Protocols in Networking -

1. Definition

- A **protocol** is a set of rules/standards that define how data is **formatted, transmitted, and received** over a network.
- They ensure devices from different vendors can communicate properly.

2. Categories of Protocols

A. Communication Protocols

- **TCP (Transmission Control Protocol)**
  - Connection-oriented, reliable, ensures data delivery.
  - Used in web browsing, email, file transfers.
- **UDP (User Datagram Protocol)**
  - Connectionless, faster but unreliable.
  - Used in streaming, gaming, DNS queries.
- **IP (Internet Protocol)**
  - Provides addressing and routing of packets.
  - IPv4 & IPv6.

B. Application Layer Protocols

- **HTTP (HyperText Transfer Protocol)** → Web browsing.
- **HTTPS (Secure HTTP)** → Encrypted web browsing (TLS/SSL).
- **FTP (File Transfer Protocol)** → File sharing.
- **SMTP (Simple Mail Transfer Protocol)** → Sending emails.
- **IMAP/POP3** → Receiving emails.
- **DNS (Domain Name System)** → Resolves domain names to IP addresses.
- **DHCP (Dynamic Host Configuration Protocol)** → Assigns IPs automatically.

C. Security Protocols

- **SSL/TLS (Secure Sockets Layer / Transport Layer Security)** → Encryption for HTTPS.
- **IPSec (Internet Protocol Security)** → VPNs, secure communication.
- **SSH (Secure Shell)** → Secure remote login.

D. Network Management Protocols

- **SNMP (Simple Network Management Protocol)** → Monitor/manage devices.
- **ICMP (Internet Control Message Protocol)** → Error reporting, diagnostics (ping).
- **ARP (Address Resolution Protocol)** → Maps IP → MAC addresses.
- **RARP (Reverse ARP)** → Maps MAC → IP addresses.

3. Importance in Networking

- Enable interoperability between devices.
- Ensure reliable, secure, and efficient communication.
- Define error handling, flow control, and encryption.

4. Ethical Hacking Relevance

- Many **attacks exploit protocol weaknesses**:
  - **ARP Spoofing** → Exploit ARP.
  - **DNS Poisoning** → Exploit DNS.
  - **HTTP Sniffing/Session Hijacking** → Exploit HTTP.
  - **SSL Stripping** → Exploit weak HTTPS usage.
- Understanding protocols is **key to penetration testing & defense**.

📝 Short Notes: Protocols in Networking

- **Definition:** Rules for communication between devices.
- **Communication:** TCP (reliable), UDP (fast), IP (addressing).
- **Application:** HTTP/HTTPS (web), FTP (files), SMTP/IMAP/POP3 (email), DNS (name → IP), DHCP (auto IP).
- **Security:** SSL/TLS, IPSec, SSH.
- **Management:** SNMP, ICMP, ARP, RARP.
- **Hacking Use:** Exploit weak protocols (ARP spoofing, DNS poisoning, HTTP hijacking).

Network Protocol -

1. Definition

A **network protocol** is a **set of rules, standards, and procedures** that define how devices communicate and exchange data in a network.
It ensures that devices with different hardware/software can interact properly.

2. Functions of a Network Protocol

- **Data Formatting** → Defines how data is structured in packets.
- **Addressing** → Assigns sender/receiver addresses (IP, MAC).
- **Routing** → Determines path for data transfer.
- **Error Handling** → Detects and corrects transmission errors.
- **Flow Control** → Manages speed of data transfer.
- **Security** → Provides encryption & authentication.

3. Types of Network Protocols

- **Communication Protocols:** TCP, UDP, IP.

- **Application Protocols:** HTTP, HTTPS, FTP, SMTP, DNS.
- **Security Protocols:** SSL/TLS, IPSec, SSH.
- **Management Protocols:** SNMP, ICMP, ARP.

4. Importance

- Enables **interoperability** between different devices.
- Makes communication **reliable, structured, and secure**.
- Defines **standards** followed worldwide (like TCP/IP).

