

CS0424IT — LECTURE NOTES

CYBERSECURITY SPECIALIST

*Simone La Porta*



*Written in L<sup>A</sup>T<sub>E</sub>X*

---

CONTENTS

<b>I</b>	<b>Unit 1: Fundamentals of Ethical Hacking</b>	<b>6</b>
1	Introduction and Virtual Machines	6
1.1	Virtualization Process . . . . .	7
1.1.1	Virtual Hardware . . . . .	8
1.1.2	Hypervisor Role . . . . .	8
1.1.3	Execution Flow . . . . .	9
1.1.4	Benefits of Virtualization . . . . .	9
2	Networking	10
2.1	Network Categories: Geographical and Topological . . . . .	10
2.1.1	Geographical Networks . . . . .	10
2.1.2	Topological Networks . . . . .	10
2.2	Basic Networking Concepts . . . . .	11
2.2.1	Data Transmission . . . . .	12
2.2.2	Packet Structure . . . . .	12

2.3	The ISO/OSI Model . . . . .	12
2.3.1	<b>Physical Layer</b> . . . . .	14
2.3.2	<b>Data Link Layer</b> . . . . .	15
2.3.3	<b>Network Layer</b> . . . . .	17
2.3.4	<b>Transport Layer</b> . . . . .	19
2.3.5	<b>Session Layer</b> . . . . .	21
2.3.6	<b>Presentation Layer</b> . . . . .	21
2.3.7	<b>Application Layer</b> . . . . .	23
2.4	Network Address Translation (NAT) . . . . .	24
2.5	Port Address Translation (PAT) . . . . .	25
3	Firewall & Cryptography . . . . .	27
3.1	Key Classifications of Firewalls . . . . .	27
3.2	Types of traffic filtering . . . . .	28
3.2.1	Static Packet Filtering . . . . .	28
3.2.2	Stateful Filtering . . . . .	29
3.3	Web Application Firewall (WAF) . . . . .	29
3.4	Next-Generation Firewall (NGFW) . . . . .	29
3.5	Proxy . . . . .	29
3.5.1	Functions of a Proxy . . . . .	30
3.5.2	Reverse Proxy . . . . .	30
3.6	Firewall Policies . . . . .	32
3.6.1	Top-Down Policy Application . . . . .	33
3.6.2	Source and Destination IP Fields . . . . .	34
3.6.3	Default Rule Scenario . . . . .	34
3.7	Intrusion Detection and Prevention Systems . . . . .	34
3.7.1	Intrusion Detection System (IDS) . . . . .	34
3.7.2	Intrusion Prevention System (IPS) . . . . .	35
3.8	Network Zoning for Enhanced Security . . . . .	35
3.8.1	Principle of Zoning . . . . .	35
3.8.2	Zoning Implementation . . . . .	36
3.9	Multi-Tier DMZ Structure . . . . .	36

---

3.9.1	Multi-Tier DMZ Structure . . . . .	36
3.10	Encryption: Meaning and Types . . . . .	36
3.10.1	What Does Encryption Mean? . . . . .	36
3.10.2	Main Approaches in Modern Encryption . . . . .	37
3.11	Virtual Private Network (VPN) . . . . .	39
4	Operating Systems . . . . .	41
4.1	Microsoft Windows . . . . .	41
4.1.1	Windows Shell and PowerShell . . . . .	41
4.1.2	Common PowerShell Cmdlets . . . . .	42
4.1.3	Windows File System . . . . .	42
4.1.4	Components of Windows Operating System . . . . .	43
4.1.5	Windows Hybrid Kernel . . . . .	44
4.2	Linux Operating Systems . . . . .	44
4.2.1	Structure of the Linux Operating System . . . . .	45
4.2.2	Types of Kernels . . . . .	46
4.2.3	Processes in Linux . . . . .	46
4.2.4	Linux File System . . . . .	47
4.2.5	User Management in Linux . . . . .	48
5	Programming Languages . . . . .	49
5.1	Introduction . . . . .	49
5.1.1	Classification of Programming Languages . . . . .	49
5.1.2	Types of Translators . . . . .	49
5.1.3	Error Types in Programming . . . . .	50
5.2	C Programming Language . . . . .	50
5.2.1	Main Properties of C Language . . . . .	50
6	Basic Syntax and Commands in C . . . . .	51
6.1	Data Types . . . . .	51
6.1.1	Control Structures . . . . .	51
6.1.2	Functions . . . . .	52
6.1.3	Input/Output Operations . . . . .	52

7	Understanding Pointers, Memory Management, Stacks, and Arrays in C	53
7.1	Pointers . . . . .	53
7.2	Memory Management in C . . . . .	54
7.2.1	Stack . . . . .	54
7.3	Arrays in C . . . . .	57
7.4	Multidimensional Arrays . . . . .	58
8	Computer Hardware and Operating Systems	59
8.1	The von Neumann Architecture . . . . .	59
8.2	Main Components of a PC/Computing Device . . . . .	59
8.2.1	Processor (CPU) . . . . .	59
8.2.2	Motherboard . . . . .	59
8.2.3	Memory (RAM) . . . . .	59
8.2.4	Storage . . . . .	59
8.2.5	Graphics Card (GPU) . . . . .	60
8.2.6	Power Supply Unit (PSU) . . . . .	60
8.2.7	Optical Drive . . . . .	60
8.2.8	Case . . . . .	60
8.2.9	Cooling (Fans and Liquid Cooling) . . . . .	60
8.2.10	Network Interface . . . . .	60
8.2.11	Expansion Ports . . . . .	60
8.2.12	Operating System (OS) . . . . .	60
8.3	Binary System and Data Types . . . . .	61
8.4	Logic Gates . . . . .	61
8.5	Operating System Fundamentals . . . . .	62
8.5.1	The Kernel . . . . .	62
8.6	Operating System Functions . . . . .	62
8.6.1	User Interface Interaction . . . . .	63
8.7	Scenario: System Boot and Application Launch on Windows . . . . .	63
8.7.1	Phase 1: System Boot . . . . .	63
8.7.2	Phase 2: User Interface Interaction . . . . .	64
8.7.3	Phase 3: Application Management . . . . .	64

---

8.8	Overview of Kernel Types in Operating Systems . . . . .	65
8.8.1	Types of Kernels . . . . .	65
8.9	Overview of Operating Systems . . . . .	66
8.9.1	Hardware and Operating System . . . . .	67
9	Process Management . . . . .	67
9.1	Monotasking vs. Multitasking . . . . .	68
9.2	Time-Sharing Systems . . . . .	69
9.3	Preemptive and Cooperative Multitasking . . . . .	69
9.4	Memory Management in Operating Systems . . . . .	70
9.5	Memory Management Module . . . . .	71
9.5.1	Linear Allocation Example . . . . .	71
9.6	Virtual Memory . . . . .	72
9.7	File Systems in Operating Systems . . . . .	72
9.7.1	Key Concepts of File Systems . . . . .	73
9.8	I/O Device Manager . . . . .	74
9.8.1	Driver Functions . . . . .	74
9.9	User Experience . . . . .	75
9.9.1	Textual User Interface . . . . .	75
9.9.2	Graphical User Interface . . . . .	75
9.10	Security in Operating Systems . . . . .	75
9.10.1	Authentication . . . . .	76
9.10.2	Basic Authentication . . . . .	77
9.10.3	Examples of Second Authentication Factors . . . . .	77
9.10.4	Authorization . . . . .	77
9.10.5	Accounting . . . . .	78

## Part I

# Unit 1: Fundamentals of Ethical Hacking

## 1 INTRODUCTION AND VIRTUAL MACHINES

The Cybersecurity Specialist course aims to train professionals in the field of information security. These individuals will possess strong technical skills, providing added value to companies in the fight against cybercrime. The course is divided into three units:

1. **Unit 1** focuses on the theoretical prerequisites and technical skills necessary for an Ethical Hacker. It covers topics such as networking, operating systems, and an introduction to programming;
2. **Unit 2** centers on the phases of Penetration Testing, exploring the tools and techniques used by hackers in the real world;
3. **Unit 3** provides students with a comprehensive understanding of how to monitor security events, manage ongoing attacks, and adopt best practices at the enterprise level to minimize the impact on business activities.

In the past century, computing and the web were primarily the domain of experts, including hackers. The term *hacker*, often associated with digital piracy, encompasses three distinct types of hackers:

- **White Hat Hackers**, also known as "Ethical Hackers," operate with a strict adherence to ethical standards. Their work involves improving security with the consent of the system owner;
- **Grey Hat Hackers** operate in a legal and ethical gray area. They often act without the owner's permission but with the intent of improving security. While their actions can uncover vulnerabilities, they can also be controversial and sometimes illegal;
- **Black Hat Hackers** are criminals who break into computer networks with malicious intent. They may deploy malware to destroy files, steal information, hold computers hostage, or pilfer passwords, credit card numbers, and other personal data. Their motivations are typically opportunistic, such as financial gain. Stolen data is often

sold on the dark web, where items like credit card details, online payment system access, medical records, and even streaming service accounts are traded.

An Ethical Hacker is a cybersecurity expert capable of simulating cyber-attacks to identify potential vulnerabilities in a company's systems. These simulations, known as *Penetration Tests*, are crucial for detecting and fixing security issues in digital networks, software, and devices, thereby protecting enterprises and public entities from cybercriminal activities. Key responsibilities of an Ethical Hacker include:

- Conducting penetration tests on IT infrastructures and web applications;
- Ensuring the security of sensitive and private data, such as payment details, login credentials, and passwords.

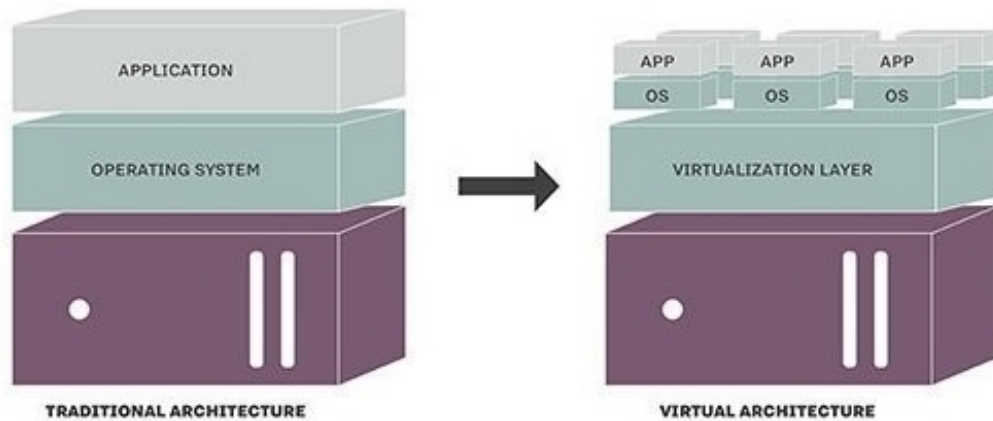
It is crucial to emphasize that penetration testing should only be performed with the formal consent of the system or network owner. Conducting such tests without permission is illegal and can lead to severe legal consequences.

### 1.1 *Virtualization Process*

Virtual machines (VMs) are emulations of computer systems that provide the functionality of a physical computer. They run on a physical host machine and are managed by a virtualization layer. The process of creating and managing VMs is known as virtualization (Figure 1). Virtualization involves several key steps:

1. **Hypervisor Installation:** The hypervisor, also known as the Virtual Machine Monitor (VMM), is installed on the host machine. The hypervisor can be of two types:
  - **Type 1 (Bare Metal):** Runs directly on the host's hardware.
  - **Type 2 (Hosted):** Runs on top of an existing operating system.
2. **Resource Allocation:** The hypervisor allocates resources (CPU, memory, storage, and network) from the physical host to each VM.
3. **VM Creation:** Virtual machines are created by defining their virtual hardware specifications (number of CPUs, amount of memory, disk size, etc.).

## TRADITIONAL AND VIRTUAL ARCHITECTURE



*Figure 1: Traditional vs Virtual architecture.*

4. **Operating System Installation:** An operating system is installed on the virtual machine, just as it would be on a physical machine.
5. **VM Management:** The hypervisor manages the execution of VMs, handles resource allocation dynamically, and ensures isolation between VMs.

### 1.1.1 Virtual Hardware

A virtual machine emulates physical hardware components such as:

- **CPU:** Virtual CPUs (vCPUs) are assigned to the VM by the hypervisor.
- **Memory:** Virtual memory is allocated from the host's physical memory.
- **Storage:** Virtual disks are created using files on the host's storage system.
- **Network:** Virtual network interfaces connect VMs to the host's network and to other VMs.

### 1.1.2 Hypervisor Role

The hypervisor plays a critical role in:



- **Resource Management:** Dynamically allocating resources to VMs based on their needs.
- **Isolation:** Ensuring that each VM operates independently and securely.
- **Efficiency:** Optimizing resource usage to maximize the performance of VMs.

### 1.1.3 Execution Flow

The execution flow of a VM includes:

1. **Boot Process:** The VM goes through a boot process similar to a physical machine, loading its operating system from the virtual disk.
2. **Application Execution:** Applications run within the VM, utilizing the virtualized hardware.
3. **Hypervisor Interaction:** The VM interacts with the hypervisor for tasks like I/O operations, which the hypervisor translates to the physical hardware.

### 1.1.4 Benefits of Virtualization

Virtualization offers several advantages:

- **Cost Efficiency:** Reduces the need for physical hardware.
- **Scalability:** Easily create, modify, and delete VMs as needed.
- **Isolation:** Provides a secure and isolated environment for each VM.
- **Resource Utilization:** Maximizes the usage of physical resources.

Virtualization is a powerful technology that enables the efficient use of hardware resources, providing flexibility, scalability, and isolation. Virtual machines are an integral part of modern computing environments, supporting a wide range of applications from development to production systems.

## 2 NETWORKING

### 2.1 *Network Categories: Geographical and Topological*

#### 2.1.1 Geographical Networks

Networks can be categorized based on the geographical distance between devices. The primary types of geographical networks include:

**WAN - WIDE AREA NETWORK** Wide Area Networks (WANs) connect computers over large distances, covering extensive geographical areas. WANs use various transmission methods such as satellites, fiber optics, and cables. The quintessential example of a WAN is the Internet, which allows computers in different continents to communicate within seconds.

