INFRASTRUCTURE & SECURITY AUDIT

AUDIT REPORT

Pinecrest Ledger, Inc.

A fictional US payments start-up used to show the format of a Tier 1 audit.

SAMPLE — FICTIONAL COMPANY, ILLUSTRATIVE FINDINGS

PREPARED FOR

Engineering and leadership, Pinecrest Ledger, Inc. (fictional)

PREPARED BY

Fahad Ameen Rajput, Streetlight DevSecOps Division

ENGAGEMENT

Tier 1 — Infrastructure & Security Audit

ACCESS USED

Read-only: source, CI configuration, cloud accounts, public endpoints

Every name, host, number and finding in this document is invented to illustrate the deliverable. Hosts use the reserved `.example` domain. Nothing here describes a real company.

CONTENTS

What's in this report

How to read the findings

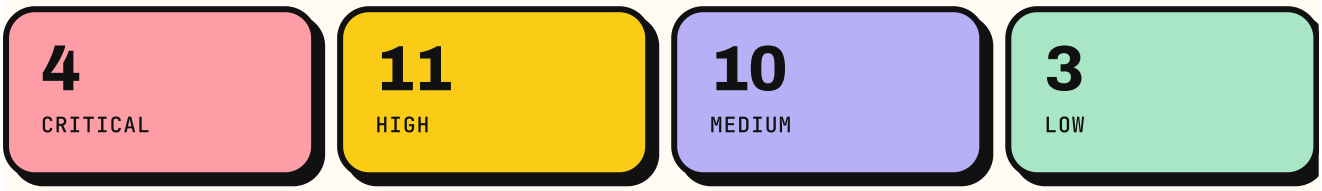
Every finding has an ID (C for CI/CD, A for cloud architecture, S for security surface, G for AI-generated code), a severity, the effort to fix it and a suggested owner. Section 7 lists all of them in one table, in the order to fix them.

SEVERITY	MEANING
CRITICAL	Exploitable now from the internet or by any contributor, with direct access to customer data or production. Fix this week.
HIGH	A realistic path to data exposure or takeover that needs one more condition. Fix within 30 days.
MEDIUM	Weakens a control or slows recovery; not exploitable on its own. Fix within 60–90 days.
LOW	Hygiene and defence in depth. Fix when the surrounding code is touched.

EFFORT	MEANING
S	Under a day for one engineer.
M	Two to five days, may need a short change window.
L	More than a week or needs a design decision first.

Overall risk rating: High

Pinecrest Ledger runs a TypeScript API and a React dashboard on AWS: containers on EKS, a PostgreSQL database on RDS, files in S3, releases through GitHub Actions. The product works and the team ships quickly. The risk comes from how quickly: the pipeline trusts code it hasn't checked, the cloud account has one very powerful key, and roughly a third of recent backend code was written with an AI assistant and merged without a security review.



The five things to fix first

1. **A pull request from any fork can read the production deploy key** (C-1). The `pull_request_target` workflow checks out and runs the contributor's code with repository secrets loaded.
2. **The CI user has AdministratorAccess through a long-lived access key** (C-2, A-1). One leaked key is the whole AWS account.
3. **The reports endpoint builds SQL from query parameters** (G-1, CWE-89). An authenticated user can read other customers' transactions.
4. **The webhook tester fetches any URL the user types** (G-4, CWE-918), including the cloud metadata service that hands out the node's credentials.
5. **An uploads bucket is publicly listable** (A-3). It holds customer-uploaded invoices.

What is already in good shape

- TLS is correctly configured on every public endpoint, with modern ciphers only.
- The database is private: no public endpoint, reachable only from the cluster's subnets.
- Infrastructure for the core network is already in Terraform, so most fixes are code changes, not console clicks.

With the 30-day items in section 8 done, the rating drops to **Medium**; with the 90-day plan done, to **Low**.

