



PART 1
CYBERSECURITY FOUNDATIONS

FROM BITS TO SECURITY

A Practical Journey into Cybersecurity

ZOHAIB HAYAT

CYBER SECURITY LEARNER

About the Author

Zohaib Hayat is a Cyber Security Learner with a growing interest in cloud security, and digital forensics. His journey into cybersecurity began with a simple curiosity about what happens behind the screen whenever a user opens a website, sends a message, or connects to the Internet.

As he continued learning, that curiosity led him into networking, Linux, web security, cloud computing, AWS, security tools, and digital forensics. Along the way, he discovered that many cybersecurity concepts that seem complicated at first become much easier when they are explained from the fundamentals and connected to real-world examples.

From Bits to Security is the result of that learning journey.

Rather than presenting cybersecurity as something that can only be understood by experts, this book is written for students, beginners, and curious learners who want to understand how computers, networks, websites, and cloud systems work before diving deeper into cybersecurity.

The book follows a simple philosophy:

“Understand how the system works before trying to understand how to secure it.”

Through simple explanations, practical examples, diagrams, and a gradual learning path, Zohaib hopes to help readers take their first steps into the world of cybersecurity and discover which part of the field interests them most.

This book is not the end of the journey. It is the beginning.

Author's Note

Dear Reader,

Thank you for picking up this book.

I am not writing this as an expert looking down at a subject. I am writing it as someone who is still climbing the same hill you may be standing at the bottom of right now. Every explanation in these pages is something I first had to explain to myself, often more than once, before it finally made sense.

I wrote *From Bits to Security* because I wished a book like this had existed when I started. Most resources either assumed too much background knowledge or jumped straight into tools and attacks without explaining what was actually happening underneath. I wanted to build the foundation first — the computer, the network, the request, the response — before ever talking about how any of it could be attacked or defended.

This is Part 1 of a series. It covers the foundations: how computers work, how the internet really moves data, and what happens in the seconds between opening a browser and seeing a webpage appear. The parts that follow will move closer to security itself — web applications, common vulnerabilities, and the practical side of this field.

If something in this book is unclear, incomplete, or simply wrong, I want to know. This book will keep improving with every edition, and honest feedback from readers like you is the main way that happens. My contact details are on the copyright page — please do reach out.

I hope this book gives you the same thing it gave me while I was writing it: the confidence that this field is learnable, one layer at a time.

— Zohaib Hayat

From Bits to Security

A Practical Journey into Cybersecurity — Part 1

Copyright © 2026 Zohaib Hayat

All rights reserved.

No part of this publication may be reproduced, distributed, or transmitted in any form or by any means, including photocopying, recording, or other electronic or mechanical methods, without the prior written permission of the author, except in the case of brief quotations used in reviews and certain other noncommercial uses permitted by copyright law.

First Edition — 2026

This is Part 1 of the cybersecurity fundamentals series.

Contact

Email: hayatzohaib443@gmail.com

LinkedIn: Zohaib Hayat

Phone: +92 327-1172443

Release Notes

First Edition — Part 1

Released 2026

What's in This Release

This is the first published edition of Part 1, covering the foundations of how computers, networks, and the internet work — from bits and bytes up through TCP/IP, DNS, TLS/HTTPS, and the complete journey of opening a website.

What's Coming Next

Part 1 is the beginning of a planned multi-part series. The parts that follow will build on this foundation and move progressively into web application mechanics, common vulnerabilities, hands-on practice, and eventually applied cybersecurity topics.

Corrections and Feedback

This book will continue to be revised as errors are found and as clearer ways of explaining a concept are discovered. If you spot a mistake, find a diagram confusing, or think something could be explained better, please reach out using the contact details on the copyright page. Reader feedback directly shapes future editions and future parts of this series.

Version History

v1.0 — First public release of Part 1 (2026).

Table of Contents

FRONT MATTER

About the Author.....	i
Author's Note.....	ii
Copyright.....	iii
Release Notes.....	iv

PART 1 — WELCOME TO THE DIGITAL WORLD

Chapter 1 — Your Digital Life Is Not as Simple as It Looks.....	1
What actually happens when you open Instagram, YouTube, or Google?.....	1
Where does your data go?.....	1
What is a server?.....	1
What is the Internet, really?.....	2
Why do we need cybersecurity?.....	2
Chapter 2 — Inside Your Computer.....	3
What Is a CPU?.....	3
What Is RAM?.....	3
What is Storage?.....	3
What is an Operating System?.....	4
What is a Process?.....	4
What is a File?.....	4
What is a GPU?.....	5
What are Bits and Bytes?.....	6
Introduction to Binary.....	6
Chapter 3 — How the Internet Really Works.....	7
Client & Server.....	7
What is an IP Address?.....	7
What is a MAC Address?.....	8
What is a Port?.....	8
What is a Router?.....	9
What is a Switch?.....	9
What is a NIC?.....	10
What is a Network Packet?.....	10
What is a Frame? and Summary.....	11

Complete Travel Map.....	14
Chapter 4 — The Journey of a Single Packet.....	16
OSI Model — Real Example: Opening a Website.....	16
Layer 7 — Application Layer.....	17
Layer 6 — Presentation Layer.....	18
Layer 5 — Session Layer.....	19
Layer 4 — Transport Layer.....	20
Layer 3 — Network Layer.....	20
Layer 2 — Data Link Layer.....	21
Layer 1 — Physical Layer.....	22
Now Look at the Complete Encapsulation.....	22
But What Happens When the Frame Reaches the Router?	23
Complete Real-World Map.....	24
The One-Line Story.....	25
Encapsulation and De-encapsulation.....	27
Encapsulation.....	27
Step 1 — Application Layer.....	28
Step 2 — Presentation / Session Functions.....	28
Step 3 — Transport Layer.....	28
Step 4 — Network Layer.....	28
Step 5 — Data Link Layer.....	29
Step 6 — Physical Layer.....	29
What Happens at the Router?.....	31
De-encapsulation.....	31
Complete Journey.....	33
The Most Important Concept.....	34
How TCP and IP Work Together.....	38
1. Finding the Server.....	38
2. TCP Three-Way Handshake.....	39
Now the Actual Data Is Sent.....	40
6. What Does IP Do?.....	41
7. What Happens to the MAC Address?.....	42
8. What Does TCP Do?.....	43
9. TCP and IP Working Together.....	44

10. Complete Journey.....	45
The Complete Picture.....	47
TLS Certificates — How They Are Generated and Exchanged.....	48
1. How is a Certificate Created?.....	48
2. What Happens When the Browser Connects?.....	49
3. Does the Certificate Encrypt the Whole Connection?.....	50
4. Why Does the Server Keep the Private Key Secret?.....	51
5. Simple Real Example.....	51
In Simple Words.....	52
Chapter 5 — What Happens When You Open a Website?.....	53
1. You Enter the Website Address.....	55
2. DNS Finds the Server.....	56
3. TCP Establishes the Connection.....	56
4. TLS Secures the Connection.....	57
5. The Browser Sends an HTTP Request.....	58
6. The Data Travels Through the Network.....	59
7. The Server Receives the Request.....	61
8. The Server Processes the Request.....	61
9. The Server May Contact a Database.....	62
10. The Server Creates the Response.....	62
11. The Response Travels Back.....	63
12. The Browser Receives the Response.....	64
13. The Browser Builds the Webpage.....	65
Complete Journey.....	66
One Important Thing to Remember.....	67
In Simple Words.....	68
CLOSING	
Part 1 — What We Learned.....	69
A Message From My Experience.....	71
Think Like a Detective.....	71
Experiment.....	72
Don't Be Afraid of Technology.....	72
Don't Just Run Commands.....	72
Mistakes Are Part of Learning.....	72

Where This Journey Goes Next.....	73
The Next Part.....	74
Thank You.....	76

 PART 1 — WELCOME TO THE DIGITAL WORLD

Chapter 1 — Your Digital Life Is Not as Simple as It Looks

What actually happens when you open Instagram, YouTube, or Google?

When we turn on our phone or computer, we simply open apps like Instagram, Facebook, or YouTube. We click, scroll, upload photos, and watch videos without thinking about what happens behind the screen.

But have you ever wondered what actually happens after a single click?

When you upload a photo, where does it go? How does it reach the server? Where is it stored? How does the application know that the photo belongs to you? And when you open it again, how does that same photo appear on your screen within seconds?

For us, it is just a simple click.

Behind that click, however, is a complex digital process involving applications, networks, servers, databases, and storage systems.

Where does your data go?

When we click “Post,” have you ever wondered where the words, images, or videos we upload actually go?

That data travels through the internet and reaches a server, where it is processed and stored. Later, when we open the app again, our device sends a request through the network to retrieve that data.

The request passes through different layers of the network before reaching the server. The server then processes the request and sends the required data back through the network to our device.

What looks like a simple click is actually a complete journey of requests, data, servers, and responses happening in seconds.

What is a server?

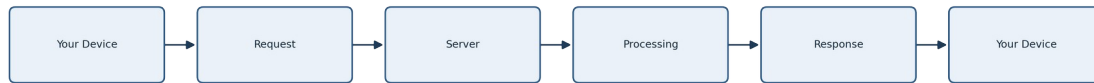
A server is a computer system that provides data, applications, or services to other devices over a network. A server is not necessarily a special kind of computer; it can be a physical computer or a virtual machine configured to provide a particular service.

For example, when you open a website, your device sends a request to the server hosting that website. The server receives the request, processes it, and sends a response back to your device. That response may contain information such as a webpage, image, video, or other data.

A server may also perform different jobs. A web server can deliver websites, a database server can store and provide access to data, and a file server can provide files to users over a network.

In simple terms:

A Simple Request



So, whenever you access an online service, a server is often working in the background to receive your request and provide the information or service you need.

What is the Internet, really?

The Internet is a global network that digitally connects people, devices, and systems around the world.

When you send a message to someone in another part of the world, the data travels through multiple networks before reaching its destination. It typically moves from your device to your router, then through your Internet Service Provider (ISP) and across interconnected (Connected ISP) networks. It may pass through several ISPs and routing systems before reaching the recipient's ISP, their router, and finally their device.

In simple terms:

How Data Travels Across the Internet



All of this happens within seconds, making communication across the world feel almost instant.

Why do we need cybersecurity?

We live in a digitally connected world where our devices, applications, networks, and data are constantly communicating with each other. Every time we send a message, upload a photo, access a website, or make an online transaction, data travels through digital systems.

But these systems can also be targeted by attackers who may try to steal, modify, destroy, or misuse information.

Cybersecurity protects these digital systems, data, and communications from unauthorized access, attacks, and misuse.

In simple words, we need cybersecurity because everything connected to the digital world needs to be protected.

Goal: Create curiosity and make the reader want to understand what is happening behind the screen.

Chapter 2 — Inside Your Computer

What Is a CPU?

The CPU (Central Processing Unit) is often called the brain of the computer because it executes instructions and processes data.

We often hear that computers work using 1s and 0s, but we do not normally see binary values on our screen. This is because computers process data at the machine level using binary, while the operating system and applications present that information to us in a human-readable form.

Whenever we open an application, type something, click a button, or perform any action, the CPU executes the instructions required to process that action.

For example, when you type a message, the system processes your input and eventually displays the characters you can understand. Behind this simple action, the CPU is continuously executing instructions and working with other components such as RAM, storage, and input/output devices.

In simple words, the CPU is the part of the computer that executes instructions and processes data, making it possible for the computer to perform tasks.

What Is RAM?

RAM (Random Access Memory) is the computer's temporary working memory. It holds the data and instructions that applications and processes currently need while they are running.

For example, when you open a browser or an app like TikTok, it uses some RAM to keep the information it needs readily available. When you close the application, the RAM it was using can be freed and used by other processes.

The more RAM a device has, the more applications and processes it can generally handle at the same time without slowing down.

For example, you might be downloading a file in one browser tab, watching a video in another, and using another application at the same time. RAM helps keep these active processes readily available so the system can handle them efficiently.

In simple words, RAM is temporary working memory that helps your device run multiple active processes smoothly.

What is Storage?

Storage is the part of a device where your data is saved for a long time. When you download a file, save a photo, install an app, or store a video, that data is saved in your device's storage.

Storage can be an HDD (Hard Disk Drive) or SSD (Solid State Drive). It is not a part of the CPU; it is a separate component connected to the computer.

RAM is temporary. When you open an app or work on something, RAM temporarily holds the data needed by that process. When you turn off your computer, the data in RAM is cleared.

Storage is different because the data saved there remains even after you turn off your computer. Your photos, videos, apps, documents, and other files stay in storage until you delete them.

In simple words, RAM temporarily holds the data you are currently working with, while storage keeps your data saved even when the device is turned off.

What is an Operating System?

An Operating System (OS) works like a manager of the computer. It helps different parts of the computer communicate and work together.

For example, when you open Chrome, the operating system manages the resources Chrome needs. It allows Chrome to use the CPU and RAM, and it manages access to data stored on the SSD or HDD.

The operating system also acts as a bridge between the user, applications, and hardware. When we click something, the operating system helps turn that action into instructions that the hardware can process.

Operating systems like Windows and Linux give us different levels of control. Linux, in particular, gives users more direct control over the system, which is one reason it is widely used by developers, system administrators, and cybersecurity professionals.

In simple words, the operating system is the manager that helps the user, applications, and hardware communicate and work together.

What is a Process?

A process is basically a task or program that is currently running on your device.

Whenever you open something, a process is created to handle that work. For example, when you open Chrome, the operating system creates and manages processes for Chrome. These processes use resources such as CPU and RAM to perform their tasks.

In the background, the operating system manages these processes and allows them to communicate with different parts of the computer.

So, whenever some work is happening on your device, the running work is basically a process.

In simple words, a process is a program or task that is currently running and using the computer's resources.

What is a File?

A file is basically a collection of data saved on a device. Whenever you save something like a photo, video, document, song, or text, it is stored as a file.

For example, when you take a photo with your phone, that photo is saved as a file in your device's storage. Similarly, when you download a PDF or save a Word document, it is also stored as a file.

A file has a name, format, and location where it is stored.

In simple words, a file is a piece of data saved in storage that you can open, use, copy, move, or delete.

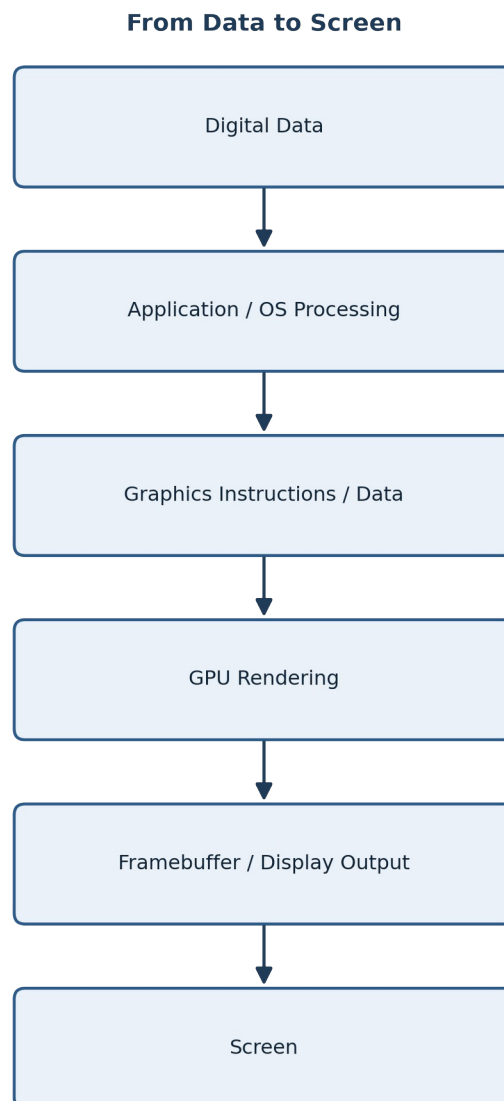
What is a GPU?

