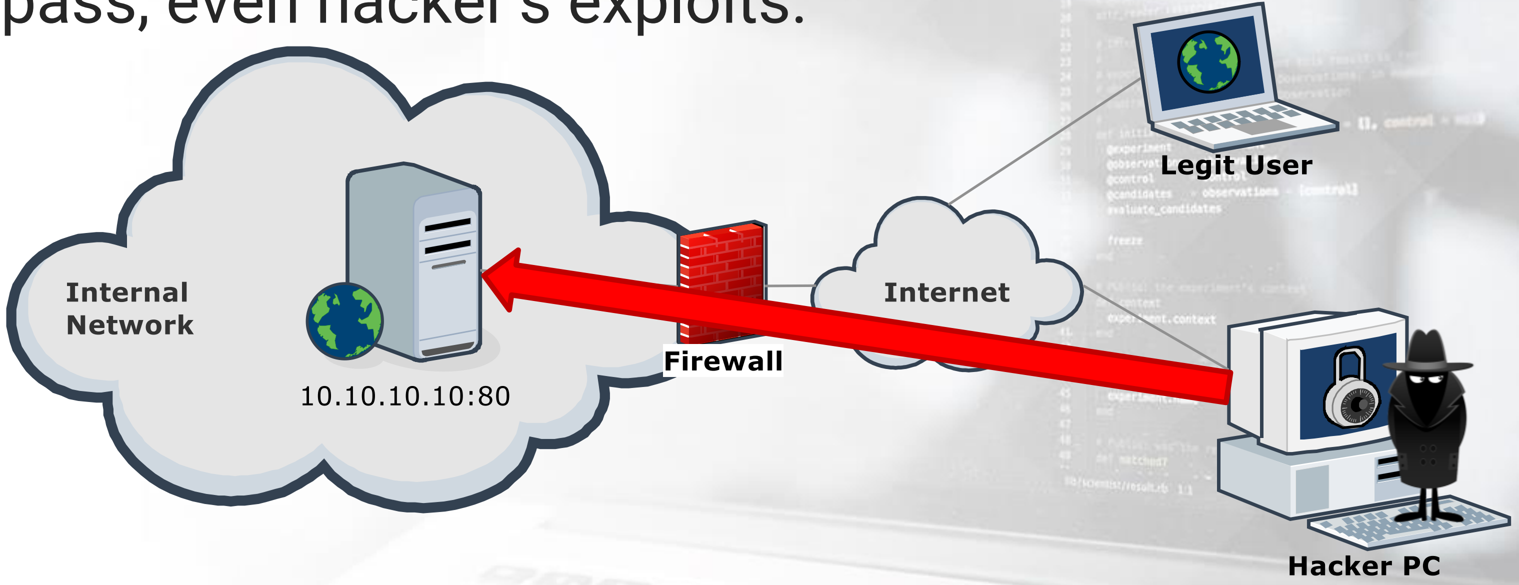
2.6.2.1 Packet Filtering vs. Application Attacks

In the same way, application exploits will go through, because the firewall cannot see the difference between web browsing and a web application exploit.



2.6.2.1 Packet Filtering vs. Application Attacks

The firewall can only filter traffic by using IP addresses, ports, and protocols. **Any** kind of application layer traffic will pass, even hacker's exploits.



2.6.2.1 Packet Filtering vs. Application Attacks

An application layer exploit could be an XSS, a buffer overflow, a SQL injection and much, much more.

Packet filtering is not enough to stop layer 7 attacks.

Let’s take a look at another scenario.

EXAMPLE

2.6.2.2 Packet Filtering vs. Trojan Horse

A hacker manages to exploit a workstation in a company network and wants to install a Trojan horse to remotely manage that machine.

The Trojan, by default, can be configured to **accept** connections on port 123 TCP or UDP or to **connect back** to the hacker's machine on port 123 TCP or UDP.

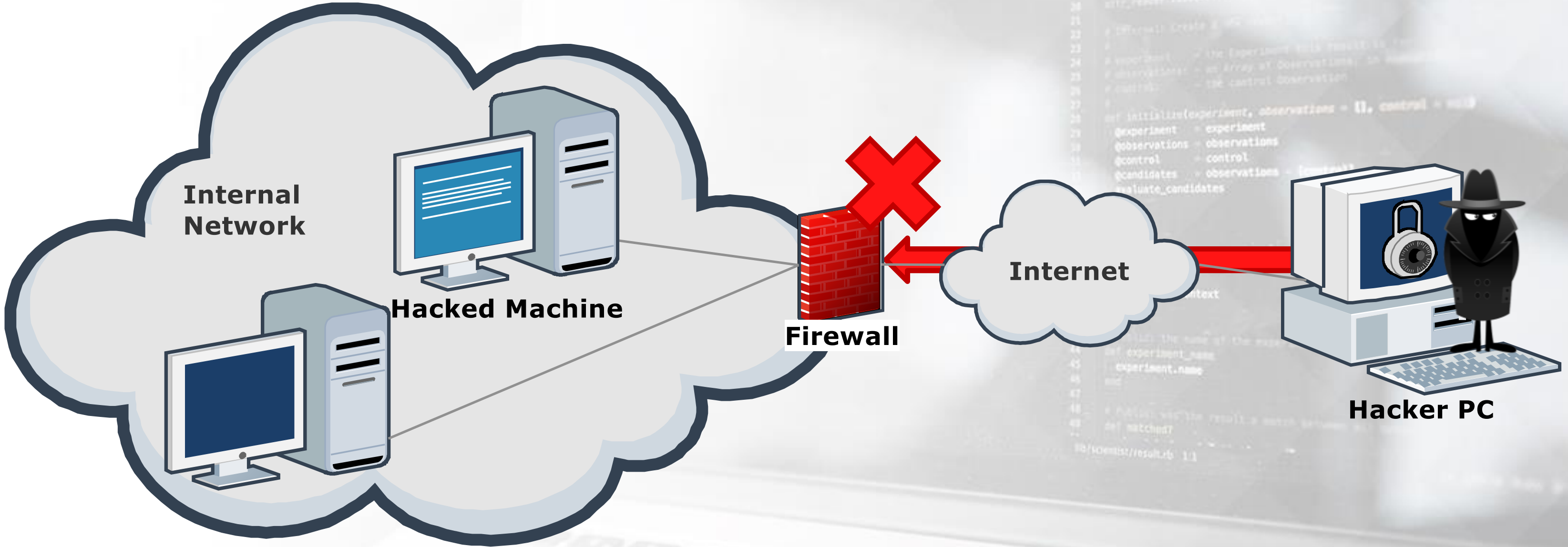
2.6.2.2 Packet Filtering vs. Trojan Horse

A typical firewall configuration rule is to let HTTP traffic (TCP.dst = 80) pass, so internal workstations can browse the Internet. The remaining traffic is dropped.

What happens to the Trojan horse?

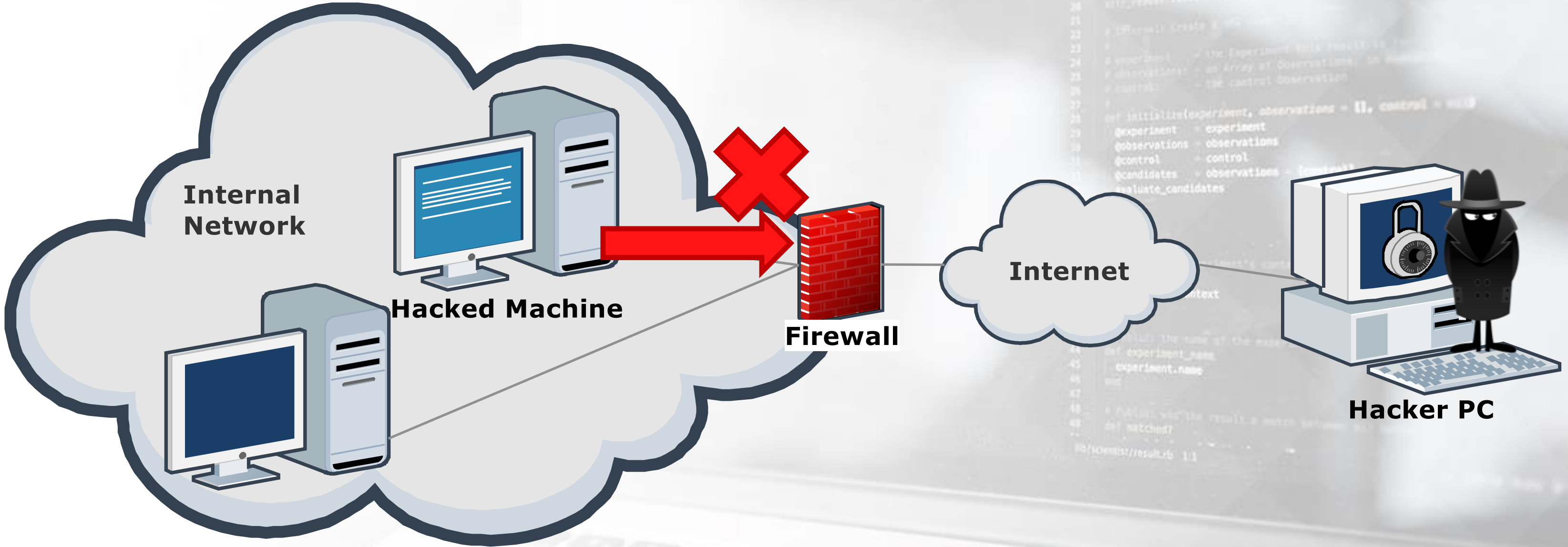
2.6.2.2 Packet Filtering vs. Trojan Horse

The hacker cannot connect to the infected machine, as there are no rules to allow traffic coming to port 123.



2.6.2.2 Packet Filtering vs. Trojan Horse

Similarly, the firewall will deny the traffic from the infected machine to the hacker's PC.

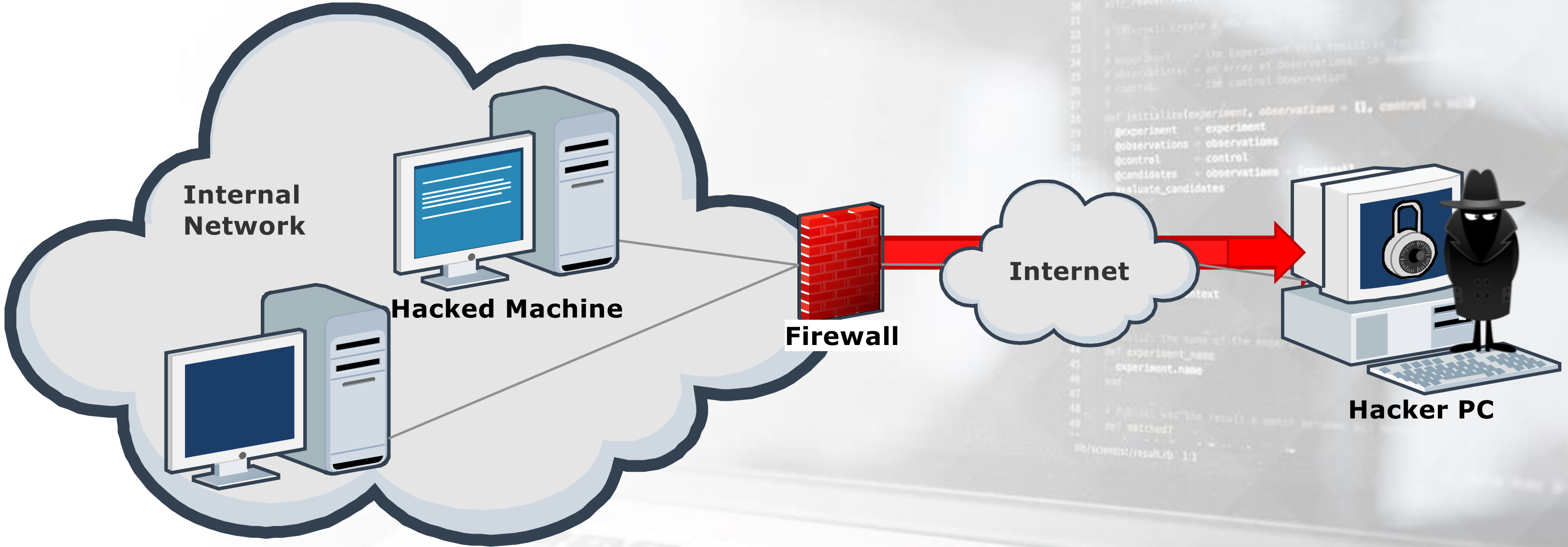


2.6.2.2 Packet Filtering vs. Trojan Horse

What happens if the hacker, who is smart and knows all the well-known ports by heart, configures the Trojan to **connect back** to his or her machine on port 80?

2.6.2.2 Packet Filtering vs. Trojan Horse

The firewall will **allow** the connection!



2.6.3 Application Layer Firewalls

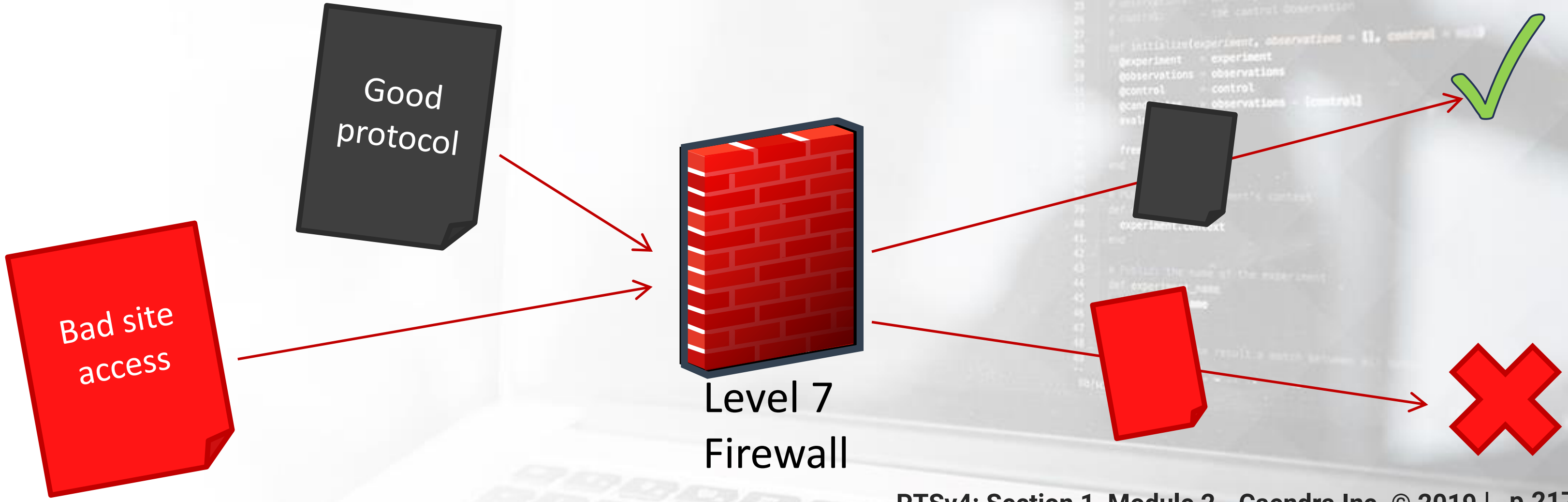
Application level firewalls work by checking all the OSI 7 layers.

They provide a more comprehensive protection because they inspect the actual content of a packet, not just its headers. For example:

- Drop any peer-to-peer application packet.
- Prevent users from visiting a site.

