SOC TIER 1 – Technical Interview Preparation Guide

1. Ports

What are Ports?

- Ports are **virtual communication endpoints** used by computers to manage different types of traffic.
- Think of a port as a door number on a building — the building is your IP address, and the door is the port.

Types:

- **TCP (Transmission Control Protocol):** Reliable, connection-based (e.g., web browsing, emails).
- **UDP (User Datagram Protocol):** Faster, connectionless (e.g., streaming, VoIP).

Common Ports to Know:

Protocol	Port	Description
HTTP	80	Web traffic (not encrypted)
HTTPS	443	Secure web traffic
SSH	22	Secure shell (remote admin)
FTP	21	File transfer
SMTP	25	Sending emails
DNS	53	Domain name resolution
RDP	3389	Remote desktop
SMB	445	File sharing on Windows networks

Simpler Explanation - Ports

Imagine your computer is a big office building. This building has many doors. Each door is like a **port** — it's a specific entrance that lets different kinds of visitors (data) come in or go out.

- **Each port has a number** (like door number 80 or 443).
- Different services or apps use different ports, so they know where to "listen" for messages.

Why are Ports Important?

When your computer talks to other computers over the internet, it needs to keep different conversations separate — just like you wouldn't want all your phone calls, emails, and deliveries going through the same door.

Example:

- If you're browsing a website, your browser talks to port **80** or **443** on the web server.
- If you're sending an email, your email program talks to port **25** (SMTP).
- If you're using a remote connection, it might use port **22** (SSH).

Two Main Types of Ports:

- **TCP ports:** Like a phone call — a two-way, reliable conversation.
- **UDP ports:** Like a quick message — fast but no guarantee it arrives.

Common Port Numbers to Know:

Port Number	Service
80	HTTP (web)
443	HTTPS (secure web)
22	SSH (remote login)
25	SMTP (email send)
53	DNS (address lookup)

In short:

Ports are like doors on your computer that let different kinds of network traffic in and out. Each door has its own number and purpose.

2. Three-Way Handshake (TCP Handshake)

Purpose:

Used to establish a **reliable connection** between two systems using TCP.

Steps:

1. **SYN:** Client sends a synchronize packet to the server saying "I want to start a connection."
2. **SYN-ACK:** Server replies "I received your SYN and here's my response."
3. **ACK:** Client replies again to confirm, and now the connection is established.

➡️ **Why it's important in security:** Many attacks, like SYN floods, target this process.

(Simplified) What is the Three-Way Handshake?

It's the way two computers **start a reliable connection** using TCP (the "phone call" of the internet), making sure both sides are ready before sending actual data.

Imagine This Scenario:

You want to call your friend and have a good conversation without interruptions. But first, you both need to say:

- "Hey, I want to talk."
- "Okay, I'm listening."
- "Great, let's start."

This is exactly what happens between two computers with the three-way handshake.

The 3 Steps Explained Simply:

1. SYN (Synchronize) – The "Hello, I want to talk"

- Your computer (the client) sends a special message called **SYN** to the server.
- This message says: "Hey server, I want to start a connection!"

2. SYN-ACK (Synchronize-Acknowledge) – The "Okay, I hear you, I'm ready"

- The server receives your SYN and replies with **SYN-ACK**.
- It means: "I got your message, and I'm ready too!"

3. ACK (Acknowledge) – The "Great, let's start talking"

- Your computer sends back an **ACK** message.
- This says: "Thanks for confirming. Let's start exchanging data!"

After This:

- Both computers know the connection is set up.
- Now, they can start sending actual data safely and reliably.

Why is This Important?

- It prevents data loss by making sure both sides are ready.
- Helps computers keep track of the connection.
- Allows TCP to provide a **reliable** communication channel.

Quick Analogy:

- SYN = "Can we talk?"
- SYN-ACK = "Yes, I'm listening."
- ACK = "Thanks, let's talk."

Bonus: What Can Go Wrong?

- **SYN flood attack:** An attacker sends many SYN messages but never replies with ACK, causing the server to wait and get overwhelmed.
- SOC analysts watch for lots of half-open connections to detect this.

3. Common Attacks

Type	Description	Example
Phishing	Tricking users to give credentials or download malware	Fake login pages
SQL Injection	Inserting SQL code into forms to access databases	' OR '1'='1
XSS (Cross-Site Scripting)	Injecting malicious JavaScript into websites	<script>alert('xss')</script>
MITM (Man-in-the-Middle)	Attacker intercepts communication	Intercepting login over public Wi-Fi
Brute Force	Trying many password combinations	Using tools like Hydra
DDoS	Overloading a system with traffic	Botnets flooding a server
Ransomware	Encrypts files and demands payment	WannaCry

🛡️ Common Cyber Attacks – Detailed Explanation for SOC Analysts

Phishing

What It Is:

Phishing is when an attacker tries to trick someone into giving away sensitive information — like passwords, credit card numbers, or clicking on malicious links — by pretending to be a trusted source (e.g., a bank, coworker, or government agency).

How It Works:

- The attacker sends an email that **looks real**, but contains a **fake link**.
- When the user clicks it and enters info, it goes **to the attacker**, not the real site.