5. Ethical Hacking Relevance

- Many vulnerabilities arise from **weak or misconfigured protocols**.
  - **ARP Spoofing** → Exploit ARP.
  - **DNS Poisoning** → Exploit DNS.
  - **MITM Attacks** → Exploit HTTP, SSL.
- Understanding protocols = knowing **how to attack & defend networks**.

📝 Short Notes: Network Protocol

- **Definition:** Set of rules for communication in a network.
- **Functions:** Data formatting, addressing, routing, error handling, flow control, security.
- **Types:** Communication (TCP, UDP, IP), Application (HTTP, FTP, DNS), Security (SSL, IPSec, SSH), Management (SNMP, ARP, ICMP).
- **Importance:** Ensures reliable & secure communication.
- **Hacking Use:** Exploiting weak protocols (ARP spoofing, DNS poisoning, MITM).

Types of Protocols -

Protocols are divided based on function and layer in the OSI/TCP-IP model.

1. Communication Protocols (Transport & Network Layer)

- **TCP (Transmission Control Protocol)**
  - **Connection-oriented, reliable (acknowledgements, retransmission).**
  - **Used in: Web browsing, email, file transfer.**
  - **Example Port: 80 (HTTP), 443 (HTTPS).**
- **UDP (User Datagram Protocol)**
  - **Connectionless, faster but unreliable (no error correction).**
  - **Used in: DNS, video streaming, gaming, VoIP.**
  - **Example Port: 53 (DNS).**
- **IP (Internet Protocol)**
  - **Provides addressing & routing of packets.**

- Versions: IPv4 (32-bit), IPv6 (128-bit).

2. Application Layer Protocols

- HTTP (Hypertext Transfer Protocol) → Web browsing (Port 80).
- HTTPS (HTTP Secure) → Encrypted web browsing (Port 443).
- FTP (File Transfer Protocol) → File transfers (Ports 20, 21).
- SMTP (Simple Mail Transfer Protocol) → Sending emails (Port 25).
- IMAP/POP3 → Receiving emails (Ports 143, 110).
- DNS (Domain Name System) → Converts domain names to IPs (Port 53).
- DHCP (Dynamic Host Configuration Protocol) → Assigns IPs automatically (Ports 67, 68).

3. Security Protocols

- SSL/TLS (Secure Sockets Layer / Transport Layer Security) → Encrypts HTTP, emails, VPNs.
- IPSec (Internet Protocol Security) → Provides secure tunneling (VPNs).
- SSH (Secure Shell) → Secure remote login & tunneling (Port 22).

4. Network Management Protocols

- SNMP (Simple Network Management Protocol) → Manage/monitor network devices (Port 161).
- ICMP (Internet Control Message Protocol) → Error messages & diagnostics (used in Ping).
- ARP (Address Resolution Protocol) → Maps IP → MAC addresses.
- RARP (Reverse ARP) → Maps MAC → IP addresses.

5. File Sharing & Remote Access Protocols

- SMB (Server Message Block) → Windows file sharing (Port 445).
- NFS (Network File System) → Unix/Linux file sharing.
- Telnet → Remote login (insecure, Port 23).

6. Routing Protocols

- OSPF (Open Shortest Path First) → Finds shortest path in a network.
- BGP (Border Gateway Protocol) → Routing between ISPs, Internet backbone.
- RIP (Routing Information Protocol) → Older, distance-vector based.

Ethical Hacking Relevance

- Weak Protocols → Telnet (plaintext passwords), FTP (unencrypted files).
- Common Attacks:
  - ARP Spoofing (ARP).
  - DNS Poisoning (DNS).
  - Session Hijacking (HTTP).
  - SSL Stripping (HTTPS).
  - SNMP Exploits (default community strings).

