CyberTestify

# Automated Security Scan Report

ornek.com · Full-Scope Pentest Bundle

**CyberTestify Security Scan — Completed**

Report No: **CT-ÖRNEK-332E**

Verification Code: **EB4F-F4B1**

This report is not a formal penetration test / certification.

# Table of Contents

| TARGET | PACKAGE | REPORT NO |
|---|---|---|
| ornek.com | Full-Scope Pentest Bundle | CT-ÖRNEK-332E |

## 1. Executive Summary

**⚠ This sample report is taken from an intentionally vulnerable test application. On real websites the number and severity of findings vary considerably depending on the target's architecture.**

**⚡ Management Decision**

**2** high/critical indicator(s) requiring priority attention were found; an urgent remediation plan is recommended. Medium/low items can be addressed via planned improvement.

- **Overall risk level: High** — the highest risk is in the **Authenticated Injection (SQLi/XSS)** area.
- **Scope:** This section operates in an **authenticated (logged-in)** context; it covers cookie/session/authorization, authenticated injection/IDOR and indicators of privilege escalation + multi-step business logic with **deterministic security controls** (the backend runs this safely; NO completion of payment/account change). Cross-account IDOR (another user's data) is outside the scope of this version.
- **Recommended first step:** Remediate the findings from the checks that were run; ready-made steps are in the "AI Solution Recommendations" section.

### Improvement Priorities

⚡ **Most Urgent:** SQL Injection indicator; Authentication bypass indicator — ' OR 1=1-- (POST /rest/user/login).

🛠 **Quick Wins (~1-2 days):** Insufficient DMARC policy (missing or weak) (ornek.com); Missing HSTS (ornek.com); Missing Content-Security-Policy (ornek.com); Missing Referrer-Policy (ornek.com).

🗓 **Medium-term / Process:** Weak session invalidation (/rest/user/whoami); Unauthorized endpoint access (forced browsing) (/rest/admin/application-configuration); Unauthorized endpoint access (forced browsing) (/api/Users); Unauthorized endpoint access (forced browsing) (/metrics); Unauthorized endpoint access (forced browsing) (/rest/admin/application-version); /api/products/1; /api/users/1; Insecure postMessage origin (main.js · message handler); Sensitive data in client storage (main.js · localStorage['token']); Sensitive data in client storage (main.js · localStorage['totp_tmp_token']); No API rate-limit (/rest/user/login); Excessive data exposure (API/BOPLA) (/api/Products); No API rate-limit (/api/BasketItems); Reflected XSS indicator; Web cache / sensitive caching indicator (/rest/user/whoami); Missing CAA record (ornek.com); Missing Permissions-Policy (ornek.com).

## 2. Findings

### 2.1 Vulnerability Distribution

| Critical | High | Medium | Low |
|---|---|---|---|

A total of **23** finding indicators were detected; the severity distribution is below.

## 2.2 Master Findings Table

| ID | Title | State | Severity |
|---|---|---|---|
| CT-1 | SQL Injection indicator | Open | High |
| CT-2 | Authentication bypass indicator — ' OR 1=1-- — POST /rest/user/login | Open | High |
| CT-3 | Weak session invalidation — /rest/user/whoami | Open | Medium |
| CT-4 | Unauthorized endpoint access (forced browsing) — /rest/admin/application-configuration | Open | Medium |
| CT-5 | Unauthorized endpoint access (forced browsing) — /api/Users | Open | Medium |
| CT-6 | Unauthorized endpoint access (forced browsing) — /metrics · confidence: low (indirect indicator) | Open | Medium |
| CT-7 | Unauthorized endpoint access (forced browsing) — /rest/admin/application-version | Open | Medium |
| CT-8 | /api/products/1 | Open | Medium |
| CT-9 | /api/users/1 | Open | Medium |
| CT-10 | Insecure postMessage origin — main.js · message handler | Open | Medium |
| CT-11 | Sensitive data in client storage — main.js · localStorage['token'] | Open | Medium |
| CT-12 | Sensitive data in client storage — main.js · localStorage['totp_tmp_token'] | Open | Medium |
| CT-13 | No API rate-limit — /rest/user/login | Open | Medium |
| CT-14 | Excessive data exposure (API/BOPLA) — /api/Products | Open | Medium |
| CT-15 | No API rate-limit — /api/BasketItems | Open | Medium |
| CT-16 | Insufficient DMARC policy (missing or weak) — ornek.com | Open | Medium |
| CT-17 | Missing HSTS — ornek.com | Open | Medium |
| CT-18 | Reflected XSS indicator · confidence: low (indirect indicator) | Open | Low |
| CT-19 | Web cache / sensitive caching indicator — /rest/user/whoami | Open | Low |
| CT-20 | Missing CAA record — ornek.com | Open | Low |
| CT-21 | Missing Content-Security-Policy — ornek.com | Open | Low |
| CT-22 | Missing Referrer-Policy — ornek.com | Open | Low |
| CT-23 | Missing Permissions-Policy — ornek.com | Open | Low |

**20 control areas were run — findings: 2 high, 15 medium, 6 low**

✓ EXECUTED — NO VULNERABILITY EVIDENCE

- **JWT / Token Security · Client-Side / JS Analysis · Input & Header Depth · Configuration & Exposure Depth · Subdomain Takeover (Dangling DNS)**

⚠ FINDING PRESENT — DETAILED IN THE SECTIONS BELOW

- **Logout / Session Invalidation · Forced Browsing / Function-Level Authorization · Authenticated Injection (SQLi/XSS) · Authenticated IDOR (own resources) · Login Bypass (SQLi Indicator) · Client-Side Static Analysis · Authentication Depth · API Security Depth (OWASP API Top 10) · E-mail & DNS Depth (Anti-Spoofing) · Transport Layer, CORS & Security-Header Depth**

⚠ NOT APPLICABLE / OUT OF SCOPE — DOES NOT MEAN “SECURE”

- **Session Cookie Flags · Session Fixation · Privilege Escalation · Multi-Step Business Logic · Session Security Depth**

## 2.3 Detailed Findings

### [HIGH] CT-1 · SQL Injection indicator

State: Open

**Description:** An input reaches a DB query (injection indicator).

**How it was detected:** A database error signature was observed in the response (with the "')" payload): "SQLITE_ERROR"

**Business Impact:** Unauthorized access/modification of customer/account records is possible; data breach, breach of UK GDPR / Data Protection Act 2018 and loss of trust.

**RECOMMENDED FIX**

Use parameterized queries/prepared statements; hide DB error messages.

**Reference:** CWE-89 · OWASP A03:2021 Injection

### [HIGH] CT-2 · Authentication bypass indicator — ' OR 1=1--

State: Open

**Affected point:** `POST /rest/user/login`

**Description:** Auth-bypass indicator via a classic SQLi payload on the login form.

**How it was detected:** Control (invalid credentials) → HTTP 401 (failed). SQLi payload ' OR 1=1-- → HTTP 200 + CLEAR success signal: "{"authentication":{"token":"","bid":1,"umail":"admin@juice-sh.op"}}". Authentication is BEING BYPASSED via SQL injection (a session/authorization token was returned — strong evidence). No session takeover/exploitation was performed; the token value is not shown in the report (redacted).

**Business Impact:** Authentication bypass may allow unauthorized access; account and data security directly at risk.

**RECOMMENDED FIX**

Use parameterized queries in auth; validate input; return uniform error messages.