Types:

- **Spear Phishing:** Targeted to a specific person.
- **Whaling:** Targets high-level individuals like executives.
- **Smishing:** Phishing over SMS text.
- **Vishing:** Phishing by voice call.

SOC Analyst's Role:

- Detect phishing attempts in **email logs** (strange sender domains, suspicious links).
- Look for users visiting **phishing URLs** in web proxy logs.
- Check if credentials were entered on fake login pages.

Malware (Malicious Software)

What It Is:

Software designed to **harm** or **steal data** from a system.

Types of Malware:

Type	**Description**
Virus	Attaches to files and spreads when opened.
Worm	Spreads on its own across networks.
Trojan	Pretends to be a normal program but hides harmful code.
Spyware	Secretly monitors your activity.
Ransomware	Encrypts files and demands payment to unlock them.

SOC Analyst's Role:

- Detect suspicious files using **EDR** or **antivirus alerts**.
- Investigate strange **file names**, **high CPU usage**, or **unauthorized access** attempts.
- Watch for **C2 (Command & Control)** communication attempts in logs.

🔍 What is EDR?

EDR stands for **Endpoint Detection and Response**.

It's a **security tool** that protects **endpoints** — meaning **computers**, **laptops**, **servers**, or any other device connected to your network.

🧠 Think of EDR Like This:

Imagine antivirus, but **smarter**, **faster**, and able to **detect modern threats**, not just known viruses.

While antivirus is like a **guard dog that barks at known intruders**, **EDR is like a detective** that:

- Watches everything happening on the device
- Notices strange or suspicious behavior
- Responds to stop threats
- Helps security teams investigate attacks

✅ What Does EDR Do?

Feature	What It Means
Detection	Constantly monitors devices for suspicious activity like unusual file changes, weird connections, or processes.
Response	Can take automatic action — like isolating the device, killing malware, or blocking a file.
Visibility	Gives SOC teams full logs and visibility into what happened on a device (like a security camera timeline).
Investigation	Helps analysts understand how the attack happened, what was affected, and how to fix it.
Threat Hunting	Allows analysts to actively search across all devices for signs of a threat (called "IOCs" — Indicators of Compromise).

SQL Injection (SQLi)

What It Is:

A web attack where attackers insert **malicious SQL commands** into a form field to manipulate or steal data from a database.

Example:

A login form asks for:

Username: ' OR 1=1 --

Password: [anything]

This tricks the database into thinking the login is valid.

What They Can Do:

- Read sensitive data (usernames, passwords, credit card info)
- Modify or delete data
- Bypass login

SOC Analyst's Role:

- Look for **strange URL patterns or inputs** in web server logs.
- Monitor for **SQL errors** in responses.
- Work with devs to ensure input validation is in place.

4. Cross-Site Scripting (XSS)

What It Is:

An attacker injects **malicious JavaScript** into a trusted website. When other users load the page, the code runs in their browser.

Example:

A comment section allows <script>alert('hacked')</script>. Other users see the popup or worse — their session cookies get stolen.

Types:

- **Stored XSS:** Code is saved on the server.
- **Reflected XSS:** Code is in the URL and run immediately.
- **DOM-based XSS:** Manipulates the page structure through client-side code.

Stored XSS (a.k.a. Persistent XSS)

Simple Explanation:

Stored XSS is when the bad code is **saved (stored)** in the website’s database, and then shown to **anyone** who visits that page.

Real-Life Example:

Let’s say you visit a blog site with a comment section.

1. A hacker writes this comment:

Great article! <script>alert('I hacked you!');</script>

2. The website **stores** that comment in its database.
3. When **any user** visits the article, the browser runs the script.
4. The alert pops up — or worse, the script steals the user's session cookie!

Why it's dangerous:

- Anyone who sees the page is affected.
- The code stays on the site until removed.

Reflected XSS (a.k.a. Non-Persistent XSS)

Simple Explanation:

Reflected XSS happens when the bad code is **not stored**, but is immediately reflected back from the server into the browser — usually through a URL.

Real-Life Example:

A hacker sends you a **link** like this:

https://example.com/search?q=<script>alert('You got hacked')</script>

1. You click the link.

2. The website reflects the q parameter (search query) back onto the page without checking it.
3. The script runs in **your** browser only.
4. Again, it could steal your cookies or redirect you.

Why it's dangerous:

- It only affects the person who clicks the link.
- It’s often used in **phishing attacks**.

SOC Analyst’s Role:

- Check WAF logs for XSS patterns (like <script> or alert()).
- Analyze unusual activity in session logs (many users suddenly logged out?).
- Flag sites that don’t sanitize input/output properly.

Brute Force Attacks

What It Is:

The attacker tries many combinations of usernames and passwords until they find the correct one.

Types:

- **Online:** Tries login pages directly.
- **Offline:** Cracks hashed passwords using tools.

Example Tools:

- Hydra
- John the Ripper
- Hashcat

SOC Analyst’s Role:

- Detect **many failed login attempts** from the same IP.
- Look for login attempts from unusual locations.
- Use rate-limiting and lockout policies to stop brute force attacks.