📝 Short Notes: Types of Protocols

- **Communication: TCP, UDP, IP.**
- **Application: HTTP/HTTPS, FTP, SMTP, POP3, IMAP, DNS, DHCP.**
- **Security: SSL/TLS, IPSec, SSH.**
- **Management: SNMP, ICMP, ARP, RARP.**
- **File/Remote: SMB, NFS, Telnet.**
- **Routing: OSPF, BGP, RIP.**
- **Hacking Relevance: Weak protocols → ARP spoofing, DNS poisoning, MITM, SSL stripping, Telnet sniffing.**

How TCP Works -

1. Definition

- **TCP (Transmission Control Protocol)** is a **connection-oriented, reliable transport protocol**.
- It ensures data is delivered **in order, without loss or duplication** between sender and receiver.
- Works at the **Transport Layer (Layer 4)** of the OSI model.

2. Main Features of TCP

- **Connection-Oriented** → Requires connection setup (handshake) before data transfer.
- **Reliable** → Retransmits lost packets.
- **Ordered Delivery** → Data arrives in correct sequence.
- **Error Detection** → Uses checksums to detect corrupted data.
- **Flow Control** → Prevents sender from overwhelming receiver.
- **Congestion Control** → Adjusts transmission rate during heavy traffic.

3. How TCP Works

Step 1: Connection Establishment (Three-Way Handshake)

1. **SYN** → Client sends a request to server (synchronize).
2. **SYN-ACK** → Server acknowledges and sends its own request.
3. **ACK** → Client acknowledges → connection established.

This ensures **both sides are ready** for communication.

Step 2: Data Transmission

- Data is broken into **segments**.

- Each segment has a **sequence number** for ordering.
- Receiver sends back **ACK (Acknowledgement)** after receiving data.
- If ACK not received, sender **retransmits** the data.

Step 3: Connection Termination (Four-Way Handshake)

1. Client sends **FIN** (finish).
2. Server sends **ACK**.
3. Server sends **FIN** when ready to close.
4. Client sends **ACK** → connection closed.

4. Example

- Web browsing:
  - Browser (client) → sends SYN to web server.
  - Server → replies with SYN-ACK.
  - Browser → sends ACK.
  - Now they exchange HTTP data using TCP.

5. Ethical Hacking Relevance

- **TCP Scanning** → Nmap uses TCP SYN scans to detect open ports.
- **Session Hijacking** → Exploit sequence numbers to inject packets.
- **DoS Attacks** → SYN Flooding (overloading server with half-open connections).
- **Sniffing** → Capture TCP packets using Wireshark for analysis.

📝 Short Notes: How TCP Works

- **Definition:** Reliable, connection-oriented transport protocol (Layer 4).
- **Features:** Reliable, ordered, error detection, flow & congestion control.
- **Working:**
  - **3-Way Handshake** (SYN → SYN-ACK → ACK).
  - **Data Transfer** (sequence numbers + ACKs).
  - **4-Way Termination** (FIN/ACK exchange).
- **Hacking Use:** TCP scanning, session hijacking, SYN flooding, packet sniffing.

TCP vs UDP -

1. TCP (Transmission Control Protocol)

- **Connection-Oriented** → Requires 3-way handshake before sending data.
- **Reliable** → Guarantees delivery with acknowledgements (ACK).
- **Ordered Delivery** → Packets arrive in correct sequence.

- **Error Checking** → Checksums, retransmission if errors/loss.
- **Speed** → Slower (due to reliability overhead).
- **Use Cases**: Web browsing (HTTP/HTTPS), email (SMTP/IMAP/POP3), file transfer (FTP).
- **Ports Example**: HTTP (80), HTTPS (443), FTP (21), SMTP (25).

2. UDP (User Datagram Protocol)

- **Connectionless** → No handshake, just sends data.
- **Unreliable** → No guarantee of delivery (best effort).
- **Unordered** → Packets may arrive out of order or get dropped.
- **Lightweight** → No retransmission, lower overhead → faster.
- **Speed** → Faster than TCP.
- **Use Cases**: Streaming (YouTube, Netflix), online gaming, VoIP, DNS queries.
- **Ports Example**: DNS (53), DHCP (67/68), SNMP (161).

