

SAMPLE REPORT · ANONYMIZED

Backend Security Audit Report

A plain-English review of an application's backend against Golden Rock's 22-point checklist, with every finding ranked by real risk and a concrete fix for each.

CLIENT

Anonymized: a subscription web platform

WHAT WAS AUDITED

Customer-facing app, admin area, payment webhooks, background worker

STACK

Next.js, React, PostgreSQL, Stripe, Cloudflare, Docker

PACKAGE

Audit + Fix

AUDIT DATE

October 2026

AUDITOR

Golden Rock

6 / 22

PASSED BEFORE FIXES

21 / 22

PASSED AFTER FIXES

ISSUES FOUND (15 FIXED)

Critical	1
High	6
Medium	6
Low	3

This sample shows the format and depth of a Golden Rock audit. The findings come from a real engagement on a live production platform; identifying details have been removed. A real report also comes with an internal evidence log (file and line references, request and response records), which is not included here.

Summary

One critical and six high-severity issues were found, all fixed and re-checked. The most serious was in the admin area: its pages relied on a sign-in check that the framework can skip, so a crafted request could have read administrative data. The money path also had a gap: a payment that arrived during a crash could have left a paying customer without the service they paid for. There were no backups of customer records at all.

After the fixes, 21 of 22 checklist items pass. The remaining item (request IDs on every log line) is a low-priority improvement that isn't needed at this size.

How to read this report: findings are sorted by severity. "What we found" is the business impact in plain English with one technical detail; "Recommended fix" is one concrete action. **Critical** = data exposure or account takeover possible now. **High** = likely to cause a breach or outage. **Medium** = weakens defenses. **Low** = best practice.

Findings

#	CHECKLIST ITEM	SEVERITY	WHAT WE FOUND	RECOMMENDED FIX	EFFORT	RESULT
2	Auth enforced server-side	CRITICAL	Seven admin pages (clients, leads, approvals, call list, requests) relied on the admin layout's sign-in check. The framework can render a page without re-running its layout, so a crafted request carrying any cookie could have read admin data.	Every admin page checks for an admin session itself, on the server.	1 hour	Fail → Fixed
12	Transactions for multi-step writes	HIGH	When a payment arrived, "mark paid", "create the customer login" and "queue the launch" were three separate writes. A crash between them left a paying customer with no service, and the payment provider's automatic retry then skipped them as already handled.	One database transaction for all three; emails go out only after it commits.	2 hours	Fail → Fixed
16	Backups with a tested restore	HIGH	No backups at all of customer records, payments or uploaded files.	Nightly automated backups of the database, files and job data, an automatic restore test after every run, 14 days kept, plus an off-site copy.	3 hours	Fail → Fixed
4	Rate-limited sign-in	HIGH	The sign-in form would email unlimited sign-in links to any customer's address: inbox flooding, and it burns the email sending quota.	3 links per address per 15 minutes and 10 requests per visitor; checkout and search limited too.	1 hour	Fail → Fixed

#	CHECKLIST ITEM	SEVERITY	WHAT WE FOUND	RECOMMENDED FIX	EFFORT	RESULT
10	Least-privilege database user	HIGH	The website connected to its database as the all-powerful owner account.	A limited login that can read and write rows but not change the database's structure; migrations keep a separate owner login.	2 hours	Fail → Fixed
8	Size limits on requests	HIGH	Four public form endpoints would read a request body of any size.	Bodies capped (16 KB, or 64 KB for the chat widget); oversized requests get a "413 too large".	1 hour	Fail → Fixed
22	Dependencies updated and scanned	HIGH	Ten high-severity advisories in production dependencies (an image library inside the deployment tool; a build-only tool shipped to production).	Updated the deployment tool; moved build-only tools out of production. Production audit: 0 vulnerabilities.	30 min	Fail → Fixed
11	CORS locked down	MEDIUM	The lead-capture endpoint answered any website, so another site could make visitors' browsers submit fake leads.	Exact allowlists per endpoint; browser requests from other sites refused.	1 hour	Fail → Fixed
14	Indexes on filtered fields	MEDIUM	No indexes on foreign keys or on the job queue the worker checks every 5 seconds: fine today, slow as data grows.	Eight indexes added in a versioned migration.	30 min	Fail → Fixed
17	Timeouts on outside calls	MEDIUM	Calls to the DNS provider, payment provider, email server and database had no time limit and could hang a checkout or the worker indefinitely.	Timeouts on all four; payment retries use idempotency keys; 15-second database statement limit.	1 hour	Fail → Fixed
6	CSRF protection	MEDIUM	Sign-out (a custom route) accepted requests from any website. Built-in form actions were already protected.	Sign-out refuses requests from other origins.	15 min	Fail → Fixed
1	Per-object authorization	MEDIUM	A public order-confirmation page, whose address is built from the business name, showed the customer's full email to anyone who guessed it.	Email masked (j...@gmail.com). Uploads and dashboards were already scoped to the signed-in customer.	15 min	Fail → Fixed
7	Validate everything coming in	MEDIUM	One admin field (a status value) wasn't validated. Public forms, checkout and uploads already used schemas.	Schema validation; note length capped.	15 min	Fail → Fixed

