Rescue Audit: Sample Report

Segment Holdings · PHP / Laravel / Statamic Rescue Audit · Fixed price $500

> **This is an anonymized sample report.** The app audited here is an early-stage SaaS product that had not yet launched. We audited it exactly as we would audit a client's app, so you can see what the $500 Rescue Audit delivers. Names, file paths, advisory IDs, exact versions and other identifying details have been removed or generalized, and exploit-level detail has been left out. Nothing in the repository was changed during the audit.

Application	A Laravel 12 + Statamic SaaS that turns RSS feeds into scheduled newsletter digests for a subscriber list
Repository	Private repository, main branch
Commit reviewed	Latest commit on the main branch at the time of the audit (about seven months old)
Stack	PHP · Laravel 12 · Statamic 6 (marketing site) · Filament 5 (customer app) · Livewire 4 · Laravel Cashier (Stripe) · Spatie Media Library · Vite / Tailwind 4
Audit date	October 31, 2026
Method	Read-only review of the source code, Composer and npm lock files, configuration, tests and repository settings. We checked the locked versions against the OSV.dev / GitHub Advisory Database and ran `npm audit --package-lock-only`. We compared versions against Packagist and the official PHP, Laravel and Statamic support policies.

1. Executive summary (plain English)

This is a tidy, modern codebase. It runs on current framework versions, the code is well organized, customer data is scoped to the logged-in account on every screen we checked, and there is a meaningful automated test suite (19 test files). It was clearly built by someone who knows Laravel.

It is **not ready to take money from customers yet**. These are the four things a business owner needs to know:

1. **The software it runs on has known security holes.** Since the dependencies were last updated, **about 100 security advisories** have been published that affect the exact versions it uses, **2 rated critical and 29 high**, including an authentication-related issue in the content admin. All of the important ones are fixed upstream within the same major versions, so this is a few hours of careful updating and testing, not a rewrite.
2. **A billing shortcut could give away paid plans for free.** Outside a correctly configured production environment, choosing a plan creates a paid subscription without charging a card. One configuration mistake would hand out free accounts.
3. **The core product isn't built yet.** Customers can add feeds, subscribers and a sending schedule, and they can pay. But no code fetches the feeds or sends the digests, and subscribers have no page to sign up, confirm or unsubscribe. Today the app charges for something it can't deliver.

4. **A database rule mixes up customers.** A subscriber's email address can exist only once across the *whole system*. If two customers have the same subscriber, the second customer gets an error. That breaks imports and reveals that the address is on someone else's list.

Bottom line: about **15–26 hours** of work (estimated) closes every critical and high-risk security item and makes the app safe to keep developing. After that, the sending engine (about 30–50 hours, estimated) is what turns this into a product people can pay for.

---

2. Risk scorecard

Severity	Count	What it means
🔴 **Critical**	**1**	Exploitable now with serious impact. Fix before going live.
🟠 **High**	**4**	Significant business, security or data-integrity risk. Fix before launch.
🟡 **Medium**	**9**	Real risk or cost that grows over time. Schedule soon.
🟢 **Low**	**7**	Hygiene and maintainability. Fix during normal work.
Total	**21**	

Platform support at a glance (sources: php.net supported versions, Laravel support policy, Statamic support policy):

Component	In use	Bug fixes until	Security fixes until	Status
PHP (declared minimum)	8.2	ended Dec 31, 2024	**Dec 31, 2026**	Attention Security support ends in 2 months
PHP (actually required by the lock file)	≥ 8.4	Dec 31, 2026	Dec 31, 2028	OK OK, but doesn't match the declared minimum (see M-1)
Laravel	12.x (well behind the latest 12.x release)	**ended Aug 13, 2026**	Feb 24, 2027	Attention Security fixes only. Plan the Laravel 13 upgrade.
Statamic	6.x (dozens of minor releases behind)	Mar 2027	Dec 2027	OK Supported major, but far behind on patches
Filament	5.x (several minor releases behind)	current major	current major	OK Supported major, behind on patches

---

3. Findings

*Effort figures are **estimates** in engineering hours for a developer who knows the stack. Final scope is confirmed in a fixed-price quote. In the client version of this report, every finding cites exact files and line numbers; those references are generalized here.*