3. Key Differences (TCP vs UDP Table)

Feature	TCP	UDP
Connection	Connection-oriented (3-way handshake)	Connectionless (no setup needed)
Reliability	Reliable (ACK, retransmission)	Unreliable (no ACK, no retransmit)
Order	Ordered delivery of packets	No order guaranteed
Speed	Slower	Faster
Error Handling	Yes (checksum, retransmission)	Minimal error handling
Overhead	High (headers: 20–60 bytes)	Low (header: 8 bytes)
Use Cases	Web, email, file transfer	Streaming, gaming, DNS, VoIP
Hacking Risks	SYN flooding, TCP hijacking	DNS spoofing, UDP flooding (DoS)

4. Ethical Hacking Relevance

- **TCP Attacks**:
  - SYN Flooding (DoS attack).
  - TCP Session Hijacking.
  - Port Scanning (Nmap SYN scan).
- **UDP Attacks**:
  - UDP Flooding (DoS attack).
  - DNS Spoofing / Amplification.
  - Exploiting services like SNMP (UDP 161).

📝 Short Notes: TCP vs UDP

- **TCP** → Reliable, connection-oriented, ordered, slower, heavy overhead.
  - Used for web, email, file transfers.
  - Attacks: SYN Flood, Session Hijacking.
- **UDP** → Unreliable, connectionless, unordered, faster, lightweight.
  - Used for streaming, gaming, VoIP, DNS.
  - Attacks: UDP Flood, DNS Spoofing.

TCP Flags -

1. What are TCP Flags?

- TCP **flags** are **control bits** in the TCP header used to manage the **state and control** of a TCP connection.
- Each flag is **1 bit** (either ON = 1 or OFF = 0).
- They help in **establishing, maintaining, and terminating** connections.

2. Important TCP Flags

🔹 **SYN (Synchronize)**

- Used to **initiate a connection** between client and server.
- Part of **3-way handshake**.
- Example: SYN → SYN-ACK → ACK.

🔹 **ACK (Acknowledgement)**

- Confirms that data/connection requests were **received successfully**.
- Present in almost every packet after handshake.

🔹 **FIN (Finish)**

- Used to **gracefully terminate a connection**.
- In **4-way handshake** for closing TCP connections.

🔹 **RST (Reset)**

- Immediately **resets/terminates** a connection (forcefully).
- Used when a port is **closed** or an invalid packet is received.

🔹 **PSH (Push)**

- Instructs receiver to **immediately push data** to the application layer (without buffering).
- Used in interactive data (e.g., Telnet).

🔹 **URG (Urgent)**

- Marks certain data as **urgent**.
- Uses the **urgent pointer field** to indicate high-priority data.
- Rarely used today.

3. Ethical Hacking Relevance

- **SYN Flag** → Used in SYN flooding (DoS attack).
- **ACK Flag** → Used in ACK scans to check firewall rules.
- **FIN Flag** → Used in stealth scans (Nmap FIN scan).
- **RST Flag** → Used to quickly close unwanted connections.
- **PSH Flag** → Can be abused in some exploits for pushing malicious payloads.
- **URG Flag** → Used in older attacks (e.g., Urgent Pointer exploit).

📝 Short Notes: TCP Flags

- **SYN** → Start connection (3-way handshake).
- **ACK** → Acknowledges received data/packets.
- **FIN** → Gracefully end connection.
- **RST** → Forcefully reset connection.
- **PSH** → Push data immediately to application.
- **URG** → Marks urgent data (rarely used).

👉 **Hacking Use** → TCP flags are heavily used in **port scanning, DoS attacks, firewall evasion, and session hijacking**.

Working of Routers -

1. What is a Router?

- A **router** is a networking device that **connects multiple networks** (usually LAN → WAN/Internet).
- Works at **Layer 3 (Network Layer)** of the OSI model.
- Uses **IP addresses** to forward data packets between networks.