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

### [MEDIUM] CT-3 · Weak session invalidation

State: Open

**Affected point:** `/rest/user/whoami`

**Description:** Session not invalidated server-side after logout.

**How it was detected:** Even AFTER logout was called, the protected endpoint (/rest/user/whoami) returned 200 with the SAME session token — the session is not invalidated server-side.

**Business Impact:** If the session stays valid after logout, a stolen/shared session can be reused; account takeover on shared devices.

**RECOMMENDED FIX**

Invalidate the session server-side on logout (revocation/expiry).

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

### [MEDIUM] CT-4 · Unauthorized endpoint access (forced browsing)

State: Open

**Affected point:** `/rest/admin/application-configuration`

**Description:** An admin/hidden endpoint is reachable with a low-privileged session.

**How it was detected:** The endpoint /rest/admin/application-configuration returned 200 and JSON data with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report).

**Business Impact:** An unauthorized user can reach admin/hidden endpoints; unauthorized data view/modify and insider-abuse risk.

**RECOMMENDED FIX**

Enforce server-side role/authorization on every sensitive endpoint; UI hiding is insufficient.

**Reference:** CWE-284 · OWASP A01:2021 Broken Access Control

### [MEDIUM] CT-5 · Unauthorized endpoint access (forced browsing)

State: Open

**Affected point:** `/api/Users`

**Description:** An admin/hidden endpoint is reachable with a low-privileged session.

**How it was detected:** The endpoint /api/Users returned 200 and JSON data with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report).

**Business Impact:** An unauthorized user can reach admin/hidden endpoints; unauthorized data view/modify and insider-abuse risk.

**RECOMMENDED FIX**

Enforce server-side role/authorization on every sensitive endpoint; UI hiding is insufficient.

**Reference:** CWE-284 · OWASP A01:2021 Broken Access Control

### [MEDIUM] CT-6 · Unauthorized endpoint access (forced browsing)

State: Open

**Affected point:** `/metrics`

**Description:** An admin/hidden endpoint is reachable with a low-privileged session.

**How it was detected:** The endpoint /metrics returned 200 and content with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report).

**Business Impact:** An unauthorized user can reach admin/hidden endpoints; unauthorized data view/modify and insider-abuse risk.

**RECOMMENDED FIX**

Enforce server-side role/authorization on every sensitive endpoint; UI hiding is insufficient.

**Reference:** CWE-284 · OWASP A01:2021 Broken Access Control

### [MEDIUM] CT-7 · Unauthorized endpoint access (forced browsing)

State: Open

**Affected point:** `/rest/admin/application-version`

**Description:** An admin/hidden endpoint is reachable with a low-privileged session.

**How it was detected:** The endpoint /rest/admin/application-version returned 200 and JSON data with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report).

**Business Impact:** An unauthorized user can reach admin/hidden endpoints; unauthorized data view/modify and insider-abuse risk.

**RECOMMENDED FIX**

Enforce server-side role/authorization on every sensitive endpoint; UI hiding is insufficient.

**Reference:** CWE-284 · OWASP A01:2021 Broken Access Control

### [MEDIUM] CT-10 · Insecure postMessage origin

State: Open

**Affected point:** `main.js · message handler`

**Description:** Indicator of missing origin validation in a postMessage handler.

**How it was detected:** In main.js, a message event listener appears to process data without event.origin validation — an indicator of insecure cross-origin message handling. The sender origin should be validated with an allowlist.

**Business Impact:** Indicator of missing postMessage origin validation; a malicious page can inject messages/exfiltrate data.

**RECOMMENDED FIX**

Validate `event.origin` against a strict allowlist in `message` handlers.

**Reference:** CWE-346 · OWASP A05:2021 Security Misconfiguration

### [MEDIUM] CT-11 · Sensitive data in client storage

State: Open

**Affected point:** `main.js · localStorage['token']`

**Description:** Sensitive data (token/session) written to localStorage/sessionStorage.

**How it was detected:** In main.js, localStorage.setItem('token', ...) — sensitive data (session/identity indicator) is written to client storage (value not shown). localStorage/sessionStorage can be read via XSS; an HttpOnly cookie should be preferred for session data.

**Business Impact:** Sensitive data (token/session) written to localStorage/sessionStorage; readable via XSS — prefer HttpOnly cookies.

**RECOMMENDED FIX**

Keep the session token in an `HttpOnly + Secure` cookie instead of localStorage.

**Reference:** CWE-922 · OWASP A04:2021 Insecure Design

### [MEDIUM] CT-12 · Sensitive data in client storage

State: Open

**Affected point:** `main.js · localStorage['totp_tmp_token']`

**Description:** Sensitive data (token/session) written to localStorage/sessionStorage.

**How it was detected:** In main.js, localStorage.setItem('totp_tmp_token', ...) — sensitive data (session/identity indicator) is written to client storage (value not shown). localStorage/sessionStorage can be read via XSS; an HttpOnly cookie should be preferred for session data.

**Business Impact:** Sensitive data (token/session) written to localStorage/sessionStorage; readable via XSS — prefer HttpOnly cookies.

**RECOMMENDED FIX**

Keep the session token in an `HttpOnly + Secure` cookie instead of localStorage.

**Reference:** CWE-922 · OWASP A04:2021 Insecure Design

### [MEDIUM] CT-13 · No API rate-limit

State: Open

**Affected point:** `/rest/user/login`

**Description:** No rate-limit/quota observed on the API endpoint.

**How it was detected:** After 5 consecutive failed login attempts, no lockout, 429 or noticeable delay was observed — a weak/absent lockout & rate-limit indicator (may be exposed to brute-force). No real account was locked (a throwaway user was used).

**Business Impact:** No rate-limit/quota observed on the API endpoint; open to brute-force, scraping and resource exhaustion (DoS).

**RECOMMENDED FIX**

Add per-account+IP+endpoint rate-limit + quota; cost-based limits on heavy endpoints.

**Reference:** CWE-770 · OWASP API4:2023 Unrestricted Resource Consumption

### [MEDIUM] CT-14 · Excessive data exposure (API/BOPLA)

State: Open

**Affected point:** `/api/Products`

**Description:** API response returns excessive/sensitive fields (password-hash/role/internal-id).

**How it was detected:** The API response (/api/Products) contains internal/excessive fields the UI does not need (deletedAt) — internal ID/role/status fields leak to the client (an indicator). Restrict the response to only the needed fields (allowlist DTO).

**Business Impact:** API response returns excessive/sensitive fields (password-hash/role/internal-id); data leak and authz mapping.

**RECOMMENDED FIX**

Constrain responses with a field allowlist (response DTO); never send sensitive fields.

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

### [MEDIUM] CT-15 · No API rate-limit

State: Open

**Affected point:** `/api/BasketItems`

**Description:** No rate-limit/quota observed on the API endpoint.

**How it was detected:** After 10 consecutive requests to an API endpoint (/api/BasketItems), no 429 or rate-limit header was observed — an unrestricted resource consumption (API4) indicator (may be exposed to brute-force/scraping/DoS). The target was not exhausted (modest burst). Rate-limit + quota are recommended.

**Business Impact:** No rate-limit/quota observed on the API endpoint; open to brute-force, scraping and resource exhaustion (DoS).

**RECOMMENDED FIX**

Add per-account+IP+endpoint rate-limit + quota; cost-based limits on heavy endpoints.

