HORIZONSHIELD

Securing Your Digital Future, Today.

APPSEC SERIES · BOOK 09 OF 16

Web Application Security Guide

OWASP Top 10, API Security & Modern Attacks — 2026

Deep-dive coverage of web application vulnerabilities, modern attack techniques, and defence strategies. SQL injection, XSS, SSRF, broken authentication, API security, and supply chain attacks — with working examples and remediation code.

VULNERABILITIES	STANDARD	API STANDARD
50+	**OWASP Top 10 2021**	**OWASP API Top 10**

LEVEL

Intermediate

Contents

WEB APPLICATION SECURITY 2026

CHAPTER 01

OWASP Top 10 — 2021

The industry-standard framework for web application security risk, based on data from thousands of organisations.

A01-A10 | INJECTION | ACCESS CONTROL | CRYPTOGRAPHY

OWASP Top 10 — 2021 Complete Reference

Rank	Category	Root Cause	Primary Test	Key Prevention
A01	Broken Access Control	No enforcement of least privilege on server — relies on client-side checks	IDOR: change ?id=123 to ?id=124. Forced browsing to admin URLs without auth.	Deny by default server-side. RBAC with audit logs. Never trust client-supplied role/ID.
A02	Cryptographic Failures	Cleartext data, weak algorithms, secrets in code or logs	HTTPS enforcement check. testssl.sh for TLS version/cipher. Check DB for cleartext passwords.	TLS 1.3. AES-256-GCM. Argon2id for passwords. No MD5/SHA1. HSTS header.
A03	Injection (SQLi/XSS/CMD)	User input interpreted as commands/code without sanitisation	Single quote test ('). UNION SELECT. . ; whoami for CMDi.	Parameterised queries. Output encoding (context-aware). WAF as defence-in-depth.

Rank	Category	Root Cause	Primary Test	Key Prevention
A04	Insecure Design	Missing threat model, insufficient security requirements, no business logic controls	Business logic bypass: skip payment, negative amounts, unlimited discount codes.	Threat modelling (STRIDE). Security requirements. Business logic tests in QA.
A05	Security Misconfiguration	Default credentials, verbose errors, unnecessary features, cloud misconfiguration	Nikto/Nuclei scan. Check /.git /.env /phpinfo.php /actuator. Default credential test.	Hardening guides. Automated config scanning. Remove default accounts. Disable debug.
A06	Vulnerable Components	Known CVE-affected libraries, outdated server, unpatched CMS	Retire.js for JS libs. Snyk/OWASP DC for dependencies. WPScan for WordPress.	SCA in CI/CD pipeline. Automated dependency updates (Dependabot). SBOM generation.
A07	Auth & Session Failures	Weak passwords, missing MFA, session fixation, weak session tokens	JWT alg:none attack. Session token entropy. Brute force login.	MFA mandatory. Secure session management. Account lockout. JWT RS256 or ES256 only.
A08	Software & Data Integrity	Unsigned updates, insecure deserialization, CI/CD pipeline compromise	Java ysoserial payloads. PHP object injection. Unsigned npm packages.	Code signing. SLSA supply chain framework. Signed commits. SCA in pipeline.

Rank	Category	Root Cause	Primary Test	Key Prevention
A09	Logging & Monitoring Failures	No security event logging, no alerting, no forensic capability	Submit 10 failed logins — alerted? Inject log pollution: user=admin%0aSuccessful+Login.	Centralised logging. Alert on failed auth, access violations, app errors.
A10	Server-Side Request Forgery	Server-side URL fetch without input validation — can reach internal services	url=http://169.254.169.254/latest/meta-data/ (AWS metadata). url=http://localhost:6379.	URL allowlist not blocklist. Disable HTTP redirects. Block private IP ranges 169.254.x.x, 10.x.x.x.

A01 BROKEN ACCESS CONTROL — THE #1 RISK

Broken Access Control moved to #1 in OWASP 2021 — found in 94% of applications tested. The root cause is trusting the client to enforce access rules. Server-side: every request must be validated against the authenticated user's permissions. The user should never be able to specify their own role, user ID, or access level.

CHAPTER 03

Cross-Site Scripting (XSS)

XSS allows attackers to execute JavaScript in victims' browsers — stealing sessions, credentials, and performing actions as the victim.