**LAN - LOCAL AREA NETWORK** Local Area Networks (LANs) connect computers within a smaller geographical area, such as the floors of a building in a company. LANs are popular due to their high speed and relative ease of installation.

**PAN - PERSONAL AREA NETWORK** Personal Area Networks (PANs) cover very limited areas, such as the connection between a mobile phone and a computer.

#### 2.1.2 Topological Networks

Networks can also be categorized based on their physical configuration. The primary types of topological networks include the following, also shown in Figure 2.

**BUS TOPOLOGY** In a bus topology, devices share a single transmission channel, typically a single cable. One of the advantages of bus topology is its simplicity and ease of implementation.

**RING TOPOLOGY** In a ring topology, devices are connected in a circular manner. These networks are more complex and expensive than bus networks and are generally used in enterprise environments for corporate LANs.

## Network topology

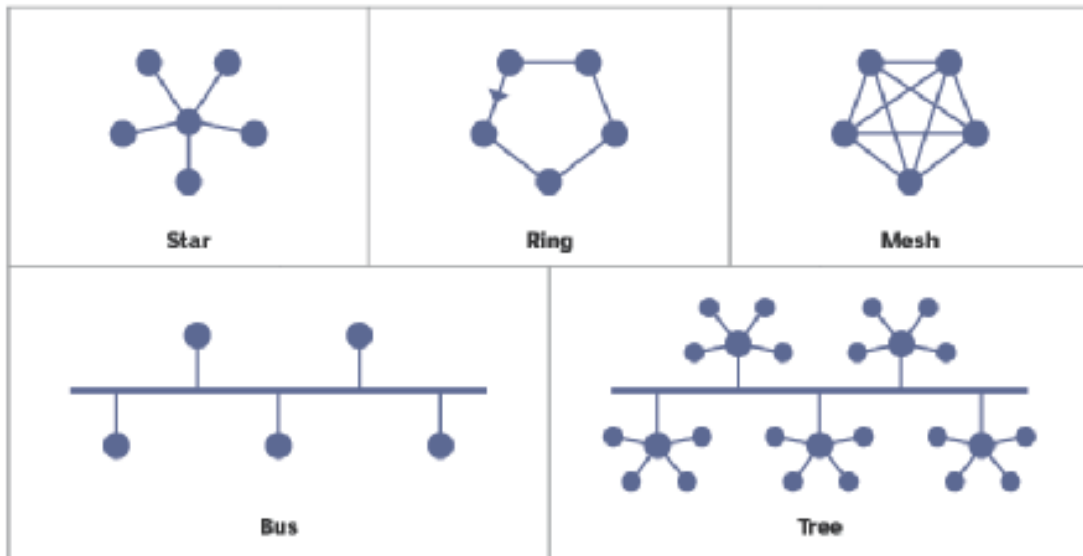


Figure 2: Network topologies.

**STAR TOPOLOGY** In a star topology, devices are individually connected to a central device, such as a switch or router, using dedicated cables or other physical media. Each computer has a dedicated link to the central device and can communicate with other devices through it. Star networks are the most widely used due to their performance and security characteristics.

Understanding the different types of networks, both geographical and topological, is essential for designing and implementing efficient and secure network infrastructures. WANs, LANs, and PANs cater to different geographical scopes, while bus, ring, and star topologies offer various configurations to meet specific organizational needs.

### 2.2 Basic Networking Concepts

A computer network enables and facilitates communication between people, applications, and servers regardless of their geographical location. The Internet is the largest example of a computer network that we use on a daily basis. In a computer network, machines communicate with each other using communication *protocols*, which ensure communication between computers with different hardware and software.

### 2.2.1 Data Transmission

Communication occurs through an exchange of data, or information, which is transported in the form of *packets*. Packets are streams of bits exchanged via electrical signals over a physical medium. The physical medium can be a LAN (local area network) cable or, very commonly, the air in a Wi-Fi network.

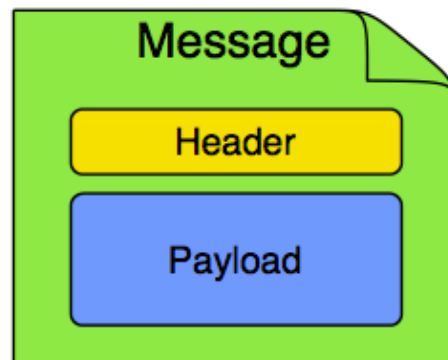


Figure 3: Structure of a Packet: Header and Payload.

### 2.2.2 Packet Structure

A *packet* has a defined structure as shown in Figure 3. The *Header* depends on the protocol used in the communication. Its task is to ensure that the receiving computer can interpret the *Payload* and manage the communication. The *Payload*, on the other hand, is the actual information. For instance, it could be a message, or part of a message sent from one computer to another, an email, or other data.

## 2.3 The ISO/OSI Model

To standardize network communication, in 1984 the International Organization for Standardization (ISO) published a theoretical model subsequently called the *Open System Interconnection (OSI)* model. The ISO/OSI model is used as a theoretical reference. It is based on a stack of 7 layers, also called *layers*, where each layer serves the upper layer and has exclusive protocols.

In a communication between two computers, the data follows the logical model as represented by the line in Figure 5.

	Layer	Description
7	Application	Enables the end user applications to access the network
6	Presentation	Converts data into an understandable format and encrypts it
5	Session	Enables and manages sessions of communication between computers
4	Transport	Ensures reliable transfer of packets of data between users
3	Network	Determines the transmission path of data using routing protocols
2	Data Link	Formats data on the network and sends it from node to node
1	Physical	Manages the relationship between the physical device and the transmission medium, be it wireless or cable

Figure 4: The ISO/OSI Model Layers.

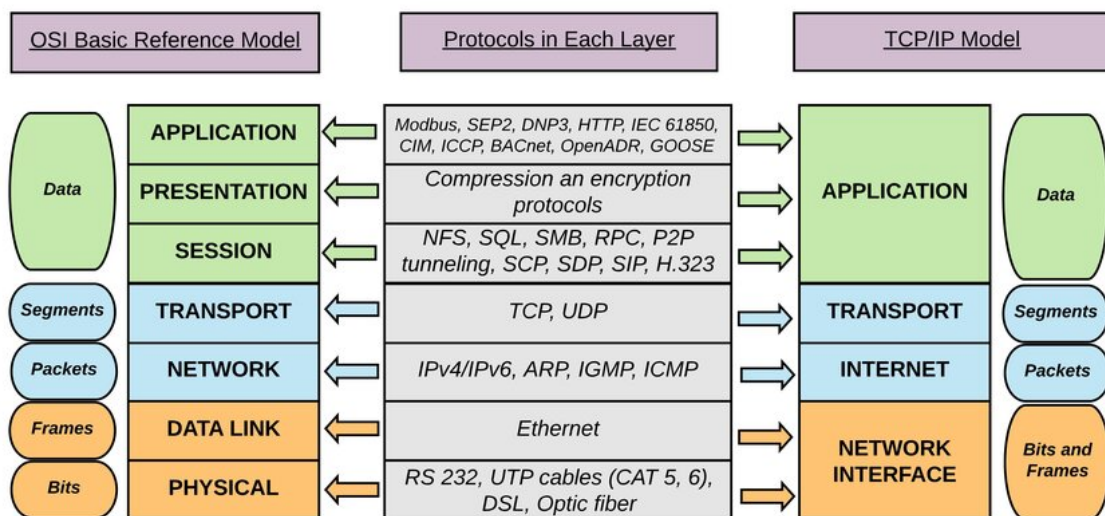


Figure 5: The ISO/OSI vs TCP/IP model.

Note that the ISO/OSI model is a theoretical model to be used as a reference. In practice, applications use the *TCP/IP* model, which slightly varies from the theoretical ISO/OSI model.

The differences between TCP/IP and the ISO/OSI model are depicted in Figure 5 and summarized in Figure 6. For theoretical study of concepts, we will follow the notions of

the ISO/OSI model.

OSI Model	TCP/IP Model
It is developed by ISO (International Standard Organization)	It is developed by ARPANET (Advanced Research Project Agency Network).
OSI model provides a clear distinction between interfaces, services, and protocols.	TCP/IP doesn't have any clear distinguishing points between services, interfaces, and protocols.
OSI refers to Open Systems Interconnection.	TCP refers to Transmission Control Protocol.
OSI uses the network layer to define routing standards and protocols.	TCP/IP uses only the Internet layer.
OSI follows a vertical approach.	TCP/IP follows a horizontal approach.
OSI layers have seven layers.	TCP/IP has four layers.
In the OSI model, the transport layer is only connection-oriented.	A layer of the TCP/IP model is both connection-oriented and connectionless.
In the OSI model, the data link layer and physical are separate layers.	In TCP, physical and data link are both combined as a single host-to-network layer.
Session and presentation layers are a part of the OSI model.	There is no session and presentation layer in the TCP model.
It is defined after the advent of the Internet.	It is defined before the advent of the internet.
The minimum size of the OSI header is 5 bytes.	The minimum header size is 20 bytes.

*Figure 6: The ISO/OSI vs TCP/IP model.*

### 2.3.1 Physical Layer

The physical layer is responsible for the transmission of raw data bits over a physical medium. This transmission can occur via various types of cabling, such as copper wires or fiber optics. Data from higher layers of the source computer is segmented and sent to the physical layer of the receiving computer in the form of bits.

**BITS AND BINARY SYSTEM** The bit, short for binary digit, is the basic unit of information in computing, represented as either '0' or '1'. Unlike the decimal system, which uses ten digits (0-9), computers operate using the binary system. Consequently, information is

transmitted over a physical cable as a sequence of binary digits, for example, 10010010 10011110, etc.

### 2.3.2 Data Link Layer

The Data Link Layer utilizes the services of the physical layer to send and receive bits over communication channels. Packets at this layer are referred to as frames. The main functions of the Data Link Layer include:

- Providing an interface to the Network Layer (Layer 3)
- Handling transmission errors
- Regulating the flow of bits between two communicating devices

The Data Link Layer plays a crucial role by defining protocol standards such as IEEE 802.3 for Ethernet (wired connections) and IEEE 802.11 for Wireless connections.

**MAC ADDRESS** In a computer network, two personal computers communicate at the Data Link Layer using what is called a physical address or more commonly, a MAC Address. The MAC address is a 48-bit identifier typically represented in hexadecimal format. Below is a table showing the conversion between binary and hexadecimal systems:

An example of a MAC Address is 00:AA:11:BB:22:CC. The MAC address is a unique identifier assigned to the network interfaces of devices.

**ETHERNET FRAME STRUCTURE** The structure of an Ethernet 802.3 frame includes fields for the Destination MAC Address and the Source MAC Address. These fields are automatically populated with the MAC addresses of the source and destination when an exchange of information begins at the Data Link Layer.

**ROLE OF SWITCHES** Switches are vital network devices for communication but come with certain disadvantages. They route broadcast packets across the entire network. Broadcast packets are sent to a specific MAC address, FF:FF:FF:FF:FF:FF, which is received by the switch and routed to all nodes connected to it. This creates what is known as a broadcast domain.

Binary	Hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

*Table 1: Binary to Hexadecimal Conversion*

A broadcast domain includes all the devices on a network segment that receive broadcast frames from any device within the segment. In a large network with numerous devices, this can cause significant latency issues.

**ADDRESS RESOLUTION PROTOCOL (ARP)** ARP, or Address Resolution Protocol, is a network protocol used to map IP addresses to MAC addresses within a local network. Its primary functions include:

- **ARP Request:** A device broadcasts an ARP request on the local network, asking for the MAC address associated with a specific IP address.
- **ARP Reply:** The destination device responds with a packet containing its MAC address associated with the requested IP address.
- **ARP Cache:** Devices maintain an ARP table or ARP cache that stores IP-to-MAC address mappings.

ARP is crucial for communication within the same local network. When a device needs to send data to another device within the same subnet, it uses ARP to discover the MAC address corresponding to the destination IP address.



**VIRTUAL LAN (VLAN)** Switches allow the segmentation of broadcast domains through the creation of Virtual LANs (VLANs). A VLAN is a logical grouping of hosts and network devices. VLANs are created by adding a "tag" or VLAN ID to the switch ports, or by mapping the MAC address of the host to the VLAN ID.

Assume we want to restrict communication and the broadcast domain to hosts A, B, and C. We create a VLAN identified by ID 100 and associate the MAC addresses of hosts A, B, and C with that VLAN.

When host A sends a packet to the broadcast address, it will only be received by hosts B and C. Therefore, VLANs segment broadcast domains and eliminate latency issues in large networks.

VLANs are essential for managing broadcast domains within a network. By associating specific MAC addresses with VLAN IDs, we can ensure that broadcast traffic is limited to only those hosts within the same VLAN, thus improving network efficiency and reducing latency.

### 2.3.3 Network Layer

In networking, just as there are specific devices at the Data Link Layer that facilitate communication between personal computers on the same network, there are devices at the Network Layer, such as router-gateways, that enable data routing between personal computers connected to different networks. For instance, consider the following network configuration:

**ROLE OF THE ROUTER** A switch operates at Layer 2 and does not route packets to other networks because it forwards packets based on MAC addresses rather than IP addresses. The solution is to use a Layer 3 device like a router-gateway.

The router receives the packet from the switch and checks its routing table to determine which of its interfaces to forward the packet to reach the destination network.

**EXAMPLE OF ROUTER OPERATION** To better understand how a router-gateway functions, consider a router connected with three interfaces to three different networks:

If a personal computer on network 2.228.1.0 wants to send a packet to a PC on network 2.175.1.0, the packet will be sent to the router, which will check its routing table to see

which interface to use to forward the packet.

In a network, computers interact using both MAC and IP addresses. For example, if PC A wants to send a packet to PC B, it will structure the packet as follows:

- The destination IP address of B in the datagram header (Layer 3).
- The MAC address of the router as the destination in the frame header (Layer 2). The router, in this case, is the "next hop."
- Its IP address as the source in the datagram header.
- Its MAC address as the source in the frame header.

The router will receive the packet and set:

- The destination MAC address to that of B.
- The source MAC address to that of its corresponding interface.

Routers play a crucial role in network communication by enabling the routing of data between different subnets. This process involves intricate interactions between Layer 2 (MAC addresses) and Layer 3 (IP addresses) addressing schemes, ensuring seamless communication across diverse network segments.