What was reviewed, and how

IN SCOPE	DETAILS
Source code	Two repositories: `ledger-api` (TypeScript, Express, 61k lines) and `ledger-web` (React). Full history of the default branch.
CI/CD	Eleven GitHub Actions workflows, repository and organisation secrets (names only, never values), branch protection, environments.
Cloud	One AWS account, one region: EKS, RDS for PostgreSQL, S3, CloudFront, IAM, CloudTrail, KMS, VPC configuration.
Public surface	`app.pinecrest.example`, `api.pinecrest.example`, `admin.pinecrest.example` and the DNS zone.

OUT OF SCOPE	WHY
Active exploitation	This is a review, not a penetration test. Nothing was attacked; issues were confirmed from configuration and code.
Mobile app	Not yet released.
Third-party payment processor	Covered by the processor's own attestations.

Method

1. **Read the pipeline.** Every workflow file, what triggers it, which secrets it can see and what it can deploy.
2. **Map the cloud account.** Configuration exported read-only and checked against the CIS AWS Foundations Benchmark, then reviewed by hand for anything a benchmark can't know, such as which role the pipeline really needs.
3. **Scan the code.** Static analysis (Semgrep with the TypeScript and Node rule packs), dependency audit (`npm audit`, Trivy for images) and secret scanning of the full history (gitleaks).
4. **Review AI-generated code by hand.** Commits the team marked as assistant-authored, plus every handler added in the last six months, read line by line. Scanners miss logic flaws; this step found four of the eight code findings.
5. **Check the public surface.** Response headers, TLS, DNS records and exposed paths.

Tool names are listed so the checks can be repeated. Findings were confirmed by hand; no tool output is quoted as a finding on its own.

The pipeline trusts too much

Releases go from a merge on `main` to production in about nine minutes with no human step. Speed is fine; the problem is what the pipeline will run and what it can reach while it does.

C-1 · CRITICAL · EFFORT S · OWNER: PLATFORM LEAD

Fork pull requests run with production secrets

`preview.yml` triggers on `pull_request_target`, which runs in the context of the base repository with its secrets, and then checks out the pull request's head. Any fork can change the build script and print `AWS_DEPLOY_KEY`.

```
# preview.yml (excerpt)
on: pull_request_target
jobs:
  preview:
    steps:
      - uses: actions/checkout@v4
        with:
          ref: ${{ github.event.pull_request.head.sha }}   # untrusted code…
      - run: npm ci && npm run build:preview             # …run with secrets loaded
```

Fix: switch to `pull_request` (no secrets for forks), and build previews for trusted branches only.

C-2 · CRITICAL · EFFORT M · OWNER: PLATFORM LEAD

A long-lived AWS key with AdministratorAccess

Deploys authenticate with an access key stored as a repository secret. The key belongs to an IAM user with `AdministratorAccess` and was created 19 months ago. Replace it with GitHub's OIDC provider and a role that can only push images and update the two deployments it touches (A-1 covers the role).

C-3 · HIGH · EFFORT S · OWNER: PLATFORM LEAD

Third-party actions pinned to tags, not commits

Nine third-party actions are referenced by version tag. A tag can be moved to malicious code after review. Pin each to a full commit SHA and let Dependabot propose updates.

C-4 · HIGH · EFFORT M · OWNER: BACKEND LEAD

No security gate before merge

Pull requests run unit tests only. There is no static analysis, dependency check or secret scan, so every finding in section 6 reached production unchallenged. Add Semgrep, `npm audit --audit-level=high` and gitleaks as required checks.

C-5 · MEDIUM · EFFORT S · OWNER: PLATFORM LEAD

Production deploys need no approval

The `production` environment has no required reviewers, and branch protection on `main` allows administrators to push directly. Require one approval for the environment and apply protection to administrators too.

C-6 · MEDIUM · EFFORT S · OWNER: BACKEND LEAD

Images run as root and aren't scanned

Both Dockerfiles use the default root user, and images are pushed without a vulnerability scan. Add a non-root user and a Trivy scan that fails the build on critical issues.