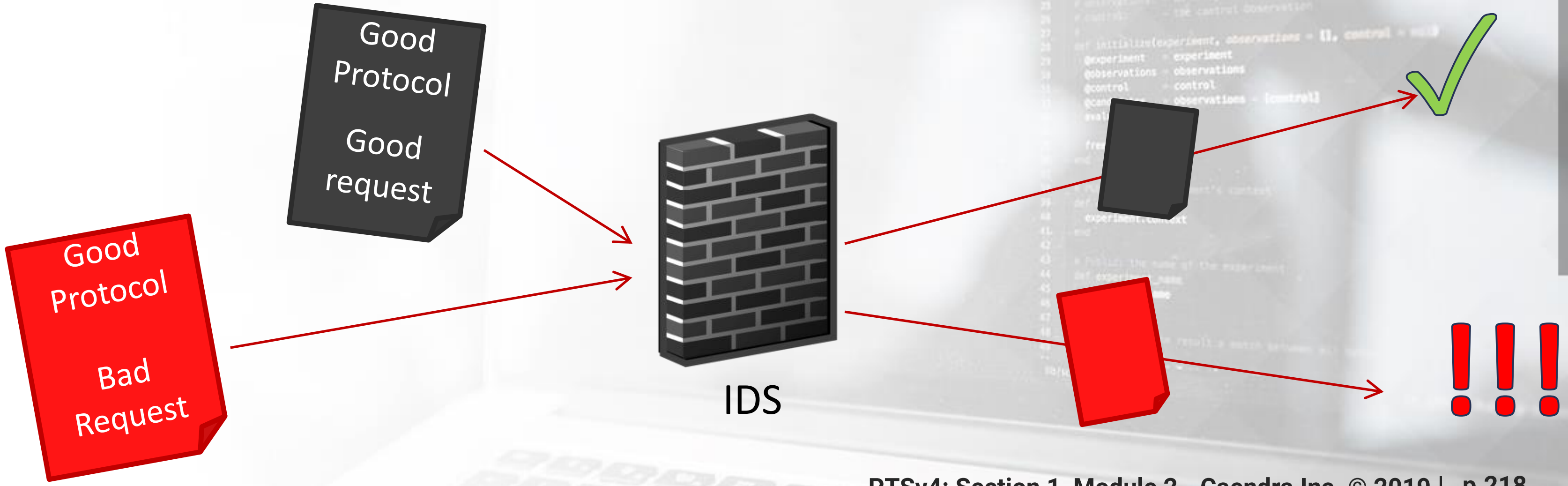
2.6.3 Application Layer Firewalls

Level 7 firewalls are indeed able to understand most of the application layer protocols in use nowadays. Organizations use them not only to protect their network from hackers but also to filter unwanted traffic.



2.6.4 IDS

There is not just traffic detection, but intrusion detection! **Intrusion Detection Systems** (IDS) inspect the application payload trying to detect any potential attack.



2.6.4 IDS

An IDS is specialized software used for **detecting ongoing intrusions**. It checks for attack vectors like ping sweeps, port scans, SQL injections, buffer overflows and so on.

IDS can also **identify traffic** generated by a virus or a worm. Pretty much every kind of network threat can be detected by a well-configured IDS.

2.6.4 IDS

An IDS, like an antivirus, detects risky traffic by means of **signatures**. The vendor provides frequent signature updates as soon as new attack vectors are found in the wild. Without the right signatures an IDS cannot detect and report an intrusion; the **IDS cannot detect something if it does not already know**.

There are also **false positives**. They happen when legit traffic is flagged as malicious.

2.6.4 IDS

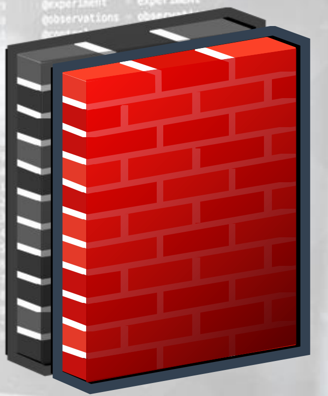
Detection is performed by a multitude of **sensors**, software components that inspect network traffic.

Sensors passively intercept intrusions and communicate them to the **IDS manager**, software in charge of maintaining policies and which provides a management console to the system administrator.

2.6.4 IDS

IDSs **do not substitute firewalls**.

They support firewalls by providing a further layer of security protecting the network from mainstream and well-known attack vectors.



2.6.4 IDS

IDSs fall into two main categories:

Network Intrusion Detection Systems (NIDS)

Host Intrusion Detection Systems (HIDS)

2.6.4.1 NIDS

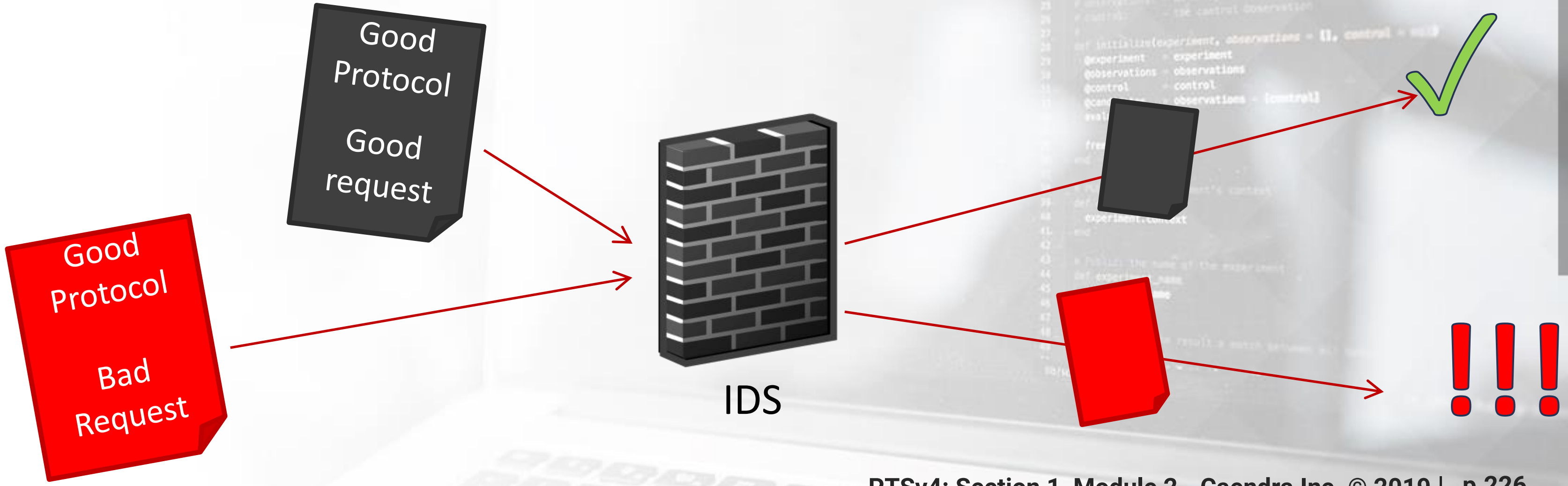
Network intrusion detection systems inspect network traffic by means of sensors which are usually placed on a router or in a network with a high intrusion risk, like a DMZ.

2.6.4.2 HIDS

On the other hand, host IDS sensors monitor application logs, file-system changes and changes to the operating system configuration.

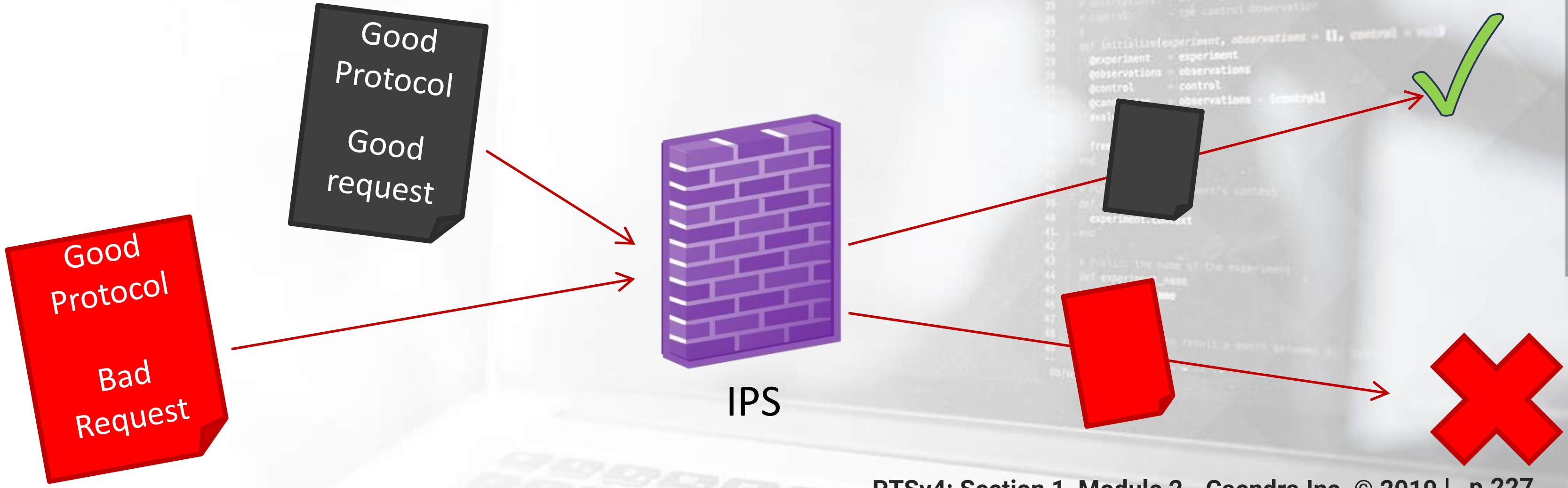
2.6.5 IPS

IDSs, unlike firewalls, can detect suspicious activities and report them to the network administrator. Suspicious activity is logged for future analysis, but **it is not blocked**.



2.6.5 IPS

Intrusion Prevention Systems (IPS) can **drop** malicious requests when the threat has a risk classification above a pre-defined threshold.



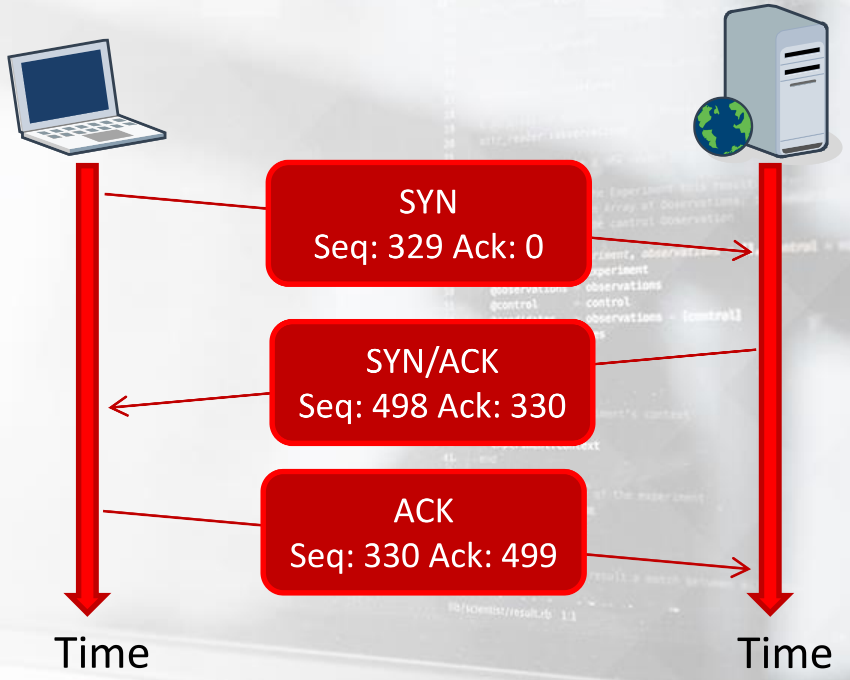
2.6.6 Spot an Obstacle

During penetration testing activities, you might want to identify if a firewall-like mechanism is used in the environment.

If you suspect presence of a firewall, you might want to check for anomalies in TCP Three-Way Handshake that was introduced previously in this module.

2.6.6 Spot an Obstacle

To recall, proper Three-way handshake looks like:

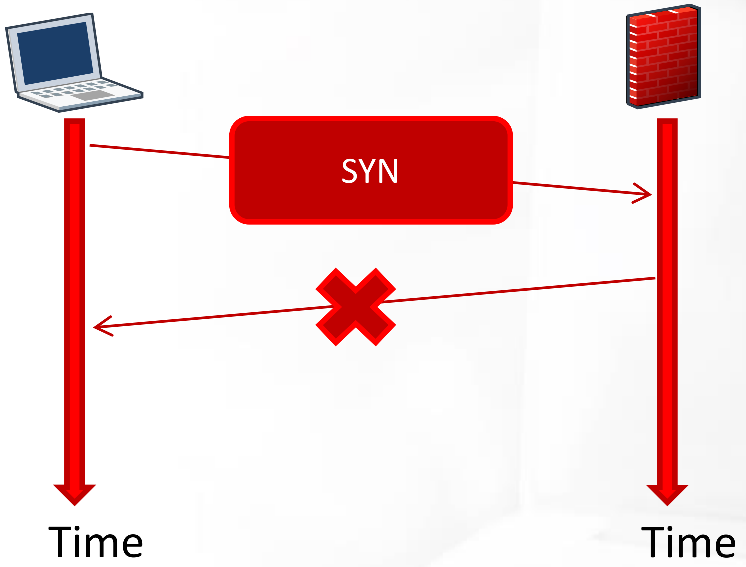


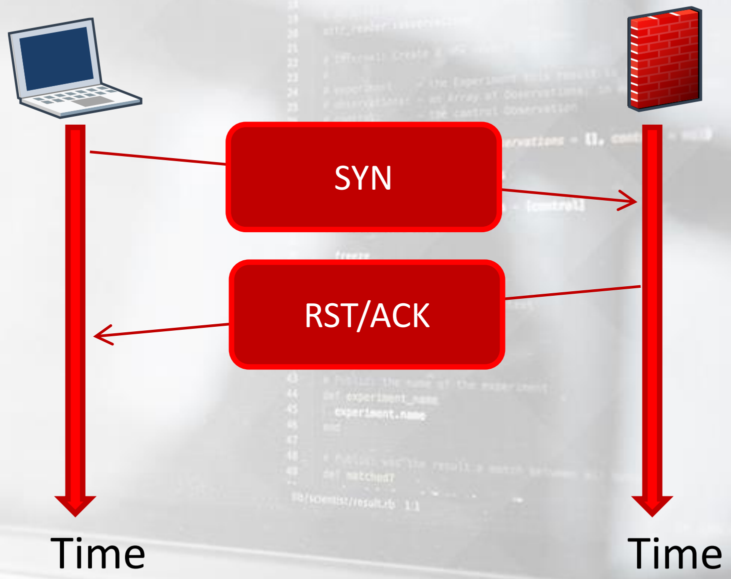
2.6.6 Spot an Obstacle

When a firewall is in place, the following behavior may be spotted:

- **TCP SYN** are sent, but there no **TCP SYN/ACK** replies
- **TCP SYN** packets are sent but a **TCP RST/ACK** reply is received

2.6.6 Spot an Obstacle





2.6.6 Spot an Obstacle

Note, that this type of observation does not determine whether the detected obstacle is a **firewall**, an **IDS**, or **any other** device; this just helps you to identify **environmental constraints**.

2.6.7 NAT and Masquerading

Firewalls not only filter packets but can also be used to implement **Network Address Translation** or **NAT**.

Note: Your home router is most probably running NAT protocol to connect all your home devices to the internet without having to have a public IP assigned for each of them.

What is NAT and why is it needed?

2.6.7 NAT and Masquerading

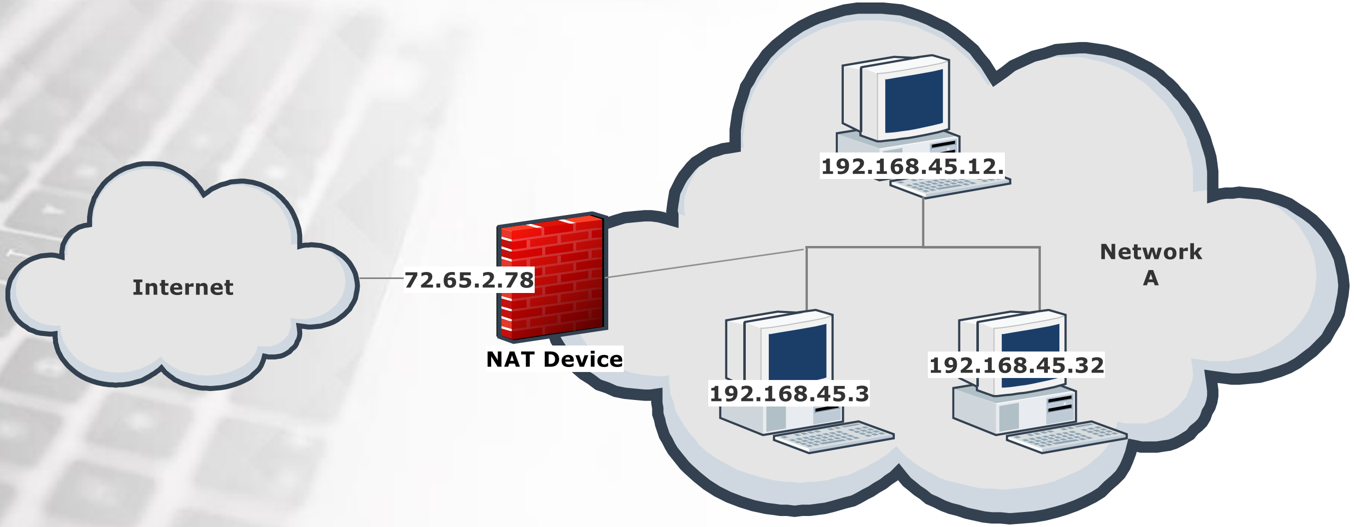
As you know, every machine on the Internet must have a unique IP address. This does not mean that every device that can **access** the internet must have a unique **public** IP address.

