CyberTestify

Automatisierter Sicherheits-Scan-Bericht

ornek.com · Umfassendes Pentest-Paket

CyberTestify-Sicherheitsscan — Abgeschlossen

Bericht-Nr.: **CT-ÖRNEK-332E**

Verifizierungscode: **EB4F-F4B1**

Dieser Bericht ist kein formeller Penetrationstest / keine Zertifizierung.

Inhaltsverzeichnis

1. Managementzusammenfassung

⚠ Dieser Beispielbericht stammt aus einer absichtlich verwundbar belassenen Testanwendung. Bei echten Websites variieren Anzahl und Schweregrad der Befunde je nach Architektur des Ziels erheblich.

⚡ 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 **Authentifizierte Injektion (SQLi/XSS)**.

Verbesserungsprioritäten

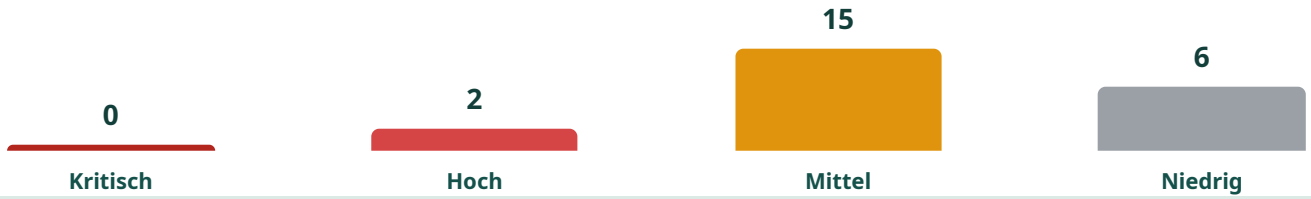
⚡ **Am dringendsten:** SQL-Injection-Indikator; Authentifizierungs-Bypass-Indikator — ' OR 1=1-- (POST /rest/user/login).

🛠 **Schnelle Erfolge (Serverkonfiguration, ~1-2 Tage):** Unzureichende DMARC-Richtlinie (fehlend oder schwach) (ornek.com); Fehlendes HSTS (ornek.com); Fehlende Content-Security-Policy (ornek.com); Fehlende Referrer-Policy (ornek.com).

🗓 **Mittelfristig / Prozess (manuelle Verifizierung oder Code/Architektur):** Schwache Session-Invalidierung (/rest/user/whoami); Unbefugter Endpunkt-Zugriff (Forced Browsing) (/rest/admin/application-configuration); Unbefugter Endpunkt-Zugriff (Forced Browsing) (/api/Users); Unbefugter Endpunkt-Zugriff (Forced Browsing) (/metrics); Unbefugter Endpunkt-Zugriff (Forced Browsing) (/rest/admin/application-version); /api/products/1; /api/users/1; Unsicherer postMessage-Origin (main.js · message handler); Sensible Daten im Client-Storage (main.js · localStorage['token']); Sensible Daten im Client-Storage (main.js · localStorage['totp_tmp_token']); Kein API-Rate-Limit (/rest/user/login); Übermäßige Datenpreisgabe (API/BOPLA) (/api/Products); Kein API-Rate-Limit (/api/BasketItems); Reflected-XSS-Indikator; Web-Cache-/sensibles Caching-Indikator (/rest/user/whoami); Fehlender CAA-Eintrag (ornek.com); Fehlende Permissions-Policy (ornek.com).

2. Befunde

2.1 Schwachstellenverteilung



In diesem Scan wurden insgesamt **23** 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
CT-2	Authentifizierungs-Bypass-Indikator — ' OR 1=1-- — POST /rest/user/login	Offen	Hoch
CT-3	Schwache Session-Invalidierung — /rest/user/whoami	Offen	Mittel
CT-4	Unbefugter Endpunkt-Zugriff (Forced Browsing) — /rest/admin/application-configuration	Offen	Mittel
CT-5	Unbefugter Endpunkt-Zugriff (Forced Browsing) — /api/Users	Offen	Mittel
CT-6	Unbefugter Endpunkt-Zugriff (Forced Browsing) — /metrics	Offen	Mittel
CT-7	Unbefugter Endpunkt-Zugriff (Forced Browsing) — /rest/admin/application-version	Offen	Mittel
CT-8	/api/products/1	Offen	Mittel
CT-9	/api/users/1	Offen	Mittel
CT-10	Unsicherer postMessage-Origin — main.js · message handler	Offen	Mittel
CT-11	Sensible Daten im Client-Storage — main.js · localStorage['token']	Offen	Mittel
CT-12	Sensible Daten im Client-Storage — main.js · localStorage['totp_tmp_token']	Offen	Mittel
CT-13	Kein API-Rate-Limit — /rest/user/login	Offen	Mittel
CT-14	Übermäßige Datenpreisgabe (API/BOPLA) — /api/Products	Offen	Mittel
CT-15	Kein API-Rate-Limit — /api/BasketItems	Offen	Mittel
CT-16	Unzureichende DMARC-Richtlinie (fehlend oder schwach) — ornek.com	Offen	Mittel
CT-17	Fehlendes HSTS — ornek.com	Offen	Mittel
CT-18	Reflected-XSS-Indikator	Offen	Niedrig
CT-19	Web-Cache-/sensibles Caching-Indikator — /rest/user/whoami	Offen	Niedrig
CT-20	Fehlender CAA-Eintrag — ornek.com	Offen	Niedrig
CT-21	Fehlende Content-Security-Policy — ornek.com	Offen	Niedrig
CT-22	Fehlende Referrer-Policy — ornek.com	Offen	Niedrig
CT-23	Fehlende Permissions-Policy — ornek.com	Offen	Niedrig

20 Kontrollbereiche wurden ausgeführt — Befunde: 2 hoch, 15 mittel, 6 niedrig

✓ **AUSGEFÜHRT — KEIN SCHWACHSTELLENNACHWEIS**

- **JWT / Token-Sicherheit · Client-Side / JS-Analyse · Eingabe- & Header-Tiefe · Konfigurations- & Preisgabetiefe · Subdomain-Takeover (Dangling DNS)**

⚠ **BEFUND VORHANDEN — DETAILS IN DEN ABSCHNITTEN UNTEN**

- **Logout / Sitzungsungültigmachung · Forced Browsing / Funktionsebenen-Berechtigung · Authentifizierte Injektion (SQLi/XSS) · Authentifiziertes IDOR (eigene Ressourcen) · Login-Bypass (SQLi-Indikator) · Client-Side statische Analyse ·**

Authentifizierungstiefe · API-Sicherheitstiefe (OWASP API Top 10) · E-Mail- & DNS-Tiefe (Anti-Spoofing) · Transportschicht-, CORS- & Sicherheits-Header-Tiefe

⚠ NICHT ANWENDBAR / AUSSERHALB DES UMFANGS — BEDEUTET NICHT „SICHER“

- **Sitzungs-Cookie-Flags · Session Fixation · Rechteausweitung (Privilege Escalation) · Mehrstufige Geschäftslogik · Sitzungssicherheitstiefe**

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

[MITTEL] CT-3 · Schwache Session-Invalidierung

Status: Offen

Betroffener Punkt: `/rest/user/whoami`

Beschreibung: Die Session wird nach dem Logout serverseitig nicht invalidiert.

Wie es erkannt wurde: Auch NACH dem Aufruf des Logouts gab der geschützte Endpunkt (/rest/user/whoami) mit DEMSELBEN Sitzungs-Token 200 zurück — die Sitzung wird serverseitig nicht ungültig gemacht.

Geschäftliche Auswirkung: Bleibt die Session nach dem Logout gültig, kann eine gestohlene/geteilte Session wiederverwendet werden; Kontoübernahme auf gemeinsam genutzten Geräten.

EMPFOHLENE BEHEBUNG

Invalidieren Sie die Session beim Logout serverseitig (Revocation/Ablauf).

Referenz: CWE-613 · OWASP A07:2021 Identification & Authentication Failures

[MITTEL] CT-4 · Unbefugter Endpunkt-Zugriff (Forced Browsing)

Status: Offen

Betroffener Punkt: `/rest/admin/application-configuration`

Beschreibung: Ein Admin-/versteckter Endpunkt ist mit einer niedrig privilegierten Session erreichbar.

Wie es erkannt wurde: Der Endpunkt /rest/admin/application-configuration gab mit der Sitzung des niedrig privilegierten Testkontos 200 und JSON-Daten zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).

Geschäftliche Auswirkung: Ein unbefugter Nutzer kann Admin-/versteckte Endpunkte erreichen; Risiko unbefugter Datenansicht/-änderung und Insider-Missbrauch.

EMPFOHLENE BEHEBUNG

Erzwingen Sie serverseitige Rollen-/Autorisierungsprüfung an jedem sensiblen Endpunkt; UI-Verbergen genügt nicht.

Referenz: CWE-284 · OWASP A01:2021 Broken Access Control

[MITTEL] CT-5 · Unbefugter Endpunkt-Zugriff (Forced Browsing)

Status: Offen

Betroffener Punkt: `/api/Users`

Beschreibung: Ein Admin-/versteckter Endpunkt ist mit einer niedrig privilegierten Session erreichbar.