DDoS (Distributed Denial of Service)

What It Is:

Attackers **flood a server or network** with so much traffic that it crashes or becomes unusable.

How:

- They use a **botnet** (thousands of infected devices) to send fake requests.

Types:

- **Volumetric:** Overload with traffic.
- **Protocol-based:** Exploit flaws in protocols (e.g., SYN flood).
- **Application Layer:** Target the web app directly (Layer 7 DDoS).

SOC Analyst’s Role:

- Use monitoring tools to detect **traffic spikes**.
- Coordinate with **ISPs or Cloud providers** to block IPs or use anti-DDoS services.
- Activate **incident response plans** and update firewall rules.

MITM (Man-in-the-Middle)

What It Is:

An attacker intercepts communication between two parties without them knowing, often to **steal data** or **modify messages**.

Example:

- On a public Wi-Fi, an attacker tricks your device into thinking they're the router.
- All your traffic (emails, passwords) goes through them.

SOC Analyst's Role:

- Look for **suspicious ARP traffic or duplicate IPs** (in Layer 2).
- Use **HTTPS** and **TLS inspection** to prevent eavesdropping.
- Detect **SSL stripping** or unexpected certificate changes.

Ransomware

What It Is:

A type of malware that **encrypts your files** and demands money (usually in Bitcoin) to get them back.

Signs:

- Files suddenly renamed (e.g., document.docx.locked)
- A note appears saying "Your files are encrypted"
- High CPU usage, unknown processes running

SOC Analyst's Role:

- Detect unusual file behavior (mass encryption, renaming).
- Isolate infected machines quickly.
- Work with backup teams to **restore systems** without paying.
- Monitor for **initial access points** (often via phishing or RDP).

Credential Stuffing

What It Is:

Attackers use **username/password combos leaked from other sites** and try them on your systems, hoping people reuse the same password.

Example:

A user uses the same password on Facebook and your company system. If Facebook is hacked, the attacker tries the same combo at work.

SOC Analyst's Role:

- Detect **many login attempts using known breach credentials**.
- Monitor dark web leaks.
- Encourage use of **MFA (Multi-Factor Authentication)**.

Zero-Day Exploits

What It Is:

A **zero-day** is a vulnerability that no one (not even the software creator) knows about — until it's found and attacked in the wild.

SOC Analyst's Role:

- Stay updated on **threat intelligence feeds**.
- Look for **suspicious behavior**, not just known signatures.
- Help contain breaches while patches are being developed.

Summary Table of Attacks

Attack Type	**Goal**	**What to Monitor/Detect**
Phishing	Steal credentials/data	Email logs, DNS requests
Malware	Damage/steal data	EDR alerts, suspicious files
SQL Injection	Steal/manipulate DB	Web logs, error responses
XSS	Run code in browsers	WAF logs, suspicious scripts
Brute Force	Guess passwords	Login attempts, failed logins
DDoS	Crash system	Traffic spikes, performance issues
MITM	Intercept communication	Duplicate IPs, ARP anomalies
Ransomware	Encrypt files	File access patterns, alerts
Credential Stuffing	Account takeover	Repeated login attempts
Zero-Day	Exploit unknown flaws	Behavior analysis, threat intel

4. WAF (Web Application Firewall)

What is a WAF?

A security system that **protects web applications** by filtering and monitoring HTTP traffic.

What it does:

- Blocks SQL injections, XSS, file inclusion, and more.
- Works at the **application layer** (Layer 7 of OSI).
- Can be hardware-based or cloud-based (e.g., Cloudflare, AWS WAF).

➡️ **Use case:** You deploy a website. A WAF protects it from common attacks like input injection.

---

5. OSI Model

7-Layer Model of Network Communication:

Layer	Name	Function	Example
7	**Application**	User interface	HTTP, FTP
6	**Presentation**	Data formatting, encryption	SSL, JPEG
5	**Session**	Connection management	NetBIOS
4	**Transport**	End-to-end delivery	TCP, UDP
3	**Network**	Routing packets	IP
2	**Data Link**	MAC addressing	Ethernet
1	**Physical**	Hardware transmission	Cables, NICs

➡️ **Important for SOC:** Helps you pinpoint where attacks happen and what tools operate on which layer (e.g., WAF at Layer 7, Firewall at Layer 4/3).

OSI Model Deep Dive

The **OSI (Open Systems Interconnection) model** is a conceptual framework used to understand how data travels across a network. It divides network communication into **7 layers**, each with a specific role. Knowing these layers helps SOC analysts identify where attacks happen and which defenses apply.

Overview of the 7 Layers

Layer #	Name	Function Summary	Key Protocols/Technologies
7	Application	User-facing services and apps	HTTP, FTP, SMTP, DNS, Telnet
6	Presentation	Data format and encryption	SSL/TLS, JPEG, MPEG
5	Session	Managing connections (sessions)	NetBIOS, SAP, RPC
4	Transport	End-to-end data delivery	TCP, UDP
3	Network	Routing and addressing	IP, ICMP, OSPF
2	Data Link	Framing and MAC addressing	Ethernet, Wi-Fi (802.11), ARP
1	Physical	Transmission of bits over media	Cables, hubs, repeaters, NIC cards

---

Layer-by-Layer Explanation

Layer 7: Application Layer

- **Role:** This is where applications and user services operate — the “front door” for network communications. It’s what users interact with directly.
- **Examples:** Web browsers (HTTP/HTTPS), email clients (SMTP/POP3), file transfer (FTP).
- **Security relevance:** This layer is targeted by many attacks such as **SQL injection, cross-site scripting (XSS), and malware delivery**.
- **Tools:** WAFs operate mainly here to inspect and filter web traffic.