Network Address Translation (NAT) and **IP masquerading** are two techniques used to provide access to a network from another network.

EXAMPLE

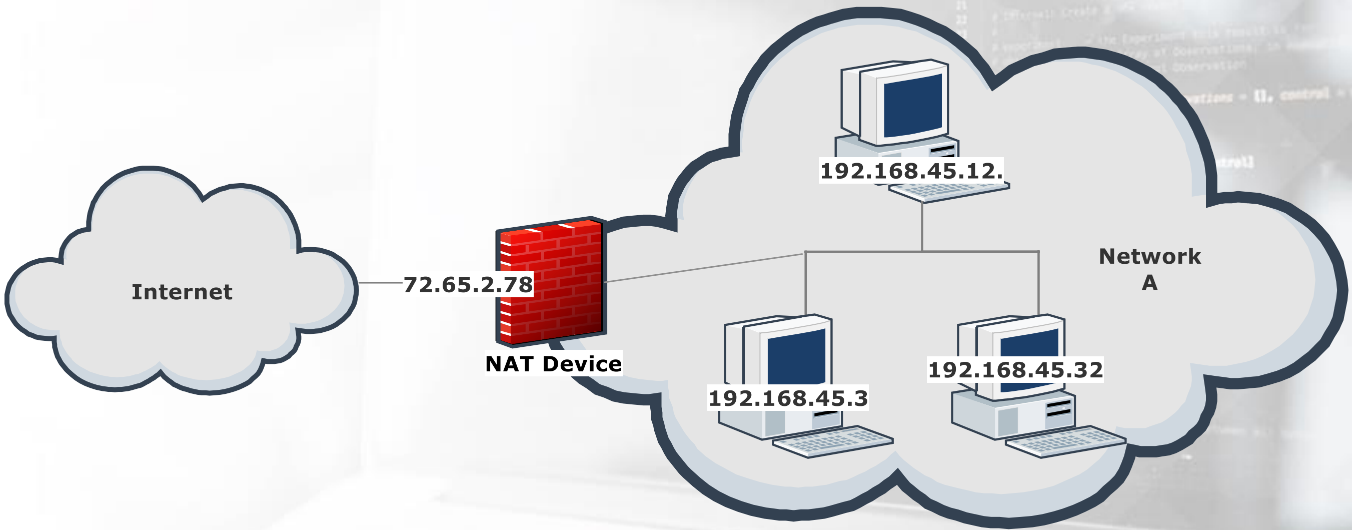
2.6.7 NAT and Masquerading

Network A can be a private network using a NAT device to access the Internet. A machine on the internet cannot directly access a machine in *Network A*.



2.6.7 NAT and Masquerading

But, a machine in *Network A* can access the Internet, if the NAT device allows the traffic to pass.



2.6.7 NAT and Masquerading

Every machine inside *Network A* will use the NAT device as **its default gateway**, thus routing its Internet traffic through it. The NAT device then rewrites the **source IP address** of every packet setting it to `72.65.2.78` (in our example), thus masquerading the original client's IP address.

A machine on the Internet will never know the original client's IP address.

2.6.8 Hera Lab – Find the Secret Server

It is now **practice time**!

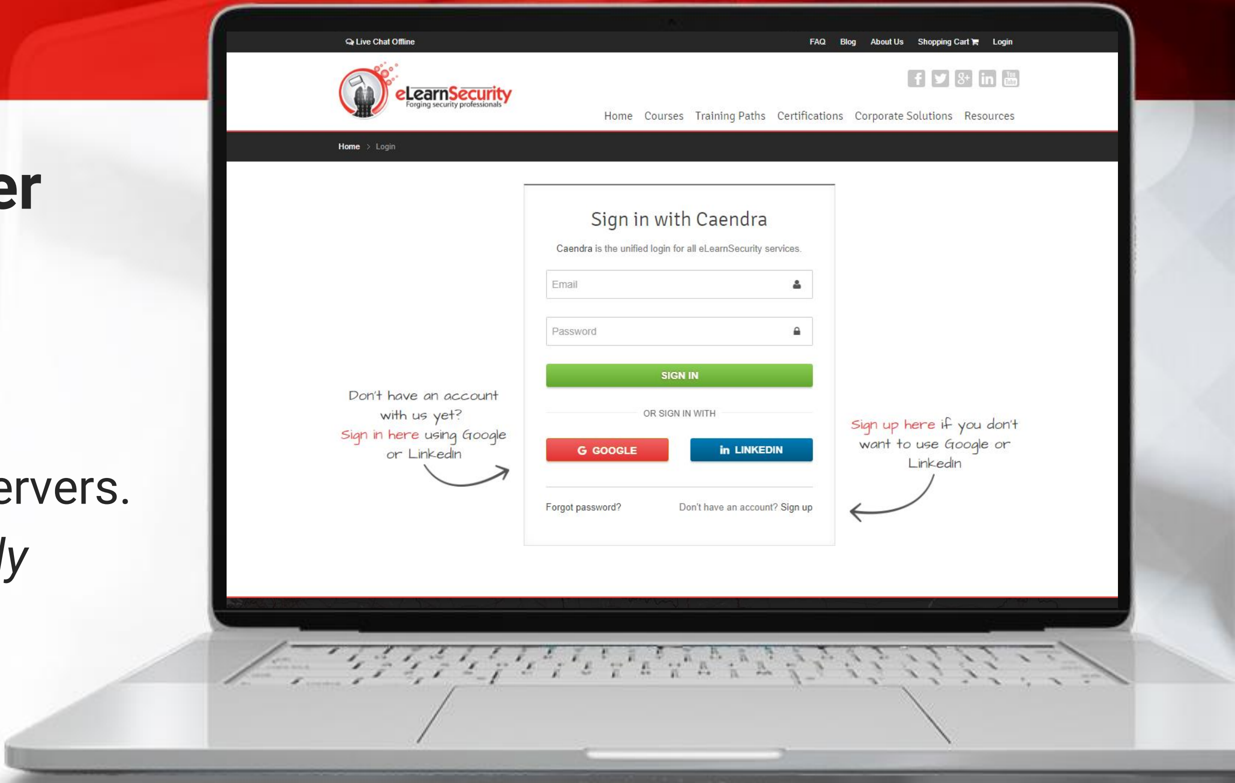
In the following lab, you will learn how security through obscurity cannot stop a skilled attacker. Try to solve the challenge by yourself; if you get stuck, you can check the solutions in the member's area.

2.6.8 Hera Lab – Find the Secret Server

Find the Secret Server

In this lab, you will:

- Understand routing information.
- Explore different web servers.
- Connect to an *apparently unreachable* server!



Live Chat Offline
FAQ
Blog
About Us
Shopping Cart
Login
eLearnSecurity
Forging security professionals
Home
Courses
Training Paths
Certifications
Corporate Solutions
Resources
Home > Login
Sign in with Caendra
Caendra is the unified login for all eLearnSecurity services.
Email
Password
SIGN IN
OR SIGN IN WITH
GOOGLE
LINKEDIN
Forgot password?
Don't have an account? Sign up
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.*

2.6.9 Resources

- Netfilter the Linux kernel packet filtering framework.
- Book: Firewall Fundamentals the essential guide to understanding and using firewalls to protect personal computers and your network.
- OSSEC IDS an Open Source Host-based Intrusion Detection System.
- IDS FAQ answers to simple questions related to detecting intruders who attack systems through the network.

http://www.netfilter.org/
http://www.amazon.com/Firewall-Fundamentals-Wes-Noonan/dp/1587052210/ref=cm_cr_pr_sims_t
http://www.ossec.net/
http://www.linuxsecurity.com/resource_files/intrusion_detection/network-intrusion-detection.html

2.7

DNS

2.7 DNS

How does this support my pentesting career?

- SSL/TLS certificates validation relies on DNS
- Mounting spoofing attacks
- Performing information gathering

2.7 DNS

The **Domain Name System**, or **DNS**, is the only application layer protocol you will see in this module.

The DNS primarily converts human-readable names, like `www.elearnsecurity.com`, to IP addresses and is a fundamental **support protocol** for the Internet and computer networks in general. It is widely recognized that the entire internet security is relying upon DNS.

2.7 DNS

For this course, you will need to know how the DNS service provides name resolution because every common operation on the Internet such as opening a web site, sending an email, and sharing a document involves the use of a DNS to resolve resource names to IP addresses (and vice versa).

2.7.1 DNS Structure

A DNS name such as `www.elearnsecurity.com` or `members.elearnsecurity.com` can be broken down into the following parts:

- Top level domain (TLD)
- Domain part
- Subdomain part (if applicable)
- Host part

EXAMPLE

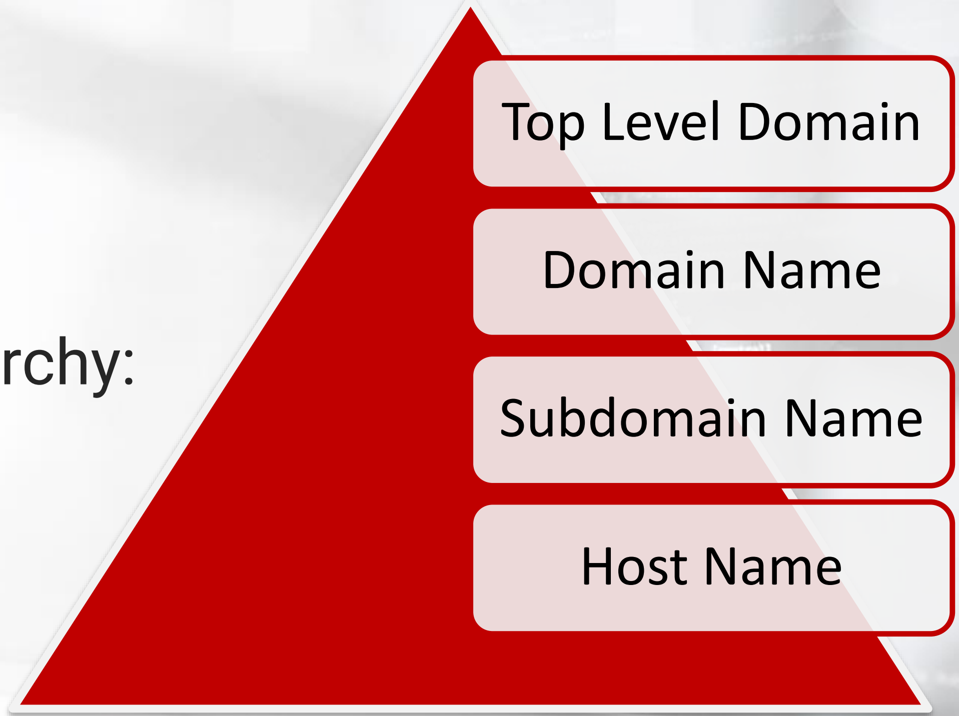
2.7.1 DNS Structure

$$\underbrace{\texttt{members}}_{\text{Host}}\texttt{.}\underbrace{\texttt{elearnsecurity}}_{\text{Domain}}\texttt{.}\underbrace{\texttt{com}}_{\substack{\text{Top Level}\\\text{Domain (TLD)}}}$$

$$\underbrace{\texttt{www}}_{\text{Host}}\texttt{.}\underbrace{\texttt{sub}}_{\substack{\text{Sub}\\\text{domain}}}\texttt{.}\underbrace{\texttt{domain}}_{\text{Domain}}\texttt{.}\underbrace{\texttt{com}}_{\substack{\text{Top Level}\\\text{Domain}}}$$

2.7.1 DNS Structure

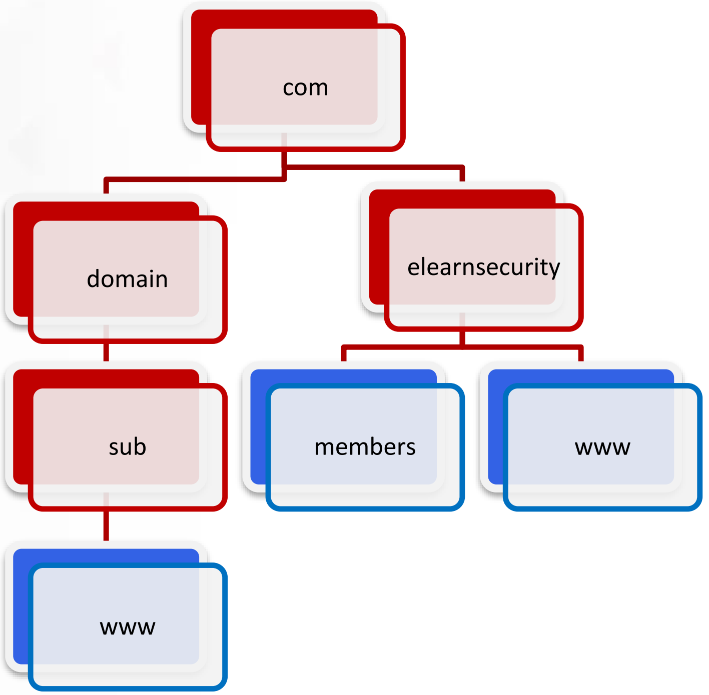
These parts form a hierarchy:



EXAMPLE

2.7.1 DNS Structure

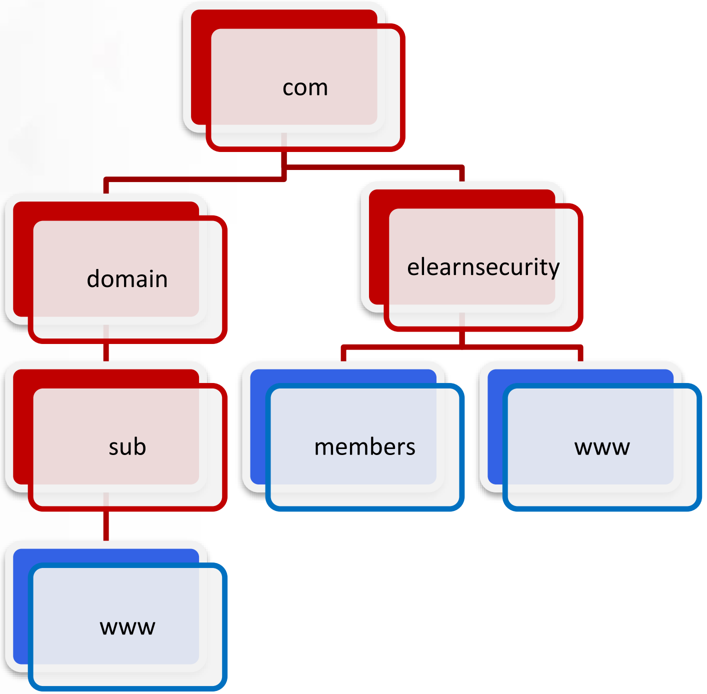
So, we can rewrite the previous names as:



EXAMPLE

2.7.1 DNS Structure

Where the blue squares are the hosts and the red ones are the top/sub/domain names.



2.7.1 DNS Structure

Name resolution is performed by **resolvers**, servers that contact the top level domain (TLD) DNS servers and follow the hierarchy of the DNS name to resolve the name of a host.

Resolvers are DNS servers provided by your ISP or publicly available like OpenDNS or Google DNS.

2.7.2 DNS Names Resolution

To convert a DNS name into an IP address, the operating system must contact a **resolver** server to perform the DNS resolution.

The resolver breaks down the DNS name in its parts and uses them to convert a DNS name into an IP address.

2.7.2.1 DNS Resolution Algorithm

1. Firstly, the resolver contacts one of the **root name servers**; these servers contain information about the top level domains.

2. Then, it asks the TLD name server what's the name server that can give information (authoritative name server) about the **domain** the resolver is looking for.

3. If there are one or more **subdomains**, step 2 is performed again on the authoritative DNS server for every subdomain

4. Finally, the resolver asks for the name resolution of the **host** part.

EXAMPLE

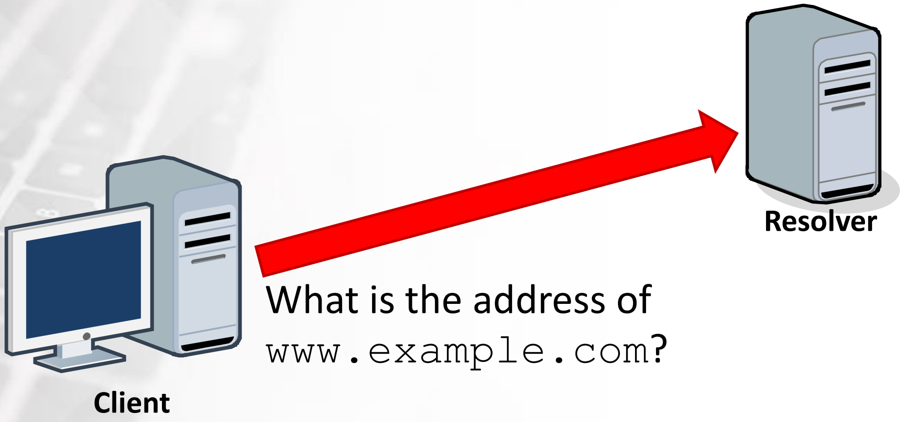
2.7.2.2 DNS Resolution Example

A computer needs to open a web page on `www.example.com.`

EXAMPLE

2.7.2.2 DNS Resolution Example

To do that, it contacts the resolver configured by the local administrator (e.g., OpenDNS).

