WAS U5

21. Five Common Threats to Web Applications:

1. **SQL Injection (SQLi):**
   Attackers insert malicious SQL queries into input fields to access, modify, or delete database data.
2. **Cross-Site Scripting (XSS):**
   Malicious scripts are injected into web pages, which then execute in users' browsers, leading to data theft or session hijacking.
3. **Cross-Site Request Forgery (CSRF):**
   Forces an authenticated user to perform unintended actions on a web application without their knowledge.
4. **Broken Authentication:**
   Weak or improperly implemented login systems allow attackers to bypass authentication and gain unauthorized access.
5. **Security Misconfiguration:**
   Poorly configured servers, APIs, or applications expose sensitive data or create backdoors for attackers.
6. **Insecure Direct Object References (IDOR):**
   Attackers manipulate references (like IDs or URLs) to access unauthorized data or functions within the application.
7. **Sensitive Data Exposure:**
   Failure to properly protect sensitive information such as credit card numbers, passwords, or personal data, often due to weak encryption or poor security practices.

22. Web Application Firewall (WAF) – Explained in Detail

A **Web Application Firewall (WAF)** is a security tool that protects web applications by monitoring and filtering HTTP/HTTPS traffic between the internet and the web application.
It is specifically designed to detect and prevent attacks that target application vulnerabilities, which traditional network firewalls may not catch.

- **How WAF Works:**

- A WAF acts as a **shield** between the web application and the user.
- It **analyzes incoming requests** (from clients) and **outgoing responses** (from servers).
- Based on pre-defined **rules or policies**, it blocks, allows, or challenges the traffic.
- These rules can detect malicious patterns such as:
    - SQL Injection
    - Cross-Site Scripting (XSS)
    - File Inclusion
    - Cross-Site Request Forgery (CSRF)

✅ Two Important Functions of WAF:

1. Protection from Common Web Attacks

- WAFs automatically detect and block attacks like:
    - **SQL Injection**: Attackers try to run malicious SQL queries in the backend.
    - **Cross-Site Scripting (XSS)**: Attackers inject harmful scripts in webpages.
- Ye attacks data leak, website crash, ya unauthorized access cause kar sakte hain.
- WAF in sabko real-time me detect kar ke block kar deta hai.

2. Traffic Monitoring and Filtering:

- WAF har incoming request ko analyze karta hai based on **predefined security rules**.
- Agar koi suspicious pattern milta hai (like bots, unusual IPs, or DDoS attempts), toh WAF us traffic ko block ya redirect kar deta hai.
- Ye help karta hai to maintain **application uptime and availability** during attack scenarios.

🔹 Types of WAFs:

1. **Network-based WAF** – Installed on hardware appliances; offers low latency.
2. **Host-based WAF** – Installed on the same server as the application.
3. **Cloud-based WAF** – Offered as a service by cloud providers like AWS, Cloudflare, etc.

🔹 Modes of Operation:

1. **Detection Mode (Monitoring):**
    - WAF only logs suspicious activity without blocking it.

    - Used to understand traffic patterns before applying protection.
2. **Prevention Mode (Active):**
    - WAF actively blocks or filters out malicious traffic in real-time.

🔹 Examples of WAF Providers:

- AWS WAF
- Cloudflare WAF
- Azure Application Gateway WAF

23,30. Web Security Testing

- Web Security Testing is the process of evaluating a web application or website to identify security vulnerabilities, weaknesses, and misconfigurations that attackers could exploit.
- It ensures that the application's confidentiality, integrity, and availability requirements are met and that sensitive data (e.g., user credentials, payment details) remains protected.
- Web security testing is an essential, multi-layered practice that employs both automated tools and manual expertise to uncover and fix vulnerabilities.
- By integrating it throughout the development lifecycle (DevSecOps), organizations can reduce risk, maintain regulatory compliance, and protect both their systems and their users.

1. Objectives of Web Security Testing

- **Identify Vulnerabilities:** Discover coding flaws or configuration errors (e.g., SQL injection, cross-site scripting).
- **Assess Risk Impact:** Evaluate how serious each vulnerability is—could it leak data, allow unauthorized access, or enable defacement?
- **Ensure Compliance:** Verify that the application meets industry standards (e.g., OWASP Top 10, PCI-DSS for payment processing).
- **Enable Remediation:** Provide clear guidance and evidence (logs, proofs of concept) so developers can fix issues effectively.

2. Types of Web Security Tests

1. **Static Application Security Testing (SAST)**

- Examines source code or binaries without running the application.
- Finds issues like buffer overflows, insecure function calls, hard-coded credentials.

2. **Dynamic Application Security Testing (DAST)**
   - Tests the running web application by sending crafted HTTP requests.
   - Detects runtime problems such as broken authentication, XSS, SQL injection.
3. **Interactive Application Security Testing (IAST)**
   - Combines elements of SAST and DAST by instrumenting the application during execution.
   - Provides deeper insights into vulnerabilities with fewer false positives.
4. **Penetration Testing ("Pen Test")**
   - Ethical hackers simulate real attacks to exploit discovered vulnerabilities.
   - Often includes social engineering, business-logic abuse, and chained exploits.
5. **Configuration and Deployment Testing**
   - Verifies secure server settings (TLS versions, HTTP headers, cookies flags).
   - Checks for unnecessary services, default credentials, directory listings.