C-7 · LOW · EFFORT S · OWNER: PLATFORM LEAD

Workflow permissions default to write

The repository's default `GITHUB_TOKEN` permission is read-write. Set it to read-only and grant write per job.

Pipeline after the fixes

STAGE	TODAY	AFTER
Pull request	Unit tests	Unit tests, Semgrep, dependency audit, secret scan — all required
Build	Image built as root, pushed unscanned	Non-root image, Trivy scan, pushed only if clean
Credentials	Static key, AdministratorAccess	OIDC, short-lived role scoped to two deployments
Release	Automatic on merge	Automatic to staging, one approval for production

One account, one region, a few wide doors

The layout is sensible for a company of this size. The issues are permissions and exposure, not the shape of the system.

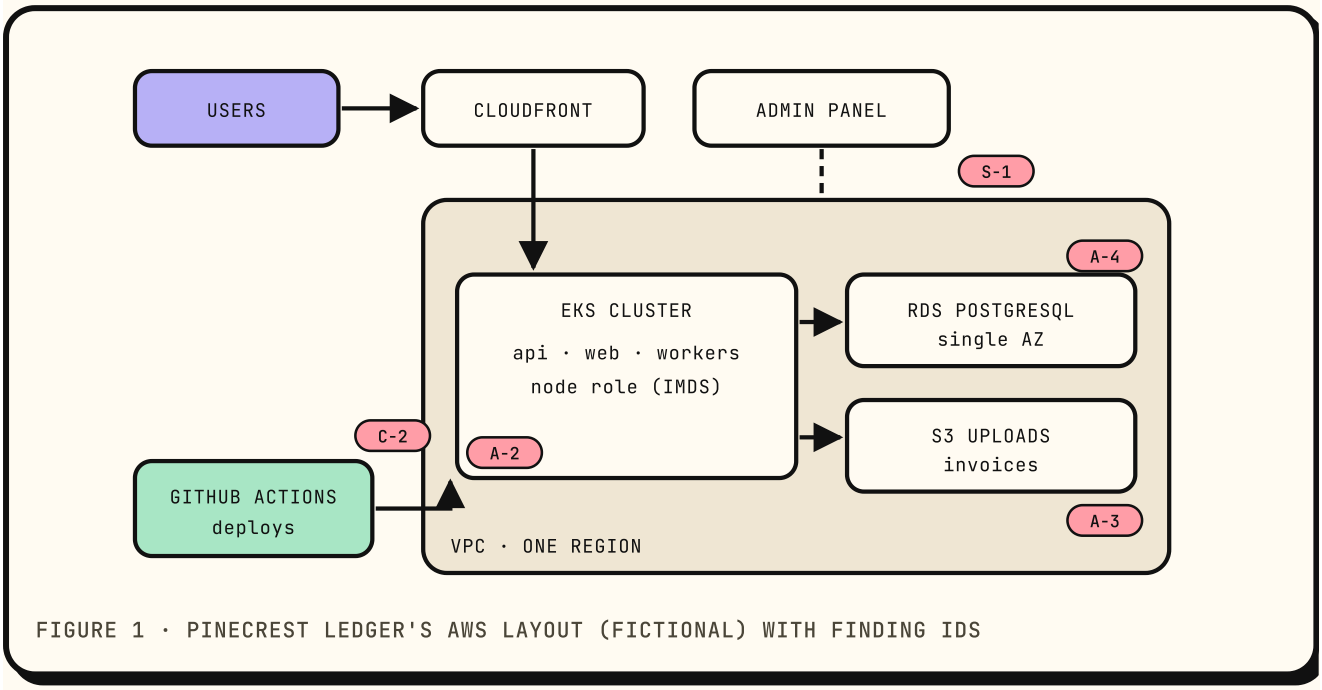


FIGURE 1 · PINECREST LEDGER'S AWS LAYOUT (FICTIONAL) WITH FINDING IDS

A-1 · CRITICAL · EFFORT M · OWNER: PLATFORM LEAD

AdministratorAccess on the deploy identity

The pipeline needs to push to one ECR repository and update two Kubernetes deployments. It can currently delete the account's KMS keys. Scope a new role to exactly those actions (with C-2).