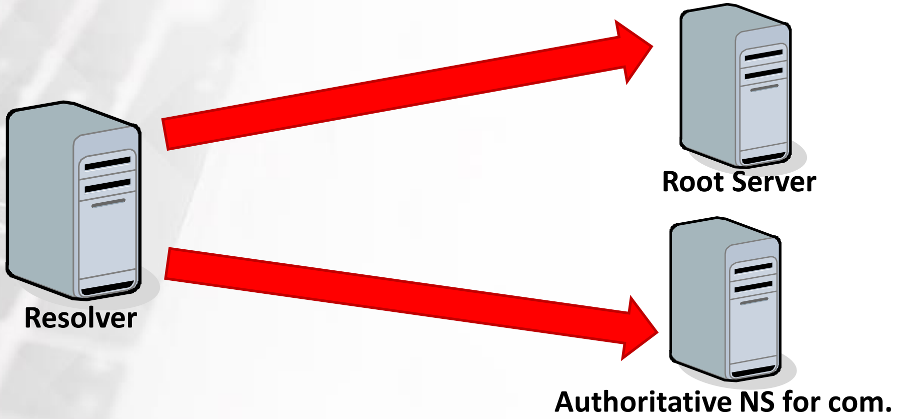


EXAMPLE

2.7.2.2 DNS Resolution Example

The resolver contacts a **root server** and asks about the authoritative name server(s) for the `com.` domain.

Then the resolver contacts that authoritative name server and asks what is an authoritative name server for the `example.com.` domain.

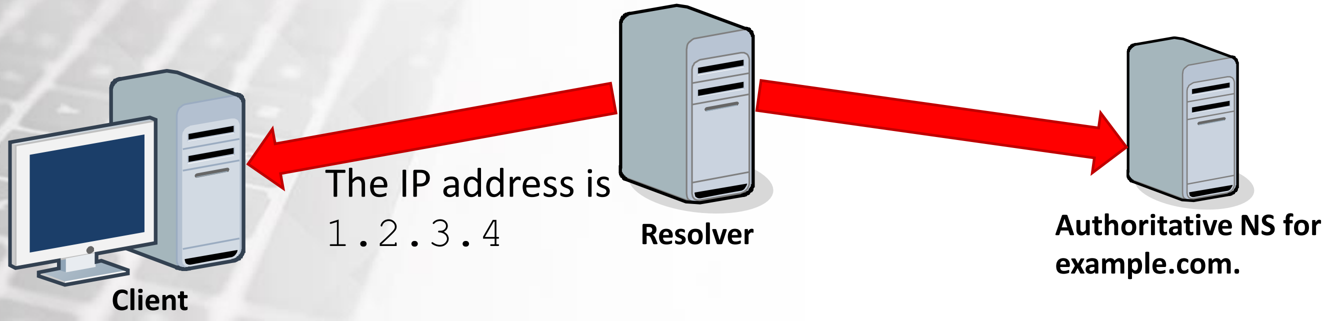


EXAMPLE

2.7.2.2 DNS Resolution Example

The resolver then asks what the address of www is.

Finally, the resolver sends the IP address back to the client.



2.7.3 Resolvers and Root Servers

How can a resolver know how to contact a **root name server**?

IP addresses of the root servers are **hardcoded in the configuration** of the resolver. System administrators keep the list updated, otherwise, the resolver would not be able to contact a root server!

2.7.4 Reverse DNS Resolution

The domain name system can also perform the inverse operation; it can convert an **IP address to a DNS name**.

Keep in mind that this is not always the case; the administrator of a domain must have enabled and configured this feature for the domain to make it work.

EXAMPLE

2.7.4 Reverse DNS Resolution

Many tools use the reverse DNS if it's available.

The Linux ping utility performs a reverse DNS query after receiving every response from the target.

```
$ ping www.yahoo.com
PING fd-fp3.wg1.b.yahoo.com (46.228.47.115) 56(84) bytes of data.
64 bytes from ir1.fp.vip.ir2.yahoo.com (46.228.47.115): icmp_req=1 ttl=49 time=125 ms

--- fd-fp3.wg1.b.yahoo.com ping statistics ---
2 packets transmitted, 1 received, 50% packet loss, time 1001ms
rtt min/avg/max/mdev = 125.706/125.706/125.706/0.000 ms
```

2.7.5 More about the DNS

In this section, you have seen how the DNS is used to perform name resolution. The domain name system is not just that, it is used to identify what the mail servers for a domain are, to know what is the right server for a specific role and much more.

Please refer to RFC1034 and RFC1035 to dig more into DNS!

NOTE: the DNS is also very important to the security of the whole internet because breaking DNS security means breaking SSL and TLS. This, however, is beyond the scope of this training course.

https://www.ietf.org/rfc/rfc1034.txt
https://www.ietf.org/rfc/rfc1035.txt

2.8

Wireshark

2.8 Wireshark

The best way to deeply understand the topics of this module is to see the actual protocols in action. You can do that by using a sniffer tool.

This section will enhance your Networking and **Wireshark skills**.

2.8 Wireshark

As you know, Wireshark is a network sniffer and protocol analyzer.

This means that you can use it to analyze every packet, traffic stream, or connection **that hits your computer network interface(s)**.

2.8 Wireshark

Knowing this tool is extremely important to understand how networking works.

Wireshark is widely used by network administrators, networking protocol researchers, and hackers.

2.8 Wireshark

Wireshark can capture all the traffic **seen** by the network card of the computer running it.

To understand what traffic a network card sees, you have to know that most network cards, also known as Network Interface Cards (NIC), can work in **promiscuous or monitor mode**.

2.8.1 NIC Promiscuous Mode

During normal operations, a network card **discards** any packet addressed to another NIC. In **promiscuous mode**, a network card will accept and process **any** packet it receives.

For example, in a hub-based network, a NIC will receive traffic addressed to other machines. The NIC usually drops these packets but accepts them while in promiscuous mode.

2.8.1 NIC Promiscuous Mode

With the introduction of switched networks, sniffing other machines Ethernet traffic got harder. You have to perform an attack such as ARP poisoning or MAC flooding in order to do that.

WiFi medium (the air), instead, is broadcast by nature, so it’s possible to still detect traffic destined to a different host. In this chapter, we will concentrate on Ethernet traffic only.

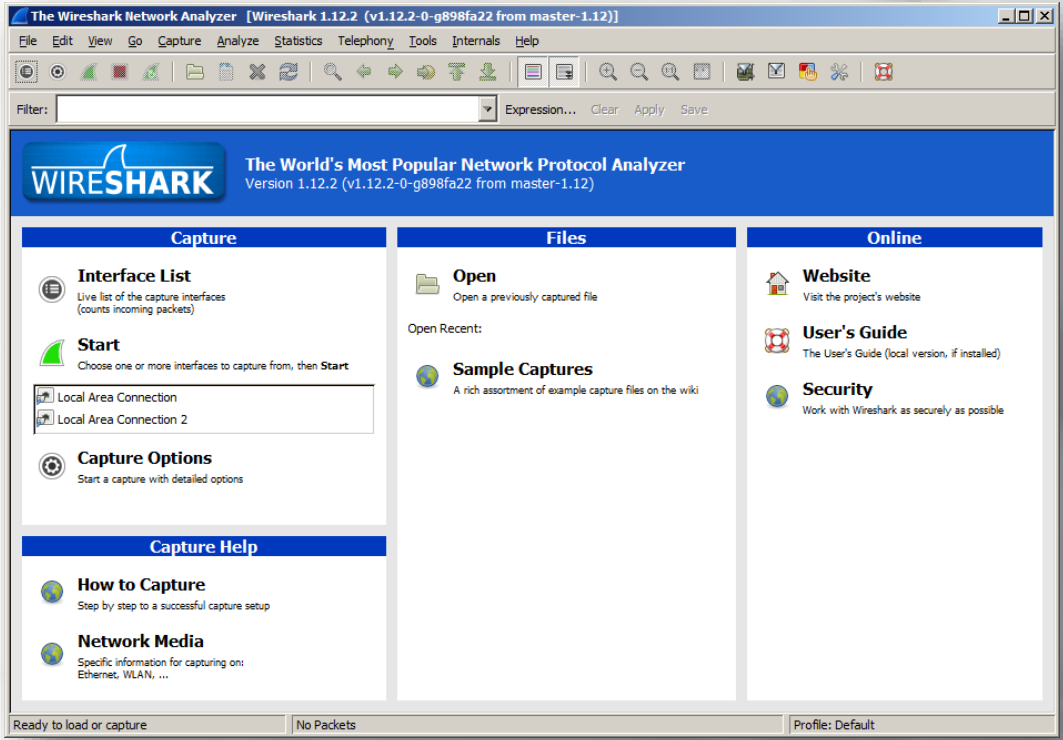
2.8.2 Configuring Wireshark

Wireshark is free software that can run on practically all modern operating systems. You can download it from https://www.wireshark.org/ or https://www.wireshark.org/download/ for older versions.

In the following slides, we are going to see how to configure it to use its main features. The version covered is 1.12.2.

2.8.2 Configuring Wireshark

Here we see Wireshark’s main window.



The World's Most Popular Network Protocol Analyzer
Version 1.12.2 (v1.12.2-0-g898fa22 from master-1.12)

Live list of the capture interfaces (counts incoming packets)

Choose one or more interfaces to capture from, then **Start**

Start a capture with detailed options

Step by step to a successful capture setup

Specific information for capturing on: Ethernet, WLAN, ...

Open a previously captured file

A rich assortment of example capture files on the wiki

Visit the project's website

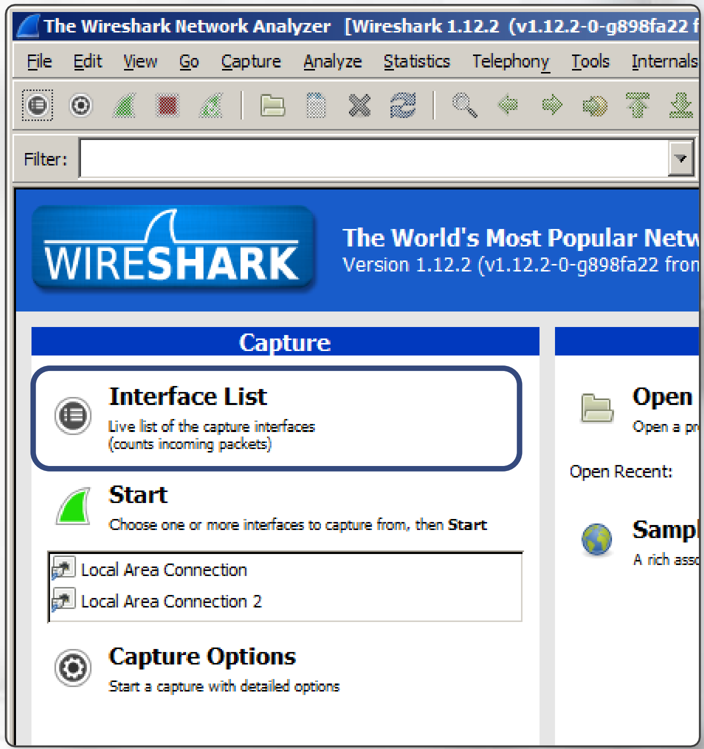
The User's Guide (local version, if installed)

Work with Wireshark as securely as possible

Ready to load or capture | No Packets | Profile: Default

2.8.2 Configuring Wireshark

Clicking on **Interface List** opens a window with a list of your network cards (wired, wireless, VPNs, virtual interfaces, etc.).


The Wireshark Network Analyzer [Wireshark 1.12.2 (v1.12.2-0-g898fa22 f

File Edit View Go Capture Analyze Statistics Telephony Tools Internals
Filter:
WIRESHARK

The World's Most Popular Netw
Version 1.12.2 (v1.12.2-0-g898fa22 fron

Capture

Interface List
Live list of the capture interfaces (counts incoming packets)

Start
Choose one or more interfaces to capture from, then **Start**

Local Area Connection
Local Area Connection 2

Capture Options
Start a capture with detailed options

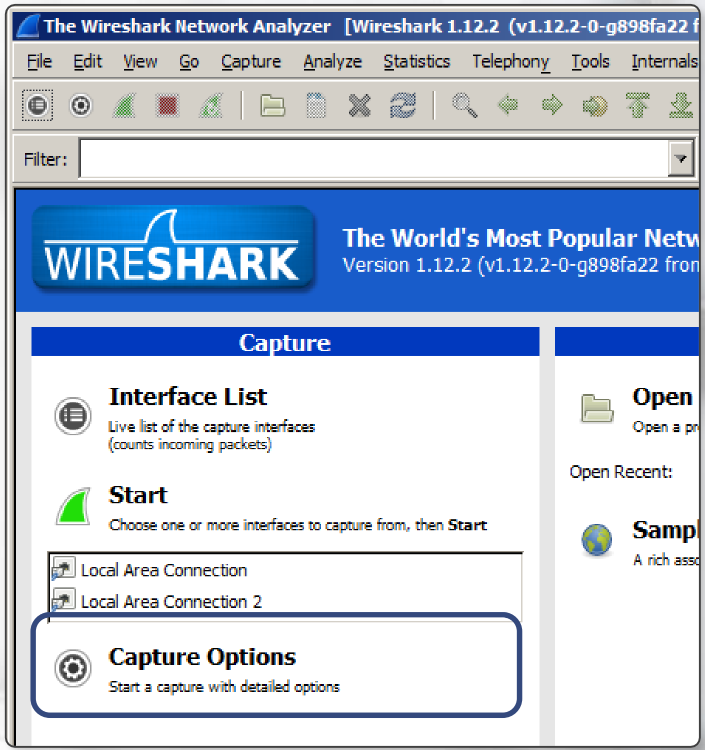
Open
Open a pr

Open Recent:

Sampl
A rich asso

2.8.2 Configuring Wireshark

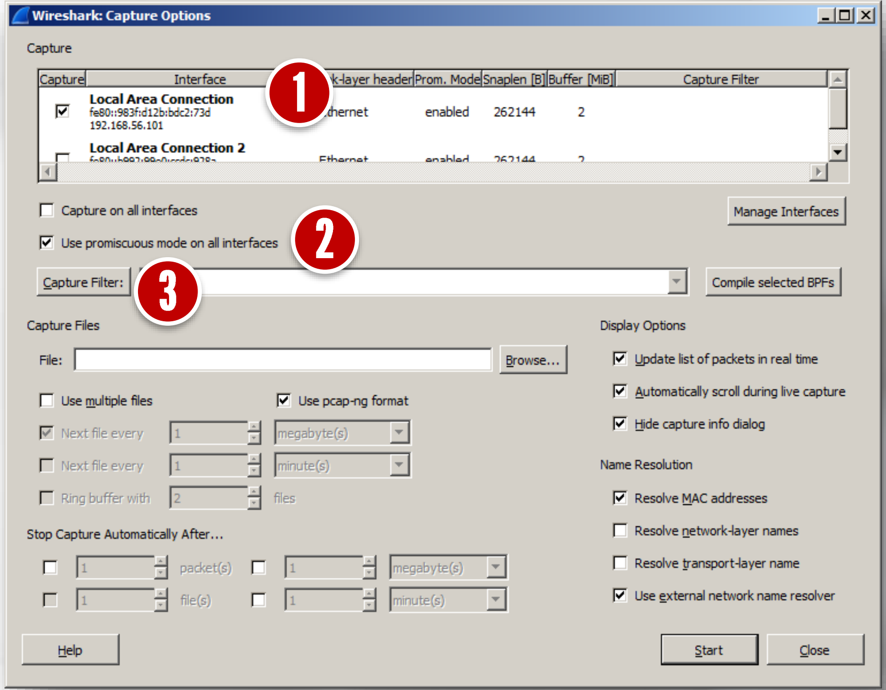
Clicking on **Capture Options** opens...