3. Importance or why neccessay before deployment

🔐 1. To Prevent Cyber Attacks

Before deployment, the application must be tested for common vulnerabilities like **SQL Injection, XSS, CSRF**, etc. If these are not fixed, attackers can steal data, modify content, or take control of the server.

🛡️ 2. To Protect User Data

Web applications often handle **sensitive user information** such as usernames, passwords, emails, and payment details. Web security ensures this data is **encrypted and secure**, maintaining **confidentiality and trust**.

✅ 3. To Ensure Compliance

Many industries have legal and regulatory standards (like **GDPR, PCI-DSS**) that require secure handling of data. Security testing helps meet these standards and **avoids legal penalties**.

🔍 4. To Identify and Fix Vulnerabilities Early

Security testing before deployment helps in detecting **security flaws early**, when fixing them is **cheaper and easier**. Fixing bugs after deployment may lead to **downtime** or user impact.

🔒 5. To Maintain Application Integrity

Security ensures that the application behaves as expected and is not **modified or misused by unauthorized users**. This helps maintain the integrity of the web application's functionality.

⚠️ 6. To Avoid Financial and Reputation Loss

Security breaches can lead to **loss of customer trust**, legal actions, and **financial losses** due to fraud or ransomware. Ensuring security before deployment reduces this risk significantly.

7. Ensures Business Continuity:

A secure website remains **online and functional**, avoiding downtime caused by hacking or malware.

8. Maintains User Trust:

When users feel that their data is safe, it builds **trust and credibility** in the application or business.

24. Describe the difference between authentication and authorization in web apps.

Point	Authentication	Authorization
1. **Definition**	Verifies the identity of a user trying to access the system.	Determines the access level or permissions a user has.
2. **Purpose**	To confirm that the user is who they claim to be.	To control what resources or actions the user can perform.
3. **Process**	User provides credentials like username and password.	System checks user roles or privileges after login.
4. **When it Occurs**	Happens **before** any access to application resources.	Happens **after** successful authentication.
5. **Example**	Logging in by entering username and password.	Accessing admin panel allowed only for admin users.
6. **Focus**	Identity verification.	Access control and permission enforcement.
7. **Involves**	Credentials verification like passwords, biometrics, OTP.	Role-based or policy-based permission checks.
8. **Security Goal**	Ensures only legitimate users can enter the system.	Ensures users can only access resources they are allowed to.

---

25. Penetration Testing (Pen Testing) :

Penetration Testing is a **controlled and authorized simulated cyber attack** on a computer system, network, or web application.

The goal is to identify and exploit security vulnerabilities before malicious hackers can use them.

Key Points:

- **Purpose:** To find security weaknesses and assess how an attacker could gain unauthorized access or cause damage.
- **Process:** Ethical hackers (penetration testers) use the same tools and techniques as attackers to test the system's defenses.
- **Types:** Can be **black-box** (tester has no prior knowledge), **white-box** (tester has full knowledge), or **gray-box** (partial knowledge).
- **Outcome:** Provides detailed reports on vulnerabilities found, risk level, and recommendations to fix the issues.
- **Importance:** Helps organizations strengthen security, prevent data breaches, and comply with security standards.

differentiate Penetration Testing and Vulnerability Scanning:

1. **Purpose**
   Penetration Testing simulates real attacks to evaluate risks, while Vulnerability Scanning only detects potential vulnerabilities without exploiting them.
2. **Methodology**
   Penetration Testing is a manual, expert-driven process, whereas Vulnerability Scanning is mostly automated using tools.
3. **Exploitation**
   Penetration Testing actively exploits vulnerabilities to see their impact, but Vulnerability Scanning only identifies weaknesses without exploitation.
4. **Reporting**
   Penetration Testing provides detailed reports including risk impact and recommendations; Vulnerability Scanning lists vulnerabilities with severity levels.
5. **Human Involvement**
   Penetration Testing requires skilled ethical hackers; Vulnerability Scanning requires minimal human effort.
6. **Time and Cost**
   Penetration Testing is time-consuming and costly due to manual work; Vulnerability Scanning is faster and less expensive.

7. **Frequency**

   Penetration Testing is usually performed less frequently due to complexity; Vulnerability Scanning can be done regularly for ongoing monitoring.

8. **Focus Area**

   Penetration Testing focuses on real-world attack scenarios; Vulnerability Scanning focuses on identifying as many potential issues as possible.

