

AI-POWERED SOURCE-CODE SECURITY ASSESSMENT

Executive summary

Harborline Supply Co. - fictional wholesale business
Company website: <https://harborline-supply.example>

This example review describes six issues in a customer ordering portal. Two need attention before the next release. One could let a buyer see another company's orders. The other could allow a used password reset link to work again. The remaining issues affect shared reports, large imports, alerts, and audit records.

Recommended decision for this example: Hold the release until SA-01 and SA-02 are fixed and checked. Assign an owner to every issue. Confirm the remaining risks in a safe test environment and agree on a plan before release.

2 HIGH	3 MEDIUM	1 LOW
--------	----------	-------

Assessment record

Report / review level	SAMPLE-HLS-20261002-01 / Reviewed Assessment
Codebase / repository no.	Harborline Buyer Portal / SAMPLE-042
Repository / branch	harborline-demo/buyer-portal / release/2026.10
Commit reviewed (fictional)	a825292988118a04645a3dd34359ed14f5b764c5
Example scan started	October 2, 2026 at 09:00 CDT (UTC-05:00)
Example scan completed	October 2, 2026 at 10:18 CDT (UTC-05:00)
AI engine / report owner	OpenAI Daybreak Blue / 417 Solutions, LLC
Example review date	October 2, 2026 at 14:30 CDT (UTC-05:00)

All client details, files, timestamps, code observations, and results are invented. This is a 417 Solutions sample deliverable, not an OpenAI-issued report or a record of an actual Daybreak Blue scan.

SCOPE AND PRIORITIES

What this review covers

The example application is a TypeScript / Node.js ordering portal with a PostgreSQL database. Buyers manage orders and downloads for their company. Catalog editors import stock data; administrators manage access.

Source code is required. This service reviews an agreed repository and code revision. A website address alone is not enough. This example contains no live-site attack, production access, or runtime test results.

Complete findings list

ID	Priority	Finding	Page
SA-01	High	Order lookup misses the company boundary	3
SA-02	High	Password reset links can be used more than once	3
SA-03	Medium	Shared export links never expire	4
SA-04	Medium	Catalog imports have no application size limit	4
SA-05	Medium	Alert changes lack a cross-site request check	5
SA-06	Low	Audit records omit who changed a report share	5

How to read the findings

High: Potential access to another account or company's data. **Medium:** Meaningful harm with additional conditions or a narrower reach. **Low:** A limited security or investigation gap. These are qualitative priorities for this fictional scenario, not measured scores.

Evidence: Each item illustrates a code observation, its limits, a fix, and an acceptance check. Every item remains open in this sample. No finding is claimed to be reproduced or fixed.

Boundaries and remaining questions

The illustrated scope includes first-party order, recovery, report-sharing, import, preference, and audit code at the listed revision. Hosting, network settings, third-party services, dependencies, and production data are outside this example. Confirm route exposure, cookie settings, request limits, and real role rules during an authorized follow-up. A code review cannot prove that an application has no other weaknesses.

TECHNICAL FINDINGS

High priority

Invented code observations. Paths and line numbers are illustrative.

HIGH | SA-01 | OPEN - EXAMPLE

Order lookup misses the company boundary

src/http/orders/read-order.ts:41-59
src/data/order-store.ts:73-88

Business impact. A signed-in buyer could read another company's order details, including delivery addresses and negotiated prices.

Code observation. In the example code, the request supplies an order ID. The handler checks for a session, then fetches the order using only that ID. The database query never compares the order's company ID with the company in the session.

Conditions and limits. Requires a valid buyer account and another order ID. Route exposure and real account permissions have not been tested.

Recommended fix. Bind every order lookup to the authenticated company. Keep the check in the data-access layer so alternate routes cannot skip it.

Acceptance check. Use two fictional companies in a test database. Each can read its own orders; a request for the other company's order must return no order data.

HIGH | SA-02 | OPEN - EXAMPLE

Password reset links can be used more than once

src/identity/recovery/complete-reset.ts:52-90

Business impact. Someone holding a used reset link could change the password again while that link remains valid.

Code observation. The example reset handler checks the stored token hash and expiry, then updates the password. It neither deletes the token nor marks it as used. A second submission follows the same successful branch.

Conditions and limits. Requires possession of an unexpired reset link; token guessing is not alleged. No email account or running application was accessed.

Recommended fix. Consume the reset token atomically with the password update. Reject tokens already used or expired, and invalidate existing sessions according to the account recovery policy.

Acceptance check. Submit one reset token twice, including two concurrent requests. Exactly one may succeed. Confirm that the former password and sessions behave as the recovery policy requires.

TECHNICAL FINDINGS

Medium priority

Invented code observations. Paths and line numbers are illustrative.