Wie es erkannt wurde: Der Endpunkt /api/Users gab mit der Sitzung des niedrig privilegierten Testkontos 200 und JSON-Daten zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).

Geschäftliche Auswirkung: Ein unbefugter Nutzer kann Admin-/versteckte Endpunkte erreichen; Risiko unbefugter Datenansicht/-änderung und Insider-Missbrauch.

EMPFOHLENE BEHEBUNG

Erzwingen Sie serverseitige Rollen-/Autorisierungsprüfung an jedem sensiblen Endpunkt; UI-Verbergen genügt nicht.

Referenz: CWE-284 · OWASP A01:2021 Broken Access Control

[MITTEL] CT-6 · Unbefugter Endpunkt-Zugriff (Forced Browsing)

Status: Offen

Betroffener Punkt: `/metrics`

Beschreibung: Ein Admin-/versteckter Endpunkt ist mit einer niedrig privilegierten Session erreichbar.

Wie es erkannt wurde: Der Endpunkt /metrics gab mit der Sitzung des niedrig privilegierten Testkontos 200 und Inhalt zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).

Geschäftliche Auswirkung: Ein unbefugter Nutzer kann Admin-/versteckte Endpunkte erreichen; Risiko unbefugter Datenansicht/-änderung und Insider-Missbrauch.

EMPFOHLENE BEHEBUNG

Erzwingen Sie serverseitige Rollen-/Autorisierungsprüfung an jedem sensiblen Endpunkt; UI-Verbergen genügt nicht.

Referenz: CWE-284 · OWASP A01:2021 Broken Access Control

[MITTEL] CT-7 · Unbefugter Endpunkt-Zugriff (Forced Browsing)

Status: Offen

Betroffener Punkt: `/rest/admin/application-version`

Beschreibung: Ein Admin-/versteckter Endpunkt ist mit einer niedrig privilegierten Session erreichbar.

Wie es erkannt wurde: Der Endpunkt /rest/admin/application-version gab mit der Sitzung des niedrig privilegierten Testkontos 200 und JSON-Daten zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).

Geschäftliche Auswirkung: Ein unbefugter Nutzer kann Admin-/versteckte Endpunkte erreichen; Risiko unbefugter Datenansicht/-änderung und Insider-Missbrauch.

EMPFOHLENE BEHEBUNG

Erzwingen Sie serverseitige Rollen-/Autorisierungsprüfung an jedem sensiblen Endpunkt; UI-Verbergen genügt nicht.

Referenz: CWE-284 · OWASP A01:2021 Broken Access Control

[MITTEL] CT-10 · Unsicherer postMessage-Origin

Status: Offen

Betroffener Punkt: `main.js · message handler`

Beschreibung: Indikator für fehlende Origin-Validierung in einem postMessage-Handler.

Wie es erkannt wurde: In main.js scheint ein message-Event-Listener ohne event.origin-Verifikation zu arbeiten — Indikator für unsichere Cross-Origin-Nachrichtenverarbeitung. Die Sender-Origin sollte mit einer Allowlist verifiziert werden.

Geschäftliche Auswirkung: Unsicherer postMessage-Origin (Wildcard) kann das Lesen/Einschleusen von Daten durch andere Ursprünge ermöglichen.

EMPFOHLENE BEHEBUNG

Validieren Sie `event.origin` in `message`-Handlern gegen eine strikte Allowlist.

Referenz: CWE-346 · OWASP A05:2021 Security Misconfiguration

[MITTEL] CT-11 · Sensible Daten im Client-Storage

Status: Offen

Betroffener Punkt: `main.js · localStorage['token']`

Beschreibung: Sensible Daten (Token/Session) werden in localStorage/sessionStorage geschrieben.

Wie es erkannt wurde: In main.js localStorage.setItem('token', ...) — sensible Daten (Sitzungs-/Identitätsindikator) werden in den Client-Speicher geschrieben (Wert wird nicht angezeigt). localStorage/sessionStorage kann per XSS gelesen werden; für Sitzungsdaten sollte ein HttpOnly-Cookie bevorzugt werden.

Geschäftliche Auswirkung: Sensible Daten im Client-Storage (localStorage/sessionStorage) sind per XSS/geteiltem Gerät zugänglich; Risiko der Datenpreisgabe.

EMPFOHLENE BEHEBUNG

Bewahren Sie das Session-Token in einem `HttpOnly + Secure`-Cookie statt in localStorage auf.

Referenz: CWE-922 · OWASP A04:2021 Insecure Design

[MITTEL] CT-12 · Sensible Daten im Client-Storage

Status: Offen

Betroffener Punkt: `main.js · localStorage['totp_tmp_token']`

Beschreibung: Sensible Daten (Token/Session) werden in localStorage/sessionStorage geschrieben.

Wie es erkannt wurde: In main.js localStorage.setItem('totp_tmp_token', ...) — sensible Daten (Sitzungs-/Identitätsindikator) werden in den Client-Speicher geschrieben (Wert wird nicht angezeigt). localStorage/sessionStorage kann per XSS gelesen werden; für Sitzungsdaten sollte ein HttpOnly-Cookie bevorzugt werden.

Geschäftliche Auswirkung: Sensible Daten im Client-Storage (localStorage/sessionStorage) sind per XSS/geteiltem Gerät zugänglich; Risiko der Datenpreisgabe.

EMPFOHLENE BEHEBUNG

Bewahren Sie das Session-Token in einem `HttpOnly + Secure`-Cookie statt in localStorage auf.

Referenz: CWE-922 · OWASP A04:2021 Insecure Design

[MITTEL] CT-13 · Kein API-Rate-Limit

Status: Offen

Betroffener Punkt: `/rest/user/login`

Beschreibung: Kein Rate-Limit/Kontingent am API-Endpunkt beobachtet.

Wie es erkannt wurde: Nach 5 aufeinanderfolgenden fehlgeschlagenen Login-Versuchen wurde keine Sperre, kein 429 und keine deutliche Verzögerung beobachtet — Indikator für schwachen/fehlenden Lockout & Rate-Limit (könnte für Brute-Force offen sein). Das echte Konto wurde nicht gesperrt (ein Throwaway-Benutzer wurde verwendet).

Geschäftliche Auswirkung: Kein API-Rate-Limit; anfällig für Brute-Force, Enumeration und Ressourcenerschöpfung.

EMPFOHLENE BEHEBUNG

Fügen Sie Rate-Limit + Kontingent pro Konto+IP+Endpunkt hinzu; kostenbasierte Limits auf schweren Endpunkten.

Referenz: CWE-770 · OWASP API4:2023 Unrestricted Resource Consumption

[MITTEL] CT-14 · Übermäßige Datenpreisgabe (API/BOPLA)

Status: Offen

Betroffener Punkt: `/api/Products`

Beschreibung: Die API-Antwort gibt übermäßige/sensible Felder zurück (Passwort-Hash/Rolle/interne ID).

Wie es erkannt wurde: Die API-Antwort (/api/Products) enthält interne/übermäßige Felder, die die UI nicht benötigt (deletedAt) — interne ID-/Rollen-/Status-Felder sickern zum Client (Indikator). Beschränken Sie die Antwort auf die nur benötigten Felder (Allowlist-DTO).

Geschäftliche Auswirkung: Übermäßige Datenpreisgabe über die API (BOPLA); mehr Felder als nötig werden zurückgegeben, was das Datenleck-Risiko erhöht.

EMPFOHLENE BEHEBUNG

Begrenzen Sie Antworten mit einer Feld-Allowlist (Response-DTO); senden Sie sensible Felder niemals.

Referenz: CWE-213 · OWASP API3:2023 Broken Object Property Level Authorization

[MITTEL] CT-15 · Kein API-Rate-Limit

Status: Offen

Betroffener Punkt: `/api/BasketItems`

Beschreibung: Kein Rate-Limit/Kontingent am API-Endpunkt beobachtet.

Wie es erkannt wurde: Nach 10 aufeinanderfolgenden Anfragen an einen API-Endpunkt (/api/BasketItems) wurde kein 429 oder Rate-Limit-Header beobachtet — Indikator für uneingeschränkten Ressourcenverbrauch (API4) (könnte für Brute-Force/Scraping/DoS offen sein). Das Ziel wurde nicht ermüdet (moderater Burst). Rate-Limit + Kontingent werden empfohlen.

Geschäftliche Auswirkung: Kein API-Rate-Limit; anfällig für Brute-Force, Enumeration und Ressourcenerschöpfung.

EMPFOHLENE BEHEBUNG

Fügen Sie Rate-Limit + Kontingent pro Konto+IP+Endpunkt hinzu; kostenbasierte Limits auf schweren Endpunkten.

Referenz: CWE-770 · OWASP API4:2023 Unrestricted Resource Consumption

[MITTEL] CT-16 · Unzureichende DMARC-Richtlinie (fehlend oder schwach)

Status: Offen

Betroffener Punkt: `ornek.com`

Beschreibung: Kein DMARC; SPF/DKIM werden nicht durchgesetzt.