A GPU (Graphics Processing Unit) is a part of a computer that mainly processes and renders graphics and visual data.

When we open an image or watch a video, the data is stored in binary form. The computer processes this data, and the GPU helps turn the visual information into pixels and graphics that we can see on the screen.

For example, a photo may look like a simple picture to us, but inside the computer it is represented as digital data. The GPU processes the visual data and renders it into pixels, allowing us to see the actual image.

In simple words, the GPU helps turn digital data into graphics that we can see on the screen.



What are Bits and Bytes?

A bit is the smallest unit of digital data. It can have only two values: 0 or 1.

A byte is a group of 8 bits.

1 Byte = 8 Bits

Whenever we save something on a device, such as a photo, video, or document, the data is ultimately stored as binary data made of 0s and 1s in the storage.

When we open that file, the operating system and CPU process the data and make it understandable to the application. If it is an image, the GPU helps process and render the image as pixels, which are then displayed on the screen.

So, in a simple way:

Where an Image on Screen Comes From



We see a simple image or file on our screen, but behind it, the computer is working with huge amounts of bits and bytes.

Introduction to Binary

Binary is the basic language that computers use to represent and process digital data. It uses only two values: 0 and 1.

Everything we use on a computer—such as text, images, videos, files, and applications—is ultimately represented in binary form.

For us, a picture may look like an image and a message may look like simple text. But inside the computer, this information is represented using 0s and 1s.

In simple words, binary is the way computers represent digital information using 0 and 1.

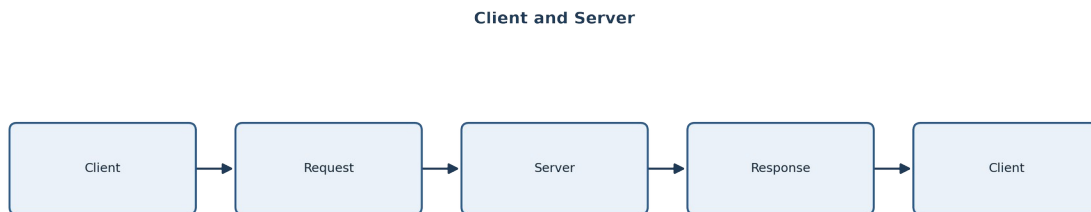
Chapter 3 — How the Internet Really Works

Client & Server

Whenever we open an app or a website on the Internet, our device usually sends a request to a server, and the server sends a response back.

For example, when we open a website in a browser, the browser acts as the client, while the server is the system that receives the request, processes it, and sends the required data back.

In simple words:



The client asks for something, and the server provides the required response.

What is an IP Address?

An IP address is like an address for a device on a network. Just like a house has an address, devices use IP addresses so that data can find the correct destination.

For example, when we open a website in a browser, we usually use its domain name instead of its IP address. The browser uses DNS (Domain Name System) to find the IP address associated with that domain.

DNS works like a phonebook of the Internet. It finds the IP address, and then the device can communicate with the destination using that address.

An IPv4 address is 32 bits, divided into 4 bytes (octets):

$$8 \text{ bits} + 8 \text{ bits} + 8 \text{ bits} + 8 \text{ bits} = 32 \text{ bits}$$

Each byte can have a value from 0 to 255.

For example:

192.168.1.10

After understanding this 32-bit structure, we can move to subnetting, where we learn how IP addresses are divided into networks and hosts.

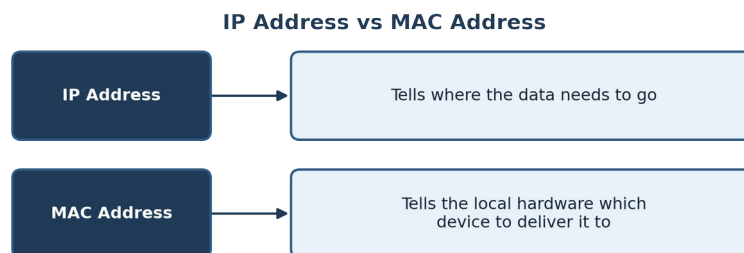
What is a MAC Address?

A MAC address is basically the physical or hardware address of a network device, while an IP address is the network-level address used to decide where the data needs to go.

The IP address helps routers decide the final destination of the data. But hardware on the local network does not directly use the IP address for physical delivery. It uses the MAC address to identify the next device.

For example, a router checks its routing table using the destination IP address and decides where the data should go next. Then, the data is sent to the next device using its MAC address. This MAC address can change at every network hop because each network has its own local devices.

In simple words:



IP helps with routing, while MAC helps with local physical delivery.

What is a Port?

A port is like a door through which network data reaches the correct application or service on a computer.

When a packet comes to your computer, it first reaches the NIC (Network Interface Card). The packet contains a port number, and the operating system checks that port number to decide which application or service should receive the packet.

For example, web traffic commonly uses port 80 for HTTP and port 443 for HTTPS.

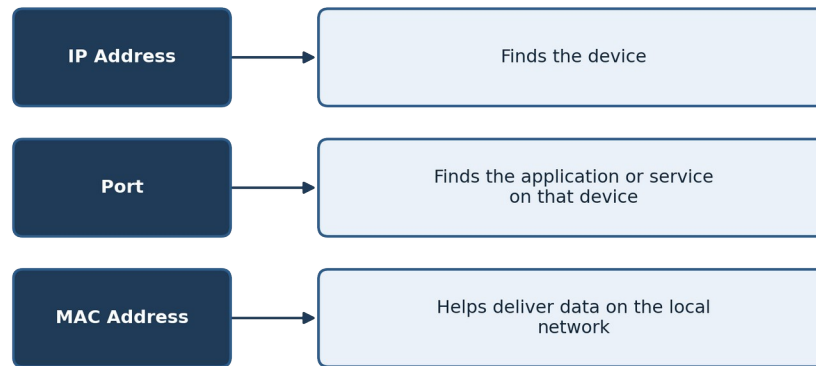
There are 65,536 possible port numbers, from 0 to 65,535.

- 0-1023 = Well-known ports → Commonly used by standard services such as HTTP (80), HTTPS (443), and SSH (22).
- 1024-49,151 = Registered ports → Can be registered for specific applications or services.
- 49,152-65,535 = Dynamic/Private ports → Usually used temporarily by applications for client-side connections.

One important thing is that these ports are not permanently locked to a service. A service can be configured to use a different port.

In simple words:

IP, Port, and MAC — Their Jobs



What is a Router?

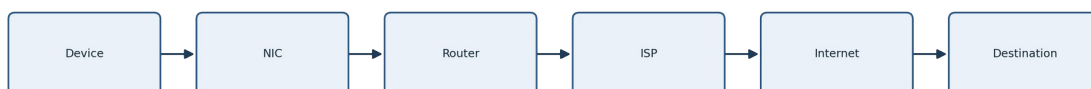
A router is a networking device that connects different devices like mobile phones, laptops, and computers to a network or the Internet. Its main job is to forward data to the right destination.

When data leaves our device through the NIC (Network Interface Card), it first reaches the router. The router then checks its routing table, which works like a map that tells the router where to send the data next.

The data we send is divided into small parts called packets. These packets are sent through the network, and the router forwards them to the next network or device. In a home network, the router usually sends the packets to our ISP (Internet Service Provider), which then forwards them toward the Internet and eventually to the destination.

In simple words:

Path Through a Router



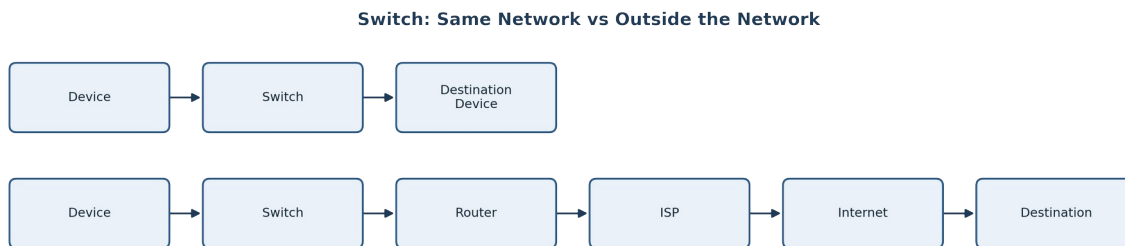
What is a Switch?

A switch is a networking device that connects multiple devices within the same local network (LAN).

For example, if a network has many devices, we can connect a switch to the router and then connect multiple devices to the switch. If there are even more devices, multiple switches can be connected together to expand the network.

The switch manages communication between devices inside the same network by using MAC addresses. If the data needs to go outside the local network, the switch forwards it toward the router, and the router sends it toward the ISP and the Internet.

For example:



In a large network, multiple switches can be used so that many devices can connect and communicate within the same network without requiring the router to have a separate port for every device.

In simple words, a switch connects and manages many devices inside a local network, while the router connects that network to other networks and the Internet.

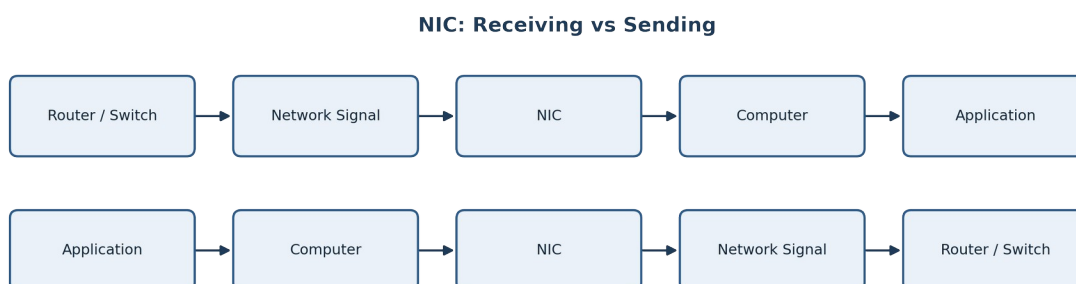
What is a NIC?

A NIC (Network Interface Card) is the part of a computer or device that allows it to communicate with a network.

When data comes from a network, such as from a router or switch, it reaches the NIC first. The NIC receives the signal from the network and converts it into data that the computer's networking system can process.

Similarly, when an application such as Chrome wants to send data over the network, the data is passed through the computer's networking system to the NIC. The NIC converts the digital data into the appropriate network signal and sends it through the network toward the router, switch, or another network device.

For example:



In simple words, the NIC acts as the communication interface between the computer and the network, allowing the computer to send and receive data over the network.

What is a Network Packet?

A packet is a small piece of data that travels through a network.

For example, when you open Chrome and request some data from a server, your device creates a request. Before this data is sent through the network, it may be divided into smaller pieces called packets.

The main reason for dividing data into smaller packets is to make it easier to send and manage across the network. Instead of sending one large piece of data, the network can send it in smaller pieces.

These packets then travel through the network toward the destination. They may not always take exactly the same path or arrive in the same order. At the destination, the packets can be put back together to reconstruct the original data.

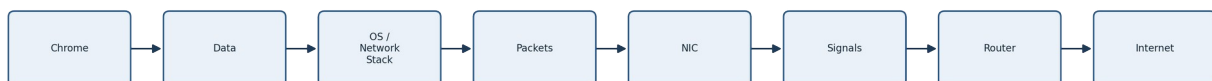
For example:

Data Becomes Packets



In simple words, a packet is a small unit of data that carries information from one point to another across a network.

Chrome's Data Reaches the Internet

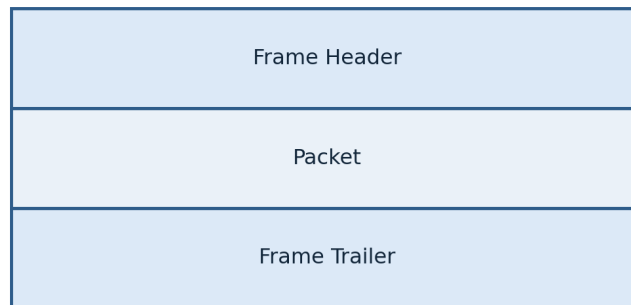


What is a Frame? and Summary

A frame is a data-link layer unit that carries a packet across a local network.

When a device wants to send data over a network, the application creates the data, which is then processed by different layers. The network layer creates a packet and adds IP addresses to it. The packet is then placed inside a frame.

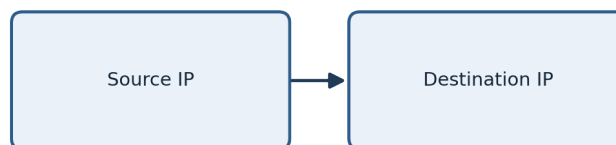
You can think of it like this:

Frame = Header + Packet + Trailer

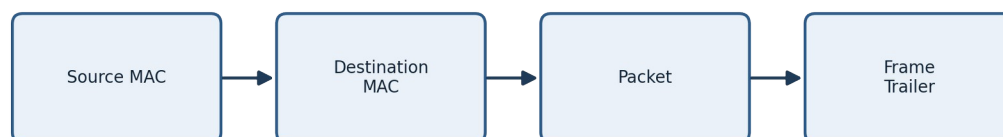
The frame header contains information such as the source MAC address and destination MAC address. The packet inside the frame contains the source and destination IP addresses.

For example:

Packet:

Inside the Packet

Frame:

Inside the Frame

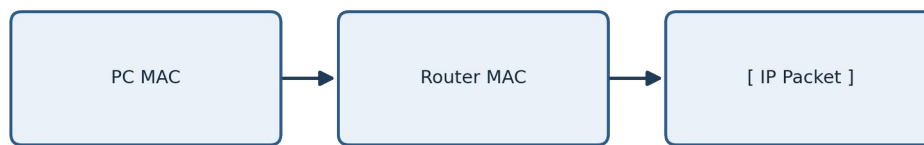
The important thing is that the packet can travel across multiple networks, while the frame is mainly used for the current local network hop.

For example, imagine your computer wants to communicate with a server on the Internet:

Your PC → Router → ISP Router → Next Router → Server

When your PC sends the data to its router, the packet is placed inside a frame:

Frame — First Hop



When the router receives the frame, it removes the frame and looks at the packet inside it. The router checks the destination IP and decides where the packet should go next.

The router then creates a new frame for the next network hop:

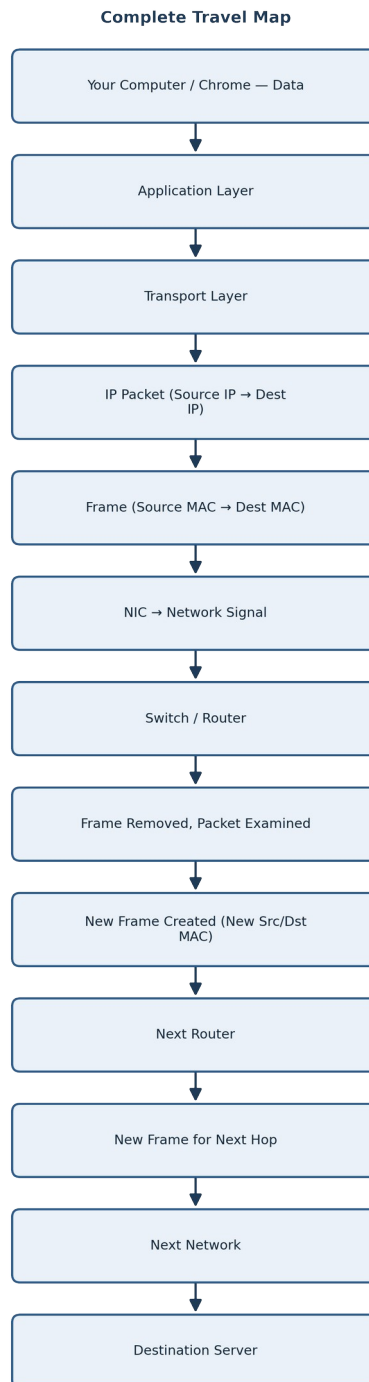
Frame — Next Hop



At the next router, the same process happens again. The old frame is removed and a new frame is created for the next hop.

