Proactive Java Dependency Vulnerability Finder – v1 Design

Goal: Continuously detect high and critical (CVSS > 8) vulnerabilities affecting transitive Gradle Java dependencies within 24 hours of CVE publication, faster than existing tools, using direct CVE feed polling.

Execution Model

Long-running Java service that centrally polls CVE feeds, scrapes GitHub advisory pages, scans repositories, and raises alerts exactly once per CVE + repository combination.

Core Architecture

1. CVE Polling Scheduler (RSS, interval-based)
2. CVE Scraper & Normalizer (GitHub advisory pages only)
3. CVE Store (persistent DB)
4. Gradle Dependency Scanner (gradle dependencies output)
5. Matcher & Alert Engine
6. HTML Report Generator and Notifier

CVE Intake Rules

1. Poll RSS feed at configured interval
2. Process only CVSS > 8 vulnerabilities
3. Structured version ranges only
4. Maven Central Java artifacts only
5. Ignore qualifiers like -SNAPSHOT or -RC

Dependency Resolution

1. Use gradle dependencies output as source of truth
2. Resolve all transitive dependencies
3. Match against CVE vulnerable ranges
4. No context or usage analysis in v1

Alert Semantics

1. Alert created only once per CVE + repo + artifact
2. Alert fires on first successful match

3. No repeated alerts on subsequent scans
4. HTML report generated per alert

Fix Recommendation Strategy

1. Extract available versions from Maven metadata page
2. Select latest stable version outside vulnerable range
3. Prefer speed and clarity over minimal upgrade

Non-Goals (v1)

1. No static or runtime usage analysis
2. No IDE integration
3. No external vulnerability APIs
4. No non-Java ecosystems
5. No RSS listener push model

v1 Success Criteria

1. Detect and alert vulnerable dependencies within 24 hours of CVE publication
2. Lower alert latency compared to Dependabot
3. High-confidence, low-noise alerts suitable for platform teams