



Generelle Sicherheitsanforderungen an Webanwendungen

Merkblatt 01



IT-Sicherheitsbeauftragter UZH

Versionskontrolle

Version	Änderung	Autor	Datum
1.0	Erstellung	Sacha Schweizer	n/a
3.0	Diverse Überarbeitungen	Sacha Schweizer	n/a
4.0	Revision	Joel Schneider	20.6.2025

1. Sicherheitsgrundsätze

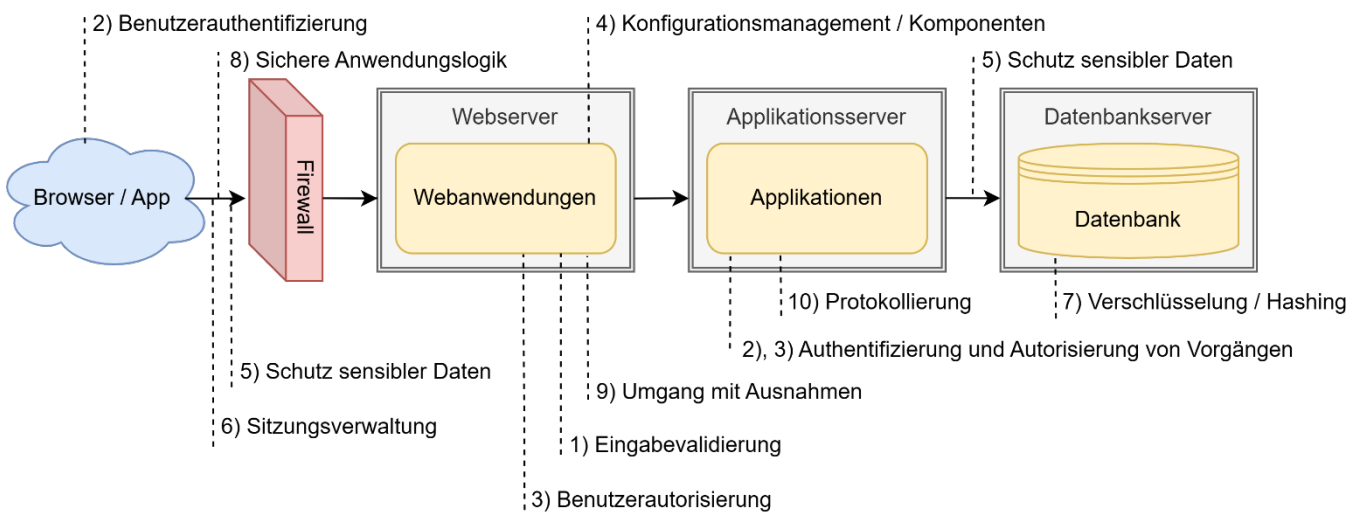
Folgende an OWASP Top 10¹ angelehnte Sicherheitsgrundsätze sollen durch die Einhaltung der Anforderungen in Kapitel 3 erreicht werden:

- Benutzereingaben müssen validiert und entschärft werden (z.B. Eingabefelder, hochgeladene Dateien, aber auch URL und HTTP-Header)
- Implementierung eines sicheren Authentifikationsverfahrens (Passwortkomplexität, Multifaktorauthentifizierung)
- Umsetzung einer robusten, rollenbasierten Zugriffskontrolle (sog. «role based access control», RBAC) mit Unterteilung in öffentliche und eingeschränkte Bereiche wo nötig
- Die verwendeten Komponenten bzw. Bibliotheken sollen aktuellen Sicherheitsstandards entsprechen und das System allgemein sicher konfiguriert sein
- Bestimmung und Schutz von sensiblen Daten
- Schutz und zeitliche Begrenzung von Benutzersitzungen
- Verschlüsselung von ruhenden Daten und Daten in Transit
- „Security by Design“: bereits in der Planungs- und Designphase soll die Cybersicherheit eine zentrale Rolle spielen. Dabei ist es wichtig, auch allfällige logische Fehler zu adressieren (Beispiel siehe unten).
- Sicherer Umgang mit Ausnahmen und Fehlermeldungen von Systemen bzw. Applikationen
- Aufbewahrung von Audit- und Systemlogs zwecks Nachvollzug von Aktivitäten und Transaktionen

2. Sicherheitsarchitektur

Überblick

Die nachfolgende Grafik zeigt auf, wo die geforderten Sicherheitsanforderungen greifen.