So, the MAC addresses in the frame can change at every network hop, while the source and destination IP addresses in the packet normally remain the same from the original sender to the final destination.

Complete Travel Map



The basic idea is:

IP Address → Helps decide where the packet needs to go

MAC Address → Helps deliver the frame to the next device on the local network

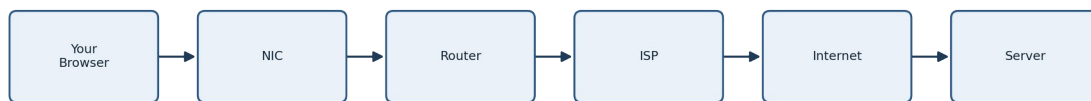
Frame → Carries the packet across the current local network link

In simple words, the packet contains the network-level information, while the frame wraps that packet and helps deliver it from one network device to the next.

Chapter 4 — The Journey of a Single Packet

Follow a packet step by step:

The Journey of a Single Packet



OSI Model — Real Example: Opening a Website

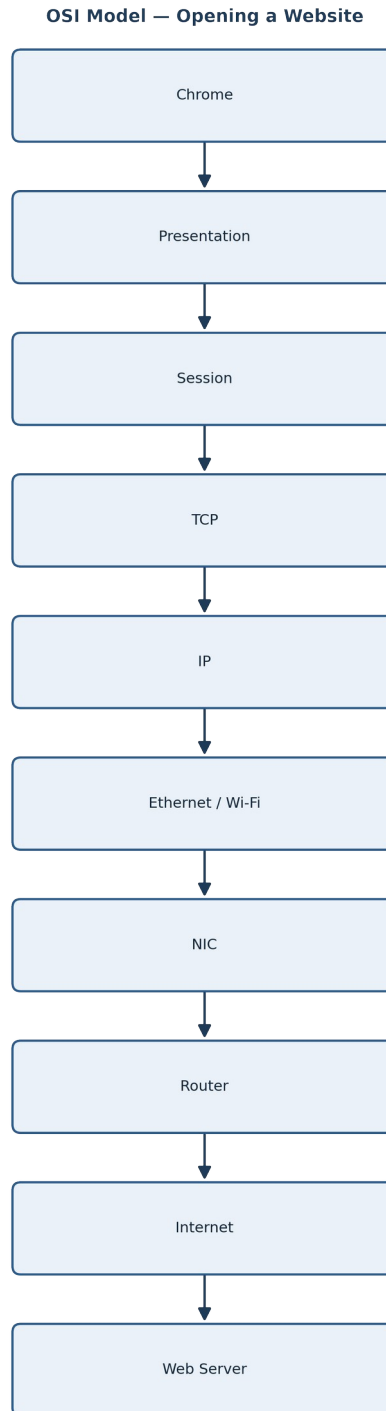
Imagine you open Chrome and type:

`https://example.com`

Then you press Enter.

Your computer needs to send a request to the web server.

The journey can be understood like this:



Now let's understand every step.

Layer 7 — Application Layer

This is where the application interacts with network services.

You open Chrome and request:

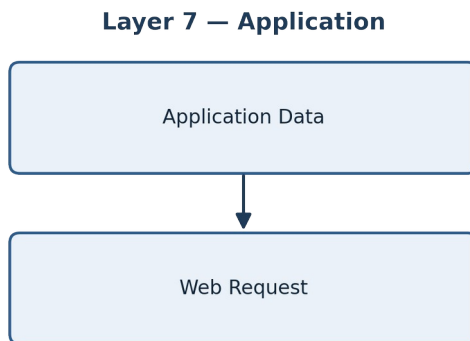
https://example.com

Chrome needs to communicate with the web server.

For an HTTP request, the application-level data might conceptually contain information such as:

GET / HTTP/1.1
Host: example.com

So at this stage, we can simply say:



Nothing has become a packet or frame yet. The application has simply created the data that needs to be sent.

Main idea: Application Layer → Creates/uses the application-level data.

Examples of application protocols include:

- HTTP
- HTTPS
- DNS
- SMTP
- FTP

Layer 6 — Presentation Layer

Now comes the Presentation layer.

Its job is related to how the data is represented, including things such as:

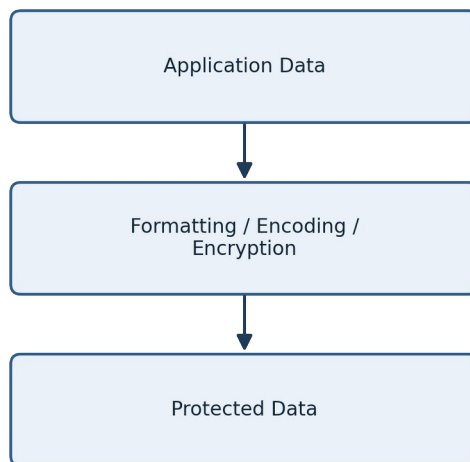
- Data formatting
- Encoding/translation
- Encryption
- Compression

For HTTPS, encryption is important.

For example, the HTTP data should not simply travel across the Internet as readable text. TLS is used to protect the application data.

So conceptually:

Layer 6 — Presentation



But remember: the Presentation layer does not necessarily add a “Presentation Header.” In real Internet networking, these functions are often handled by protocols such as TLS and application/software libraries rather than by a separate OSI Layer-6 header.

Main idea: Presentation Layer → Prepares the data's representation, including functions such as encryption and formatting.

Layer 5 — Session Layer

The Session layer is responsible conceptually for managing communication sessions.

For example, it deals with things such as:

Layer 5 — Session



Again, in modern networking, you usually will not see a separate Session Header being added to every packet. Session-related functions can be handled by higher-level protocols and applications.

Main idea: Session Layer → Manages the communication session between systems/applications.

Layer 4 — Transport Layer

Now things become much more concrete.

For HTTPS, TCP is commonly used.

The Transport layer adds a TCP header.

The TCP header contains information such as: Source Port, Destination Port, Sequence Number, Acknowledgment Number, Flags, Window, Checksum.

For example:

Source Port → 52000 Destination Port → 443

Port 443 tells the destination system that this traffic is intended for the HTTPS service.

The application data is now carried by TCP. Conceptually:

TCP Segment (Layer 4)



When TCP is used, this unit is commonly called a TCP segment.

So: Data → TCP Segment

Main idea: Transport Layer → Handles end-to-end transport and uses ports to identify applications/services.

Layer 3 — Network Layer

Now the segment moves to the Network layer.

Here, IP is involved. The IP layer adds an IP header.

For example:

Source IP → 192.168.1.10 Destination IP → 93.x.x.x

The IP header contains information such as: Source IP, Destination IP, TTL / Hop Limit, Protocol / Next Header, Other IP control information.

Now the structure becomes:

IP Packet (Layer 3)

This is an IP packet.

So: TCP Segment → placed inside an IP Packet

The destination IP helps routers determine where the packet needs to go.

Main idea: Network Layer → IP addressing and routing.

Layer 2 — Data Link Layer

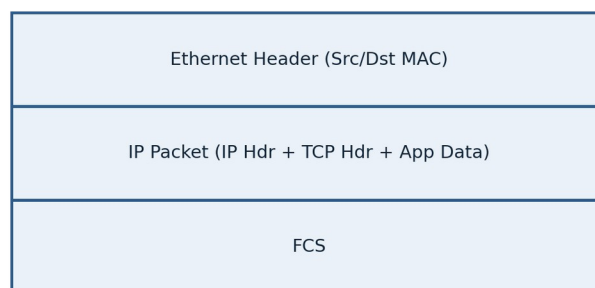
Now we have an IP packet.

But your computer cannot simply throw the IP packet onto an Ethernet or Wi-Fi network by itself.

The Data Link layer creates a frame and places the IP packet inside it.

For example, suppose your computer is sending the packet to its local router. The frame might conceptually look like:

Source MAC → Your PC Destination MAC → Your Router

Frame (Layer 2)

Now we have: IP Packet → placed inside a Frame

The frame uses MAC addresses for delivery on the current local network/link.

Main idea: Data Link Layer → Creates frames and handles local delivery using MAC addresses.

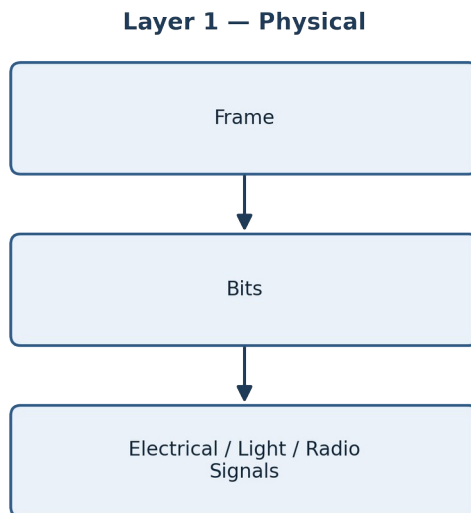
Layer 1 — Physical Layer

Now we finally reach the Physical layer.

The frame is transmitted as signals through the network medium.

For example: Ethernet → Electrical signals, Fiber → Light signals, Wi-Fi → Radio signals.

Conceptually:



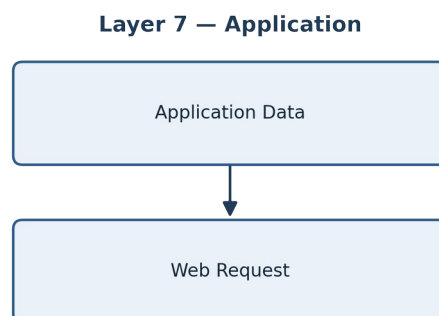
The NIC is the network interface that sends and receives these signals.

So your computer is now physically transmitting the data toward the next network device.

Now Look at the Complete Encapsulation

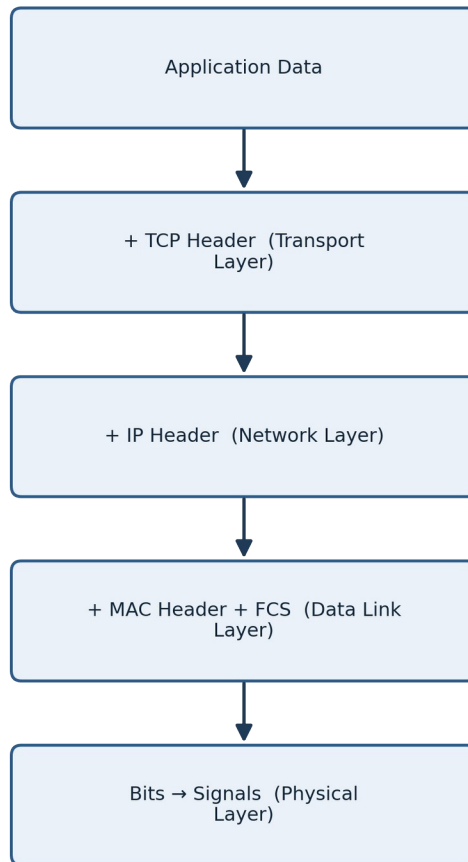
This is the most important part.

You started with:



Then as the data moves down:

Encapsulation — Building Up the Data



That is encapsulation.

But What Happens When the Frame Reaches the Router?

This is where your previous Frame + MAC address concept becomes very important.

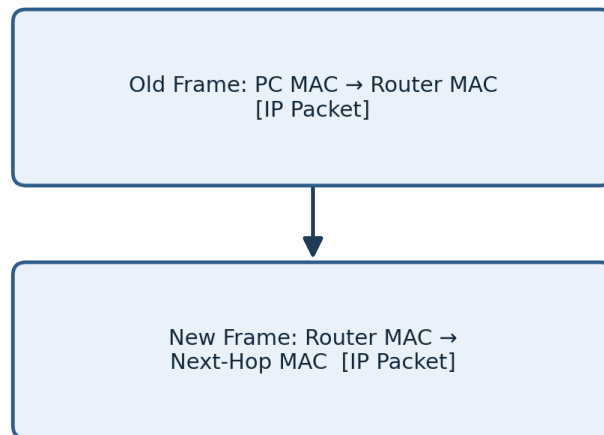
Your computer sends:

PC → Switch → Router

The router receives the frame. The router does not simply forward the exact same frame to the next router. Instead, the router processes the Layer-2 frame, gets the IP packet, checks the destination IP, and determines the next hop. Then it creates a new Layer-2 frame for the next link.

For example:

Frame Is Rewritten at the Router



So:

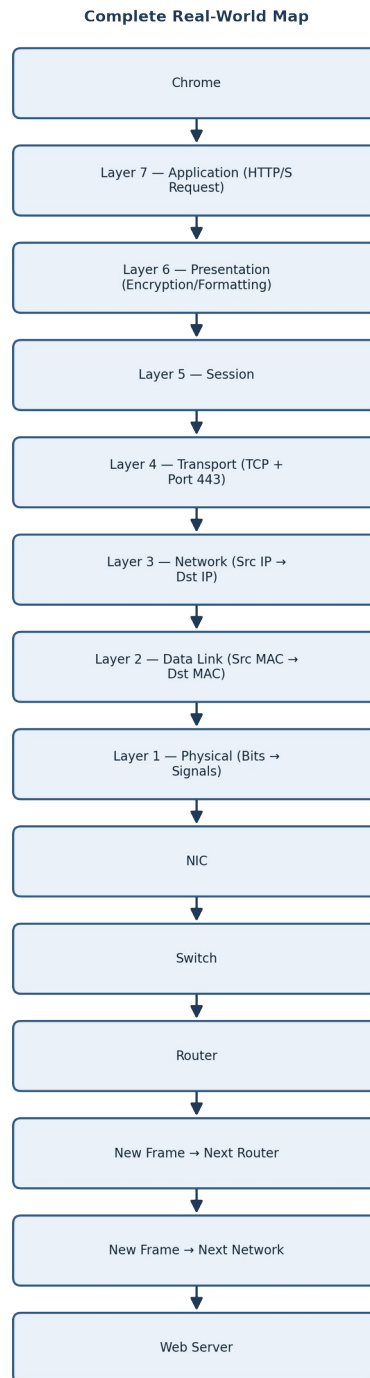
MAC addresses — Can change at each hop.

IP addresses — Normally remain the same from the original source to the final destination, although some technologies such as NAT can modify them.

TCP information — Travels inside the IP packet toward the destination.