**Reference:** CWE-770 · OWASP API4:2023 Unrestricted Resource Consumption

### [MEDIUM] CT-16 · Insufficient DMARC policy (missing or weak)

State: Open

**Affected point:** `ornek.com`

**Description:** No DMARC; SPF/DKIM not enforced.

**How it was detected:** DMARC p=none (organization domain (ornek.com)) — monitoring only; spoofing e-mails are not blocked at the recipient (an indicator). A gradual tightening p=quarantine→reject is recommended.

**Business Impact:** Without a DMARC policy, spoofed emails are not blocked; phishing and brand-reputation risk.

**RECOMMENDED FIX**

Publish `_dmarc` `v=DMARC1; p=quarantine` (start monitoring, then reject).

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

### [MEDIUM] CT-17 · Missing HSTS

State: Open

**Affected point:** `ornek.com`

**Description:** No HSTS; browser not forced to HTTPS.

**How it was detected:** The HTTPS response has no Strict-Transport-Security — the first connection is not forced to HTTPS, a MITM/downgrade risk. max-age≥15768000; includeSubDomains; preload is recommended.

**Business Impact:** Without forced HTTPS, traffic can be intercepted/redirected via man-in-the-middle.

**RECOMMENDED FIX**

Add `Strict-Transport-Security: max-age=31536000; includeSubDomains` (if fully HTTPS).

**Reference:** CWE-319 · OWASP A05:2021 Security Misconfiguration

### [LOW] CT-18 · Reflected XSS indicator

State: Open

**Description:** User input reflects unencoded in HTML (reflected XSS indicator).

**How it was detected:** The marker string was reflected but PARTIALLY/ENCODED (special characters < > " were escaped) — a context-dependent low-confidence indicator; manual verification is recommended.

**Business Impact:** Running script in the victim's browser enables session theft/impersonation and account takeover.

**RECOMMENDED FIX**

Context-encode output (HTML entity encoding); restrict inline script via CSP.

**Reference:** CWE-79 · OWASP A03:2021 Injection

### [LOW] CT-19 · Web cache / sensitive caching indicator

State: Open

**Affected point:** `/rest/user/whoami`

**Description:** Sensitive responses are cacheable or there is unkeyed header reflection.

**How it was detected:** The authenticated response (/rest/user/whoami) does not contain Cache-Control: no-store/no-cache/private (observed: none) — a browser/intermediate proxy could cache sensitive content.

**Business Impact:** Indicator that sensitive responses are cacheable / unkeyed header reflection; cache poisoning or data leak.

**RECOMMENDED FIX**

Set `Cache-Control: no-store` on sensitive responses; include relevant headers in the cache key.

**Reference:** CWE-525 · OWASP A05:2021 Security Misconfiguration

### [LOW] CT-20 · Missing CAA record

State: Open

**Affected point:** `ornek.com`

**Description:** No CAA record; any CA can issue certificates.

**How it was detected:** No CAA record was observed — any CA can issue a certificate for ornek.com (a mis-issuance risk indicator). CAA can restrict authorized CAs.

**Business Impact:** No CAA record; any CA can issue certificates for the domain — mis-issuance risk.

**RECOMMENDED FIX**

Restrict authorized CAs with a `CAA` record (e.g. `0 issue "letsencrypt.org"`).

**Reference:** CWE-295 · OWASP A05:2021 Security Misconfiguration

### [LOW] CT-21 · Missing Content-Security-Policy

State: Open

**Affected point:** `ornek.com`

**Description:** No Content-Security-Policy; no browser-side XSS mitigation.

**How it was detected:** There is no Content-Security-Policy header — no browser-side mitigation layer against injected scripts (a hardening gap, low).

**Business Impact:** No browser-side mitigation against content injection/XSS; injected scripts can execute.

**RECOMMENDED FIX**

Define a strict CSP (`default-src 'self'`) and allowlist third-party sources.

**Reference:** CWE-693 · OWASP A05:2021 Security Misconfiguration

### [LOW] CT-22 · Missing Referrer-Policy

State: Open

**Affected point:** `ornek.com`

**Description:** No Referrer-Policy; full URL may leak to external links.

**How it was detected:** There is no Referrer-Policy header — address/session information can leak to external links (informational). Referrer-Policy: strict-origin-when-cross-origin is recommended.

**Business Impact:** Address/session info may leak to external links; low-impact information exposure.

**RECOMMENDED FIX**

Add `Referrer-Policy: strict-origin-when-cross-origin`.

**Reference:** CWE-200 · OWASP A05:2021 Security Misconfiguration

### [LOW] CT-23 · Missing Permissions-Policy

State: Open

**Affected point:** `ornek.com`

**Description:** No Permissions-Policy header; browser feature access is unrestricted.

**How it was detected:** There is no Permissions-Policy header — browser feature access (camera/microphone/location) is not restricted (informational hardening).

**Business Impact:** No Permissions-Policy; browser feature access (camera/mic/geolocation) is unrestricted (informational hardening).

**RECOMMENDED FIX**

Add a `Permissions-Policy` disabling unused features (e.g. `geolocation=(), camera=(), microphone=()`).

**Reference:** CWE-693 · OWASP A05:2021 Security Misconfiguration

### Positive Assurance

The following controls were executed with no significant vulnerability indicator: JWT / Token Security, Client-Side / JS Analysis, Input & Header Depth, Configuration & Exposure Depth, Subdomain Takeover (Dangling DNS).

## 3. Controls & Methodology

### Assessment Summary (Authenticated)

**This scan was performed in an AUTHENTICATED (logged-in) context using the session of the provided TEST account.** 20 authenticated checks were assessed; **311** requests in total. The highest risk is in the **Authenticated Injection (SQLi/XSS)** area (detailed below).

*The password was never sent externally/to a third-party service; the backend performed a deterministic login and used only the session (cookie/token).*

### CONTROLS SUMMARY

| Control | Result | Confidence |
|---|---|---|
| Session Cookie Flags | No applicable input point (out of scope) | Out of scope |
| Session Fixation | No applicable input point (out of scope) | Out of scope |
| Logout / Session Invalidation | ⚠ Limited indicator | Medium |
| Forced Browsing / Function-Level Authorization | ⚠ Limited indicator | Medium |
| Authenticated Injection (SQLi/XSS) | ⚠ Vulnerability indicator | High |
| Authenticated IDOR (own resources) | ⚠ Limited indicator | Medium |
| Privilege Escalation | Not assessable — no safely testable surface | Out of scope |
| Multi-Step Business Logic | Not assessable — no safely testable surface | Out of scope |
| JWT / Token Security | ✓ No vulnerability evidence | High |
| Login Bypass (SQLi Indicator) | ⚠ Vulnerability indicator | High |

| Control | Result | Confidence |
|---|---|---|
| Client-Side / JS Analysis | ✓ No vulnerability evidence | High |
| Client-Side Static Analysis | ⚠ Limited indicator | Medium |
| Authentication Depth | ⚠ Limited indicator | High |
| Session Security Depth | No applicable input point (out of scope) | Out of scope |
| Input & Header Depth | ✓ No vulnerability evidence | Medium |
| Configuration & Exposure Depth | ✓ No vulnerability evidence | High |
| API Security Depth (OWASP API Top 10) | ⚠ Limited indicator | Medium |
| E-mail & DNS Depth (Anti-Spoofing) | ⚠ Limited indicator | Medium |
| Transport Layer, CORS & Security-Header Depth | ⚠ Limited indicator | Medium |
| Subdomain Takeover (Dangling DNS) | ✓ No vulnerability evidence | Medium |

> Confidence is shown only for actually applicable checks (input/cookie/endpoint found); non-applicable checks are **out of scope** (e.g. cookie-flag/fixation on a session that uses a token instead of cookies).