2. How Routers Work (Step by Step)

1. **Packet Receiving**
   - Router receives a data packet from one network (e.g., your computer → router).
   - Each packet has a **Source IP** and a **Destination IP**.
2. **Reading Destination IP**
   - Router looks at the **destination IP address** inside the packet.
3. **Routing Table Lookup**
   - Router checks its **routing table** (list of networks and best paths).
   - Determines **where to forward** the packet (next hop).
4. **Forwarding Decision**
   - If destination is in the **same network (LAN)** → router sends packet directly.
   - If destination is in a **different network (Internet)** → router forwards it to the **next router / ISP gateway**.
5. **Packet Transmission**
   - Packet is encapsulated with the **next hop's MAC address** (using ARP).
   - Router sends it out through the correct **interface/port**.

3. Key Functions of a Router

- **Packet Forwarding** → Sends data to correct network.
- **Routing** → Uses protocols like RIP, OSPF, BGP for path selection.
- **NAT (Network Address Translation)** → Converts private IPs to public IPs for Internet access.
- **Firewall/ACLs** → Filters traffic for security.
- **DHCP (sometimes)** → Provides IP addresses to devices.

4. Ethical Hacking Relevance

- **Router Attacks**:
  - Password Cracking (default credentials).
  - Misconfigured ACL exploitation.
  - DNS Hijacking (changing DNS settings in router).
  - DoS attacks (overloading router CPU/memory).
- **Router Fingerprinting**: Hackers identify router models for targeted exploits.
- **Port Forwarding Abuse**: Used by attackers to gain access to internal devices.

📝 Short Notes: Working of Routers

- **Definition** → Layer 3 device, connects multiple networks using IP addresses.
- **Process** → Receive packet → check destination IP → routing table lookup → forward packet.
- **Functions** → Routing, NAT, Packet filtering, ACLs, DHCP (optional).

- **Protocols** → RIP, OSPF, BGP.
- **Hacking Use** → Attacks: password cracking, DNS hijack, DoS, ACL bypass.

DHCP & ARP -

1. DHCP (Dynamic Host Configuration Protocol)

- **Definition** → A protocol that **automatically assigns IP addresses** and other network configs (subnet mask, gateway, DNS) to devices.
- **Layer** → Application Layer (but uses UDP for transport).
- **Ports** → UDP 67 (server), UDP 68 (client).

DHCP Working (DORA Process)

1. **Discover** → Client broadcasts request for an IP.
2. **Offer** → DHCP server replies with an available IP.
3. **Request** → Client requests that offered IP.
4. **Acknowledge** → Server confirms assignment, client gets IP.

Ethical Hacking Relevance

- **DHCP Starvation Attack** → Attacker requests all available IPs, exhausting DHCP pool → DoS for legitimate users.
- **Rogue DHCP Server** → Attacker sets up fake DHCP, assigns malicious gateway/DNS → leads to **MITM (Man-in-the-Middle)** attack.

2. ARP (Address Resolution Protocol)

- **Definition** → A protocol used to **map IP addresses (Layer 3)** to **MAC addresses (Layer 2)** inside a LAN.
- **Layer** → Works between **Layer 2 (Data Link)** and **Layer 3 (Network)**.

ARP Working

1. Device wants to send packet to 192.168.1.5.
2. It checks its ARP table (IP ⇔ MAC mapping).
3. If not found → broadcasts an **ARP Request** → “Who has 192.168.1.5?”
4. The device with that IP replies with its **MAC address**.
5. The sender stores this mapping in its ARP table.

Ethical Hacking Relevance

- **ARP Spoofing / Poisoning** → Attacker sends fake ARP replies, linking their MAC to victim's IP (or gateway).
  - Result: Victim sends traffic to attacker → enables **MITM attack, sniffing, or DoS**.
- Tools: arpspoof, ettercap, Bettercap.

📝 Short Notes: DHCP & ARP

DHCP

- Assigns IP, subnet, gateway, DNS automatically.
- Process: **DORA (Discover → Offer → Request → Acknowledge)**.
- Ports: **UDP 67, 68**.
- **Hacking**: DHCP starvation, Rogue DHCP (MITM).

ARP

- Maps **IP → MAC** in a LAN.
- Works by ARP Request (broadcast) & Reply (unicast).
- Stored in ARP Table.
- **Hacking**: ARP Spoofing/Poisoning → MITM, DoS.

