CyberTestify

Automatisierter Sicherheits-Scan-Bericht

ornek.com · Paket Aktive Verifikation

CyberTestify-Sicherheitsscan — Abgeschlossen

Bericht-Nr.: **CT-ÖRNEK-715E**

Verifizierungscode: **4B2E-B284**

Dieser Bericht ist kein formeller Penetrationstest / keine Zertifizierung.

Inhaltsverzeichnis

ZIEL
ornek.com

PAKET
Paket Aktive Verifikation

BERICHT-NR.
CT-ÖRNEK-715E

1. Managementzusammenfassung

GESAMTBEWERTUNG Hohes Risiko

In den ausgeführten Kontrollen wurde das höchste Risiko im Bereich Injektions-(SQLi/XSS)-Verifikation (mit aktiver Verifikation wurde eine Injektions-Schwachstelle NACHGEWIESEN.) festgestellt; es wird empfohlen, dieses vorrangig zu beheben/zu verifizieren. 8 eindeutige Seiten/Endpunkte gescannt, an 19 Einstiegspunkten insgesamt 149 Anfragen gesendet. SSRF/RCE sind zeit-basiert/indirekt. Nachfolgend wird jede Kontrolle separat berichtet.

⚡ Management-Entscheidung

Es wurden **2** Indikator(en) mit hohem/kritischem Schweregrad festgestellt, die vorrangig zu behandeln sind; ein dringender Behebungsplan wird empfohlen. Mittlere/geringe Punkte können durch geplante Verbesserung behoben werden.

- **Allgemeine Risikostufe: Hoch** — das höchste Risiko liegt im Bereich **Injektions-(SQLi/XSS)-Verifikation** (mit aktiver Verifikation wurde eine Injektions-Schwachstelle NACHGEWIESEN.).

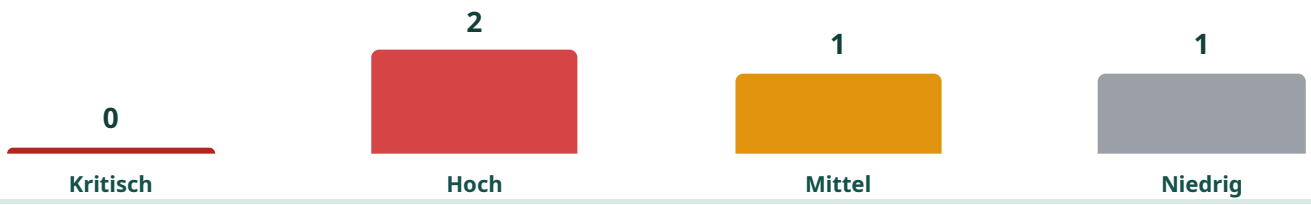
Verbesserungsprioritäten

⚡ **Am dringendsten:** SQL-Injection-Indikator; Authentifizierungs-Bypass-Indikator — ' OR 1=1-- (POST /rest/user/login).

🗓 **Mittelfristig / Prozess (manuelle Verifizierung oder Code/Architektur):** /api/products/1; Reflected-XSS-Indikator.

2. Befunde

2.1 Schwachstellenverteilung



In diesem Scan wurden insgesamt **4** Befund-Indikatoren festgestellt; die Verteilung nach Schweregrad ist unten dargestellt.

2.2 Master-Befundtabelle

ID	Titel	Status	Schweregrad
CT-1	SQL-Injection-Indikator	Offen	Hoch

ID	Titel	Status	Schweregrad
CT-2	Authentifizierungs-Bypass-Indikator — ' OR 1=1-- — POST /rest/user/login	Offen	Hoch
CT-3	/api/products/1	Offen	Mittel
CT-4	Reflected-XSS-Indikator	Offen	Niedrig

8 Verifizierungskategorien wurden ausgeführt — 2 sauber, 3 mit Befund, 3 nicht prüfbar

✓ VERSUCHT — KEIN SCHWACHSTELLENNACHWEIS GEFUNDEN

- **SSRF-Verifikation** Sauber (2 Einstiegspunkte versucht, kein Nachweis gefunden)
- **RCE- / Command-Injection-Verifikation** Sauber (4 Einstiegspunkte versucht, kein Nachweis gefunden)