**SUBNETTING** Subnetting is the process of dividing a larger network into smaller subnets. This helps in efficient IP address management, improves network performance, and enhances security.

**Classless Inter-Domain Routing (CIDR)** is a method for allocating IP addresses and routing. CIDR notation specifies an IP address along with a routing prefix.

Consider an IP address and subnet mask:

IP Address : 192.168.1.0

Subnet Mask : 255.255.255.0

In CIDR notation, this is written as:

192.168.1.0/24

Let's divide this network into smaller subnets. For example, if we want to create four subnets, we need to borrow 2 bits from the host part of the address. The new subnet mask will be:

255.255.255.192 or /26

This gives us four subnets:

Subnet 1: 192.168.1.0/26 (Hosts: 192.168.1.1 - 192.168.1.62)

Subnet 2: 192.168.1.64/26 (Hosts: 192.168.1.65 - 192.168.1.126)

Subnet 3: 192.168.1.128/26 (Hosts: 192.168.1.129 - 192.168.1.190)

Subnet 4: 192.168.1.192/26 (Hosts: 192.168.1.193 - 192.168.1.254)

Each subnet provides 62 usable IP addresses (64 total addresses minus 2 for network and broadcast addresses).

#### 2.3.4 Transport Layer

The transport layer, also known as layer 4 in the OSI model, is responsible for establishing a logical connection or channel between applications on different computers. It ensures the reliable transmission of data packets between a source and a destination. Packets may get lost during transmission due to network congestion, communication errors, or network issues. In some scenarios, minor transmission issues are tolerable, such as in phone conversations or video streaming, where a lost packet might result in a brief glitch. However, for activities like financial transactions or logging into a portal, it's crucial that communication is lossless.

**PROTOCOLS IN THE TRANSPORT LAYER** The transport layer provides two fundamental protocols for communication between hosts:

- **TCP (Transmission Control Protocol):** TCP guarantees data traffic control and reliable delivery to the recipient, making it more secure for applications requiring

guaranteed packet delivery. The reliability of TCP is due to its connection-oriented nature, which establishes a communication channel before data exchange. TCP follows a three-step process called the *three-way handshake* to establish this connection.

1. The client initiating the connection sends a TCP packet with the SYN flag set and a random sequence number.
  2. The server responds with a packet that has both SYN and ACK flags set, along with its own sequence number and an acknowledgment number equal to the client's sequence number plus one.
  3. The client completes the synchronization by sending a packet with the ACK flag set, acknowledging the server's sequence number.
- **UDP (User Datagram Protocol):** UDP is a connectionless protocol that does not require establishing a communication channel before data exchange. It is more lightweight and faster, suitable for activities requiring data streaming (e.g., video or audio). However, it does not guarantee packet delivery.

**PORTS AND SERVICES** To identify the target process or service for a given packet, both TCP and UDP use ports, represented as `<ip>:<port>`. While the IP identifies the destination machine, the port provides information about the service. Ports can be categorized into:

- **Well-known ports:** Used for standard services, ranging from 0 to 1023.
- **High ports:** Used for non-standard services, ranging from 1024 to 65535.

Protocol	Port
SMTP	25
SFTP	115
HTTP	80
HTTPS	443
SSH	22
Telnet	23
POP3	110
FTP	21

Table 2: Commonly Used Ports

Understanding the transport layer is crucial for ensuring reliable communication between applications on different hosts. TCP and UDP provide different levels of service,

making them suitable for various applications. TCP's connection-oriented nature and reliable delivery make it ideal for critical applications, while UDP's connectionless nature and speed make it suitable for real-time data streaming.

### 2.3.5 Session Layer

In order to enable communication between two hosts, or between a client and a server, it is necessary to first establish a session. This is precisely the role of the protocols that are part of the session layer. Sessions are essential to ensure the correct transfer of information. It is very important to define the rules for opening, closing, or maintaining a session. We can identify two main purposes of the session layer:

- Definition and management of the session:
  - **Initiating or opening a session** between a user who intends to use a service that is listening on a specific server. A common example is the *SSH* (Secure Shell) protocol, which is used by system administrators to create sessions on remote servers.
  - **Maintaining the session** for its duration, ensuring the session remains active during the flow of information.
  - **Closing the session**, either upon the request of the user or the server.
- Synchronization: saving intermediate checkpoints (or synchronization points) during a data flow. This allows for the preservation of information in the event of an abnormal session interruption.

### 2.3.6 Presentation Layer

The Presentation Layer is the sixth level in the OSI (Open Systems Interconnection) model. This layer is responsible for the preparation of data in transit between two hosts before it is presented to the users. One of the critical functions of the Presentation Layer is **data encryption**. In a communication channel, data can transit in one of two ways:

- **In clear text:** Data is transmitted in a visible and readable form to everyone.
- **Encrypted:** Data is encoded such that only authorized parties can view the content.

Encryption transforms clear text data into encrypted data using what is known as an **encryption algorithm**. The primary purpose of encryption is to ensure that the content of the information is available only to specific users, thereby securing the data from potential malicious actors.

The Presentation Layer performs several critical functions, including:

1. **Translation:** This layer translates data between the application layer and the network format. It ensures that data from the sender's application layer can be understood by the receiver's application layer.
2. **Data Encryption and Decryption:** As mentioned, encryption is a fundamental aspect of the Presentation Layer. It encodes the data before it is transmitted over the network and decodes it upon arrival.
3. **Data Compression:** The Presentation Layer can compress data to reduce the bandwidth needed for transmission. This can lead to faster transmission times and reduced network load.
4. **Character Code Translation:** It translates character codes from one coding system to another. For example, it can convert ASCII to EBCDIC.
5. **Data Serialization:** This process involves converting complex data structures into a format that can be easily transmitted over the network.

**Encryption** Encryption is a process that involves transforming readable data, known as **plaintext**, into an unreadable format, known as **ciphertext**. This transformation uses an algorithm and an encryption key. Only authorized parties who possess the corresponding decryption key can convert the ciphertext back into readable plaintext.

Encryption serves several essential purposes:

- **Confidentiality:** Ensures that data is only accessible to those who have the appropriate decryption key.
- **Integrity:** Protects data from being altered during transmission.
- **Authentication:** Verifies the identities of the communicating parties.

- **Non-repudiation:** Prevents parties from denying the transmission or receipt of data.

The Presentation Layer plays a crucial role in ensuring that data is properly encrypted and decrypted, thereby maintaining the security and integrity of the information being transmitted across networks.

### 2.3.7 Application Layer

The application layer is the seventh and final layer of the ISO/OSI model. This layer interacts directly with the applications used by the end user, providing interface services for applications and support for network access. Among the most well-known protocols of the application layer are:

- **HTTP/HTTPS (Hyper Text Transfer Protocol):** This is the main protocol for transmitting information on the web. HTTPS is the secure, encrypted version of HTTP. The protocol operates on a request/response mechanism, typical of the client/server model, where a host requests a resource from a remote web server.
- **DNS (Domain Name System):** This protocol translates domain name requests (e.g., `www.google.com`) into IP addresses. DNS is a fundamental protocol for the functioning of the internet. A DNS name like `www.store.google.com` can be divided as follows:
  - `www` - Host
  - `store` - Subdomain
  - `google` - Domain
  - `com` - Top-Level Domain (TLD)

The resolution of DNS names is handled by DNS resolvers, which are DNS servers generally provided by ISPs (Internet Service Providers), such as Vodafone, TIM, or Wind3. Public DNS servers, like Google's DNS, are also available.

- **FTP (File Transfer Protocol):** This protocol manages the transfer of data between hosts and is based on TCP.

- **DHCP (Dynamic Host Configuration Protocol):** This protocol allows devices on a LAN to receive network configuration automatically. An host connecting to a network where the DHCP service is active will have its IP address and other network information auto-assigned. For example, when you connect to your home network, you likely don't manually assign an IP address to your PC or smartphone. This is because DHCP is typically enabled on private networks for automatic IP assignment.

The application layer provides crucial functionalities that directly support user applications and network services. Here are some key points:

- **User Interface Support:** This layer offers services that directly interface with user applications, enabling seamless communication and data transfer over the network.
- **Protocol Implementation:** Protocols at this layer define the rules for communication between different devices and software applications. These protocols include not only HTTP, DNS, FTP, and DHCP but also others like SMTP (Simple Mail Transfer Protocol), POP3 (Post Office Protocol), and IMAP (Internet Message Access Protocol), which are used for email transmission and retrieval.
- **Resource Sharing:** It facilitates resource sharing across the network, such as file transfers, web pages, and email.
- **Network Transparency:** It provides a way for applications to interact with the network in a manner that abstracts the complexities of lower-layer operations, making network interactions more user-friendly and efficient.

The application layer is indispensable for ensuring that user applications can operate over the network effectively, providing the necessary protocols and services to support diverse network-based activities.

### 2.4 Network Address Translation (NAT)

Network Address Translation (NAT) is a technology developed to address the exhaustion of IPv4 addresses. Introduced in the 1990s, it delineates the distinction between private and public IP addresses. NAT is typically configured on routers or firewalls.



A public IP address allows devices to access the Internet, whereas private IP addresses are used within local networks (e.g., home or corporate networks) and are not directly accessible from the Internet. There are various types of NAT, but the basic concept involves translating private IP addresses (used within the local network) into one or more public IP addresses.

### 2.5 *Port Address Translation (PAT)*

Port Address Translation (PAT) is a specific form of NAT. In addition to translating IP addresses, PAT also translates the port numbers of devices within the local network. Ports are 16-bit numbers associated with each connection, and PAT enables multiple devices to share the same public IP address by distinguishing their connections based on port numbers.

For instance, if you have three devices using PAT and sharing the same public IP address, the first device's connection might use port 5000, the second device might use port 5001, and the third device might use port 5002. This allows the router to keep track of which connection belongs to which device on the local network.

In summary, while NAT translates private IP addresses into one or more public IP addresses, PAT goes further by also translating ports. This enables multiple devices to share the same public IP address, distinguishing them based on the ports used. Both technologies are fundamental for managing the scarcity of public IP addresses and enabling Internet connectivity for multiple devices within a local network.

**PRIVATE IP ADDRESS RANGES** The following table lists the ranges of IP addresses defined as "private." These IP addresses are assigned only within private networks and are not exposed or reachable via the Internet.

IPv4 Address Classes and Ranges						
Address Class	Type	Range	Default Subnet Mask	Number of Networks	No of Hosts Per Network	Use
A	Public	1.0.0.0 to 127.0.0.0	255.0.0.0	126	16,777,214	Governments and Large Number of Hosts
	Private	10.0.0.0 to 10.255.255.255				
B	Public	128.0.0.0 to 191.255.255.255	255.255.0.0	16,382	65,534	Medium Companies
	Private	172.16.0.0 to 172.31.255.255				
C	Public	192.0.0.0 to 223.255.255.255	255.255.255.0	2,097,150	254	Small Companies and LANs
	Private	192.168.0.0 to 192.168.255.255				
D	N/A	224.0.0.0 to 239.255.255.255	Not Applicable	N/A	N/A	Reserved for Multicasting
E	N/A	240.0.0.0 to 254.255.255.255	Not Applicable	N/A	N/A	Experimental
Special	Special	127.0.0.1 to 127.255.255.255	N/A	N/A	N/A	Loopback Testing

**Note:**

- Addresses 127.0.0.1 to 127.255.255.255 cannot be used and are reserved for loopback testing

- APIPA address range is 169.254.0.1 to 169.254.255.254 and has 65,534 usable IP addresses, with the subnet mask of 255.255.0.0.

*Figure 7: IP addresses types.*

---

### 3 FIREWALL & CRYPTOGRAPHY

A firewall is a crucial component in the realm of cybersecurity. It acts as a barrier that protects a network or system from external threats by managing and filtering incoming and outgoing network traffic. Essentially, a firewall serves as a gatekeeper between an internal network and the broader internet, analyzing network traffic and determining whether to allow or block data packets based on pre-established security rules.

#### 3.1 *Key Classifications of Firewalls*

##### 1. Firewall Types by Implementation:

- **Hardware Firewalls:** These are physical devices dedicated to the task of network security. They are typically used by businesses and organizations with significant network traffic and complex security needs.

*Advantages:* high performance due to dedicated resources, capable of handling large volumes of traffic, centralized security management.

*Disadvantages:* higher cost and maintenance ,complexity in configuration and deployment.

- **Software Firewalls:** These are programs installed on individual computers or servers. They provide a flexible and often more economical solution, especially for personal use or smaller networks.

*Advantages:* cost-effective and easy to install, flexible and can be customized for specific needs, useful for personal devices and small businesses.

*Disadvantages:* consumes system resources, which can affect performance, requires individual management on each device.

##### 2. Firewall Placement:

- **Perimeter Firewalls:** Positioned at the boundary of a network, perimeter firewalls are designed to safeguard the internal network from external threats. They act as the first line of defense, regulating access between the internal network and the internet or other untrusted networks.

*Advantages:* provides a robust first line of defense, centralized point of security control for the entire network.

*Disadvantages:* cannot protect against threats that bypass the perimeter, such as internal attacks.

- **Host-based Firewalls:** These are installed on individual devices within the network. They monitor and control traffic to and from the device on which they are installed, providing an additional layer of security.

*Advantages:* adds an extra layer of protection for individual devices, useful for controlling local traffic and internal threats.

*Disadvantages:* requires installation and management on each individual device, potentially increases the complexity of the overall security infrastructure.

## 3.2 Types of traffic filtering

Firewalls implement various types of traffic filtering to control which data packets can pass through or be blocked. Each type of firewall and filtering mechanism offers different levels of security and control over the network traffic.

### 3.2.1 Static Packet Filtering

Static packet filtering is a type of network traffic filtering where decisions to permit or block traffic are based on static criteria, such as IP addresses, source and destination ports, and protocols.

- **Static Rules:** The firewall is configured with static rules created by the network administrator. These rules specify the criteria based on which the firewall evaluates the traffic.
- **Traffic Evaluation:** When the firewall receives a data packet, it compares the packet against the static rules. For example, it can check if the source and destination IP addresses are allowed, if the destination port is authorized, and if the protocol used is permitted.
- **Blocking or Permitting Decisions:** Based on these static rules, the firewall decides whether to permit or block the packet. If the packet matches the specified rules, it is

allowed through; otherwise, it is blocked.