A-2 · HIGH · EFFORT M · OWNER: PLATFORM LEAD

The cluster API endpoint is public, and pods can reach node credentials

The EKS control-plane endpoint accepts connections from the whole internet, and the instance metadata service allows hop-limit 2, so any pod can read the node role's credentials — which G-4 makes reachable from outside. Restrict the endpoint to the office VPN range, require IMDSv2 with hop limit 1, and move workloads to IAM roles for service accounts.

A-3 · HIGH · EFFORT S · OWNER: BACKEND LEAD

The uploads bucket is publicly listable

A bucket policy grants `s3:ListBucket` to everyone. Object names include customer IDs and invoice numbers. Turn on Block Public Access at account level, remove the statement and serve downloads through short-lived presigned URLs.

A-4 · HIGH · EFFORT M · OWNER: PLATFORM LEAD

The database runs in one availability zone, with unencrypted manual snapshots

Automated backups are on, but the instance is single-AZ and two manual snapshots taken before a migration are unencrypted. Enable Multi-AZ, copy the snapshots with a KMS key and delete the originals, and test a point-in-time restore.

A-5 · MEDIUM · EFFORT S · OWNER: PLATFORM LEAD

CloudTrail covers one region and nobody reads it

Make the trail multi-region with log file validation, send it to a separate bucket with object lock, and alert on root-account use and IAM policy changes.

A-6 · MEDIUM · EFFORT M · OWNER: PLATFORM LEAD

Parts of the estate are not in Terraform

The network and cluster are in Terraform; S3, CloudFront and IAM were created in the console. Import them, so the fixes above stay fixed and drift shows up in review.

A-7 · LOW · EFFORT S · OWNER: PLATFORM LEAD

No VPC flow logs

Turn on flow logs for the cluster subnets with a 30-day retention. They are the first thing an incident responder asks for.

What the internet can see

S-1 · HIGH · EFFORT S · OWNER: BACKEND LEAD

The admin panel is on the public internet with password-only login

`admin.pinecrest.example` accepts a username and password with no second factor and no rate limit. Put it behind the company's identity provider with MFA, or restrict it to the VPN.

S-2 · HIGH · EFFORT S · OWNER: PLATFORM LEAD

A dangling DNS record allows subdomain takeover

`status.pinecrest.example` is a CNAME to a hosted status-page service where the account was closed. Anyone can claim that name there and serve content from the company's domain. Delete the record.

S-3 · MEDIUM · EFFORT S · OWNER: FRONTEND LEAD

Missing security headers on the dashboard

No Content-Security-Policy, no HSTS, and `X-Frame-Options` is absent, so the dashboard can be framed. Add the headers at CloudFront; start the CSP in report-only mode for a week.

S-4 · MEDIUM · EFFORT S · OWNER: BACKEND LEAD

Verbose errors reveal stack traces

Unhandled errors on the API return the stack trace and library versions in the response body. Return a generic message with a request ID; log the detail.

S-5 · MEDIUM · EFFORT M · OWNER: BACKEND LEAD

Outdated dependencies with known high-severity issues

The audit reports 14 high-severity advisories across 6 packages, most in transitive dependencies of the PDF export. Upgrade, then keep them current with Dependabot (C-4 stops new ones).

S-6 · LOW · EFFORT S · OWNER: PLATFORM LEAD

No security.txt

Publish `/.well-known/security.txt` so a researcher who finds something knows where to send it.

Fast code, unreviewed

The team estimates that about a third of the backend written in the last six months came from an AI assistant. That code is not worse by nature, but it was merged with less scrutiny: it compiles, the tests pass, and the patterns look familiar. Eight findings came out of it; four were found only by reading the code, because the pattern was valid TypeScript that a scanner had no rule for.