¹ <https://owasp.org/www-project-top-ten/>

Ausführungen

1. Eingabevalidierung	Angriffe durch Einbettung von bösartigen Strings in Abfragen, Webformularen, Cookies, URL und HTTP-Headern. Diese beinhalten auch die Ausführung von «Cross-Site Scripting» (XSS), «SQL Injection» und «Buffer Overflow»-Attacken.
2. Authentifizierung	Nachahmen von bestehenden Identitäten, «Password Cracking», Brute-Force-Angriffe, Phishing etc..
3. Autorisierung	Unberechtigter Zugriff auf Daten und Funktionen, z.B. durch «Privilege Escalation».
4. Konfigurationsmanagement und Komponenten	Ausnutzung von unsicheren Konfigurationen, offenen Ports, schwachen Berechtigungen, ungeschützten Administrations-Bereichen, unnötigen Diensten. Ebenfalls Schwachstellen in genutzten Komponenten oder Bibliotheken.
5. Sensible Daten	Offenlegung (Datenlecks) oder Manipulation von schützenswerten Informationen (z.B. Prüfungsergebnisse von Studenten, Passwörter, Personaldaten).
6. Sitzungsverwaltung	Schwache oder keine Session Identifier führen zu «Session Hijacking» und Nachahmung von Identitäten.
7. Verschlüsselung / Hashing	Schwache Verschlüsselung von ruhenden Daten oder Daten im Transit, unsicheres Schlüsselmanagement (hartcodierte Passwörter), Man-in-the-Middle.
8. Sichere Anwendungslogik	Sicherheitsüberlegungen werden nicht von Beginn an berücksichtigt. Die Systemlogik selbst ist fehlerhaft oder es fehlen Schutzmechanismen gegen absehbare Risiken. Beispiel: Ein Online-Shop erlaubt eine negative Mengenangabe im Warenkorb, was zu einer Zahlung an den Kunden führt.
9. Umgang mit Ausnahmen	Ausnutzung von detaillierten Fehlermeldungen, die einem Angreifer sensible Informationen offenlegen könnten. Kann auch zu Denial-of-Service führen, wenn Ausnahmen nicht korrekt behandelt werden.
10. Protokollierung	Fehlende oder unzureichende Protokollierung sicherheitsrelevanter Ereignisse: Angreifer bleiben unentdeckt, fehlende Nachvollziehbarkeit von Änderungen am System, erschwerte Problembehandlung, Verlust von allfälligen Beweisen.

3. Anforderungen

Nachfolgend sind die wichtigsten Anforderungen bezüglich Entwicklung von Webanwendungen enthalten (A01...10:2021). Diese stellen nur die am meisten ausgenutzten Schwachstellen dar. Generell muss ein Leistungsanbieter oder Auftragnehmer entsprechende Vorkehrungen in seinem Entwicklungsprozess gemäss ISO27001 / NIST treffen, um eine den heutigen Sicherheitsstandards genügenden Applikation entwickeln zu können.

Um eine Idee über einen potenziellen Testablauf zu erhalten, sei an diesem Punkt auf den OWASP Testing Guide verwiesen:

- https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf
- Gesamtkatalog OWASP Top 10: <https://owasp.org/Top10/>

#	Anforderung (Schutz vor...)	Beschreibung	Kategorie	Referenz (direct Link)
1.	A01 - Broken Access Control	Beschränkungen, was authentifizierte Benutzer tun dürfen, werden oft nicht richtig durchgesetzt. Angreifer können diese Schwachstellen ausnutzen, um auf nicht autorisierte Funktionen und bzw. Daten zuzugreifen oder diese zu ändern (z. B. Konten anderer Benutzer), sensible Dateien anzuzeigen, Zugriffsrechte zu ändern usw.	Webserver/ Browser / Datenbankserver	Broken Access Control.
2.	A02 – Cryptographic Failures	Viele Webanwendungen und APIs schützen sensible Daten wie Finanz-, Gesundheits- und personenbezogene Daten nicht richtig. Angreifer können solche schwach geschützten Daten stehlen oder verändern, um Kreditkartenbetrug, Identitätsdiebstahl oder andere Verbrechen durchzuführen. Sensible Daten können ohne zusätzlichen Schutz, wie z. B. zeitgemässe Verschlüsselung im Ruhezustand oder bei der Übertragung, kompromittiert werden und erfordern besondere Vorsichtsmassnahmen, wenn sie mittels Browser ausgetauscht werden.	Browser / Webanwendung	Cryptographic Failures.

