Product Requirements Document (PRD)

Vulnerability Management Dashboard for Container Images

Version: 1.0

Date: April 11, 2025

Author: Peeyush Yadav

Table of Contents

1. Introduction

This document outlines the product requirements for a security solution that scans container images for known vulnerabilities. The solution will provide users with insights and actionable paths to manage and mitigate vulnerabilities in a large repository of images.

---

2. Background

Containerized environments have become standard in deploying scalable applications. However, containers bundle applications along with their dependencies, many of which might have security vulnerabilities. Security teams need an effective way to monitor and respond to these risks, especially when dealing with thousands of images.

---

3. Objectives

- Provide comprehensive visibility into container image vulnerabilities with severity-based classification
- Enable rapid identification and prioritization of images requiring remediation
- Support repositories with thousands of images through efficient filtering and search capabilities
- Reduce time-to-remediation for critical and high vulnerabilities by 70%
- Improve overall security posture with actionable insights and clear remediation paths

---

4. Use Cases

- View a high-level overview of image vulnerabilities.
- Search and filter container images by severity.
- Drill into details of a specific image’s vulnerabilities.
- Export vulnerability reports for auditing.
- Configure alerts for critical vulnerabilities.

---

5. Functional Requirements

1. **Vulnerability Dashboard**
    - Overview of all container images and their vulnerability status
    - Visualization of vulnerability count by severity level
    - Trending data showing security posture over time
    - Quick filtering by critical factors (severity, team, application)
2. **Image Repository Scanner**
    - Support for scanning thousands of container images
    - Integration with common container registries
    - Scheduled and on-demand scanning capabilities
    - Differential analysis to identify new vulnerabilities
3. **Detailed Vulnerability View**
    - Comprehensive list of vulnerabilities per container image
    - Detailed information for each vulnerability (CVE ID, severity, affected component)
    - Remediation guidance with suggested fixes
    - Context about exploit potential and business impact
4. **Filtering and Search System**
    - Advanced filtering by multiple criteria (severity, age, team, application)
    - Saved search functionality for common queries
    - Tag-based organization for large repositories
    - Smart sorting to prioritize most critical issues
5. **Remediation Workflow**
    - Assign vulnerabilities to teams/individuals
    - Track remediation status and progress

- Integration with notification systems (email, Slack)
- Exception management with justification documentation

---

6. Time Frame

TARGET DATE	MILESTONE	DESCRIPTION	EXIT CRITERIA
2025-05-15	Alpha	Internal testing with security team	No P0 bugs for 7 consecutive days
2025-06-01	Beta	Early access	10+ users actively using product
2025-07-15	Early Access	Expanded EA	Positive feedback from 80% of users
2025-08-15	Launch	Available to all	Stable performance with full feature set

7. Key Flows

Dashboard Overview Flow

1. User logs into the system
2. System presents dashboard showing:
    - Total number of container images
    - Number of vulnerable images by severity
    - Trend chart showing vulnerability counts over time
    - List of top 10 most vulnerable images that need immediate attention
3. User can filter by team, application, or date range
4. User can click any section to drill down for more details

Vulnerability Discovery Flow

1. User selects an image from the dashboard or search results

2. System displays image details page showing:
    - Image metadata (name, tag, size, creation date)
    - List of all vulnerabilities grouped by severity
    - Affected components and versions
    - CVE IDs with links to detailed information
3. User can sort and filter vulnerabilities
4. User can export findings or share with team members

Remediation Workflow Flow

1. User identifies a critical vulnerability from image details
2. User clicks "Create Remediation Task" button
3. System displays remediation form with:
    - Vulnerability details pre-populated
    - Assignment dropdown for team/individual
    - Due date selector
    - Priority level
    - Optional notes field
4. User completes form and submits
5. System notifies assignee and tracks status
6. Assignee updates status when fixed
7. System verifies fix with next scan and updates status

---

8. Key Logic

1. **Vulnerability Severity Classification (CVSS Score):**
    - **Critical:** 9.0-10.0

