FOR PARTNERS, STORE OWNERS AND TECHNICAL REVIEWERS

How CardArc is built to be secure

CardArc holds the things a card business cannot afford to lose: the money ledger, the inventory, and customers' personal details. This document explains the controls that protect them, how each one is enforced, and how we prove every day that it still is.

3
independent layers every admin request must cross: edge, identity, application

102/102
tenant-isolation assertions passing on the current release, re-run before every deploy

60s
between live probes of the running system: services, TLS, firewall, secrets, git sync

Prepared by
Milo, CTO
Coco Milo Holdings

Independent review
Cece, CISO
weekly written report, last cycle 24 Aug 2026

Release covered
CardArc build 7eb8278
26 Aug 2026

IN PLAIN WORDS

Security here is a process, not a checkbox

Every control on these pages exists on the running system. Each one is either **probed automatically every 60 seconds** or **attested with a date and a name.** A control that drifts turns red on our own dashboard by itself, and an independent reviewer checks the whole picture every week.

Defense in depth

No single control is trusted alone. A request to anything sensitive crosses at least three independent layers: the edge (TLS and routing), identity (who are you), and the application (are you allowed to do this to this row).

Least privilege

Every app listens only on the machine's internal interface, admin identities are allowlisted, access is scoped by role and by workspace, and secrets are files readable by one system user and nobody else.

Verified, not assumed

We check the effective state, not the config file. The dashboard reads what the server is actually doing right now, so a claim on this page cannot outlive the control behind it.

Independent oversight

Cece, our AI security officer, audits every product weekly against a written charter, files a dated report and pushes findings into a tracked queue. The CTO fixes; the owner signs off on risk.

What CardArc protects, and from what

MONEY	Every buy, sale, deposit, payout and correction is a line in an append-only ledger. History is never rewritten; a mistake is fixed by adding a reversal, and the audit trail shows who did what and when.
INVENTORY	Each store's stock is walled to its own workspace. No store can read, price, sell or even see another store's cards, even by guessing an ID.
PEOPLE	Customer details are collected only where the storefront needs them, are never exposed to other tenants, and can be deleted by the customer themselves (UU PDP and GDPR self-service).
ACCESS	Staff see what their role allows. A cashier cannot change prices; a manager cannot touch the owner's account; a suspended account is locked out instantly.

The stack is deliberately simple: a Next.js application, a SQLite database per instance, a Caddy edge that issues and renews TLS automatically, and systemd on one hardened Linux server. Simple systems have fewer places to hide a mistake, and every piece is something we can inspect directly.

HOW A REQUEST REACHES OUR SYSTEMS

Three walls between the internet and your data

ORIGIN

Visitor

- Browser or API client on the public internet

HTTPS only, plain HTTP is redirected

WALL 1 · EDGE

Caddy reverse proxy

- TLS certificates issued and renewed automatically for every domain, including each store's custom domain
- The single entry point for everything we run; nothing else is reachable from outside
- Host firewall allows only web traffic and an authenticated admin path
- Browser security headers on every domain: forced HTTPS (HSTS), no framing of admin pages, no content sniffing

routed by domain and path

Public storefront routes

Straight to the app

- Store pages, catalog, item pages, auctions
- Public write endpoints (enquiries, bids, pre-order sign-ups) are rate-limited and bound-checked
- The store is resolved from the domain, never from anything the client sends

Admin and internal dashboards

WALL 2 · IDENTITY

Google single sign-on

- Verified Google identity required, with an explicit email allowlist, not "any Google account"

signed identity, re-checked below

WALL 3 · APPLICATION

The app checks again

- The app never trusts the proxy blindly: session cookies are httpOnly and re-verified on every request
- No identity means a 401 or a redirect to sign-in
- Cross-origin requests are closed by default (no CORS headers); cookies are SameSite so other sites cannot ride a login

every action re-checks role and ownership

INSIDE

Application layer

- Role-based access scoped per workspace and branch; every server action guards itself and never relies on the UI having hidden a button
- All input validated and bounded server-side; all database access through parameterised statements, so input is data and never SQL
- Financial writes are atomic transactions; the ledger is append-only

internal interface only

BOTTOM

Data layer

- Databases are reachable only from the server itself, never from the network
- Secrets live in files readable by one system user; they are never in the code repository, and CI scans every push for leaks

What is structurally impossible

Reaching an app or a database directly on its port from the internet. Every service is bound to the machine's internal interface. The only road in is through the edge above, and our dashboard probes that binding live, every minute.