SIP & DIP -

1. SIP (Source IP Address)

- **Definition** → The IP address of the device that **sends** the data packet.
- Found in the **IP header** of every packet.
- Example: If your computer (192.168.1.10) sends a request to Google, your **SIP = 192.168.1.10**.

Ethical Hacking Relevance

- Attackers often **spoof SIP** to hide identity (IP Spoofing).
- Used in **DoS/DDoS attacks** → attacker changes SIP to flood victim with fake requests.
- Firewalls and IDS often filter/block based on SIP.

2. DIP (Destination IP Address)

- **Definition** → The IP address of the device that is **supposed to receive** the data packet.
- Also found in the **IP header**.
- Example: When you send a packet to Google's server → **DIP = 142.250.xx.xx (Google's server IP)**.

Ethical Hacking Relevance

- Knowing the **DIP** is critical for attackers to target the victim system/server.
- In **MITM attacks**, attackers trick victims into sending packets to the wrong DIP.
- Firewalls, routers, and NAT devices use DIP to **decide forwarding rules**.

3. SIP & DIP Together in Networking

- Every packet has **SIP → DIP** mapping (sender → receiver).
- Routers read **DIP** to decide where to forward.
- NAT changes **SIP/DIP** to allow private network devices to access the Internet.

📝 Short Notes: SIP & DIP

- **SIP (Source IP)** → IP of sender (who sends packet).
- **DIP (Destination IP)** → IP of receiver (who should get packet).
- Both are found in **IP header** of packets.
- **Hacking Relevance** →
  - SIP Spoofing (hide attacker identity, DDoS).
  - DIP Targeting (attacks, scanning, MITM).
  - Firewalls & NAT use SIP/DIP for filtering.

Requests & Responses -

1. Request

- A **message sent from client → server** asking for data or service.
- Contains:
  - **Method** (e.g., GET, POST, PUT, DELETE in HTTP).
  - **Headers** (extra info like User-Agent, cookies, content type).
  - **Body (optional)** → Data sent (e.g., login form, JSON).

Example (HTTP Request)

GET /index.html HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

- Client is requesting /index.html from server.

2. Response

- A **message sent from server → client** with requested data or result.
- Contains:
  - **Status Code** (e.g., 200 OK, 404 Not Found, 500 Server Error).
  - **Headers** (server type, content length, cookies).
  - **Body** (actual content like HTML, JSON, file).

Example (HTTP Response)

HTTP/1.1 200 OK

Content-Type: text/html

Content-Length: 120

<html><body>Hello User</body></html>

- Server replied with webpage data.

3. Request-Response Cycle

1. Client sends a **request** (e.g., browser → server).
2. Server processes it.
3. Server sends a **response** back.
4. Connection may close or stay open (persistent).

4. Ethical Hacking Relevance

- **Intercepting Requests/Responses** → Tools like **Burp Suite, OWASP ZAP, Wireshark** allow hackers to capture/modify them.
- **Manipulation**:
  - Modify **request parameters** (e.g., price change in shopping cart).
  - View/steal **response data** (e.g., hidden API info).
- **Attacks**:
  - SQL Injection → modifying request input.
  - XSS → injecting scripts via request.
  - Session Hijacking → stealing response cookies.

📝 Short Notes: Requests & Responses

- **Request** → Client → Server, contains method, headers, body.
- **Response** → Server → Client, contains status code, headers, body.
- **Cycle** → Request sent → Server processes → Response returned.
- **Hacking** →

- Interception (Burp Suite, ZAP, Wireshark).
- Modify requests (SQLi, XSS).
- Steal responses (cookies, tokens).

Request -

1. Definition

A **request** is a message sent by a **client (user/device)** to a **server** asking for resources, services, or actions.

- Examples:
  - Opening a webpage → browser sends a **request**.
  - Logging into Facebook → your form data goes as a **request**.
  - Sending WhatsApp message → your app sends a **request** to WhatsApp servers.