Wie es erkannt wurde: DMARC p=none (Org-Domain (ornek.com)) — nur Monitoring; Spoofing-E-Mails werden beim Empfänger nicht blockiert (Indikator). Schrittweise Verschärfung p=quarantine→reject empfohlen.

Geschäftliche Auswirkung: Ohne DMARC-Richtlinie werden gefälschte E-Mails nicht blockiert; Phishing- und Reputationsrisiko.

EMPFOHLENE BEHEBUNG

Veröffentlichen Sie `_dmarc` `v=DMARC1; p=quarantine` (mit Monitoring beginnen, dann auf reject erhöhen).

Referenz: CWE-290 · OWASP A07:2021 Identification & Authentication Failures

[MITTEL] CT-17 · Fehlendes HSTS

Status: Offen

Betroffener Punkt: `ornek.com`

Beschreibung: Kein HSTS; der Browser wird nicht auf HTTPS gezwungen.

Wie es erkannt wurde: In der HTTPS-Antwort kein Strict-Transport-Security — die erste Verbindung wird nicht auf HTTPS erzwungen, MITM-/Downgrade-Risiko. max-age≥15768000; includeSubDomains; preload empfohlen.

Geschäftliche Auswirkung: Ohne erzwungenes HTTPS kann der Verkehr per Man-in-the-Middle abgehört/umgeleitet werden.

EMPFOHLENE BEHEBUNG

Fügen Sie `Strict-Transport-Security: max-age=31536000; includeSubDomains` hinzu (wenn vollständig HTTPS).

Referenz: CWE-319 · OWASP A05:2021 Security Misconfiguration

[NIEDRIG] CT-18 · 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

[NIEDRIG] CT-19 · Web-Cache-/sensibles Caching-Indikator

Status: Offen

Betroffener Punkt: `/rest/user/whoami`

Beschreibung: Sensible Antworten sind cachebar oder es gibt unkeyed Header-Reflexion.

Wie es erkannt wurde: Die authentifizierte Antwort (/rest/user/whoami) enthält kein Cache-Control: no-store/no-cache/private (beobachtet: keins) — Browser/Zwischen-Proxy könnten sensible Inhalte zwischenspeichern.

Geschäftliche Auswirkung: Indikator, dass sensible Antworten cachebar sind / unkeyed Header-Reflexion; Cache-Poisoning oder Datenleck.

EMPFOHLENE BEHEBUNG

Setzen Sie `Cache-Control: no-store` auf sensible Antworten; nehmen Sie relevante Header in den Cache-Key auf.

Referenz: CWE-525 · OWASP A05:2021 Security Misconfiguration

[NIEDRIG] CT-20 · Fehlender CAA-Eintrag

Status: Offen

Betroffener Punkt: `ornek.com`

Beschreibung: Kein CAA-Eintrag; jede beliebige CA kann Zertifikate ausstellen.

Wie es erkannt wurde: CAA-Eintrag nicht beobachtet — jede beliebige CA kann für ornek.com ein Zertifikat ausstellen (Indikator für Fehlausstellungsrisiko). Mit CAA können die autorisierten CAs beschränkt werden.

Geschäftliche Auswirkung: Ohne CAA-Eintrag kann jede beliebige Zertifizierungsstelle ein Zertifikat für Ihre Domain ausstellen; erhöht das Missbrauchsrisiko.

EMPFOHLENE BEHEBUNG

Beschränken Sie autorisierte CAs mit einem `CAA`-Eintrag (z. B. `0 issue "letsencrypt.org"`).

Referenz: CWE-295 · OWASP A05:2021 Security Misconfiguration

[NIEDRIG] CT-21 · Fehlende Content-Security-Policy

Status: Offen

Betroffener Punkt: `ornek.com`

Beschreibung: Keine Content-Security-Policy; keine browserseitige XSS-Minderung.

Wie es erkannt wurde: Content-Security-Policy-Header fehlt — keine browserseitige Minderungsschicht gegen injizierte Skripte (Härtungslücke, niedrig).

Geschäftliche Auswirkung: Keine browserseitige Minderung gegen Content-Injection/XSS; injizierte Skripte können ausgeführt werden.

EMPFOHLENE BEHEBUNG

Definieren Sie eine strikte CSP (`default-src 'self'`) und setzen Sie Drittanbieter-Quellen auf eine Allowlist.

Referenz: CWE-693 · OWASP A05:2021 Security Misconfiguration

[NIEDRIG] CT-22 · Fehlende Referrer-Policy

Status: Offen

Betroffener Punkt: `ornek.com`

Beschreibung: Keine Referrer-Policy; die vollständige URL kann an externe Links durchsickern.

Wie es erkannt wurde: Referrer-Policy-Header fehlt — Adress-/Sitzungsinformationen könnten zu externen Links durchsickern (informativ). Referrer-Policy: strict-origin-when-cross-origin empfohlen.

Geschäftliche Auswirkung: Adress-/Session-Informationen können an externe Links durchsickern; geringfügige Informationspreisgabe.

EMPFOHLENE BEHEBUNG

Fügen Sie `Referrer-Policy: strict-origin-when-cross-origin` hinzu.

Referenz: CWE-200 · OWASP A05:2021 Security Misconfiguration

[NIEDRIG] CT-23 · Fehlende Permissions-Policy

Status: Offen

Betroffener Punkt: `ornek.com`

Beschreibung: Kein Permissions-Policy-Header; der Browser-Feature-Zugriff ist unbeschränkt.

Wie es erkannt wurde: Permissions-Policy-Header fehlt — der Browser-Funktionszugriff (Kamera/Mikrofon/Standort) wird nicht eingeschränkt (informative Härtung).

Geschäftliche Auswirkung: Fehlende Permissions-Policy; Browser-Funktionen (Kamera/Mikrofon/Geolokalisierung) sind nicht eingeschränkt.

EMPFOHLENE BEHEBUNG

Fügen Sie eine `Permissions-Policy` hinzu, die ungenutzte Funktionen deaktiviert (z. B. `geolocation=(), camera=(), microphone=()`).

Referenz: CWE-693 · OWASP A05:2021 Security Misconfiguration

3. Kontrollübersicht & Methodik

Bewertungsübersicht (Authentifiziert)

Dieser Scan wurde im AUTHENTIFIZIERTEN (eingeloggten) Kontext mit der Sitzung des bereitgestellten TEST-Kontos durchgeführt. 20 authentifizierte Kontrollen wurden bewertet; insgesamt **311** Anfragen. Das höchste Risiko liegt im Bereich **Authentifizierte Injektion (SQLi/XSS)** (nachfolgend im Detail).

Das Passwort wurde in keiner Phase nach außen/an einen Drittdienst gesendet; das Backend führte einen deterministischen Login durch und nutzte nur die Sitzung (Cookie/Token).

KONTROLLÜBERSICHT

Kontrolle	Ergebnis	Zuversicht
Sitzungs-Cookie-Flags	Kein anwendbarer Einstiegspunkt (Außerhalb des Umfangs)	Außerhalb des Umfangs
Session Fixation	Kein anwendbarer Einstiegspunkt (Außerhalb des Umfangs)	Außerhalb des Umfangs
Logout / Sitzungsungültigmachung	⚠ Eingeschränkter Indikator	Mittel
Forced Browsing / Funktionsebenen-Berechtigung	⚠ Eingeschränkter Indikator	Mittel
Authentifizierte Injektion (SQLi/XSS)	⚠ Schwachstellen-Indikator	Hoch
Authentifiziertes IDOR (eigene Ressourcen)	⚠ Eingeschränkter Indikator	Mittel
Rechteausweitung (Privilege Escalation)	Nicht untersuchbar — keine sicher testbare Oberfläche	Außerhalb des Umfangs
Mehrstufige Geschäftslogik	Nicht untersuchbar — keine sicher testbare Oberfläche	Außerhalb des Umfangs
JWT / Token-Sicherheit	✓ Kein Schwachstellennachweis	Hoch
Login-Bypass (SQLi-Indikator)	⚠ Schwachstellen-Indikator	Hoch

Kontrolle	Ergebnis	Zuversicht
Client-Side / JS-Analyse	✓ Kein Schwachstellennachweis	Hoch
Client-Side statische Analyse	⚠ Eingeschränkter Indikator	Mittel
Authentifizierungstiefe	⚠ Eingeschränkter Indikator	Hoch
Sitzungssicherheitstiefe	Kein anwendbarer Einstiegspunkt (Außerhalb des Umfangs)	Außerhalb des Umfangs
Eingabe- & Header-Tiefe	✓ Kein Schwachstellennachweis	Mittel
Konfigurations- & Preisgabetiefe	✓ Kein Schwachstellennachweis	Hoch
API-Sicherheitstiefe (OWASP API Top 10)	⚠ Eingeschränkter Indikator	Mittel
E-Mail- & DNS-Tiefe (Anti-Spoofing)	⚠ Eingeschränkter Indikator	Mittel
Transportschicht-, CORS- & Sicherheits-Header-Tiefe	⚠ Eingeschränkter Indikator	Mittel
Subdomain-Takeover (Dangling DNS)	✓ Kein Schwachstellennachweis	Mittel

> Die Zuversicht wird nur für Kontrollen angezeigt, die tatsächlich anwendbar waren (bei denen ein Login/Cookie/Endpunkt gefunden wurde); nicht anwendbare Kontrollen sind **Außerhalb des Umfangs** (z. B. Cookie-Flags/Fixation bei einer Sitzung, die Token statt Cookie verwendet).