Complete Real-World Map

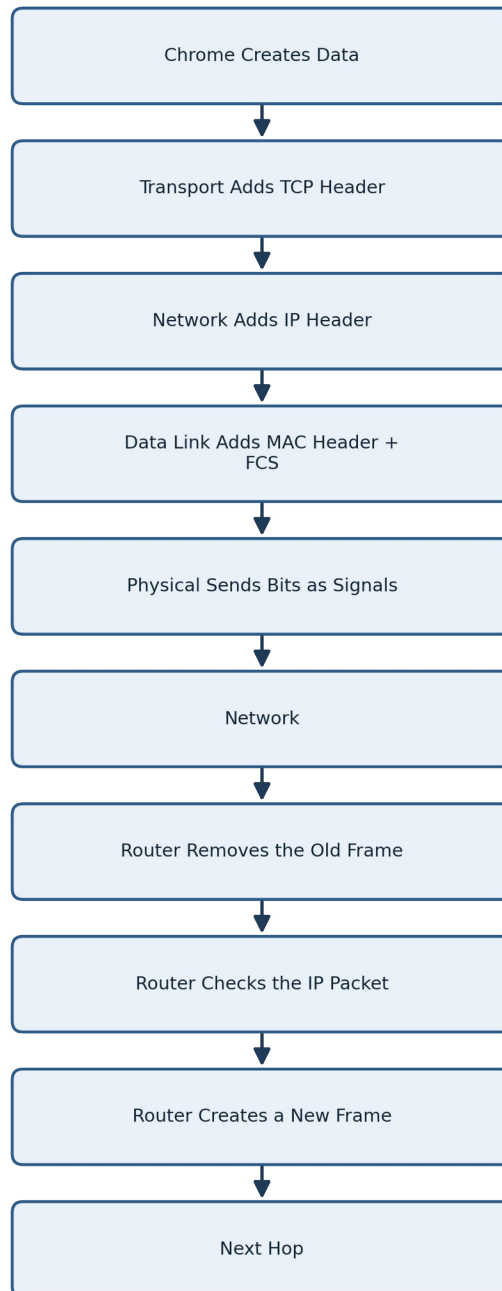
Now put everything together:



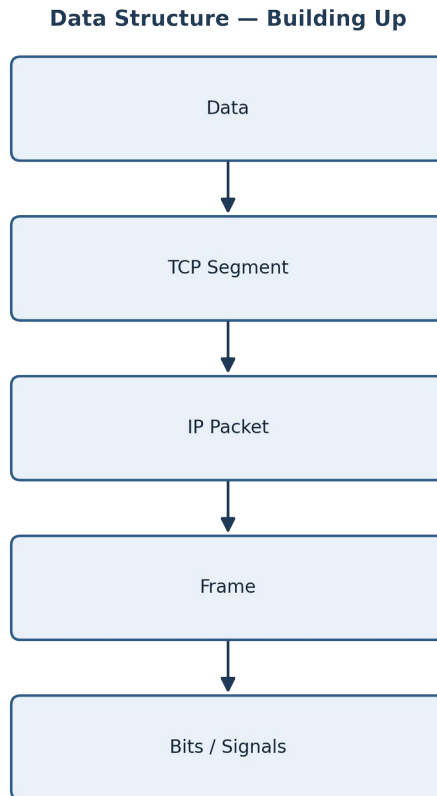
The One-Line Story

If you remember only one flow, remember this:

The One-Line Story



And the structure is:



This is the foundation you need before learning encapsulation, de-encapsulation, packets, frames, MAC addresses, IP addresses, ports, switches, routers, and NICs together.

Encapsulation and De-encapsulation

When data travels from one computer to another, it passes through different layers of the network.

At the sender, each layer adds the information needed for communication. This process is called encapsulation.

At the receiver, the added information is processed and removed layer by layer. This process is called de-encapsulation.

Encapsulation

Let's take a real example. You open Chrome and request:

<https://example.com>

Your computer needs to send a request to the web server.

Step 1 — Application Layer

Chrome creates the application-level data.

Application Data

For example, this could contain an HTTP request.

Step 2 — Presentation / Session Functions

The data may be formatted or encrypted, especially when HTTPS is being used.

Encrypted Application Data

There is not necessarily a separate Presentation or Session header in modern Internet communication.

Step 3 — Transport Layer

The Transport layer adds a TCP header.

For example: Source Port → 52000, Destination Port → 443, Sequence Number → ...

Now:

TCP Segment (Layer 4)



This is a TCP segment.

Step 4 — Network Layer

The Network layer adds an IP header.

For example: Source IP → Your IP, Destination IP → Server IP

Now:

IP Packet (Layer 3)



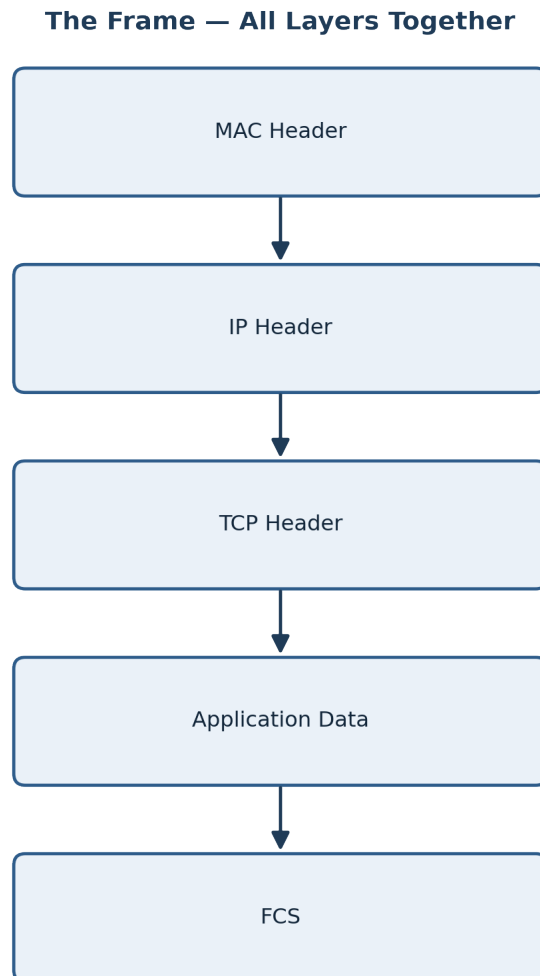
This is an IP packet.

Step 5 — Data Link Layer

Now the packet is placed inside a frame.

The frame gets Layer-2 information such as: Source MAC → Your NIC MAC, Destination MAC → Router MAC

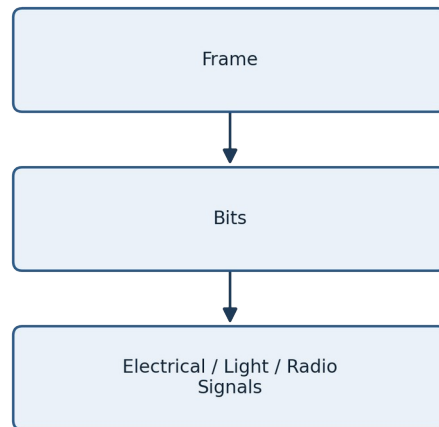
Now:



This is the frame.

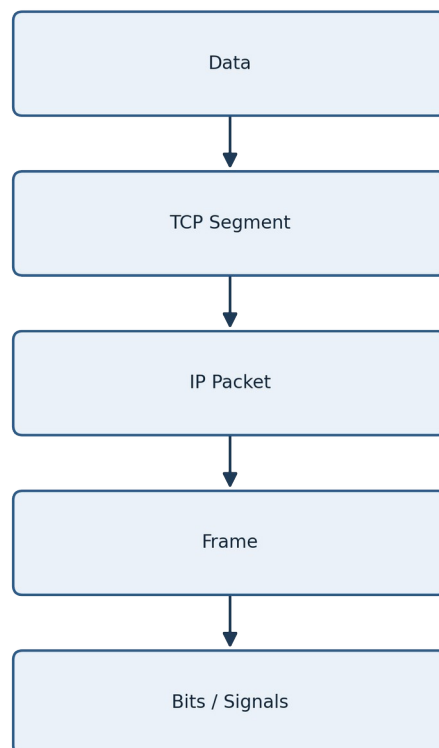
Step 6 — Physical Layer

Finally, the frame is transmitted as bits/signals through the network.

Layer 1 — Physical

The NIC sends these signals through the network.

So the complete encapsulation process is:

Data Structure — Building Up

Each layer adds information around the data from the layer above. That is why it is called encapsulation.

What Happens at the Router?

Now the frame reaches your router.

The router receives the signal through its network interface and processes the frame.

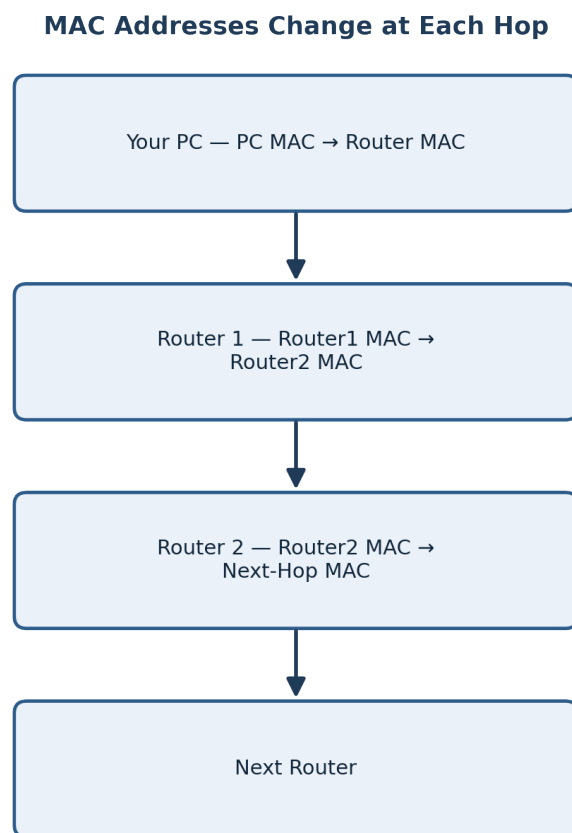
The router looks at the Layer-2 information and obtains the IP packet inside the frame.

The router checks the destination IP address and decides where to send the packet next.

The old frame is not simply sent unchanged to the next router. Instead, the router creates a new frame for the next network.

This is why MAC addresses can change at each hop.

For example:

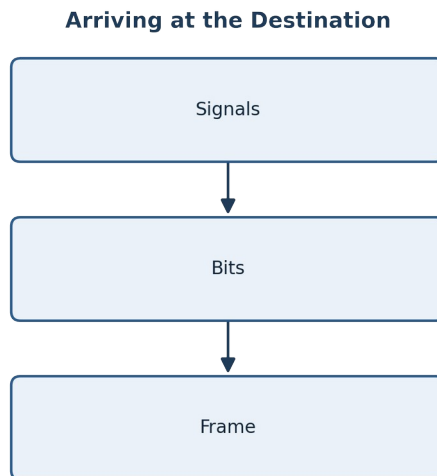


The Layer-2 frame changes at each hop, while the IP packet continues toward the destination.

De-encapsulation

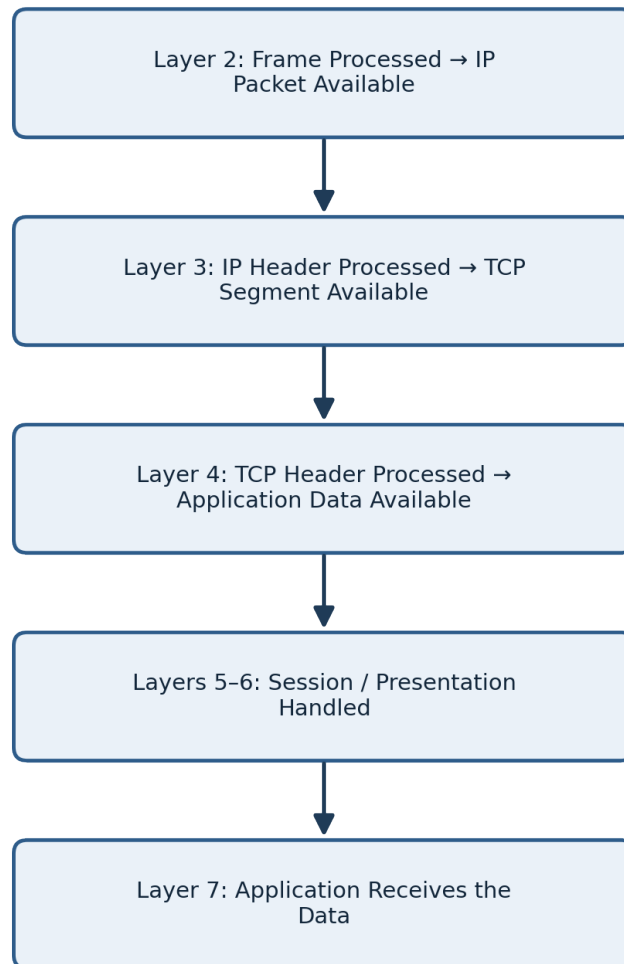
When the frame finally reaches the destination server, the reverse process happens. This is called de-encapsulation.

The server receives the signals:



The server's network interface receives the frame. The Data Link layer processes the frame and removes/processes the Layer-2 information. The Network layer processes the IP header and checks information such as the destination IP. The Transport layer processes the TCP header and uses the destination port to determine which application/service should receive the data. TCP also processes information such as sequence and acknowledgment information. Session-related and presentation-related functions are handled as required, and for HTTPS, encrypted data is processed so that the application can use the appropriate data. Finally, the data reaches the application, and the web server receives and processes the request.

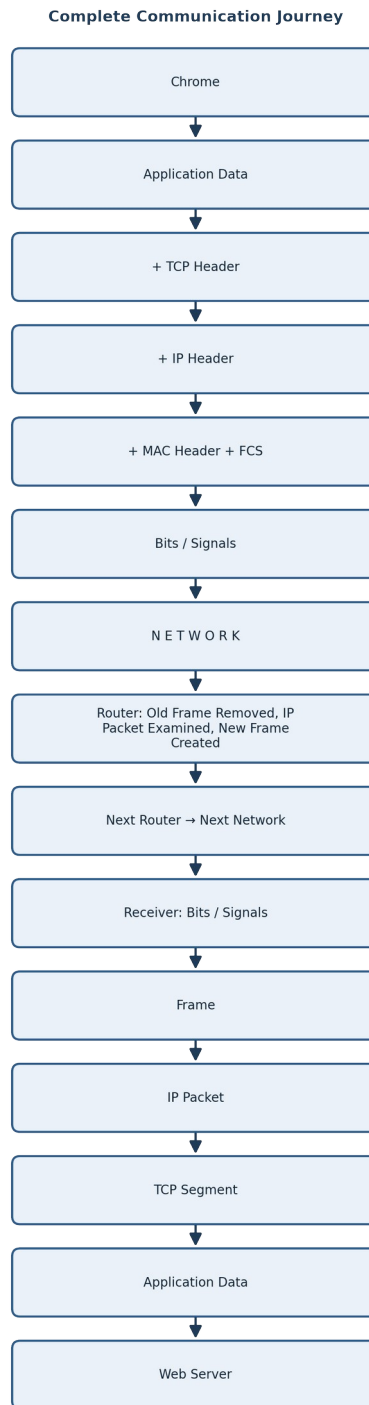
De-encapsulation, Layer by Layer



This is de-encapsulation.

Complete Journey

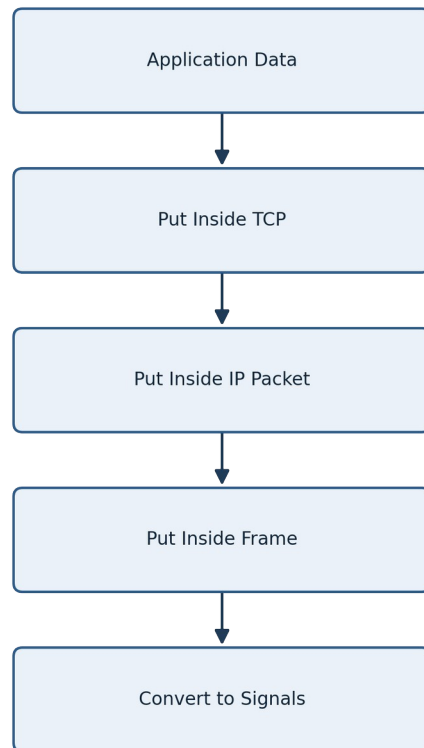
The complete communication looks like this:



The Most Important Concept

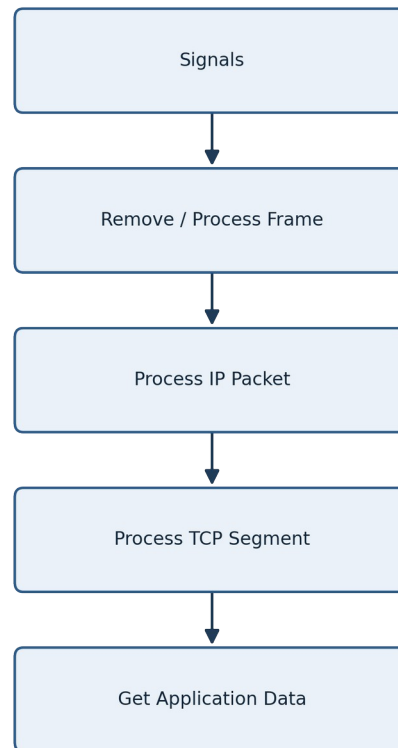
Think of it like putting a letter inside multiple envelopes.