⚠ BEFUND VORHANDEN — DETAILS IN DEN ABSCHNITTEN UNTEN

- **Injektions-(SQLi/XSS)-Verifikation** Befund vorhanden (2 — Schwachstellen-Indikator; oben)
- **Unbefugter-Zugriff-(IDOR)-Verifikation** Befund vorhanden (1 — eingeschränkter/indirekter Indikator; oben)
- **Login-Bypass (SQLi-Indikator)** Befund vorhanden (1 — Schwachstellen-Indikator; oben)

⚠ NICHT PRÜFBAR — BEDEUTET NICHT „SICHER“

- **Datei-Upload-Verifikation** Nicht untersuchbar (kein testbarer Einstiegspunkt gefunden — NICHT „sauber")
- **Geschäftslogik-Verifikation** Nicht untersuchbar (kein testbarer Einstiegspunkt gefunden — NICHT „sauber")
- **Race- / Mass-Assignment-Verifikation** Nicht untersuchbar (kein testbarer Einstiegspunkt gefunden — NICHT „sauber")

2.3 Detaillierte Befunde

[HOCH] CT-1 · SQL-Injection-Indikator

Status: Offen

Beschreibung: Eine Eingabe erreicht eine DB-Abfrage (Injection-Indikator).

Wie es erkannt wurde: In der Antwort wurde eine Datenbank-Fehlersignatur beobachtet (mit dem Payload „')"): „SQLITE_ERROR"

Geschäftliche Auswirkung: Unbefugter Zugriff/Änderung von Kunden-/Kontodatensätzen in der Datenbank möglich; Risiko von Datenpanne, **Verstoß gegen die DSGVO** und Vertrauensverlust.

EMPFOHLENE BEHEBUNG

Verwenden Sie parametrisierte Abfragen / Prepared Statements; verbergen Sie DB-Fehlermeldungen.

Referenz: CWE-89 · OWASP A03:2021 Injection

[HOCH] CT-2 · Authentifizierungs-Bypass-Indikator — ' OR 1=1--

Status: Offen

Betroffener Punkt: `POST /rest/user/login`

Beschreibung: Auth-Bypass-Indikator über einen klassischen SQLi-Payload im Login-Formular.

Wie es erkannt wurde: Kontrolle (ungültige Anmeldedaten) → HTTP 401 (fehlgeschlagen). SQLi-Payload ' OR 1=1-- → HTTP 200 + EINDEUTIGES Erfolgssignal: „{"authentication":{"token":"","bid":1,"umail":"admin@juice-sh.op"}}". Die Authentifizierung wird per SQL-Injektion UMGANGEN (ein Sitzungs-/Autorisierungs-Token wurde zurückgegeben — starker Nachweis). Es wurde KEINE Sitzungsübernahme/-ausnutzung durchgeführt; der Token-Wert wird im Bericht nicht angezeigt (redigiert).

Geschäftliche Auswirkung: Eine Umgehung der Authentifizierung kann unbefugten Zugriff ermöglichen; Konto- und Datensicherheit direkt gefährdet.

EMPFOHLENE BEHEBUNG

Verwenden Sie in der Authentifizierung parametrisierte Abfragen; validieren Sie Eingaben; geben Sie einheitliche Fehlermeldungen zurück.

Referenz: CWE-287 · OWASP A07:2021 Identification & Authentication Failures

[NIEDRIG] CT-4 · Reflected-XSS-Indikator

Status: Offen

Beschreibung: Nutzereingabe wird unkodiert im HTML reflektiert (Reflected-XSS-Indikator).

Wie es erkannt wurde: Die Markierungszeichenfolge wurde reflektiert, aber TEILWEISE/KODIERT (Sonderzeichen < > " maskiert) — kontextabhängiger, gering zuverlässiger Indikator; manuelle Verifikation empfohlen.

Geschäftliche Auswirkung: Durch Ausführen von Skript im Browser des Opfers sind Session-Diebstahl/Identitätsvortäuschung und Kontoübernahme möglich.

EMPFOHLENE BEHEBUNG

Kodieren Sie die Ausgabe kontextabhängig (HTML-Entity-Encoding); beschränken Sie Inline-Skripte per CSP.

Referenz: CWE-79 · OWASP A03:2021 Injection

3. Kontrollübersicht & Methodik

Bewertungsübersicht

Alle 7 aktiven Sicherheitskontrollkategorien wurden bewertet. 8 eindeutige Seiten/Endpunkte gescannt, **19** Einstiegspunkte getestet, insgesamt **149** Anfragen gesendet. Bei **2** Kontrollen wurde ein Indikator für eine Schwachstelle hoher/kritischer Stufe gefunden (nachfolgend im Detail).