Summary table

ID	Severity	Area	Finding	Evidence	Fix	Est. hours
C-1	🔴 Critical	Security / dependencies	About 100 published advisories affect locked versions (2 critical, 29 high), including an auth-related issue in the CMS fixed upstream	Composer lock file	Update to patched versions within current majors, retest, add `composer audit` to CI	6–10
H-1	🟠 High	Billing	Non-production environments grant free active subscriptions	Plan selection component; example environment file	Gate the test path behind an explicit, default-off flag that can never be on in production; use Stripe test mode instead	2–3
H-2	🟠 High	Data model / multi-tenancy	Subscriber email is globally unique across all customers	Subscribers migration; subscriber importer	Composite unique key per project plus validation	2–4
H-3	🟠 High	Product completeness	No feed fetching, digest sending, scheduling, or public subscribe/confirm/unsubscribe flow exists	Console and web route files; no jobs, mailables or HTTP calls in the app code	Build the delivery engine (see Fix Plan)	30–50
H-4	🟠 High	Auth / deployment	The production seeder creates a real account with the default factory password	Production seeder; user factory	Remove it, or require a random/prompted password, and send a reset link instead	0.5–1
M-1	🟡 Medium	Platform	Composer declares PHP 8.2, but locked packages require PHP ≥ 8.4. PHP 8.2 security support ends Dec 31, 2026	`composer.json`; Composer lock file	Declare 8.4, set the platform PHP version, and pin the server PHP	0.5–1
M-2	🟡 Medium	Platform	Laravel 12 bug-fix support ended Aug 13, 2026; security support ends Feb 24, 2027	`composer.json`	Plan the Laravel 13 upgrade (Statamic 6 supports Laravel 12–13)	6–12
M-3	🟡 Medium	Security / secrets	Saved email-provider credentials are decrypted and sent back to the browser in the settings form	Project settings form	Never pre-fill secret fields; only overwrite when a new value is typed	2–3
M-4	🟡 Medium	Ops / quality	No CI: tests never run automatically; the automated dependency-update PR queue is full and stalled	Repository settings; dependency-bot config	CI pipeline: code style, tests, `composer audit`, `npm audit`	3–4
M-5	🟡 Medium	Security / dependencies	`npm audit`: 23 vulnerabilities (19 high, 3 moderate, 1 low), including a headless-browser package that runs on the server	npm manifest and lock file	Update the affected packages to patched versions, then re-audit	2–4
M-6	🟡 Medium	Compliance (email)	Subscribers can be imported or marked "confirmed" without opt-in, and no unsubscribe endpoint exists	Subscribers table actions; subscriber importer; web routes	Double opt-in for imports, signed unsubscribe links, `List-Unsubscribe` headers	4–8

ID	Severity	Area	Finding	Evidence	Fix	Est. hours
M-7	🟡 Medium	Billing / revenue	Advertised plan limits (feeds, subscribers) are never enforced, and the overage setting is saved but never read	Pricing-tier content; feed management page; billing page	Central plan-limits service checked on create/import	4–6
M-8	🟡 Medium	Portability	Dashboard charts use MySQL-only SQL, but the default and documented production DB is SQLite	Dashboard chart widgets; example environment file; README	Group in PHP or use driver-aware SQL; run tests on the production DB engine	1–2
M-9	🟡 Medium	Deployment / ops	Starter-template README and example environment file: no payment keys/webhook secret, queue worker or scheduler steps; sync queue; debug-level production logging	README; example environment file; billing config	Real runbook, complete example environment file, queue worker and cron, error monitoring	3–5
L-1	🟢 Low	Bug	Chart month labels skip and duplicate months at the end of long months	Dashboard chart widgets	Use overflow-safe date math	0.5
L-2	🟢 Low	Repo hygiene	A personal AI coding-assistant settings file is committed, with a local machine path and broad pre-approved shell permissions	Committed local settings file; `.gitignore`	Remove it from the repo and ignore it	0.25
L-3	🟢 Low	Repo hygiene	Stale build artifacts are committed, including files the project already ignores	Build config; `.gitignore`	Delete the tracked artifacts and keep builds in CI/deploy	0.5
L-4	🟢 Low	Authorization	No authorization policies; tenant isolation relies only on query scoping, and owner keys are mass-assignable	Admin resource; core models	Add policies (defense in depth) and drop owner keys from mass assignment	1–2
L-5	🟢 Low	Code quality	No static analysis (PHPStan/Larastan); dead code and leftovers	`composer.json`; unused form class; service provider	Add Larastan at level 5+ to CI and remove dead code	2–4
L-6	🟢 Low	Security (future)	Feed URLs and customer mail hosts accept any address, a future server-side request risk once fetching/sending exists	Feed form; project settings form	Block private, loopback and metadata addresses at fetch/connect time	2–3
L-7	🟢 Low	Billing / robustness	Plan selection trusts input from the browser, and plan swaps don't handle payment failures	Plan selection component	Validate against the configured plans and handle incomplete payments	0.5–1

Estimated total: about 73–124 hours. About 30–50 of those are building the delivery engine (H-3).

