



Web-Security-Checkliste

Die wichtigsten Sicherheitsmassnahmen für Webanwendungen, kompakt und praxisnah.

Version 1.0 · Oktober 2026 · switech-solution.com

Diese Checkliste deckt die häufigsten Schwachstellen von Webanwendungen ab, angelehnt an die OWASP Top 10. Sie richtet sich an Entwickler, Solo-Gründer und kleine Teams, die ihre Anwendung vor dem Livegang oder im laufenden Betrieb prüfen wollen.

SO NUTZT DU DIESE CHECKLISTE

Geh die Punkte der Reihe nach durch und hake ab, was erledigt ist. Beginne mit allen Punkten «Kritisch». Wiederhole die Prüfung vor jedem grösseren Release und mindestens einmal pro Quartal.

Prioritäten: **KRITISCH** **WICHTIG** **EMPFOHLEN**

1 AUTHENTIFIZIERUNG & SESSIONS

- ☐ **Passwörter sicher hashen** **KRITISCH**
Argon2id oder bcrypt verwenden. Nie im Klartext speichern und keine schnellen Hashes wie MD5 oder SHA-1.
- ☐ **Sinnvolle Passwortregeln** **WICHTIG**
Mindestens 12 Zeichen, Abgleich mit bekannten geleakten Passwörtern. Keine erzwungenen Sonderzeichen-Regeln und kein regelmässiger Zwangswechsel.
- ☐ **Zwei-Faktor-Authentifizierung** **KRITISCH**
Für Admin-Konten zwingend, für alle anderen Nutzer als Option anbieten.
- ☐ **Schutz vor Login-Angriffen** **WICHTIG**
Rate Limiting und Verzögerung nach Fehlversuchen. Neutrale Fehlermeldung: «E-Mail oder Passwort falsch».
- ☐ **Sichere Session-Cookies** **KRITISCH**
Attribute HttpOnly, Secure und SameSite=Lax (oder Strict) setzen. Session-ID nach dem Login erneuern, Sessions nach Inaktivität beenden.
- ☐ **Passwort-Reset absichern** **WICHTIG**
Einmal-Token mit kurzer Gültigkeit. Nicht verraten, ob ein Konto mit dieser E-Mail existiert.

2 ZUGRIFFSKONTROLLE

- ☐ **Berechtigungen auf dem Server prüfen** **KRITISCH**
Bei jeder Anfrage serverseitig prüfen. Versteckte Buttons im Frontend sind kein Schutz.

- ☐ **Besitz von Objekten prüfen** **KRITISCH**
Wer /rechnung/123 aufruft, darf nur eigene Rechnungen sehen. IDs in URLs und Anfragen immer gegen den angemeldeten Nutzer prüfen.

- ☐ **Minimale Rechte** **WICHTIG**
Rollen klar trennen (z. B. Nutzer, Admin). Jeder bekommt nur die Rechte, die er braucht.

- ☐ **Standardmässig verweigern** **WICHTIG**
Neue Endpunkte und Seiten sind gesperrt, bis sie bewusst freigegeben werden.

3 EINGABEN & AUSGABEN

- ☐ **Eingaben serverseitig validieren** **KRITISCH**
Typ, Länge und Format prüfen, wo möglich mit erlaubten Werten (Allowlist). Prüfungen im Browser sind nur Komfort.

- ☐ **SQL-Injection verhindern** **KRITISCH**
Nur parametrisierte Abfragen oder ein ORM verwenden. Nie Nutzereingaben in SQL-Strings zusammensetzen.

- ☐ **Cross-Site-Scripting (XSS) verhindern** **KRITISCH**
Automatisches Escaping des Frameworks nutzen. Nutzerdaten nie ungeprüft per innerHTML oder dangerouslySetInnerHTML einfügen.

- ☐ **Datei-Uploads einschränken** **WICHTIG**
Dateityp und Grösse prüfen, Dateien umbenennen und ausserhalb des Web-Verzeichnisses speichern. Hochgeladene Dateien nie ausführen.

- ☐ **CSRF-Schutz** **WICHTIG**
Zustandsändernde Anfragen (Formulare, Zahlungen) mit SameSite-Cookies und CSRF-Token schützen.

4 TRANSPORT & SICHERHEITS-HEADER

- ☐ **Überall HTTPS** **KRITISCH**
HTTP automatisch auf HTTPS umleiten und HSTS aktivieren.

- ☐ **Content-Security-Policy** **WICHTIG**
Festlegen, von welchen Quellen Skripte, Styles und Bilder geladen werden dürfen. Das begrenzt den Schaden bei XSS.

- ☐ **Weitere Header** **EMPFOHLEN**
X-Content-Type-Options: nosniff, Referrer-Policy, Schutz vor Einbettung (frame-ancestors bzw. X-Frame-Options).

- ☐ **Restriktives CORS** **WICHTIG**
Nur eigene Domains erlauben. Kein Platzhalter «*» in Kombination mit Cookies oder Logins.

5 DATENSCHUTZ & GEHEIMNISSE

- ☐ **Geheimnisse nie im Code** **KRITISCH**
API-Schlüssel und Passwörter gehören in Umgebungsvariablen oder einen Secret-Manager, nie ins Repository oder ins Frontend.

- ☐ **Datensparsamkeit & Verschlüsselung** **WICHTIG**
Nur nötige Daten erheben. Besonders schützenswerte Daten zusätzlich verschlüsseln.

- ☐ **Backups** **WICHTIG**
Automatisch, verschlüsselt und an einem getrennten Ort. Die Wiederherstellung regelmässig testen.
- ☐ **Keine sensiblen Daten in Logs und URLs** **WICHTIG**
Passwörter, Tokens und Personendaten nicht loggen und nicht in URL-Parameter schreiben.

6 ABHÄNGIGKEITEN & INFRASTRUKTUR

- ☐ **Pakete aktuell halten** **WICHTIG**
Regelmässig aktualisieren und automatisch prüfen lassen (z. B. npm audit, Dependabot).
- ☐ **Aufräumen** **WICHTIG**
Ungenutzte Pakete, Debug-Endpunkte, Test- und Standardkonten entfernen.
- ☐ **Fehlerseiten ohne Details** **WICHTIG**
In Produktion keine Stacktraces, Pfade oder Versionsnummern anzeigen.
- ☐ **Umgebungen trennen** **EMPFOHLEN**
Entwicklung, Test und Produktion getrennt halten. Nie mit echten Kundendaten testen.

7 ÜBERWACHUNG & NOTFALL

- ☐ **Sicherheitsereignisse protokollieren** **WICHTIG**
Logins, Fehlversuche, Rechteänderungen und Admin-Aktionen festhalten.
- ☐ **Alarmer einrichten** **EMPFOHLEN**
Benachrichtigung bei auffälligen Mustern, z. B. vielen Fehlversuchen oder ungewöhnlichem Datenabfluss.
- ☐ **Notfallplan** **WICHTIG**
Festlegen, wer was tut. Meldepflichten bei Datenpannen kennen (EDÖB in der Schweiz, Aufsichtsbehörde in der EU).
- ☐ **Regelmässig prüfen** **EMPFOHLEN**
Diese Checkliste pro Quartal durchgehen. Vor grossen Launches einen Penetrationstest einplanen.

Diese Checkliste ersetzt keine individuelle Sicherheitsprüfung. Sie bietet eine solide Grundlage, deckt aber nicht jeden Spezialfall ab.

Fragen oder Unterstützung nötig?

Ich helfe kleinen Unternehmen und Gründern bei sicheren KI-Tools und Automatisierungen. Das Erstgespräch ist kostenlos.

www.switech-solution.com/de/