*Reconnaissance-Methode: Dieser Scan wurde durchgeführt, indem die Seiten **mit einem Headless-Browser gerendert** wurden, da ein per JavaScript gerendertes Ziel (SPA) erkannt wurde.*

KONTROLLÜBERSICHT

Kontrolle	Ergebnis	Zuversicht
Injektions-(SQLi/XSS)-Verifikation	⚠ Schwachstellen-Indikator	Hoch
Unbefugter-Zugriff-(IDOR)-Verifikation	⚠ Eingeschränkter Indikator	Mittel
SSRF-Verifikation	✓ Kein Schwachstellennachweis	Mittel

Kontrolle	Ergebnis	Zuversicht
Datei-Upload-Verifikation	Kein Einstiegspunkt (Außerhalb des Umfangs)	Außerhalb des Umfangs
Geschäftslogik-Verifikation	Kein Einstiegspunkt (Außerhalb des Umfangs)	Außerhalb des Umfangs
Race- / Mass-Assignment-Verifikation	Kein Einstiegspunkt (Außerhalb des Umfangs)	Außerhalb des Umfangs
RCE- / Command-Injection-Verifikation	✓ Kein Schwachstellennachweis	Mittel
Login-Bypass (SQLi-Indikator)	⚠ Schwachstellen-Indikator	Hoch

> Die Zuversicht wird nur für Kontrollen angezeigt, die tatsächlich ausgeführt wurden (bei denen ein Einstiegspunkt gefunden wurde); Kontrollen ohne gefundenen Einstiegspunkt sind **Außerhalb des Umfangs**. Bei den getesteten: SSRF/RCE indirekt (zeit-basiert, kein OOB) → Mittel; beobachtend (Datei-Upload/Geschäftslogik/Race) → Niedrig; Injektion fehler-/reflexionsbasiert → Hoch.

POSITIVE ZUSICHERUNG — VERSUCHTE AKTIVE VERIFIKATIONSMETHODEN

Einschließlich der Kontrollen ohne Befund wurde jede der 8 aktiven Kontrollkategorien auf der entdeckten Oberfläche tatsächlich ausgeführt (insgesamt **149** Anfragen, **8** eindeutige Seiten). Die folgende Tabelle zeigt auch die „kein Befund"-Ergebnisse transparent — wie viele Einstiegspunkte versucht wurden, bei wie vielen kein Nachweis gefunden wurde:

Kontrolle	Versuchter Einstiegspunkt	Gesendete Anfrage	Ergebnis
Injektions-(SQLi/XSS)-Verifikation	10	121	**⚠ Befund vorhanden (2 — Schwachstellen-Indikator; oben)**
Unbefugter-Zugriff-(IDOR)-Verifikation	2	6	**⚠ Befund vorhanden (1 — eingeschränkter/indirekter Indikator; oben)**
SSRF-Verifikation	2	3	**✓ Sauber (2 Einstiegspunkte versucht, kein Nachweis gefunden)**
Datei-Upload-Verifikation	0	1	**⚠ Nicht untersuchbar (kein testbarer Einstiegspunkt gefunden — NICHT „sauber")**
Geschäftslogik-Verifikation	0	1	**⚠ Nicht untersuchbar (kein testbarer Einstiegspunkt gefunden — NICHT „sauber")**
Race- / Mass-Assignment-Verifikation	0	1	**⚠ Nicht untersuchbar (kein testbarer Einstiegspunkt gefunden — NICHT „sauber")**
RCE- / Command-Injection-Verifikation	4	13	**✓ Sauber (4 Einstiegspunkte versucht, kein Nachweis gefunden)**
Login-Bypass (SQLi-Indikator)	1	3	**⚠ Befund vorhanden (1 — Schwachstellen-Indikator; oben)**

> **Drei-Zustands-Unterscheidung (Ehrlichkeit):** ✓ *Sauber* = Kontrolle lief, kein Nachweis gefunden · ⚠ *Befund vorhanden* = oben im Detail · ⚠ *Nicht untersuchbar* = kein testbarer Einstiegspunkt gefunden (bedeutet NICHT sicher).