Like Nested Envelopes — Sending



At the destination, the process is reversed:

Opening the Envelopes — Receiving



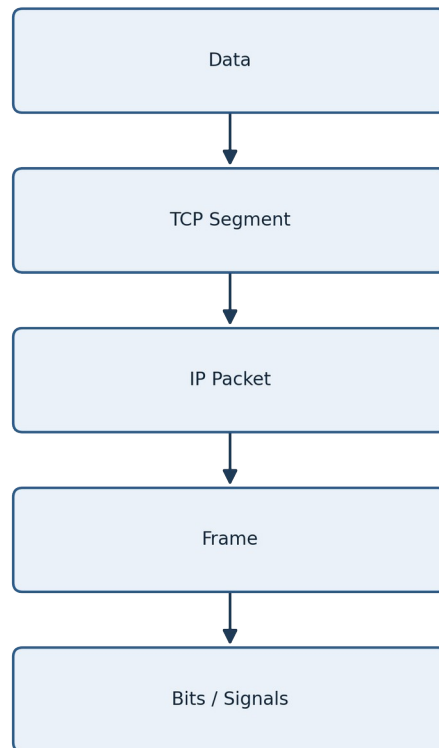
So:

Encapsulation = Adding information while moving DOWN the layers.

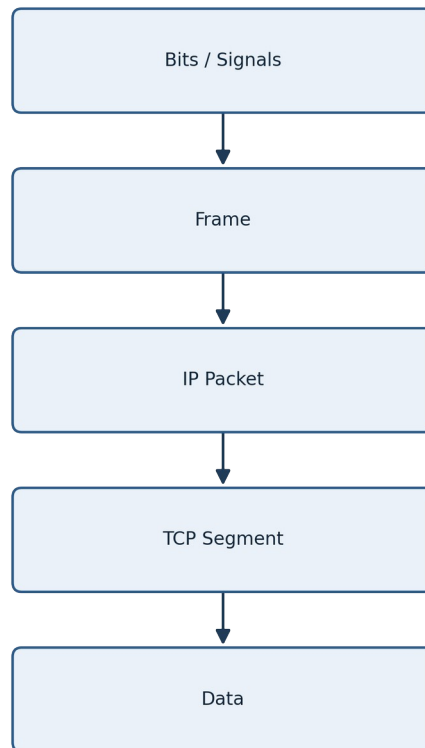
De-encapsulation = Processing/removing that information while moving UP the layers.

And the basic flow to remember is:

Data Structure — Building Up



At the destination:

Data Structure — Stripping Down

How TCP and IP Work Together

To understand TCP and IP, let's follow a real example.

Suppose you open Chrome and enter:

`https://example.com`

Your computer needs to communicate with the web server.

TCP and IP have different jobs:

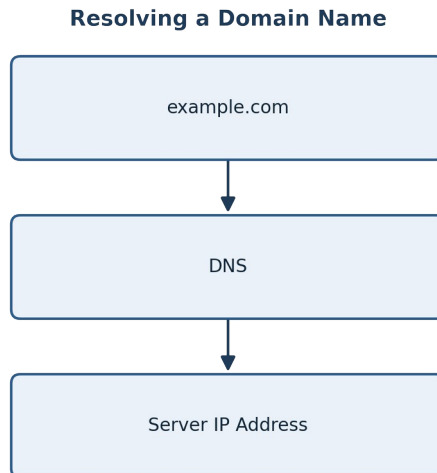
TCP → Manages the communication between the two endpoints.

IP → Provides addressing and routes packets toward the destination.

Let's see how they work together.

1. Finding the Server

First, the domain name needs to be resolved to an IP address.



Now your computer knows where the server is.

For example: Your PC: 192.168.1.10 Server: 93.x.x.x

2. TCP Three-Way Handshake

If TCP is being used, the client first establishes a TCP connection with the server.

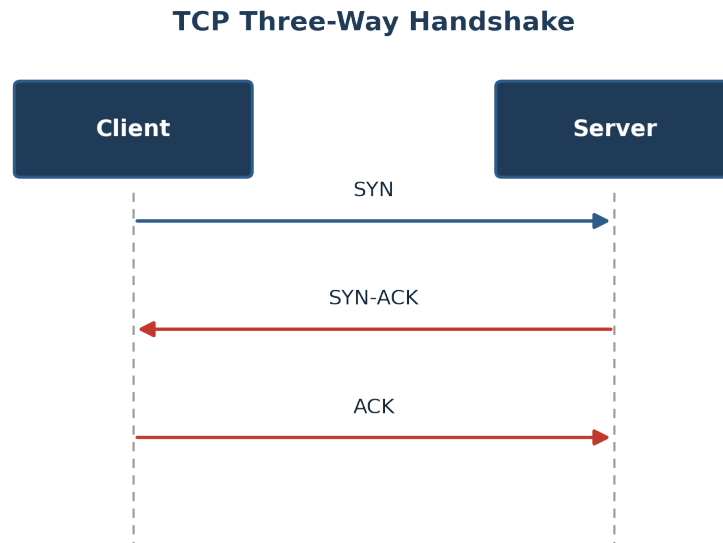
The connection starts with a three-way handshake.

Step 1 — SYN: Your computer sends a TCP packet to the server: Client → SYN → Server. The TCP header contains Source Port → 52000, Destination Port → 443, Sequence Number, SYN Flag → 1. The IP header contains Source IP → 192.168.1.10, Destination IP → 93.x.x.x.

Step 2 — Server Sends SYN-ACK: The server receives the SYN and responds: Client ← SYN-ACK ← Server. The addresses are reversed: Source IP → 93.x.x.x, Destination IP → 192.168.1.10, Source Port → 443, Destination Port → 52000.

Step 3 — Client Sends ACK: The client receives the SYN-ACK and sends: Client → ACK → Server. Now the TCP connection is established.

The complete handshake is:



The handshake allows both sides to establish the TCP connection and synchronize sequence information.

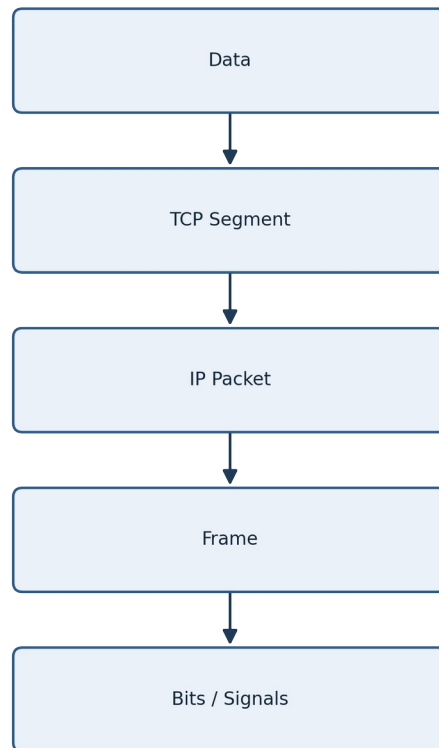
Now the Actual Data Is Sent

Once the TCP connection is established, Chrome can send application data.

For example: Chrome → HTTPS Request. TCP adds its header, creating a TCP segment. Then IP adds its header, creating an IP packet. Then the Data Link layer puts that packet inside a frame. Finally, the frame is converted into bits/signals and transmitted by the NIC.

So:

Data Structure — Building Up

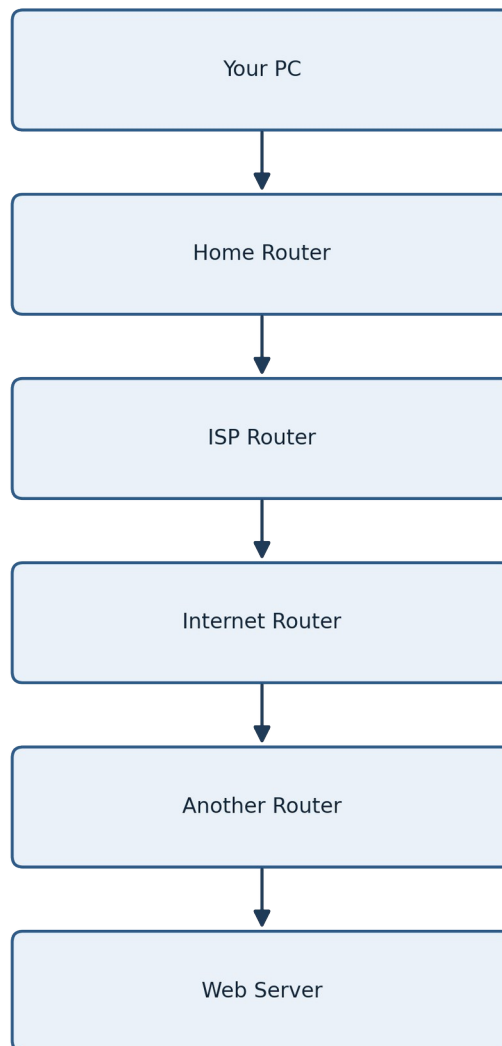


6. What Does IP Do?

IP is responsible for addressing and routing.

Suppose your packet needs to travel:

Routing Across the Internet



The destination IP tells routers where the packet ultimately needs to go.

For example: Source IP → 192.168.1.10, Destination IP → 93.x.x.x

A router receives the packet, checks the destination IP, looks at its routing information, and decides where to forward it next.

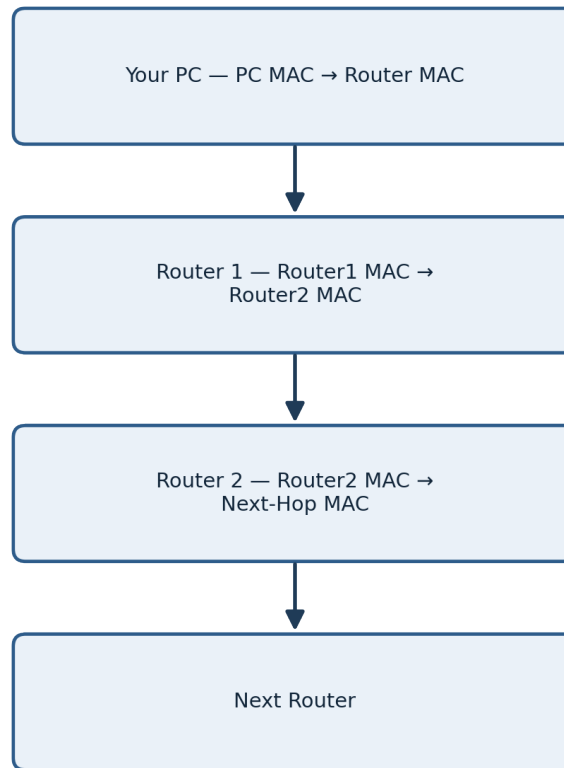
So IP's basic job is: "Where does this packet need to go?"

7. What Happens to the MAC Address?

This connects with the Frame concept we learned earlier.

Your PC may send the first frame like this, and the router receives the frame, processes the Layer-2 information, examines the IP packet, and creates a new frame for the next network. Therefore, MAC addresses can change from hop to hop while the IP packet continues toward the destination.

MAC Addresses Change at Each Hop



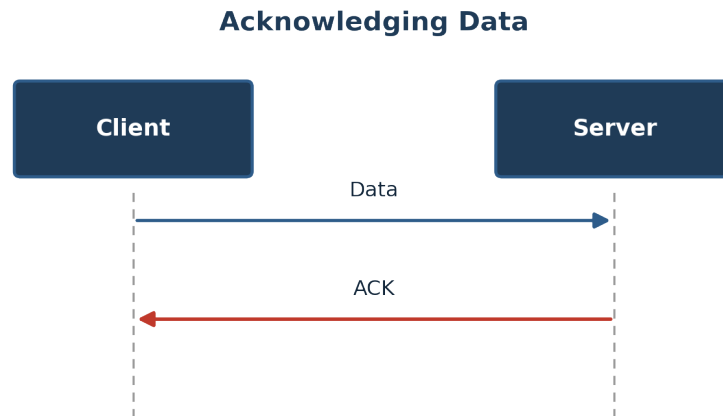
8. What Does TCP Do?

While IP is moving packets through the network, TCP manages the end-to-end communication between your computer and the server.

TCP can use:

Sequence Numbers — TCP uses sequence numbers to keep track of the data.

Acknowledgments — The receiver sends acknowledgments to confirm received data.



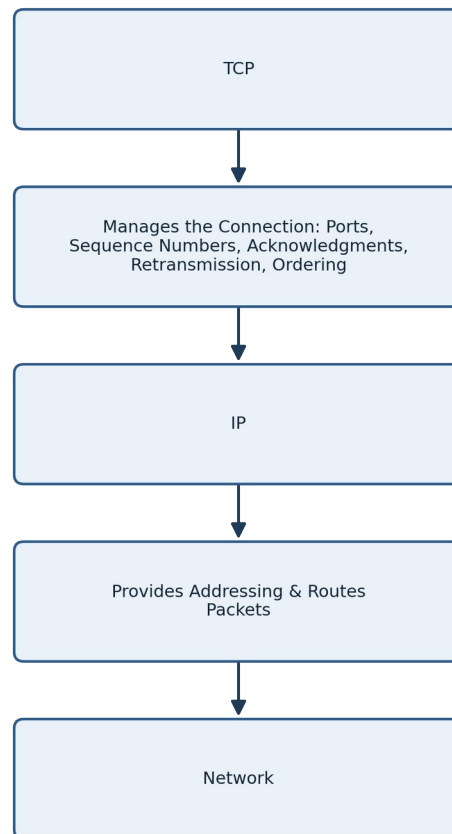
Retransmission — If TCP determines that data was lost, it can retransmit it.

Ordering — TCP can use sequence information to ensure that the byte stream is delivered to the application in the correct order.

So TCP's basic question is: “How do I manage reliable communication between these two endpoints?”

9. TCP and IP Working Together

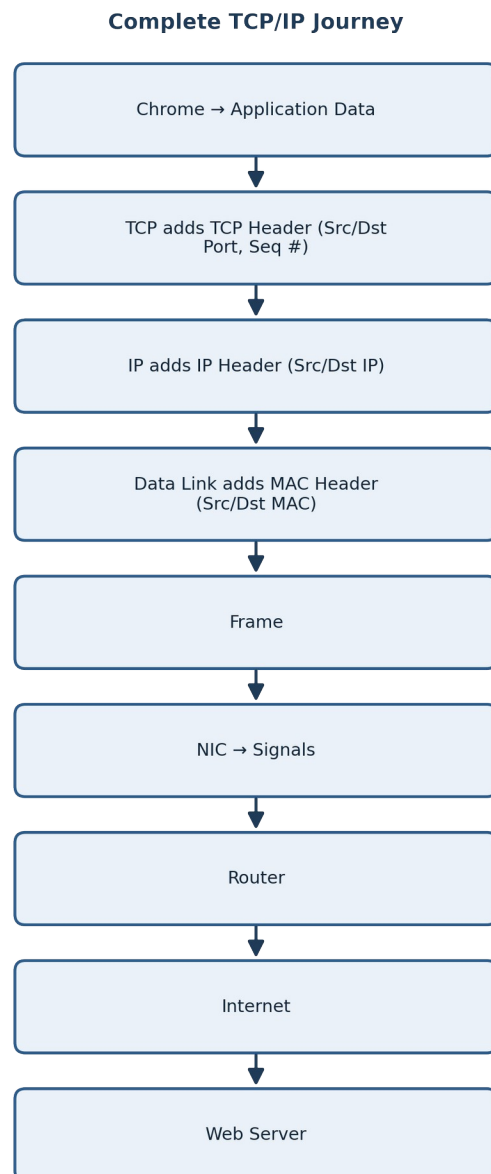
This is the easiest way to remember their jobs:

TCP and IP — Their Roles

Think of it like this: IP is responsible for getting the packet toward the correct destination. TCP is responsible for managing the data transfer between the source and destination.

