**Security Orchestration & Risk Platform  
Low-Level Design & Implementation Guide**

Detailed Technical Design • API Contracts • DB Schema • Cursor Prompts

*August 22, 2025*

# 1. Introduction

This Low-Level Design (LLD) builds upon the HLD and DDP, defining class structures, API contracts, database schema, and implementation guidance.

# 2. Backend Design (Spring Boot)

Package Structure: com.secplatform.{core, scanner, api, db, security, parsers}

API Contracts:

* **POST /scans/start:** Start a scan (input: repo URL, scan type, tool). Returns jobId.
* **GET /scans/{id}:** Retrieve scan status and findings.
* **GET /findings:** Filter by assetId, severity, status.
* **POST /policies/evaluate:** Run policy checks on given asset.

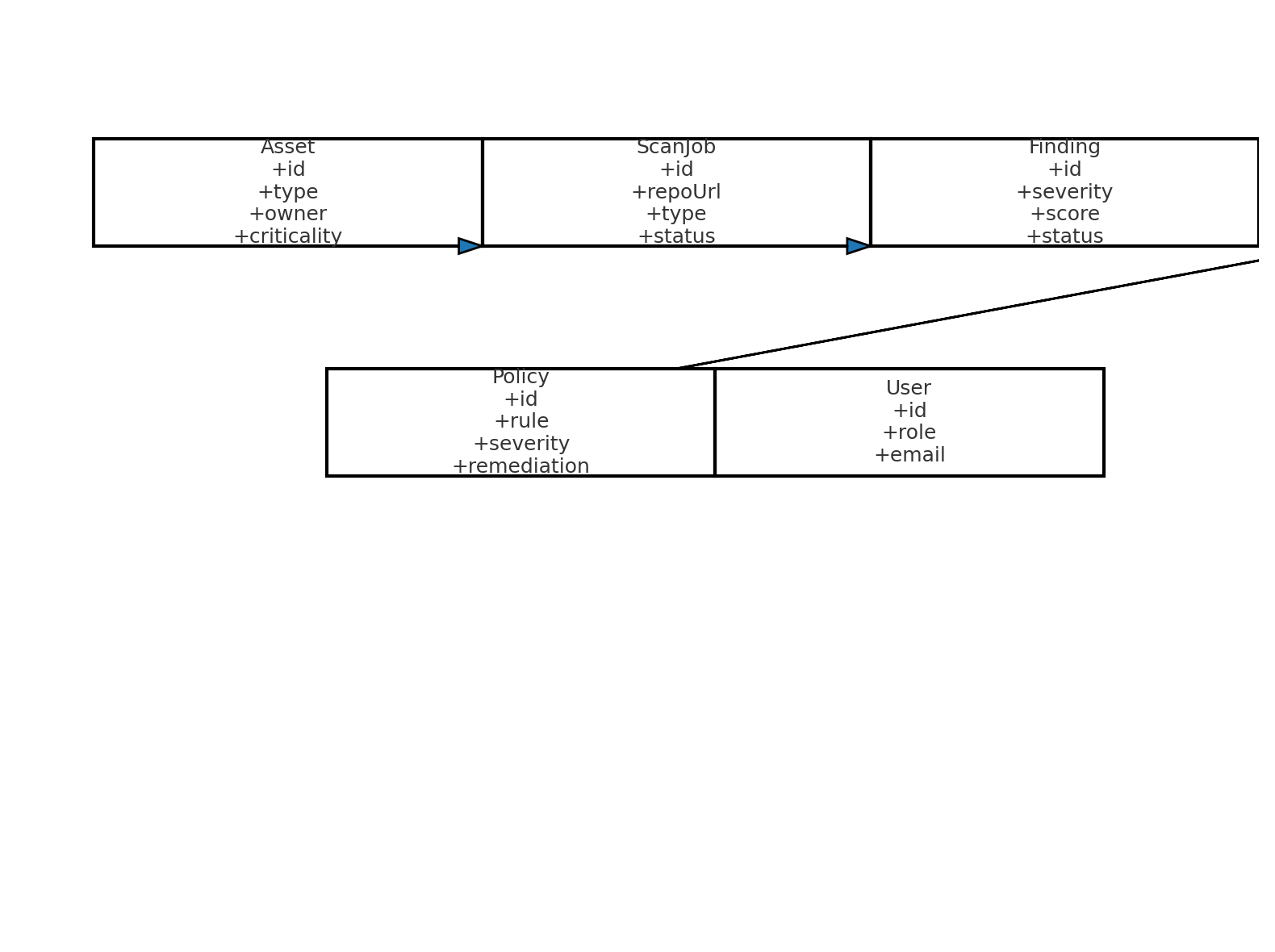


Figure 1: Core entity class diagram.

