Product Requirements Document (PRD)

Product Name: VulnScanner

Feature: Container Vulnerability Scanner Dashboard

Problem Statement

Container images bundle applications with all their dependencies. These images may include packages or libraries with known vulnerabilities. Organizations managing thousands of container images need a clear, efficient way to identify, prioritize, and remediate those vulnerabilities.

As a user:

- I need to understand which container images have vulnerabilities.
- I want to know the severity of those vulnerabilities (Critical, High, Medium, Low).
- I want to take quick action on the most critical images needing a fix.
- I should be able to view, scan, and get reports across thousands of container images efficiently.

Goals & Objectives

- Provide a centralized dashboard showing the vulnerability status of container images.
- Enable users to perform quick scans and display real-time results.
- Highlight containers with Critical and High vulnerabilities for prioritization.
- Offer actionable insights and remediation options.
- Allow users to view detailed reports and history.

Key Features

1. **Dashboard Overview**
    - Summary of scanned container images.
    - Quick access to critical vulnerability statuses.

- Filter/sort containers by severity or number of vulnerabilities.

2. **Scan Management**
    - Start new scans.
    - View scan history and progress.
    - Schedule recurring scans.
3. **Vulnerability Details View**
    - List of detected vulnerabilities per container.
    - Severity level and description.
    - Suggested fix (e.g., upgrade library/package, rebase image).
4. **Reports**
    - Downloadable reports in PDF/JSON formats.
    - Timeline of vulnerabilities over time.
    - Remediation status and compliance checks.
5. **Settings**
    - Configure scan preferences.
    - Enable/disable notifications.
    - Manage API keys and image repository integrations.

User Flow Summary

1. User logs into VulnScanner.
2. Lands on Dashboard showing all container images and their current vulnerability status.
3. User clicks Start New Scan to initiate a scan.
4. Scanned results are shown as individual container cards.
5. User can click View & Fix to drill down into specific vulnerabilities and remediation suggestions.
6. Reports are accessible under the Reports section for auditing or export.

Wireframe Overview

The interface presents:

- A sidebar for navigation (Dashboard, Scan, Reports, Settings).
- A call-to-action button: Start New Scan.
- Cards displaying:
    - Container name
    - Vulnerability status with severity badges (Critical, Medium, Low)
    - Count of detected vulnerabilities
    - A View & Fix button for each container

Success Metrics

- 90%+ scan completion rate across container repositories
- 80%+ user engagement with remediation suggestions
- Reduced average time to identify & fix critical vulnerabilities
- Positive feedback from end users on dashboard usability

Bonus: Development Action Items

These are initial development tasks that can be discussed with the engineering team:

Frontend

- Implement responsive UI layout as per wireframe.
- Build Dashboard using React (or preferred frontend tech).
- Connect container cards to API endpoints dynamically.
- Integrate View & Fix modal with detailed vulnerability data.
- Add filters/sorting to the container list by severity/date.

Backend

- Integrate with container vulnerability scanners like Trivy or Grype.
- Build REST API to fetch and serve scan results.

- Design DB schema for storing image scans and vulnerabilities (e.g., MongoDB or PostgreSQL).
- Implement job scheduler for periodic scans (e.g., cron jobs or background workers).
- Create endpoints for reports and settings configuration.

Security & Performance

- Secure scan endpoints with auth tokens or OAuth.
- Optimize scan performance for large container repositories.
- Handle rate limits and retry logic when scanning external registries.

DevOps

- Containerize the app itself using Docker.
- Set up CI/CD pipelines for automated build and test.
- Deploy on cloud (e.g., AWS, GCP, Azure) with logging and monitoring.