Product Requirement Document

1. Introduction

Purpose: Provide users with a clear, scalable interface to scan container images, visualize vulnerabilities by severity, and prioritize remediation.

Scope: From ingestion of image metadata to deep-dive views of individual findings, with filters, sorting, and integrations into CI/CD pipelines and registries.

2. Assumed Personas

1. DevOps Engineer

    - Needs to quickly spot failing or risky images before deployment.

2. Security Analyst

    - Reviews vulnerability trends, enforces patch policies, and drives SLAs.

3. Engineering Manager

    - Tracks remediation progress and team compliance against security targets.

3. Problem Statement

Thousands of container images live in registries. Each image can contain multiple CVEs of varying severity. Users must know which images to fix first, based on severity and business impact.

4. Goals & Objectives

Visibility: Surface overall vulnerability posture at a glance.

Prioritization: Highlight images with critical/high-severity issues.

Actionability: Enable drill-down into details and link fixes back to source repos

ticketing tools.

Automation: Integrate with registries (e.g. Docker Hub, ECR, ACR) and CI/CD (e.g. Jenkins, GitHub Actions).

5. Functional Requirements

- Dashboard View: Summary cards (total images, critical/high/medium/low counts), top-10 at-risk images.
- Filters & Sorting: By image name, registry, severity, last scan date, owner/team.
- Image Detail Page: List of findings (CVE ID, package, installed vs. fixed version, CVSS score, advisory link).
- Search: Global search bar for image names or CVE IDs.
- Alerts & Notifications: Email or Slack alerts when new critical vulnerabilities appear.
- Integrations: Connectors for Docker Hub, Amazon ECR, Azure ACR; CI/CD webhook triggers.
- Role-Based Access: Viewer, Editor, Admin roles with scoped permissions.

6. Non-Functional Requirements

- Scalability: Support up to 100,000 images with incremental scans.
- Performance: Dashboard initial load <2 seconds; filter operations <1 second.
- Reliability: 99.9% uptime.
- Security: Encrypted data at rest and in transit; audit logs for user actions.

7. User Stories

1. DevOps Engineer

   - As a DevOps Engineer, I want to filter images by critical severity so I can immediately address top risks.

2. Security Analyst

   - As a Security Analyst, I want automated email alerts for new critical vulnerabilities so I can enforce patch policies.

3. Engineering Manager

   - As an Engineering Manager, I want a chart of weekly remediation progress so I can report to stakeholders.

8. Success Metrics

- Mean Time to Detect (MTTD) new critical vulnerabilities < 1 hour.
- Mean Time to Remediate (MTTR) critical vulnerabilities < 24 hours.
- Dashboard adoption rate: 90% of DevOps teams using within first month.
- Scan coverage: ≥95% of images scanned daily.

9. Development Action Items

1. Epic: Scanning Engine

   - Integrate Trivy/Clair; schedule daily full & incremental scans.

2. Epic: Data Storage

   - Design schema for images, findings, scan history.

3. Epic: UI Components

   - Build dashboard cards, tables, filter components in React.

4. Epic: Alerts & Integrations

- Implement webhooks, email/Slack notifications; registry connectors.

End of Document