### 3.2.2 Stateful Filtering

Stateful filtering is an advanced type of filtering. Its distinguishing feature is the ability to track the state of network connections, enabling the firewall to make decisions based on contextual information about the connection.

- **Initiated Connections:** A stateful firewall keeps track of connections initiated from the internal network and allows outgoing traffic associated with these connections.
- **Established Connections:** After detecting an initial connection, the stateful firewall maintains a list of established connections. This list contains information such as source and destination IP addresses, ports, and the current state of the connection.
- **Subsequent Packets:** When the firewall receives subsequent packets that are part of a previously established connection, it compares them with the connection information and decides whether to permit or block them. For example, if a packet is part of an established connection, it will usually be permitted.

## 3.3 *Web Application Firewall (WAF)*

A Web Application Firewall (WAF) is a cybersecurity component specifically designed to protect web applications from various online threats and attacks. This tool focuses on the application layer, analyzing incoming and outgoing web traffic to identify and block suspicious or dangerous activities.

## 3.4 *Next-Generation Firewall (NGFW)*

A Next-Generation Firewall (NGFW) is an advanced cybersecurity solution that combines traditional firewall functions with other advanced features and deep traffic inspection capabilities to provide more sophisticated protection against cyber threats.

## 3.5 *Proxy*

A proxy, or proxy server, is an intermediary server between a client (e.g., a computer or device) and a server the client wants to access. The proxy acts as a middleman between the

client and the destination server, forwarding the client's requests and returning responses from the server.

#### 3.5.1 Functions of a Proxy

- **IP Address Hiding:** A proxy can hide the client's IP address from the destination server. When the client sends a request through the proxy, the proxy's IP address appears as the sender of the request to the server.
- **Content Filtering:** Some proxies are configured to filter traffic based on certain rules. For example, they can block access to specific websites or limit access to certain types of content.
- **Caching:** Proxies can locally store responses to client requests. This process allows the proxy to return responses to requests without having to forward them to the destination server, improving efficiency and speed.
- **Anonymity and Security:** Some proxies offer a degree of anonymity and security. For example, a proxy can hide the client's identity, making it more difficult for websites to track the user.
- **Remote Access:** Proxies can be used to allow remote access to internal network resources while protecting the security of the network.

#### 3.5.2 Reverse Proxy

A reverse proxy is a type of proxy server that sits between client devices and a web server, handling requests from clients on behalf of the web server. It acts as an intermediary, forwarding client requests to the appropriate backend server and then returning the server's response to the client. Here is a detailed explanation of its functioning:

##### 1. Client Request Handling:

- When a client sends a request to access a web application or resource, the request is first received by the reverse proxy server instead of the actual web server.

- The client perceives the reverse proxy as the actual server, not knowing that their request is being handled by an intermediary.

## **2. Request Forwarding:**

- The reverse proxy examines the client request and determines which backend server is best suited to handle the request. This determination can be based on various factors, such as load balancing policies, server health, or specific application logic.
- After identifying the appropriate backend server, the reverse proxy forwards the client request to this server.

## **3. Response Handling:**

- Once the backend server processes the request, it sends the response back to the reverse proxy.
- The reverse proxy then sends this response to the client, making it appear as though the response originated from the reverse proxy itself.

## **4. Load Balancing:**

- One of the key functions of a reverse proxy is to distribute incoming client requests across multiple backend servers to ensure no single server becomes overwhelmed with traffic. This process is known as load balancing.
- Load balancing can be implemented using various algorithms, such as round-robin, least connections, or IP hash, to effectively distribute the traffic load.

## **5. Caching:**

- A reverse proxy can cache responses from backend servers. When subsequent requests for the same resource are received, the reverse proxy can serve the cached response instead of forwarding the request to the backend server, reducing latency and server load.

## **6. SSL Termination:**

- Reverse proxies often handle SSL termination, which involves decrypting incoming SSL/TLS connections and forwarding the decrypted requests to the backend servers. This offloads the encryption/decryption overhead from the backend servers, improving their performance.

#### 7. Security and Anonymity:

- Reverse proxies enhance security by hiding the identity and structure of the backend servers. Clients interact only with the reverse proxy, making it difficult for attackers to target the actual backend servers.
- They can also implement additional security measures such as Web Application Firewall (WAF) functionality, filtering malicious requests, and protecting backend servers from attacks.

#### 8. Compression:

- Reverse proxies can compress responses from backend servers before sending them to clients. This reduces the amount of data transmitted over the network, improving load times and reducing bandwidth usage.

#### 9. Monitoring and Logging:

- Reverse proxies can monitor and log traffic between clients and backend servers, providing valuable insights into performance, usage patterns, and potential security issues. These logs can be used for troubleshooting, capacity planning, and security auditing.

Overall, a reverse proxy acts as a mediator between clients and backend servers, enhancing performance, security, and scalability while simplifying the client-server interaction.

### 3.6 Firewall Policies

A firewall operates at various levels of the ISO/OSI model, providing diverse functionalities and security measures. The primary function of the firewall is to filter incoming or outgoing packets based on established rules known as firewall policies. These policies can filter packets based on:



- **Source or Destination IP Address**
- **Destination Protocol or Port**
- **Geolocation** (origin of the request)
- **Application Used**
- **Type of Client**

When a firewall inspects a packet, it can decide how to handle it with the following actions:

- **Allow:** The firewall lets the packet pass.
- **Drop:** The firewall discards the packet without sending any diagnostic message to the source.
- **Deny:** The firewall blocks the packet and informs the source.

These actions are specified in the firewall policies. For example, a policy might drop a packet directed to `google.com` on port 443 (HTTPS).

### 3.6.1 Top-Down Policy Application

Firewalls apply policies within the policy set using a top-down approach. For a given communication between a source and a destination, the firewall searches the policy set for a rule that manages the traffic. Once found, the firewall stops searching.

Source IP	Destination IP	Port	Action
192.168.1.15	10.10.10.10	443	DENY
192.168.1.24	10.11.11.12	80, 53	ACCEPT
192.168.23.40	10.10.11.11	0-1023	ACCEPT
192.168.23.40-41	192.168.33.54	443, 444, 445	DENY
10.10.10.0/24	Group-ip-microsoft	443, 1234, 998	DROP
Object-google-ip	10.10.10.0/24	High-ports-group	DENY
ANY	ANY	ANY	DENY

POLICY EVALUATION EXAMPLE Consider the following example:

- The firewall intercepts a flow and examines the policy set to determine the appropriate action.
- It first checks the policy set's top rule. If it doesn't match the flow, the firewall moves to the next rule.
- If the third rule includes the traffic flow, the firewall handles the flow according to the "ACTION" parameter, allowing the traffic to pass ("ACCEPT").

#### 3.6.2 Source and Destination IP Fields

The source and destination IP fields can include:

- Multiple IP addresses
- Subnets
- Groups / Objects

If the firewall does not find any rules that manage the flow, it discards the flow using a default rule. This rule, present in all policy sets and not modifiable, rejects all communications that do not match any other rule.

#### 3.6.3 Default Rule Scenario

**What happens if this default policy is placed at the top of the policy set?**

If the default deny rule is the first rule evaluated by the firewall, all traffic will be blocked because it matches the "ANY ANY ANY DENY" rule. This setup would effectively prevent all communications through the firewall.

### 3.7 *Intrusion Detection and Prevention Systems*

#### 3.7.1 Intrusion Detection System (IDS)

An Intrusion Detection System (IDS) is a cybersecurity tool designed to detect and report suspicious activities or intrusions in networks or computer systems. It continuously monitors network traffic, system logs, and other events to identify anomalies that may indicate a security threat.

- **Traffic Analysis:** IDS analyzes network traffic or system logs to identify known attack signatures, anomalous behaviors, or policy violations.
- **Alert Generation:** When a potential threat is detected, IDS generates alerts or notifications for security administrators to take appropriate actions.
- **Passive System:** IDS does not take direct action to stop an attack; it only provides alerts.

### 3.7.2 Intrusion Prevention System (IPS)

An Intrusion Prevention System (IPS) is a cybersecurity tool that, unlike an IDS, can take active measures to block or prevent detected attacks. IPS operates in real-time to immediately interrupt malicious or unwanted activities.

- **Real-Time Response:** IPS not only detects threats but also takes proactive actions to stop them, such as blocking suspicious traffic or disconnecting users.
- **Preventive Actions:** When a threat is detected, IPS can block traffic, issue alerts, or take other actions to prevent the attack from causing damage.
- **Active System:** The primary goal of IPS is to prevent cyber attacks from harming the system or network before any damage occurs. However, careful configuration is required to avoid false positives, which could block legitimate traffic.

## 3.8 *Network Zoning for Enhanced Security*

After discussing some network security devices, let's explore a commonly used technique to significantly enhance network security: zoning. This technique involves dividing the network into different zones.

### 3.8.1 Principle of Zoning

- **Application Area:** A zone dedicated to applications.
- **User PCs Area:** A zone dedicated to user PCs.
- **Admin PCs Area:** A zone dedicated to administrators' PCs.

Clearly, a network can have areas or zones that require higher levels of security due to their sensitivity.

#### 3.8.2 Zoning Implementation

- **Segregation:** Networks are segmented into zones based on asset criticality and the required security level.
- **Security Levels:** Different zones have varying security measures, ensuring that sensitive areas have stricter controls.

### 3.9 *Multi-Tier DMZ Structure*

Cyber threats primarily originate from the internet, prompting many organizations to implement a multi-tier DMZ (Demilitarized Zone) network structure. This setup involves:

#### 3.9.1 Multi-Tier DMZ Structure

- **Zone Division:** The network is divided into zones based on the criticality of assets on each segment.
- **Multiple Security Layers:** Additional security layers, such as firewalls, proxies, or other security devices, are added.

### 3.10 *Encryption: Meaning and Types*

#### 3.10.1 What Does Encryption Mean?

Encryption refers to the process of converting data or a message into an unreadable or unintelligible format unless one possesses a specific key or method to decrypt it. The main goal of encryption is to protect sensitive or confidential information, making it inaccessible to unauthorized individuals. Encryption can be used for various purposes, including:

- **Communication Security:** To protect the privacy of online communications, such as during financial transactions or instant messaging, ensuring that only the authorized sender and recipient can read the message.

- **Data Storage Protection:** To secure data stored on devices like hard drives, USB flash drives, or servers, so that if someone physically accesses the data, they cannot read it without the correct access key.
- **Authentication and Digital Signatures:** To verify the authenticity of messages or digital documents and ensure they have not been altered during transmission or storage.
- **Protection of Trade Secrets:** In businesses, encryption is often used to protect trade secrets, customer data, and other sensitive information.

### 3.10.2 Main Approaches in Modern Encryption

There are two main approaches in modern computer encryption: symmetric key encryption and asymmetric key encryption.

**SYMMETRIC KEY ENCRYPTION** Symmetric key encryption is like a lock with a single key. Both the sender and recipient share the same secret key to encrypt and decrypt the data.

- **Encryption Process:** The sender uses the secret key to transform plaintext into an unreadable format.
- **Decryption Process:** The recipient uses the same secret key to decrypt the message and restore it to its original form.
- **Efficiency:** This method is fast and efficient but requires secure key management.

Symmetric Key Encryption example using Advanced Encryption Standard (AES):

#### 1. Preparation:

- **Plaintext Message:** "HELLO WORLD!"
- **Secret Key:** "SECRETKEY123456"

#### 2. Encryption Process:

- **Algorithm:** AES uses multiple rounds of substitution, permutation, and combination.

- **Combination:** Plaintext and secret key are combined through several rounds to generate ciphertext.
- **Result:** Ciphertext is a seemingly random sequence of data.

#### 3. Decryption Process:

- **Secret Key:** The same key used for encryption.
- **Inverse Operations:** AES performs the inverse operations to restore plaintext from ciphertext.
- **Decrypted Message:** The original message "HELLO WORLD!" is obtained.

ASYMMETRIC KEY ENCRYPTION (PUBLIC/PRIVATE KEY ENCRYPTION) Asymmetric key encryption, also known as public/private key encryption, involves two distinct keys that work complementarily to encrypt and decrypt data.

- **Public Key:**
  - Available publicly and can be distributed widely.
  - Each user has their own public key, which can be freely shared with others.
  - Used to encrypt data before sending it to a recipient.
- **Private Key:**
  - Kept secret and known only to the owner.
  - Used to decrypt data encrypted with the corresponding public key.
- **Digital Signatures:**
  - Used to create digital signatures, serving as an "electronic seal" that verifies the sender's authenticity and the integrity of the data.
  - The process includes key pair generation, document hashing, signing, and verification, as shown in Figure 8.

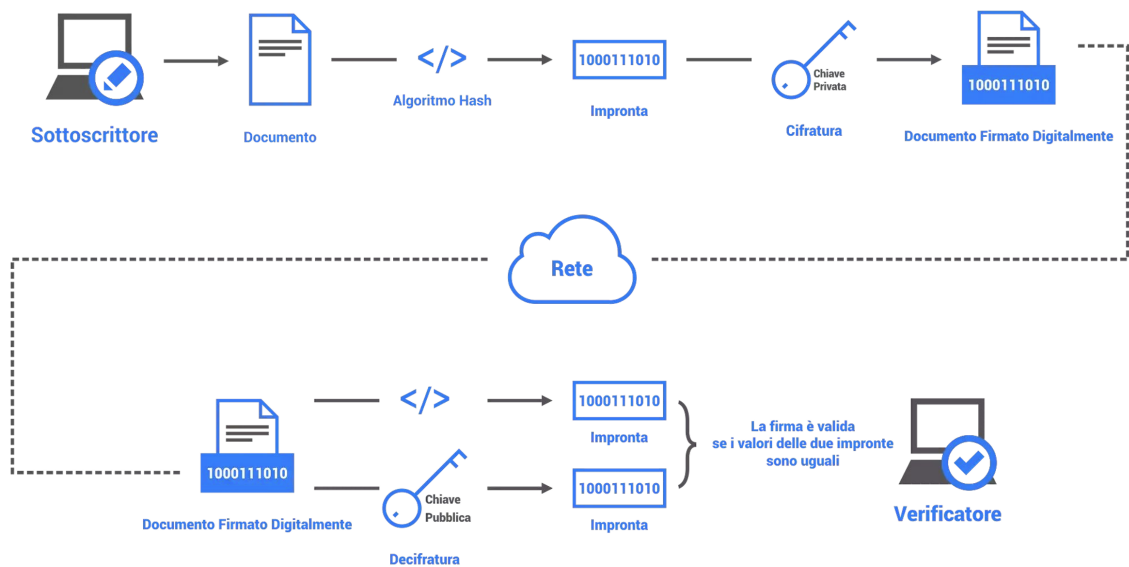


Figure 8: Digital signature schematics.