Detailed findings

C-1 · Critical · Known-vulnerable dependency versions

The lock file hadn't been updated in about seven months. In that time, advisories have been published against the exact locked versions. As of October 31, 2026, OSV.dev / the GitHub Advisory Database list **about 100 advisories across two dozen locked Composer packages** (2 critical, 29 high, the rest moderate or low). The ones that matter most:

Package	What the advisories cover	Status
Statamic CMS	A critical authentication-related issue affecting Control Panel accounts, plus several high-severity issues (privilege escalation, code execution reachable by Control Panel users, stored XSS). About 30 advisories in total.	Fixed upstream in later 6.x releases
Laravel framework	A high-severity input-validation issue relevant to how this app handles email addresses.	Fixed upstream in a later 12.x release
Filament	High-severity XSS plus lower-severity issues on the authentication pages.	Fixed upstream in later 5.x releases
Spatie Media Library	A high-severity upload-restriction issue relevant to the app's logo upload, plus a lower-severity request-forgery issue.	Fixed upstream in a later 11.x release
Supporting libraries (Markdown, HTTP client, mail and HTTP components, Livewire, others)	XSS filter, header-injection and denial-of-service issues	Fixed upstream

Not applicable here (checked): We also confirmed that some headline advisories, including a critical one in a framework component this app doesn't use, don't affect this app's configuration. A good audit separates "listed" from "exploitable here."

Fix: Upgrade to the patched versions within the current majors (no breaking upgrades needed), retest, and add `composer audit` to CI so new advisories fail the build. **Est. 6–10 h.**

H-1 · High · Free subscriptions outside "production"

The plan selection component charges through Stripe only when the app believes it's in production. In every other environment it writes an active subscription straight to the database without charging, and plan changes skip billing too. The example environment file ships a non-production value. If the live server's environment setting is ever wrong or missing, anyone who registers gets a paid plan for free, and the subscription check lets them in.

Fix: Delete the test path, or put it behind an explicit, default-off flag that refuses to run in production. Use Stripe test-mode keys for local testing. **Est. 2–3 h.**

H-2 · High · Subscriber emails are unique across all customers

The subscribers table puts a unique index on the email column by itself, so an address is unique across **every customer's list**, not per project. So:

- If two customers (or two projects of the same customer) share a subscriber, the second insert fails with a database error.
- The importer looks up the subscriber within the current project, finds nothing, tries to insert, and fails. Imports quietly drop rows that exist elsewhere.
- The error tells one customer that the address is on another customer's list.

Fix: Add a migration for a composite unique key on project and email, add matching validation to the form and importer, and add tests. **Est. 2–4 h.**

H-3 · High · The product's delivery engine doesn't exist yet

The dashboard shows a "Next Digest" time, a "Digests Sent" count, and per-project mail-driver settings. But in the code we reviewed:

- The console routes contain only the framework's sample command, with no scheduled tasks.
- The web routes define no public subscribe, confirm or unsubscribe pages.
- The app code contains no jobs, no mailables or notifications, and no HTTP client calls. Nothing fetches feeds, records fetch times, creates digest records, or uses the stored mail credentials.

Customers can subscribe and pay through Stripe, but the service can't fetch feeds or send digests.

Fix: Build the engine: a scheduled dispatcher, a queued feed fetcher (with request guards, see L-6), digest assembly, a per-project mailer built from the saved mail settings, signed confirm/unsubscribe routes (M-6), plan-limit checks (M-7), and tests. **Est. 30–50 h** (scoped separately).

H-4 · High · Production seeder creates a known-password account

The main database seeder always runs a production seeder, which creates a real admin account through the user factory, and the factory uses the framework's well-known default password. One seed command on a live server creates an account anyone could guess.

Fix: Remove the production seeder, or have it prompt for or generate a strong password and send a reset link. **Est. 0.5–1 h.**

M-1 · Medium · PHP version declared vs. required

`composer.json` declares PHP 8.2, but some locked components need PHP ≥ 8.4, and Statamic 6 supports PHP 8.3–8.5. A server on 8.2 or 8.3 will fail at `composer install`, and PHP 8.2 security support ends **December 31, 2026**.