REFLECTED STORED DOM-BASED CSP

XSS — Types, Testing & Remediation

XSS Type	Description	Persistence	Severity	Test Payload
Reflected XSS	Malicious script in URL parameter reflected in response — requires social engineering to deliver	Not stored — requires victim to click crafted link	HIGH — credentials/session theft via phishing link	?q=
Stored XSS	Malicious script stored in database (comments, profiles, messages) — executes for EVERY visitor	Persistent — stored in DB, executes on every page load	CRITICAL — self-propagating, affects all users without interaction	Profile field:
DOM-based XSS	Vulnerability in JavaScript — DOM manipulation uses attacker-controlled data without sanitisation	Not stored — client-side only, no server involvement	HIGH — modern SPAs heavily vulnerable	fragment: # when innerHTML used
XSS via HTTP Headers	Malicious content in User-Agent, Referer reflected in admin panel or logs	Depends on reflection context	HIGH — targets admin users specifically	User-Agent:

XSS Remediation — Context-Specific Output Encoding

```
// WRONG — direct innerHTML injection:
elem.innerHTML = userInput;  // XSS if userInput = <script>...

// CORRECT — use textContent for text:
elem.textContent = userInput;  // Safe — browser treats as text

// CORRECT — HTML encoding for HTML context:
function htmlEncode(str) {
  return str.replace(/&/g,'&amp;').replace(/</g,'&lt;')
    .replace(/>/g,'&gt;').replace(/"/g,'&quot;')
    .replace(/'/g,'&#x27;');
}

// CORRECT — JavaScript context encoding:
const safeJson = JSON.stringify(userInput);
const jsValue = `const x = ${safeJson};`;

// CORRECT — Content Security Policy (defence in depth):
// HTTP header: Content-Security-Policy:
// default-src 'self';
// script-src 'self' 'nonce-RANDOM_PER_REQUEST';
// style-src 'self' 'unsafe-inline';
// img-src 'self' data:;
// frame-ancestors 'none';
// base-uri 'self';

// CSP blocks inline scripts even if XSS payload executes
// nonce must be cryptographically random per request
```

DOM XSS IN MODERN SPAS IS CRITICALLY UNDERDETECTED

React, Vue, and Angular prevent most XSS via their virtual DOM — but dangerouslySetInnerHTML (React), v-html (Vue), and bypassSecurityTrustHtml (Angular) bypass all protections. Search your codebase for these patterns. Every use requires security review. Alternative: use DOMPurify sanitization before any HTML rendering.

CHAPTER 05

OWASP API Security Top 10

APIs are the primary attack surface in modern applications. OWASP API Security Top 10 (2023) maps the specific risks.

API1-API10 | BROKEN AUTH | EXCESSIVE DATA | MASS ASSIGNMENT

OWASP API Security Top 10 — 2023

Rank	Category	Description	Real-World Example	Prevention
API1	Broken Object Level Authorization	API returns data for any object ID — no verification caller has access	GET /api/orders/12345 — change to /api/orders/12346 and get another user's order	Server-side check: does authenticated user own this resource? Every request.
API2	Broken Authentication	Weak auth, credential stuffing, token leakage in URLs or logs	API key in URL parameter logged in server access logs	Bearer token in Authorization header. No API keys in URLs. Rate limit auth endpoints.
API3	Broken Object Property Level Authorization	API exposes all object properties including sensitive/admin fields	GET /api/users/me returns is_admin=false — PUT with is_admin=true in body → privilege escalation	Explicit allowlist of properties returned/accepted. Never auto-bind request body to model.
API4	Unrestricted Resource Consumption	No rate limiting — allows DoS, account enumeration, credential stuffing	POST /api/verify-otp with no rate limit — enumerate 10,000 6-digit OTPs	Rate limiting per IP + per user + per endpoint. Return 429 with Retry-After header.

Rank	Category	Description	Real-World Example	Prevention
API5	Broken Function Level Authorization	HTTP method or verb controls access — attackers switch from GET to PUT/DELETE	GET /api/users allowed. DELETE /api/users/123 (not documented) also works.	Explicit allow: each endpoint + each method. Never security through obscurity.
API6	Unrestricted Access to Sensitive Business Flows	No fraud/abuse detection — high-volume legitimate-looking requests	Add unlimited items to cart. Enumerate coupon codes. Mass account creation for spam.	Business logic rate limits. Fraud detection. CAPTCHA for sensitive flows.
API7	Server-Side Request Forgery	API fetches remote URL on behalf of client without validation	POST /api/webhook with url=http://169.254.169.254/metadata	Allowlist of permitted URL schemes and domains. Validate resolved IP is not private.
API8	Security Misconfiguration	Default API keys, CORS wildcard, verbose errors, debug endpoints left enabled	CORS: Access-Control-Allow-Origin: * on authenticated endpoints	Explicit CORS configuration. Remove debug endpoints. Custom error messages. TLS only.
API9	Improper Inventory Management	Shadow APIs, undocumented endpoints, old API versions still accessible	v1 API vulnerable — v2 is patched — v1 still reachable at /api/v1/	API gateway with version management. Deprecate old versions. API inventory maintained.
API10	Unsafe Consumption of APIs	Application blindly trusts third-party API data — no validation or sanitisation	Third-party API returns HTML — application renders it as HTML → stored XSS	Validate and sanitise all external API data. Apply same security rules as user input.

HS

H O R I Z O N S H I E L D

Securing Your Digital Future, Today.

Web Application Security Guide 2026

Part of the HorizonShield Security Series — 16 comprehensive professional cybersecurity manuals covering every domain of modern enterprise security.