### 3.11 Virtual Private Network (VPN)

A VPN, or Virtual Private Network, is a technology that creates a secure connection over the internet between your device and a remote server or between two devices, as shown in Figure 9.

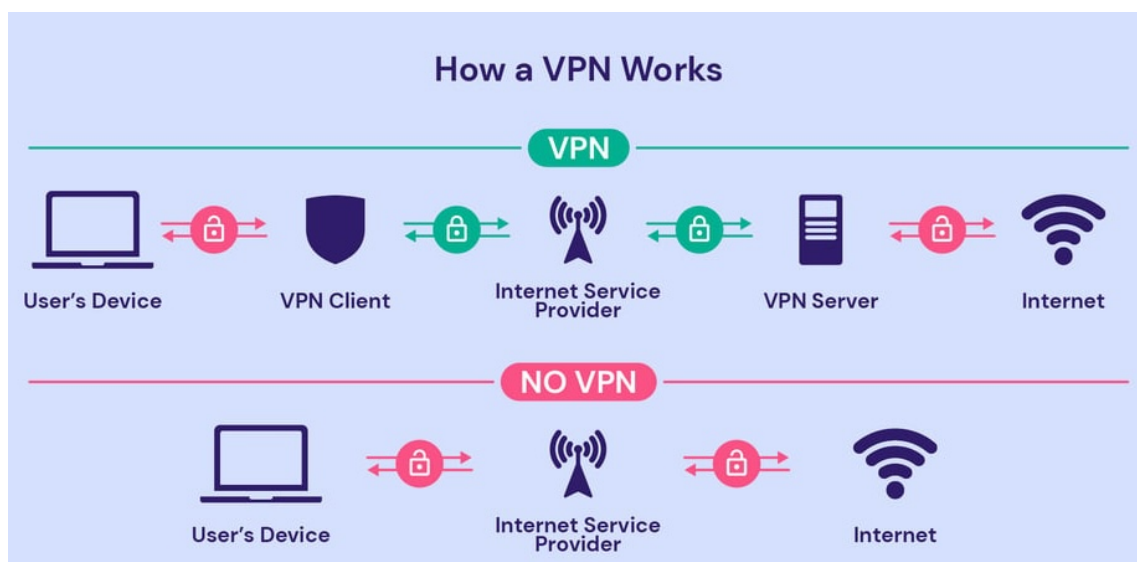


Figure 9: VPN functioning.

- **Security:** Encrypts traffic between your device and the VPN server, ensuring intercepted data cannot be read or interpreted.
- **Privacy:** Hides your real IP address and geographic location, making it difficult for websites and online services to track your activity or location.
- **Access to Geo-Blocked Resources:** Allows you to choose a VPN server, enabling you to bypass geographic restrictions.
- **Public Network Security:** Protects you from potential attacks when connected to public Wi-Fi networks, such as in cafes or airports.



---

## 4 OPERATING SYSTEMS

### 4.1 *Microsoft Windows*

Microsoft Windows, commonly referred to as Windows, is a family of graphical operating systems developed, marketed, and sold by Microsoft Corporation since 1985. It is designed for use on personal computers, workstations, servers, and smartphones. The name "Windows" originates from the window-based graphical user interface (GUI) known as File Explorer. Windows is a multitasking operating system, meaning it can run multiple programs simultaneously. From its inception, Windows has been conceived as a graphical operating system, implementing the "desktop metaphor" with icons, buttons, backgrounds, taskbars, status bars, and many other features.

#### 4.1.1 Windows Shell and PowerShell

The shell, or command interpreter, is a program that allows interaction with the operating system via a terminal and command-line interface. Windows PowerShell is an advanced shell integrated into all Microsoft operating systems starting from Windows 7. It includes an interactive prompt and a scripting environment that can be used separately or in combination. PowerShell is a command-line shell designed specifically for system administrators, offering extensive scripting capabilities to automate routine tasks. Key features and advantages of PowerShell include:

- **Integration with Microsoft .NET:** PowerShell leverages the .NET framework, providing extensive functionality through reusable libraries and functions.
- **Object-Oriented:** Unlike traditional shells that return plain text, PowerShell returns objects, enabling more complex data manipulation.
- **Command-Lets (cmdlets):** PowerShell commands are structured as verb-noun pairs, facilitating intuitive command usage.
- **Admin-Oriented:** PowerShell includes features designed for system administrators, such as remote management and the ability to perform administrative tasks.

### 4.1.2 Common PowerShell Cmdlets

One of PowerShell's strengths is its cmdlets, which are structured as a verb followed by a noun. To display the complete set of available cmdlets, the `Get-Command` cmdlet can be used. PowerShell allows the use of the pipe operator (`—`) to chain multiple cmdlets, using the output of one cmdlet as the input for the next. This technique is particularly useful for text processing tasks.

Cmdlet	Aliases	Description
<code>Set-Location</code>	<code>cd, chdir, sl</code>	Sets the current working location to a specified location.
<code>Get-Content</code>	<code>cat, gc, type</code>	Gets the content of the item at the specified location.
<code>Add-Content</code>	<code>ac</code>	Adds content to the specified items, such as adding words to a file.
<code>Set-Content</code>	<code>sc</code>	Writes or replaces the content in an item with new content.
<code>Copy-Item</code>	<code>copy, cp, cpi</code>	Copies an item from one location to another.
<code>Remove-Item</code>	<code>del, erase, rd, ri, rm, rmdir</code>	Deletes the specified items.
<code>Move-Item</code>	<code>mi, move, mv</code>	Moves an item from one location to another.
<code>Set-Item</code>	<code>si</code>	Changes the value of an item to the value specified in the command.
<code>New-Item</code>	<code>ni</code>	Creates a new item.
<code>Start-Job</code>	<code>sajb</code>	Starts a Windows PowerShell background job.
<code>Compare-Object</code>	<code>compare, dif</code>	Compares two sets of objects.
<code>Group-Object</code>	<code>group</code>	Groups objects that contain the same value for specified properties.

Table 4: Common PowerShell Cmdlets

Cmdlet	Aliases	Description
<code>Invoke-WebRequest</code>	<code>curl, iwr, wget</code>	Gets content from a web page on the Internet.
<code>Measure-Object</code>	<code>measure</code>	Calculates numeric properties of objects, and the characters, words, and lines in string objects.
<code>Resolve-Path</code>	<code>rvpa</code>	Resolves the wildcard characters in a path and displays the path contents.
<code>Resume-Job</code>	<code>rujb</code>	Restarts a suspended job.
<code>Set-Variable</code>	<code>set, sv</code>	Sets the value of a variable. Creates the variable if one with the requested name does not exist.
<code>Show-Command</code>	<code>shcm</code>	Creates Windows PowerShell commands in a graphical command window.
<code>Sort-Object</code>	<code>sort</code>	Sorts objects by property values.
<code>Start-Service</code>	<code>sasv</code>	Starts one or more stopped services.
<code>Start-Process</code>	<code>saps, start</code>	Starts one or more processes on the local computer.
<code>Suspend-Job</code>	<code>sujb</code>	Temporarily stops workflow jobs.
<code>Wait-Job</code>	<code>wjb</code>	Suppresses the command prompt until one or all of the Windows PowerShell background jobs running in the session are complete.
<code>Where-Object</code>	<code>?, where</code>	Selects objects from a collection based on their property values.

Table 5: Additional PowerShell Cmdlets

### 4.1.3 Windows File System

The Windows file system is an organizational structure within the operating system that regulates the naming, storage, and retrieval of files. PowerShell provides cmdlets to interact with the file system, files, and directories.

- `Get-ChildItem`: Displays the contents of a directory.
- `Copy-Item`: Copies a file from one location to another. This cmdlet accepts two parameters: the source file and the destination.

#### 4.1.4 Components of Windows Operating System

The fundamental components of the Windows operating system are depicted in the diagram (Figure 10). Windows OS distinguishes between user space and kernel space as a security measure, separating tasks that require special privileges from those that do not. The DLL (Dynamic Link Library) subsystems interface between kernel mode and user mode, providing function libraries that users and applications can utilize to request the kernel to perform specific tasks.

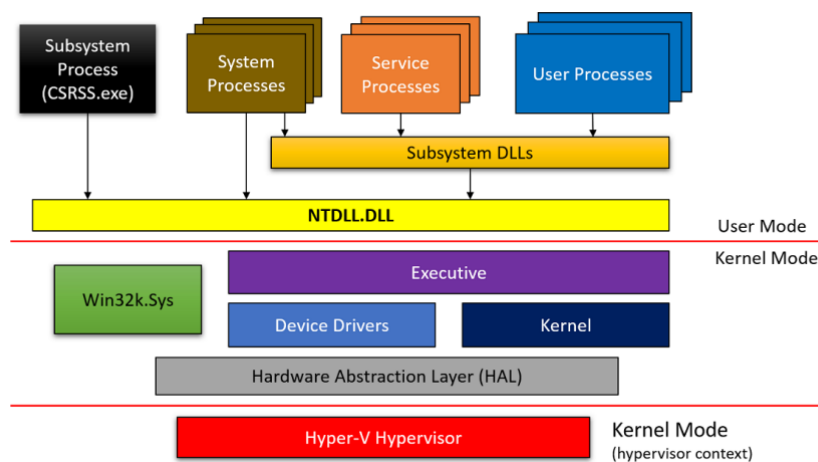


Figure 10: Components of Microsoft Windows Operating System

- **User Applications:** User applications run in user mode and can be either 32-bit or 64-bit, depending on the operating system.
- **Services:** This includes active processes and services in Windows, such as the task manager or network services.
- **Executive:** The executive contains the core functionalities of the operating system, including memory management, security features, and process management.
- **Driver Management:** Manages calls to input/output devices.
- **Kernel:** The kernel encompasses low-level OS functions, such as thread scheduling, interrupts, exception handling, and multiprocessor synchronization.

### 4.1.5 Windows Hybrid Kernel

The Windows kernel is a hybrid kernel, meaning it incorporates elements of both monolithic and microkernel architectures.

- **Monolithic Kernel:** A monolithic kernel runs most operations directly in kernel space, including memory management, process management, and system calls, providing high performance but potentially lower stability due to the lack of isolation between components.
- **Microkernel:** A microkernel, conversely, delegates many functionalities to user space processes, keeping only essential functions in the kernel, which enhances modularity and stability at the cost of potential performance overhead due to increased inter-process communication.
- **Hybrid Kernel:** Windows hybrid kernel aims to balance the speed of a monolithic kernel with the modularity and stability of a microkernel by implementing critical functions within the kernel and allowing certain components, such as device drivers, to run in user space.

## 4.2 *Linux Operating Systems*

The origins of the Unix Operating System date back to the late 1960s with the development of the MULTICS (Multiplexed Information and Computing Service) project. This project was a collaboration between AT&T, Honeywell, General Electric, and the Massachusetts Institute of Technology (MIT), sponsored by ARPA, the research arm of the U.S. Department of Defense. The goal was to create an operating system capable of continuing to function even if some parts of the computer were shut down or deactivated, thereby not affecting the work of users utilizing the still-active components. The modularity of this system allowed for improvements or expansions by simply adding new modules, without the need to rebuild the entire system from scratch.

In 1991, Linus Torvalds, a computer science student from Helsinki, released a version of the Minix monolithic kernel that could run on a standard Intel 386 processor, and he distributed it freely along with its source code. This project evolved into what we now know as the Linux kernel, which has been periodically updated and revised over the years.

Today, Linux is one of the most widely used operating systems for:

- Web Servers
- High-performance applications
- Embedded systems
- Network systems

Linux is freely distributed, meaning that all users can download it and contribute to the project. The term "free software" was initially popularized by Richard Stallman, emphasizing that users are granted the following rights:

- The freedom to run the program
- The freedom to study how the program works
- The freedom to redistribute copies of the program
- The freedom to modify the program

#### 4.2.1 Structure of the Linux Operating System

As an operating system, Linux serves as an interface between the user and the machine, providing a set of tools and functionalities. The primary components are:

- Hardware
- Operating System
- Shell
- GUI (Graphical User Interface)
- Applications

The operating system sits between the hardware and user applications, providing routines for managing processes, memory, and other resources. This type of kernel is known as a "monolithic" kernel.

### 4.2.2 Types of Kernels

There are several types of kernels used in operating systems:

- **Microkernel:** A minimalistic kernel that includes only the basic functionalities required for the operating system to function. An example is MINIX.
- **Monolithic Kernel:** A single binary file that includes most of the operating system's functionalities. An example is the Unix kernel.
- **Modular Kernel:** An extension of the monolithic kernel, capable of adding or removing modules as needed.

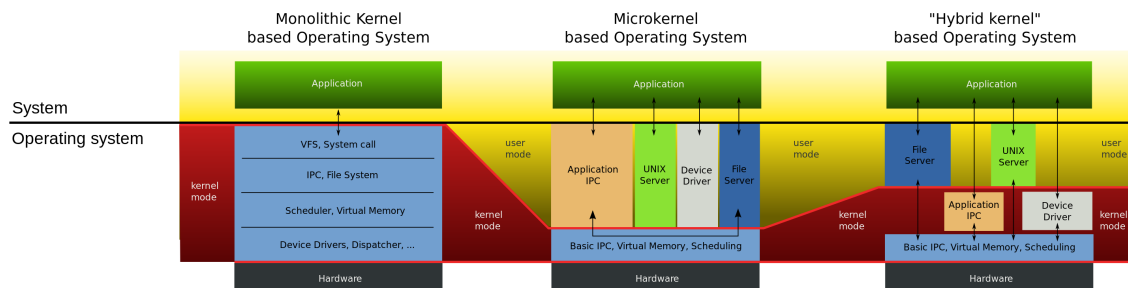


Figure 11: Types of Kernels: Microkernel, Modular, Monolithic

### 4.2.3 Processes in Linux

In Linux, a process is an instance of a program in execution, assigned to a processor for instruction execution. A process has a state, which is read by the CPU for scheduling and process switching. Linux processes are complex structures characterized by:

- Memory addresses used
- Content of the addressed memory
- Pointers to input/output devices used
- Data, i.e., variables used by the process
- Threads

**THREADS** Threads are tasks within a process. A process can perform various functions using multiple threads, each handling separate tasks. Threads within a process share the same state and memory information, allowing independent action on the used resources. Proper prioritization by the operating system is crucial to avoid simultaneous modification of the same resource by different threads (synchronization issue).

