Sample report: public-surface review of aisend.app

What this is. A real review we ran on our own site, aisend.app (a transactional email API made by the same maker as Bugsly), on 2026-11-26. It is a sample of the format and depth of the report you get. It covers only what anyone can see from outside: page headers, robots.txt, sitemap, cookies, public JavaScript, DNS and TLS. It did not log in, did not submit anything, did not send any payload, and did not look at source code.

What you get on top, in the paid review. We read your repo (read-only invite or a zip, no credentials): secret scan including git history, a route-by-route check of who can call what, cross-tenant and row-level-security checks, a dependency audit from your lockfile, and config and error-handling checks. Every finding is read by a person before the report is sent. This sample has none of those, and we say so below.

What this is not. It is a list of findings, not a security guarantee. A clean report does not mean your app is safe, and we cannot promise it finds everything.

Made with AI tools, checked by a person before it was published. Raw outputs are in the evidence file.

Summary

No High findings. 2 Medium, 3 Low, 3 Info. 9 checks passed. Every item below comes from a request or a command listed in the evidence file.

Severity key: **High** = anyone can abuse it now. **Medium** = weakens a control that matters for this kind of app. **Low** = hardening. **Info** = good to know, no action needed.

#	Severity	Finding	Effort to fix
F1	Medium	No Content-Security-Policy header on the website	1-2 h (report-only first)
F2	Medium	DMARC is `p=none` on aisend.app, SPF ends in `~all`	30 min, then wait for reports
F3	Low	Dashboard routes return an indexable page shell to anonymous visitors	30 min
F4	Low	HSTS on the website is weaker than on the API host	5 min
F5	Low	`X-Powered-By: Next.js` and the old `X-XSS-Protection: 1; mode=block` header on the website	5 min
F6	Info	No `security.txt`, no CAA record; certificate ends 2027-01-18	15 min
F7	Info	Attribution cookie `ais_attr` has no `HttpOnly`	none unless a script does not need it
F8	Info	The API root answers without a key and lists its routes	none, decide if intended

Findings

F1. No Content-Security-Policy (Medium)

What we saw. None of the 7 pages and headers we fetched (home, /login, /overview, /admin, one sitemap URL, home with a foreign Origin header, the API root) sent a `Content-Security-Policy` header. See evidence section 4.

Why it matters here. The site has a signed-in area where users create and see API keys. If a script ever got injected through a dependency, a blog template or a third-party tag, a CSP is the control that limits what it can load and where it can send data.

Fix prompt for your own AI agent: "Add a Content-Security-Policy to this Next.js app in middleware, with a per-request nonce for scripts. Start with `Content-Security-Policy-Report-Only`, list the origins we really use (our own domain, the analytics host, the API host), run it for a week, read the reports, then switch to enforcing. Do not use `unsafe-inline` for scripts."

What we could not check from outside. Whether any page injects user-supplied HTML. That needs the code.

F2. DMARC `p=none` and SPF `~all` (Medium)

What we saw. `_dmarc.aisend.app` is `v=DMARC1; p=none; rua=mailto:dmarc@aisend.app; fo=1`. The SPF record on aisend.app is `v=spf1 include:amazonses.com include:_spf.mx.cloudflare.net ~all`. See evidence section 7.

Why it matters here. `p=none` means receivers are told to only report, not to reject mail that fails the checks. For a product that sends email, a forged mail from the company domain would still be delivered by many receivers. The `rua` address means reports are being collected, which is the right first step.

Fix prompt: "Read the last 4 weeks of DMARC aggregate reports sent to dmarc@aisend.app. List every sender that fails alignment. Fix those, then move the policy to `p=quarantine; pct=25`, then 100, then `p=reject`. Change SPF from `~all` to `-all` only after the sender list is confirmed."

What we could not check. The content of the reports, and which senders you really use. We only read public DNS.

F3. Dashboard routes return an indexable page shell to anonymous visitors (Low)

What we saw. An anonymous GET of `/overview` and `/admin` (both named in robots.txt) returns HTTP 200 and an HTML shell with the dashboard menu. `/admin` shows "Loading...". Both pages carry `<meta name="robots" content="index, follow">`, the same title as the homepage and no canonical tag. `/login` is the same. robots.txt has `Disallow` for these paths, which stops crawling but does not stop a linked URL from being indexed. See evidence section 5.

Why it matters here. The shell held no account data in the HTML we received, so we are not claiming a data leak. The risk is duplicate or junk pages in search results, and one question we cannot answer from outside: does the server refuse the data calls that the shell makes without a session? That is the route-auth sweep, and it needs the repo.

Fix prompt: "For every route under the signed-in area, set `robots: { index: false, follow: false }` in its metadata and a unique title. Then write a test that calls every API route that the dashboard uses with no session and with a session of a different organization, and expects 401 or 403."