26. Differentiate between a traditional firewall and a Web Application Firewall (WAF).

Aspect	Traditional Firewall	Web Application Firewall (WAF)
OSI Layer	Operates mainly at **Layer 3 (Network)** and **Layer 4 (Transport)**	Operates at **Layer 7 (Application Layer)**
Purpose	Controls traffic based on IP addresses, ports, and protocols	Filters and monitors HTTP/HTTPS traffic to web applications
Protection Scope	Protects networks and systems from unauthorized access	Protects **web applications** from attacks like SQL injection, XSS
Types of Attacks Blocked	Blocks DDoS, port scanning, malware, and unauthorized access	Blocks **application-layer attacks** like XSS, CSRF, SQLi
Traffic Filtering	Uses packet filtering, stateful inspection, and access rules	Uses **rule sets** to analyze and block malicious web requests
Visibility	Cannot understand web content or application behavior	Has deep visibility into **web requests and responses**
Deployment Location	Deployed at the **network perimeter** or gateway	Deployed between **user and web server (reverse proxy)**
Examples	Cisco ASA, Palo Alto Firewall	Cloudflare WAF, AWS WAF, Imperva WAF

27. Five Key Features or Areas to Be Reviewed During a Web Application Security Test:

1. **Authentication and Authorization:**

   Ensure that only valid users can log in, and users can only access data or actions permitted to them. Test for weak passwords, broken login mechanisms, and role-based access control.

2. **Input Validation:**

   Check whether the application properly validates user input to prevent **SQL Injection**, **Cross-Site Scripting (XSS)**, and other injection attacks.

3. **Session Management:**

   Verify that sessions are securely handled using techniques like **secure cookies**, **session timeouts**, and **unique session IDs** to prevent session hijacking.

4. **Data Encryption:**

   Ensure sensitive data is encrypted during transmission (using **HTTPS/SSL**) and at rest. Test for proper certificate installation and secure handling of encryption keys.

5. **Error Handling and Logging:**

   Check that the application handles errors gracefully without revealing sensitive system information and that all security events are **properly logged** for auditing.

28. Role of Input Validation in Web Application Security:

Input validation is the process of verifying and sanitizing user inputs before they are processed by the web application. It plays a **critical role in protecting applications from malicious data** that can lead to security breaches.

🔒 Key Roles:

1. **Prevents Injection Attacks:**

   Input validation helps block attacks like **SQL Injection**, **Cross-Site Scripting (XSS)**, and **Command Injection** by rejecting or sanitizing harmful inputs.

2. **Ensures Data Integrity:**

   It ensures that only properly formatted and expected data is accepted, which helps maintain the **accuracy and consistency** of application data.

3. **Reduces Server Load and Errors:**

   Validating input at the client and server sides prevents **processing invalid data**, reducing crashes or unnecessary processing.

4. **Improves Application Stability and Security:**

   It stops malformed or unexpected inputs from causing **unexpected behavior or vulnerabilities** in the application logic.

29. Three network or web security solutions used to protect web applications:

1. **Web Application Firewall (WAF):**

A **Web Application Firewall** is a specialized firewall that focuses on monitoring and filtering traffic specifically at the **application layer (Layer 7)** of the OSI model. Unlike traditional firewalls that filter traffic based on IP addresses or ports, a WAF understands and inspects **HTTP/HTTPS requests and responses**.

- **Function:** It analyzes incoming web traffic to detect and block malicious payloads that aim to exploit vulnerabilities in the web application, such as **SQL Injection**, **Cross-Site Scripting (XSS)**, or **file inclusion attacks**.
- **How it works:** The WAF uses predefined security rules or machine learning to distinguish legitimate traffic from attack patterns.
- **Benefit:** It acts as a shield for web apps without modifying the underlying application code, providing protection from many common web threats.

2. **Secure Sockets Layer (SSL)/Transport Layer Security (TLS):**

SSL and its successor **TLS** are cryptographic protocols designed to provide **secure communication** over a computer network, primarily the internet.

- **Function:** They encrypt the data transmitted between the user's browser and the web server, ensuring **confidentiality and integrity** of sensitive information like login credentials, personal details, and payment data.
- **How it works:** When a user connects to a website via HTTPS, SSL/TLS encrypts the data packets so that even if intercepted, they cannot be read or altered by attackers.
- **Benefit:** Protects against eavesdropping, man-in-the-middle attacks, and data tampering, which are critical for building user trust.

3. **Intrusion Detection and Prevention Systems (IDPS):**

An **Intrusion Detection System (IDS)** monitors network or system activities for suspicious behavior or known attack signatures. An **Intrusion Prevention System (IPS)** extends IDS capabilities by actively blocking detected threats.

- **Function:** IDPS monitors traffic in real-time to detect unauthorized access attempts, malware infections, brute force attacks, or other malicious activities targeting the web server or network.

- **How it works:** Uses signature-based detection (matching known attack patterns) or anomaly-based detection (identifying unusual behavior) to spot threats.
- **Benefit:** Enables early detection and automatic prevention of attacks, reducing the risk of successful breaches and minimizing damage.

Summary:

- **WAF** protects the web app at the application level by filtering malicious HTTP requests.
- **SSL/TLS** ensures secure, encrypted communication between users and servers.
- **IDPS** monitors the network and systems to detect and prevent intrusions.