In Linux, each process is identified by a Process Identifier (PID), while each thread within a process is identified by a Thread Identifier (TID). Processes are created by invoking the system call `Fork()`. The child process created with `Fork()` inherits attributes from the parent process.

Command	Description
<code>mkdir</code>	Creates a directory
<code>rmdir</code>	Removes a directory
<code>cp</code>	Copies files and directories
<code>rm</code>	Removes files
<code>mv</code>	Moves or renames files
<code>ls</code>	Lists directory contents
<code>cd</code>	Changes the current directory

*Table 6: Common Linux File System Commands*

#### 4.2.4 Linux File System

The Linux file system originates in the root directory, denoted as `/`. The root user, or system administrator, has extensive control over the system. Key directories in the Linux file system include:

- `/bin`: Contains essential system programs available for booting the system.
- `/home`: Contains user-specific directories and files.
- `/usr`: Contains installed programs, manual files, and documentation.
- `/sbin`: Similar to `/bin` but for system administration.
- `/etc`: Contains system and application configuration files.

### 4.2.5 User Management in Linux

Linux systems often support multiple users, necessitating mechanisms to separate user data and protect privacy. Each user is assigned a unique user identifier (UID) and a home directory. File permissions ensure that private files are not accessible to other users. User information is stored in `/etc/passwd`, which includes:

- Group IDs
- User IDs
- Login shell
- Encrypted password (hashed)

Linux, with its rich history and robust architecture, continues to be a cornerstone in modern computing. Its versatility, security, and open-source nature make it a preferred choice for servers, high-performance applications, embedded systems, and network systems.



---

## 5 PROGRAMMING LANGUAGES

### 5.1 *Introduction*

Becoming an Ethical Hacker necessitates a profound understanding of various programming languages. Many hacking tools are custom-developed by hackers themselves, thus, a solid foundation in key programming languages is essential. This document explores some of the fundamental programming languages used in cybersecurity, particularly focusing on:

- **C Language** is considered the foundation of many programming languages, with most libraries and Linux code written in C.
- **Python Language**, an interpreted, object-oriented language, is widely used by hackers due to its simplicity and effectiveness.

#### 5.1.1 Classification of Programming Languages

Today, there are innumerable programming languages, which can be categorized into the following traditional classifications:

- **Machine Languages:** in machine language, algorithms are encoded using binary code, often referred to as "words" of 0s and 1s (bits). The computer directly executes instructions written in machine language.
- **Assembly Languages:** assembly language defines a set of elementary operations using a more human-readable syntax. This language is closer to human language but still requires a translator (assembler) for the computer to execute the code.
- **High-Level Languages:** high-level programming languages are much closer to human languages. Their instructions are comprehensible and easy to use. However, they also require a translator (compiler or interpreter) to convert high-level language into machine language (sequences of 0s and 1s).

#### 5.1.2 Types of Translators

There are two primary types of translators:

- **Compiler:** Translates a program written in a high-level language into machine language.
- **Interpreter:** Analyzes and translates a program written in a high-level language and executes the instructions simultaneously.

### 5.1.3 Error Types in Programming

When programming, it is beneficial to understand the types of errors that might be encountered. Programming errors can be divided into three categories:

- **Syntax Errors:** Syntax errors are typographical mistakes made during the coding phase. These are the easiest to resolve as compilers often indicate the location of the error by providing the line number where the error occurred.
- **Logical Errors:** Logical errors occur during the design phase of the algorithm. These are more challenging to fix because the compiler does not identify them; the program will still compile successfully.
- **Runtime Errors:** Runtime errors occur post-compilation when the program is running. These are difficult to identify as they are not detected by the compiler.

## 5.2 *C Programming Language*

The C programming language is a general-purpose, procedural computer programming language that was developed in the early 1970s by Dennis Ritchie at Bell Labs. It has since become one of the most widely used programming languages of all time due to its efficiency, portability, and flexibility.

### 5.2.1 Main Properties of C Language

- **Low-Level Access:** C provides low-level access to memory through pointers, making it ideal for system programming.
- **Portability:** C programs can be compiled and run on various computer systems with minimal or no modification.

- 
- **Rich Library:** The Standard C Library provides numerous built-in functions for performing input/output, string manipulation, memory allocation, and more.
  - **Modularity:** C supports modular programming through the use of functions, allowing code to be organized and reused.
  - **Structured Language:** C allows complex programs to be broken into simpler sub-programs or functions, facilitating easier management and debugging.

## 6 BASIC SYNTAX AND COMMANDS IN C

### 6.1 Data Types

C supports several built-in data types, which can be categorized as follows:

Data Type	Description	Size (bytes)
int	Integer	2 or 4
float	Floating point	4
double	Double precision floating point	8
char	Character	1
void	Empty type	0

*Table 7: Basic Data Types in C*

#### 6.1.1 Control Structures

C provides several control structures for directing the flow of a program:

- **Conditional Statements**
  - if, else if, else
  - switch, case, default
- **Loops**
  - for loop
  - while loop
  - do-while loop
- **Jump Statements**

- break
- continue
- return
- goto

### 6.1.2 Functions

Functions in C are used to modularize code. The general form of a function definition is:

```
return_type function_name(parameters) {  
    // body of the function  
}
```

Keyword	Description
void	Specifies that the function does not return a value
int	Specifies that the function returns an integer value
float	Specifies that the function returns a floating-point value
char	Specifies that the function returns a character value

*Table 8: Common Function Return Types in C*

### 6.1.3 Input/Output Operations

C uses the Standard Input and Output library (`stdio.h`) for input and output operations.

The most commonly used functions are:

Function	Description
<code>printf()</code>	Prints formatted output to the console
<code>scanf()</code>	Reads formatted input from the console
<code>getchar()</code>	Reads a single character from the console
<code>putchar()</code>	Writes a single character to the console
<code>fgets()</code>	Reads a string from a file or the console
<code>fputs()</code>	Writes a string to a file or the console

*Table 9: Common Input/Output Functions in C*

The C programming language is foundational to many other programming languages and is extensively used in system and application software, embedded systems, and high-performance server and client applications. Its efficiency, portability, and modularity make it a valuable language to learn for aspiring ethical hackers and cybersecurity professionals.

---

## 7 UNDERSTANDING POINTERS, MEMORY MANAGEMENT, STACKS, AND ARRAYS IN C

### 7.1 Pointers

Pointers are a type of variable that contain memory addresses. A pointer in C is declared by prefixing the variable name with an asterisk (\*). For example:

```
int *i; // pointer to an integer
char *c; // pointer to a character
```

There are two main operations that can be performed with a pointer:

- Change the content at the memory address the pointer points to.
- Change the address that the pointer points to.

Consider the following example:

```
int var = 123;
int *i;
i = &var; // pointer i points to the address of var
*i = 5; // the content at the address pointed by i is changed to 5
```

After execution, var will be 5.

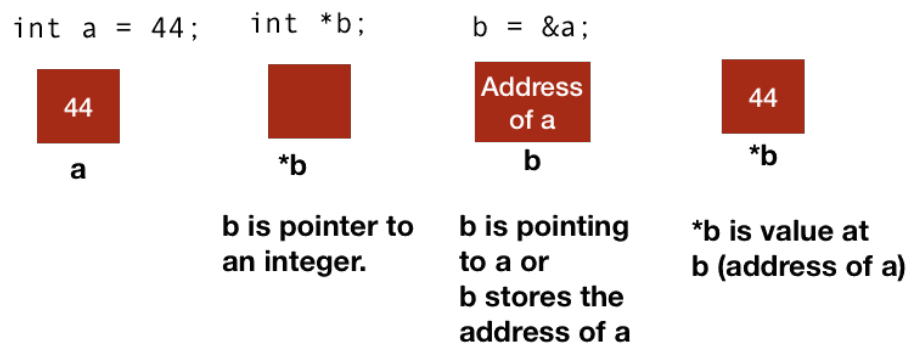


Figure 12: Pointers in C.

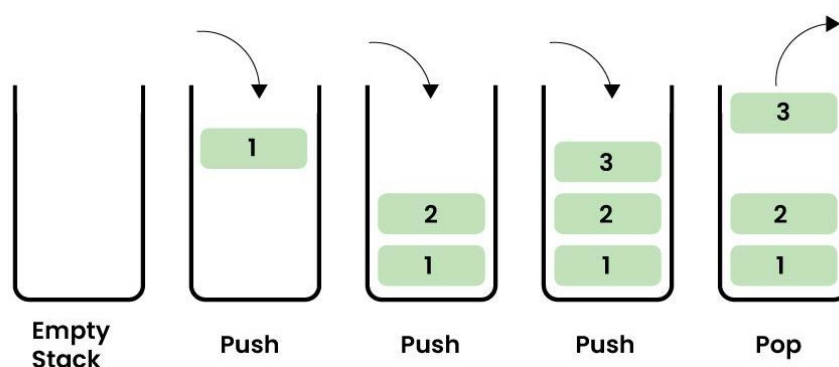
## 7.2 Memory Management in C

During the execution of a program, memory in C is managed in two ways:

- **Static Memory Management:** A fixed memory area is allocated by the operating system for the entire duration of the program.
- **Dynamic Memory Management:** Two distinct memory areas are allocated and used as needed: the stack and the heap.

### 7.2.1 Stack

The stack in C is a special region of a program's memory that stores temporary variables created by each function (including the main function). It plays a crucial role in managing function calls, local variables, and control flow. The stack is used for variables and function parameters. For each function call, a new stack frame is created with its parameters. When the function completes, its stack frame is removed, and the stack reverts to its previous state. The stack operates on the Last In, First Out (LIFO) principle. This means that the last item pushed onto the stack is the first one to be popped off. Think of it like a stack of plates: you can only add or remove the top plate.



**HEAP** The heap is a memory area managed by the programmer and can be dynamically allocated as needed using C's memory management libraries.

- The **ESP** (Extended Stack Pointer) register points to the top of the current stack frame. It keeps track of the last used location in the stack. When a function is called, the

ESP is adjusted to allocate space for the function's local variables. When a function returns, the ESP is adjusted back to free the space.

- The **EBP** (Extended Base Pointer) register points to the base of the current stack frame. EBP is used to reference function parameters and local variables in a consistent way, even if the ESP changes during the function execution (e.g., due to push and pop operations). When a function is called, the current EBP value is pushed onto the stack, and EBP is then set to the current value of ESP.
- The **EIP** (Extended Instruction Pointer) register holds the address of the next instruction to be executed. When a function call occurs, the current EIP (pointing to the next instruction in the calling function) is pushed onto the stack. This allows the program to return to the correct place after the function call completes. This push operation saves the address of the next instruction to be executed after the function returns, facilitating the call-return mechanism.

#### STACK OPERATION DURING A FUNCTION CALL

1. **Function Call (CALL Instruction):** When a function is called, the current EIP (instruction pointer) is pushed onto the stack. This saves the address of the instruction to return to after the function completes. The current EBP is also pushed onto the stack to preserve its value. ESP is decremented to make space for the local variables and the function's stack frame.
2. **Function Prologue:** The new function sets its EBP to the current ESP value to establish a new base for its stack frame. Local variables and saved registers are pushed onto the stack as needed.
3. **Accessing Parameters and Local Variables:** Function parameters are accessed at positive offsets from EBP. Local variables are accessed at negative offsets from EBP.
4. **Function Return (RET Instruction):** The ESP is restored to the value of EBP to free the stack frame. The previous EBP is popped off the stack to restore the caller's base pointer. The saved EIP (return address) is popped from the stack into the EIP register to resume execution at the point after the function call.

EXAMPLE OF STACK USAGE Consider the following C code:

```
void func(int a, int b) {  
    int x, y;  
    x = a + b;  
    y = x * 2;  
}  
  
int main() {  
    int p = 5, q = 10;  
    func(p, q);  
    return 0;  
}
```

Here's how the stack operations would work:

- **Calling func(p, q):**

- The current EIP (address of the next instruction after the call) is pushed onto the stack.
- The current EBP is pushed onto the stack.
- ESP is adjusted to allocate space for func's stack frame.

- **Inside func:**

- EBP is set to the current value of ESP.
- Space is allocated for local variables x and y on the stack by adjusting ESP.
- Parameters a and b are accessed relative to the new EBP.

- **Returning from func:**

- ESP is restored to the value of EBP.
- The previous EBP is popped from the stack.
- The saved return address (EIP) is popped from the stack, and execution resumes in main.



**VISUALIZATION** Here's a simplified visualization of the stack state at different stages:

1. Initial State (Before Function Call)

High Address	...	
Low Address	...	

2. After Pushing Return Address and EBP

High Address	...	
	Return EIP	
	Old EBP	
Low Address	...	

3. Inside Function (After Allocating Local Variables)

High Address	...	
	Return EIP	
	Old EBP	
	Local var	
	Local var	
Low Address	...	

The stack in C is a critical component of memory management, enabling function calls, local variable storage, and control flow management. Understanding the stack's LIFO nature, the role of registers like ESP, EBP, and EIP, and the operations performed during function calls and returns is essential for effective programming and debugging in C.

### 7.3 Arrays in C

C provides arrays, which are sequences of homogeneous elements. An array in C is declared as follows:

```
int array[10]; // array of 10 integers
```

Each element in the array is accessible via its index, starting from 0. For example:

```
array[0] = 1;  
array[9] = 10;
```

### 7.4 *Multidimensional Arrays*

C also supports multidimensional arrays, such as matrices. A multidimensional array is defined as:

```
int matrix[3][3]; // 3x3 matrix
```

Each element is accessed using two indices (row and column). For example:

```
matrix[0][0] = 1;  
matrix[2][2] = 9;
```

The concepts for manipulating multidimensional arrays are similar to those for one-dimensional arrays, with the addition of multiple indices.

---

## 8 COMPUTER HARDWARE AND OPERATING SYSTEMS

In the study of techniques used by Ethical Hackers, it is essential to have a thorough understanding of both the components of a computing device and the operating systems. This lesson will explore how a computing device and an operating system function, including the concepts of processes, memory, and the file system.