2. Structure of a Request (HTTP Example)

A request generally has 3 parts:

1. **Request Line**
   - Contains: **Method + Resource + Protocol version**
   - Example:
   - GET /index.html HTTP/1.1
2. **Headers**
   - Metadata about request.
   - Examples:
     - Host: www.example.com
     - User-Agent: Mozilla/5.0
     - Cookie: sessionid=12345
3. **Body (Optional)**
   - Data sent with request.
   - Example: Form data, JSON, XML, file upload.
   - Only in methods like **POST, PUT**.

3. Types of Requests (Common in HTTP)

- **GET** → Request data (e.g., webpage, API data).
- **POST** → Send data (e.g., login form, upload).
- **PUT** → Update resource.
- **DELETE** → Remove resource.
- **HEAD** → Fetch headers only.

4. Ethical Hacking Relevance

- Attackers can **capture & modify requests** before they reach server.
- Tools: **Burp Suite, OWASP ZAP, Wireshark**.
- Attacks:
  - SQL Injection (modify request input).
  - XSS (inject code in request).
  - Session Hijacking (steal cookies in request headers).

📝 Short Notes: Request

- **Definition** → Message from client → server asking for data/service.
- **Parts** → Request line, headers, body (optional).
- **Types (HTTP)** → GET, POST, PUT, DELETE, HEAD.
- **Ethical Hacking** → Requests can be intercepted & modified → SQLi, XSS, Session Hijacking.

Response -

1. Definition

A **response** is a message sent by the **server → client** after processing a request.

- It contains the **status of the request** and **data/content** if available.
- Example:
  - You request a webpage → server responds with HTML file.
  - You send login details → server responds with “Login Success” or “Invalid Password.”

2. Structure of a Response (HTTP Example)

A response has 3 main parts:

1. **Status Line**
   - Shows request result → Protocol + Status Code + Status Message
   - Example:
   - HTTP/1.1 200 OK
2. **Headers**
   - Extra information about server/data.
   - Examples:
     - Content-Type: text/html
     - Set-Cookie: sessionid=12345
     - Server: Apache/2.4.1

3. **Body (Optional)**
   - The actual content.
   - Examples: HTML page, JSON, images, files.
   - May be empty in responses like **HEAD**.

3. Common Response Status Codes

- **1xx (Informational)** → Request received, processing.
- **2xx (Success)** → Request successful.
  - 200 OK, 201 Created.
- **3xx (Redirection)** → Redirect to another resource.
  - 301 Moved Permanently, 302 Found.
- **4xx (Client Error)** → Problem in request.
  - 400 Bad Request, 401 Unauthorized, 403 Forbidden, 404 Not Found.
- **5xx (Server Error)** → Problem in server.
  - 500 Internal Server Error, 503 Service Unavailable.

4. Ethical Hacking Relevance

- Hackers analyze responses to extract info:
  - **Server headers** leak technology (Apache, Nginx, PHP version).
  - **Error messages** may reveal DB details → useful in SQL Injection.
  - **Cookies in response** may store session tokens (target for hijacking).
- Tools: **Burp Suite, Wireshark, curl**.
- Attacks:
  - **Response Manipulation** (modifying content).
  - **Information Disclosure** (finding sensitive data in responses).

📝 Short Notes: Response

- **Definition** → Message from server → client with status & data.
- **Parts** → Status line, headers, body.
- **Status Codes** →
  - 1xx Info, 2xx Success, 3xx Redirect, 4xx Client Error, 5xx Server Error.
- **Hacking Use** → Analyze headers, error messages, cookies → exploits like SQLi, Session Hijacking.

Types of Requests & Responses -

1. Types of Requests (Mainly HTTP Methods)

A **request** is how the client asks the server to do something.
Common request types:

1. **GET** → Retrieve data (read-only).
   - Example: GET /index.html → Fetches webpage.
   - Data sent in **URL**.
2. **POST** → Send data (create resource).
   - Example: Submitting login/signup form.
   - Data sent in **body**.
3. **PUT** → Update existing resource.
   - Example: Update user profile info.
4. **DELETE** → Remove resource.
   - Example: Delete a comment.
5. **HEAD** → Get **headers only** (no body).
   - Example: Check if a file exists.
6. **OPTIONS** → Ask server what methods are allowed.
   - Example: Check if CORS requests are permitted.
7. **PATCH** → Partial update of a resource.
   - Example: Update only the password field of a user.