Layer 6: Presentation Layer

- **Role:** Responsible for **data formatting, encryption, and compression**. Converts data from the application into a common format for transmission.
- **Examples:** SSL/TLS (for encryption), data formats like JPEG, MPEG.
- **Security relevance:** Encryption/decryption (like HTTPS) happens here — if SSL/TLS is weak or misconfigured, attackers can intercept or manipulate data.
- **SOC Tip:** Monitor for SSL certificate anomalies or weak cipher suites.

Layer 5: Session Layer

- **Role:** Establishes, manages, and terminates communication sessions between devices.
- **Examples:** NetBIOS sessions, RPC (Remote Procedure Calls).
- **Security relevance:** Session hijacking attacks occur here, where attackers take over a valid session (e.g., stealing session cookies).
- **SOC Tip:** Look for unusual session establishment or unexpected session terminations.

Layer 4: Transport Layer

- **Role:** Provides **end-to-end communication** control and reliability. It segments data and manages flow control and error checking.
- **Protocols:**
  - **TCP (Transmission Control Protocol):** Reliable, connection-oriented; includes the three-way handshake.
  - **UDP (User Datagram Protocol):** Faster, connectionless, but no error correction.
- **Security relevance:**
  - TCP handshake can be exploited (SYN flood attacks).
  - Ports are defined at this layer, so monitoring open/closed ports is crucial.
- **SOC Tip:** Analyze suspicious port scans or unusual TCP/UDP activity.

Layer 3: Network Layer

- **Role:** Handles **logical addressing and routing** so packets find their destination across different networks.
- **Protocols:** IP (Internet Protocol), ICMP (ping), OSPF (routing).
- **Security relevance:**
  - IP spoofing attacks happen here (fake IPs to hide attackers).
  - ICMP can be used in reconnaissance (ping sweeps).
- **SOC Tip:** Monitor for abnormal IP traffic patterns or ICMP flood attacks.

Layer 2: Data Link Layer

- **Role:** Transfers data between devices on the **same physical network** (LAN). Handles MAC addressing and error detection.
- **Technologies:** Ethernet, Wi-Fi (802.11), ARP (Address Resolution Protocol).
- **Security relevance:**
  - ARP spoofing (poisoning) attacks to intercept traffic.
  - MAC flooding to overwhelm switches.
- **SOC Tip:** Watch for unusual MAC address changes or ARP cache poisoning alerts.

Layer 1: Physical Layer

- **Role:** The physical transmission of raw bits over cables, fiber optics, or wireless.
- **Devices:** Cables, switches, hubs, repeaters, network interface cards (NICs).
- **Security relevance:**
  - Physical tampering or cable tapping.
  - Jamming in wireless networks.
- **SOC Tip:** Though physical security is often handled by other teams, SOC should be aware of indicators of physical compromise.

Why OSI Model Matters for a SOC Analyst

- **Attack Detection:** Knowing which layer an attack targets helps focus investigation (e.g., DDoS on Layer 3, SQL injection on Layer 7).
- **Tool Application:** Different security tools work on different layers (firewalls at Layer 3/4, WAF at Layer 7, EDR on endpoints).
- **Incident Response:** Understanding how data flows across layers helps in tracing attack paths and identifying affected systems.

---

6. EDR (Endpoint Detection and Response)

What is it?

Security software installed on endpoints (laptops, servers) to detect suspicious behavior and respond to threats.

Key Features:

- **Real-time monitoring**
- **Threat detection using behavior analytics**
- **Automatic response (isolation, blocking)**
- **Forensics (what happened and how)**

➡️ Example Tools: CrowdStrike, SentinelOne, Microsoft Defender for Endpoint

➡️ SOC Role: You'll often investigate alerts coming from EDR tools.

---

7. FW (Firewall)

What is a Firewall?

A network security device that **monitors and controls** incoming and outgoing traffic based on rules.

Types:

- **Network Firewall:** Blocks/allows traffic by IPs, ports, protocols.
- **Host Firewall:** Software installed on the endpoint (e.g., Windows Defender Firewall).
- **Next-Gen Firewall (NGFW):** Includes IDS/IPS, DPI (deep packet inspection), app control.

➡️ Example Rule: Allow port 443 outbound, deny port 23 (Telnet) inbound.

Stateful Firewall (Thinks like a security guard who remembers people)

- It **remembers active connections** (like conversations).
- It knows which traffic is part of a **legit ongoing connection**.
- Makes decisions based on the **full context** of a session.

✅ **Example:**

You open a website → your browser sends a request → the site sends data back.
The **stateful firewall** remembers:

"This incoming data is part of a connection the user started. Let it through."

It **tracks the full conversation** (request and response).