The Wireshark Network Analyzer [Wireshark 1.12.2 (v1.12.2-0-g898fa22 f
File Edit View Go Capture Analyze Statistics Telephony Tools Internals
Filter:
WIRESHARK
The World's Most Popular Netw
Version 1.12.2 (v1.12.2-0-g898fa22 from
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
Local Area Connection
Local Area Connection 2
Capture Options
Start a capture with detailed options
Open
Open a pr
Open Recent:
Sampl
A rich ass

2.8.2 Configuring Wireshark

... the **capture options** window, which has a huge impact on your capture session as you can configure:

1. Which interfaces to use during the capture
2. NIC promiscuous mode
3. Capture filtering


Wireshark: Capture Options

Capture

Capture	Interface	k-layer header	Prom. Mode	Snaplen [B]	Buffer [MiB]	Capture Filter
☑	**Local Area Connection** fe80::983f:d12b:bdc2:73d 192.168.56.101	thernet	enabled	262144	2	
☐	**Local Area Connection 2**	Ethernet	enabled	262144	2	

☐ Capture on all interfaces
☑ Use promiscuous mode on all interfaces
Capture Filter:

Capture Files
File:
☐ Use multiple files
☑ Use pcap-ng format
☑ Next file every 1 megabyte(s)
☐ Next file every 1 minute(s)
☐ Ring buffer with 2 files

Stop Capture Automatically After...
☐ 1 packet(s) ☐ 1 megabyte(s)
☐ 1 file(s) ☐ 1 minute(s)

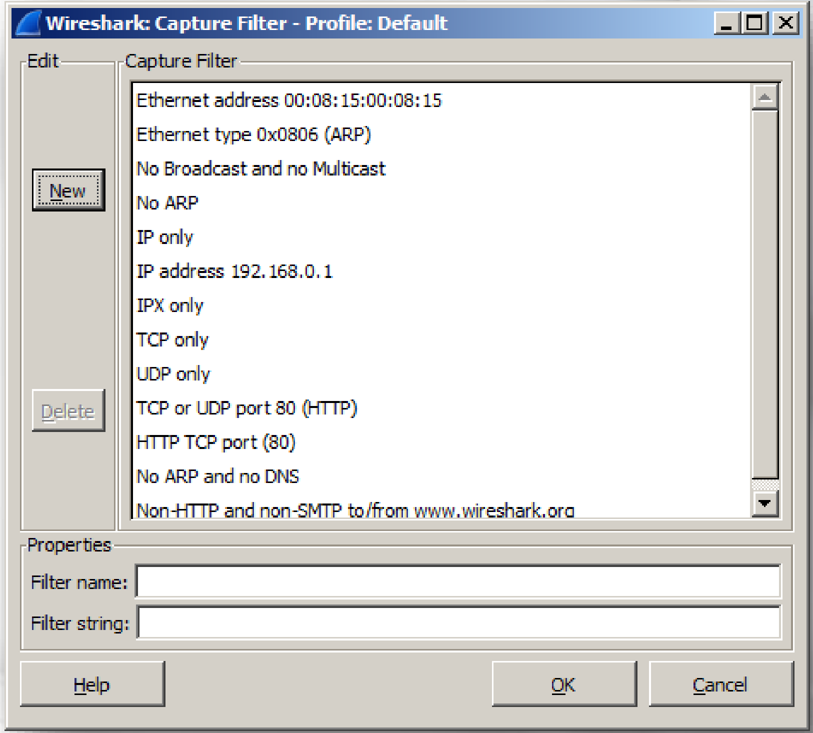
Display Options
☑ Update list of packets in real time
☑ Automatically scroll during live capture
☑ Hide capture info dialog

Name Resolution
☑ Resolve MAC addresses
☐ Resolve network-layer names
☐ Resolve transport-layer name
☑ Use external network name resolver

2.8.2 Configuring Wireshark

Capture filters will make Wireshark discard packets that do not match the filter. These filters impact how many packets your computer must process and how big the capture file will be.

This is very useful to limit captured traffic in high traffic networks.

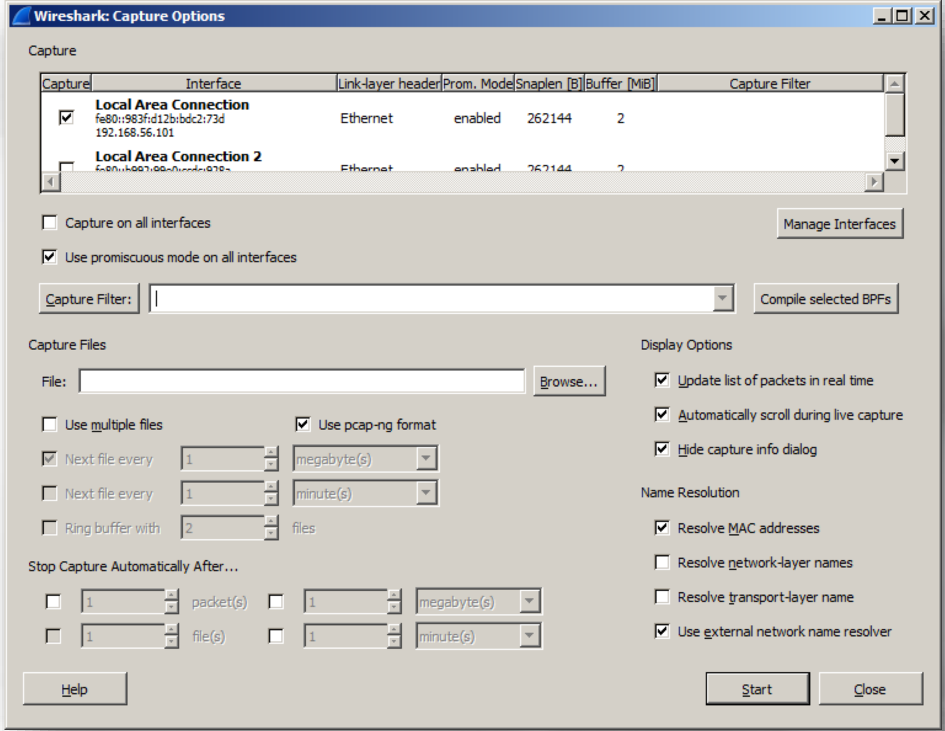


Ethernet address 00:08:15:00:08:15
Ethernet type 0x0806 (ARP)
No Broadcast and no Multicast
No ARP
IP only
IP address 192.168.0.1
IPX only
TCP only
UDP only
TCP or UDP port 80 (HTTP)
HTTP TCP port (80)
No ARP and no DNS
Non-HTTP and non-SMTP to/from www.wireshark.org

Filter name:
Filter string:

2.8.2 Configuring Wireshark

During your first captures, leave the filter blank and just click **start**.



Wireshark: Capture Options

Capture

Capture	Interface	Link-layer header	Prom. Mode	Snaplen [B]	Buffer [MiB]	Capture Filter
☑	Local Area Connection fe80::983f:d12b:bdc2:73d 192.168.56.101	Ethernet	enabled	262144	2	
☐	Local Area Connection 2	Ethernet	enabled	262144	2	

☐ Capture on all interfaces

☑ Use promiscuous mode on all interfaces

Capture Filter:

Capture Files

File:

☐ Use multiple files ☑ Use pcap-ng format

☑ Next file every 1 megabyte(s)

☐ Next file every 1 minute(s)

☐ Ring buffer with 2 files

Stop Capture Automatically After...

☐ 1 packet(s) ☐ 1 megabyte(s)

☐ 1 file(s) ☐ 1 minute(s)

Display Options

☑ Update list of packets in real time

☑ Automatically scroll during live capture

☑ Hide capture info dialog

Name Resolution

☑ Resolve MAC addresses

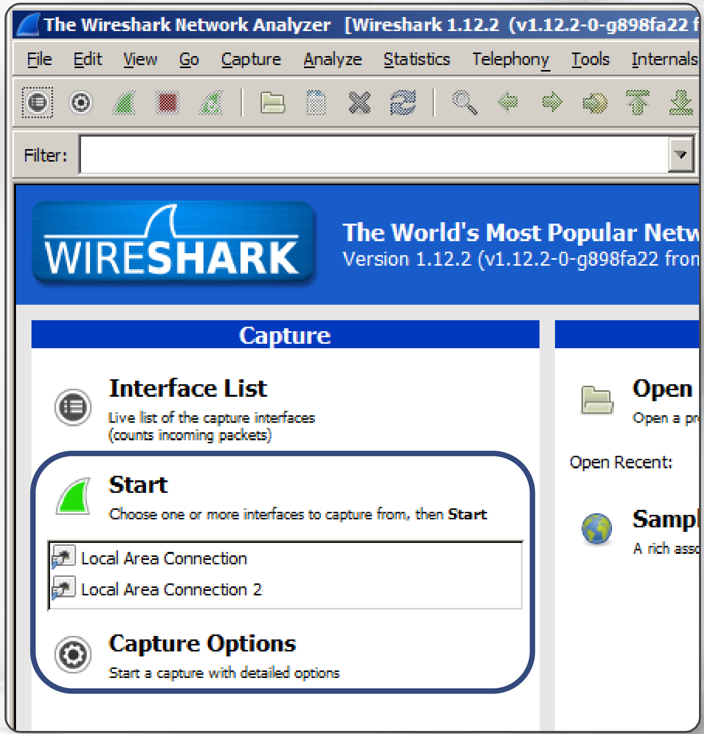
☐ Resolve network-layer names

☐ Resolve transport-layer name

☑ Use external network name resolver

2.8.2 Configuring Wireshark

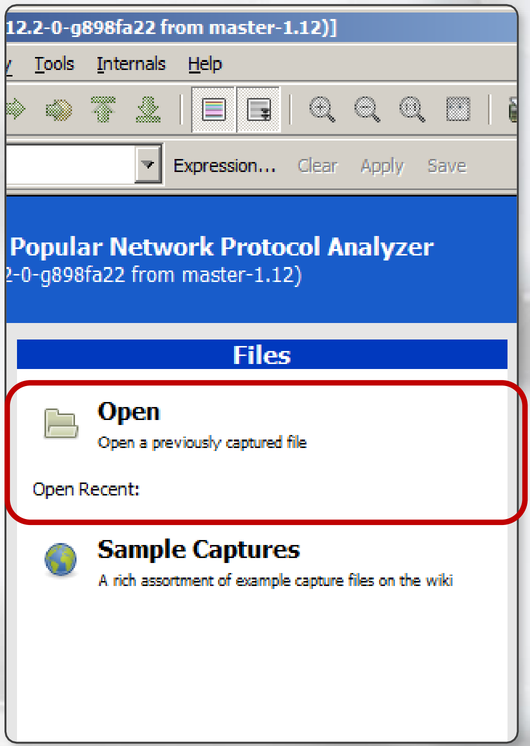
To perform these very same operations, you can just select the capture interface and then click on **capture options** or **start** from the main window.


The Wireshark Network Analyzer [Wireshark 1.12.2 (v1.12.2-0-g898fa22 f
File Edit View Go Capture Analyze Statistics Telephony Tools Internals
Filter:
WIRESHARK
The World's Most Popular Netw
Version 1.12.2 (v1.12.2-0-g898fa22 fron
Capture
Interface List
Live list of the capture interfaces (counts incoming packets)
Start
Choose one or more interfaces to capture from, then Start
Local Area Connection
Local Area Connection 2
Capture Options
Start a capture with detailed options
Open
Open a pr
Open Recent:
Sampl
A rich ass

2.8.2 Configuring Wireshark

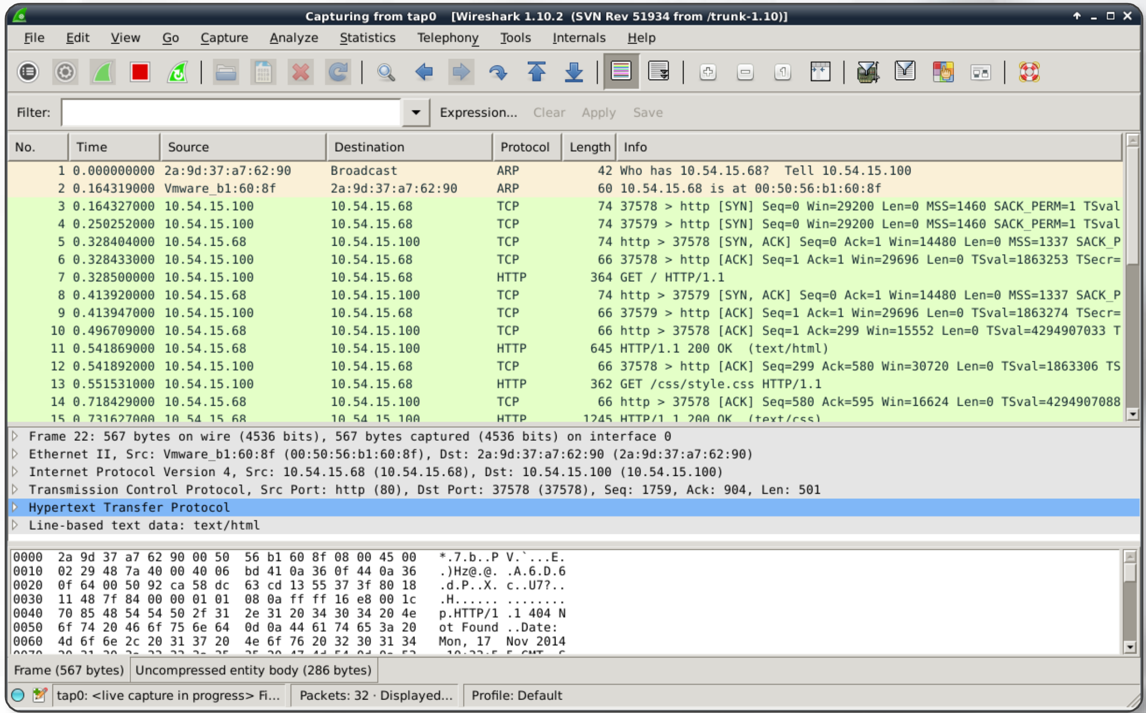
PCAP files store an entire capture (from a previous capture session).

If you already have a PCAP file, you can open it using this button.


12.2-0-g898fa22 from master-1.12)]
Tools Internals Help
Expression... Clear Apply Save
Popular Network Protocol Analyzer
2-0-g898fa22 from master-1.12)
Files
Open
Open a previously captured file
Open Recent:
Sample Captures
A rich assortment of example capture files on the wiki

2.8.3 The capture window

In either case, doing a live capture or opening a previous one, you will see this interface.


Capturing from tap0 [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time        Source             Destination        Protocol Length Info
   1 0.000000000 2a:9d:37:a7:62:90  Broadcast          ARP        42 Who has 10.54.15.68?  Tell 10.54.15.100
   2 0.164319000 Vmware_b1:60:8f    2a:9d:37:a7:62:90  ARP        60 10.54.15.68 is at 00:50:56:b1:60:8f
   3 0.164327000 10.54.15.100       10.54.15.68        TCP        74 37578 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
   4 0.250252000 10.54.15.100       10.54.15.68        TCP        74 37579 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
   5 0.328404000 10.54.15.68        10.54.15.100       TCP        74 http > 37578 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
   6 0.328433000 10.54.15.100       10.54.15.68        TCP        66 37578 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863253 TSecr=
   7 0.328500000 10.54.15.100       10.54.15.68        HTTP      364 GET / HTTP/1.1
   8 0.413920000 10.54.15.68        10.54.15.100       TCP        74 http > 37579 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
   9 0.413947000 10.54.15.100       10.54.15.68        TCP        66 37579 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863274 TSecr=
  10 0.496709000 10.54.15.68        10.54.15.100       TCP        66 http > 37578 [ACK] Seq=1 Ack=299 Win=15552 Len=0 TSval=4294907033 T
  11 0.541869000 10.54.15.68        10.54.15.100       HTTP      645 HTTP/1.1 200 OK  (text/html)
  12 0.541892000 10.54.15.100       10.54.15.68        TCP        66 37578 > http [ACK] Seq=299 Ack=580 Win=30720 Len=0 TSval=1863306 TS
  13 0.551531000 10.54.15.100       10.54.15.68        HTTP      362 GET /css/style.css HTTP/1.1
  14 0.718429000 10.54.15.68        10.54.15.100       TCP        66 http > 37578 [ACK] Seq=580 Ack=595 Win=16624 Len=0 TSval=4294907088
  15 0.731627000 10.54.15.68        10.54.15.100       HTTP     1245 HTTP/1.1 200 OK  (text/css)
```

```
▷ Frame 22: 567 bytes on wire (4536 bits), 567 bytes captured (4536 bits) on interface 0
▷ Ethernet II, Src: Vmware_b1:60:8f (00:50:56:b1:60:8f), Dst: 2a:9d:37:a7:62:90 (2a:9d:37:a7:62:90)
▷ Internet Protocol Version 4, Src: 10.54.15.68 (10.54.15.68), Dst: 10.54.15.100 (10.54.15.100)
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 37578 (37578), Seq: 1759, Ack: 904, Len: 501
▷ Hypertext Transfer Protocol
▷ Line-based text data: text/html
```