2. Types of Responses (Status Codes)

A **response** is how the server answers the client.
Main response types:

1. **1xx → Informational**
   - Example: 100 Continue
2. **2xx → Success**
   - 200 OK → Successful request
   - 201 Created → New resource created
   - 204 No Content → Success but no body
3. **3xx → Redirection**
   - 301 Moved Permanently → New URL
   - 302 Found → Temporary redirect
   - 304 Not Modified → Cached version
4. **4xx → Client Error**
   - 400 Bad Request → Invalid syntax
   - 401 Unauthorized → Login needed
   - 403 Forbidden → No permission
   - 404 Not Found → Resource missing
5. **5xx → Server Error**
   - 500 Internal Server Error → General failure
   - 502 Bad Gateway → Wrong upstream response
   - 503 Service Unavailable → Server overloaded

3. Ethical Hacking Relevance

- **Requests** → Hackers modify them to inject attacks.
  - Example: SQLi, XSS, CSRF.
- **Responses** → Hackers analyze them for sensitive info.
  - Example: Error pages revealing DB, cookies in headers, server version leaks.
- Tools: **Burp Suite, Wireshark, curl, Postman**.

📝 Short Notes: Types of Requests & Responses

Requests (HTTP Methods)

- GET → Retrieve
- POST → Create
- PUT → Update
- DELETE → Remove
- HEAD → Headers only
- OPTIONS → Allowed methods
- PATCH → Partial update

Responses (Status Codes)

- 1xx → Info
- 2xx → Success (200, 201, 204)
- 3xx → Redirect (301, 302, 304)
- 4xx → Client Error (400, 401, 403, 404)
- 5xx → Server Error (500, 502, 503)

HTTP Request -

1. Definition

- **HTTP Request** = A message sent from **client (browser, app) → server** to ask for data or action.
- Works on **HyperText Transfer Protocol (HTTP/HTTPS)**.
- Foundation of web communication (client-server model).

2. Structure of HTTP Request

An HTTP Request has 4 main parts:

1. **Request Line**

- Contains **Method + URL + HTTP Version**
- Example:
- GET /index.html HTTP/1.1

2. **Headers**
   - Extra info about request/client.
   - Examples:
     - Host: www.example.com
     - User-Agent: Mozilla/5.0
     - Accept-Language: en-US
     - Cookie: sessionid=12345
3. **Body (Optional)**
   - Contains data sent by client (mainly in POST/PUT/PATCH).
   - Example:
   - {
   - "username": "hacker",
   - "password": "12345"
   - }
4. **Blank Line**
   - Separates headers from body.

3. Common HTTP Methods (Requests Types)

- **GET** → Fetch data.
- **POST** → Submit data.
- **PUT** → Update resource.
- **DELETE** → Remove resource.
- **HEAD, PATCH, OPTIONS** → Other specific tasks.

4. Example of HTTP Request

POST /login HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0

Content-Type: application/json

Cookie: sessionid=xyz123

{

"username": "admin",

"password": "12345"

}

5. Ethical Hacking Relevance

- Attackers can **modify HTTP requests** to exploit vulnerabilities:
  - **SQL Injection** → Inject malicious queries.
  - **Cross-Site Scripting (XSS)** → Inject scripts.
  - **Cookie Manipulation** → Hijack sessions.
  - **Header Injection** → Exploit weak configurations.
- Tools: **Burp Suite, OWASP ZAP, Wireshark, curl**.

📝 Short Notes: HTTP Request

- **Definition** → Client → Server message asking for resource/action.
- **Parts** → Request line, Headers, Body, Blank line.
- **Methods** → GET, POST, PUT, DELETE, HEAD, PATCH, OPTIONS.
- **Hacking Use** → Request manipulation → SQLi, XSS, CSRF, Cookie theft.