Was dieses Paket bewertet und was NICHT

TUT (nach dem Prinzip „nachweisen — nicht ausnutzen"; harmlose, daten-unverändernde Proben): Indikatoren für SQLi-/XSS-Injektion, unbefugten Zugriff (IDOR), SSRF, Datei-Upload, Geschäftslogik, Race/Mass-Assignment und RCE/Command-Injection — auf der **nicht authentifizierungspflichtigen** Oberfläche, auf den 8 entdeckten Seiten.

TUT NICHT: Daten-verändernde/löschende Ausnutzung, Zahlungsabschluss oder echte RCE-Ausführung werden **nicht durchgeführt** (es werden nur Indikatoren/Nachweise erhoben). Die Kontrollen IDOR / Geschäftslogik / Race-Mass-Assignment laufen **nur auf der vor dem Login erreichbaren Oberfläche**; tiefe Autorisierungs-/Rechteausweitungs-/Geschäftslogik-Schwachstellen innerhalb der Sitzung **NACH** dem Login liegen **außerhalb** dieses Pakets — diese fallen in den Umfang des **Umfassenden Pentests** (authentifiziert, mit Umfangsvereinbarung). Daher ist in diesen drei Kategorien bei manchen Zielen ein **eingeschränktes oder „Nicht untersuchbar"**-Ergebnis normal und zu erwarten (wegen der geringen zielspezifischen Oberfläche; nicht wegen eines Engine-Mangels). „Kein Befund" bei einer Kontrolle **BEWEIST NICHT, dass Sie sicher sind**, da die aktive Ausnutzung bewusst eingeschränkt/passiv-sicher gehalten wird.

Injektions-(SQLi/XSS)-Verifikation

WAS GEPRÜFT WURDE

Auf den aus den gescannten Seiten (Startseite + interne Links + wohlbekannte Pfade) entdeckten Einstiegspunkten (URL-Query-Parameter + Formularfelder) wurden pro Einstiegspunkt harmlose Verifikations-Proben durchgeführt:

- **SQLi (fehlerbasiert):** Ein einfaches Anführungszeichen (`'`) wurde injiziert und in der Antwort nach einer Datenbank-Fehlersignatur (MySQL/PostgreSQL/Oracle/MSSQL/SQLite) gesucht.
- **SQLi (zeit-basiert):** An Stellen, wo kein Fehler auftrat, wurde mit einer einzelnen harmlosen Verzögerungsprobe (SLEEP) die Antwortzeit gegen die Baseline gemessen (Blind-SQLi-Indikator).
- **SQLi (boolean-basiert):** An numerischen/ID-ähnlichen Stellen wurden zwei bedingte Anfragen mit TRUE (`1=1`) und FALSE (`1=2`) gesendet und ihre Antworten (Status + Inhaltslänge) verglichen; ist die TRUE-Wiederholung konsistent und DAUERHAFT verschieden von FALSE, ist dies ein boolean-basierter SQLi-Indikator (zur Absicherung gegen False Positives wird eine Stabilitätsverifikation durchgeführt).
- **XSS (reflektiert):** Eine eindeutige, harmlose Markierungszeichenfolge wurde injiziert und geprüft, ob sie im Antwort-HTML **unmaskiert (unencoded)** reflektiert wird (kein JS ausgeführt; kein Stored XSS versucht).

BEFUNDE

Einstiegspunkt	Typ	Technik	Nachweis	Zuversicht (Begründung)	Schweregrad
GET /rest/products/search?q	SQLi	fehlerbasiert	In der Antwort wurde eine Datenbank-Fehlersignatur beobachtet (mit dem Payload „')"): „SQLITE_ERROR"	Hoch — Datenbank-Fehlersignatur in der Antwort (direkter Nachweis)	Hoch
GET /redirect?to	XSS	Reflexion	Die Markierungszeichenfolge wurde reflektiert, aber TEILWEISE/KODIERT (Sonderzeichen `< > "` maskiert) — kontextabhängiger, gering zuverlässiger Indikator; manuelle Verifikation empfohlen.	Niedrig — reflektiert, aber kodiert/maskiert; kontextabhängiger schwacher Indikator	Niedrig

> **Umfang und Methode:** Dieses Paket arbeitet nach dem Prinzip „nachweisen — nicht ausnutzen". Das Backend hat eine begrenzte Anzahl **harmloser** Verifikations-Proben an das Ziel gesendet; es wurden keine Daten abgerufen, verändert oder gelöscht. Zwischen den Anfragen werden Wartezeiten und ein Ziel-Gesundheits-Schutzschalter (aufeinanderfolgende 5xx / übermäßige Verlangsamung / WAF) angewendet. Authentifizierungspflichtige Bereiche und interne Logik liegen außerhalb des Umfangs dieses Pakets.