Sitzungs-Cookie-Flags

WAS GEPRÜFT WURDE

- Von den nach dem Login beobachteten Cookies wurden nur die echten **Server-Sitzungs-Cookies** bewertet.
- Für jedes Sitzungs-Cookie wurden die Sicherheits-Flags **Secure / HttpOnly / SameSite** geprüft.
- Analytics-/Drittanbieter-Cookies (_ga, _fbp, _clck usw.) gelten nicht als Sitzungs-Cookies (HttpOnly ist bei diesen unmöglich) — sie werden nicht in die Bewertung einbezogen.

BEFUNDE

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

> Es wurde kein serverseitiges Sitzungs-Cookie beobachtet (die Sitzung wird per Bearer/Token übertragen) — die Analyse der Set-Cookie-Sicherheits-Flags ist für dieses Ziel **außerhalb des Umfangs**.

> **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. Dieser Abschnitt wurde im authentifizierten (eingeloggten) Kontext mit der Sitzung des von Ihnen bereitgestellten TEST-Kontos ausgeführt; der Abschluss von Zahlungs-/Kontostatusänderungen ist auf Codeebene gesperrt.

Session Fixation

WAS GEPRÜFT WURDE

- Der VOR dem Login (nicht authentifiziert) von der Startseite erhaltene Sitzungs-Cookie-Wert wurde beobachtet.
- Er wurde mit dem Sitzungs-Cookie-Wert NACH dem Login **verglichen** (sind sie gleich, erneuert der Server die Sitzung nicht).
- Nur ein einziges GET; kein Zustand wurde verändert.

BEFUNDE

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

> Die Sitzung ist nicht cookie-basiert (Bearer/Token) — die Session-Fixation-Analyse (Cookie-Erneuerung) ist für dieses Ziel **außerhalb des Umfangs**.

Logout / Sitzungsungültigmachung

WAS GEPRÜFT WURDE

- Ein geschützter Endpunkt (whoami/Profil), der mit der Sitzung 200 zurückgibt, wurde festgestellt.
- Ein per GET aufrufbarer Logout-Endpunkt wurde versucht (**kein POST** — kein Zustand verändert).
- Es wurde geprüft, ob NACH dem Logout mit DEMSELBEN Token weiterhin auf den geschützten Endpunkt zugegriffen werden kann.

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
/rest/user/whoami	Token-Wiederverwendung nach Logout	Auch NACH dem Aufruf des Logouts gab der geschützte Endpunkt (/rest/user/whoami) mit DEMSELBEN Sitzungs-Token 200 zurück — die Sitzung wird serverseitig nicht ungültig gemacht.	Mittel	keins	Mittel

Forced Browsing / Funktionsebenen-Berechtigung

WAS GEPRÜFT WURDE

- An gängige Admin-/Verwaltungs-Endpunkte wurde mit der Sitzung des VORHANDENEN (vermutlich niedrig privilegierten) Testkontos eine **GET**-Anfrage gesendet.
- Um False Positives zu vermeiden, galten nur 200er als Befund, die einen **ANDEREN** Inhalt/JSON als die Startseiten-Shell zurückgaben.
- Ein einzelner Versuch, GET-only, dem Schutzschalter unterliegend.

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
/rest/admin/application-configuration	Forced Browsing mit niedrig privilegierter Sitzung (GET)	Der Endpunkt `/rest/admin/application-configuration` gab mit der Sitzung des niedrig privilegierten Testkontos **200** und JSON-Daten zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter	Mittel	keins	Mittel

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
		Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).			
/api/Users	Forced Browsing mit niedrig privilegierter Sitzung (GET)	Der Endpunkt `/api/Users` gab mit der Sitzung des niedrig privilegierten Testkontos **200** und JSON-Daten zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).	Mittel	keins	Mittel
/metrics	Forced Browsing mit niedrig privilegierter Sitzung (GET)	Der Endpunkt `/metrics` gab mit der Sitzung des niedrig privilegierten Testkontos **200** und Inhalt zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).	Niedrig	keins	Mittel
/rest/admin/application-version	Forced Browsing mit niedrig privilegierter Sitzung (GET)	Der Endpunkt `/rest/admin/application-version` gab mit der Sitzung des niedrig privilegierten Testkontos **200** und JSON-Daten zurück — Indikator für fehlende Funktionsebenen-Autorisierung (möglicher unbefugter Verwaltungszugriff) (die zurückgegebenen Daten werden im Bericht nicht angezeigt).	Mittel	keins	Mittel

Authentifizierte Injektion (SQLi/XSS)

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

Authentifiziertes IDOR (eigene Ressourcen)

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 mit der authentifizierten Sitzung eine **GET**-Anfrage 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
/api/users/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)

Dieser Abschnitt wurde mit der **Sitzung (eingeloggt)** des von Ihnen bereitgestellten TEST-Kontos ausgeführt und testet den unbefugten Zugriff auf die **eigenen** aufzählbaren Ressourcen des Kontos. Der **Cross-Account**-Test (Zugriff auf die Daten eines anderen Benutzers) liegt außerhalb des Umfangs dieser Version (er erfordert zwei separate Konten). Das Fehlen eines Befunds beweist nicht, dass es in allen authentifizierten Abläufen kein IDOR gibt.

> 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.

Rechteausweitung (Privilege Escalation)

WAS GEPRÜFT WURDE

- Auf der entdeckten authentifizierten Oberfläche wurde **deterministisch** nach einem Formular/einer API mit Berechtigungsfeld (Registrierungs-/Profil-/Einstellungstyp) gesucht.

- Wurde eine sicher testbare Oberfläche gefunden, wurde mit einer **sicheren, authentifiziert-leichten** Funktion eine EINZIGE beobachtende Mass-Assignment-Probe (`role/isAdmin` -Zusatzfeld) angewendet.

⚠ Eine echte Ausweitung wurde NICHT ABGESCHLOSSEN; es wurde nicht erneut mit erhöhten Rechten eingeloggt; die Sitzung wurde nicht verlassen; auf konto-verändernde/checkout-Ziele wurde **nicht geschrieben** (Blocklist auf Codeebene).

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); eine sichere Verifikation erfordert einen manuellen Test.

> Für die Rechteausweitung wurde auf diesem Ziel kein SICHER testbares (nicht VERBOTENES; Registrierungs-/Profil-/Einstellungstyp) Formular/kein solcher Endpunkt gefunden — diese Kontrolle ist **Nicht untersuchbar** (keine geeignete authentifizierte Oberfläche für die deterministische Probe). Hinweis: manche Schwachstellen (z. B. verstecktes `role` -Akzeptieren in der API) sind in der UI/im DOM nicht sichtbar; ein sicheres Ergebnis erfordert einen manuellen Test.

Mehrstufige Geschäftslogik

WAS GEPRÜFT WURDE

- Es wurde **deterministisch** nach einem mehrstufigen Ablauf/Preis-Coupon-Feld gesucht; das Backend führte NUR **GET-Beobachtung** durch.
- Ein clientseitig veränderbares verstecktes Preis-/Mengen-/Coupon-Feld + ein ohne Vorbedingung erreichbarer „Bestätigungs"-Schritt wurden beobachtet.

⚠ Nur bis zum Warenkorb/Formular; **Zahlung/Checkout wurde NICHT ABGESCHLOSSEN** (Blocklist auf Codeebene); keine Ressource wurde verbraucht.

BEFUNDE

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

> Geschäftslogik-Schwachstellen sind kontextspezifisch; diese Kontrolle ist auf Oberflächen-/Indikatorebene.

> Diese Kontrolle wurde **deterministisch** ausgeführt (beobachtendes Schritt-Überspringen + clientseitig veränderbares Preis-/Mengen-/Coupon-Feld). Die KI-otonom analiz-Schicht ist **standardmäßig deaktiviert** (da sie im Experiment keine zusätzlichen verifizierten Nachweise erzeugte).

JWT / Token-Sicherheit

WAS GEPRÜFT WURDE

- Trägt die Sitzung ein **JWT-Bearer**, wurde der Token OFFLINE dekodiert und analysiert: Signaturalgorithmus (**alg=none / unsigniert**), ob das Signaturgeheimnis **schwach/verbreitet** ist (OFFLINE-Verifikation mit verbreiteten Geheimnissen) und **sensible/übermäßige Claims** im Token-Body (Passwort/Geheimnis, Rolle/Berechtigung).
- Zusätzlich eine EINZIGE, harmlose Beobachtung: wird ein unsigniert (alg=none) gefälschter Token an einem geschützten Endpunkt AKZEPTIERT (nur Beobachtung; der Zugriff wurde nicht genutzt).

⚠ KEINE echte Ausnutzung — es wurde keine Token-Übernahme/Rechteausweitung durchgeführt; es wurde nur ein Sicherheitsindikator berichtet.

BEFUNDE

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

> Die Indikatoren für schwaches Geheimnis und alg=none-AKZEPTANZ sind eindeutig (hohe Zuversicht); Claim-Beobachtungen sind informativ.