### 8.1 *The von Neumann Architecture*

The von Neumann model is a computer architecture proposed by mathematician and logician John von Neumann in the 1940s. This model has been fundamental to the development of modern computers and remains widely used in the design of computing systems.

### 8.2 *Main Components of a PC/Computing Device*

#### 8.2.1 Processor (CPU)

The Central Processing Unit (CPU) is the brain of the computer, responsible for executing software instructions. The primary manufacturers are Intel and AMD.

#### 8.2.2 Motherboard

The motherboard connects all the other components of the computer. It contains the main chipset, expansion ports, and connectors for RAM, CPU, and peripherals.

#### 8.2.3 Memory (RAM)

Random Access Memory (RAM) is the temporary memory used by the operating system and running programs to store temporary data. RAM is fast but volatile, meaning it loses data when the computer is turned off.

#### 8.2.4 Storage

Includes Hard Disk Drives (HDD) or Solid State Drives (SSD). HDDs are cheaper but slower, while SSDs offer higher speeds and no moving parts.

### 8.2.5 Graphics Card (GPU)

Manages graphics and accelerates image rendering. It can be integrated into the motherboard or separate, known as a dedicated graphics card.

### 8.2.6 Power Supply Unit (PSU)

Provides electrical power to the computer. Its power rating must be sufficient to support all components.

### 8.2.7 Optical Drive

Typically a CD/DVD reader or, less commonly, a Blu-ray reader.

### 8.2.8 Case

The enclosure that houses all components. It can come in various sizes and shapes.

### 8.2.9 Cooling (Fans and Liquid Cooling)

Maintains safe temperatures for computer components. Includes CPU and GPU fans, as well as optional liquid cooling systems.

### 8.2.10 Network Interface

Usually integrated into the motherboard, it provides network connectivity such as Ethernet or Wi-Fi.

### 8.2.11 Expansion Ports

USB, HDMI, DisplayPort, audio, and other ports allow the connection of external peripherals such as mice, keyboards, printers, monitors, etc.

### 8.2.12 Operating System (OS)

Software that manages the hardware and provides a user interface for interacting with the computer. Examples include Windows, macOS, and Linux.

### 8.3 Binary System and Data Types

Operating systems process information in binary format, which is a sequence of 0s and 1s. The binary system is a numerical system where all numbers are represented using 0 and 1. Everything handled by a personal computer, such as numbers, characters, and programs, is managed as a sequence of 0s and 1s. The fundamental unit is called a bit (a bit can only be 0 or 1).

Since everything is expressed as a sequence of bits, to describe the nature of the data being processed, an operating system relies on the concept of "data type" which will vary depending on whether the operating system is dealing with a number or a series of characters. The most common data types are shown in the table below:

Data Type	Represents
int	Used to represent integers
float	Used to represent real numbers
char	Used to represent characters
bool	Used to represent true or false states

Table 10: Common Data Types

### 8.4 Logic Gates

Logic gates are fundamental elements in digital circuit theory and electronic circuit design. They represent fundamental logical operations between boolean inputs (true or false values). Common logic gates include AND, OR, NOT, NAND, NOR, XOR, and XNOR.

## Logic Gates








Name	NOT	AND	NAND	OR	NOR	XOR	XNOR																																																																																																
Alg. Expr.	$\overline{A}$	$AB$	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A\oplus B$	$\overline{A\oplus B}$																																																																																																
Symbol																																																																																																							
Truth Table	<table><tr><th>A</th><th>X</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	X	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	B	A	X	0	0	0	0	1	1	1	0	1	1	1	0	<table><tr><th>B</th><th>A</th><th>X</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	B	A	X	0	0	1	0	1	0	1	0	0	1	1	1
A	X																																																																																																						
0	1																																																																																																						
1	0																																																																																																						
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	1																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	0																																																																																																					
0	1	1																																																																																																					
1	0	1																																																																																																					
1	1	0																																																																																																					
B	A	X																																																																																																					
0	0	1																																																																																																					
0	1	0																																																																																																					
1	0	0																																																																																																					
1	1	1																																																																																																					

Figure 13: Logic Gates and Their Truth Tables

### 8.5 *Operating System Fundamentals*

An operating system (OS) is a crucial software that manages a computer's resources and provides an interface between the hardware and user applications. At its core is the kernel, a critical component that handles low-level operations and directly interacts with the hardware.

#### 8.5.1 The Kernel

- **Definition:**

- The kernel is the central and most fundamental part of an operating system.
- It is responsible for managing hardware resources, including processors, memory, storage devices, and I/O.

- **Resource Management:**

- Devices: Controls and communicates with hardware devices such as keyboards, mice, and printers.

- **Interaction with Hardware:**

- The kernel provides an abstraction of the hardware, allowing applications to interact with hardware without needing to know the specific details of each device.

### 8.6 *Operating System Functions*

- **System Boot:**

- The boot process begins with the BIOS/UEFI, which loads the kernel into memory.
- The kernel starts execution and initiates the operating system's initialization process.

- **Kernel Initialization:**

- During initialization, the kernel configures device drivers.

- **Process Management:**

- When an application is started, the kernel creates a process for it.
- The kernel allocates resources and schedules the process to execute instructions on the CPU.

- **Memory Management:**

- The kernel controls memory management.

- **Device Communication:**

- The kernel handles I/O requests, interacting with device drivers to send and receive data to and from devices.

#### 8.6.1 User Interface Interaction

- **User Interface:**

- The user interface, which can be a Command-Line Interface (CLI) or a Graphical User Interface (GUI), relies on the kernel to perform operations requested by users.

- **Kernel Communication:**

- User applications communicate with the kernel through system calls, requesting specific services such as file reading/writing, memory allocation, etc.

- **Application Management:**

- The kernel manages the execution of applications, ensuring that requested resources are accessed securely and controlled.

### 8.7 Scenario: System Boot and Application Launch on Windows

#### 8.7.1 Phase 1: System Boot

- **Powering On the Computer:**

- The user powers on the computer with Windows OS installed.

- The boot process begins, managed by the bootloader and Windows kernel.

- **Kernel Initialization:**

- The Windows kernel is loaded into memory and starts the initialization process.
- Configures device drivers, establishes system settings, and prepares the system for use.

- **Loading the User Interface:**

- The kernel loads the Windows user interface, which may include the desktop and taskbar.

### 8.7.2 Phase 2: User Interface Interaction

- **Interacting with the User Interface:**

- The user interacts with the Windows UI, clicking on an application icon in the Start menu or on the desktop.

- **Kernel Request:**

- The user interface sends a request to the Windows kernel to launch the selected application.

- **Process Creation:**

- The Windows kernel creates a new process for the application, allocating resources such as memory and CPU time.

### 8.7.3 Phase 3: Application Management

- **Application Execution:**

- The Windows kernel schedules the application's process to execute on the CPU.
- The application begins executing its instructions.

- **Resource Management:**



- While the application is running, the kernel manages system resources, ensuring the application does not interfere with other processes or misuse resources.
- **User Interface Communication:**
  - The application communicates with the user interface through the Windows kernel to display output to the user and respond to interactions.

## 8.8 Overview of Kernel Types in Operating Systems

To reinforce our understanding of key concepts that will be covered in the upcoming slides, we will perform a quick review related to the kernel in general. These topics build a solid foundation for understanding advanced concepts and practical applications of the kernel within our training program.

### 8.8.1 Types of Kernels

There are several types of kernels, each with its unique architecture and functionality. Below are the main types of kernels:

- **Microkernel:** A microkernel is a minimalistic kernel that includes only the most essential functionalities of an operating system. An example of a microkernel is *MINIX*. In a microkernel architecture, many services such as device drivers, file systems, and network protocols are implemented in user space, which can lead to better modularity and reliability but may introduce performance overhead due to context switching between user space and kernel space. A microkernel, on the other hand, delegates many functionalities to processes in user space, keeping only the essential functions in the kernel itself. This design makes the system more modular and resilient to errors, as components can run in isolated user spaces. However, the communication between components across user-kernel boundaries can slow down performance.
- **Monolithic Kernel:** A monolithic kernel is a single binary file that includes most of the functionalities of an operating system. An example of a monolithic kernel is *Unix*. Monolithic kernels run all operating system services in the kernel space, which allows for efficient system calls and better performance. However, this architecture

can lead to stability issues, as a failure in one part of the kernel can crash the entire system. A monolithic kernel performs most operations directly in the kernel space, including memory management, process management, and communication between system components. This architecture is generally faster because system calls do not need to cross the boundary between user space and kernel space. However, it can lead to stability problems, as a bug in one part of the kernel can affect the entire system.

- **Modular Kernel:** A modular kernel can be seen as an extension of the monolithic kernel, with the capability to dynamically add or remove modules as needed. This provides a balance between performance and flexibility, allowing the kernel to be customized for different use cases without requiring a complete recompilation.
- **Hybrid Kernel: The Case of Windows:** The Windows kernel is a hybrid kernel, known as the *Hybrid Kernel*. This means it incorporates elements of both monolithic and microkernel architectures. The hybrid approach in Windows combines elements of both architectures. For example, certain parts of the kernel, such as memory and process management, are implemented within the kernel itself (monolithic), while other functionalities, such as device drivers, can operate in user space (microkernel-like approach). This hybrid architecture aims to balance the speed of a monolithic kernel with the modularity and stability of a microkernel.

The kernel is the core of the operating system, efficiently managing hardware resources and providing a unified interface for user applications. Through precise orchestration of low-level operations, the kernel contributes to providing a stable and secure environment for running programs and interacting with hardware.

### 8.9 Overview of Operating Systems

An operating system (OS) is a software that provides services to users of a computer system and acts as an interface between the user and the physical machine. The primary tasks performed by an operating system include:

- Memory management
- Process management, where a process is defined as a running program

- 
- Process synchronization
  - User interface management, such as handling input/output (I/O) operations for hardware peripherals like mice, keyboards, and screens

### 8.9.1 Hardware and Operating System

Hardware consists of the physical components that make up a personal computer, such as memory (RAM, hard drives, SSDs) and processors (CPUs) that are dedicated to executing instructions. The operating system is responsible for managing these components and providing a stable environment for applications to run.

To handle all these tasks, operating systems are generally composed of multiple modules, each dedicated to a specific function. The operating system organizes the interaction between these different modules, such as:

- Process management
- I/O management
- File system management
- Central memory management
- User interface

**KERNEL** The kernel is the core of the operating system, managing interactions between hardware and running processes. The file system is a mechanism that regulates the storage of data on a memory medium in an organized manner.

## 9 PROCESS MANAGEMENT

A process is a running program, and each process is associated with a state, which can be:

- **Running:** The process instructions are being executed by the processor (CPU).
- **Waiting:** The process is waiting for an event to occur (e.g., user input via keyboard).
- **Ready:** The process is waiting to be assigned to a processor to transition to the "Running" state.

- **Terminated:** The process has finished its execution.

The process manager module controls the synchronization and state of running programs. In a computer with multiple processes running simultaneously, each CPU executes the instructions of a process for a limited time before switching to another process. This switching of processes in the "Running" state is called scheduling, and the program that determines the time, manner, and sequence of switching is called the scheduler.

In modern multi-CPU systems, the process manager also handles the cooperation between different CPUs. Scheduling can be categorized into two types:

- **Preemptive Scheduling:** The CPU currently in use by a process can be reassigned to execute another process at any time.
- **Cooperative Scheduling:** Once a process is in the "Running" state, it cannot be interrupted until it completes its work and voluntarily releases the CPU.

### 9.1 Monotasking vs. Multitasking

Monotasking operating systems can run only one program at a time. In such systems, the execution of a program cannot be suspended to assign the CPU to another program. These systems are generally outdated (e.g., MS-DOS). Monotasking systems are inefficient due to frequent periods of CPU idleness, as shown in the following graph:

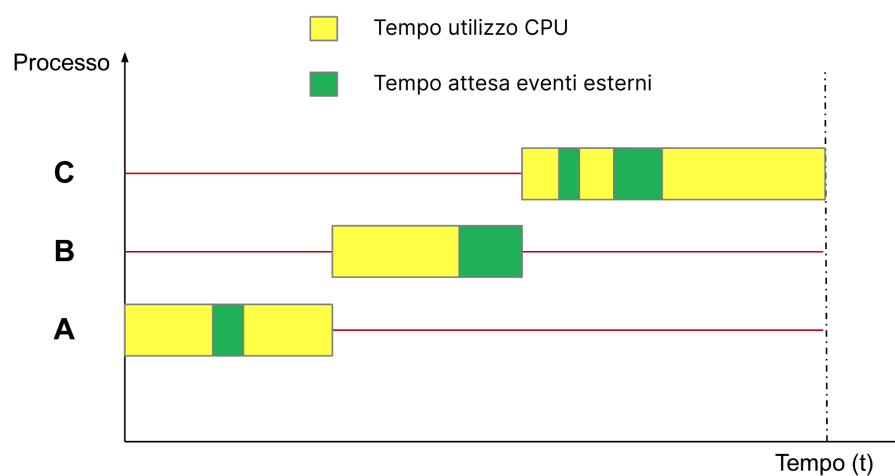


Figure 14: Inefficiency of Monotasking Systems

Multitasking operating systems allow the concurrent execution of multiple programs. Examples of multitasking operating systems include Windows-NT and Linux-based systems. In multitasking systems, processes can be interrupted to shift the CPU's attention to another process.

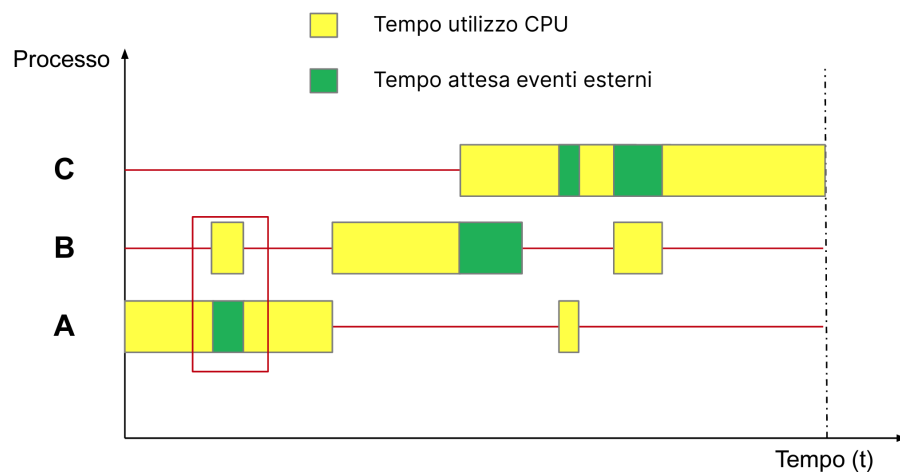


Figure 15: Efficiency of Multitasking Systems

## 9.2 Time-Sharing Systems

An evolution of multitasking systems is time-sharing systems. In a time-sharing system, each process is executed cyclically for small time intervals called "time slices" or "quanta". With a sufficiently fast CPU, a time-sharing system gives the impression of parallel process evolution.

