Unsichere API

Softwaretechnik-Projekt IT-Sicherheit
Wintersemester 2024/25

Lastenheft

Hochschule Mittweida

Betreuer: M.Sc Engler, Philipp

Name	Email
Michael P.	mprimke@hs-mittweida.de
Jonas G.	jgermann@hs-mittweida.de
Noah H.	nhiller@hs-mittweida.de
Franz S.	fsander@hs-mittweida.de
Jakob H.	jhindemi@hs-mittweida.de
Hannes L.	hlange3@hs-mittweida.de
Jon R.	jroemmli@hs-mittweida.de

Mittweida 21.01.2025

Inhaltsverzeichnis

1. Ziel / Produktzweck

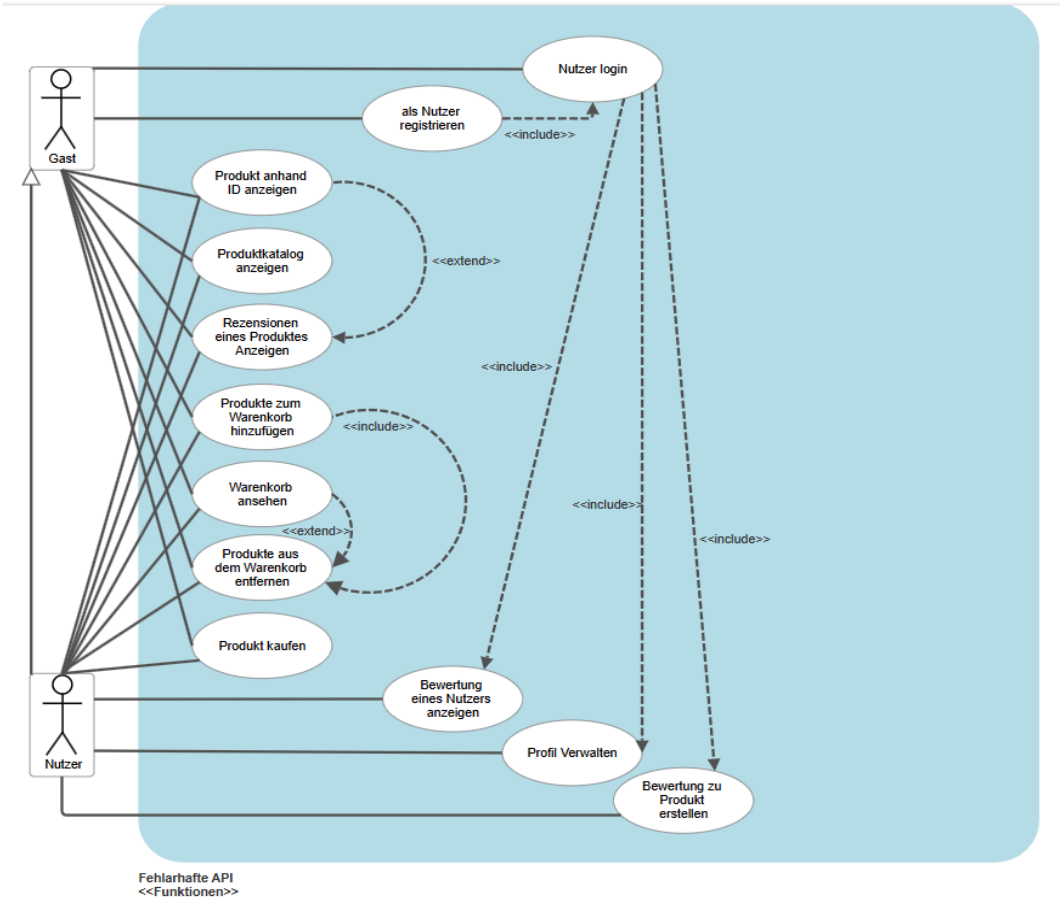
Das Ziel des Projekts ist die Entwicklung einer Software, die als Demonstrationsplattform für Sicherheitslücken in APIs dient. Das Produkt soll typische Schwachstellen aufzeigen, um das Bewusstsein für sichere Softwareentwicklung zu fördern.

2. Produkteinsatz

Die Software wird in Vorlesungen, Schulungen oder Präsentationen eingesetzt, um anhand eines simulierten Online-Shops Sicherheitsmängel zu verdeutlichen. Nutzer sollten die Möglichkeit haben, reale Sicherheitsrisiken zu erkennen und deren Auswirkungen zu verstehen.