```
0000  2a 9d 37 a7 62 90 00 50  56 b1 60 8f 08 00 45 00   *.7.b..P V.`...E.
0010  02 29 48 7a 40 00 40 06  bd 41 0a 36 0f 44 0a 36   .)Hz@.@. .A.6.D.6
0020  0f 64 00 50 92 ca 58 dc  63 cd 13 55 37 3f 80 18   .d.P..X. c..U7?..
0030  11 48 7f 84 00 00 01 01  08 0a ff ff 16 e8 00 1c   .H...... ........
0040  70 85 48 54 54 50 2f 31  2e 31 20 34 30 34 20 4e   p.HTTP/1 .1 404 N
0050  6f 74 20 46 6f 75 6e 64  0d 0a 44 61 74 65 3a 20   ot Found ..Date: 
0060  4d 6f 6e 2c 20 31 37 20  4e 6f 76 20 32 30 31 34   Mon, 17  Nov 2014
```

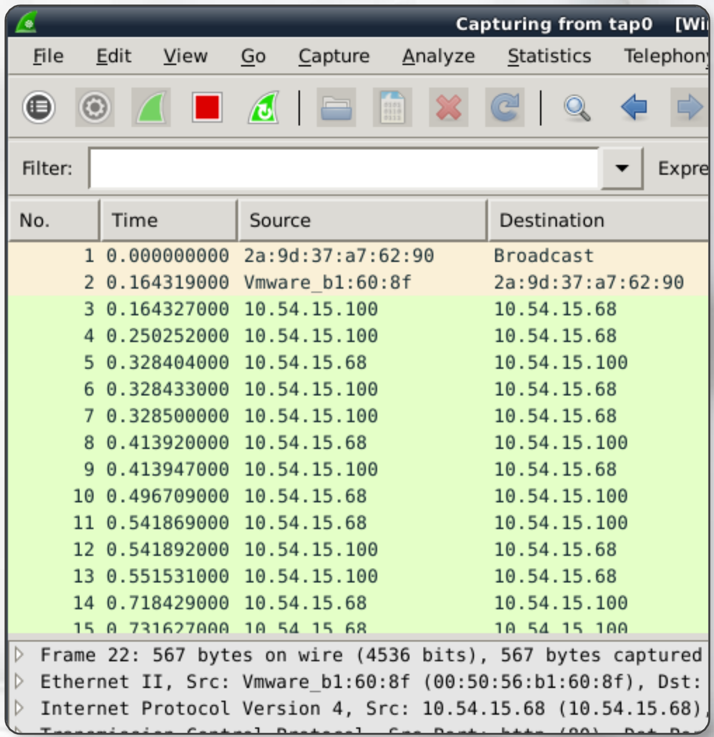
Frame (567 bytes) | Uncompressed entity body (286 bytes)

tap0: <live capture in progress> Fi... | Packets: 32 · Displayed... | Profile: Default

2.8.3 The capture window

The first two columns of the upper pane contain:

- The **number** of the captured packet
- The **arrival time** of the packet in seconds. The arrival time is relative to the start of the capture.



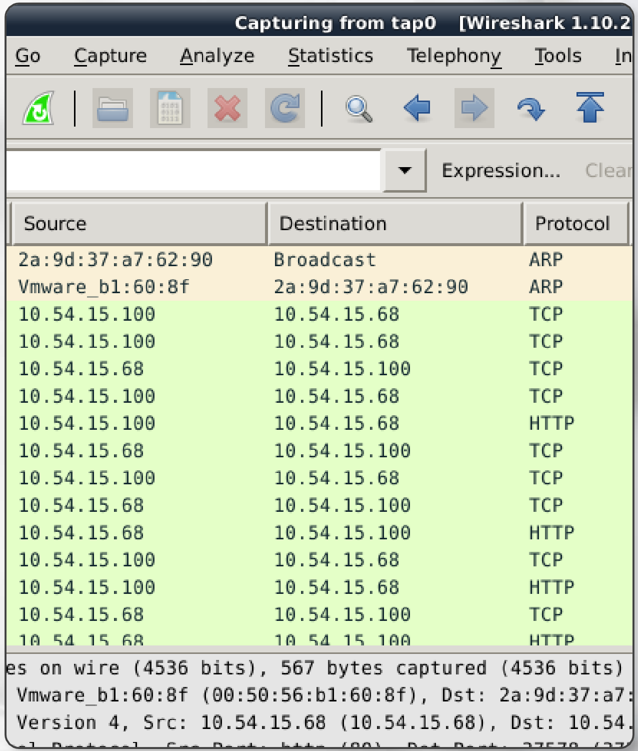
```
No.    Time          Source               Destination
     1 0.000000000   2a:9d:37:a7:62:90    Broadcast
     2 0.164319000   Vmware_b1:60:8f      2a:9d:37:a7:62:90
     3 0.164327000   10.54.15.100         10.54.15.68
     4 0.250252000   10.54.15.100         10.54.15.68
     5 0.328404000   10.54.15.68          10.54.15.100
     6 0.328433000   10.54.15.100         10.54.15.68
     7 0.328500000   10.54.15.100         10.54.15.68
     8 0.413920000   10.54.15.68          10.54.15.100
     9 0.413947000   10.54.15.100         10.54.15.68
    10 0.496709000   10.54.15.68          10.54.15.100
    11 0.541869000   10.54.15.68          10.54.15.100
    12 0.541892000   10.54.15.100         10.54.15.68
    13 0.551531000   10.54.15.100         10.54.15.68
    14 0.718429000   10.54.15.68          10.54.15.100
    15 0.731627000   10.54.15.68          10.54.15.100
```

```
▷ Frame 22: 567 bytes on wire (4536 bits), 567 bytes captured
▷ Ethernet II, Src: Vmware_b1:60:8f (00:50:56:b1:60:8f), Dst:
▷ Internet Protocol Version 4, Src: 10.54.15.68 (10.54.15.68),
▷ Transmission Control Protocol, Src Port: http (80), Dst Po
```

2.8.3 The capture window

You can then see the **source, destination** and **protocol** columns.

Note how the source and destination address vary according to the protocol.


Capturing from tap0 [Wireshark 1.10.2
Go Capture Analyze Statistics Telephony Tools In
Expression... Clear

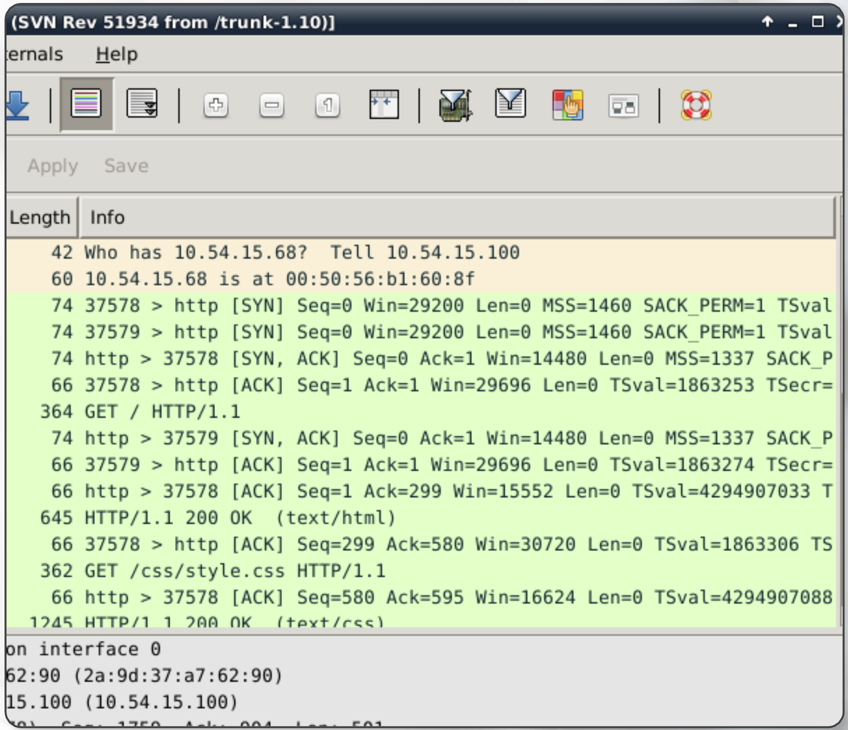
Source	Destination	Protocol
2a:9d:37:a7:62:90	Broadcast	ARP
Vmware_b1:60:8f	2a:9d:37:a7:62:90	ARP
10.54.15.100	10.54.15.68	TCP
10.54.15.100	10.54.15.68	TCP
10.54.15.68	10.54.15.100	TCP
10.54.15.100	10.54.15.68	TCP
10.54.15.100	10.54.15.68	HTTP
10.54.15.68	10.54.15.100	TCP
10.54.15.100	10.54.15.68	TCP
10.54.15.68	10.54.15.100	TCP
10.54.15.68	10.54.15.100	HTTP
10.54.15.100	10.54.15.68	TCP
10.54.15.100	10.54.15.68	HTTP
10.54.15.68	10.54.15.100	TCP
10.54.15.68	10.54.15.100	HTTP

```
es on wire (4536 bits), 567 bytes captured (4536 bits)
 Vmware_b1:60:8f (00:50:56:b1:60:8f), Dst: 2a:9d:37:a7:
 Version 4, Src: 10.54.15.68 (10.54.15.68), Dst: 10.54.
```

2.8.3 The capture window

In the last two columns, you can find the **size of the packet** and some related **information**.

The *info* column is protocol specific.

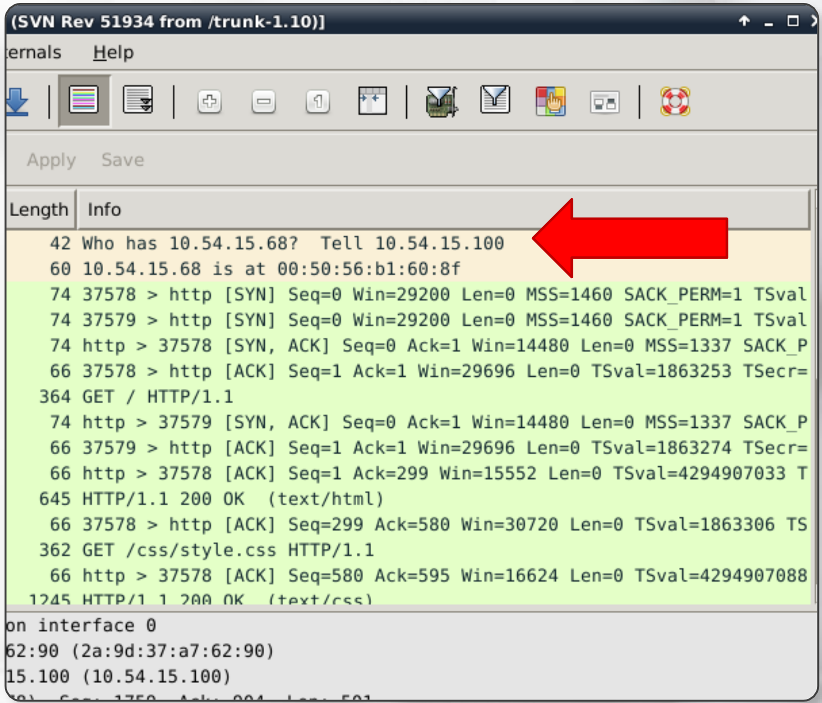


```
 42 Who has 10.54.15.68?  Tell 10.54.15.100
 60 10.54.15.68 is at 00:50:56:b1:60:8f
 74 37578 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
 74 37579 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
 74 http > 37578 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
 66 37578 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863253 TSecr=
364 GET / HTTP/1.1
 74 http > 37579 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
 66 37579 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863274 TSecr=
 66 http > 37578 [ACK] Seq=1 Ack=299 Win=15552 Len=0 TSval=4294907033 T
645 HTTP/1.1 200 OK  (text/html)
 66 37578 > http [ACK] Seq=299 Ack=580 Win=30720 Len=0 TSval=1863306 TS
362 GET /css/style.css HTTP/1.1
 66 http > 37578 [ACK] Seq=580 Ack=595 Win=16624 Len=0 TSval=4294907088
1245 HTTP/1.1 200 OK  (text/css)

on interface 0
62:90 (2a:9d:37:a7:62:90)
15.100 (10.54.15.100)
```

2.8.3 The capture window

For example, the first two packets are ARP requests and replies.

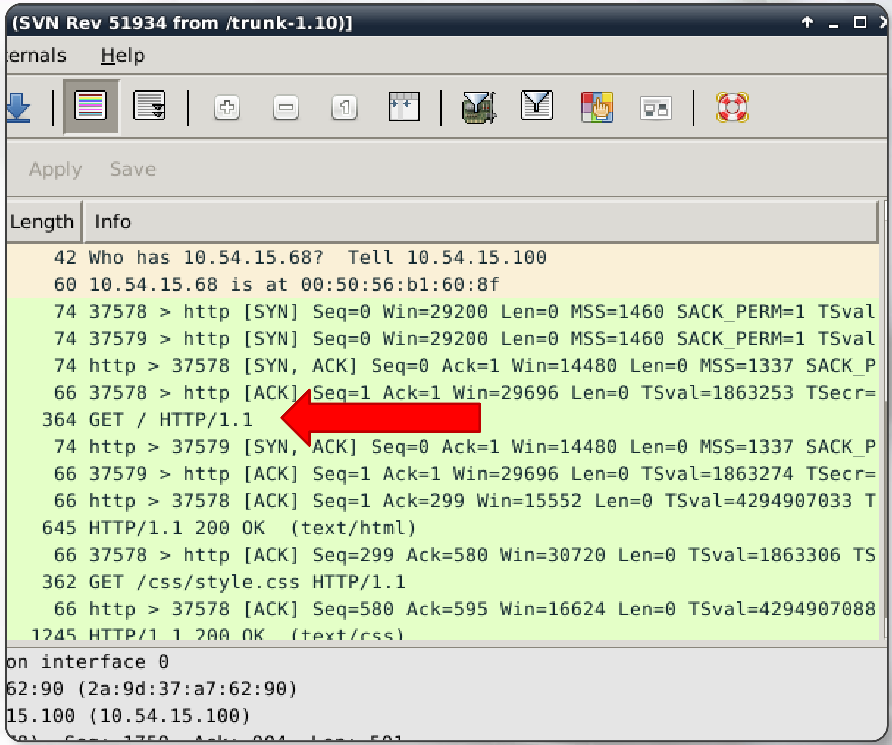


```
  42 Who has 10.54.15.68?  Tell 10.54.15.100
  60 10.54.15.68 is at 00:50:56:b1:60:8f
  74 37578 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
  74 37579 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
  74 http > 37578 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
  66 37578 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863253 TSecr=
 364 GET / HTTP/1.1
  74 http > 37579 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
  66 37579 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863274 TSecr=
  66 http > 37578 [ACK] Seq=1 Ack=299 Win=15552 Len=0 TSval=4294907033 T
 645 HTTP/1.1 200 OK  (text/html)
  66 37578 > http [ACK] Seq=299 Ack=580 Win=30720 Len=0 TSval=1863306 TS
 362 GET /css/style.css HTTP/1.1
  66 http > 37578 [ACK] Seq=580 Ack=595 Win=16624 Len=0 TSval=4294907088
1245 HTTP/1.1 200 OK  (text/css)
```

```
on interface 0
62:90 (2a:9d:37:a7:62:90)
15.100 (10.54.15.100)
```

2.8.3 The capture window

This packet is an HTTP request.


(SVN Rev 51934 from /trunk-1.10)]

ernals Help

Apply Save

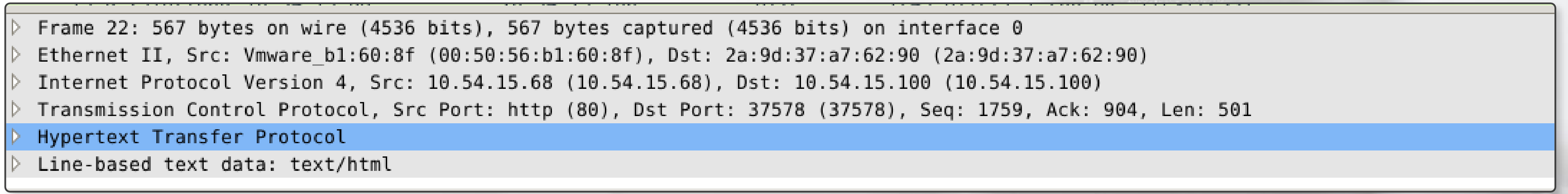
```
Length Info
    42 Who has 10.54.15.68?  Tell 10.54.15.100
    60 10.54.15.68 is at 00:50:56:b1:60:8f
    74 37578 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
    74 37579 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
    74 http > 37578 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
    66 37578 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863253 TSecr=
   364 GET / HTTP/1.1
    74 http > 37579 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
    66 37579 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863274 TSecr=
    66 http > 37578 [ACK] Seq=1 Ack=299 Win=15552 Len=0 TSval=4294907033 T
   645 HTTP/1.1 200 OK  (text/html)
    66 37578 > http [ACK] Seq=299 Ack=580 Win=30720 Len=0 TSval=1863306 TS
   362 GET /css/style.css HTTP/1.1
    66 http > 37578 [ACK] Seq=580 Ack=595 Win=16624 Len=0 TSval=4294907088
  1245 HTTP/1.1 200 OK  (text/css)
```