## Session Cookie Flags

### WHAT WAS CHECKED

- Among the cookies observed after login, only real **server session cookies** were assessed.
- For each session cookie, the **Secure / HttpOnly / SameSite** security flags were checked.
- Analytics/3rd-party cookies (_ga, _fbp, _clck etc.) are not counted as session cookies (HttpOnly is impossible on them) — they are not assessed.

### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> No server-side session cookie was observed (the session is carried via bearer/token) — Set-Cookie security-flag analysis is **out of scope** for this target.

> **Scope and method:** This package operates under the "prove — do not exploit" principle. The backend sent a limited number of **harmless** verification probes to the target; no data was retrieved, modified or deleted. Inter-request delays and a target-health circuit breaker (consecutive 5xx / excessive slowdown / WAF) are applied. This section was run with the session of the TEST account you provided in an authenticated (logged-in) context; completion of payment/account-status change is blocked at the code level.

## Session Fixation

### WHAT WAS CHECKED

- The session cookie value taken from the home page BEFORE login (unauthenticated) was observed.
- It was **compared** with the session cookie value AFTER login (if equal, the server does not renew the session).

- A single GET only; no state was changed.

#### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> The session is not cookie-based (bearer/token) — session fixation (cookie renewal) analysis is **out of scope** for this target.

### Logout / Session Invalidation

#### WHAT WAS CHECKED

- A protected endpoint (whoami/profile) that returns 200 with the session was identified.
- A logout endpoint callable via GET was tried (**no POST** — no state change).
- It was checked whether the protected endpoint can be accessed again with the SAME token AFTER logout.

#### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| /rest/user/whoami | token reuse after logout | Even AFTER logout was called, the protected endpoint (/rest/user/whoami) returned 200 with the SAME session token — the session is not invalidated server-side. | Medium | none | Medium |

### Forced Browsing / Function-Level Authorization

#### WHAT WAS CHECKED

- A **GET** request was sent to common admin/management endpoints with the session of the AVAILABLE (presumably low-privilege) test account.
- To avoid false positives, only 200s returning content/JSON **DIFFERENT** from the home-page shell were counted as a finding.
- A single attempt, GET-only, subject to the circuit breaker.

#### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| /rest/admin/application-configuration | forced browsing with a low-privilege session (GET) | The endpoint `/rest/admin/application-configuration` returned **200** and JSON data with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report). | Medium | none | Medium |

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| /api/Users | forced browsing with a low-privilege session (GET) | The endpoint `/api/Users` returned **200** and JSON data with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report). | Medium | none | Medium |
| /metrics | forced browsing with a low-privilege session (GET) | The endpoint `/metrics` returned **200** and content with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report). | Low | none | Medium |
| /rest/admin/application-version | forced browsing with a low-privilege session (GET) | The endpoint `/rest/admin/application-version` returned **200** and JSON data with the low-privilege test account's session — an indicator of missing function-level authorization (possible unauthorized admin access) (the returned data is not shown in the report). | Medium | none | Medium |

## Authenticated Injection (SQLi/XSS)

### WHAT WAS CHECKED

On the input points discovered from the scanned pages (home page + internal links + well-known paths) — URL query parameters + form fields — harmless verification probes per input point:

- **SQLi (error-based):** A single quote ( `'` ) was injected and a database error signature (MySQL/PostgreSQL/Oracle/MSSQL/SQLite) was searched for in the response.
- **SQLi (time-based):** At points with no error, a single harmless delay probe (SLEEP) measured the response time against the baseline (blind-SQLi indicator).
- **SQLi (boolean-based):** At numeric/ID-like points, two requests with TRUE ( `1=1` ) and FALSE ( `1=2` ) conditions were sent and their responses (status + content length) compared; if the TRUE response is consistent on repeat and PERSISTENTLY different from FALSE, it is a boolean-based SQLi indicator (a stability check is performed against false positives).
- **XSS (reflected):** A unique, harmless marker string was injected and it was checked whether it is reflected **unencoded** in the response HTML (no JS was run; stored XSS was not attempted).

### FINDINGS

| Input Point | Type | Technique | Evidence | Confidence (rationale) | Severity |
|---|---|---|---|---|---|
| GET /rest/products/search?q | SQLi | error-based | A database error signature was observed in the response (with the "')" payload): "SQLITE_ERROR" | High — database error signature in the response (direct evidence) | High |

| Input Point | Type | Technique | Evidence | Confidence (rationale) | Severity |
|---|---|---|---|---|---|
| GET /redirect?to | XSS | reflection | The marker string was reflected but PARTIALLY/ENCODED (special characters `< > "` were escaped) — a context-dependent low-confidence indicator; manual verification is recommended. | Low — reflected but encoded/escaped; a weak context-dependent indicator | Low |

## Authenticated IDOR (own resources)

### WHAT WAS CHECKED

On endpoints discovered from the scanned pages that contain a predictable/numeric ID (e.g. `?id=123`, `/user/45`):

- The ID value was changed **to a neighbouring value** (N-1 / N+1) and a **GET** request was sent with the authenticated session.
- Only the response **status and size** were compared; **the returned content was not stored/quoted**.
- Returning a different, valid-looking resource was counted as an enumerable-access indicator.

### FINDINGS

| Endpoint | ID | Observation | Severity |
|---|---|---|---|
| /api/products/1 | path-id | Without authentication, the neighbouring ID (2) returned a 200 response with the SAME STRUCTURE (JSON skeleton) but DIFFERENT CONTENT — most likely another record's data (the returned data is not shown in the report). An indicator of enumerable resource access (possible IDOR). | Medium |
| /api/users/1 | path-id | Without authentication, the neighbouring ID (2) returned a 200 response with the SAME STRUCTURE (JSON skeleton) but DIFFERENT CONTENT — most likely another record's data (the returned data is not shown in the report). An indicator of enumerable resource access (possible IDOR). | Medium |

### SCOPE (IMPORTANT)

This section was run with the **session (logged in)** of the TEST account you provided and tests unauthorized access to the account's **own** enumerable resources. A **cross-account** test (accessing another user's data) is outside the scope of this version (it requires two separate accounts). The absence of findings does not prove that there is no IDOR in all authenticated flows.

> In addition, sequential numeric IDs ( `/{1..3}` ) were derived from **1** collection-like endpoint (e.g. `/rest/products` ) and tested via GET using the content-difference method.

## Privilege Escalation

### WHAT WAS CHECKED

- The discovered authenticated surface was searched **deterministically** for forms/APIs with a privilege field (registration/profile/settings type).
- If a safely testable surface was found, ONE observational mass-assignment probe ( `role/isAdmin` extra field) was applied with a **safe, authenticated-light** function.

⚠ No real escalation was COMPLETED; no re-login with elevated privilege; the session was not left; account-changing/checkout targets were **not written to** (code-level blocklist).

### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> The mass-assignment indicator was derived only from the first response (low confidence); definitive verification requires a manual test.

> A SAFELY testable form/endpoint for privilege escalation (non-forbidden; registration/profile/settings type) was not found on this target — this check was **Not assessable** (no suitable authenticated surface for the deterministic probe). Note: some vulnerabilities (e.g. hidden acceptance of `role` in the API) are not visible in the UI/DOM; a definitive result requires a manual test.

