Product Requirement Document - Container Vulnerability Scanner

Product Requirements

Product Title: Container Image Vulnerability Scanner

Objective:

Build a user-friendly interface and backend system to scan container images, identify vulnerabilities, and prioritize critical/high severity issues for remediation, especially in environments with thousands of images.

Target Users:

- DevOps Engineers
- Security Analysts
- SREs (Site Reliability Engineers)
- Platform Engineers

Core Features:

1. Dashboard:
   - Total number of scanned images
   - Summary chart: # of Critical, High, Medium, Low vulnerabilities
   - Top 10 images with most critical vulnerabilities

2. Image List View:
   - Paginated list of images with columns: Image Name, Tags, Vulnerability Count, Risk Score, Last Scan Time
   - Filters by severity, fixable vulnerabilities, image name, and date

Product Requirement Document - Container Vulnerability Scanner

3. Image Detail View:

- Metadata: Name, Tag, Digest
- Breakdown of vulnerabilities by severity, component, layer, fix availability
- CVE links and suggested fix paths

4. Notifications and Alerts:

- Real-time alerts on new critical vulnerabilities
- Integration with Slack, Email, Jira

5. Export and Reporting:

- Export findings in CSV/JSON/PDF formats
- Schedule reports

User Journeys:

- A DevOps engineer logs in, checks dashboard, filters critical images, exports list.
- A security analyst is notified about a risky image and checks layer-wise vulnerabilities and available fixes.

Technical Requirements:

- REST API integrations with tools like Trivy/Clair
- Responsive UI for large-scale image datasets
- Secure authentication (OAuth2, JWT)
- CI/CD scan integration

Non-Functional Requirements:

- High performance, fault tolerance, and RBAC

Low-Fidelity Wireframes

Product Requirements Document (PRD)

Product Requirements Document (PRD)

Product Title: Container Image Vulnerability Scanner

Objective:

Build a user-friendly interface and backend system to scan container images, identify vulnerabilities, and prioritize critical/high severity issues for remediation, especially in environments with thousands of images.

Target Users:

- DevOps Engineers
- Security Analysts
- SREs (Site Reliabilbility Engineers)
- Platform Engineers

Core Features:

#70 Dashboard

- Total number of scanned images
- Summary chart, No Critical, High/High / Medium/ Low vulneunlnerabilities
- Top 10 images with most critical vulnerabilitilities
- Scan health status (Last scan time, number of eumber of errors)

#70 Image List View

- Paginated view of all auscanned images
- Columns:
  - Image Name
  - Tags Vulnerability Count (Critical/High/Medium/Low)
  - Risk Score (Color-coded badge)
  - Last Scan Time
- Filters: Severity (Critical, High, etc.)
- Date Range (last scan)
- Fixableb vulnerabilities
- Image name or tag search

#70 Image Detail View

Product Requirement Document - Container Vulnerability Scanner

Development Action Items

Backend:

- Design database schema for images, vulnerabilities, CVEs, and scan history
- Integrate vulnerability scanners (Trivy, Clair, Grype)
- Develop RESTful APIs for image list, filters, details, and exporting data

Frontend:

- Build Dashboard, Image List, and Image Details UI
- Use ReactJS or VueJS; charts via D3.js or Chart.js

DevOps:

- Setup CI/CD to auto-trigger scans on new image push
- Configure CRON for regular scanning
- Enable security alerts via integrations (Slack, Email)

Security & Auth:

- Implement RBAC and OAuth2-based login
- Audit logging and encryption at rest/in-transit

Testing:

- Unit & integration tests for scan parsers and UI components
- Load testing for scale support with thousands of images