#	Anforderung (Schutz vor...)	Beschreibung	Kategorie	Referenz (direct Link)
3.	A03 – Injection	Injektionsfehler, wie SQL-, NoSQL-, OS- und LDAP-Injektion, treten auf, wenn Daten als Teil eines Befehls oder einer Abfrage an einen Interpreter gesendet werden. Die injizierten Zeichenfolgen des Angreifers können den Interpreter dazu verleiten, unbeabsichtigte Befehle auszuführen oder auf Daten ohne entsprechende Berechtigung zuzugreifen.	Webanwendung / Webserver	Injection
4.	A04 – Insecure Design	Ausnutzung von grundlegenden Schwachstellen, die bereits in der Konzeptions- und Designphase entstehen. Beispiel: mangelhafte Logik eines Passwort-Reset-Mechanismus (z.B. fehlende MFA, obwohl im Standard-Anmeldeprozess vorhanden).	Alle involvierten Systeme	Insecure Design
5.	A05 – Security Misconfiguration	Schwachstellen aufgrund von Konfigurationsfehlern. Beispiel: unbeabsichtigte Verwendung von Standard-Anmeldedaten wie admin:admin.	Browser / Anwendung / Datenbankserver	Security Misconfiguration.
6.	A06 – Vulnerable and Outdated Components	Komponenten wie z.B. Bibliotheken, Frameworks oder andere Softwaremodule werden meistens mit vollen Berechtigungen ausgeführt. Wenn eine verwundbare Komponente ausgenutzt wird, kann ein solcher Angriff zu schwerwiegendem Datenverlust oder bis zu einer Serverübernahme führen. Applikationen, die Komponenten mit bekannten Schwachstellen einsetzen, können Schutzmassnahmen unterwandern und so zahlreiche Angriffe und Auswirkungen ermöglichen.	Alle involvierten Systeme	Vulnerable and Outdated Components.
7.	A07 – Identification and Authentication Failures	Anwendungsfunktionen im Zusammenhang mit der Authentifizierung und Sitzungsverwaltung sind oft fehlerhaft implementiert, so dass Angreifer Passwörter, Schlüssel oder Sitzungs-Tokens kompromittieren oder andere Implementierungsfehler ausnutzen können, um die Identität anderer Benutzer vorübergehend oder dauerhaft anzunehmen.	Browser / Webanwendung / Webserver	Identification and Authentication Failures

#	Anforderung (Schutz vor...)	Beschreibung	Kategorie	Referenz (direct Link)
8.	A08 – Software and Data Integrity Failures	Diese Kategorie beschreibt Schwachstellen, die durch fehlende Überprüfung der Integrität (also der Unversehrtheit und Echtheit) von Software oder Daten entstehen. Es geht um das blinde Vertrauen in externe Quellen, ohne deren Authentizität zu prüfen. Beispiel: Supply-Chain-Angriff – Einspielung eines [maliziösen] Updates ohne Überprüfung der Signatur / der Prüfsumme.	Webserver/ Webanwendung	Software and Data Integrity Failures.
9.	A09 – Security Logging and Monitoring Failures	Ohne ausreichende Protokollierung (Logging) und Überwachung (Monitoring) können Angriffe oft unbemerkt stattfinden, sich ausbreiten und grossen Schaden anrichten, bevor sie überhaupt entdeckt werden. Die meisten Studien zu Sicherheitsverletzungen zeigen, dass die Zeit bis zur Entdeckung einer Sicherheitsverletzung mehr als 200 Tage beträgt und in der Regel durch externe Parteien und nicht durch interne Prozesse oder Überwachung entdeckt wird.	Entdecken & Reagieren	Security Logging and Monitoring Failures
10.	A10 – Server-Side Request Forgery	Schwachstelle, bei der ein Angreifer einen verwundbaren Server dazu zwingen kann, eine von ihm kontrollierte Anfrage an eine andere, beliebige Ressource zu senden. Beispiel: Webshop überprüft Produktverfügbarkeit beim Lieferanten via URL. Angreifer manipuliert die Anfrage und versucht so, Verbindung mit interner Datenbank des Lieferanten herzustellen.	Webanwendung	Server-Side Request Forgery