## Multi-Step Business Logic

### WHAT WAS CHECKED

- Multi-step flows/price-coupon fields were searched for **deterministically**; the backend performed **GET observation** ONLY.
- A client-modifiable hidden price/quantity/coupon field + a "confirmation" step reachable without a precondition were observed.

⚠ Only up to the cart/form; **payment/checkout NOT completed** (code-level blocklist); no resource was consumed.

### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> Business-logic vulnerabilities are context-specific; this check is at the surface/indicator level.

> This check was run **deterministically** (observational step-skipping + client-modifiable price/quantity/coupon field). The AI otonom analiz layer is **disabled by default** (as it produced no additional confirmed evidence in the experiment).

## JWT / Token Security

### WHAT WAS CHECKED

- If the session carries a **JWT bearer**, the token was decoded and analysed OFFLINE: the signature algorithm (**alg=none / unsigned**), whether the signing secret is **weak/common** (OFFLINE verification with common secrets), and **sensitive/excessive claims** in the token body (password/secret, role/privilege).
- In addition, a SINGLE harmless observation: whether an unsigned (alg=none) forged token is ACCEPTED at a protected endpoint (observation only; the access was not used).

⚠ No real exploitation — no token takeover/privilege escalation; only the security indicator was reported.

### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> Weak-secret and alg=none ACCEPTANCE indicators are definitive (high confidence); claim observations are informational.

> The token has no expiry (exp) claim — a non-expiring token carries the risk of permanent access if stolen. No verifiable protected read endpoint (whoami/profile) was found for the alg=none acceptance observation — this observation was

skipped. No JWT/token security indicator was found (not alg=none, no weak secret confirmed, no sensitive claim).

## Login Bypass (SQLi Indicator)

### WHAT WAS CHECKED

- The login endpoint was first sent **invalid credentials** (control); then classic SQLi payloads ( `' OR '1'='1` etc.) were tried and it was observed whether — contrary to the control — a session/success (token/2xx) was returned.
- The login POST is already a permitted flow; it is a SINGLE, harmless observation.

⚠ NO session takeover/exploitation — only the observation of "whether an authentication-bypass indicator is present".

### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| POST /rest/user/login | login bypass (SQLi: `' OR 1=1--` ) | Control (invalid credentials) → HTTP 401 (failed). SQLi payload `' OR 1=1--` → HTTP 200 + CLEAR success signal: "{"authentication": {"token":"***","bid":1,"umail":"admin@juice-sh.op"}}". Authentication is BEING BYPASSED via SQL injection (a session/authorization token was returned — strong evidence). No session takeover/exploitation was performed; the token value is not shown in the report (redacted). | High | none | High |

The indicator is based on comparison with the control attempt; definitive verification requires a manual test.

## Client-Side / JS Analysis

### WHAT WAS CHECKED

- The JS bundles the page loads (script srcs + inline) were fetched and analysed **statically** — NO active exploitation, only download + read.
- **A) Secret scan:** REAL secrets such as private keys, AWS/GCP credentials, Stripe **sk_live_**, GitHub/GitLab/Slack tokens, DB connection strings were searched for (values REDACTED).
- **Public-by-design distinction:** Firebase apiKey, GTM/GA measurement id, Google Maps browser key, Stripe **pk_**, reCAPTCHA site key are public by design → NOT counted as "exposure/secret", listed only informationally.
- **B) Source-map exposure:** `//# sourceMappingURL` comments + `.js.map` candidates were tried; a reachable `.map` → leakage of the original source code/tree.
- **C) Known-vulnerable library:** loaded libraries + their VERSIONS were detected and CONSERVATIVELY matched against known CVEs; if the version could not be read safely, NO CVE match was made ("patch confirmation required" language; not "exploitable").

### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

Everything is passive/static; library findings are version-based indicators (your distribution may be patched/backported — confirmation recommended).

Scanned: **3** same-origin JS files + **1** inline script; **0** library(ies) detected; **3** source-map candidate(s) tried. (0 external/CDN scripts examined for version.)

## Client-Side Static Analysis

### WHAT WAS CHECKED

- 6 client-side checks were performed **statically** (NO active exploitation) on the same JS/HTML corpus.
- **1) DOM-based XSS (indicator):** whether a dangerous sink (innerHTML/document.write/eval/.html()/location=) and a user-controlled source (location.hash/search, referrer, window.name) are in the SAME expression — **NOT proven XSS**, a low-confidence indicator (dynamic verification required).
- **2) postMessage:** whether `message` event listeners perform an **event.origin check**.
- **3) Browser storage:** writing of **sensitive data** (token/JWT/session) to localStorage/sessionStorage (static; value REDACTED).
- **4) Missing SRI:** whether external script/style has an `integrity` attribute (supply chain).
- **5) Reverse tabnabbing:** whether `target="_blank"` links have `rel="noopener/noreferrer"`.
- **6) Open redirect:** a **single safe probe** to redirect parameters OBSERVED in the HTML — a harmless external URL was sent and Location observed; **the redirect was NOT followed**.

### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| main.js · message handler | web messaging (postMessage) origin-validation analysis | In `main.js`, a `message` event listener appears to process data **without event.origin validation** — an indicator of insecure cross-origin message handling. The sender origin should be validated with an allowlist. | Medium | none | Medium |
| main.js · localStorage['token'] | browser storage static sensitive-data analysis | In `main.js`, `localStorage.setItem('token', …)` — sensitive data (session/identity indicator) is written to client storage (value not shown). localStorage/sessionStorage can be read via XSS; an HttpOnly cookie should be preferred for session data. | Medium | none | Medium |
| main.js · localStorage['totp_tmp_token'] | browser storage static sensitive-data analysis | In `main.js`, `localStorage.setItem('totp_tmp_token', …)` — sensitive data (session/identity indicator) is written to client storage (value not shown). localStorage/sessionStorage can be read via XSS; an HttpOnly cookie should be preferred for session data. | Medium | none | Medium |

DOM-XSS findings are STATIC indicators (high false-positive potential; dynamic verification required). The others are deterministic observations.

Statically parsed: **3** same-origin JS + **1** inline scripts; **0** external resources checked for SRI, **0** `target=_blank` links checked for tabnabbing; **0** redirect parameters tested with a safe probe (redirect NOT followed). DOM-XSS matches are a STATIC

**indicator** (not proven XSS); they carry a high false-positive potential and require dynamic verification. No observed redirect parameter was found in the HTML — the open-redirect probe is **out of scope**.