MEDIUM | SA-03 | OPEN - EXAMPLE

Shared export links never expire

src/reports/sharing/open-export.ts:28-61
src/reports/sharing/issue-link.ts:17-39

Business impact. A leaked report link could continue to expose a saved purchasing report long after it was needed.

Code observation. The example link generator stores a random token hash beside an export ID. The download route accepts a matching token without checking an expiry or revocation state. Anyone holding that link can retrieve the saved export.

Conditions and limits. The token is treated as unguessable. Risk depends on link disclosure and report contents; neither was observed in a live system.

Recommended fix. Add a short, explicit validity period and a revocation control. For sensitive reports, also require a session and company-level authorization.

Acceptance check. Confirm an allowed current link works. Expired and revoked links must return no file, and a user outside the company must not pass any required access check.

MEDIUM | SA-04 | OPEN - EXAMPLE

Catalog imports have no application size limit

src/catalog/import/read-upload.ts:35-78

Business impact. A large import could consume worker memory and delay other users' work.

Code observation. The example import flow loads the full uploaded file into memory and builds an array for every row. No application check limits file bytes, row count, or field length before parsing.

Conditions and limits. Requires catalog-editor access. An upstream request limit may reduce exposure; its configuration and worker memory limits are outside this code sample.

Recommended fix. Reject oversized input before parsing. Stream supported files, enforce row and field limits, and cap concurrent import jobs per company.

Acceptance check. Check files just below and above each agreed limit. Oversized files must fail cleanly without large memory growth; ordinary imports must still complete.

TECHNICAL FINDINGS

Medium and low priority

Invented code observations. Paths and line numbers are illustrative.

MEDIUM | SA-05 | OPEN - EXAMPLE

Alert changes lack a cross-site request check

src/http/preferences/save-alerts.ts:22-64

Business impact. A buyer could be tricked into changing notification settings, including turning off order-change alerts.

Code observation. The example POST route uses the session cookie and accepts ordinary form fields. It checks neither a per-session anti-forgery token nor the request origin before saving the settings.

Conditions and limits. A successful cross-site request depends on deployed cookie behavior and browser context. Those conditions have not been reproduced; runtime validation is required.

Recommended fix. Require an anti-forgery token on browser-driven changes and check allowed origins. Use a suitable SameSite cookie policy as an additional control.

Acceptance check. Reject missing and invalid tokens without changing settings. Confirm a valid same-site request succeeds, then test the deployed cookie policy in a safe test environment.

LOW | SA-06 | OPEN - EXAMPLE

Audit records omit who changed a report share

src/audit/record-sharing-event.ts:14-33

Business impact. The team may struggle to explain who enabled or removed report access during an investigation.

Code observation. The example audit writer records the time and event type, but leaves out the actor, company, affected report, and request identifier. Several share changes would produce indistinguishable events.

Conditions and limits. This is a traceability gap, not proof of unauthorized access. Other monitoring may supply some context; external log systems were not reviewed.

Recommended fix. Record stable actor, company, report, and request identifiers for each change. Restrict audit-log access and omit secret link tokens and report contents.

Acceptance check. Generate two changes by different test users. Each event must identify the correct actor and report, while containing no secret token or report data.

PRINTABLE WORKSHEET

Action item checklist

Harborline Supply Co. (fictional) | Repository SAMPLE-042
Report SAMPLE-HLS-20261002-01 | Example scan: October 2, 2026

Print this page for your release review. Check each box only after the fix and its acceptance check pass.
Record the evidence in your issue tracker.

1. Resolve the high-priority items before release

☐ **SA-01 - High.** Require company-scoped order access. Test allowed and denied lookups.
Owner: _____ Date: _____

☐ **SA-02 - High.** Make reset tokens single-use. Test repeat and concurrent submissions.
Owner: _____ Date: _____

2. Assign and verify the remaining fixes

☐ **SA-03 - Medium.** Expire and revoke report links. Test current, expired, and revoked access.
Owner: _____ Date: _____

☐ **SA-04 - Medium.** Bound import size and concurrency. Check memory use at the limits.
Owner: _____ Date: _____

☐ **SA-05 - Medium.** Protect alert changes from cross-site requests. Test unchanged data on rejection.
Owner: _____ Date: _____

☐ **SA-06 - Low.** Add safe audit context. Verify actor, report, and request IDs without secrets.
Owner: _____ Date: _____

3. Close the review

☐ Recheck the affected code at a new commit; record pass/fail evidence.

☐ Document unresolved items, an owner, and the next review date.

☐ Choose the next scan: each release, weekly, monthly, or on request.

Release owner: _____ Next review: _____
Retested commit: _____