> Der Token hat keinen Ablauf-(exp-)Claim — ein unbefristeter Token birgt im Falle eines Diebstahls das Risiko dauerhaften Zugriffs. Für die alg=none-Akzeptanzbeobachtung wurde kein verifizierbarer geschützter Lese-Endpunkt (whoami/Profil) gefunden — diese Beobachtung wurde übersprungen. Es wurde kein JWT-/Token-Sicherheitsindikator gefunden (kein alg=none, kein schwaches Geheimnis verifiziert, kein sensibler Claim).

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; eine EINZIGE, 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 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).	Hoch	keins	Hoch

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

Client-Side / JS-Analyse

WAS GEPRÜFT WURDE

- Die von der Seite geladenen JS-Bundles (script-src + inline) wurden abgerufen und **statisch** analysiert — KEINE aktive Ausnutzung, nur herunterladen + lesen.
- **A) Geheimnis-Scan:** es wurde nach ECHTEN Geheimnissen wie privaten Schlüsseln, AWS/GCP-Anmeldedaten, Stripe **sk_live_**, GitHub/GitLab/Slack-Token, DB-Verbindungszeichenfolge gesucht (Werte REDIGIERT).
- **Public-by-design-Unterscheidung:** Firebase apiKey, GTM/GA-Mess-ID, Google-Maps-Browser-Key, Stripe **pk_**, reCAPTCHA-Site-Key sind gestalterisch öffentlich → gelten NICHT als „Preisgabe/Geheimnis", sondern werden nur informativ aufgeführt.
- **B) Source-Map-Preisgabe:** `//# sourceMappingURL`-Kommentare + `.js.map`-Kandidaten wurden versucht; eine erreichbare `.map` → Leck des Original-Quellcodes/-Baums.
- **C) Bekannt-verwundbare Bibliothek:** die geladenen Bibliotheken + ihre VERSIONEN wurden festgestellt und KONSERVATIV mit bekannten CVEs abgeglichen; konnte die Version nicht sicher gelesen werden, erfolgte KEIN CVE-Abgleich (Sprache „Patch-Bestätigung erforderlich"; es wird nicht „ausnutzbar" gesagt).

BEFUNDE

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

> Alles ist passiv/statisch; Bibliotheksbefunde sind versionsbasierte Indikatoren (Ihre Distribution könnte gepatcht/gebackportet sein — Bestätigung empfohlen).

> Gescannt: **3** Same-Origin-JS-Dateien + **1** Inline-Skript; **0** Bibliotheken erkannt; **3** Source-Map-Kandidaten versucht. (0 externe/CDN-Skripte auf Version untersucht.)

Client-Side statische Analyse

WAS GEPRÜFT WURDE

- Auf demselben JS-/HTML-Corpus wurden **statisch** (KEINE aktive Ausnutzung) 6 clientseitige Kontrollen durchgeführt.
- **1) DOM-based XSS (Indikator):** stehen ein gefährlicher Sink (innerHTML/document.write/eval/.html()/location=) und eine benutzergesteuerte Quelle (location.hash/search, referrer, window.name) im SELBEN Ausdruck — **KEIN nachgewiesenes XSS**, ein gering zuverlässiger Indikator (dynamische Verifikation erforderlich).
- **2) postMessage:** ob in den `message`-Event-Listenern eine **event.origin-Verifikation** vorhanden ist.
- **3) Browser Storage:** Schreiben **sensibler Daten** (Token/JWT/Sitzung) in localStorage/sessionStorage (statisch; Wert REDIGIERT).
- **4) Fehlende SRI:** ob externe Skripte/Styles ein `integrity`-Attribut haben (Lieferkette).
- **5) Reverse Tabnabbing:** ob `target="_blank"`-Links `rel="noopener/noreferrer"` haben.
- **6) Open Redirect:** eine **einzige sichere Probe** an die im HTML BEOBACHTETEN Weiterleitungsparameter — eine harmlose externe URL gesendet und die Location beobachtet; **der Redirect wurde NICHT VERFOLGT**.

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
main.js · message handler	Web-Messaging-(postMessage)-Origin-Verifikationsanalyse	In `main.js` scheint ein `message`-Event-Listener **ohne event.origin-Verifikation** zu arbeiten — Indikator für unsichere Cross-Origin-Nachrichtenverarbeitung. Die Sender-Origin sollte mit einer Allowlist verifiziert werden.	Mittel	keins	Mittel
main.js · localStorage['toke	statische Browser-	In `main.js` `localStorage.setItem('token', …)` — sensible	Mittel	keins	Mittel

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
n']	Storage-Analyse sensibler Daten	Daten (Sitzungs-/Identitätsindikator) werden in den Client-Speicher geschrieben (Wert wird nicht angezeigt). localStorage/sessionStorage kann per XSS gelesen werden; für Sitzungsdaten sollte ein HttpOnly-Cookie bevorzugt werden.			
main.js · localStorage['totp_tmp_token']	statische Browser-Storage-Analyse sensibler Daten	In `main.js` `localStorage.setItem('totp_tmp_token', …)` — sensible Daten (Sitzungs-/Identitätsindikator) werden in den Client-Speicher geschrieben (Wert wird nicht angezeigt). localStorage/sessionStorage kann per XSS gelesen werden; für Sitzungsdaten sollte ein HttpOnly-Cookie bevorzugt werden.	Mittel	keins	Mittel

DOM-XSS-Befunde sind STATISCHE Indikatoren (hohes False-Positive-Potenzial; dynamische Verifikation erforderlich). Die übrigen sind deterministische Beobachtungen.

Statisch geparst: **3** Same-Origin-JS + **1** Inline-Skript; **0** externe Ressourcen auf SRI, **0** `target=_blank`-Links auf Tabnabbing geprüft; **0** Redirect-Parameter mit sicherer Probe versucht (Redirect NICHT VERFOLGT). DOM-XSS-Übereinstimmungen sind STATISCHE **Indikatoren** (kein nachgewiesenes XSS); sie tragen ein hohes False-Positive-Potenzial und erfordern dynamische Verifikation. Im HTML wurde kein beobachteter Weiterleitungs-(redirect-)Parameter gefunden — die Open-Redirect-Probe ist **außerhalb des Umfangs**.

Authentifizierungstiefe

WAS GEPRÜFT WURDE

- 9 Authentifizierungstiefe-Kontrollen (WSTG-ATHN/IDNT) — **sicher, geringes Volumen**; KEIN Brute-Force/DoS.
- **1) Konto-Enumeration:** Antwortunterschied bei gültigem (Testkonto) vs. ungültigem (zufälligem) Benutzer — je ein FEHLGESCHLAGENER Versuch; erstellt kein Konto.
- **2) Standard-Anmeldedaten:** kleine feste Liste (admin/admin usw.) — nur fehlgeschlagener Login; bei Akzeptanz wird die Sitzung NICHT GENUTZT.
- **3) Schwacher Lockout/Rate-Limit:** einige Fehlversuche mit einem THROWAWAY-(zufälligen)-Benutzer — **das echte Konto wird NIEMALS gesperrt**.
- **4) Passwort-Reset:** Mechanismus-Beobachtung (Sicherheitsfrage/Token) — **es wird KEINE echte Reset-E-Mail AUSGELÖST** (nicht existierende E-Mail).
- **5) Passwort-/Registrierungsrichtlinie:** nur clientseitige Beobachtung — **es wird KEINE echte Registrierung DURCHGEFÜHRT**.
- **6) „Angemeldet bleiben"-Cookie · 7) Cache authentifizierter Seiten (Cache-Control) · 8) MFA-Vorhandensein (informativ) · 9) Anmeldedaten über unverschlüsselten Kanal (HTTP).**

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
/rest/user/login	Kontosperr- / Login-Ratenbegrenzungsbeobachtung (mit Throwaway-Benutzer)	Nach 5 aufeinanderfolgenden fehlgeschlagenen Login-Versuchen wurde keine Sperre, kein 429 und keine deutliche Verzögerung beobachtet — **Indikator für schwachen/fehlenden Lockout & Rate-Limit** (könnte für Brute-Force offen sein). Das echte Konto wurde nicht gesperrt (ein Throwaway-Benutzer wurde verwendet).	Mittel	keins	Mittel
/rest/user/whoami	Beobachtung des Cache-Headers der authentifizierten Antwort	Die authentifizierte Antwort (`/rest/user/whoami`) enthält kein `Cache-Control: no-store/no-cache/private` (beobachtet: `keins`) — Browser/Zwischen-Proxy könnten sensible Inhalte zwischenspeichern.	Mittel	keins	Niedrig

> Enumeration-/Lockout-/Reset-Befunde sind „Indikatoren" (Verifikation erforderlich). Sicherheit: das echte Konto wurde nicht gesperrt, es ging keine Reset-E-Mail raus, es wurde keine Registrierung durchgeführt, und selbst wenn ein erfolgreicher Login gefunden worden wäre, wurde die Sitzung nicht genutzt.

