Product Requirement and Low-Fidelity Wireframes

Container Image Vulnerability Scanner

1 Overview

Container images package applications with their dependencies. However, these dependencies may include known vulnerabilities that pose security risks. Organizations managing thousands of container images need an efficient way to identify, assess, and prioritize the remediation of such vulnerabilities.

This document defines the requirements for a product that scans container images for vulnerabilities, displays findings clearly, and supports users in making data-driven decisions to remediate risks.

2 Goals

- Provide visibility into vulnerabilities across all container images.
- Allow users to identify high-risk (critical/high severity) images quickly.
- Enable filtering and sorting of image scan results at scale.
- Suggest actionable remediation steps for each vulnerability.

3 Target Users

- **DevOps Engineers:** Manage CI/CD pipelines and containerized infrastructure.
- **Security Analysts:** Monitor vulnerabilities and assess organizational risk.
- **Developers:** Maintain secure and up-to-date application images.

4 User Stories

- As a user, I want to view all container images and their vulnerability status.
- As a user, I want to filter images by severity to find critical/high risks.
- As a user, I want to click an image and see detailed CVE data and fix suggestions.
- As a user, I want to re-scan an image after updating or patching it.

5 Core Features

Feature	Description
Image List Dashboard	Display all container images with counts of vulnerabilities and severity.
Vulnerability Severity	Show CVEs categorized as Low, Medium, High, and Critical.
Filtering and Sorting	Allow filtering by severity level, image name, or vulnerability count.
Image Details Page	Detailed view of a single image's vulnerabilities with descriptions and CVE IDs.
Fix Recommendations	Provide steps to resolve vulnerabilities (e.g., upgrade base image or package).
Re-scan Option	Users can re-trigger scans for updated/fixed images.
Search Functionality	Search images by name or tag for quick access.

6 User Flow

1. User logs in to the system.
2. Dashboard shows a list of container images and their vulnerability summaries.
3. User applies filters to view high/critical severity images.
4. User clicks an image to view full CVE list with descriptions and fix instructions.
5. User applies fixes and clicks "Re-scan" to validate the updated image.

7 Wireframes

Low-fidelity wireframes are included separately and illustrate the following:

- Dashboard View (image list with severity indicators and filters).
- Image Detail View (CVE list, severity levels, fix suggestions).

8 Success Metrics

- Reduced number of unresolved critical/high vulnerabilities.
- Time-to-fix metrics improve across teams.
- High adoption rate of the scanning interface within DevOps/Security teams.

9 Optional: Developer Action Items

- Build API to list container images and scan data.
- Integrate a vulnerability scanning tool (e.g., Trivy, Clair, Grype).
- Create a responsive web interface for dashboard and details.

- Implement sorting/filtering/search features in the frontend.
- Enable manual and scheduled image re-scanning.
- (Optional) Add role-based access control for different user types.

10 Optional: Tech Stack Suggestions

- **Frontend:** React with Tailwind CSS, Figma for UI design.
- **Backend:** Node.js or Python (Flask/FastAPI).
- **Scanner:** Trivy, Clair, or Grype.
- **Database:** PostgreSQL or MongoDB.
- **Deployment:** Docker/Kubernetes.