Unbefugter-Zugriff-(IDOR)-Verifikation

WAS GEPRÜFT WURDE

Auf den aus den gescannten Seiten entdeckten Endpunkten mit vorhersehbarer/numerischer ID (z. B. `?id=123`, `/user/45`):

- Der ID-Wert wurde auf einen **benachbarten Wert** (N-1 / N+1) geändert und eine **GET**-Anfrage ohne Authentifizierung gesendet.
- Es wurden nur **Status und Größe** der Antwort verglichen; **der zurückgegebene Inhalt wurde nicht gespeichert/zitiert**.
- Die Rückgabe einer anderen, gültig erscheinenden Ressource galt als Indikator für aufzählbaren Zugriff.

BEFUNDE

Endpunkt	ID	Beobachtung	Schweregrad
/api/products/1	path-id	Ohne Authentifizierung wurde für die benachbarte ID (2) eine 200-Antwort mit derselben STRUKTUR (JSON-Gerüst), aber ANDEREM INHALT zurückgegeben — höchstwahrscheinlich die Daten eines anderen Datensatzes (die zurückgegebenen Daten werden im Bericht nicht angezeigt). Indikator für aufzählbaren Ressourcenzugriff (mögliches IDOR).	Mittel

UMFANGSGRENZE (WICHTIG)

Dieses Paket arbeitet **ohne Authentifizierung**. Daher kann es nur unbefugten Zugriff auf **öffentlich zugängliche, aufzählbare Ressourcen** feststellen. Klassisches IDOR (dass ein Benutzer auf die Daten eines anderen angemeldeten Benutzers zugreift) erfordert **zwei verschiedene Konten/Sitzungen** und liegt außerhalb des Umfangs dieses Pakets. Das Fehlen eines Befunds in diesem Abschnitt **beweist nicht**, dass es in authentifizierten Abläufen kein IDOR gibt — dies erfordert einen separaten authentifizierten Test (**Prüfung erforderlich / Außerhalb des Umfangs**).

> Außerdem wurden aus **1** sammlungsartigen Endpunkt (z. B. `/rest/products`) sequenzielle numerische IDs (`/{1..3}`) abgeleitet und per GET mit der Inhalts-Differenz-Methode getestet.

SSRF-Verifikation

WAS GEPRÜFT WURDE

- Parameter, die einen serverseitigen Fetch auslösen könnten (url/webhook/image/redirect usw.), wurden festgestellt.
- Diesen Parametern wurde eine verzögerte Echo-URL **unter unserer Kontrolle** übergeben; die Antwortzeit des Ziels wurde mit der Baseline verglichen (ruft der Server diese URL ab, verzögert sich die Antwort).
- Internes Netzwerk / Cloud-Metadaten / localhost (169.254.169.254, RFC1918, 127.0.0.1 usw.) wurden **niemals** ins Visier genommen (fest im Code verankerter Hard-Guard).

BEFUNDE

Gegen die gesendeten harmlosen Proben wurde kein eindeutiger Schwachstellen-Indikator gefunden.

> Da keine OOB-Verifikationsinfrastruktur verwendet wurde, ist diese Feststellung **zeit-basiert, indirekt und von mittlerer Zuverlässigkeit**; für eine sichere Verifikation wird ein zusätzlicher/manueller Test empfohlen.

Datei-Upload-Verifikation

WAS GEPRÜFT WURDE

- Ein Datei-Upload-Formular (input type=file) wurde festgestellt.
- Eine einmalige, **harmlose und nicht ausführbare (inerte)**, doppelt-erweiterte (.php.txt) Testdatei wurde gesendet; nur der Annahme-/Ablehnungsstatus wurde beobachtet.
- Die hochgeladene Datei wurde **nicht wieder abgerufen/ausgeführt** (fest im Code verankerte Regel).

BEFUNDE

Gegen die gesendeten harmlosen Proben wurde kein eindeutiger Schwachstellen-Indikator gefunden.