3. Produktübersicht

Abbildung 1: Interaktion des Produkts



4. Produktfunktionen

/LF010/

Funktion: Nutzer-Login
Akteur: Gast
Beschreibung: Ein Gast loggt sich mit den persönlichen Daten ein.

/LF020/

Funktion: Nutzer-Registrierung
Akteur: Gast
Beschreibung: Der Gast legt sich ein Konto an, indem er einen Namen und ein Passwort anlegt.

/LF030/

Funktion: Produktkatalog anzeigen
Akteur: Gast, Nutzer
Beschreibung: Der Nutzer kann die Inhalte des Kataloges jederzeit ansehen.

/LF040/

Funktion: Produkt anhand ID anzeigen
Akteur: Gast, Nutzer
Beschreibung: Produkte können mit ihrer ID gesucht und angezeigt werden.

/LF045/

Funktion: Produkt Rezensionen Anzeigen
Akteur: Gast, Nutzer
Beschreibung: Gast/Nutzer kann die Rezensionen zu einem Produkt anzeigen lassen.

/LF050/

Funktion: Produkt kaufen
Akteur: Gast, Nutzer
Beschreibung: Benutzer können Produkte kaufen.

/LF060/

Funktion: Produktbewertung erstellen
Akteur: Nutzer
Beschreibung: Nutzer können Produktbewertungen erstellen.

/LF070/ **Funktion:** Produkte zum Warenkorb hinzufügen
Akteur: Gast, Nutzer
Beschreibung: Gast/Nutzer kann Produkte zum Warenkorb hinzufügen

/LF080/ **Funktion:** Warenkorb ansehen
Akteur: Gast, Nutzer
Beschreibung: Gast/Nutzer kann den Warenkorb und dessen Inhalt sehen.

/LF085/ **Funktion:** Produkte aus Warenkorb entfernen
Akteur: Gast, Nutzer
Beschreibung: Gast/Nutzer kann Produkte aus dem Warenkorb entfernen.

/LF90/ **Funktion:** Nutzer Rezensionen anzeigen
Akteur: Nutzer
Beschreibung: Nutzer können die Rezensionen von anderen Nutzern anzeigen lassen.

5. Produktdaten

/LD10/ Nutzerkontodaten
Maximale Anzahl: ca. 1.000 Nutzerkonten
Beschreibung: Enthält die persönlichen Daten von Nutzern, wie Benutzername, Passwort (nicht gehasht) und E-Mail-Adresse.

/LD20/ Produktkatalogdaten
Maximale Anzahl: ca 1000 Produkte
Beschreibung: Enthält Informationen zu Produkten wie Produkt-ID, Name, Beschreibung, Preis, Verfügbarkeit, Rabatt und ein Thumbnail.

/LD30/ Warenkorbdaten
Maximale Anzahl: ca. 10.000 aktive Warenkörbe gleichzeitig
Beschreibung: Temporäre Speicherung von Produkt-IDs, Mengen und

Zwischensummen für jeden Nutzer, der Produkte in den Warenkorb gelegt hat.

/LD40/ Rezensionsdaten
Maximale Anzahl: ca. 1.000 Rezensionen pro Produkt
Beschreibung: Speichert Bewertungen von Nutzern mit Informationen wie Bewertungspunktzahl, Rezensionstext, Datum der Erstellung und der Nutzer-ID des Verfassers.

/LD50/ Bestelldaten
Maximale Anzahl: ca. 1.000.000 Bestellungen
Beschreibung: Enthält die Bestell-ID, User-ID, den Zeitstempel, Produktdetails (Produkt-ID und Menge).

/LD60/ Sitzungsdaten
Maximale Anzahl: ca. 100.000 aktive Sitzungen gleichzeitig
Beschreibung: Temporäre Speicherung von Sitzungs-IDs, Nutzerdaten und Ablaufzeiten für eingeloggte Nutzer.

/LD70/ Authentifizierungsdaten
Maximale Anzahl: ca. 1.000 aktive Tokens
Beschreibung: Speichert temporäre JWT-Token, die zur Authentifizierung der Benutzer verwendet werden. Es existiert kein Ablaufdatum für die Tokens und es ist ein schwaches Secret zur Signierung der Tokens festgelegt.

