TECHNICAL OVERVIEW

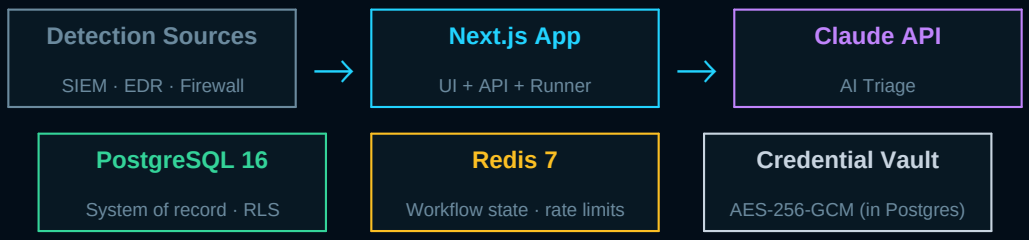
Architecture, Stack & Security Design

Self-Hosted SOAR · Multi-Tenant · AI-Embedded

Version 1.1 · For Architects & Security Engineers

1 · System Architecture

Synapse IronNode is a self-hosted, containerized SOAR platform built on a single Next.js application (React UI + API routes), a PostgreSQL system of record, and a Redis layer for workflow state and rate limiting. The embedded AI analyst calls the Anthropic Claude API. Everything is orchestrated with Docker Compose on an isolated bridge network.



Request & Execution Flow

- An inbound webhook (`/api/webhook/<slug>/<path>`) is rate-limited per tenant, resolves the playbook, and returns **202** immediately.
- The runner executes nodes asynchronously: **Enrich** (integration HTTP calls) → **AI Triage** (Claude).
- At the AI node the runner serializes state to Redis + Postgres and **suspends**, surfacing the incident for human triage.
- On analyst approval, the runner resumes from the suspended state and dispatches only approved actions.

2 · Technology Stack

Layer	Technology	Role
Frontend	Next.js 14 (React 18), Tailwind CSS, React Flow	Dashboard, visual playbook builder, triage UI
Backend	Next.js API routes (Node 20)	Auth, runner, webhooks, REST endpoints
Database	PostgreSQL 16 (JSONB + RLS)	Tenanted system of record, immutable ledger
Cache/Queue	Redis 7 (ioredis)	Suspended workflow state, per-tenant rate limits
AI	Anthropic Claude (claude-sonnet-4-6)	Deterministic JSON-schema triage analysis
ORM	Prisma 5	Typed data access + tenant client extension
Auth	jose (JWT), bcrypt, otplib + qrcode	15-min JWT, refresh cookies, TOTP MFA
Infra	Docker + Docker Compose	Containerized, isolated bridge network

3 · Core Subsystems

Subsystem	Design
Zero-Trust Identity	Context-aware login (role, origin, MFA). Short-lived 15-min access JWTs; HttpOnly refresh tokens (7d, revocable). Step-up & enforceable TOTP MFA.
Visual SOAR Engine	Node-based React Flow builder. Playbooks stored as JSON (nodes + edges). Trigger → Enrich → AI Triage → Human Approval → Execute.
Schema-Driven Runner	Execution decoupled from UI. Integrations defined by JSON schema (baseUrl, endpoints, body templates). {{field}} interpolation from the live payload.
Credential Vault	Per-tenant AES-256-GCM (unique IV + auth tag per record). Decrypted & injected into outbound requests at runtime only; never reaches the browser.
AI Co-Pilot	Isolated system prompt + strict JSON-schema enforcement → deterministic { riskScore, analyst_guidance, proposed_actions }. Per-tenant key & on/off toggle.
HITL Triage	Runner suspends at the AI node (state in Redis + Postgres, callback token) to avoid HTTP timeouts. Progressive-disclosure UI; granular, role-gated approvals.
Immutable Ledger	Append-only JSON per incident: TRIGGER → ENRICHMENT → AI_ANALYSIS → HUMAN_APPROVAL → EXECUTION. Exportable as JSON or branded PDF.

4 · Multi-Tenancy & Isolation

Isolation is enforced in depth. Every tenant row carries a `tenantId`; a Prisma client extension scopes all queries at the application layer AND sets a per-transaction Postgres session variable that drives **Row-Level Security** policies on every table.

```
set_config('app.current_tenant', <uuid>, true) + FORCE ROW LEVEL SECURITY
```

- **App layer:** list/aggregate reads & writes auto-filtered by tenantId.
- **DB layer:** RLS USING/WITH CHECK confines every row — even the table owner is subject (FORCE).
- **System context:** a deliberately un-scoped client handles bootstrap, login lookup, and webhook tenant resolution only.
- **MSSP-ready:** a Platform Admin provisions organizations, each with its own admin, slug & webhook rate limit.

5 · RBAC Capability Model

Capabilities are enforced server-side per request; roles inherit cumulatively.

Role	Adds capability
1st Line Analyst	`DASHBOARD_VIEW, INCIDENT_VIEW, TRIAGE_APPROVE_LOW`
2nd Line Analyst	`TRIAGE_APPROVE_HIGH, PLAYBOOK_MANAGE`
3rd Line Analyst	`INTEGRATION_MANAGE (APIs, vault, AI key)`
Administrator	`USER_MANAGE, MFA_ENFORCE, AUDIT_VIEW`
Platform Admin	`ORG_MANAGE (cross-tenant provisioning)`

High-risk approvals are gated at execution time: approving any action the AI flagged HIGH risk requires TRIAGE_APPROVE_HIGH (2nd Line+).

6 · Data Model

Table	Purpose
`organizations`	Tenant root: slug, MFA policy, AI toggle, webhook rate limit.
`users`	Accounts: role, MFA secret/flags. Email globally unique.
`refresh_tokens`	Revocable 7-day refresh tokens.
`playbooks`	Node/edge JSON + unique webhook path.
`integrations`	Schema-driven outbound API definitions + vault key reference.
`credential_vault`	AES-256-GCM encrypted secrets (data, IV, auth tag).
`incidents`	Status, suspended runner state, immutable ledger, severity.
`audit_logs`	Append-only security events with actor & detail.

7 · Security & Deployment

Security Controls

- Defense-in-depth tenant isolation (app scoping + Postgres RLS).
- Secrets encrypted at rest (AES-256-GCM); JWT secrets & vault key supplied via environment.
- Enforceable MFA (org-wide or per-user) with TOTP enrolment; step-up for privileged actions.
- Per-tenant webhook rate limiting (Redis) prevents cross-tenant interference.
- Append-only audit logging of logins, triage, user/org and AI-config changes.

Deployment

- `docker compose up --build` brings up app, PostgreSQL and Redis on a private network; only the app port is exposed.
- The container entrypoint runs `prisma migrate deploy` on start (RLS policies applied), then serves the standalone Next.js build.
- First launch creates the initial organization + Platform Admin in one atomic transaction.
- Secrets (JWT, refresh, vault key, optional Claude key) are provided via `.env`.

See USER_GUIDE.pdf for operator workflows and SETUP.md for environment configuration.

— End of Technical Overview —