## Authentication Depth

### WHAT WAS CHECKED

- 9 authentication-depth checks (WSTG-ATHN/IDNT) — **safe, low volume**; NO brute-force/DoS.
- **1) Account enumeration:** response difference for a valid (test account) vs invalid (random) user — one FAILED attempt each; creates no account.
- **2) Default credentials:** a small fixed list (admin/admin etc.) — only a failed login; if accepted, the session is NOT used.
- **3) Weak lockout/rate-limit:** a few failed attempts with a THROWAWAY (random) user — **a real account is NEVER locked**.
- **4) Password reset:** mechanism observation (security question/token) — **NO real reset e-mail is TRIGGERED** (non-existent e-mail).
- **5) Password/registration policy:** client-side observation only — **NO real registration is performed**.
- **6) "Remember me" cookie · 7) Authenticated page cache (Cache-Control) · 8) MFA presence (informational) · 9) Credentials over an unencrypted channel (HTTP).**

### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| /rest/user/login | account lockout / login rate-limit observation (with a throwaway user) | After 5 consecutive failed login attempts, no lockout, 429 or noticeable delay was observed — **a weak/absent lockout & rate-limit indicator** (may be exposed to brute-force). No real account was locked (a throwaway user was used). | Medium | none | Medium |
| /rest/user/whoami | authenticated-response cache-header observation | The authenticated response ( `/rest/user/whoami` ) does not contain `Cache-Control: no-store/no-cache/private` (observed: `none` ) — a browser/intermediate proxy could cache sensitive content. | Medium | none | Low |

Enumeration/lockout/reset findings are "indicators" (verification required). Safety: no real account was locked, no reset e-mail was sent, no registration was performed, and even if a successful login was found the session was not used.

A password-reset endpoint was observed; the token/mechanism could not be validated statically (no real e-mail was triggered) — limited for this section. No client-side password field was observed on the home page (may be SPA/separate page) — the password policy was only examined statically in a limited way. The session is bearer/token-based (a cookie-based "remember me" persistence cookie is out of scope). No MFA/2FA indicator was observed in the login flow (informational; a second factor is recommended). Performed: **16** safe auth probes (default credentials only as a failed login; lockout with a THROWAWAY user; reset with a non-existent e-mail; NO registration; no real account locked).

## Session Security Depth

### WHAT WAS CHECKED

- IF a REAL server-side session cookie (Set-Cookie session) is present, 5 session-management checks — all **read-only/observational** (NO state-changing submission / no real CSRF attack). On a bearer/JWT/token-based target this section is **out of scope**.
- **1) CSRF (SESS-05):** observation of SameSite on the session cookie + anti-CSRF token in a state-changing POST form (static; form NOT submitted).
- **2) Session-id entropy (SESS-01):** structural analysis of the session id (length/charset/entropy) — NO brute; if a JWT, left to a separate section.
- **3) Session in URL (SESS-04):** whether the session id is exposed in the URL/query (jsessionid/sid etc.).
- **4) Timeout / concurrent sessions (SESS-07/11):** an observational indicator (definitive test is manual).
- **5) Cookie prefix (SESS-02):** whether the session cookie lacks a __Host-/__Secure- prefix.

#### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> Findings are "indicators" (no real CSRF attack/brute performed; token/session REDACTED). Without a server-side session cookie the checks are out of scope (token/JWT security in a separate section).

> No server-side session cookie (Set-Cookie session) was observed — the session is carried via a **Bearer/JWT token**. The session-COOKIE security checks (CSRF SameSite / cookie prefix / session-id entropy / session in URL) are **out of scope** for this target. Token/JWT security is covered in the separate **JWT / Token Security** section.

### Input & Header Depth

#### WHAT WAS CHECKED

- Safe/read-only input & header checks (GET/OPTIONS/TRACE only — NO state-changing/destructive request).
- **D1 Host header injection:** a spoofed `X-Forwarded-Host` was sent and reflection in the response observed (indicator).
- **D2 HTTP parameter pollution:** the same parameter was repeated and the processing difference observed (observational).
- **D3 HTTP methods / TRACE:** allowed-method discovery via OPTIONS + TRACE observation — **no PUT/DELETE SENT**.
- **D4 Mixed content:** static detection of HTTP resources (script/img/iframe) on an HTTPS page.
- **D5 Stored-XSS entry point:** CANDIDATE fields that could be reflected into a persistent context were flagged — **NO data was SENT/saved** (low confidence, dynamic verification required).

#### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> Host-injection/HPP/D5 are "indicator/candidate" (no real attack/submission performed). TRACE/mixed-content are deterministic observations.

> No parameterised same-origin endpoint was observed — no testable surface for HPP. Performed: **4** safe input/header probes (GET/OPTIONS/TRACE only — PUT/DELETE NOT SENT; stored-XSS candidate only, no data CREATED). Mixed content: 0 HTTP resources.

### Configuration & Exposure Depth

#### WHAT WAS CHECKED

- Safe GET + static configuration/exposure checks. Guessed paths that return an **SPA catch-all 200 (home-page shell)** are NOT counted as real — only genuinely reachable, DISTINCTIVE responses produce a finding (Column 0 provenance).
- **E1 Backup/old file:** `.env/.bak/.sql/.git/config` etc. via safe GET (shell 200 filtered out).
- **E2 Admin interface:** whether common management paths are externally reachable (same provenance).
- **E3 Cloud storage:** public S3/GCS/Azure bucket reference in HTML/JS + whether it is **listable**.
- **E4 Cache / poisoning indicator:** Cache-Control/Vary + unkeyed-header reflection (NO poisoning performed).
- **E5 Comment & metadata leak:** dev comment / internal IP-hostname / server file path (static).

#### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> Backup/admin findings are produced only for genuinely reachable, NON-SPA-shell responses. Cache/host indicators are "indicators"; content/sensitive data REDACTED.

> Performed: **14** backup/old files + **11** admin paths (SPA catch-all shell 200s FILTERED OUT — non-real/non-distinctive responses were not counted as findings); **0** cloud-storage reference(s); comments/metadata scanned statically.

### API Security Depth (OWASP API Top 10)

#### WHAT WAS CHECKED

- 5 checks on REAL API endpoints discovered from same-origin JS/HTML that return JSON (NOT an SPA catch-all shell) — all read-only. Without a real API (Firebase/client-SDK/SPA) this section is **out of scope**.
- **F1 BOLA/BFLA (API1/API5):** object/function-level authorization — evaluated in the **Authenticated IDOR** + **Forced Browsing** sections (not re-probed here to avoid duplicate findings).
- **F2 Excessive data exposure / BOPLA (API3):** sensitive/excessive field in the API response (password-hash/role/internal-ID) — value REDACTED.
- **F3 Rate-limit (API4):** whether a 429/rate-limit header appears after a MODEST burst (NOT DoS) — the target was not exhausted.
- **F4 Shadow/deprecated version (API9):** whether version endpoints like /v1,/v2,/api/v1 are reachable (SPA shell filtered out).
- **F5 GraphQL introspection (APIT-99):** if a GraphQL endpoint exists, whether introspection is enabled — **read-only query; NO mutation**.

#### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| /api/Products | excessive data exposure / BOPLA (API3) — response field observation | The API response ( `/api/Products` ) contains internal/excessive fields the UI does not need ( `deletedAt` ) — internal ID/role/status fields leak to the client (an indicator). Restrict the response to only the needed fields (allowlist DTO). | Medium | none | Medium |

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| /api/BasketItems | API rate-limit / unrestricted-consumption observation (modest burst — not DoS) | After 10 consecutive requests to an API endpoint ( `/api/BasketItems` ), **no 429 or rate-limit header was observed** — an unrestricted resource consumption (API4) indicator (may be exposed to brute-force/scraping/DoS). The target was not exhausted (modest burst). Rate-limit + quota are recommended. | Medium | none | Medium |

Findings are "indicators"; only at genuinely observed (JSON, non-SPA-shell) endpoints. Sensitive data REDACTED; no mutation/data-change/DoS performed.