6. Produktleistungen

/LL10/ Die Ladezeit des Produktkatalogs
Die Ladezeit des gesamten Produktkatalogs darf bei einem Zugriff auf den Server maximal 2 Sekunden betragen, bei einer durchschnittlichen Netzverbindung (100 Mbit/s).

/LL20/ Die Aufrufzeit eines spezifischen Produkts
Die Anzeige der Details eines Produkts anhand seiner ID darf maximal 1 Sekunde in Anspruch nehmen.

/LL30/ Die Antwortzeit des Warenkorbs
Das Hinzufügen eines Produkts zum Warenkorb muss innerhalb von 1,5 Sekunden abgeschlossen sein.

/LL40/ Die Antwortzeit beim Nutzer-Login
Die Authentifizierung eines Nutzers darf maximal 1 Sekunde dauern, einschließlich Tokengenerierung.

/LL50/ Die Ladezeit der Nutzerrezensionen
Die Anzeige der Rezensionen eines Produkts darf bei maximal 100 Rezensionen nicht länger als 1,5 Sekunden dauern.

/LL60/ Die Belastbarkeit der API
Die API muss mindestens 500 gleichzeitige Anfragen verarbeiten können, ohne die oben definierten Antwortzeiten zu überschreiten.

7. Qualitätsanforderungen nach ISO/IEC 25010:2011

Produktqualität	Sehr gut	Gut	Normal	Nicht relevant
Funktionale Eignung		**X**		
Kompatibilität	**X**			
Sicherheit				**X**
Zuverlässigkeit			**X**	
Benutzbarkeit		**X**		
Leistungs-effizienz			**X**	
Wartbarkeit			**X**	
Portabilität	**X**			

8. Use Cases

Use Case 1: Produkt anhand ID anzeigen
Akteure: Gast, Nutzer

Beschreibung:
Der Akteur möchte die Details eines spezifischen Produkts einsehen. Er kann nach dem Produkt filtern, indem er nach der passenden Produkt ID sucht. Das System zeigt dann die gesamten Produktinformationen an.

Use Case 2: Produktkatalog anzeigen
Akteure: Gast, Nutzer
Beschreibung:
Der Akteur möchte den gesamten Produktkatalog ansehen. Dafür wird die entsprechende API-Route angesteuert. Das System zeigt alle verfügbaren Produkte mit den Informationen in einer Liste an.

Use Case 3: Bewertung zu einem Produkt erstellen
Akteure: Nutzer
Beschreibung:
Der Nutzer möchte eine Bewertung für ein Produkt abgeben. Der Nutzer wählt das Produkt aus und gibt seine Bewertung ein. Das System speichert die Bewertung in der Datenbank und verknüpft sie mit dem Nutzer sowie dem Produkt anhand der IDs.

Use Case 4: Rezensionen eines Produkts anzeigen
Akteure: Gast, Nutzer
Beschreibung:
Der Akteur möchte die Rezensionen eines spezifischen Produkts einsehen. Beim Aufrufen der Produktseite werden die Rezensionen unterhalb des Produkts angezeigt.
Dazu gehören die Bewertung, der Verfasser und der Inhalt der Rezension.

Use Case 5: Bewertungen eines Nutzers anzeigen
Akteure: Nutzer
Beschreibung:

Der Nutzer möchte alle seine bisherigen Bewertungen einsehen. Dazu gibt er seine Nutzer-ID an und das System zeigt alle Bewertungen des Nutzers an.

Use Case 6: Nutzer registrieren
Akteure: Gast
Beschreibung:
Der Gast möchte sich als Nutzer registrieren. Dazu gibt er seine Daten ein und das System speichert diese in der Datenbank.

Use Case 7: Nutzer Login
Akteure: Gast
Beschreibung:
Der Gast möchte sich als Nutzer einloggen. Dazu gibt er seine Anmeldedaten ein und das System überprüft diese. Bei erfolgreicher Anmeldung wird der Nutzer eingeloggt.

Use Case 8: Profil Verwalten
Akteure: Nutzer
Beschreibung:
Der Nutzer kann jederzeit sein Profil bearbeiten und ansehen.

Use Case 9: Produkte zum Warenkorb hinzufügen
Akteure: Gast, Nutzer
Beschreibung:
Der Gast oder der Nutzer können Produkte zum Warenkorb hinzufügen.

