The OWASP Top 10 Explained: Web Application Security Essentials

Author: Kamakshaiah Musunuru | **Published:** 11 September 2026 | **Category:** Resource | **Read time:** 2 min read

Tags: OWASP, web security, penetration testing, vulnerabilities, application security

The OWASP Top 10 is the de-facto standard for understanding the most critical web application security risks. Every one of these applies to applications built in-house or by vendors — and each is discoverable in a professional penetration test.

1. Broken Access Control Users gain access to functions or data they should not have. Examples include manipulating URLs to view others' records or escalating privileges. Prevention: deny by default, enforce server-side role checks and validate ownership of every object.

2. Cryptographic Failures Sensitive data is exposed because of weak or missing encryption. Prevention: use modern libraries, encrypt data at rest and in transit, and never roll your own crypto.

3. Injection Untrusted input is executed as code — SQL injection is the classic case. Prevention: parameterised queries, server-side validation with allow lists, and output encoding.

4. Insecure Design Risks arise from missing controls in the design phase rather than implementation bugs. Prevention: threat modelling and security reviews early in the design process.

5. Security Misconfiguration Default accounts, verbose error pages and open ports invite attackers. Prevention: minimal configuration, automated hardening checks and a disciplined patching cadence.

6. Vulnerable and Outdated Components Using libraries with known vulnerabilities is extremely common. Prevention: dependency scanning in CI, remove unused packages and update on every security release.

7. Identification and Authentication Failures Weak credential handling compromises passwords and sessions. Prevention: strong password policies, multi-factor authentication and framework-managed sessions.

8. Software and Data Integrity Failures Unverified code or data — supply chain attacks, unsafe deserialisation. Prevention: sign and verify artifacts, pin dependencies and validate all inputs.

9. Security Logging and Monitoring Failures Breaches go unnoticed without audit trails and alerts. Prevention: log security events, alert on anomalies and protect logs from tampering.

10. Server-Side Request Forgery (SSRF) Attackers make the server fetch unintended resources, often internal services. Prevention: validate and allow-list URLs, restrict outbound traffic and block redirects.

Where to Start

Run automated scans, review the top risks against your codebase and commission a manual penetration test for a realistic view of exploitability. Codingfigs offers OWASP-aligned application security assessments, vulnerability management and remediation roadmaps — contact us to schedule an audit.