> Auf den 8 gescannten eindeutigen Seiten wurde weder ein Datei-Upload-Formular (input type=file) noch ein Datei-Upload-Endpunkt im Netzwerkverkehr gefunden. **Hinweis:** Das Ziel ist eine per JavaScript gerenderte Anwendung (SPA), und dieser Scan wurde durchgeführt, indem die Seiten **mit einem Headless-Browser gerendert** wurden; dass dennoch kein testbarer Einstiegspunkt gefunden wurde, zeigt, dass auf der nach dem Rendern vorliegenden Seite tatsächlich kein Einstiegspunkt existiert (keine Roh-HTML-Beschränkung — ein stärkerer „sauber"-Indikator; dennoch liegen authentifizierte Abläufe außerhalb des Umfangs).

Geschäftslogik-Verifikation

WAS GEPRÜFT WURDE

- Auf der Startseite/den Formularen wurden **clientseitig veränderbare** Preis-/Mengenfelder (hidden input) beobachtet (nur Beobachtung — keine Anfrage gesendet).
- Es wurde geprüft, ob eine „Erfolgs-/Bestätigungs"-Schrittseite ohne Vorbedingung **nur per GET** erreichbar ist (Schritt-Übersprung-Indikator).

⚠ Diese Kontrolle sendet **keine zustandsverändernde Anfrage (POST/PUT/…)** — Warenkorb/Zahlung werden **niemals** erstellt/abgeschlossen (fest im Code verankerte Regel).

BEFUNDE

Gegen die gesendeten harmlosen Proben wurde kein eindeutiger Schwachstellen-Indikator gefunden.

> Geschäftslogik-Schwachstellen sind kontextspezifisch; diese Kontrolle ist auf Oberflächen-/Indikatorebene. Eine sichere Verifikation erfordert authentifiziertes manuelles Testen.

> Auf den 8 gescannten eindeutigen Seiten wurde weder ein beobachtbares clientseitiges Preis-/Mengenfeld noch ein direkt erreichbarer „Bestätigungs"-Schritt gefunden. **Hinweis:** Das Ziel ist eine per JavaScript gerenderte Anwendung (SPA), und dieser Scan wurde durchgeführt, indem die Seiten **mit einem Headless-Browser gerendert** wurden; dass dennoch kein testbarer Einstiegspunkt gefunden wurde, zeigt, dass auf der nach dem Rendern vorliegenden Seite tatsächlich kein Einstiegspunkt existiert (keine Roh-HTML-Beschränkung — ein stärkerer „sauber"-Indikator; dennoch liegen authentifizierte Abläufe außerhalb des Umfangs).

Race- / Mass-Assignment-Verifikation

WAS GEPRÜFT WURDE

- Ein registrierungs-/profilartiges POST-Formular wurde festgestellt (Zahlungs-/Abschluss-Endpunkte wurden **ausgeschlossen** — fest im Code verankerte Blocklist).
- Es wurde **eine einzige** Anfrage gesendet, in der dem Formular zusätzliche `isAdmin/role`-Felder hinzugefügt wurden; nur Annahme/Ablehnung wurde beobachtet (eine Berechtigungsänderung wurde **nicht bestätigt**; **keine** Wiederholung/kein Retry).
- Der Race-Condition-(Nebenläufigkeits-)Test wurde **nicht automatisch ausgeführt**, da er das Risiko birgt, eine verbrauchbare Ressource tatsächlich zu verändern (Hinweis unten).

BEFUNDE

Gegen die gesendeten harmlosen Proben wurde kein eindeutiger Schwachstellen-Indikator gefunden.

> Der Mass-Assignment-Indikator wurde nur aus der ersten Antwort abgeleitet (geringe Zuversicht). Für die Race-Condition wird eine manuelle Verifikation mit einem sicheren/testbaren Endpunkt empfohlen.

> Der Race-Condition-(Nebenläufigkeits-)Test wurde in diesem automatischen Scan **nicht ausgeführt**, da er das Risiko birgt, eine verbrauchbare Ressource (Coupon/Bestand) tatsächlich zu verändern; eine manuelle Verifikation mit einem sicheren/testbaren Endpunkt wird empfohlen. Auf den 8 gescannten eindeutigen Seiten wurde kein für Mass-Assignment geeignetes (nicht-abschluss-/zahlungsbezogenes) Registrierungs-/Profilformular gefunden. **Hinweis:** Das Ziel ist eine per JavaScript gerenderte Anwendung (SPA), und dieser Scan wurde durchgeführt, indem die Seiten **mit einem Headless-Browser gerendert** wurden; dass dennoch kein testbarer Einstiegspunkt gefunden wurde, zeigt, dass auf der nach dem Rendern vorliegenden Seite tatsächlich kein Einstiegspunkt existiert (keine Roh-HTML-Beschränkung — ein stärkerer „sauber"-Indikator; dennoch liegen authentifizierte Abläufe außerhalb des Umfangs).