Use Case 10: Warenkorb Ansehen
Akteure: Gast, Nutzer
Beschreibung:
Der Gast oder der Nutzer kann den Warenkorb und die Produkte, die er hinzugefügt hat, ansehen.

Use Case 11: Produkte aus dem Warenkorb entfernen
Akteure: Gast, Nutzer
Beschreibung:

Der Gast oder der Nutzer können Produkte aus dem Warenkorb entfernen.

Use Case 12: Produkt kaufen
Akteure: Gast, Nutzer
Beschreibung:
Der Gast, sowie der Nutzer können ein Produkt kaufen.

9. Sicherheitslücken gefolgt von Maßnahmen

9.1 Unsichere Authentifizierung

Die Authentifizierung verwendet JSON Web Tokens (JWT), jedoch fehlen wesentliche Sicherheitsmaßnahmen:

- Tokens ohne Ablaufzeit
- Unsicheres Geheimnis (*secret*)

Mögliche Angriffsvektoren:

- Token Diebstahl: Ein kompromittiertes oder gestohlenes Token kann unbegrenzt verwendet werden.
- Token Fälschung: Ein schwaches Geheimnis ermöglicht es Angreifern, gültige Tokens zu generieren und sich als andere Benutzer auszugeben

Maßnahmen zur Behebung:

1. Token-Ablaufzeit hinzufügen (exp-Claim):
    - Tokens sollten ein Ablaufdatum enthalten, um ihre Gültigkeit zu begrenzen.
2. Starkes Geheimnis verwenden:
    - Ein langes, zufälliges Geheimnis (mindestens 256 Bits) sollte verwendet und sicher gespeichert werden
3. Token-Überprüfung verbessern:
    - Zusätzliche Prüfungen wie IP-Adresse oder Gerätebindung einführen

9.2 Cross-Site Scripting

Angreifer können schädliches JavaScript einschleusen, das bei anderen Nutzern ausgeführt wird, z. B. um Cookies zu stehlen oder Schadcode auszuführen.

Beispiel:
Speichern von Text (z. B. in Rezensionen), ohne HTML- oder JavaScript-Code zu filtern
<script>alert('Hier könnte ihr Name stehen');</script>

Maßnahmen:

- Eingabevalidierung oder Output Encoding (schädliche Zeichen umwandeln -> < zu &lt > zu &gt)

9.3 Fehlende Ratenbegrenzung

Ohne Beschränkung der Anfragen pro Nutzer können Angreifer Brute-Force-Angriffe auf beispielsweise Login-Endpunkte durchführen oder den Server durch Überlastung lahmlegen.

Maßnahmen:

- Rate Limiting (zum Beispiel express-rate-limiting)

Beispiel:
import rateLimit from 'express-rate-limit';
*const limiter = rateLimit({ windowMs: 15 * 60 * 1000, // 15 Minuten*
max: 100, // Max. 100 Anfragen pro IP });
app.use('/login', limiter);

9.4 Unsichere Datenspeicherung

Passwörter oder andere sensible Informationen werden im Klartext gespeichert, was bei einem Datenbank Leck katastrophal ist.

Maßnahmen:

- Hashing, Verschlüsselung

Beispiel:
import bcrypt from 'bcrypt';
const hashedPassword = await bcrypt.hash(password, 10);

9.5 Übermäßige Daten Exposition

Die API gibt unnötig viele Details zurück, z.B. interne IDs oder vertrauliche Daten

Maßnahmen:

- Datenfilterung (nur notwendige Felder zurückgeben)

9.6 Fehlende HTTPS-Verschlüsselung

HTTP überträgt Daten unverschlüsselt, bietet keine Authentizität und ist anfällig für Abhör- und Manipulation Angriffe. Somit besteht die Möglichkeit des Abfangen des Tokens oder des API-Schlüssels.

HTTPS hingegen verwendet SSL/TLS zur Verschlüsselung, stellt die Datenintegrität sicher und nutzt digitale Zertifikate, um sicherzustellen, dass Nutzer mit der echten Website verbunden sind.

9.7 Kein CSRF-Schutz

Der Anwendung fehlt es an Schutzmaßnahmen gegen Cross-Site Request Forgery (CSRF). Dadurch können Angreifer unautorisierte

Aktionen im Namen eines authentifizierten Benutzers durchführen, indem sie den Browser des Benutzers dazu bringen, Anfragen an den Server zu senden.