> Der Passwort-Reset-Endpunkt wurde beobachtet; Token/Mechanismus konnte nicht statisch verifiziert werden (keine echte E-Mail ausgelöst) — für diesen Abschnitt eingeschränkt. Auf der Startseite wurde kein clientseitiges Passwortfeld beobachtet (könnte SPA/separate Seite sein) — die Passwortrichtlinie wurde statisch nur eingeschränkt untersucht. Die Sitzung ist Bearer-/Token-basiert (cookie-basiertes persistentes „Angemeldet bleiben"-Cookie außerhalb des Umfangs). Im Login-Ablauf wurde kein MFA-/2FA-Indikator beobachtet (informativ; ein zweiter Faktor wird empfohlen). Versucht: **16** sichere Auth-Proben (Standard-Anmeldedaten nur fehlgeschlagener Login; Lockout mit THROWAWAY-Benutzer; Reset mit nicht existierender E-Mail; KEINE Registrierung durchgeführt; echtes Konto nicht gesperrt).

Sitzungssicherheitstiefe

WAS GEPRÜFT WURDE

- FALLS ein ECHTES serverseitiges Sitzungs-Cookie (Set-Cookie session) vorhanden ist, 5 Sitzungsmanagement-Kontrollen — alle **read-only/beobachtend** (KEINE zustandsverändernde Übermittlung / kein echter CSRF-Angriff). Bei einem Bearer-/JWT-/token-basierten Ziel ist dieser Abschnitt **außerhalb des Umfangs**.
- **1) CSRF (SESS-05):** SameSite im Sitzungs-Cookie + Beobachtung des Anti-CSRF-Tokens in einem zustandsverändernden POST-Formular (statisch; Formular NICHT ABGESENDET).
- **2) Session-ID-Entropie (SESS-01):** strukturelle Analyse der Sitzungs-ID (Länge/Charset/Entropie) — KEIN Brute; bei JWT wird es einem separaten Abschnitt überlassen.
- **3) Sitzung in der URL (SESS-04):** ob die Session-ID in URL/Query (jsessionid/sid usw.) preisgegeben wird.
- **4) Timeout / gleichzeitige Sitzung (SESS-07/11):** beobachtender Indikator (ein sicherer Test ist manuell).
- **5) Cookie-Prefix (SESS-02):** ob im Sitzungs-Cookie der __Host-/__Secure--Prefix fehlt.

BEFUNDE

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

> Die Befunde sind „Indikatoren" (kein echter CSRF-Angriff/Brute durchgeführt; Token/Sitzung REDIGIERT). Ist kein Server-Sitzungs-Cookie vorhanden, sind die Kontrollen außerhalb des Umfangs (Token-/JWT-Sicherheit in separatem Abschnitt).

> Es wurde kein serverseitiges Sitzungs-Cookie (Set-Cookie session) beobachtet — die Sitzung wird per **Bearer/JWT-Token** übertragen. Die Sitzungs-COOKIE-Sicherheitskontrollen (CSRF SameSite / Cookie-Prefix / Session-ID-Entropie / Sitzung-in-URL) sind für dieses Ziel **außerhalb des Umfangs**. Token-/JWT-Sicherheit wird im separaten Abschnitt **JWT / Token-Sicherheit** behandelt.

Eingabe- & Header-Tiefe

WAS GEPRÜFT WURDE

- Sichere/read-only Eingabe- & Header-Kontrollen (nur GET/OPTIONS/TRACE — KEINE zustandsverändernde/destruktive Anfrage).
- **D1 Host-Header-Injection:** ein gefälschter `X-Forwarded-Host` wurde gesendet und die Reflexion in der Antwort beobachtet (Indikator).
- **D2 HTTP Parameter Pollution:** derselbe Parameter wurde wiederholt und der Verarbeitungsunterschied beobachtet (beobachtend).
- **D3 HTTP-Methoden / TRACE:** Entdeckung erlaubter Methoden per OPTIONS + TRACE-Beobachtung — **PUT/DELETE NICHT GESENDET**.
- **D4 Mixed Content:** statische Erkennung von HTTP-Ressourcen (script/img/iframe) auf einer HTTPS-Seite.
- **D5 Stored-XSS-Einstiegspunkt:** KANDIDATENfelder, die in einen persistenten Kontext reflektiert werden könnten, wurden markiert — **es wurden keine Daten GESENDET/gespeichert** (geringe Zuversicht, dynamische Verifikation erforderlich).

BEFUNDE

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

> Host-Injection/HPP/D5 sind „Indikatoren/Kandidaten" (kein echter Angriff/keine Übermittlung durchgeführt). TRACE/Mixed-Content sind deterministische Beobachtungen.

> Es wurde kein parametrisierter Same-Origin-Endpunkt beobachtet — keine testbare Oberfläche für HPP. Versucht: **4** sichere Eingabe-/Header-Proben (nur GET/OPTIONS/TRACE — PUT/DELETE NICHT GESENDET; Stored-XSS nur Kandidat, keine Daten ERSTELLT). Mixed Content: 0 HTTP-Ressourcen.

Konfigurations- & Preisgabetiefe

WAS GEPRÜFT WURDE

- Sichere GET + statische Konfigurations-/Preisgabe-Kontrollen. Erratene Pfade, die eine **SPA-Catch-all-200 (Startseiten-Shell)** zurückgeben, gelten NICHT als ECHT — nur ein tatsächlich erreichbarer, UNTERSCHEIDBARER Antwort erzeugt einen Befund (Spalte-0-Provenance).
- **E1 Backup-/Altdatei:** `.env/.bak/.sql/.git/config` usw. sichere GET (Shell-200 aussortiert).
- **E2 Admin-Interface:** ob gängige Verwaltungspfade von außen erreichbar sind (gleiche Provenance).
- **E3 Cloud Storage:** öffentliche S3/GCS/Azure-Bucket-Referenz in HTML/JS + ob sie **auflistbar** ist.
- **E4 Cache- / Poisoning-Indikator:** Cache-Control/Vary + Reflexion ungekeyter Header (KEIN Poisoning DURCHGEFÜHRT).
- **E5 Kommentar- & Metadaten-Leck:** Dev-Kommentar / interne IP-Hostname / Server-Dateipfad (statisch).

BEFUNDE

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

> Backup-/Admin-Befunde werden nur für TATSÄCHLICH erreichbare, NICHT-SPA-Shell-Antworten erzeugt. Cache-/Host-Indikatoren sind „Indikatoren"; Inhalt/sensible Daten REDIGIERT.

> Versucht: **14** Backup-/Altdateien + **11** Admin-Pfade (SPA-Catch-all-Shell-200er AUSSORTIERT — nicht echte/unterscheidbare Antworten galten nicht als Befund); **0** Cloud-Storage-Referenzen; Kommentare/Metadaten statisch gescannt.

API-Sicherheitstiefe (OWASP API Top 10)

WAS GEPRÜFT WURDE

- Auf den aus Same-Origin-JS/HTML entdeckten, JSON zurückgebenden (NICHT-SPA-Catch-all-Shell) ECHTEN API-Endpunkten 5 Kontrollen — alle read-only. Gibt es keine echte API (Firebase/Client-SDK/SPA), ist dieser Abschnitt **außerhalb des Umfangs**.
- **F1 BOLA/BFLA (API1/API5):** Objekt-/Funktionsebenen-Berechtigung — in den Abschnitten **Authentifiziertes IDOR** + **Forced Browsing** bewertet (um Doppelbefunde zu vermeiden, hier nicht erneut geprobt).
- **F2 Übermäßige Datenpreisgabe / BOPLA (API3):** sensibles/übermäßiges Feld in der API-Antwort (Passwort-Hash/Rolle/interne ID) — Wert REDIGIERT.
- **F3 Rate-Limit (API4):** ob nach einem MODERATEN Burst (KEIN DoS) ein 429/Rate-Limit-Header erscheint — das Ziel wurde nicht ermüdet.
- **F4 Shadow/deprecated Version (API9):** ob Versions-Endpunkte wie /v1,/v2,/api/v1 erreichbar sind (SPA-Shell aussortiert).
- **F5 GraphQL-Introspection (APIT-99):** falls ein GraphQL-Endpunkt vorhanden ist, ob die Introspection offen ist — **read-only Abfrage; KEINE Mutation**.

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
/api/Products	übermäßige Datenpreisgabe / BOPLA (API3) — Beobachtung der Antwortfelder	Die API-Antwort (`/api/Products`) enthält interne/übermäßige Felder, die die UI nicht benötigt (`deletedAt`) — interne ID-/Rollen-/Status-Felder sickern zum Client (Indikator). Beschränken Sie die Antwort auf die nur benötigten Felder (Allowlist-DTO).	Mittel	keins	Mittel
/api/BasketItems	API-Rate-Limit / Beobachtung uneingeschränkten Verbrauchs (moderater Burst — kein DoS)	Nach 10 aufeinanderfolgenden Anfragen an einen API-Endpunkt (`/api/BasketItems`) wurde **kein 429 oder Rate-Limit-Header beobachtet** — Indikator für uneingeschränkten Ressourcenverbrauch (API4) (könnte für Brute-Force/Scraping/DoS offen sein). Das Ziel wurde nicht ermüdet (moderater Burst). Rate-Limit + Kontingent werden empfohlen.	Mittel	keins	Mittel

> Die Befunde sind „Indikatoren"; nur an TATSÄCHLICH beobachteten (JSON, nicht-SPA-Shell) Endpunkten. Sensible Daten REDIGIERT; keine Mutation/Datenänderung/DoS durchgeführt.