```
on interface 0
62:90 (2a:9d:37:a7:62:90)
15.100 (10.54.15.100)
```

2.8.3 The capture window

The center pane gives you access to all the protocol layers used by a packet.

This actually allows you to read the entire packet layer by layer!

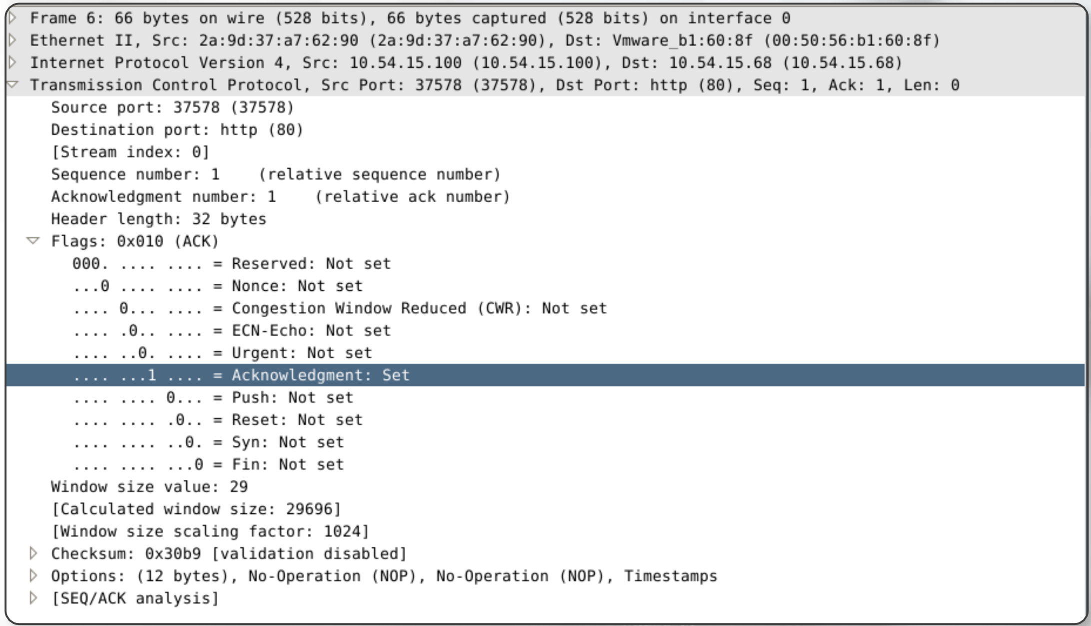


```
▷ Frame 22: 567 bytes on wire (4536 bits), 567 bytes captured (4536 bits) on interface 0
▷ Ethernet II, Src: Vmware_b1:60:8f (00:50:56:b1:60:8f), Dst: 2a:9d:37:a7:62:90 (2a:9d:37:a7:62:90)
▷ Internet Protocol Version 4, Src: 10.54.15.68 (10.54.15.68), Dst: 10.54.15.100 (10.54.15.100)
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 37578 (37578), Seq: 1759, Ack: 904, Len: 501
▷ Hypertext Transfer Protocol
▷ Line-based text data: text/html
```

2.8.3 The capture window

You can drill down to get any information you want about a packet.

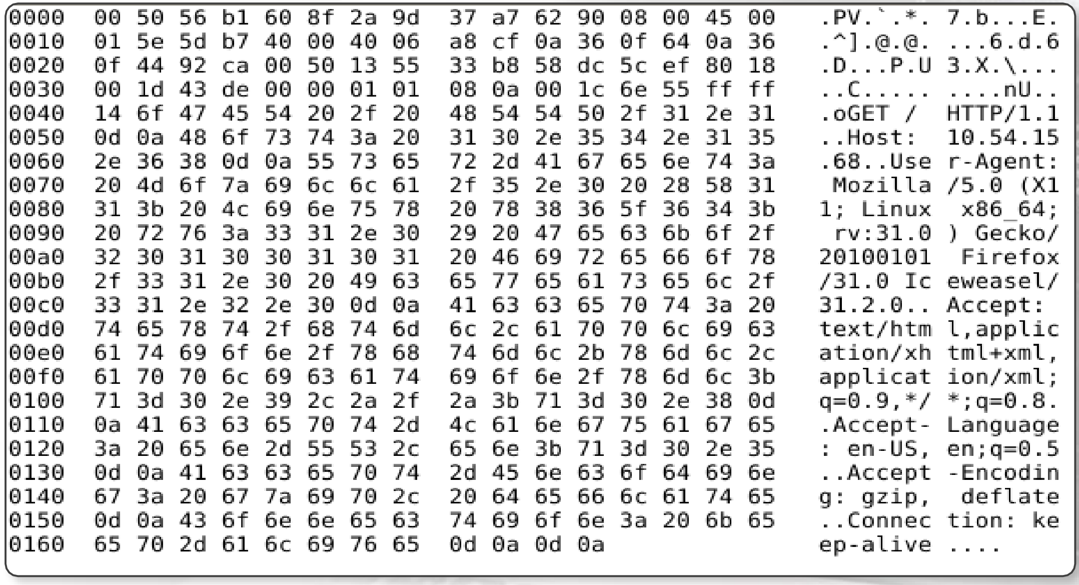
For example, this packet has the *ACK* TCP flag on.



```
▷ Frame 6: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▷ Ethernet II, Src: 2a:9d:37:a7:62:90 (2a:9d:37:a7:62:90), Dst: Vmware_b1:60:8f (00:50:56:b1:60:8f)
▷ Internet Protocol Version 4, Src: 10.54.15.100 (10.54.15.100), Dst: 10.54.15.68 (10.54.15.68)
▽ Transmission Control Protocol, Src Port: 37578 (37578), Dst Port: http (80), Seq: 1, Ack: 1, Len: 0
    Source port: 37578 (37578)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    Acknowledgment number: 1    (relative ack number)
    Header length: 32 bytes
  ▽ 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
    Window size value: 29
    [Calculated window size: 29696]
    [Window size scaling factor: 1024]
  ▷ Checksum: 0x30b9 [validation disabled]
  ▷ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▷ [SEQ/ACK analysis]
```

2.8.3 The capture window

In the bottom pane, you can see the actual packet payload. In this example we see an HTTP GET request.



```
0000  00 50 56 b1 60 8f 2a 9d  37 a7 62 90 08 00 45 00   .PV.`.*. 7.b...E.
0010  01 5e 5d b7 40 00 40 06  a8 cf 0a 36 0f 64 0a 36   .^].@.@. ...6.d.6
0020  0f 44 92 ca 00 50 13 55  33 b8 58 dc 5c ef 80 18   .D...P.U 3.X.\...
0030  00 1d 43 de 00 00 01 01  08 0a 00 1c 6e 55 ff ff   ..C..... ....nU..
0040  14 6f 47 45 54 20 2f 20  48 54 54 50 2f 31 2e 31   .oGET /  HTTP/1.1
0050  0d 0a 48 6f 73 74 3a 20  31 30 2e 35 34 2e 31 35   ..Host:  10.54.15
0060  2e 36 38 0d 0a 55 73 65  72 2d 41 67 65 6e 74 3a   .68..Use r-Agent:
0070  20 4d 6f 7a 69 6c 6c 61  2f 35 2e 30 20 28 58 31    Mozilla /5.0 (X1
0080  31 3b 20 4c 69 6e 75 78  20 78 38 36 5f 36 34 3b   1; Linux  x86_64;
0090  20 72 76 3a 33 31 2e 30  29 20 47 65 63 6b 6f 2f    rv:31.0 ) Gecko/
00a0  32 30 31 30 30 31 30 31  20 46 69 72 65 66 6f 78   20100101  Firefox
00b0  2f 33 31 2e 30 20 49 63  65 77 65 61 73 65 6c 2f   /31.0 Ic eweasel/
00c0  33 31 2e 32 2e 30 0d 0a  41 63 63 65 70 74 3a 20   31.2.0.. Accept: 
00d0  74 65 78 74 2f 68 74 6d  6c 2c 61 70 70 6c 69 63   text/htm l,applic
00e0  61 74 69 6f 6e 2f 78 68  74 6d 6c 2b 78 6d 6c 2c   ation/xh tml+xml,
00f0  61 70 70 6c 69 63 61 74  69 6f 6e 2f 78 6d 6c 3b   applicat ion/xml;
0100  71 3d 30 2e 39 2c 2a 2f  2a 3b 71 3d 30 2e 38 0d   q=0.9,*/ *;q=0.8.
0110  0a 41 63 63 65 70 74 2d  4c 61 6e 67 75 61 67 65   .Accept- Language
0120  3a 20 65 6e 2d 55 53 2c  65 6e 3b 71 3d 30 2e 35   : en-US, en;q=0.5
0130  0d 0a 41 63 63 65 70 74  2d 45 6e 63 6f 64 69 6e   ..Accept -Encodin
0140  67 3a 20 67 7a 69 70 2c  20 64 65 66 6c 61 74 65   g: gzip,  deflate
0150  0d 0a 43 6f 6e 6e 65 63  74 69 6f 6e 3a 20 6b 65   ..Connec tion: ke
0160  65 70 2d 61 6c 69 76 65  0d 0a 0d 0a               ep-alive ....
```

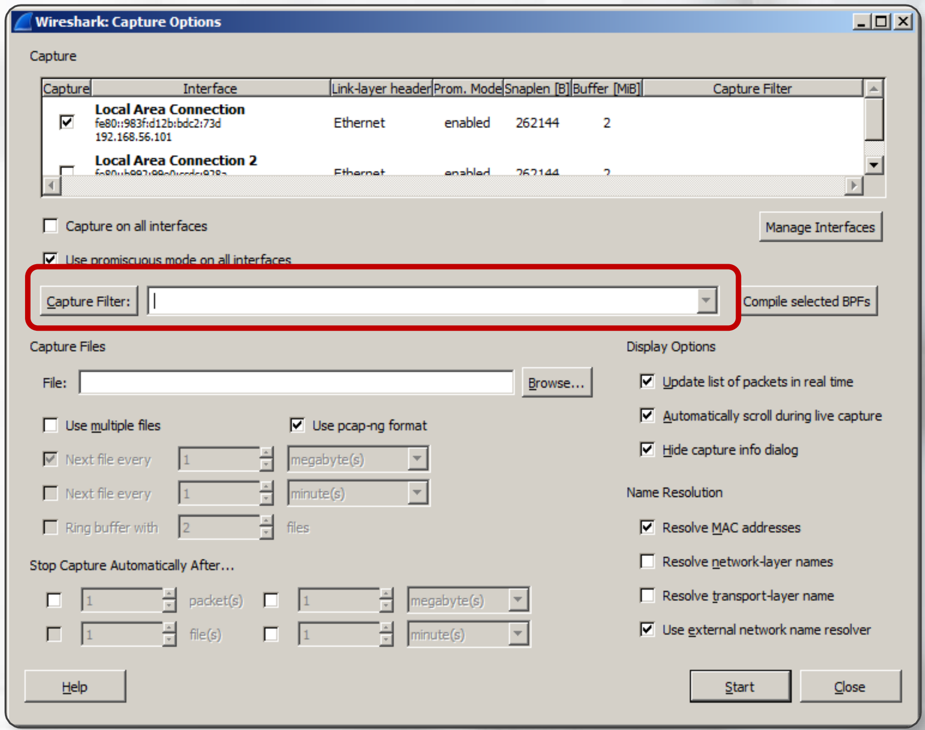
2.8.4 Filtering

A traffic capture can be overwhelming, even on a network with just a couple of dozens of nodes.

Wireshark can **filter** traffic at **capture** or at **display** time. Each method has its own pros and cons.

2.8.4.1 Capture Filters

You can set capture filters **before** starting the capture so that Wireshark will capture only packets matching the filters.


Wireshark: Capture Options

Capture

Capture	Interface	Link-layer header	Prom. Mode	Snaplen [B]	Buffer [MiB]	Capture Filter
✓	**Local Area Connection** fe80::983f:d12b:bdc2:73d 192.168.56.101	Ethernet	enabled	262144	2	
☐	**Local Area Connection 2**	Ethernet	enabled	262144	2	

☐ Capture on all interfaces

✓ Use promiscuous mode on all interfaces

Capture Filter:

Capture Files

File:

☐ Use multiple files

✓ Use pcap-ng format

✓ Next file every 1 megabyte(s)

☐ Next file every 1 minute(s)

☐ Ring buffer with 2 files

Stop Capture Automatically After...

☐ 1 packet(s) ☐ 1 megabyte(s)

☐ 1 file(s) ☐ 1 minute(s)

Display Options

✓ Update list of packets in real time

✓ Automatically scroll during live capture

✓ Hide capture info dialog

Name Resolution

✓ Resolve MAC addresses

☐ Resolve network-layer names

☐ Resolve transport-layer name

✓ Use external network name resolver

2.8.4.1 Capture Filters

Here are some basic capture filters.

Syntax	Description
ip	Only packets using IP as layer 3 protocol.
not ip	The opposite of the previous syntax.
tcp port 80	Packets where the source or destination TCP port is 80.
net 192.168.54.0/24	Packets from and to the specified network.
src port 1234	The source port must be 1234; the transport protocol does not matter.
src net 192.168.1.0/24	The source IP address must be in the specified network.
host 192.168.45.65	All the packets from or to the specified host.
host www.examplehost.com	All the packets from or to the specified hostname.

2.8.4.1 Capture Filters

Capture filters will downsize the amount of traffic gathered.

The final capture will be **smaller**, and it will contain **only** the **needed traffic**.

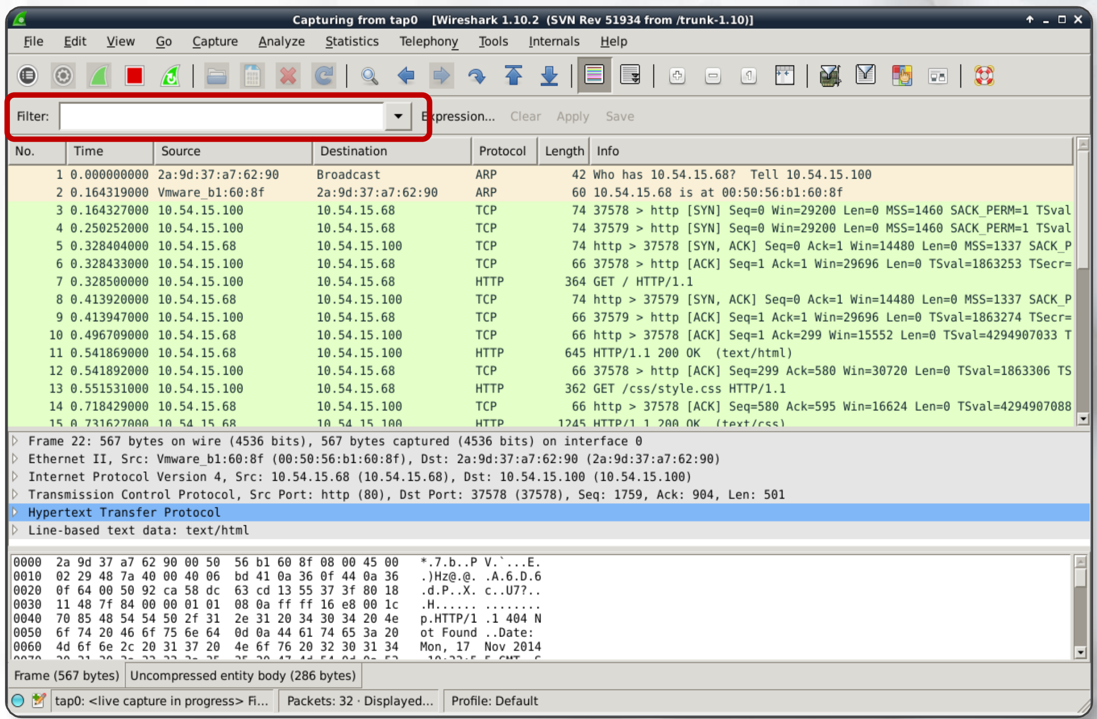
2.8.4.2 Display Filters

However, capture filters might not catch interesting traffic! Display filters instead allow you to inspect and apply very granular filters to every field of the captured packets. Wireshark then displays only the packets matching the filters.

You can always remove or fine tune a display filter, something you can’t do with the capture filter (you would have to re-start the capture from scratch).

2.8.4.2 Display Filters

You can use the filter textbox to apply a display filter.


Capturing from tap0 [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time        Source             Destination        Protocol  Length Info
   1 0.000000000 2a:9d:37:a7:62:90  Broadcast          ARP           42 Who has 10.54.15.68?  Tell 10.54.15.100
   2 0.164319000 Vmware_b1:60:8f    2a:9d:37:a7:62:90  ARP           60 10.54.15.68 is at 00:50:56:b1:60:8f
   3 0.164327000 10.54.15.100       10.54.15.68        TCP           74 37578 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
   4 0.250252000 10.54.15.100       10.54.15.68        TCP           74 37579 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval
   5 0.328404000 10.54.15.68        10.54.15.100       TCP           74 http > 37578 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
   6 0.328433000 10.54.15.100       10.54.15.68        TCP           66 37578 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863253 TSecr=
   7 0.328500000 10.54.15.100       10.54.15.68        HTTP         364 GET / HTTP/1.1
   8 0.413920000 10.54.15.68        10.54.15.100       TCP           74 http > 37579 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1337 SACK_P
   9 0.413947000 10.54.15.100       10.54.15.68        TCP           66 37579 > http [ACK] Seq=1 Ack=1 Win=29696 Len=0 TSval=1863274 TSecr=
  10 0.496709000 10.54.15.68        10.54.15.100       TCP           66 http > 37578 [ACK] Seq=1 Ack=299 Win=15552 Len=0 TSval=4294907033 T
  11 0.541869000 10.54.15.68        10.54.15.100       HTTP         645 HTTP/1.1 200 OK  (text/html)
  12 0.541892000 10.54.15.100       10.54.15.68        TCP           66 37578 > http [ACK] Seq=299 Ack=580 Win=30720 Len=0 TSval=1863306 TS
  13 0.551531000 10.54.15.100       10.54.15.68        HTTP         362 GET /css/style.css HTTP/1.1
  14 0.718429000 10.54.15.68        10.54.15.100       TCP           66 http > 37578 [ACK] Seq=580 Ack=595 Win=16624 Len=0 TSval=4294907088
  15 0.731627000 10.54.15.68        10.54.15.100       HTTP        1245 HTTP/1.1 200 OK  (text/css)
```

```
▷ Frame 22: 567 bytes on wire (4536 bits), 567 bytes captured (4536 bits) on interface 0
▷ Ethernet II, Src: Vmware_b1:60:8f (00:50:56:b1:60:8f), Dst: 2a:9d:37:a7:62:90 (2a:9d:37:a7:62:90)
▷ Internet Protocol Version 4, Src: 10.54.15.68 (10.54.15.68), Dst: 10.54.15.100 (10.54.15.100)
▷ Transmission Control Protocol, Src Port: http (80), Dst Port: 37578 (37578), Seq: 1759, Ack: 904, Len: 501
▷ Hypertext Transfer Protocol
▷ Line-based text data: text/html
```