Angriffsvektor:

- Ein Angreifer erstellt eine bösartige Webseite, die versteckte Anfragen (z. B. POST-Requests) an den Server sendet. Der Browser des Opfers überträgt automatisch gespeicherte Authentifizierungsdaten (z.B. Cookies oder Tokens), wodurch der Server die Anfrage als legitim ansieht

Auswirkungen:

- Unautorisierte Änderungen können im Namen des authentifizierten Benutzers erfolgen

Maßnahmen:

- CSRF-Tokens einführen:
    - Zufällige Tokens, die mit jeder geschützten Anfrage gesendet und validiert werden
- Origin- und Referer-Header prüfen:
    - Sicherstellen, dass die Anfrage von der autorisierten Domain stammt
- Cookies absichern:
    - Verwendung von HTTP-Only- und Secure-Flags, um Cookies vor Diebstahl zu schützen

10. Glossar

Gast:
ist ein neuer Interessent, der an Produkten interessiert ist. Die Registrierung ist noch nicht oder bewusst nicht erfolgt.

Nutzer:
ein Gast, der entschieden hat, sich zu registrieren.

Nutzerrezension:
eine Bewertung oder Meinung, die ein Nutzer zu einem Produkt äußert. Diese besteht aus einer Sterne Bewertung von 1-5 und optional ein Kommentar und eventuell Bilder und Videos.

Onlineshop:
eine Website oder Plattform, auf der Produkte
über das Internet angeboten und verkauft werden. Kunden können dort stöbern, bestellen und bezahlen, ohne ein physisches Geschäft zu besuchen.

API (Application Programming Interface):
Eine Schnittstelle, die es ermöglicht, Softwarekomponenten miteinander zu verbinden und Daten oder Funktionen auszutauschen.

JWT (JSON Web Token):
Ein kompakter, URL-sicherer Standard zur Übertragung von Informationen zwischen zwei Parteien als JSON-Objekt. Wird oft zur Authentifizierung verwendet.

XSS (Cross-Site Scripting):
Eine Sicherheitslücke, bei der Angreifer schädliches JavaScript einschleusen können, das bei anderen Nutzern im Browser ausgeführt wird.

Rate Limiting:
Eine Technik, um die Anzahl der Anfragen an eine API oder einen Server innerhalb eines bestimmten Zeitraums zu begrenzen, um Missbrauch oder Überlastung zu verhindern.

Hashing:
Ein kryptographischer Prozess, bei dem eine Eingabe (z. B. ein

Passwort) in eine feste Länge umgewandelt wird, um Daten sicher zu speichern. Beispiele: bcrypt, SHA-256.

HTTPS (HyperText Transfer Protocol Secure):
Eine Erweiterung von HTTP, die eine verschlüsselte Kommunikation zwischen dem Nutzer und der Website sicherstellt.

SQL (Structured Query Language):
Eine Sprache zur Verwaltung und Abfrage von Daten in relationalen Datenbanken.
Token:
Ein digitaler Schlüssel, der oft für die Authentifizierung oder Autorisierung verwendet wird, z. B. in API-Interaktionen.

OWASP (Open Web Application Security Project):
Eine gemeinnützige Organisation, die sich auf die Verbesserung der Sicherheit von Softwareanwendungen konzentriert.

Prepared Statements:
Eine Methode zur sicheren Ausführung von SQL-Abfragen, die verhindert, dass Benutzereingaben direkt in SQL-Abfragen eingefügt werden.

Output Encoding:
Die Umwandlung von Zeichen in sichere Darstellungen, um sicherzustellen, dass Eingaben nicht als ausführbarer Code interpretiert werden.

DSGVO (Datenschutz-Grundverordnung):
Eine EU-Verordnung, die Regeln für den Schutz personenbezogener Daten vorgibt.

Bcrypt:
Eine Passwort-Hashing-Funktion, die zur sicheren Speicherung von Passwörtern verwendet wird, da sie zusätzliche Sicherheitsmaßnahmen wie Salting integriert.

Salting:
Ein Verfahren, bei dem ein zufälliger Wert zu einem Passwort hinzugefügt wird, bevor es gehasht wird, um Hash-Wiederholungen zu verhindern.