BOLA/BFLA (OWASP API1/API5 — Objekt-/Funktionsebenen-Berechtigung) wurde in den Abschnitten **Authentifiziertes IDOR** und **Forced Browsing** bewertet (um Doppelbefunde zu vermeiden, hier nicht erneut geprobt). Entdeckte echte API-Endpunkte: **8** (Same-Origin, JSON, nicht-SPA-Shell). Versucht: **38** sichere Proben (nur GET + read-only Introspection; KEINE Mutation/Datenänderung/DoS).

E-Mail- & DNS-Tiefe (Anti-Spoofing)

WAS GEPRÜFT WURDE

- Nur aus der ECHTEN Org-Domain gelesene DNS-Einträge; alle PASSIVES DNS/TXT + ein sicheres MTA-STS-GET (kein Angriff/keine Zustandsänderung). Für die Subdomain wird DMARC auf Ebene der **organisatorischen Domain** bewertet.
- **G1 DMARC** Policy-Stärke (p=none/quarantine/reject, pct, sp Subdomain, adkim/aspf, rua).
- **G2 SPF** Tiefe (-all/~all/?all/+all + Anzahl der Top-Level-DNS-Lookups, RFC 7208).
- **G3 DKIM** Entdeckung verbreiteter Selektoren (default/google/selector1…) + Schlüssellänge (~1024/2048) + Widerruf (p= leer).
- **G4 MTA-STS** (_mta-sts TXT + Policy: enforce/testing/none) · **G5 TLS-RPT** Reporting.
- **G6 DNSSEC** (Signatur/Validierung — AD-Flag, nur Beobachtung) · **G7 CAA** (Zertifikatsausstellungs-Beschränkung) + **BIMI** (informativ).

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
ornek.com	passive DNS/TXT-Beobachtung	DMARC **p=none** (Org-Domain (ornek.com)) — nur Monitoring; Spoofing-E-Mails werden beim Empfänger nicht blockiert (Indikator). Schrittweise Verschärfung `p=quarantine` → `reject` empfohlen.	Hoch	keins	Mittel
ornek.com	passive DNS/TXT-Beobachtung	CAA-Eintrag **nicht beobachtet** — jede beliebige CA kann für `ornek.com` ein Zertifikat ausstellen (Indikator für Fehlausstellungsrisiko). Mit CAA können die autorisierten CAs beschränkt werden.	Hoch	keins	Niedrig

Nachweis = der tatsächlich von DNS zurückgegebene öffentliche Eintrag (keine Redaktion nötig). „Indikator"-Sprache; bei einer nicht E-Mail-versendenden Domain (kein MX) wird der Schweregrad fehlender DMARC/SPF gesenkt. Keine erfundenen/geratenen Einträge.

DMARC Org-Domain (ornek.com): `v=DMARC1; p=none; rua=mailto:rua@dmarc.brevo.com` SPF: `v=spf1 include:spf.improvmx.com ~all` SPF **~all** (Softfail) — akzeptabel; für sicheren Schutz kann `-all` erwogen werden. DKIM: bei verbreiteten Selektoren (default/google/selector1…) wurde **kein** Eintrag beobachtet — dies ist KEIN sicheres Fehlen; möglicherweise wird ein eigener Selektor verwendet (ehrlich). MTA-STS: `_mta-sts` TXT nicht beobachtet — kein SMTP-MITM/Downgrade-Schutz (niedriger/Reife-Indikator). TLS-RPT (`_smtp._tls` TXT) nicht beobachtet — SMTP-TLS-Zustellprobleme werden nicht gemeldet (informativ). DNSSEC: die Domain ist **signiert und validiert** (AD-Flag) — die DNS-Integrität ist geschützt (positiv). BIMI (`default._bimi` TXT) nicht beobachtet — informativ, keine Sicherheitsauswirkung. Abgefragte Org-Domain: **ornek.com** (Ziel-Subdomain: ornek.com). Insgesamt **20** passive DNS-Abfragen + höchstens 1 sicheres MTA-STS-GET. KEIN Angriff/keine Zustandsänderung/kein Resolver-Angriff.

Transportschicht-, CORS- & Sicherheits-Header-Tiefe

WAS GEPRÜFT WURDE

- Alles read-only/passive Beobachtung — KEINE Datenänderung, kein DoS, keine Cipher-Ausnutzung.
- **H1 CORS:** eine einzige Anfrage mit fiktivem `Origin` an die entdeckten ECHTEN Endpunkte — ACAO-Reflexion + ACAC=true (Leck mit Anmeldedaten → Hoch), nur Reflexion (Mittel, „könnte Absicht sein"), `*` (informativ), `null`-Akzeptanz (Indikator).
- **H3 TLS-Protokoll/Cipher:** 1 sicherer Handshake pro Protokoll — TLS 1.0/1.1 (veraltet, Indikator), schwacher Cipher (RC4/3DES/NULL/EXPORT). Zertifikatsdauer/-hostname/-kette nur bei einem ECHTEN Problem ein Befund (kein Doppel-CT).
- **H4 Sicherheits-Header-Tiefe:** HSTS (Vorhandensein + max-age-Angemessenheit + preload), Clickjacking (X-Frame-Options **oder** CSP frame-ancestors Doppelmechanismus), CSP-Schwäche (unsafe-inline/eval/*), Referrer-Policy / Permissions-Policy / X-Content-Type-Options.
- Die grundlegende Header-VORHANDENSEINS-Prüfung verbleibt in anderen Paketen; dieser Abschnitt fügt TIEFE hinzu (nicht kopiert).

BEFUNDE

Login/Endpunkt	Technik	Nachweis	Zuversicht	Nebenwirkungsrisiko	Schweregrad
ornek.com	HSTS-(Strict-Transport-Security)-Header-Beobachtung	In der HTTPS-Antwort **kein Strict-Transport-Security** — die erste Verbindung wird nicht auf HTTPS erzwungen, MITM-/Downgrade-Risiko. `max-age≥15768000; includeSubDomains; preload` empfohlen.	Hoch	keins	Mittel
ornek.com	CSP-Vorhandensein-Beobachtung	Content-Security-Policy-Header **fehlt** — keine browserseitige Minderungsschicht gegen injizierte Skripte (Härtungslücke, niedrig).	Hoch	keins	Niedrig
ornek.com	Referrer-Policy-Header-Beobachtung	Referrer-Policy-Header fehlt — Adress-/Sitzungsinformationen könnten zu externen Links durchsickern (informativ). `Referrer-Policy: strict-origin-when-cross-origin` empfohlen.	Hoch	keins	Niedrig
ornek.com	Permissions-Policy-Header-Beobachtung	Permissions-Policy-Header fehlt — der Browser-Funktionszugriff (Kamera/Mikrofon/Standort) wird nicht eingeschränkt (informative Härtung).	Hoch	keins	Niedrig

Die Befunde beruhen auf echter Beobachtung („Indikator, Verifikation erforderlich"); reflected-CORS + credentials ist EINDEUTIG schlecht (Hoch), Reflexion ohne credentials könnte Absicht sein (Mittel). Deterministisch (gleicher Host → gleiches Protokoll/gleiche Header).

CORS: `/` `Access-Control-Allow-Origin: *` (keine credentials) — für eine öffentliche Ressource üblich/akzeptabel (informativ). TLS: unterstützte Protokolle — **TLS 1.2, TLS 1.3** (TLS 1.2/1.3 vorhanden — positiv). TLS-Zertifikat (Dauer/Hostname/Kette) erscheint während des Handshakes gültig — kein deutliches Zertifikatsproblem beobachtet (positiv). Versucht: **11** sichere Beobachtungen (CORS-Origin-Probe + TLS-Handshake + Header-Lesen). Keine

> Datenänderung / kein DoS / keine Cipher-Ausnutzung / kein Takeover-Claim. Zertifikatsgültigkeit nur bei einem ECHTEN Problem ein Befund (kein Doppel-CT).

Subdomain-Takeover (Dangling DNS)

WAS GEPRÜFT WURDE

- CNAME-Auflösung an den mit Certificate Transparency (crt.sh/certSpotter) entdeckten ECHTEN Subdomains.
- Verweist der CNAME auf einen bekannten Drittanbieterdienst (S3/GitHub Pages/Heroku/Azure/Netlify/Fastly/Shopify/... umfangreiche Liste) **UND** gibt einen „nicht beanspruchten" Fingerprint (NoSuchBucket / „There isn't a GitHub Pages site here" / NXDOMAIN usw.) zurück → MÖGLICHES Takeover.
- Nur DNS-Auflösung + ein sicheres GET (Fingerprint-Beobachtung) — es wird KEINE Registrierung/kein Claim DURCHGEFÜHRT. Keine erfundenen Subdomains.

BEFUNDE

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

> Nur bei Übereinstimmung von dangling + Fingerprint ein Befund; ein aktiver/beanspruchter CNAME ist informativ. Ist die CT-Quelle nicht erreichbar, „Nicht untersuchbar" (nicht sauber).

