



Web Security Checklist

The most important security measures for web applications, compact and practical.

Version 1.0 · October 2026 · switech-solution.com

This checklist covers the most common weaknesses of web applications, based on the OWASP Top 10. It is meant for developers, solo founders and small teams who want to review their application before launch or while it is running.

HOW TO USE THIS CHECKLIST

Work through the items in order and tick off what is done. Start with everything marked "Critical". Repeat the review before every major release and at least once per quarter.

Priorities: **CRITICAL** **IMPORTANT** **RECOMMENDED**

1 AUTHENTICATION & SESSIONS

- ☐ **Hash passwords securely** **CRITICAL**
Use Argon2id or bcrypt. Never store passwords in plain text and never use fast hashes like MD5 or SHA-1.
- ☐ **Sensible password rules** **IMPORTANT**
At least 12 characters, checked against known leaked passwords. No forced special-character rules and no mandatory periodic changes.
- ☐ **Two-factor authentication** **CRITICAL**
Mandatory for admin accounts, offered as an option to all other users.
- ☐ **Protect the login** **IMPORTANT**
Rate limiting and increasing delays after failed attempts. Neutral error message: "Email or password incorrect".
- ☐ **Secure session cookies** **CRITICAL**
Set HttpOnly, Secure and SameSite=Lax (or Strict). Renew the session ID after login and end sessions after inactivity.
- ☐ **Secure password resets** **IMPORTANT**
Single-use tokens with a short lifetime. Do not reveal whether an account exists for an email address.

2 ACCESS CONTROL

- ☐ **Check permissions on the server** **CRITICAL**
Check on every request, server-side. Hidden buttons in the frontend are not protection.
- ☐ **Check object ownership** **CRITICAL**
Whoever opens /invoice/123 may only see their own invoices. Always check IDs in URLs and requests against the logged-in user.

☐ **Least privilege** **IMPORTANT**
Separate roles clearly (e.g. user, admin). Everyone gets only the rights they need.

☐ **Deny by default** **IMPORTANT**
New endpoints and pages stay locked until they are deliberately opened.

3 INPUT & OUTPUT

☐ **Validate input on the server** **CRITICAL**
Check type, length and format, using allowed values (allowlists) where possible. Checks in the browser are only for convenience.

☐ **Prevent SQL injection** **CRITICAL**
Only use parameterised queries or an ORM. Never build SQL strings from user input.

☐ **Prevent cross-site scripting (XSS)** **CRITICAL**
Use the framework's automatic escaping. Never insert user data via `innerHTML` or `dangerouslySetInnerHTML` without sanitising it.

☐ **Restrict file uploads** **IMPORTANT**
Check file type and size, rename files and store them outside the web root. Never execute uploaded files.

☐ **CSRF protection** **IMPORTANT**
Protect state-changing requests (forms, payments) with SameSite cookies and CSRF tokens.

4 TRANSPORT & SECURITY HEADERS

☐ **HTTPS everywhere** **CRITICAL**
Redirect HTTP to HTTPS automatically and enable HSTS.

☐ **Content Security Policy** **IMPORTANT**
Define which sources scripts, styles and images may be loaded from. This limits the damage of XSS.

☐ **Further headers** **RECOMMENDED**
X-Content-Type-Options: nosniff, Referrer-Policy, and protection against framing (frame-ancestors or X-Frame-Options).

☐ **Restrictive CORS** **IMPORTANT**
Only allow your own domains. No "*" wildcard combined with cookies or logins.

5 DATA PROTECTION & SECRETS

☐ **Never put secrets in code** **CRITICAL**
API keys and passwords belong in environment variables or a secret manager, never in the repository or the frontend.

☐ **Data minimisation & encryption** **IMPORTANT**
Collect only the data you need. Additionally encrypt particularly sensitive data.

☐ **Backups** **IMPORTANT**
Automatic, encrypted and stored in a separate location. Test restoring them regularly.

- ☐ **No sensitive data in logs or URLs** **IMPORTANT**
Do not log passwords, tokens or personal data, and do not put them in URL parameters.

6 DEPENDENCIES & INFRASTRUCTURE

- ☐ **Keep packages up to date** **IMPORTANT**
Update regularly and check automatically (e.g. npm audit, Dependabot).
- ☐ **Clean up** **IMPORTANT**
Remove unused packages, debug endpoints, test and default accounts.
- ☐ **Error pages without details** **IMPORTANT**
Never show stack traces, paths or version numbers in production.
- ☐ **Separate environments** **RECOMMENDED**
Keep development, staging and production apart. Never test with real customer data.

7 MONITORING & INCIDENTS

- ☐ **Log security events** **IMPORTANT**
Record logins, failed attempts, permission changes and admin actions.
- ☐ **Set up alerts** **RECOMMENDED**
Get notified about unusual patterns, such as many failed logins or unusual data downloads.
- ☐ **Incident plan** **IMPORTANT**
Define who does what. Know your data breach notification duties (FDPIC in Switzerland, supervisory authority in the EU).
- ☐ **Review regularly** **RECOMMENDED**
Go through this checklist every quarter. Plan a penetration test before major launches.

This checklist does not replace an individual security assessment. It provides a solid baseline but does not cover every special case.

Questions or need support?

I help small businesses and founders build secure AI tools and automations. The intro call is free.

www.switech-solution.com