4. Weitere funktionale Anforderungen und Erläuterungen

#	Anforderung	Beschreibung
1.	Rollenbasiertes Zugriffskonzept (RBAC)	Die Applikation muss ein Zugriffskonzept vorweisen, welches nach dem «need-to-know» Prinzip Zugriffe auf Funktionen / Informationen einschränken kann. Das System muss ebenfalls eine Gruppenverwaltung sowie Rollenkonzepte unterstützen, um nebst der Authentifizierung auch die Autorisierung steuern zu können.
2.	Autorisierung	Die Autorisierung auf die Applikation muss durch die Applikation erzwungen werden
3.	Authentifizierung	Die Applikation muss die Möglichkeit bieten, weitere Authentifizierungsmechanismen einzubinden (AAI) ohne die Autorisierungsvorgaben zu unterwandern.
4.	Limitation nicht erfolgreicher Authentifizierungsversuche	<ul style="list-style-type: none"> • Die Applikation unterstützt eine Limitierung der Anzahl nicht erfolgreicher Login Versuche • Die Applikation unterstützt das Sperren eines Accounts nach X fehlerhaften Login-Versuchen. <ul style="list-style-type: none"> ➔ Gesperrte Accounts können via Benutzerverwaltung freigeschaltet werden.
5.	Fehlerinformationen der Applikation	Mögliche Fehlerzustände werden in einer «Try / Catch»-Funktion abgefangen. In der Catch Funktion muss eine neutrale, aber aussagekräftige Fehlerinformation (d.h. keine sensitiven Informationen wie Passwörter) auf der Benutzeroberfläche angezeigt werden.
6.	Letzte erfolgreiche Anmeldung	Das System zeigt nach einem Login an, wann sich der entsprechende Nutzer zum letzten Mal erfolgreich angemeldet hat.
7.	Keine gleichzeitigen bzw. parallelen Sitzungen mit dem gleichen Account	Das System verhindert gleichzeitige Sessions von unterschiedlichen Browserinstanzen.
8.	Session Time-out / Session Management / Session Authenticity	<ul style="list-style-type: none"> ➔ Die Applikation unterstützt ein konfigurierbares Session Time-out (Standard: automatisches Logout nach 180min) ➔ Sitzungen, welche beispielsweise durch fehlerhafte Internetverbindung unterbrochen werden, müssen geschlossen werden: Keine hängenden Sitzungen. ➔ Die Sitzungen sind einem Client eindeutig zugewiesen (z.B. Sitzungs-ID in Cookie).

#	Anforderung	Beschreibung
9.	Logout	<p>Die Applikation stellt beim erfolgreichen Logout sicher, dass alle clientseitigen und schützenswerten Informationen (u.a. Browser Cache / Cookies) gelöscht werden.</p> <p>➔ Z.B. darf das Klicken des «Zurück»-Buttons nach erfolgter Abmeldung nicht dazu führen, dass der Benutzer automatisch wieder angemeldet wird.</p>
10.	Log-Format	Sämtliche Logs sollen so aufbereitet sein, dass sie an ein SIEM (Security Information and Event Management) weitergeleitet werden können (wichtig: Syslog-Format).
11.	Ausfallsicherheit	Das System unterstützt eine redundante Auslegung bei Bedarf, sowie ein automatisches Umschalten auf eine Standby-Instanz ohne Datenverlust.
12.	Segmentierung	Die Applikation separiert Benutzer- und Administrations-Funktionen.
13.	Isolierung von sicherheitsrelevanten Funktionen	Die Applikation trennt sicherheitsrelevante Funktionen von nicht-sicherheitsrelevanten Funktionen.
14.	WAF- und Proxy-Kompatibilität	Die Applikation muss auch dann fehlerfrei funktionieren, wenn ihr Datenverkehr durch zwischengeschaltete Sicherheitssysteme (wie WAF oder Proxy) gefiltert und analysiert wird.
15.	Transaktionssicherheit	<p>Diese Anforderung sichert die Datenintegrität bei Fehlern, Systemabstürzen oder parallelem Zugriff. Kritische Operationen müssen dem ACID-Prinzip folgen:</p> <p>A - Atomarität: Eine Transaktion wird nur "ganz oder gar nicht" ausgeführt.</p> <p>C - Konsistenz: Keine ungültigen Zustände.</p> <p>I - Isolation: Gleichzeitige Transaktionen stören sich gegenseitig nicht.</p> <p>D - Dauerhaftigkeit: Erfolgreich abgeschlossene Änderungen sind permanent und überleben Systemabstürze.</p> <p>Viele Datenbanken tragen diesem Prinzip bereits Rechnung, wie z.B. PostgreSQL oder MySQL.</p>