Fix: Declare 8.4, set Composer's platform PHP version to match the server, and pin the server's PHP in the runbook. **Est. 0.5–1 h.**

M-2 · Medium · Laravel 12 is in its security-only window

Laravel 12 bug fixes ended **August 13, 2026**, and security fixes end **February 24, 2027**. Laravel 13 needs PHP ≥ 8.3. Statamic 6 already supports it.

Fix: Upgrade to Laravel 13 after C-1 and M-4 are in place, so the test suite can guard the upgrade. **Est. 6–12 h.**

M-3 · Medium · Stored email-provider secrets go back to the browser

Mail credentials are encrypted at rest, which is good. But the settings form binds the password, API key and token fields straight to the decrypted values. The admin framework fills forms from the database record, so each

saved secret is embedded in the page every time the settings screen loads, and a "reveal" toggle shows it on screen.

Fix: Never pre-fill secret fields, save them only when a new value is entered, and show "•••• saved" instead. **Est. 2–3 h.**

M-4 · Medium · No continuous integration

The repository has no CI workflows. The 19 test files never run on push, and nothing blocks a broken merge. Automated dependency updates are configured with a cap on open pull requests, and the front-end side has hit that cap with months-old untested PRs, so new updates, including security updates, can't open.

Fix: Add a CI workflow that runs code style checks, the test suite on the production database engine, `composer audit` and `npm audit`, then work through the dependency-update backlog. **Est. 3–4 h.**

M-5 · Medium · Front-end and Node dependencies

`npm audit --package-lock-only` reports **23 vulnerabilities: 19 high, 3 moderate, 1 low, 0 critical**. Most affect build-time tools only. The exception is a headless-browser package used at *runtime* on the server for screenshots, whose dependency chain carries several high-severity advisories. An HTTP client pulled in by a form-validation helper also carries many advisories.

Fix: Upgrade the affected packages to patched versions, then rerun the audit. **Est. 2–4 h.**

M-6 · Medium · Email consent and unsubscribe

For a product that sends bulk email:

- Any customer can mark any subscriber "Confirmed", one at a time or in bulk.
- CSV imports have no opt-in step.
- There is no public unsubscribe or confirm route.

Sending to unconsented lists without a working unsubscribe link is a CAN-SPAM and GDPR exposure, and it's the fastest way to get the sending domain blocklisted.

Fix: Send a confirmation email for imported and manually added subscribers (or require a recorded consent source), add signed one-click unsubscribe links and `List-Unsubscribe` headers, and log consent. **Est. 4–8 h.**

M-7 · Medium · Plan limits aren't enforced

The entry-level tier advertises a feed limit and a subscriber limit. No code checks either: the "create feed" button is always available, and imports have no cap. The "allow overage" setting is saved but nothing reads it.

Fix: Add a plan-limits service, keyed by the Stripe price, that is checked on feed create, subscriber create and import, and connect overage to metered billing. **Est. 4–6 h.**

M-8 · Medium · MySQL-only SQL with a SQLite default

Both dashboard charts group with a date function that exists only in MySQL. The example environment file and the README's production example both use SQLite. A fresh install that follows the docs gets a broken dashboard.

Fix: Group by month in PHP, or branch on the database driver. Pick one production database and run the tests on it in CI. **Est. 1–2 h.**

M-9 · Medium · Deployment and operations gaps

- The README is still the starter template. It doesn't mention the queue worker (which imports and exports need), the scheduler cron, or the Stripe webhook.
- The example environment file still carries starter-kit placeholders and uses the synchronous queue driver.
- It omits the Stripe keys, webhook secret and price variables the billing config needs.
- The README's production example logs at debug level.
- No error monitoring or uptime checks are configured.