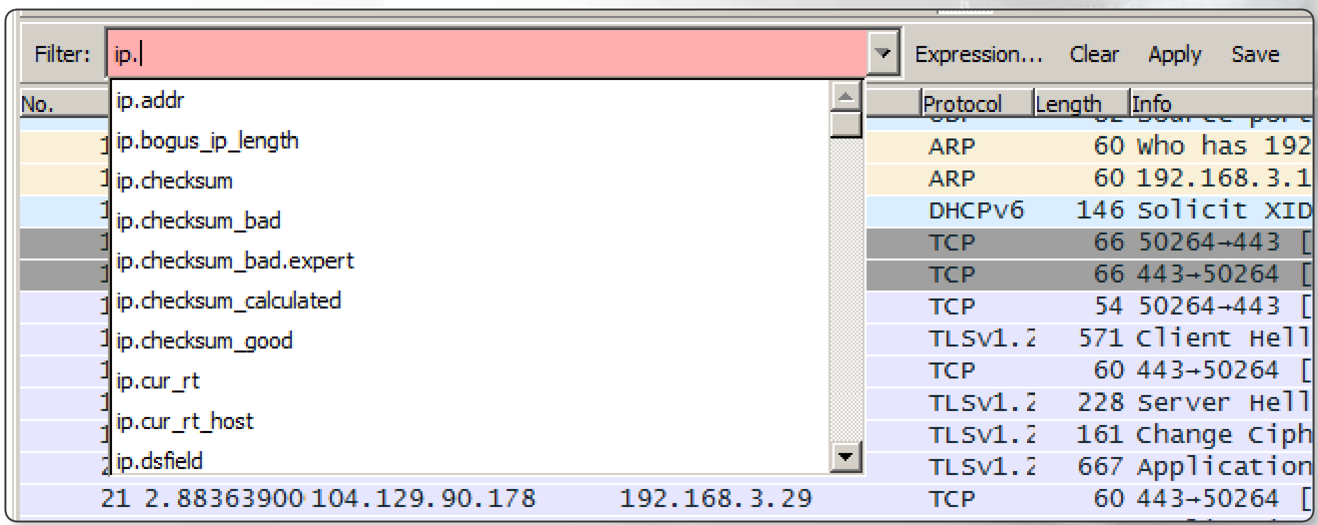
```
0000  2a 9d 37 a7 62 90 00 50  56 b1 60 8f 08 00 45 00   *.7.b..P V.`...E.
0010  02 29 48 7a 40 00 40 06  bd 41 0a 36 0f 44 0a 36   .)Hz@.@. .A.6.D.6
0020  0f 64 00 50 92 ca 58 dc  63 cd 13 55 37 3f 80 18   .d.P..X. c..U7?..
0030  11 48 7f 84 00 00 01 01  08 0a ff ff 16 e8 00 1c   .H...... ........
0040  70 85 48 54 54 50 2f 31  2e 31 20 34 30 34 20 4e   p.HTTP/1 .1 404 N
0050  6f 74 20 46 6f 75 6e 64  0d 0a 44 61 74 65 3a 20   ot Found ..Date: 
0060  4d 6f 6e 2c 20 31 37 20  4e 6f 76 20 32 30 31 34   Mon, 17  Nov 2014
```

Frame (567 bytes) Uncompressed entity body (286 bytes)

tap0: <live capture in progress> Fi... Packets: 32 · Displayed... Profile: Default

2.8.4.2 Display Filters

You can start typing a filter and Wireshark will give you valid protocol fields.

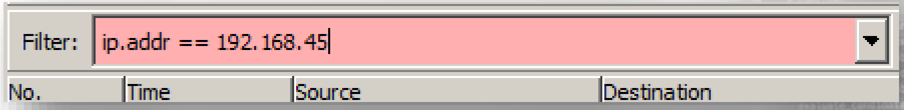


```
ip.addr
ip.bogus_ip_length
ip.checksum
ip.checksum_bad
ip.checksum_bad.expert
ip.checksum_calculated
ip.checksum_good
ip.cur_rt
ip.cur_rt_host
ip.dsfield
```

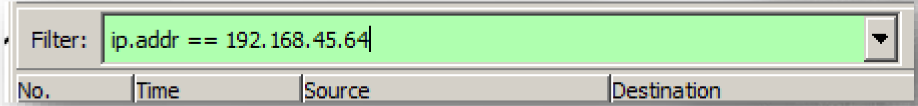
```
ARP       60 Who has 192
ARP       60 192.168.3.1
DHCPv6   146 Solicit XID
TCP       66 50264→443 [
TCP       66 443→50264 [
TCP       54 50264→443 [
TLSv1.2  571 Client Hell
TCP       60 443→50264 [
TLSv1.2  228 Server Hell
TLSv1.2  161 Change Ciph
TLSv1.2  667 Application
21 2.88363900 104.129.90.178     192.168.3.29     TCP       60 443→50264 [
```

2.8.4.2 Display Filters

The background of the text-box will turn red if the filter is invalid or green when the filter is valid.



```
Filter: ip.addr == 192.168.45
No.    Time    Source    Destination
```



```
Filter: ip.addr == 192.168.45.64
No.    Time    Source    Destination
```

2.8.4.2 Display Filters

A display filter is made by:

Syntax	Description
<Protocolname>	Displays any packet using that protocol.
<Protocolname>[.field]	Displays any packet with the specified field present.
<Protocolname>[.field] [operand value]	Displays any packet whose protocol field matches the operand and value.
<Protocolname>[.field] AND <Protocolname>[.field] [operand value]	You can combine multiple expressions by using logical operators.

2.8.4.2 Display Filters

Below is an example.

Syntax	Description
ip	Displays IP packets.
ip.addr	Displays IP packets with a populated source or destination address.
ip.addr == 192.168.12.13	Displays IP packets with 192.168.12.13 as source or destination address.
ip.addr == 192.168.12.13 or arp	The above or ARP packets.

2.8.4.2 Display Filters

You can find Wireshark display filter reference here.

For any other information, please refer to the Wireshark User's Guide.

https://www.wireshark.org/docs/dfref/
https://www.wireshark.org/docs/wsug_html_chunked/

2.8.5 Video – Using Wireshark

Using Wireshark

In this video, you will see how to configure and use Wireshark, how capture and display filters work and this powerful tool's main features.

**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.*

2.8.6 Video – Full Stack Analysis with Wireshark

Full Stack Analysis with Wireshark

This video uses the traffic captured in the previous video to perform traffic analysis, which will improve your understanding of low-level interaction between routing, IP, ARP, TCP, application-level protocols and more!

**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.*

2.8.7 Sample Traffic Captures

If you want to practice these topics a little more, you can record some traffic from your computer or you can download a capture from Wireshark website.

http://wiki.wireshark.org/SampleCaptures

2.8.8 Lab – Data Exfiltration

Apply your fresh knowledge about networking to the Data Exfiltration lab.

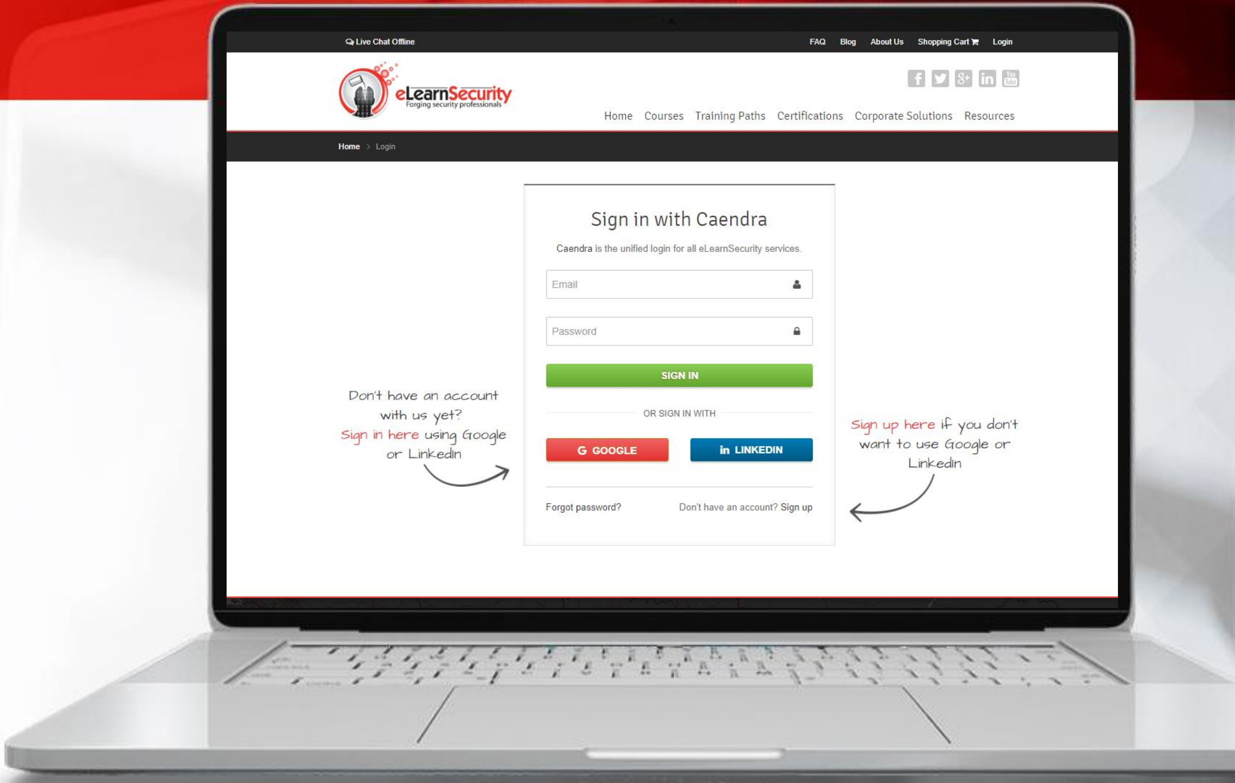
You can test it against a environment secured by a firewall, while on the other hand, learn how to bypass this kind of obstacles.

2.8.8 Lab – Data Exfiltration

Data Exfiltration

The environment is secured by a firewall.

Can you bypass it?


Live Chat Offline
FAQ Blog About Us Shopping Cart Login
eLearnSecurity Forging security professionals
Home Courses Training Paths Certifications Corporate Solutions Resources
Home > Login
Sign in with Caendra
Caendra is the unified login for all eLearnSecurity services.
Email
Password
SIGN IN
OR SIGN IN WITH
GOOGLE
LINKEDIN
Forgot password?
Don't have an account? Sign up
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

Packet Analysis Reference Guide v3.0

You can download it from the **Resources** drop-down menu of the Networking module (Section Preliminary Skills – Prerequisites)

RFC 5735 – Special Use IPv4 Addresses

http://tools.ietf.org/html/rfc5735

Classful Subnet Calculator

http://www.subnet-calculator.com/

Service Name and Transport Protocol Port Number Registry

http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

References

ISO/OSI Model

http://support.microsoft.com/kb/103884

Variable Length Subnet Table For IPv4

http://tools.ietf.org/html/rfc1878

CIDR Subnet Calculator

http://www.subnet-calculator.com/cidr.php

TCPView

http://technet.microsoft.com/en-us/sysinternals/bb897437

References

TACO IPv6 Router - a Case Study in Protocol Processor Design

https://www.researchgate.net/publication/31596630_TACO_IPv6_Router_-_a_Case_Study_in_Protocol_Processor_Design

Internet Protocol Version 6 (IPv6) Addressing Architecture

https://tools.ietf.org/html/rfc3513

IPv6 Explained for Beginners

http://www.steves-internet-guide.com/ipv6-guide/

How to find IPv6 Prefix

https://networklessons.com/ipv6/how-to-find-ipv6-prefix/

References

IP Layer Network Administration with Linux

You can download it from the **Resources** drop-down menu of the Networking module (Section Preliminary Skills – Prerequisites)

The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference

http://www.amazon.com/TCP-Guide-Comprehensive-Illustrated-Protocols/dp/159327047X/ref=sr_1_3?s=books&ie=UTF8&qid=1297880998&sr=1-3

Firewall Fundamentals

http://www.amazon.com/Firewall-Fundamentals-Wes-Noonan/dp/1587052210/ref=cm_cr_pr_sims_t

Wireshark Display Filter Reference

https://www.wireshark.org/docs/dfref/

References

TCP/IP Tutorial and Technical Overview

You can download it from the **Resources** drop-down menu of the Networking module (Section Preliminary Skills – Prerequisites)

IPTables tutorial

https://www.frozentux.net/iptables-tutorial/iptables-tutorial.html

Wireshark

https://www.wireshark.org/

Wireshark User's Guide

https://www.wireshark.org/docs/wsug_html_chunked/

References

DNS Concepts and Facilities

https://www.ietf.org/rfc/rfc1034.txt

Wireshark Sample Captures

http://wiki.wireshark.org/SampleCaptures

DNS Implementation and Specification

https://www.ietf.org/rfc/rfc1035.txt

IPv6 Calculator

https://www.ultratools.com/tools/ipv6CIDRToRange

References

Netfilter

http://www.netfilter.org/

Ossec IDS

http://www.ossec.net/

FAQ: Network Intrusion Detection Systems

http://www.linuxsecurity.com/resource_files/intrusion_detection/network-intrusion-detection.html

Videos

Using Wireshark

In this video, you will see how to configure and use Wireshark, how capture and display filters work and this powerful tool's main features.

Full Stack Analysis with Wireshark

This video uses the traffic captured in the previous video to perform traffic analysis, which will improve your understanding of low-level interaction between routing, IP, ARP, TCP, application-level protocols and more!

*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

Find the Secret Server

Use your TCP/IP routing knowledge to access a secret web server.

Data Exfiltration

The environment is secured by a firewall. Can you bypass it?

**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.*