# 3. Database Schema (PostgreSQL + Liquibase)

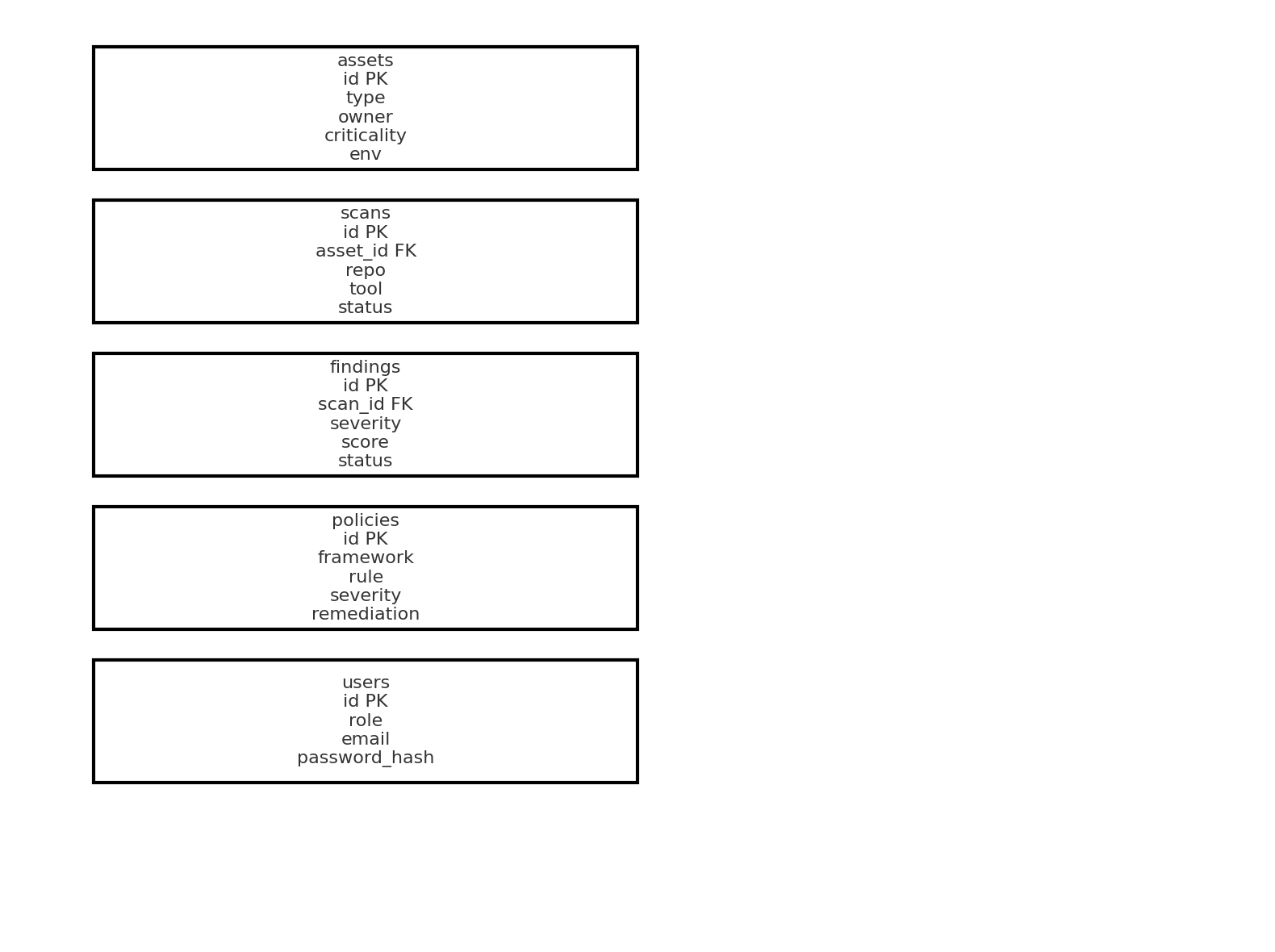


Figure 2: Relational schema for core entities.

# 4. Scanner SDK Design

Each scanner implements a YAML manifest + parser:

scanner.yaml  
id: sast-semgrep  
input: repo  
command: semgrep --config auto --json  
output\_format: json  
parser: com.secplatform.parsers.SemgrepParser

# 5. Data Flow & Orchestration

Kafka Topics:  
- scan.requests  
- scan.results  
- scan.errors

Jobs keyed by commit hash + tool version for idempotency.

# 6. Risk Scoring Logic

Risk = Severity × (Exploitability + Exposure) × Criticality ÷ Controls

# 7. Cursor Implementation Prompts

Example prompt for backend repo scaffold:

You are acting as a senior Java Spring Boot developer.  
Scaffold a project 'secplatform-core' with:  
- REST controllers for /scans, /findings  
- Entities: Asset, ScanJob, Finding, Policy, User  
- JPA repositories with Liquibase migrations  
- JWT-based Spring Security  
- Kafka integration for scan.requests/results  
- Dockerfile + Helm chart for deployment

# 8. Frontend Design (React)

Components: DashboardLayout, RiskOverview, FindingsTable, AssetGraph, ScanRequestForm.

State Management: React Query for API calls.

Charts: Chart.js for reports; D3.js for attack-path graphs.

# 9. CI/CD Pipeline

Stages:  
1. Pre-merge → Semgrep, IaC checks  
2. Build → Syft SBOM + Trivy  
3. Release → Cosign signing  
4. Deploy → Gatekeeper attestation check