OUR CONTROL FRAMEWORK

Nine squares, every product scored on each

Three columns: the **gate** (who gets in), the **code** (what the app does with what it is given) and the **ground** (the machine it all runs on). Each product's checklist maps to these nine squares on our dashboard.

GATE

Authentication

Google SSO with allowlist for admin surfaces; session-based login with httpOnly cookies inside the app; role gates on every admin page and action.

PROOF 30-second auth test after every auth change: log in, open a protected page, copy the URL, log out, paste. Must be denied.

CODE

Validation

Every field on every public and admin endpoint is typed, ranged and allowlisted on the server (grades, conditions, currencies, quantities).

PROOF Attested per release; parameterised SQL throughout.

GROUND

Infrastructure

Auto-renewed TLS on every domain; host firewall active; every app bound to the internal interface; SSH key-only with root login off.

PROOF Probed live every 60s: TLS expiry, binding, firewall state.

GATE

Isolation

Every query is scoped to the caller's workspace and branch. Every client-supplied ID is ownership-checked before it is used.

PROOF 102-assertion isolation suite run before every deploy (see page 5).

CODE

Secrets

Keys live in files with owner-only permissions, outside the repository. Never committed, verified across full git history.

PROOF gitleaks on every push; file permissions probed live.

GROUND

Monitoring and audit

Service health probed every minute; daily automated report; who-did-what audit logs on prices, edits, write-offs and impersonation.

PROOF Dashboard checks are live; audit tables are append-only.

GATE

Rate limits

Storefront enquiries, bids, sign-ups and scheduled endpoints are rate-limited so nobody can hammer a store or enumerate data.

PROOF Shipped in the 2026-06 hardening pass; re-verified in code review.

CODE

Patching

Dependency audit gate in CI; automated OS security updates on the host; nightly vulnerability sweep across all apps with an alert file if anything is found.

PROOF npm audit: 0 known vulnerabilities at last cycle.

GROUND

Recovery

Nightly database snapshots, a daily off-site copy, whole-server daily snapshots at the host, and a pre-change backup before every risky migration.

PROOF Backup runs verified daily; pre-migration snapshots on every risky change.

THE CONTROL THAT MATTERS MOST IN A MULTI-STORE PLATFORM

Only your rows, enforced in the application

CardArc runs many stores on one platform. Each store is a **workspace**; each workspace has one or more branches. The rule is simple and absolute: a request can only ever touch rows that belong to the workspace it is signed into. We enforce that rule in the application layer and test it as if we were an attacker.

Scoping on every read and write

Every inventory, sales, customer, consignment and pre-order query carries the caller's workspace and branch. There is no "list all" path that forgets the filter; helpers that resolve a workspace fail closed when they cannot.

Ownership checks on every ID

Any ID a client sends (a product, a batch, a campaign, a display, a staff member) is checked for ownership before use. Guessing another store's ID returns "not found", not their data.

Storefront resolved from the domain

Public actions (enquiry, checkout, bids, pre-order registration, consignment search) derive the store from the hostname. Nothing in the request body can point them at another store.

The owner wall

Roles are walled upward: a manager can manage staff but can never alter, demote or remove an owner, and cannot mint other managers. Platform-level flags are separate from workspace roles, so one can never be mistaken for the other.

Impersonation is read-only

Support can view a workspace as its owner to help them, for a 30-minute window that is logged. Every non-read request in that mode is blocked at the proxy, before it reaches any code.

Shared catalog, private prices

Card reference data is shared so every store benefits from one catalog. Everything a store adds on top (costs, prices, notes, photos, manual products) is owned by that store and invisible to others.

102 / 102

assertions in the tenant-isolation suite on build 7eb8278

The suite creates a sandboxed second workspace and tries, from inside it, to read or change the first workspace's inventory, staff, products, batches, displays, pre-orders, write-offs, relinks and breakdowns. Each attempt must be refused. It is part of the deploy ritual: a release does not ship if the count drops. Alongside it run a 71-assertion isolation check on the private instance, a 50-scenario pre-order simulation and a 192-scenario ledger replay.

Lessons applied. In August 2026 an internal test found a manager able to edit an owner's account because a guard checked a platform flag instead of the workspace role. We ran a full authorisation sweep of every action, route and page, fixed the class of bug in four commits, wrote the rules above into the engineering guide, and added the cases to the suite. That is the loop working as designed.