✅ **Advantages:**

- Smarter and safer
- Blocks unexpected responses from unknown sources
- Useful for protocols like **TCP** that require connection tracking

Stateless Firewall (Thinks like a bouncer who checks every person each time)

- It treats **every packet as new and unrelated**.
- It doesn't remember previous connections.
- Decisions are made **only on the current packet's info** (like IP, port, protocol).

❌ **Example:**

You open a website → browser sends a request → site sends a response.
The **stateless firewall** sees the response and thinks:

"Hmm, this is random traffic I've never seen. Drop it."

Even though it's part of an ongoing session, it doesn't know that — it doesn't keep track.

✅ **Advantages:**

- **Faster** and **simpler**
- Uses less memory
- Still useful for simple or small networks

🧠 **Summary Table**

Feature	**Stateful Firewall**	**Stateless Firewall**
Tracks connections	✅ Yes	❌ No
Decision-making	Based on **context/history**	Based on **single packet**
Accuracy	✅ Higher	⚠️ Lower
Performance	🐢 Slightly slower (more memory)	⚡ Faster (less memory)

Feature	Stateful Firewall	Stateless Firewall
Security level	🔒 Stronger	🔓 Basic

1. Hardware Firewalls (Network-Based)

These are physical devices placed between your internal network and the internet. They protect entire networks.

🔹 **Examples:**

Product	Manufacturer	Use Case
Cisco ASA	Cisco	Large enterprise networks
FortiGate	Fortinet	Medium to large businesses
Palo Alto NGFW	Palo Alto Networks	Advanced traffic inspection with App-ID
Sophos XG Firewall	Sophos	Unified Threat Management (UTM)
Check Point Firewall	Check Point	Enterprises with strong security needs

---

Software Firewalls (Host-Based)

These are installed on individual computers (endpoints) to monitor and filter incoming/outgoing traffic.

🔹 **Examples:**

Product	Platform	Description
Windows Defender Firewall	Windows	Built-in, good for personal use or small networks

Product	Platform	Description
pfSense	FreeBSD (open source)	Can run on your own hardware; great for custom setups
IPTables / nftables	Linux	Command-line firewall for Linux systems
Comodo Firewall	Windows	Free firewall with advanced settings
ZoneAlarm	Windows	Personal firewall for home users

Cloud-Based & Next-Generation Firewalls (NGFW)

These provide more advanced features like application awareness, deep packet inspection, and integration with cloud services.

🔹 **Examples:**

Product	Provider	Features
Azure Firewall	Microsoft	Protects resources in Microsoft Azure
AWS Network Firewall	Amazon	Cloud-native firewall in AWS
Cloudflare Gateway	Cloudflare	DNS and HTTP traffic filtering
Palo Alto Prisma Access	Palo Alto Networks	Cloud-based NGFW with global reach

---

8. Vulnerabilities and Exploits

Definitions:

- **Vulnerability:** A weakness or flaw in software/hardware that can be exploited.
- **Exploit:** The method/technique used by an attacker to take advantage of a vulnerability.

Example:

- **Vuln:** Outdated Apache server version.
- **Exploit:** Attacker sends crafted request to trigger remote code execution.

➡️ SOC Role: Monitor for exploit attempts (e.g., known CVEs), investigate alerts.

🔓 What Is a Vulnerability? (More Detailed Explanation)

A **vulnerability** is a **weakness** or **flaw** in a system that can be taken advantage of.

Think of it like:

🔑 A door with a broken lock. It's not *yet* broken into, but it's **easy to attack**.

✅ Examples of vulnerabilities:

- An outdated software version with known bugs
- A web form that doesn't sanitize input
- A file that gives **everyone** read/write access (misconfiguration)
- A default password still in use (admin:admin)
- A buffer overflow bug in a program

💥 What Is an Exploit?

An **exploit** is the **actual method or tool** that an attacker uses to take advantage of the vulnerability.

Think of it like:

🧰 A **tool or trick** the attacker uses to actually **open that broken lock** and walk through the door.

✅ Examples of exploits:

- Sending specially crafted input to **crash** a service and gain control (buffer overflow)
- Running a script that **uploads a web shell** via insecure file upload
- Using **Metasploit** to exploit a known Windows vulnerability (like EternalBlue)
- Triggering an **XSS script** that steals session cookies

🔁 Relationship:

Term	Explanation	Analogy
Vulnerability	A weakness in the system	A broken lock
Exploit	A way to take advantage of that weakness	Using a hairpin to open the broken lock

---

📌 Real-World Example:

- **Vulnerability**: A web server is running **PHPMyAdmin** with no password.
- **Exploit**: An attacker simply connects to it and **drops the database**.

🔐 Why It Matters in a SOC Role:

As a **SOC analyst**, your job is to:

1. **Monitor logs** and alerts for signs someone is trying to **exploit** a vulnerability.
2. Work with vulnerability management tools to **track unpatched systems**.
3. Raise tickets for teams to **patch** or **mitigate** weaknesses.

🧠 Bonus:

Not every vulnerability gets exploited. But **if it's not fixed**, it's **only a matter of time**.

---

9. Mail Relay

What is a Mail Relay?