BOLA/BFLA (OWASP API1/API5 — object/function-level authorization) was evaluated in the **Authenticated IDOR** and **Forced Browsing** sections (not re-probed here to avoid duplicate findings). Discovered real API endpoints: **8** (same-origin, JSON, not an SPA shell). Performed: **38** safe probes (GET + read-only introspection only; NO mutation/data-change/DoS).

## E-mail & DNS Depth (Anti-Spoofing)

### WHAT WAS CHECKED

- Only DNS records read from the REAL org domain; all PASSIVE DNS/TXT + a single safe MTA-STS GET (no attack/state-change). For a subdomain, DMARC is evaluated at the **organizational domain** level.
- **G1 DMARC** policy strength (p=none/quarantine/reject, pct, sp subdomain, adkim/aspf, rua).
- **G2 SPF** depth (-all/~all/?all/+all + top-level DNS-lookup count, RFC 7208).
- **G3 DKIM** common-selector discovery (default/google/selector1…) + key length (~1024/2048) + revocation (empty p=).
- **G4 MTA-STS** (_mta-sts TXT + policy: enforce/testing/none) · **G5 TLS-RPT** reporting.
- **G6 DNSSEC** (signature/validation — AD flag, observation only) · **G7 CAA** (certificate-issuance restriction) + **BIMI** (informational).

### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| ornek.com | passive DNS/TXT observation | DMARC **p=none** (organization domain (ornek.com)) — monitoring only; spoofing e-mails are not blocked at the recipient (an indicator). A gradual tightening `p=quarantine` → `reject` is recommended. | High | none | Medium |
| ornek.com | passive DNS/TXT observation | No CAA record was **observed** — any CA can issue a certificate for `ornek.com` (a mis-issuance risk indicator). CAA can restrict authorized CAs. | High | none | Low |

Evidence = the public record DNS actually returned (no redaction needed). "indicator" language; on a non-mail-sending domain (no MX), the severity of missing DMARC/SPF is lowered. No fabricated/guessed records.

DMARC organization domain (ornek.com): `v=DMARC1; p=none; rua=mailto:rua@dmarc.brevo.com` SPF: `v=spf1 include:spf.improvmx.com ~all` SPF **~all** (softfail) — acceptable; `-all` can be considered for strict protection. DKIM: no record was **observed** at common selectors (default/google/selector1...) — NOT a definitive absence; a custom selector may be in use (honest). MTA-STS: no `_mta-sts` TXT observed — no SMTP MITM/downgrade protection (a low/maturity indicator). TLS-RPT ( `_smtp._tls` TXT) not observed — SMTP TLS delivery problems are not reported (informational). DNSSEC: the domain is **signed and validated** (AD flag) — DNS integrity is protected (positive). BIMI ( `default._bimi` TXT) not observed — informational, no security impact. Queried organization domain: **ornek.com** (target subdomain: ornek.com). A total of **20** passive DNS queries + at most 1 safe MTA-STS GET. NO attack/state-change/resolver-attack.

## Transport Layer, CORS & Security-Header Depth

### WHAT WAS CHECKED

- All read-only/passive observation — NO data change, DoS or cipher exploitation.
- **H1 CORS:** a single request with a crafted `Origin` to discovered REAL endpoints — ACAO reflection + ACAC=true (credentialed leak → High), reflection only (Medium, "may be by design"), `*` (informational), `null` acceptance (indicator).
- **H3 TLS protocol/cipher:** 1 safe handshake per protocol — TLS 1.0/1.1 (old, indicator), weak cipher (RC4/3DES/NULL/EXPORT). Certificate validity/hostname/chain is a finding only on a REAL problem (no double CT).
- **H4 security-header depth:** HSTS (presence + max-age sufficiency + preload), clickjacking (X-Frame-Options **or** CSP frame-ancestors dual-mechanism), CSP weakness (unsafe-inline/eval/*), Referrer-Policy / Permissions-Policy / X-Content-Type-Options.
- The basic header-PRESENCE check remains in other packages; this section adds DEPTH (not copied).

### FINDINGS

| Input/Endpoint | Technique | Evidence | Confidence | Side-effect risk | Severity |
|---|---|---|---|---|---|
| ornek.com | HSTS (Strict-Transport-Security) header observation | The HTTPS response has **no Strict-Transport-Security** — the first connection is not forced to HTTPS, a MITM/downgrade risk. `max-age≥15768000; includeSubDomains; preload` is recommended. | High | none | Medium |
| ornek.com | CSP presence observation | There is **no** Content-Security-Policy header — no browser-side mitigation layer against injected scripts (a hardening gap, low). | High | none | Low |
| ornek.com | Referrer-Policy header observation | There is no Referrer-Policy header — address/session information can leak to external links (informational). `Referrer-Policy: strict-origin-when-cross-origin` is recommended. | High | none | Low |
| ornek.com | Permissions-Policy header observation | There is no Permissions-Policy header — browser feature access (camera/microphone/location) is not restricted (informational hardening). | High | none | Low |

Findings are based on real observation ("indicator, verification required"); reflected CORS + credentials is CLEARLY bad (High), reflection without credentials may be by design (Medium). Deterministic (same host → same protocol/header).

> CORS: `/` `Access-Control-Allow-Origin: *` (no credentials) — common/acceptable for a public resource (informational). TLS: supported protocols — **TLS 1.2, TLS 1.3** (TLS 1.2/1.3 present — positive). The TLS certificate (validity/hostname/chain) appears valid during the handshake — no notable certificate problem observed (positive). Performed: **11** safe observations (CORS origin probe + TLS handshake + header read). NO data-change / DoS / cipher exploitation / takeover-claim. Certificate validity is a finding only on a REAL problem (no double CT).

## Subdomain Takeover (Dangling DNS)

### WHAT WAS CHECKED

- CNAME resolution on REAL subdomains discovered via Certificate Transparency (crt.sh/certSpotter).
- If the CNAME points to a known 3P service (S3/GitHub Pages/Heroku/Azure/Netlify/Fastly/Shopify/... comprehensive list) **AND** an "unclaimed" fingerprint (NoSuchBucket / "There isn't a GitHub Pages site here" / NXDOMAIN etc.) is returned → POSSIBLE takeover.
- DNS resolution + a single safe GET (fingerprint observation) only — NO registration/claim is made. No fabricated subdomains.

### FINDINGS

No clear vulnerability indicator was found against the harmless probes sent.

> A finding only when dangling + fingerprint match; a live/claimed CNAME is informational. If the CT source is unreachable, "not assessable" (not clean).

> Discovered subdomains: **7** (CT log), CNAME-resolved: **7**, dangling-takeover indicators: **0**. No fabricated subdomains — only those observed in CT. No registration/claim was made.

## Methodology

| Stage | Description |
|---|---|
| **Discovery** | The external surface, pages and (for SPAs) real endpoints/params from the JS bundle are extracted. |
| **Automated detection** | Deterministic, safe (read-only) indicators are sought on the discovered surface. |
| **Manual-deterministic verification** | Findings are verified with code-based rules; “prove, don't exploit”. |
| **Authz & session** | In the authenticated package, cookie/session/authorization and post-login surface are examined. |
| **Configuration** | Header, TLS, email/DNS and exposure configurations are observed. |

## Risk Rating Criteria

| Severity | Definition |
|---|---|
| Critical | Directly/easily exploitable indicator with serious data/access impact. |
| High | Prominent, priority-to-fix strong indicator. |
| Medium | Requires attention; context-dependent verification recommended. |
| Low | Hardening/maturity opportunity; low priority. |

*Severity is a band label only (no numeric CVSS score); all findings are framed as “indicator, verification required”.*

## 4. AI Fix Suggestions

This section contains remediation recommendations for the findings detected in the authenticated checks that were run.