RCE- / Command-Injection-Verifikation

WAS GEPRÜFT WURDE

- Einstiegsparameter, die einen Befehl erreichen könnten, wurden festgestellt.
- Es wurden nur **harmlose, zeit-basierte** Verzögerungs-Payloads (sleep) gesendet; die Antwortzeit wurde mit der Baseline verglichen (Blind-Nachweis).
- Echte Befehlsausführung (Datei lesen/schreiben, Netzwerkverbindung, Reverse Shell) wurde **niemals** versucht (fest im Code verankerter Hard-Guard: nur feste Liste von sleep-Payloads).

BEFUNDE

Gegen die gesendeten harmlosen Proben wurde kein eindeutiger Schwachstellen-Indikator gefunden.

> Da keine OOB-/Canary-Infrastruktur verwendet wurde, ist diese Feststellung **zeit-basiert, indirekt und von mittlerer Zuverlässigkeit** (Netzwerklatenz kann irreführen); für eine sichere Verifikation wird ein manueller Test empfohlen.

Login-Bypass (SQLi-Indikator)

WAS GEPRÜFT WURDE

- An den Login-Endpunkt wurden zuerst **ungültige Anmeldedaten** (Kontrolle) gesendet; anschließend wurden klassische SQLi-Payloads (`' OR '1'='1` usw.) versucht und beobachtet, ob im GEGENSATZ zur Kontrolle eine Sitzung/ein Erfolg (Token/2xx) zurückgegeben wurde.
- Der Login-POST ist ohnehin ein erlaubter Ablauf; dies ist eine EINZIGE und harmlose Beobachtung.

⚠ KEINE Sitzungsübernahme/-ausnutzung — nur die Beobachtung „gibt es einen Indikator für Authentifizierungs-Umgehung".

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
POST /rest/user/login	Login-Bypass (SQLi: `' OR 1=1--`)	Kontrolle (ungültige Anmeldedaten) → HTTP 401 (fehlgeschlagen). SQLi-Payload `' OR 1=1--` → HTTP 200 + EINDEUTIGES Erfolgssignal: „{"authentication": {"token":"***","bid":1,"umail":"admin@juice-sh.op"}}". Die Authentifizierung wird	Hoch	keins	Hoch

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
		per SQL-Injektion UMGANGEN (ein Sitzungs-/Autorisierungs-Token wurde zurückgegeben — starker Nachweis). Es wurde KEINE Sitzungsübernahme/-ausnutzung durchgeführt; der Token-Wert wird im Bericht nicht angezeigt (redigiert).			

> Der Indikator beruht auf dem Vergleich mit dem Kontrollversuch; eine sichere Verifikation erfordert einen manuellen Test.

Methodik

Phase	Beschreibung
Erkundung	Die externe Oberfläche, Seiten und (bei SPAs) echte Endpunkte/Parameter aus dem JS-Bundle werden extrahiert.
Automatische Erkennung	Auf der ermittelten Oberfläche werden deterministische, sichere (read-only) Indikatoren gesucht.
Manuell-deterministische Verifizierung	Befunde werden mit code-basierten Regeln verifiziert; „nachweisen, nicht ausnutzen“.
Autorisierung & Session	Im authentifizierten Paket werden Cookie/Session/Autorisierung und die Post-Login-Oberfläche geprüft.
Konfiguration	Header-, TLS-, E-Mail-/DNS- und Preisgabe-Konfigurationen werden beobachtet.

Risikobewertungskriterien

Schweregrad	Definition
Kritisch	Direkt/leicht ausnutzbarer Indikator mit erheblicher Daten-/Zugriffsauswirkung.
Hoch	Hervorstechender, vorrangig zu behebender starker Indikator.
Mittel	Erfordert Aufmerksamkeit; kontextabhängige Verifizierung empfohlen.
Niedrig	Härtungs-/Reifegrad-Chance; niedrige Priorität.

Der Schweregrad ist nur ein Bandlabel (kein numerischer CVSS-Score); alle Befunde sind als „Indikator, Verifizierung erforderlich“ gerahmt.