#	CHECKLIST ITEM	SEVERITY	WHAT WE FOUND	RECOMMENDED FIX	EFFORT	RESULT
18	Correct overload responses	LOW	Rate-limited responses didn't say when to retry; one failure returned the wrong status.	"429" plus Retry-After for limits; "503" plus Retry-After for temporary failures.	30 min	Fail → Fixed
19	Health check tests the database	LOW	No health endpoint for an uptime monitor.	/api/health runs a database check and returns 503 when it's down; an external monitor checks it every 5 minutes.	15 min	Fail → Fixed
20	Safe logs	LOW	No tokens or passwords logged. Errors shown to users carry a reference id, but ordinary log lines have no request id.	Fine at this size; add a request-id header if traffic grows.	—	Partial

Items that already passed

#	CHECKLIST ITEM	WHAT WE VERIFIED
3	Hashed passwords	N/A: passwordless sign-in. Sign-in links and sessions are 256-bit random values stored only as SHA-256 hashes.
5	Single-use, expiring sign-in links	Links are consumed atomically (a second use fails) and expire after 20 minutes.
9	Secrets outside the code	Secrets live only in a restricted environment file excluded from images; no secret patterns in the source or its history.
13	Idempotent payments	Webhook signatures verified on the raw body; events de-duplicated by id; fulfillment driven by the webhook, never the success page.
15	Versioned migrations	Schema changes live in numbered migration files applied automatically on deploy.
21	No leaky errors	Production build; responses never include stack traces, database errors or file paths.

Other observations (not scored)

- Security headers added on every response (HSTS, nosniff, frame protection, referrer and permissions policies); the framework's version banner removed.
- A background job interrupted by a restart now resumes automatically after 15 minutes instead of sticking forever.
- Expired sessions and sign-in links are cleaned up hourly.
- Recommended: corporate email scanners can use up a sign-in link before the customer clicks it. If customers report "link expired", switch to a confirm button.

Fixed during this engagement

All fifteen failed items were fixed in code, deployed, and re-checked with twenty automated checks against the live system (security headers, rate limits, size limits, origin checks, the limited database login, and the new indexes). Both database passwords were rotated afterwards, because the old ones had appeared in a work log during the audit.

How the audit works

Every audit walks the same 22-point checklist. Each item gets exactly one result (Pass, Fail, Partial or N/A), backed by evidence: file and line references, redacted configuration, or recorded requests and responses. Live tests run only against a staging copy, with written permission and test accounts. No production data, no load testing, nothing destructive.

Access

1. Per-object authorization (one user can't reach another's data)
2. Auth enforced on the server
3. Hashed passwords
4. Rate-limited sign-in and reset
5. Single-use, expiring reset links
6. CSRF protection

Data

12. Transactions for multi-step writes
13. Idempotent payments
14. Indexes on filtered fields
15. Versioned migrations
16. Backups with a tested restore

Input

7. Validate everything coming in
8. Size limits and page caps

Secrets

9. Secrets outside the code
10. Least-privilege database user
11. CORS locked down

Reliability

17. Timeouts on outside calls
18. Correct overload responses
19. Health check tests the database

Visibility and upkeep

20. Safe logs
21. No leaky errors
22. Dependencies updated and scanned

Packages. Quick Scan: the five highest-risk items (1, 2, 3, 7, 9) from a code and configuration review. **Full Audit:** all 22 items, code plus live testing on staging. **Audit + Fix:** the full audit, then the failed items fixed and re-checked.

Scope and limits: an audit covers the listed items as of the audit date. It is not a guarantee that no other vulnerabilities exist. This document is a sample with identifying details removed. Prepared by Golden Rock.