10. Complete Journey

Now put everything together:

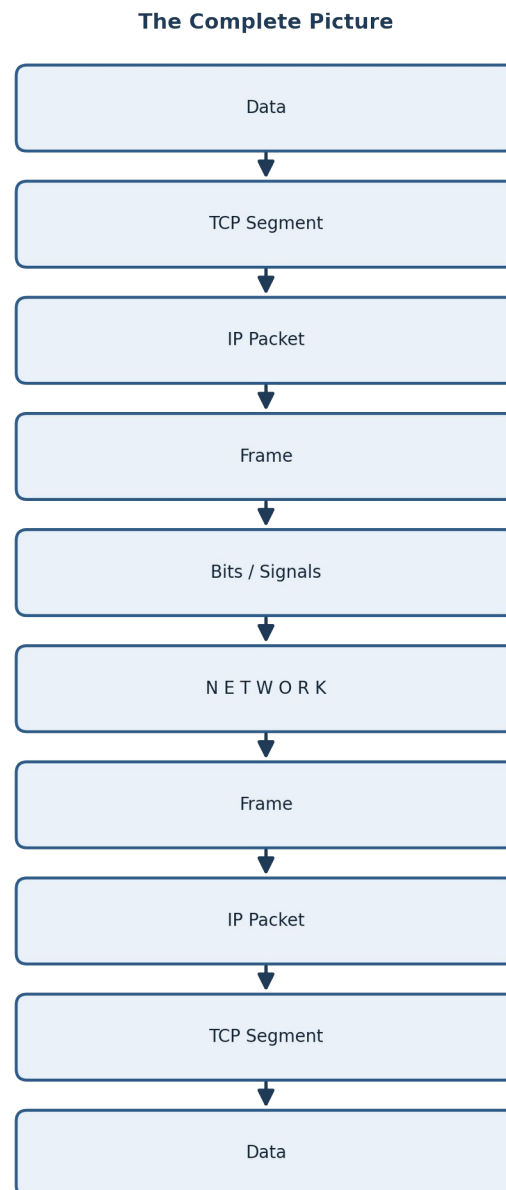


And before the actual data transfer, the three-way handshake completes and the TCP connection is established, after which data and acknowledgments flow between client and server.

At every router, the old frame is processed, the IP packet is examined, a routing decision is made, and a new frame is created for the next hop.

At the destination, the reverse process happens — this is called de-encapsulation.

The Complete Picture



So the complete story is:

Chrome creates the data → TCP manages the end-to-end communication → IP provides addressing and routing → Data Link creates the frame for the current link → Physical layer sends the signals → routers forward the packet → the destination de-encapsulates the data → the server receives it.

TLS Certificates — How They Are Generated and Exchanged

When we visit a website using HTTPS, TCP establishes the connection first. After that, TLS (Transport Layer Security) is used to create a secure connection between the client and server.

But an important question is: How does the browser know that it is really communicating with the correct server?

This is where a TLS certificate is used.

1. How is a Certificate Created?

Before a website can use HTTPS, its server needs a TLS certificate.

First, the website owner generates a private key and a public key. The private key must be kept secret on the server. The public key can be shared with others.

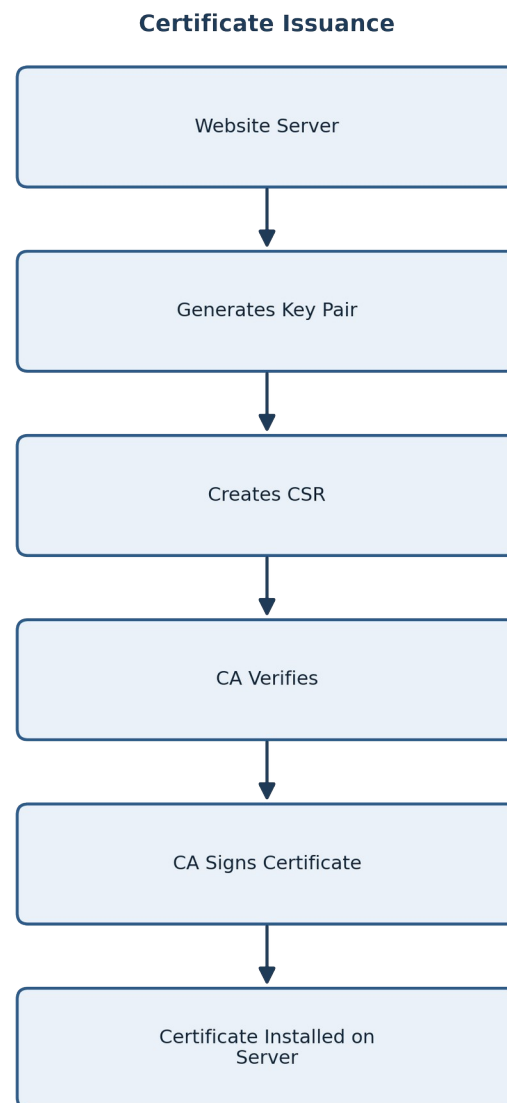
The website owner then creates a Certificate Signing Request (CSR). The CSR contains information such as: the website's domain name, the server's public key, and information about the organization, when applicable.

The CSR is sent to a Certificate Authority (CA). A Certificate Authority is a trusted organization that verifies the domain or organization and then signs the certificate.

For example, a website such as `https://example.com` may obtain a certificate from a trusted CA.

The CA digitally signs the certificate using the CA's private key. This signature allows browsers to verify that the certificate was issued by a trusted authority and has not been changed.

So, before HTTPS communication starts, the basic structure is:

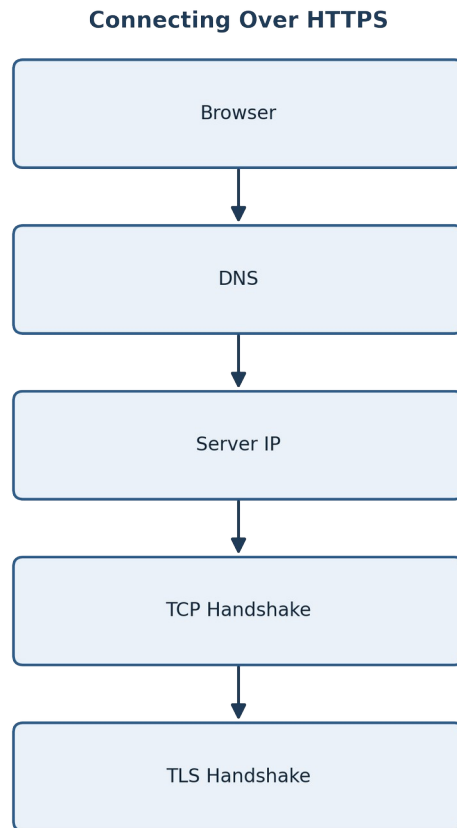


The server keeps its private key secret. The certificate and public key can be shared with clients.

2. What Happens When the Browser Connects?

Now suppose we open `https://example.com`

The process is roughly:



After the TCP connection is established, the TLS handshake begins.

The server sends its TLS certificate to the browser. The certificate contains information such as: the domain name, the server's public key, the Certificate Authority that issued it, validity dates, and the CA's digital signature.

The browser then checks the certificate. It verifies things such as: Is this certificate issued by a trusted CA? Does the certificate belong to this domain? Is the certificate still valid? Has the certificate been modified or revoked?

If these checks pass, the browser can trust that the server's identity has been properly verified.

3. Does the Certificate Encrypt the Whole Connection?

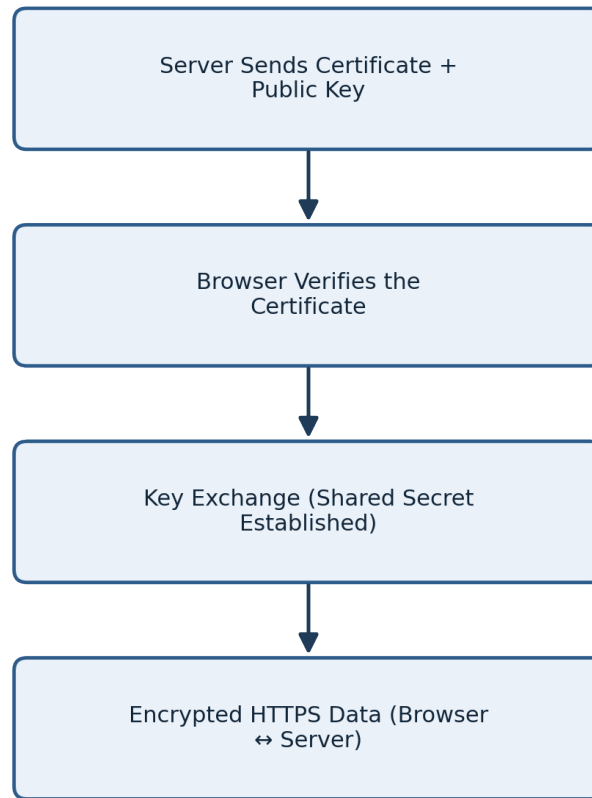
No. This is an important point.

The certificate itself is mainly used to authenticate the server and provide its public key.

Modern TLS then uses a secure key-exchange process to establish session keys that are used to encrypt the actual communication.

A simplified flow looks like this:

TLS Handshake, Simplified



The actual web data is protected using symmetric encryption because it is much faster for continuous communication.

4. Why Does the Server Keep the Private Key Secret?

The server's private key is extremely important.

The public key can be shared openly, but the private key must remain secret.

The private key is used by the server in cryptographic operations during TLS. If an attacker obtains the private key, the security of the certificate and related TLS setup can be seriously compromised.

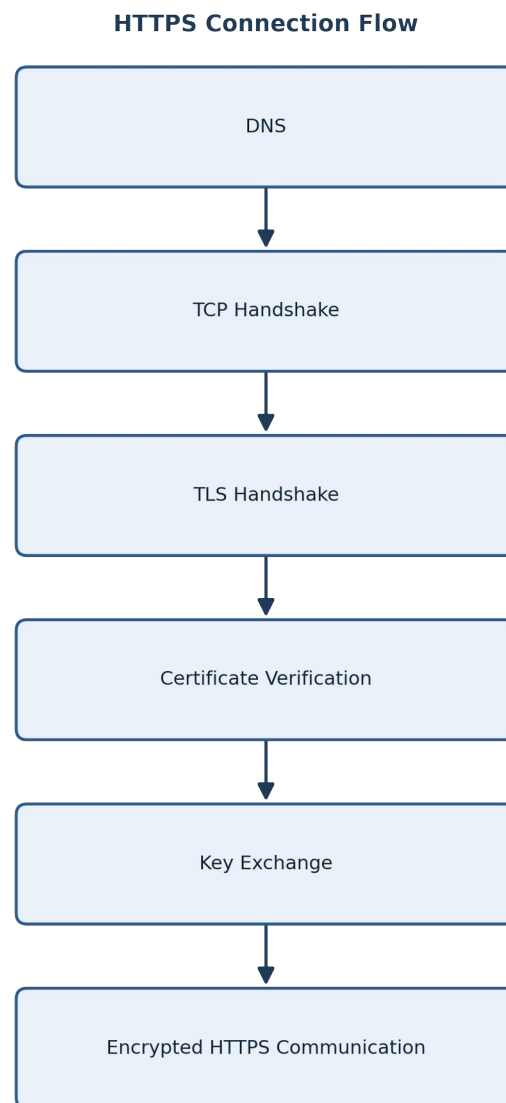
This is why the private key should never be sent to the browser. The browser receives the certificate and public information, not the server's private key.

5. Simple Real Example

Suppose you open `https://example.com`. The simplified process is:

1. DNS — The browser finds the server's IP address.
2. TCP — The browser and server establish a TCP connection using the three-way handshake.

3. TLS — The TLS handshake begins.
 4. Certificate — The server sends its TLS certificate.
 5. Verification — The browser checks the certificate, domain, validity, and CA trust.
 6. Key Exchange — The browser and server establish shared secret key material.
 7. Secure Communication — The browser and server can now exchange encrypted HTTPS data.
- So the complete simplified flow becomes:



In Simple Words

Think of the TLS certificate like a digitally verified identity card for a website.

The certificate helps the browser answer: “Am I really talking to the website I intended to visit?”

The certificate also provides the server's public key and is backed by a trusted Certificate Authority.

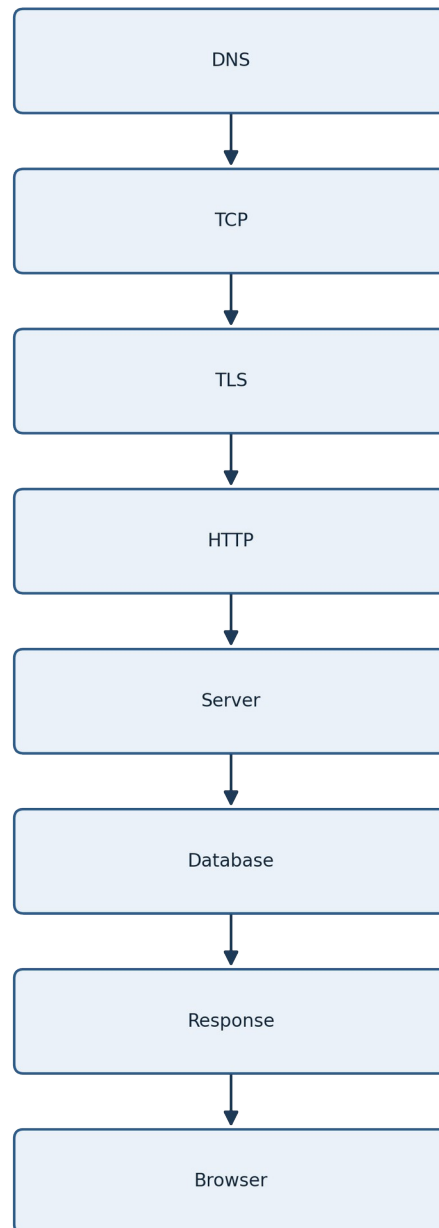
After the server is authenticated, TLS establishes secret session keys, and those keys are used to protect the actual communication.

Certificate = Server identity + Public key + CA's digital signature

TLS session keys = Protection for the actual communication

Chapter 5 — What Happens When You Open a Website?

Follow the complete journey:

Opening a Website — Overview

Chapter 5 — What Happens When You Open a Website?