4. KI-Lösungsempfehlungen

Dieser Abschnitt enthält Korrekturvorschläge für die in den ausgeführten aktiven Verifikationskontrollen festgestellten Befunde.

Injektions-(SQLi/XSS)-Verifikation

Injektion (SQLi/XSS) — Korrektur

- **SQLi:** Schreiben Sie alle Datenbankabfragen mit **parametrisierten Abfragen / Prepared Statements**; fügen Sie Benutzereingaben niemals als String in die Abfrage ein. Wenn Sie ein ORM verwenden, vermeiden Sie das Verketten von rohem SQL. Zeigen Sie dem Endbenutzer keine Datenbank-Fehlermeldungen an.
- **XSS:** Wenden Sie beim Ausgeben von Benutzereingaben in HTML eine **kontextgerechte Ausgabekodierung** (HTML-Entity-Encoding) an; verwenden Sie nach Möglichkeit eine Template-Engine mit automatischem Escaping. Beschränken Sie Inline-Skripte mit dem `Content-Security-Policy`-Header.
- Testen Sie nach der Behebung dieselben Einstiegspunkte erneut.

Unbefugter-Zugriff-(IDOR)-Verifikation

Unbefugter Zugriff (IDOR) — Korrektur

- **Objektebenen-Autorisierung:** Verifizieren Sie bei jedem Ressourcenzugriff serverseitig, ob der anfragende Benutzer ein Zugriffsrecht auf dieses Objekt hat (dass die ID gültig ist, reicht allein nicht aus).
- **Nicht erratbare Bezeichner:** Verwenden Sie statt sequenzieller numerischer IDs UUIDs/zufällige Bezeichner; erschweren Sie die Aufzählung.
- Stellen Sie Ressourcen, die nicht öffentlich zugänglich sein sollen, hinter eine Authentifizierung.

SSRF-Verifikation

SSRF — proaktive Härtung

- Wenden Sie in allen Feldern, die eine URL vom Benutzer entgegennehmen, eine serverseitige **Allowlist** + Sperrung des internen Netzwerks an (proaktiv).
- Setzen Sie bei erforderlichem externen Fetch Schema-/Host-Verifikation + Timeout + Größenlimit.

Datei-Upload-Verifikation

Datei-Upload — proaktive Härtung

- Wenden Sie an Upload-Endpunkten serverseitige Typ-/MIME-Verifikation + Allowlist + Speicherung außerhalb des Web-Roots an (proaktiv).

Geschäftslogik-Verifikation

Geschäftslogik — proaktive Härtung

- Verifizieren Sie kritische Werte (Preis/Menge) serverseitig; wenden Sie in mehrstufigen Abläufen eine Schrittreihenfolgeprüfung an (proaktiv).

Race- / Mass-Assignment-Verifikation

Race / Mass-Assignment — proaktive Härtung

- Wenden Sie beim Model-Binding eine Feld-Allowlist (Mass-Assignment-Schutz) an; verwenden Sie bei kritischen Operationen ein atomares/idempotentes Design (proaktiv).

RCE- / Command-Injection-Verifikation

RCE / Command-Injection — proaktive Härtung

- Überprüfen Sie Codepfade, die Systembefehle aufrufen; verifizieren Sie Eingaben mit einer Allowlist, vermeiden Sie Shell-String-Verkettung (proaktiv).
- Wenden Sie geringste Rechte + Einschränkung des ausgehenden Netzwerkverkehrs an.

Login-Bypass (SQLi-Indikator)

Login-Bypass / SQL-Injektion — Korrektur

- Verwenden Sie in Authentifizierungsabfragen **parametrisierte Abfragen / Prepared Statements**; setzen Sie Benutzereingaben niemals direkt in SQL ein.
- Eingabevalidierung + sichere ORM-APIs; geben Sie bei fehlerhafter Eingabe eine einheitliche Fehlermeldung zurück.

5. Anhang — Glossar

SQL Injection	Eine Injection-Schwachstelle, bei der Nutzereingaben in eine Datenbankabfrage gelangen.

XSS	Cross-Site Scripting — das Einschleusen schädlicher Skripte in Seiten.
IDOR	Insecure Direct Object Reference — Zugriff auf fremde Datensätze durch ID-Manipulation.
SSRF	Server-Side Request Forgery — den Server zu unerwünschten Anfragen zwingen.