Product Requirements Document (PRD)

Product Name:	Container Image Vulnerability Scanner
Author:	Supritha (PM Applicant)
Date:	30-05-2025
Version:	1.0
Status:	Final-Ready for submission

1. Overview (Purpose)

The goal of this product is to provide a centralized dashboard that scans container images for known vulnerabilities and helps users prioritize and remediate those vulnerabilities based on severity.

The system will help teams secure their container images at scale, even when dealing with thousands of images.

2. Background (Problem Statement)

- Container images often bundle code, dependencies, libraries, and OS-level packages — many of which may have security vulnerabilities.
- Users need to know which images are vulnerable.
- Users need to know how bad (severity) the vulnerabilities are.
- With thousands of images, users need a fast and smart way to identify and fix critical/high-risk images.

3. User Personas:

- DevOps Engineers
- Security Analysts
- Cloud Infrastructure Engineers
- Developers in CI/CD pipelines

4. Key Features & Requirements

Feature	Description	Priority (High/Med/Low)
Vulnerability Scan	Scan all container images for known vulnerabilities	High
Vulnerability Severity Display	Show severity levels with color codes (Red, Orange)	High
Search & Filter	Search images by name, filter by severity	High
Notifications	Alerts for critical or high vulnerabilities	Medium
Bulk Actions	Ability to select & fix multiple images	Medium
Dashboard Overview	Summary of vulnerabilities & stats	High

5. UI Requirements

Title/Header: "Container Image Vulnerability Scanner"

Metrics Summary Cards:

Total Container Images

Images with Critical Vulnerabilities

Images with High Vulnerabilities

Fixed Images

Primary Button:

Scan New – triggers new image scan flow

Filter Dropdown (Optional):

Allow filtering by scan status, severity, or date range.

6. Technical Requirements

- Integration with container registries (DockerHub, ECR, etc.)
- Vulnerability DB source: e.g., CVE Database, Trivy, or Clair
- Authentication (OAuth/SAML support)
- REST API support for integration with CI/CD tools

7. Security Requirements

- Role-based access control (RBAC)
- Secure handling of registry credentials (encrypted at rest)
- Scan data must not be publicly exposed or shared

8. Solution Approach

We propose a web-based dashboard that scans container images for known vulnerabilities and displays findings in a clear, actionable format.

Users will be able to:

- View vulnerability details per image
- Filter results by severity (critical , high, medium, low)
- Prioritize images needing urgent fixes
- Receive recommendations for remediation
- Export reports and set up alert notifications

The goal is to help users manage and reduce risk across large repositories efficiently.

9. Success Metrics

Metric	Target
% of know vulnerabilities correctly identified in container images	95%+ accuracy
Average time from detective to fix	<24 hours
Weekly active users engaging with the dashboard	85% of active customers
# of registries and images scanned per customer	500+
User satisfaction	4.5/5 or higher.

10. Development Considerations (Optional)

- Set up container registry scanning using Trivy or Clair
- Build REST API to fetch scan results
- Store scan data in MongoDB or PostgreSQL
- Build React/Next.js frontend for dashboard
- Deploy using Kubernetes

11. Assumptions

- Users are aware of container image vulnerabilities and know the importance of fixing them.
- The organization uses a centralized container registry where images can be accessed and scanned
- Users have permission to initiate scans and apply fixes on the container images
- The product will integrate smoothly into existing CI/CD pipelines or DevSecOps workflows
- Users will be technically capable of interpreting the scan results and taking action based on severity levels

12. Risks

Risk	Impact	mitigation
Incomplete or outdated CVE database	Users may miss critical vulnerabilities	Integrate with multiple trusted CVE sources and schedule regular updates
High false positive/negative rates in scans	Reduces user trust in the tool	Use proven scanning engines and allow user feedback to the tune rules
Scalability issues with scanning thousands of images	Performance degradation	Implement parallel scan processing and prioritize based on usage frequency
Poor user experience or confusing UI	Users might not use the tool	Conduct usability testing and iterate wireframes early
Limited devops integration	Reduced adoption	provides easy APIs and plugins for popular CI/CD tools
Access or security constraints in enterprise environments	Scan fails or access denied	Offer self-hosted /on-prem options if needed

13. Appendix

CVE: common vulnerabilities and exposure.

Severity levels: Critical, High, Medium, And Low.

Image registry: A storage for container images.

Vulnerability Scanner: A tool to detect security issues.

References:

Docker Hub

Scanner tools

CVE database