F4. HSTS differs between the website and the API host (Low)

What we saw. Website: `strict-transport-security: max-age=31536000`. API host: `max-age=63072000; includeSubDomains; preload`.

Fix prompt: "Set the website HSTS to `max-age=63072000; includeSubDomains; preload` once every subdomain is HTTPS-only. Check the subdomains first, because `includeSubDomains` also covers them."

F5. Version and legacy headers on the website (Low)

What we saw. `X-Powered-By: Next.js` and `X-XSS-Protection: 1; mode=block` on the website. The API host sends `X-XSS-Protection: 0`, which is the current advice.

Fix prompt: "In next.config set `poweredByHeader: false`, and send `X-XSS-Protection: 0` (or drop it) on all responses."

F6. No security.txt, no CAA, certificate dates (Info)

What we saw. `/.well-known/security.txt` returns 404. `dig CAA aisend.app` returns nothing. The certificate is from Let's Encrypt and is valid until 2027-01-18. We did not check how it is renewed.

Suggestion. Add a `security.txt` with a contact address so a researcher knows where to write. Add a CAA record for the one CA you use. Confirm that the renewal job runs.

F7. Attribution cookie without HttpOnly (Info)

What we saw. `ais_attr` is set with `Secure; SameSite=lax; Max-Age=7776000` and no `HttpOnly`. The value is the landing path and a timestamp, not a session or a user id. Not sensitive.

Suggestion. Add `HttpOnly` only if no client script reads this cookie. If one does, leave it.

F8. API root lists its routes without a key (Info)

What we saw. `GET https://api.aisend.app/` returns JSON with the product name, version and a list of routes. We did not call any of those routes. We did not compare the list with the public docs.

Suggestion. Fine if intended. If the list is not meant to be public, trim it.

What we checked and found nothing

Check	Result	Source
HTTP to HTTPS	`http://aisend.app/` answers 301 to `https://aisend.app/`	evidence 1
Exposed files	`/.env` and `/.git/HEAD` answer 404 (one request each, nothing else guessed)	evidence 1
Secrets in public JavaScript	0 secrets in 14 JavaScript files and the homepage HTML (about 1.17 MB). One gitleaks hit is a placeholder `re_your_api_key` in a docs code sample. One analytics project key (`phc_` prefix) is public by design	evidence 8
Source maps	0 `sourceMappingURL` references in those files. We did not request any `.map` file	evidence 8
Clickjacking	`X-Frame-Options: DENY` on all pages we fetched	evidence 4
Content sniffing, referrer, permissions	`X-Content-Type-Options: nosniff`, `Referrer-Policy: strict-origin-when-cross-origin`, `Permissions-Policy: camera=(), microphone=(), geolocation=()` on the homepage	evidence 2
CORS on the homepage	A request with a foreign `Origin` header got no `Access-Control-*` headers back	evidence 4
TLS	TLS 1.3 negotiated, certificate valid. We did not test older protocol versions	evidence 6
Sitemap	375 URLs listed, none of the dashboard routes in it. We fetched 5 of them (every 75th): all answered 200	evidence 1

Error monitoring

In the 14 JavaScript files we read there is no browser-side error reporter (we searched for a Sentry-style DSN pattern and found none). Server-side reporting cannot be seen from outside, so we do not claim anything about it. In a paid review we check how errors reach you and, if nothing does, we suggest installing an error tracker. We make Bugsly, so we would suggest it first, and we would say that it is ours.

What this sample did not do

- No repo access: no git history scan, no route-by-route auth check, no cross-tenant or row-level-security test, no dependency audit, no look at environment config.
- No login and no test accounts, so nothing behind the sign-in was looked at.
- No active testing: no injection, no fuzzing, no rate-limit testing, no brute force.
- Older TLS versions, the content of the DMARC reports and how the certificate is renewed were not checked.

How it was run

- Scope: aisend.app and api.aisend.app, our own. 2026-11-26, 01:56 to 01:59 UTC.
- 31 GET requests, one at a time, about 3 seconds apart, with a User-Agent that names us. 14 of them fetched the JavaScript files the homepage loads. Full list in evidence section 1.
- Tools: curl, openssl s_client (3 connections), dig (5 lookups), gitleaks 8.30.1, grep.
- Before this run we sent 3 homepage requests to other sites of ours (aisend.app, seobolt.app, cvposh.com) to choose which one to review.
- No numbers are estimated. Every count in this report is from the evidence file.

Report skeleton you would get

1. Scope and what we did not look at.
2. Summary table: severity, finding, effort.
3. Findings, one block each: what we saw, where (file and line, or URL), why it matters for your app, fix prompt for your own AI agent, what we could not check.
4. What we checked and found nothing.
5. Suggested order of work for the next 7 days.
6. Limits: findings, not a security guarantee.