Fix: Write a one-page runbook, complete the example environment file, add queue worker and cron config, verify the Stripe webhook, and add error monitoring (e.g. Nightwatch or Sentry). **Est. 3–5 h.**

L-1 to L-7 · Low

- **L-1 · Chart month bug.** Month subtraction overflows at the end of long months, so on the day of the audit the charts showed one month twice and skipped the previous one. *0.5 h.*
- **L-2 · Committed assistant settings.** A personal AI coding-assistant settings file is committed. It exposes a local machine path and username and pre-approves broad shell commands for anyone who opens the repo with that tool. Remove it and ignore it. *0.25 h.*
- **L-3 · Stale artifacts.** An old build manifest and image files are tracked even though the build now writes elsewhere and the folder is ignored. *0.5 h.*
- **L-4 · Authorization depth.** Isolation relies on query scoping in the admin resource. That works today, but authorization policies would protect future pages, actions and APIs. Owner keys are mass-assignable on the core models. *1–2 h.*
- **L-5 · No static analysis and some dead code.** There is no PHPStan/Larastan in the dev dependencies, and there are unused classes and constants. *2–4 h.*
- **L-6 · Request-forgery readiness.** Feed URLs and customer mail hosts accept any address. Once fetching and sending exist, block private, loopback and cloud-metadata addresses. *2–3 h.*
- **L-7 · Plan selection input.** The plan selection action accepts any value from the browser, and plan swaps don't handle incomplete payments. *0.5–1 h.*

What's already good

- Every customer-facing query we reviewed is scoped to the logged-in user, including the main admin resource and all dashboard widgets.
- Mail credentials are encrypted at rest.
- CSRF protection, session authentication, email verification and login throttling are all on.
- Real feature tests cover billing, dashboards, projects, feeds, subscribers and models.
- No `.env` file, API keys or payment secrets are committed in the files we reviewed.
- The code is modern and well organized: form schemas and tables live in separate classes, and enums are used for drivers and frequency.

---

4. Prioritized fix plan

Now: make it safe (week 1, est. 15–26 h)

1. **C-1:** update the CMS, framework, admin panel, media library and the other flagged packages to patched versions within their majors, and retest.
2. **H-1:** remove the free-subscription path.

3. **H-4:** remove the known-password production seeder.
4. **H-2:** fix subscriber uniqueness per project.
5. **M-1:** align the PHP version constraint with the server.
6. **M-3:** stop sending stored secrets to the browser.
7. **M-5:** update the vulnerable npm packages.
8. **L-2:** remove the committed assistant settings file.

Next: make it a product (weeks 2–6, est. 48–80 h)

1. **M-4:** CI with tests and dependency audits, then clear the dependency-update backlog.
2. **H-3:** build the delivery engine (fetch → assemble → send → record), with **M-6** consent and unsubscribe, **M-7** plan limits and **L-6** request guards built in from the start.
3. **M-8:** pick the production database and fix the chart SQL. Also fix the **L-1** month bug.
4. **M-9:** runbook, complete example environment file, queue worker, scheduler, Stripe webhook and monitoring.
5. **L-7:** validate plan input.

Later: keep it healthy (before Feb 2027, est. 10–19 h)

1. **M-2:** upgrade to Laravel 13 (Laravel 12 security support ends Feb 24, 2027).
2. **L-4, L-5, L-3:** policies, Larastan in CI, dead-code and build-artifact cleanup.
3. A monthly dependency and advisory review so the backlog doesn't build up again.

---

5. What happens next

Upgrade & Rescue: fixed price, from $2,500. We take the **Now** list (and as much of **Next** as you choose) and deliver it as reviewed pull requests with tests, a clean `composer audit` / `npm audit`, and a short handover note. You approve a fixed scope and price before any work starts, so there are no hourly surprises.

Care Plan: $250–$400 / month. Once the app is healthy, we keep it that way: monthly framework and security updates, a monitored advisory feed, CI kept green, uptime and error monitoring, and a small block of hours for fixes and small improvements. Backlogs like the one in this report never build up again.

Want this audit for your own PHP, Laravel or Statamic app? Email Segment Holdings at hello@segment.holdings.

Methodology and limits. This was a static, read-only review of the repository at its latest commit, using read-only repository access. We didn't run the application, connect to any server, or test against a live database. We have no access to the production configuration, so items about environment settings describe what the repository documents and makes possible, not a confirmed live setting. We reviewed all application code, routes, seeders, key configuration files, selected migrations and tests, and both lock files. We didn't review full git history for past secrets. Advisory counts come from OSV.dev / GitHub Advisory Database queries against the exact locked versions on October 31, 2026, and from `npm audit --package-lock-only` on the same date. Severity labels in Section 3 are Segment Holdings' assessment for this app; advisory severities are as published. Effort figures are estimates, not quotes. This public sample has been anonymized: identifying names, paths, versions and advisory IDs are omitted, and the client version includes exact file and line references for every finding. © 2026 Segment Holdings LLC · hello@segment.holdings