### Session Cookie Flags

### Cookie Security — proactive hardening

- Proactively enforce Secure/HttpOnly/SameSite flags on session cookies.

### Session Fixation

### Session Fixation — proactive hardening

- Make session-id renewal (session regeneration) standard on login.

### Logout / Session Invalidation

### Session Invalidation — remediation

- **Invalidate the session token server-side** on logout (revocation/expiry); client-side token deletion alone is not enough.

### Forced Browsing / Function-Level Authorization

### Function-Level Authorization — remediation

- Apply a **server-side role/permission check** on every management/sensitive endpoint; hiding in the UI alone is not enough.

### Authenticated Injection (SQLi/XSS)

### Injection (SQLi/XSS) — remediation

- **SQLi:** Write all database queries with **parameterised queries / prepared statements**; never concatenate user input into the query as a string. If you use an ORM, avoid raw SQL concatenation. Do not show database error messages to the end user.
- **XSS:** Apply **context-appropriate output encoding** (HTML entity encoding) when printing user input to HTML; where possible use a template engine that auto-escapes. Restrict inline scripts with a `Content-Security-Policy` header.
- After remediation, re-test the same input points.

### Authenticated IDOR (own resources)

### Broken Access Control (IDOR) — remediation

- **Object-level authorization:** On every resource access, verify server-side whether the requesting user has the right to access that object (a valid ID alone is not enough).
- **Unpredictable identifiers:** Use UUIDs/random identifiers instead of sequential numeric IDs; make enumeration harder.
- Place resources that should not be public behind authentication.

### Privilege Escalation

### Privilege Escalation / Mass-Assignment — proactive hardening

- Apply mass-assignment protection (field allowlist); do not accept role/privilege fields from the client (proactive).

### Multi-Step Business Logic

### Multi-Step Business Logic — proactive hardening

- Validate critical values server-side; apply step-order + coupon-reuse control (proactive).

### JWT / Token Security

### JWT / Token Security — proactive hardening

- Pin the signature algorithm, use a strong secret, add `exp` and carry no sensitive claims (proactive).

### Login Bypass (SQLi Indicator)

### Login Bypass / SQL Injection — remediation

- Use **parameterised queries / prepared statements** in authentication queries; never place user input directly into SQL.
- Input validation + safe ORM APIs; return a uniform error message on failed login.

### Client-Side / JS Analysis

### Client-Side / JS Security — proactive hardening

- Keep no secrets in client JS; do not publish source-maps in production; keep libraries current and apply dependency scanning (proactive).

### Client-Side Static Analysis

### Client-Side Static Security — remediation

- DOM-XSS: process user-controlled data with textContent + a safe API instead of innerHTML/eval/document.write; sanitise with DOMPurify if needed.
- postMessage: validate `event.origin` with an allowlist in every `message` handler.
- Storage: keep the session token in an **HttpOnly + Secure cookie** rather than localStorage.
- SRI: add `integrity` + `crossorigin` to external script/style. Tabnabbing: `rel="noopener noreferrer"` on `target="_blank"` links.
- Open redirect: restrict redirect targets server-side with an **allowlist**; do not redirect to external absolute URLs.

### Authentication Depth

### Authentication Hardening — remediation

- Enumeration: return a **uniform** response/message/timing on login/reset/register (do not leak whether a user exists).
- Default credentials: remove all default accounts / force a password change.
- Lockout: apply an **account + IP-based rate-limit / temporary lock** on consecutive failed attempts; add CAPTCHA.
- Password reset: **token-based** (single-use, short-lived), avoid security questions; do not use the Host header in the reset link.
- Password policy: **min 8+ / complexity** server-side; unencrypted channel: perform login **over HTTPS only**; `Cache-Control: no-store` for sensitive pages; offer **MFA**.

### Session Security Depth

### Session Security Hardening — proactive hardening

- Session cookie with SameSite + __Host- prefix, 128-bit random session id, no session in the URL, anti-CSRF token, reasonable timeout (proactive).

### Input & Header Depth

### Input & Header Hardening — proactive hardening

- Host allowlist, TRACE disabled, all resources HTTPS, output-encoding/sanitisation on persistent inputs, parameter de-duplication (proactive).

### Configuration & Exposure Depth

### Configuration & Exposure Hardening — proactive hardening

- Backup/.git/.env outside the directory, admin interface behind auth/network, bucket private/listing disabled, correct cache Vary, clean comments/metadata in production (proactive).

### API Security Depth (OWASP API Top 10)

### API Security Hardening — remediation

- BOLA/BFLA: enforce object-ownership + function-role checks server-side on every API request (ID validity is not enough).
- Excessive data: restrict responses with a **field allowlist (DTO)**; do not send password-hash/secret/internal-ID/role to the client.
- Rate-limit: **rate-limit + quota** per account+IP+endpoint; a cost-based limit for heavy endpoints.
- Version: disable unused/old API versions. GraphQL: **disable introspection in production**; add a query depth/complexity limit.

### E-mail & DNS Depth (Anti-Spoofing)

### E-mail & DNS Hardening — remediation

- DMARC: tighten gradually `p=none` → `quarantine` → `reject` (pct=100); enable reporting with `rua=` ; `sp=reject` for subdomains.
- SPF: use `-all` (hardfail); avoid `+all` / `?all` ; keep the DNS-lookup count ≤10 (PermError).
- DKIM: 2048-bit key, remove revoked selectors. MTA-STS: `mode=enforce` . Add TLS-RPT.
- Enable DNSSEC (DS+RRSIG). Restrict authorized CAs with CAA.

### Transport Layer, CORS & Security-Header Depth

### Transport Layer & Header Hardening — remediation

- CORS: allowlist origins; do not use a wildcard/reflection with `credentials` ; do not accept a `null` origin.
- TLS: leave only TLS 1.2/1.3; disable RC4/3DES/NULL/EXPORT ciphers; modern AEAD (ECDHE+AES-GCM/CHACHA20).
- HSTS `max-age≥15768000; includeSubDomains; preload` ; X-Frame-Options **and/or** CSP frame-ancestors against clickjacking.
- Remove unsafe-inline/unsafe-eval from CSP (nonce/hash); add Referrer-Policy/Permissions-Policy/X-Content-Type-Options.

### Subdomain Takeover (Dangling DNS)

### Subdomain Takeover — proactive hardening

- Subdomain CNAME inventory clean; no dangling/orphaned record (proactive).

## 5. Appendix — Glossary

| | |
|---|---|
| **SQL Injection** | An injection flaw where user input reaches a database query. |
| **XSS** | Cross-Site Scripting — injection of malicious scripts into pages. |
| **IDOR** | Insecure Direct Object Reference — accessing others' records via ID manipulation. |
| **CSRF** | Cross-Site Request Forgery. |
| **OWASP** | Open Web Application Security Project. |
| **TLS** | Transport Layer Security. |
| **HSTS** | HTTP Strict Transport Security header. |
| **CSP** | Content Security Policy. |

| | |
|---|---|
| **CORS** | Cross-Origin Resource Sharing. |
| **JWT** | JSON Web Token. |
| **Clickjacking** | Tricking clicks via invisible framing. |
| **Forced Browsing** | Accessing hidden endpoints by guessing URLs. |