ID	FINDING	CWE	SEVERITY	WHERE
G-1	SQL built by string concatenation from query parameters	CWE-89	CRITICAL	`reports/export.ts`
G-2	API key for the email provider committed in a test fixture	CWE-798	HIGH	`test/fixtures/env.ts`
G-3	File download joins a user-supplied name onto a directory	CWE-22	HIGH	`files/download.ts`
G-4	Webhook tester fetches any URL, including internal addresses	CWE-918	HIGH	`webhooks/test.ts`
G-5	Password-reset tokens hashed with MD5 and never expire	CWE-327, CWE-613	HIGH	`auth/reset.ts`
G-6	Customer notes rendered as HTML in the dashboard	CWE-79	MEDIUM	`web/NoteView.tsx`
G-7	Email validation regex with catastrophic backtracking	CWE-1333	MEDIUM	`lib/validate.ts`
G-8	Invoice lookup checks the ID exists, not who owns it	CWE-639	MEDIUM	`invoices/get.ts`

G-1 in detail — the kind of bug that passes review

```
// reports/export.ts — as merged
const sql = `SELECT * FROM transactions
  WHERE account_id = '${req.user.accountId}'
  AND created_at ⩾ '${req.query.from}'`;          // from = "2026-01-01' OR '1'='1"
const rows = await db.query(sql);

// fixed — parameters, never string building
const rows = await db.query(
  "SELECT * FROM transactions WHERE account_id = $1 AND created_at ⩾ $2",
  [req.user.accountId, req.query.from]
);
```

The account filter makes the query look scoped, so it read as safe in review. The `from` parameter escapes it and returns every customer's rows.

G-4 in detail — a helpful feature that reaches the cloud credentials

The webhook tester lets a customer paste a URL and see the response. It follows redirects and accepts any host, so a request to the instance metadata address, or to a hostname that resolves to it, returns the node's temporary AWS credentials (see A-2).

```
// webhooks/test.ts — fix
const url = new URL(input);
if (url.protocol !== "https:") throw new BadRequest("HTTPS only");
const addresses = await resolveAll(url.hostname);
if (addresses.some(isPrivateOrLinkLocal)) throw new BadRequest("That address isn't
allowed");
const res = await fetch(url, { redirect: "manual", signal: AbortSignal.timeout(5000) });
```

Fixes for the rest

- **G-2:** rotate the key today, then remove it from history and load fixtures from an ignored `.env.test`.
- **G-3:** resolve the path and reject anything outside the uploads directory; better, look files up by ID.
- **G-5:** store a SHA-256 of a 32-byte random token, expire it after 30 minutes and after first use.
- **G-6:** render notes as text; if formatting is needed, sanitise with an allowlist.
- **G-7:** replace the regex with a length check plus a simple pattern, or the validator library already in use elsewhere.
- **G-8:** add the owner to the query (`WHERE id = $1 AND account_id = $2`) and a test for the other account's ID.

Keeping it from coming back

1. Required Semgrep rules for the patterns above (C-4), including a custom rule for template-literal SQL.
2. A short checklist in the pull-request template for assistant-written handlers: input source, authorisation check, query building, outbound requests.
3. Label assistant-authored pull requests so reviewers know to read them as untrusted input.

Everything, in the order to fix it

#	ID	FINDING	SEVERITY	EFFORT	OWNER
1	C-1	Fork pull requests run with production secrets	CRITICAL	S	Platform lead
2	G-1	SQL injection in reports export	CRITICAL	S	Backend lead
3	C-2 + A-1	Long-lived admin key for deploys	CRITICAL	M	Platform lead
4	G-2	Email provider key committed in a fixture	HIGH	S	Backend lead
5	A-3	Uploads bucket publicly listable	HIGH	S	Backend lead
6	G-4 + A-2	SSRF to node credentials; public cluster endpoint	HIGH	M	Backend + platform
7	S-2	Dangling CNAME allows subdomain takeover	HIGH	S	Platform lead
8	S-1	Admin panel public, password-only	HIGH	S	Backend lead
9	G-3	Path traversal in file download	HIGH	S	Backend lead
10	G-5	Weak, non-expiring reset tokens	HIGH	S	Backend lead
11	C-3	Actions pinned to tags	HIGH	S	Platform lead
12	C-4	No security gate before merge	HIGH	M	Backend lead
13	A-4	Single-AZ database, unencrypted snapshots	HIGH	M	Platform lead