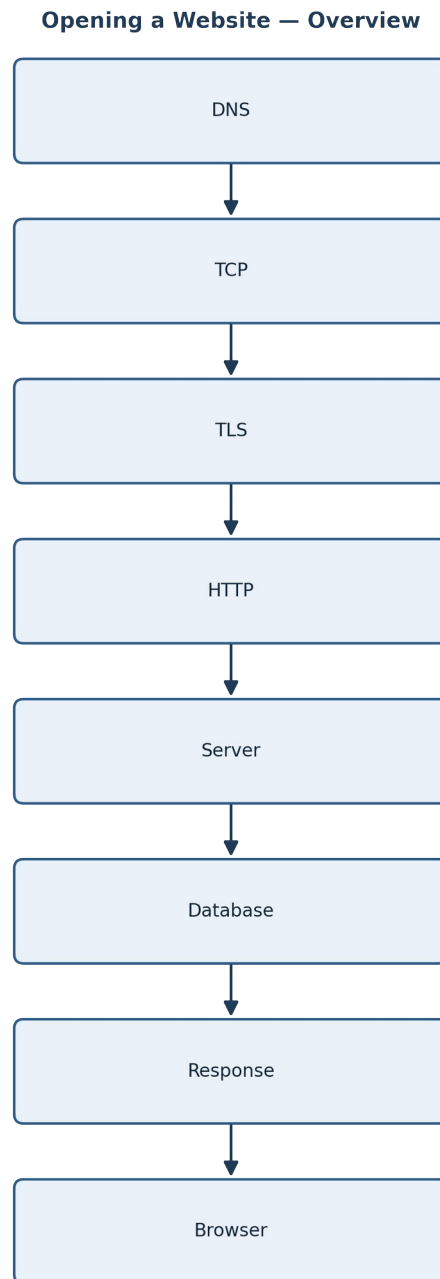
In the previous chapters, we learned about DNS, IP addresses, ports, TCP, TLS, HTTP, packets, frames, servers, databases, and browsers separately.

Now let's connect everything.

What actually happens when you type a website address such as `https://example.com` and press Enter?

A lot happens in a very short time.

The simplified journey is:



Let's follow this journey step by step.

1. You Enter the Website Address

Suppose you open Chrome and type: `https://example.com`

Your browser first needs to understand where this website is located.

The domain name `example.com` is easy for humans to remember, but computers communicate across networks using IP addresses.

So the browser needs to find the IP address of the server. This is where DNS comes in.

2. DNS Finds the Server

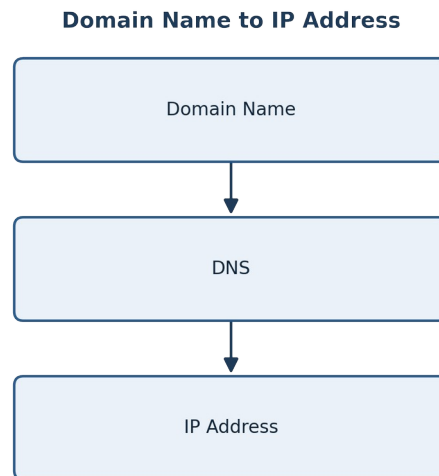
DNS stands for Domain Name System. It converts a domain name into an IP address.

For example:

`example.com` → `93.184.216.34`

The actual IP address may be different depending on the website and DNS configuration.

Now the browser knows where it needs to connect. The journey continues:



But knowing the IP address is not enough. The browser also needs to establish a reliable connection with the server.

3. TCP Establishes the Connection

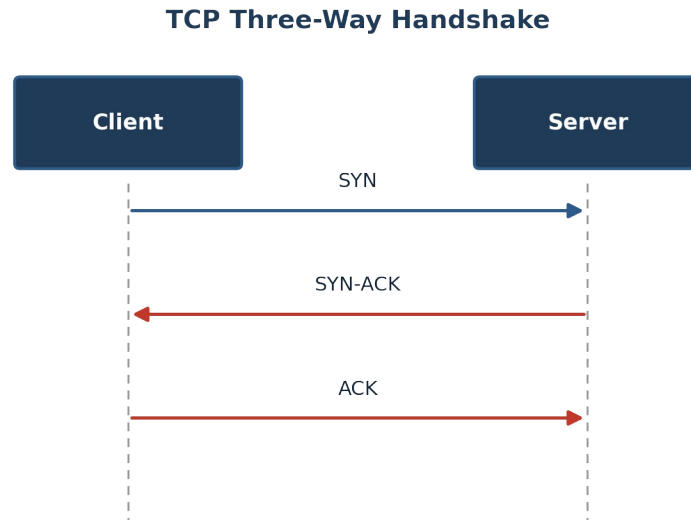
For a normal HTTPS connection using TCP, the browser starts a TCP connection with the server.

TCP uses a three-way handshake.

Step 1 — SYN: The client sends: Client → SYN → Server. The client is basically saying: “I want to establish a TCP connection.”

Step 2 — SYN-ACK: The server responds: Client ← SYN-ACK ← Server. The server is saying: “I received your request, and I am ready.”

Step 3 — ACK: The client sends: Client → ACK → Server. Now the TCP connection is established.



TCP will handle things such as: reliable delivery, sequence numbers, acknowledgments, retransmission, ordering of data, and flow control.

But there is still a problem. The connection exists, but the data is not yet protected by TLS. That's the next step.

4. TLS Secures the Connection

Because we opened `https://example.com`, the browser needs to establish a secure TLS connection.

TLS stands for Transport Layer Security.

The server sends its TLS certificate to the browser during the TLS handshake.

The certificate contains information such as: the domain name, the server's public key, certificate validity information, the Certificate Authority that issued it, and the CA's digital signature.

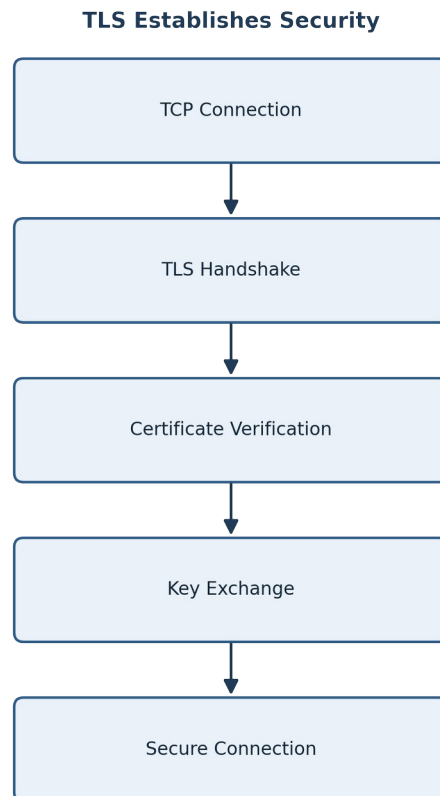
The browser checks the certificate. It asks questions such as: Does this certificate belong to the website I requested? Is it valid? Is it trusted?

If the certificate passes the necessary checks, the browser can authenticate the server.

The browser and server then perform the TLS key-exchange process and establish shared secret key material for the connection.

The actual website data can now be protected using encryption.

So the simplified flow is:



5. The Browser Sends an HTTP Request

Now the browser can send the actual web request.

For example, it may send an HTTP request asking for the webpage.

Conceptually: Browser → Server: “Please give me the webpage for example.com.”

An HTTP request contains information such as: HTTP method, requested path, headers, and other request information.

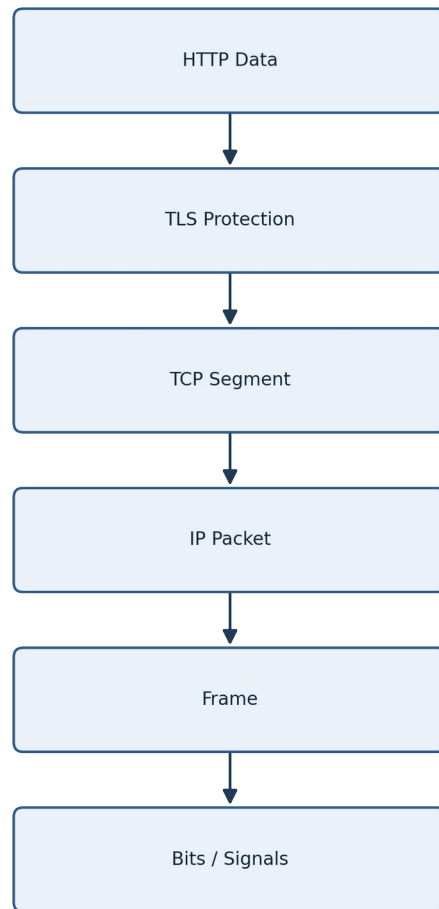
For example, a simplified request might look like:

GET / HTTP/1.1

The important thing to understand is that HTTP is responsible for the web request and response, while TLS protects the communication when HTTPS is being used.

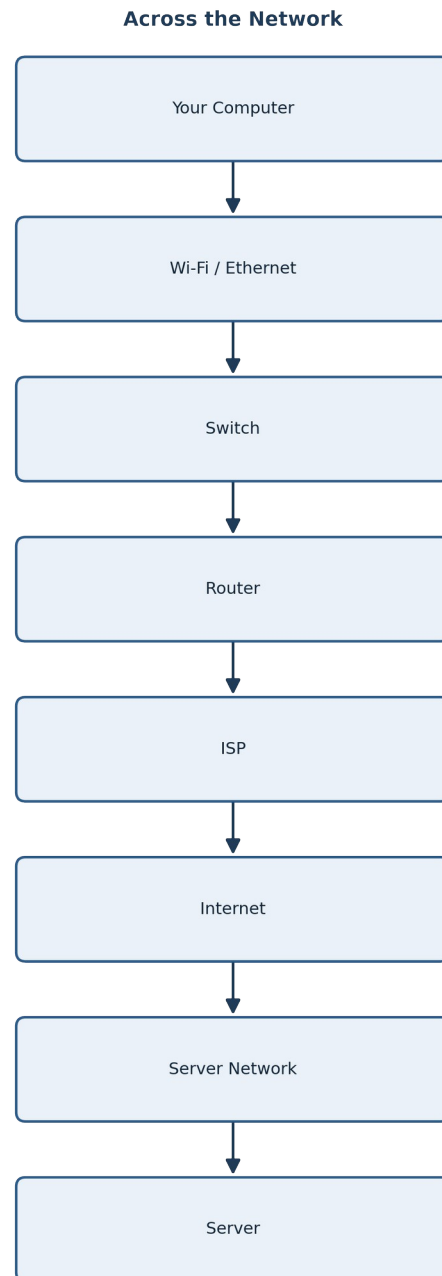
The HTTP data is passed to TLS. TLS protects it. Then TCP carries it. IP handles addressing and routing. The Data Link layer carries it across the current network link.

So the data moves through multiple layers:

Request Travels Down the Stack**6. The Data Travels Through the Network**

The request now travels from your computer toward the server.

A simplified path may look like:



At each network hop, devices process the information they need to forward the traffic.

Routers mainly use the destination IP address to decide where to forward the IP packet next.

At the Data Link layer, the frame is used for the current local link.

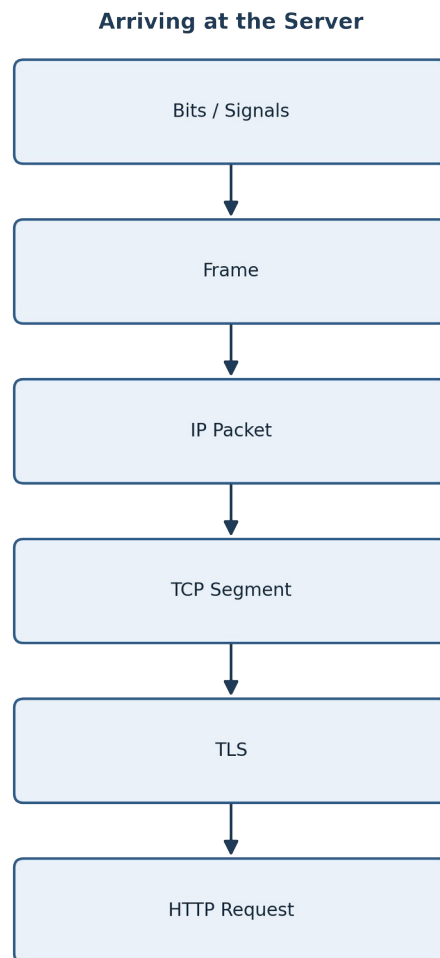
This is why MAC addresses can change from one network hop to another while the IP packet is normally carried toward the final destination.

Eventually, the request reaches the server.

7. The Server Receives the Request

Now the server receives the network traffic.

The process happens in the opposite direction. The server's networking stack processes the received data:



Eventually, the web server or application receives the HTTP request.

Now the server needs to decide what to do with it.

8. The Server Processes the Request

Not every website simply sends a stored HTML file.

Modern websites often have an application running on the server.

For example, the application might be written using technologies such as: Python, Java, JavaScript/Node.js, PHP, C#, Go.

The application receives the request and processes it.

It may check: Who is the user? What page was requested? What information is needed? Does the user have permission? Does information need to be retrieved from a database?

This is where the server-side application becomes important.

9. The Server May Contact a Database

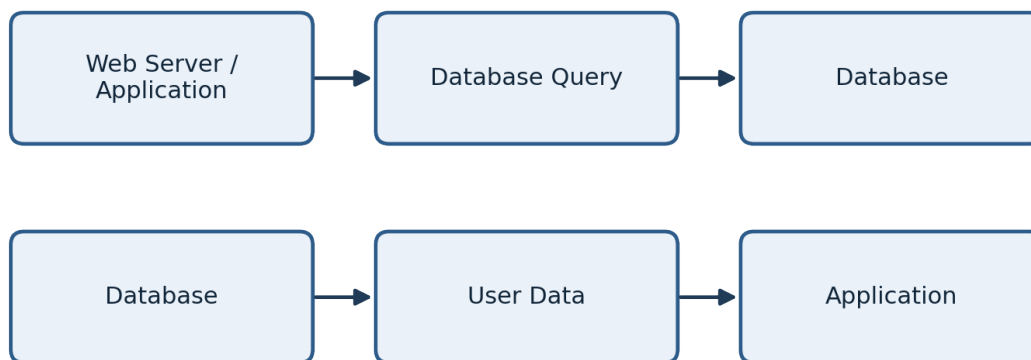
Suppose you open your profile page on a website.

The server may need information such as: your name, profile information, posts, settings, and other application data.

That information may be stored in a database.

The application sends a database query. Conceptually:

The Server Contacts a Database



The database searches for the required information. It then sends the result back to the application.

The application processes the result and prepares the response.

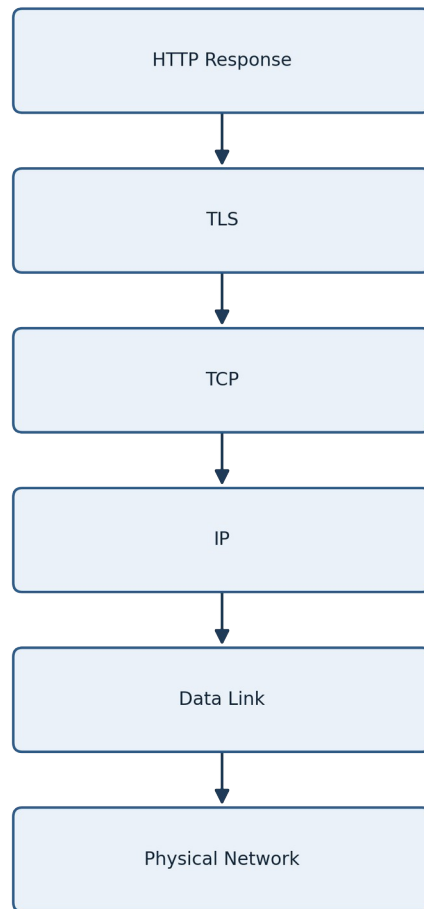
10. The Server Creates the Response

After processing the request, the server creates an HTTP response.

A simplified response might contain: HTTP status code, headers, HTML, CSS references, JavaScript references, and other data.