A mail server that **forwards email** from one server to another.

Security Concern:

- If not properly configured, it can be used for **spamming** (open relay).

Good vs Bad:

- ✅ **Good Relay:** Authenticated, controlled access.
- ❌ **Open Relay:** Anyone can send emails through it — abused by spammers.

➡️ SOC Role: Detect and alert if your organization is an open relay.

(Longer Explanation) What is a Mail Relay?

A **Mail Relay** is a **mail server** that **forwards email messages** from one server to another — like a **middleman** that helps deliver emails.

💡 Simple Analogy:

Imagine you're sending a physical letter. You give it to **Post Office A**. That post office doesn't deliver to your friend's city, so it passes it to **Post Office B**, which delivers it to the final destination.

In email terms:

- You → send an email
- Your mail server → **relays** it
- It eventually reaches the receiver's mailbox

That **middle step** is **mail relaying**.

📬 Why Do We Use Mail Relays?

- **Load balancing**: Some companies use dedicated servers just for relaying emails.
- **Security filtering**: Relays can scan for spam or malware before delivering.
- **Backup**: If the main mail server is down, the relay can queue messages.

🚨 What Is an Open Mail Relay?

An **open mail relay** is a mail server that:

- Allows **anyone on the internet** to send email through it
- Doesn't check if the sender is allowed to use it

✅ Useful 20+ years ago
❌ Dangerous today

❗ Why is this bad?

Because **spammers and hackers** can:

- Use the open mail relay to send huge amounts of **spam**, **phishing**, or **malware**
- Hide their identity — your server does the dirty work

So now:

- Your IP address might get **blacklisted**
- Your organization's email reputation is damaged
- You could be used in a **mass spam or phishing campaign**

👮 What to Look For as a SOC Analyst:

When investigating mail logs or alerts, look out for:

- **High volumes** of outbound mail from internal servers
- **Unexpected destinations** (e.g. Russia, China, random free email services)
- Repeated **SMTP traffic** from unknown IPs
- Use of **unauthorized relays** (especially on port **25**)

🔧 How to Prevent Abuse:

- Never leave mail servers as **open relays**
- Use **SMTP authentication** (only authorized users can send)
- Restrict relaying by:
  - IP range
  - Domain
  - User credentials

- Monitor logs with your **SIEM** for abnormal mail activity

🧠 **Summary:**

Term	**Meaning**
Mail Relay	Forwards emails from one server to another
Open Mail Relay	A relay that accepts mail from anyone (BAD)
Risk	Spammers use it to send phishing/malware/spam
SOC Role	Monitor, detect abuse, escalate, and block abuse

🟦 10. NAC (Network Access Control)

What is NAC?

A security solution that controls **who and what** can access the network.

How it works:

- Devices must meet policies (antivirus on, OS updated).
- Can **quarantine**, **deny**, or **allow** based on compliance.

➡️ Example: A laptop without antivirus is blocked from internal servers until fixed.

➡️ SOC Role: You may receive alerts when unauthorized devices attempt to connect.

🟦 11. System Hardening

What is it?

The process of **reducing vulnerabilities** by securing system configurations.

Includes:

- Disabling unused services
- Patching systems regularly
- Changing default passwords
- Removing unnecessary software
- Enforcing strong password policies

➡️ SOC Role: Monitor for deviations from baseline (e.g., new service running on endpoint).

---

12. Web Attacks and Responses

Common Web Attacks:

Attack	**Description**	**Response**
SQL Injection	Access/modify database via input	Input validation, parameterized queries
XSS	Malicious scripts on web pages	Output encoding, CSP headers
CSRF	Forging user actions	Anti-CSRF tokens
File Inclusion	Loading malicious server files	Validate user inputs, restrict access
Directory Traversal	Accessing sensitive directories	Sanitize file paths

SOC Response:

- Review logs (WAF, web server)
- Alert web admin
- Correlate IPs or patterns with threat intel
- Work with developers to fix code vulnerabilities
- Report the incident and apply WAF rules if needed

🌐 What Are Web Attacks?

Web attacks are attempts by hackers to **exploit vulnerabilities in websites or web applications**.

These attacks usually target:

- **User input forms** (like login or search boxes)
- **Cookies / sessions**
- **Back-end databases**
- **Authentication systems**
- **Web servers**

Why? Because the web is **publicly exposed**, and if attackers find a weakness, they can:

- Steal user data (like usernames, passwords, credit cards)
- Gain admin access
- Deface the site
- Install malware

🔥 Common Web Attacks

Here are the **most common types**, explained simply:

---

1. XSS (Cross-Site Scripting)

Attacker runs malicious JavaScript in the browser of other users.

- **Stored XSS**: The malicious script is **saved on the server** (like in a comment section) and shows to every user who loads that page.
- **Reflected XSS**: The malicious script is sent in a **link or URL**, and runs only if the victim clicks the link.

What can it do?

- Steal login cookies
- Redirect users to fake websites
- Modify how the page looks or behaves

🛡️ **Defense**:

- Input validation
- Output encoding
- Use security headers like Content-Security-Policy

---

2. SQL Injection

Attacker inserts malicious SQL commands into a form field or URL.