#	ID	FINDING	SEVERITY	EFFORT	OWNER
14	C-5	No approval for production deploys	MEDIUM	S	Platform lead
15	G-8	Invoice lookup without ownership check	MEDIUM	S	Backend lead
16	G-6	Stored XSS in customer notes	MEDIUM	S	Frontend lead
17	S-3	Missing security headers	MEDIUM	S	Frontend lead
18	S-4	Stack traces in API errors	MEDIUM	S	Backend lead
19	G-7	ReDoS in email validation	MEDIUM	S	Backend lead
20	C-6	Root containers, no image scan	MEDIUM	S	Backend lead
21	S-5	High-severity dependency advisories	MEDIUM	M	Backend lead
22	A-5	Single-region, unmonitored CloudTrail	MEDIUM	S	Platform lead
23	A-6	Console-created resources outside Terraform	MEDIUM	M	Platform lead
24	C-7	Workflow token defaults to write	LOW	S	Platform lead
25	A-7	No VPC flow logs	LOW	S	Platform lead
26	S-6	No security.txt	LOW	S	Platform lead

28 findings in 26 rows: C-2 and A-1 share one fix, as do G-4 and A-2. The counts in section 1 are per finding — 4 critical, 11 high, 10 medium, 3 low.

Ordering

Critical and quick first; then anything that turns another finding into a breach (A-2 turns G-4 from a bug into credential theft); then the gates that stop the same classes returning (C-3, C-4). Effort breaks ties.

30, 60 and 90 days

FIRST 30 DAYS · STOP THE BLEEDING

- Fix C-1 and G-1 in the first week; rotate the G-2 key the same day.
- Move deploys to OIDC with a scoped role (C-2, A-1); delete the old IAM user.
- Close the bucket (A-3), the SSRF (G-4) and the metadata path (A-2); delete the dangling record (S-2).
- Put the admin panel behind SSO with MFA (S-1).

DAYS 31–60 · PUT THE GATES IN

- Required Semgrep, dependency audit and secret scan on every pull request (C-4); pin actions (C-3).
- Fix G-3, G-5, G-6, G-7 and G-8 with a regression test each.
- Approval on production, protection for administrators (C-5); read-only default token (C-7).
- Security headers and generic errors (S-3, S-4); dependency upgrades (S-5).

DAYS 61–90 · MAKE IT RESILIENT

- Multi-AZ database, encrypted snapshots, a timed restore drill (A-4).
- Multi-region CloudTrail with alerts; VPC flow logs (A-5, A-7).
- Import console resources into Terraform (A-6); non-root, scanned images (C-6); security.txt (S-6).
- Re-run this audit's checks to confirm every finding is closed.

30 minutes, then you own the plan

MINUTES	TOPIC
0–5	The overall rating and the five things to fix first — questions on anything unclear.
5–15	Walk through the three critical findings: how each could be used, and the exact fix.
15–22	The roadmap: who owns which item, and what fits into the next two sprints.
22–27	AI-assisted code: the review checklist and the pipeline rules that back it up.
27–30	Next steps: fix in-house with the roadmap, or scope a Tier 2 project from these findings.

Before the call

- Rotate the G-2 key if it hasn't been done yet.
- Bring whoever owns the pipeline and the AWS account.
- Note any finding you think is wrong — context the review couldn't see changes severities.

ABOUT THIS SAMPLE

This report shows the structure and depth of a Tier 1 audit. Pinecrest Ledger, Inc., its systems and every finding are invented. A real report covers your systems only and is shared with you alone.