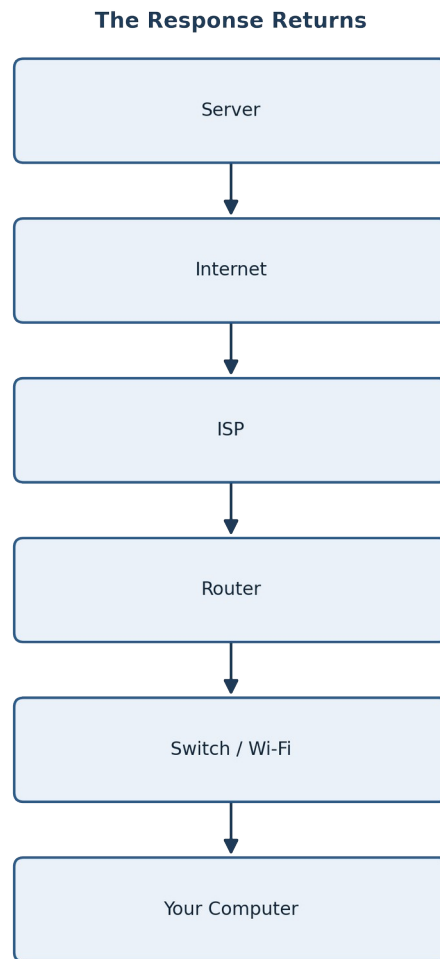
For example, 200 OK means the request was successfully processed.

The response is then passed back through the network stack. Conceptually:

Response Travels Down the Stack**11. The Response Travels Back**

The response now travels from the server back toward your computer.

The journey is roughly:

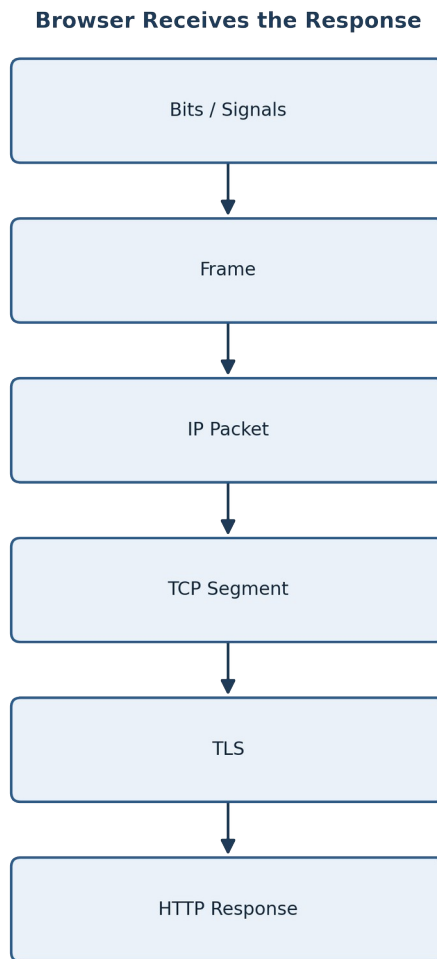


TCP helps ensure that the data arrives reliably and in the correct order. If some TCP data is lost, TCP can request or trigger retransmission.

At your computer, the data is processed by the networking stack.

12. The Browser Receives the Response

The browser finally receives the protected response. It processes the data through the layers:



Now the browser has the information it needs.

But the page may contain references to other resources. For example: CSS files, JavaScript files, images, fonts, videos, APIs.

The browser may make additional HTTP requests to retrieve these resources.

So opening one website can actually involve many individual requests and responses.

13. The Browser Builds the Webpage

The browser takes the received HTML and processes it.

It loads the required CSS and JavaScript and requests other resources when needed.

Then the browser builds the page and renders it on your screen.

The process ends with what you see: Website displayed in the browser.

What looks like a simple action—“I opened a website.”—actually involved many systems working together.

Complete Journey

Let's put the entire process together. You type: `https://example.com`

Step 1 — DNS: Domain → IP address

Step 2 — TCP: TCP three-way handshake

Step 3 — TLS: Certificate verification + secure key establishment

Step 4 — HTTP: Browser sends HTTP request

Step 5 — Network: Request travels through routers and networks

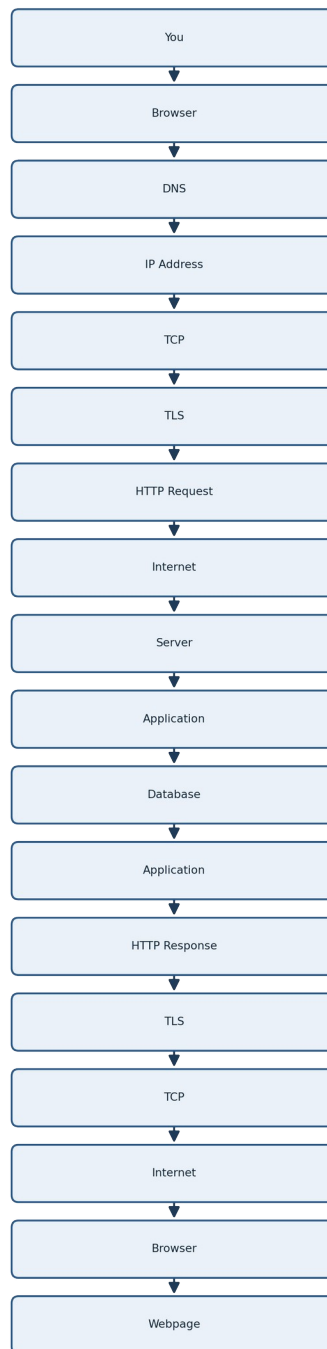
Step 6 — Server: Server receives and processes the request

Step 7 — Database: Application may retrieve required data

Step 8 — Response: Server sends HTTP response

Step 9 — Browser: Browser receives, processes, and renders the webpage

The complete simplified map is:

Complete Journey — Opening a Website

One Important Thing to Remember

These technologies do different jobs.

DNS — Finds the IP address associated with a domain.

TCP — Provides reliable transport and manages the connection.

TLS — Authenticates the server and protects the communication.

HTTP — Defines how the browser and web server communicate through requests and responses.

IP — Provides addressing and routing between networks.

Server/Application — Processes the request and generates a response.

Database — Stores and retrieves application data.

Browser — Processes the response and displays the webpage.

In Simple Words

When you open a website, you are not directly “opening a webpage.”

You are starting a chain of communication: Find the server → Connect to it → Verify and secure it → Send a request → Process the request → Get data → Send the response → Display the webpage.

That entire process can happen within seconds.

Part 1 — What We Learned

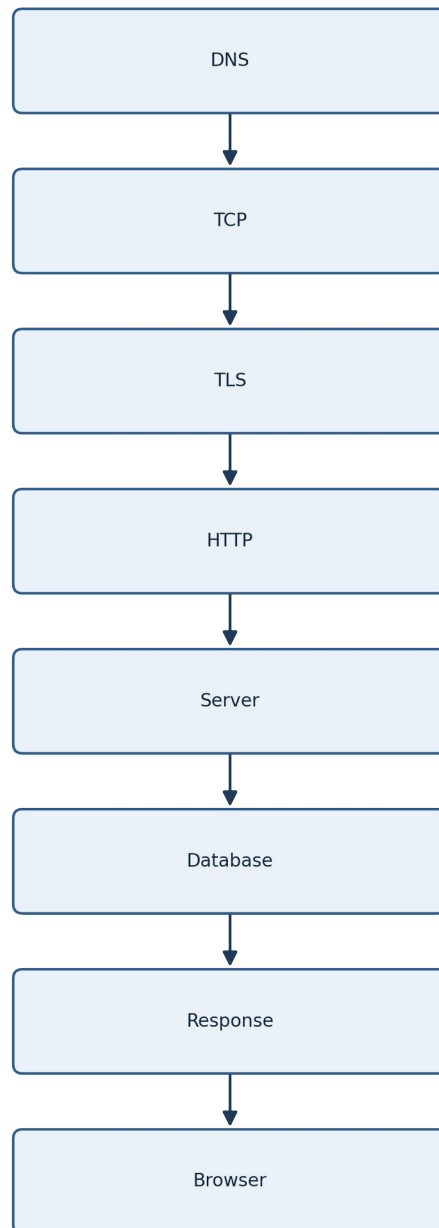
We started this part with a very simple question: What actually happens when we use a computer or open a website?

To answer that question, we went all the way from the basic components of a computer to network communication.

We learned about:

- CPU and how it processes instructions
- RAM and temporary memory
- Storage and permanent data
- Operating systems
- Processes and files
- Bits, bytes, and binary
- GPU and how visual information reaches the screen
- Client and server
- IP addresses and MAC addresses
- Ports
- NIC
- Switches and routers
- Packets and frames
- Encapsulation and de-encapsulation
- TCP and how reliable communication works
- DNS and how domain names are resolved
- TLS and HTTPS
- Certificates and how they help authenticate websites
- HTTP requests and responses
- Servers and databases

Finally, we connected everything together and followed the complete journey of opening a website.

Opening a Website — Overview

This is important because cybersecurity does not start with a hacking tool.

It starts with understanding how the digital world actually works.

A Message From My Experience

Before going further, I want to share something from my own learning journey.

When I started learning cybersecurity, I made mistakes.

One of the biggest mistakes was trying to learn tools and advanced concepts before making my foundations strong.

I wanted to use different tools, learn different technologies, and understand advanced cybersecurity concepts. But sometimes I did not stop and ask the basic questions:

What is actually happening here? Why did this happen? How did this happen? Where did the data go? Which component is responsible for this? What would happen if I changed this?

With time, I realized that knowing a tool is not the same as understanding cybersecurity.

A tool can show you a result. But you should be able to understand why that result happened.

That is why I strongly believe that before going deep into cybersecurity, you should build a strong foundation.

You should understand computers. You should understand operating systems. You should understand networks. You should understand how applications communicate. You should understand how data moves from one place to another.

Because without this foundation, cybersecurity can become a collection of commands that you run without really understanding what is happening.

Think Like a Detective

There is another thing I learned during my journey.

A cybersecurity person should develop observation and curiosity.

Think like a detective.

If something happens, don't just accept it.

Ask: Why? Then ask: How? Then ask: What caused it? And finally: Can I reproduce it?

For example, if something suddenly stops working, don't immediately search for a command to fix it.

First observe. What changed? Where did the communication stop? Is DNS working? Is the server reachable? Is the port open? Is the application running? Is the request reaching the server? Is the server returning an error?

Every answer gives you another clue.

This way of thinking is much more valuable than simply memorizing commands.

Experiment

One of the best ways to learn technology is to experiment.

Don't only read about something. Try it. Break it in a safe environment. Change something. Observe the result. Then try to understand why the result changed.

For example, if you are learning networking, don't just memorize what a port is.

Run a server. Connect to it. Change the port. Check what happens. Observe the traffic. Ask yourself why the connection works or fails.

When you experiment, concepts that looked difficult in theory start becoming real.

You begin to build a mental picture of what is happening behind the screen.

Don't Be Afraid of Technology

Another lesson I learned is simple: Don't be afraid of technology.

Sometimes a tool looks complicated. A terminal may look scary. A Linux command may look confusing. A packet capture may look like random information. A cloud console may have hundreds of options.

But remember: Everything can be broken down into smaller pieces.

You don't need to understand everything on the first day.

Take one thing. Understand it. Experiment with it. Then move to the next thing.

The goal is not to become an expert overnight. The goal is to keep understanding one more thing every day.

Don't Just Run Commands

This is something I wish I had understood earlier.

Don't become someone who only copies commands from the Internet.

If you run a command, ask yourself: What does this command actually do?

If it gives you an output, ask: Why did I get this output?

If it gives an error, don't immediately search for the exact error and copy someone else's solution.

First read the error. Understand what it is telling you. Then investigate.

Sometimes the error itself is the best teacher.

Mistakes Are Part of Learning

During my learning journey, I made many mistakes.

I got confused by networking concepts. I misunderstood some technologies. I spent time learning things in the wrong order. I tried to solve problems without fully understanding their basics.

I also learned that sometimes you can spend hours trying to fix something because you are looking at the wrong layer of the problem.

But these mistakes were not completely wasted.

They taught me how to approach problems differently.

So I don't want you to think that you must avoid every mistake.

Make mistakes. Just make them in a safe environment, understand them, and learn from them.

The goal is not to never make mistakes. The goal is to make fewer of the same mistakes over time.

Where This Journey Goes Next

So far, we have mostly learned how the digital world works underneath.

Now things become more interesting.

We know that a browser can communicate with a server. We know what an IP address is. We know how TCP establishes communication. We know how TLS protects HTTPS. We know how HTTP requests and responses work.

But now we can ask a much more interesting question: What is actually happening inside a web application?

What happens when you:

- Create an account?
- Log in?
- Upload a file?
- Submit a form?
- Change your password?
- Search for something?
- Send a message?
- Make a request to an API?

What does the server receive? Where does the data go? How does the application decide whether you are allowed to access something?

And most importantly: What can go wrong?

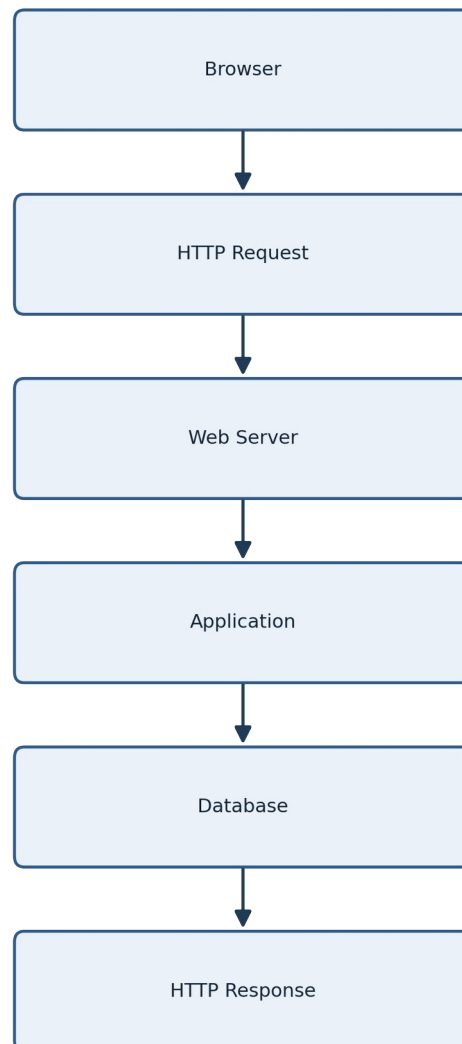
This is where our cybersecurity journey starts becoming much more practical.

The Next Part

In the next part, we will move from “How does the digital world work?” to: “How do web applications work, and where can security problems appear?”

We will start looking at:

Where the Security Journey Goes Next



And then we will start examining this flow from a security perspective.

We will learn how things such as authentication, sessions, cookies, APIs, user input, databases, and application logic fit together.

The goal will not simply be to learn attacks.

The goal will be to understand why vulnerabilities happen in the first place.

Because once you understand how something works, you are in a much better position to understand how it can fail.

And that is where the real cybersecurity journey begins.

Thank You

Thank you for reading all the way to the end of Part 1.

If you made it this far, you now understand something that most people who use computers and the internet every day never stop to think about — what is actually happening behind a single click. That is not a small thing. It is the exact foundation this entire field is built on.

Writing this book has been as much a learning experience for me as I hope reading it has been for you. Every explanation here was written with one goal: to make sure the next person starting this journey has an easier time than I did.

If this book helped you, saved you some confusion, or made a difficult concept finally click — that is genuinely all I hoped for. If you know someone else who is just starting out in tech or cybersecurity, feel free to pass this along to them too.

Part 2 is on its way, and it will build directly on everything you just learned here.

Thank you for trusting me with your time. See you in the next part.

— Zohaib Hayat

THANK YOU