> Entdeckte Subdomains: **7** (CT-Log), CNAME aufgelöst: **7**, Dangling-Takeover-Indikator: **0**. Keine erfundenen Subdomains — nur die in CT beobachteten. Es wurde keine Registrierung/kein Claim durchgeführt.

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 authentifizierten Kontrollen festgestellten Befunde.

Sitzungs-Cookie-Flags

Cookie-Sicherheit — proaktive Härtung

- Erzwingen Sie bei Sitzungs-Cookies proaktiv die Flags Secure/HttpOnly/SameSite.

Session Fixation

Session Fixation — proaktive Härtung

- Standardisieren Sie die Erneuerung der Sitzungs-ID beim Login (Session Regeneration).

Logout / Sitzungsungültigmachung

Sitzungsungültigmachung — Korrektur

- Machen Sie den Sitzungs-Token beim Logout **serverseitig ungültig** (Revocation/Expiry); das clientseitige Löschen des Tokens allein reicht nicht aus.

Forced Browsing / Funktionsebenen-Berechtigung

Funktionsebenen-Autorisierung — Korrektur

- Wenden Sie an jedem administrativen/sensiblen Endpunkt eine **serverseitige Rollen-/Berechtigungsprüfung** an; das bloße Verbergen in der UI reicht nicht aus.

Authentifizierte Injektion (SQLi/XSS)

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.

Authentifiziertes IDOR (eigene Ressourcen)

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.

Rechteausweitung (Privilege Escalation)

Rechteausweitung / Mass-Assignment — proaktive Härtung

- Wenden Sie Mass-Assignment-Schutz (Feld-Allowlist) an; akzeptieren Sie Rollen-/Berechtigungsfelder nicht vom Client (proaktiv).

Mehrstufige Geschäftslogik

Mehrstufige Geschäftslogik — proaktive Härtung

- Verifizieren Sie kritische Werte serverseitig; wenden Sie Schrittreihenfolge + Coupon-Wiederverwendungskontrolle an (proaktiv).

JWT / Token-Sicherheit

JWT / Token-Sicherheit — proaktive Härtung

- Fixieren Sie den Signaturalgorithmus, verwenden Sie ein starkes Geheimnis, fügen Sie `exp` hinzu und tragen Sie keine sensiblen Claims (proaktiv).

Client-Side / JS-Analyse

Client-Side / JS-Sicherheit — proaktive Härtung

- Bewahren Sie im Client-JS keine Geheimnisse auf; veröffentlichen Sie keine Source-Maps in der Produktion; halten Sie Bibliotheken aktuell und wenden Sie Abhängigkeitsscans an (proaktiv).

Client-Side statische Analyse

Client-Side statische Sicherheit — Korrektur

- DOM-XSS: verarbeiten Sie benutzergesteuerte Daten mit textContent + sicheren APIs statt mit innerHTML/eval/document.write; bereinigen Sie sie bei Bedarf mit DOMPurify.
- postMessage: verifizieren Sie in jedem `message`-Handler die `event.origin` mit einer Allowlist.
- Storage: bewahren Sie den Sitzungs-Token in einem **HttpOnly + Secure Cookie** statt im localStorage auf.
- SRI: fügen Sie externen Skripten/Styles `integrity` + `crossorigin` hinzu. Tabnabbing: `rel="noopener noreferrer"` an `target="_blank"`-Links.
- Open Redirect: beschränken Sie Weiterleitungsziele serverseitig mit einer **Allowlist**; leiten Sie nicht auf externe absolute URLs weiter.

Authentifizierungstiefe

Authentifizierungshärtung — Korrektur

- Enumeration: geben Sie bei Login/Reset/Register eine **einheitliche** Antwort/Meldung/Dauer zurück (verraten Sie nicht, ob ein Benutzer existiert).
- Standard-Anmeldedaten: entfernen Sie alle Standardkonten / erzwingen Sie eine Passwortänderung.
- Lockout: wenden Sie bei aufeinanderfolgenden Fehlversuchen eine **konten- + IP-basierte Ratenbegrenzung / temporäre Sperre** an; fügen Sie CAPTCHA hinzu.
- Passwort-Reset: **token-basiert** (Einmalgebrauch, kurze Lebensdauer), vermeiden Sie Sicherheitsfragen; verwenden Sie den Host-Header nicht im Reset-Link.
- Passwortrichtlinie: serverseitig **min. 8+ / Komplexität**; passwortloser Kanal: führen Sie den Login **nur über HTTPS** durch; `Cache-Control: no-store` für sensible Seiten; bieten Sie **MFA** an.

Sitzungssicherheitstiefe

Sitzungssicherheitshärtung — proaktive Härtung

- Sitzungs-Cookie mit SameSite + __Host-Prefix, 128-Bit-zufällige Session-ID, keine Sitzungsübertragung in der URL, Anti-CSRF-Token, angemessenes Timeout (proaktiv).

Eingabe- & Header-Tiefe

Eingabe- & Header-Härtung — proaktive Härtung

- Host-Allowlist, TRACE deaktiviert, alle Ressourcen über HTTPS, Ausgabekodierung/Sanitisierung bei persistenten Eingaben, Parameter-Deduplizierung (proaktiv).

Konfigurations- & Preisgabetiefe

Konfigurations- & Preisgabehärtung — proaktive Härtung

- Backup/.git/.env außerhalb des Verzeichnisses, Admin-Interface hinter Anmeldung/Netzwerk, Bucket privat/Listing deaktiviert, Cache-Vary korrekt, Kommentare/Metadaten in der Produktion sauber (proaktiv).

API-Sicherheitstiefe (OWASP API Top 10)

API-Sicherheitshärtung — Korrektur

- BOLA/BFLA: erzwingen Sie bei jeder API-Anfrage serverseitig die Objekteigentums- + Funktionsrollen-Prüfung (die Gültigkeit der ID reicht nicht).
- Übermäßige Daten: beschränken Sie Antworten mit einer **Feld-Allowlist (DTO)**; senden Sie Passwort-Hash/Geheimnis/interne ID/Rolle nicht an den Client.
- Rate-Limit: konten- + IP- + endpunktbasiertes **Rate-Limit + Kontingent**; kostenbasierte Beschränkung für schwere Endpunkte.
- Version: deaktivieren Sie ungenutzte/alte API-Versionen. GraphQL: **deaktivieren Sie die Introspection in der Produktion**; fügen Sie Abfragetiefe-/Komplexitätslimits hinzu.

E-Mail- & DNS-Tiefe (Anti-Spoofing)

E-Mail- & DNS-Härtung — Korrektur

- DMARC: verschärfen Sie schrittweise `p=none` → `quarantine` → `reject` (pct=100); aktivieren Sie das Reporting mit `rua=`; `sp=reject` für Subdomains.
- SPF: verwenden Sie `-all` (Hardfail); vermeiden Sie `+all` / `?all`; halten Sie die Anzahl der DNS-Lookups ≤10 (PermError).
- DKIM: 2048-Bit-Schlüssel, entfernen Sie widerrufene Selektoren. MTA-STS: `mode=enforce`. Fügen Sie TLS-RPT hinzu.
- Aktivieren Sie DNSSEC (DS+RRSIG). Beschränken Sie mit CAA die autorisierten CAs.

Transportschicht-, CORS- & Sicherheits-Header-Tiefe

Transportschicht- & Header-Härtung — Korrektur

- CORS: setzen Sie die Origin auf eine Allowlist; verwenden Sie mit `credentials` keine Wildcard/Reflexion; akzeptieren Sie keine `null`-Origin.
- TLS: belassen Sie nur TLS 1.2/1.3; deaktivieren Sie RC4/3DES/NULL/EXPORT-Cipher; modernes AEAD (ECDHE+AES-GCM/CHACHA20).
- HSTS `max-age≥15768000; includeSubDomains; preload`; gegen Clickjacking X-Frame-Options **und/oder** CSP frame-ancestors.
- Entfernen Sie unsafe-inline/unsafe-eval aus der CSP (nonce/hash); fügen Sie Referrer-Policy/Permissions-Policy/X-Content-Type-Options hinzu.

Subdomain-Takeover (Dangling DNS)

Subdomain-Takeover — proaktive Härtung

- Subdomain-CNAME-Inventar sauber; keine verwaisten/dangling Einträge (proaktiv).

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.
CSRF	Cross-Site Request Forgery — den Nutzer zu unbeabsichtigten Aktionen bringen.
OWASP	Open Web Application Security Project; bekannt für die Top 10 und Testleitfäden.
TLS	Transport Layer Security (Grundlage von HTTPS).
HSTS	Sicherheitsheader, der den Browser zwingt, nur HTTPS zu verwenden.
CSP	Content Security Policy — Header zur Minderung von XSS/Injection.
CORS	Cross-Origin Resource Sharing; eine lockere Konfiguration ist riskant.
JWT	JSON Web Token — signiertes Token, das Session/Berechtigung trägt.
Clickjacking	Klicks werden durch unsichtbares Framing getäuscht.
Forced Browsing	Zugriff auf versteckte Endpunkte durch Erraten von URLs.