- They can read or change data in the **backend database**.
- Can bypass logins or even delete the entire database.

Example: Typing ' OR 1=1 -- into a login box tricks the database into letting the attacker in.

🛡️ **Defense**:

- Use **parameterized queries**
- Sanitize all user input
- Restrict database permissions

---

3. Command Injection

Attacker tries to execute **system-level commands** through the website.

Example: A form lets users ping a server. If not protected, an attacker might type:

; rm -rf /

🛡️ **Defense**:

- Never trust user input
- Use safe functions in backend code
- Limit system access rights

---

4. Directory Traversal

Accessing files outside the allowed folder by using ../ sequences.

Example:
http://example.com/download?file=../../etc/passwd

🛡️ **Defense**:

- Sanitize paths
- Restrict access to allowed directories only
- Use web server configurations to block this behavior

---

5. CSRF (Cross-Site Request Forgery)

Tricks a logged-in user into doing something they didn't intend.

Example:

- You're logged into your bank in one tab.
- You click a malicious link in another tab.
- That link secretly transfers money using your credentials.

🛡️ **Defense**:

- Use anti-CSRF tokens
- Require re-authentication for sensitive actions

---

6. File Upload Attacks

A user uploads a file that looks innocent (e.g. .jpg) but actually contains malicious code (.php or .exe).

🛡️ **Defense**:

- Validate file types and sizes
- Rename uploaded files
- Store uploads outside the web root

🧑‍💻 How Does a SOC Analyst Respond?

🔍 Detection:

- Monitor logs using a **SIEM** (e.g. suspicious URLs, repeated SQL-like strings)
- Use **WAF logs** for blocked/allowed attacks
- Get alerts for strange user behavior (e.g. lots of failed logins, login at weird hours)

⚠️ When an Attack is Detected:

Step	Action
1️⃣	**Alert** triggered (e.g., from SIEM or WAF)
2️⃣	SOC analyst **investigates**: IP address, payload, affected URLs
3️⃣	Check if attacker succeeded or just attempted
4️⃣	**Block attacker's IP** or **user account**
5️⃣	Notify devs or incident response team
6️⃣	Write a **report** and document the indicators of compromise (IOCs)

🛡️ **Prevention & Mitigation:**

- Use a **WAF (Web Application Firewall)**
  → Blocks known attack patterns (e.g. XSS, SQLi)
- Enforce **input validation**
- Keep web software **patched and up to date**
- Perform **regular vulnerability scans** or **penetration testing**
- Use **HTTPS** to encrypt data
- Implement **secure coding practices**

✅ **Summary Table:**

Attack Type	**What It Does**	**Example**	**Defense**
XSS	Injects JS into web pages	<script>stealCookies()</script>	Sanitize input, use CSP
SQL Injection	Access/modifies database	' OR 1=1 --	Param queries, restrict DB rights
Command Injection	Runs system commands	; rm -rf /	Sanitize input, limit shell access
CSRF	Tricks logged-in users to act	Fake "click here" transfers money	Use CSRF tokens

Attack Type	What It Does	Example	Defense
File Upload	Uploads malware disguised as images	shell.php	Restrict file types & names
Directory Traversal	Accesses sensitive files	../../etc/passwd	Sanitize path, restrict folders

🧼 What Does Sanitization Mean?

Sanitization means **cleaning** or **modifying input** so that it can't be used maliciously.

Think of it like this:
You're letting a stranger send you a text message — but you remove any dangerous parts before saving it or using it in your system.

🔍 **Example**:
A user submits this:

<script>alert('hacked')</script>

You **sanitize** it so it becomes:

&lt;script&gt;alert('hacked')&lt;/script&gt;

Now it's just harmless text — not executable code.

✅ What Is Input Validation?

Input validation means **checking** if the user's input is acceptable before using it.

You're asking questions like:

- Is this text the right **type** (text, number, email)?
- Is it the right **length**?
- Does it match a specific **format**?

🔍 **Example**:

- If a user is supposed to enter their age → make sure it's a number between 0 and 120.
- If they're uploading a file → make sure it's a .jpg, not a .php.

🛡️ Purpose: To **reject** anything that doesn't fit what you're expecting.

🧹 What Does "Sanitize All User Input" Mean?

This means:

Any data coming from users (like forms, URLs, uploads, search bars, cookies) should be **cleaned or filtered** so it can't be used in an attack.

🧑‍💻 Example:
User enters this into a search box:

'; DROP TABLE users --

If you don't sanitize or validate it, this could wipe your database.
Instead, treat it as **plain harmless text**, not code.

📦 What Are Parameterized Queries?

Parameterized queries (aka **prepared statements**) are a way to **safely use user input in a database query**, without letting it become part of the SQL command.

Without parameters (❌ **vulnerable to SQL Injection**):

SELECT * FROM users WHERE username = '" + userInput + "';

If userInput = ' OR 1=1 --, this becomes:

SELECT * FROM users WHERE username = '' OR 1=1 --';

→ attacker gets access.

With **parameterized query** (✅ safe):

cursor.execute("SELECT * FROM users WHERE username = ?", (userInput,))

Now the input is treated as **data only**, not as part of the SQL code.