MONEY RECORDS

Nothing is ever rewritten

Append-only ledger

Buys, sales, deposits, payouts and cost corrections are lines. A cancellation adds a reversal that references the original; both stay visible. Statistics exclude reversed lines automatically.

Atomic writes

A sale that touches stock, ledger, consignor split and receipt happens in one database transaction. It either fully happens or not at all, so two cashiers acting at once cannot corrupt a count.

Traceable edits

Price changes, product detail edits, write-offs and impersonation sessions each have their own audit log with the acting account and timestamp, and an undo path where it is safe.

DELIVERY AND ASSURANCE

How code ships, and who checks it

1 · CODE

Committed and pushed

Every change is committed and pushed to GitHub. Uncommitted drift on the server shows red on the dashboard.

2 · CI GATES

Automated checks on every push

Secret scanning (gitleaks), dependency audit, and update alerts. A leaked key or a known vulnerability fails the build.

3 · RITUAL

Pre-deploy tests

Type check, build, isolation suite, pre-order and ledger simulations, staging smoke test on a separate instance.

4 · DEPLOY

Managed services, minutes to roll back

systemd-managed services with atomic restarts. Rollback is one revert and one restart. A pre-change backup is taken before risky migrations.

5 · PROBES

Live every 60 seconds

Service health, TLS expiry, git sync, secret file permissions, loopback binding, firewall. Drift shows red without anyone looking.

6 · WEEKLY AUDIT

Independent review

Cece reviews every product against a written charter and files a dated report. Findings enter a tracked queue and are closed by the CTO.

Governance

CECE · CISO	Weekly review of every product (Monday 09:00 SGT), secrets, dependencies, auth, isolation, legal pages, backups, observability and a diff review of the week's commits. Reports to the CTO.
MILO · CTO	Owns architecture, hardening and remediation. Triages every finding by severity: critical now, high this week, medium to backlog.
OWNER	Signs off on risk acceptance, credentials, legal entities and anything customer-facing. Receives only what needs a decision.
LAUNCH GATE	No product goes live without isolation verified and IDOR-tested, auth passing the 30-second test, HTTPS everywhere, and for anything collecting personal data: privacy policy, terms and self-service deletion.

IF THE WORST HAPPENS

Recoverable by design

Nightly
Snapshot
Databases, configuration, secrets and service definitions, with dated restore points retained.

Daily
Off-site copy
Database snapshots leave the server every day. The hosting provider also keeps a daily whole-server snapshot that covers images.

Weekly
Integrity check
An automated reconciler cross-checks inventory against the financial ledger and flags any drift.

On demand
Point-in-time restore
Restore from any retained point; a pre-change backup is taken before every risky migration.

POSTURE ON 26 AUG 2026

What is verified, and what we are still improving

We publish gaps as well as passes. A security page that only lists green ticks is not one you should trust.

CONTROL	STATE	STATUS
Apps bound to internal interface	All services, probed live	LIVE PROBE
HTTPS on every domain	Auto-issued, auto-renewed, expiry probed	LIVE PROBE
Browser security headers and CORS	HSTS, anti-framing, nosniff on all domains; CORS closed except two landing-page endpoints pinned to cardarc.app	LIVE PROBE
Host firewall and SSH hardening	Firewall active; key-only SSH, root login off, verified at runtime	LIVE PROBE
Secrets outside the repo, owner-only	Never in git history; permissions probed; gitleaks in CI	LIVE PROBE
Tenant isolation	102/102 suite; full authorisation sweep closed 23 Aug 2026	VERIFIED
Parameterised SQL, server-side validation	Attested per release by the CTO	ATTESTED
Known vulnerabilities	npm audit 0 across all apps; OS auto-updates active	VERIFIED
Privacy, terms, delete-my-data	Published and probed; self-service deletion live	LIVE PROBE
Backups and off-site copy	Nightly plus daily off-site and host snapshots	VERIFIED
Auth 30-second test	Automated half passes after August auth changes; manual browser half scheduled	IN PROGRESS
Key rotation runbook	Zero-downtime rotation procedure to be written	PLANNED
Application error tracking	Logs, probes and daily reports today; crash-level alerting to add	PLANNED
Single-server resilience	Accepted at current scale; recovery path tested; revisit with growth	ACCEPTED RISK

Want the detail? Ask for the latest weekly review, the deploy ritual, or a walkthrough of the isolation suite. We will show you the live dashboard rather than a slide.