# **Application Security Vulnerabilities (CIA Breakdown)**

## **1. Confidentiality Threats (stealing secrets)**

These attacks focus on **reading or exfiltrating data**.

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

* **What:** Attacker injects SQL into queries.
* **Scenario:** Login form → user enters admin' OR '1'='1 and bypasses authentication, dumping full user database.
* **Impact:** Steals passwords, financial records.
* **Defense:** Parameterized queries, sanitization.

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

* **What:** Injecting JavaScript into web pages.
* **Scenario:** Blog comment with <script>document.cookie</script> steals session cookie of anyone viewing it.
* **Impact:** Hijacks user sessions.
* **Defense:** Output encoding, CSP, sanitize inputs.

### **Path Traversal**

* **What:** Manipulating file paths (../../) to access restricted files.
* **Scenario:** File download feature → attacker requests ../../etc/passwd.
* **Impact:** Exposes system config files.
* **Defense:** Restrict directories, sanitize file paths.

### **XML External Entity (XXE)**

* **What:** Malicious XML with external entity.
* **Scenario:** XML upload references file:///etc/shadow, server returns password file.
* **Impact:** Data leakage, SSRF.
* **Defense:** Disable external entity parsing.

## **2. Integrity Threats (tampering / unauthorized actions)**

These attacks **change or manipulate data or system behavior**.

### **Command Injection**

* **What:** App passes user input to OS commands.
* **Scenario:** Web form → user types ; rm -rf / → server executes deletion.
* **Impact:** Attacker controls host system.
* **Defense:** Whitelist input, safe APIs.

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

* **What:** Tricking user into executing unintended actions.
* **Scenario:** Logged-in banking user clicks malicious link → transfers money without consent.
* **Impact:** Unauthorized transactions.
* **Defense:** Anti-CSRF tokens, SameSite cookies.

### **Insecure Deserialization**

* **What:** Deserializing untrusted data.
* **Scenario:** Attackers craft malicious serialized object → executes code when app deserializes it.
* **Impact:** Arbitrary code execution.
* **Defense:** Validate input, avoid native deserialization.

### **Format String Vulnerabilities**

* **What:** Using user input directly in printf.
* **Scenario:** Attacker enters %x %x %x → app dumps memory contents.
* **Impact:** Memory disclosure, possible RCE.
* **Defense:** Use explicit format specifiers.

### **Privilege Escalation**

* **What:** Exploiting flaws to gain higher rights.
* **Scenario:** User with basic rights leverages bug → becomes admin.
* **Impact:** Full system compromise.
* **Defense:** Principle of least privilege, patching.

## **3. Availability Threats (taking systems down)**

These focus on **denying service or breaking functionality**.

### **Buffer Overflow**

* **What:** Writing beyond buffer capacity.
* **Scenario:** Attacker inputs oversized string → overwrites return pointer → crashes app or executes payload.
* **Impact:** DoS or full takeover.
* **Defense:** Safe languages, ASLR, bounds checking.

### **Race Conditions**

* **What:** Two processes accessing resource simultaneously in unsafe order.
* **Scenario:** Attacker triggers two withdrawals at the same time → doubles money.
* **Impact:** Corruption, unexpected behavior.
* **Defense:** Proper locking, thread safety.

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

* **What:** Flooding service with excessive requests.
* **Scenario:** Botnet sends millions of requests to a shopping site → crashes.
* **Impact:** Service downtime.
* **Defense:** Rate limiting, WAF, CDNs.

## **4. Defensive Layer**

These aren’t attacks, but **techniques to stop the above**.

* **Input Validation:** Never trust user input, sanitize and reject malicious patterns.
* **Output Encoding:** Render user input safely (e.g., <script> shows as text, not code).
* **Parameterized Queries:** Prevent SQL injection by separating code and data.
* **Rate Limiting & Throttling:** Protects against DoS/DDoS.
* **Principle of Least Privilege:** Limit damage if one part is compromised.

**Summary Mental Model:**

* **Confidentiality → Stealing secrets (SQLi, XSS, Path Traversal, XXE).**
* **Integrity → Messing with actions/data (Command Injection, CSRF, Deserialization, Format String, Priv Esc).**
* **Availability → Knocking stuff offline (Buffer Overflow, Race Condition, DoS).**