📁 What Does "Sanitize Paths" Mean?

If your app uses **file paths based on user input**, you must **remove dangerous parts** like ../.

🔍 **Example**:
User inputs:

../../etc/passwd

Without sanitizing, the system might give them access to sensitive system files.

Sanitizing the path means:

- Removing .. and / from user input
- Only allowing files inside a specific folder

🔑 What Are Anti-CSRF Tokens?

A **CSRF token** is a **secret random string** added to every form or request that changes data.

It proves that the user **intended** to perform the action and not some attacker tricked them.

🧑‍💻 Example:

- When submitting a form to transfer money, the browser includes a hidden field like:

<input type="hidden" name="csrf_token" value="ABC123XYZ">

When the server sees the request, it checks:

- Is this token correct?
- Did it come from a real form on the site?

If not — the server **rejects** the request.

🛡️ CSRF tokens **prevent malicious websites from acting on behalf of logged-in users**.

🧠 Summary

Term	**Meaning**
Sanitization	Clean input by removing/replacing dangerous parts (like <script>)
Input Validation	Check if input is correct type, format, length before using it
Sanitize All Input	Never trust user data; clean/filter it before using it anywhere
Parameterized Queries	Safe way to add user input into SQL queries without risk of injection
Sanitize Paths	Remove things like ../ from file paths to block access to sensitive files
Anti-CSRF Tokens	Secret keys to prove a request is coming from the real user/form

---

🔍 IDS – Intrusion Detection System

- **What it does:**
  Detects suspicious activity or known attack patterns.
- **Action:**
  Alerts you, but does **not stop** the attack by itself.
- **Example:**
  Like a **security camera** — it sees something suspicious and notifies you.

🛡 IPS – Intrusion Prevention System

- **What it does:**
  Detects and blocks suspicious or malicious activity.
- **Action:**
  Automatically takes action — such as dropping a packet or blocking traffic.
- **Example:**
  Like a **bodyguard** — sees something suspicious and immediately **intervenes**.

🔄 Key Differences:

Feature	**IDS**	**IPS**
Action	Detects & alerts only	Detects **and blocks**
Placement	Usually **out-of-band**	**In-line** with traffic
Risk of False Positives	Low risk (doesn't block)	Can **accidentally block legit traffic**

🔄 False Positive vs. False Negative

These are **two types of detection errors** in cybersecurity systems like IDS, IPS, antivirus, etc.

❌ False Positive

The system thinks something is bad, but it's actually good.

- **Example:**
  You get an alert for an attack, but it turns out to be a harmless software update.
- **Real-life analogy:**
  A **fire alarm** goes off, but there's **no fire** — just smoke from cooking.
- **Impact:**
  Wastes time, may block legitimate traffic.

❌ False Negative

The system thinks something is good, but it's actually bad.

- **Example:**
  A real malware file sneaks in, but your antivirus **doesn't catch it**.
- **Real-life analogy:**
  A **fire** breaks out, but the fire alarm **doesn't go off**.
- **Impact:**
  Very dangerous — the attack goes unnoticed.

🧠 **Summary Table:**

Term	What it means	Danger Level
False Positive	Alerts on something harmless	Annoying
False Negative	Misses something actually malicious	**Very risky**

How Does an Attacker Keep Persistence?

Persistence means the attacker stays connected to a system even after reboots, logouts, or attempts to remove them.

Common Persistence Techniques:

- **Creating startup entries:**
  Adding malicious programs to start automatically when the system boots (e.g., in Windows Registry or startup folders).
- **Installing backdoors or services:**
  Setting up hidden services or processes that allow remote access anytime.
- **Scheduled tasks or cron jobs:**
  Running malicious scripts or programs on a schedule to maintain access.
- **Modifying legitimate programs:**
  Injecting code into trusted applications that run regularly.
- **Using valid accounts:**
  Creating or hijacking user accounts with access permissions.
- **DLL hijacking or planting:**
  Placing malicious DLLs that get loaded by legitimate apps.
- **Persistence via firmware or hardware:**
  Harder to detect, involves implanting malware in BIOS or peripheral devices.

What is Mimikatz?

- **Mimikatz** is a powerful **post-exploitation tool** used to extract **passwords, hashes, PINs, and Kerberos tickets** from memory in Windows systems.
- Created by **Benjamin Delpy** for research, but often used by attackers for **privilege escalation and lateral movement**.

Key Features:

- Dump **plaintext passwords** from memory.
- Extract **NTLM hashes** and **Kerberos tickets**.
- Perform **Pass-the-Hash** and **Pass-the-Ticket** attacks.
- Exploit **vulnerabilities in Windows authentication**.

Common Use Cases:

- **Red teamers** and **pentesters** use it to simulate attacks.
- **Attackers** use it after gaining access to escalate privileges or move across the network.

⚠️ **Note:** Mimikatz is often flagged by antivirus/EDR because it's widely used by hackers.

✅ **Final Tip: What Interviewers Want to See**

- You **understand basic technical concepts** and can explain them clearly.
- You **can analyze logs** and identify signs of attacks.
- You show **awareness of real threats** and how tools like firewalls, WAF, and EDR help defend against them.