In time-sharing systems, processes are executed for a standard period called a "quantum". The process is interrupted to execute another process for a "quantum", and so on.

### 9.3 Preemptive and Cooperative Multitasking

Common operating systems, such as Windows, macOS, and many Linux distributions, adopt preemptive multitasking to manage the simultaneous execution of multiple processes or applications. There are two main types of multitasking:

- **Preemptive Multitasking:** In a preemptive multitasking system, the operating system allocates a certain period (time slice or quantum) to each process and then

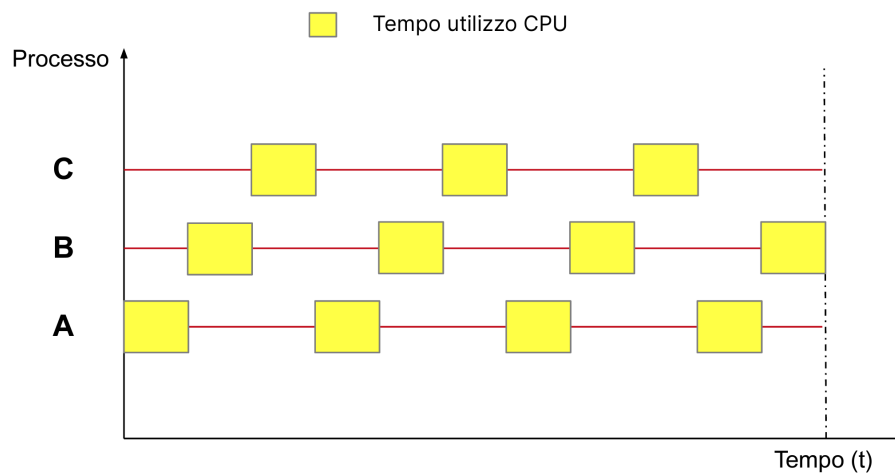


Figure 16: Time-Sharing System

automatically switches to the next process. This approach is called "preemptive" because the operating system can interrupt a running process before it completes its execution period. This ensures fair distribution of system resources among processes.

- **Cooperative Multitasking:** In a cooperative multitasking system, it is the responsibility of individual processes to release control of the CPU when they no longer need it. This requires greater cooperation among processes, as they must be designed to voluntarily yield control. However, if a process does not release control, it can negatively impact system performance.

The operating system is a crucial software component that provides services to users and acts as an interface between the user and the physical machine. By managing hardware resources efficiently and providing a stable environment for applications, the operating system plays a vital role in the overall functionality and performance of a computer system.

#### 9.4 Memory Management in Operating Systems

One of the central and most important tasks of an operating system is the management of central memory. But what is central memory? We can distinguish four types of central memory:

- **ROM (Read Only Memory):** ROM is used by operating systems for computer ini-

tialization programs. Basic routines of computer systems, such as the BIOS (Basic Input/Output System), were traditionally written in ROM. Today, it is more accurate to refer to this type of memory as EEPROM.

- CPU Registers: CPU registers are very fast access memory locations used by the CPU to store instructions to be followed for short periods of time.
- Cache Memory: Cache memory represents areas where previously used data is temporarily stored for faster future access. Cache memories are quite limited and can only contain a small amount of data.
- RAM (Random Access Memory): RAM is the main memory of an operating system, and the greater the available amount, the better the PC's performance. RAM consists of a matrix of cells, each identified by an address in binary notation. When a process is executed by the CPU, its instructions, saved on secondary memory, must be loaded into RAM cells to be read.

Secondary memory, or mass memory, refers to non-volatile memory types (i.e., where information remains saved even when the machine is turned off). Examples include hard disks, USB flash drives, and SSDs.

### 9.5 *Memory Management Module*

The memory manager is the module responsible for allocating memory to various tasks when needed and removing tasks from memory when completed. The complexity of the memory manager is directly related to the type of operating system. In multi-tasking systems, managing multiple programs loaded in memory is more complex, particularly the optimal allocation of memory spaces.

#### 9.5.1 Linear Allocation Example

Consider a multi-tasking system with three processes, A, B, and C.

1. Process A transitions to the running state, and the CPU loads A's instructions into RAM.
2. Next, it is B's turn, and the CPU loads B's instructions.

3. The instructions of C are then loaded into memory.

This model is known as linear allocation. The memory blocks occupied by each process depend on the number of instructions each process needs to execute.

**FRAGMENTATION ISSUE** Consider the following scenario:

1. Process C completes its work, and the CPU removes it from memory.
2. Process D needs to be executed, but the memory space it requires is larger than what was freed by C. As a result, there will be unused free memory until a process that fits into the space freed by C can be found.

**PAGING SOLUTION** The solution to the fragmentation problem is "paging." With this technique, memory is divided into blocks of equal size, called "pages." Each process that needs to be executed is allocated a portion of memory equal to one or more pages.

In this way, memory is used optimally, without free spaces.

### 9.6 *Virtual Memory*

There are situations where the memory is insufficient to hold all the processes. The concept of virtual memory is introduced, where the operating system creates what appears to be dedicated memory for each process. Only the necessary parts of code and data for the process are loaded into memory, while the rest is moved to a secondary memory area called the "swap area."

The mechanisms for implementing virtual memory are complex, but modern processors have hardware mechanisms to facilitate its management.

Memory management is a critical function of an operating system, involving the efficient allocation and deallocation of memory resources to various processes. Techniques like paging and virtual memory help optimize memory usage and improve overall system performance, ensuring that the system can handle multiple tasks efficiently and effectively.

### 9.7 *File Systems in Operating Systems*

The file system manager is a module of the operating system responsible for managing information stored on secondary storage devices, also known as mass storage. Secondary



## The physical RAM after the OS organizes 3 processes

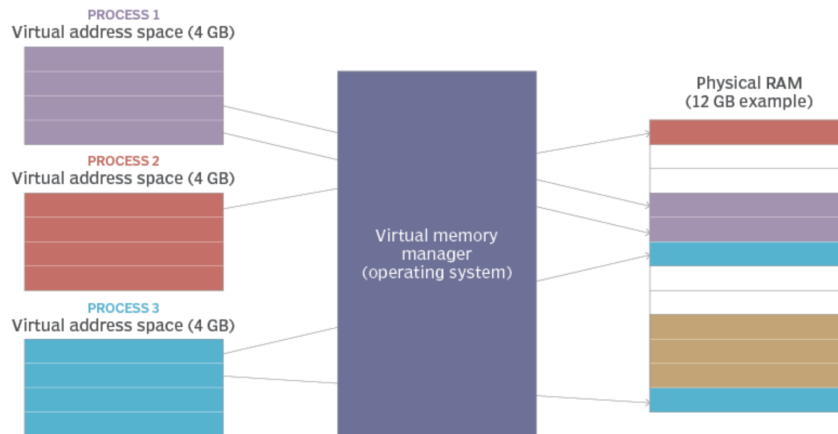


Figure 17: Virtual Memory Management

storage refers to non-volatile memory, where stored data is permanent and does not depend on the state of the computer (on or off). The main role of the file system manager is to control and ensure the correctness and consistency of the information. Various technologies today offer different file systems to users, such as FAT32, NTFS, Apple File System, ext3, and ext4. Regardless of the technology, almost all file systems use a hierarchical approach, where directories are used to group multiple files together.

### 9.7.1 Key Concepts of File Systems

Here are some key concepts related to file systems:

- **File:** a file is a logical unit of data containing information such as text, images, programs, or data.
- **Directory (Folder):** a directory is a logical container for files and other directories. Directories are organized in a tree structure, forming the hierarchy of a file system.
- **Path:** a path represents the location of a file or directory within the file system. For example, `C:\Users\username\Documents` is a path in a Windows file system.
- **Hierarchy:** the directory structure forms a hierarchy that simplifies the organization of files in an orderly manner.

**TYPES OF FILE SYSTEMS** There are various types of file systems, each with its own characteristics and uses. Some examples include NTFS (New Technology File System) and FAT32 on Windows systems, HFS+ on macOS, and ext4 on Linux systems.

**FORMATTING** Formatting a storage device creates a file system on it, preparing it for use. Formatting can affect the type of file system used.

**ACCESS AND SECURITY** File systems often handle access control and data security, defining who can read, write, or execute certain files or directories.

In summary, the file system is a crucial component of any operating system, providing an organizational structure for efficiently storing and retrieving data.

### 9.8 *I/O Device Manager*

The Input/Output (I/O) device manager is a module of the operating system responsible for allocating devices to processes that request access, such as the mouse, keyboard, screen, and so on. To do this, the I/O device manager relies on software called "drivers" (you may have heard the term: "you need to download/update drivers for video cards, for example"), which are generally released by device manufacturers.

#### 9.8.1 Driver Functions

Some main tasks performed by a driver include:

- Managing signals to external peripherals (screens, keyboards, mice, etc.).
- Handling conflicts and synchronization when two processes need the same device simultaneously.

When a process requests access to an I/O device, the driver checks its status. If it is available, it assigns it to the process and marks the device as "busy." This way, synchronization problems between processes that request the same device simultaneously are avoided.

## 9.9 *User Experience*

One of the primary goals of operating systems is to support users and facilitate their tasks. All operating systems implement mechanisms to make the "user experience" easier, i.e., the use of the system by users. The user interface is what simplifies the interaction between users and hardware systems. Two categories of user interfaces can be distinguished:

### 9.9.1 Textual User Interface

Examples include command interpreters on Linux (shell) or PowerShell on Windows.

### 9.9.2 Graphical User Interface

The program outputs are displayed on the screen within windows.

The file system manager, I/O device manager, and user interface are crucial components of any operating system. They work together to provide an efficient, secure, and user-friendly environment for managing hardware resources and data. By understanding these components, we can appreciate the complex orchestration that allows modern operating systems to function seamlessly.

## 9.10 *Security in Operating Systems*

When discussing operating system security, it is essential to consider the external environment in which the system operates to determine appropriate security measures. A server running a critical application in a server room will certainly need physical security policies, such as being housed in a locked room with surveillance mechanisms. In contrast, a personal computer does not require a server room but must be protected from threats such as:

- Unauthorized access
- Data modification or deletion
- Theft of confidential information

### 9.10.1 Authentication

Authentication is a process aimed at recognizing the user on a computer system. Generally, authentication is verified through login mechanisms that use credentials, which can be public like the username and secret like the password. To enhance the security of the login process, passwords are often subject to password security policies, which include a series of requirements, such as:

- Password length: at least 8 characters
- Password complexity: at least one uppercase letter, one number, one symbol
- Password rarity: avoiding common passwords like Password1! or repeated characters
- Periodic password change: for example, every 3 months

In the authentication process, two main actors are involved:

- The user who enters the credentials
- The authentication system, which verifies the credentials provided by the user

An authentication system verifies the user by comparing the credentials entered by the user with the credentials stored in a dedicated directory used to store data for all users who have access to the system, known as the *user repository*.

There are two critical security aspects to consider in this process:

- Credentials in the user repository must be stored encrypted, not in plaintext.
- Credentials in transit at points 2 and 3 should be either encrypted or transmitted over a secure channel.

Ensuring that data is encrypted both in transit between systems and when stored statically on a system is fundamental.

### 9.10.2 Basic Authentication

The example above illustrates basic authentication, as it uses only one step of authentication. For more sensitive applications, additional authentication methods can be added to provide an extra layer of security. Nowadays, the use of passwords alone is highly discouraged, and Multi-Factor Authentication (MFA) is strongly recommended. A second authentication factor, in addition to the password, could be one of the following:

- **Something the user has:** e.g., a phone for authentication via push notification, app, or QR code.
- **Something the user knows:** e.g., a second factor such as a PIN.
- **Something the user is:** e.g., a second factor such as a physical characteristic, like facial recognition or fingerprint.

### 9.10.3 Examples of Second Authentication Factors

The use of a second authentication factor significantly enhances security because an attacker would need both your password and your second authentication method. Examples include Google and Microsoft Authenticator apps, which provide one-time codes, and banking apps that require a second authentication step for transactions.

Second Factor	Description
Push notification – mobile app	The user receives a one-time code on a smartphone app
USB Token	A hardware device generating a unique code
Fingerprint/Facial recognition	Physical characteristic used as a second factor
Phone call	The user receives a call with a numerical code

*Table 11: Examples of Second Authentication Factors*

### 9.10.4 Authorization

While authentication handles user verification, authorization manages the privileges of a verified user. Authorization defines access rights to resources, such as read/write permissions on a file or executing a specific task. It is important to note that authorization follows authentication; once the user is verified, they are granted or denied permission to perform certain functions based on their privileges.

**ACCESS CONTROL MODELS** Authorization can be defined at the operating system level using one of the following models:

- **Discretionary Access Control (DAC):** Uses a decentralized model where the owner of a document can manage its privileges.
- **Mandatory Access Control (MAC):** Uses a centralized model, assigning security labels to resources to classify information and specify authorized user groups.
- **Role-Based Access Control (RBAC):** Assigns privileges based on user roles, adhering to the principle of least privilege.

### 9.10.5 Accounting

Accounting allows tracking of resource usage by a user. An example of accounting is system logs, which track events such as logins, logouts, and configuration changes for each user accessing the system. Given that most attacks on companies originate from internal users, accounting is crucial for identifying the perpetrators of malicious activities. For example, if your account is compromised through phishing, logs can trace all malicious activities performed with your account and possibly even the attempt to steal your credentials, thus protecting you from being held responsible.

Security in operating systems encompasses various aspects, including physical security, authentication, authorization, and accounting. By understanding and implementing these security measures, we can ensure a robust defense against both external and internal threats, providing a secure and reliable environment for users and applications.

\* \* \*