- **High:** 7.0-8.9
- **Medium:** 4.0-6.9
- **Low:** 0.1-3.9

2. **Image Prioritization Logic:**
    - Priority = (Critical × 10 + High × 5 + Medium × 2 + Low × 1) × Usage Factor
    - Usage Factor is higher for production images than development/test images
    - Recently updated images with new vulnerabilities are weighted higher
    - Images used by multiple applications get higher priority
3. **Alert Thresholds:**
    - Immediate notification for any new critical vulnerability
    - Daily digest for high vulnerabilities
    - Weekly summary for medium and low vulnerabilities
4. **Edge Cases:**
    - Handle custom/proprietary base images without public CVE data
    - Support air-gapped environments with offline scanning capability
    - Process for handling false positives with override documentation

---

9. Assumptions and Constraints

- Users will have access to scan their image repositories.
- Open-source vulnerability scanners (e.g., Trivy) will be integrated.
- The product must operate within typical enterprise security compliance.

---

10. Dependencies

- Container registry access (DockerHub, AWS ECR, etc.)

- Vulnerability scanner APIs
- Frontend framework
- Backend services and databases for storing scan results

---

11. Wireframes (Low-Fidelity)

Dashboard View

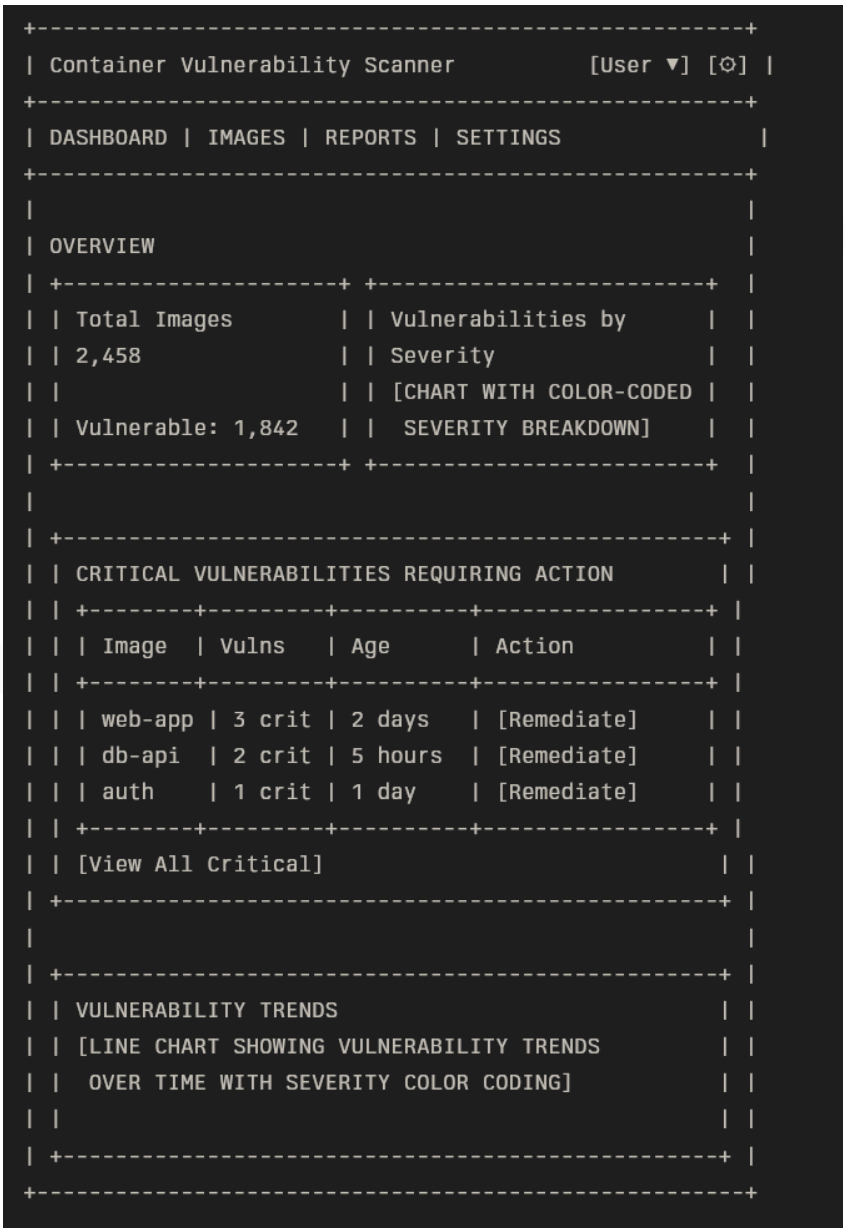
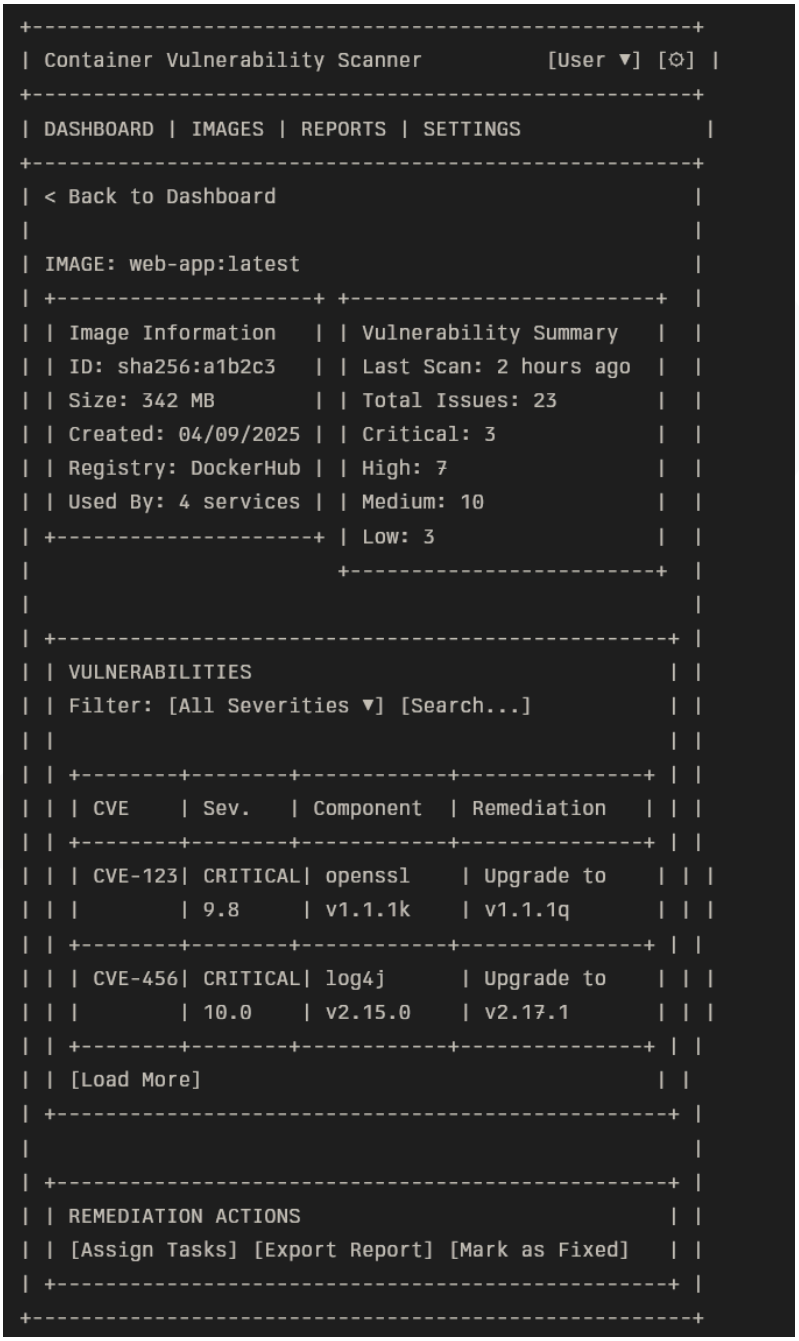
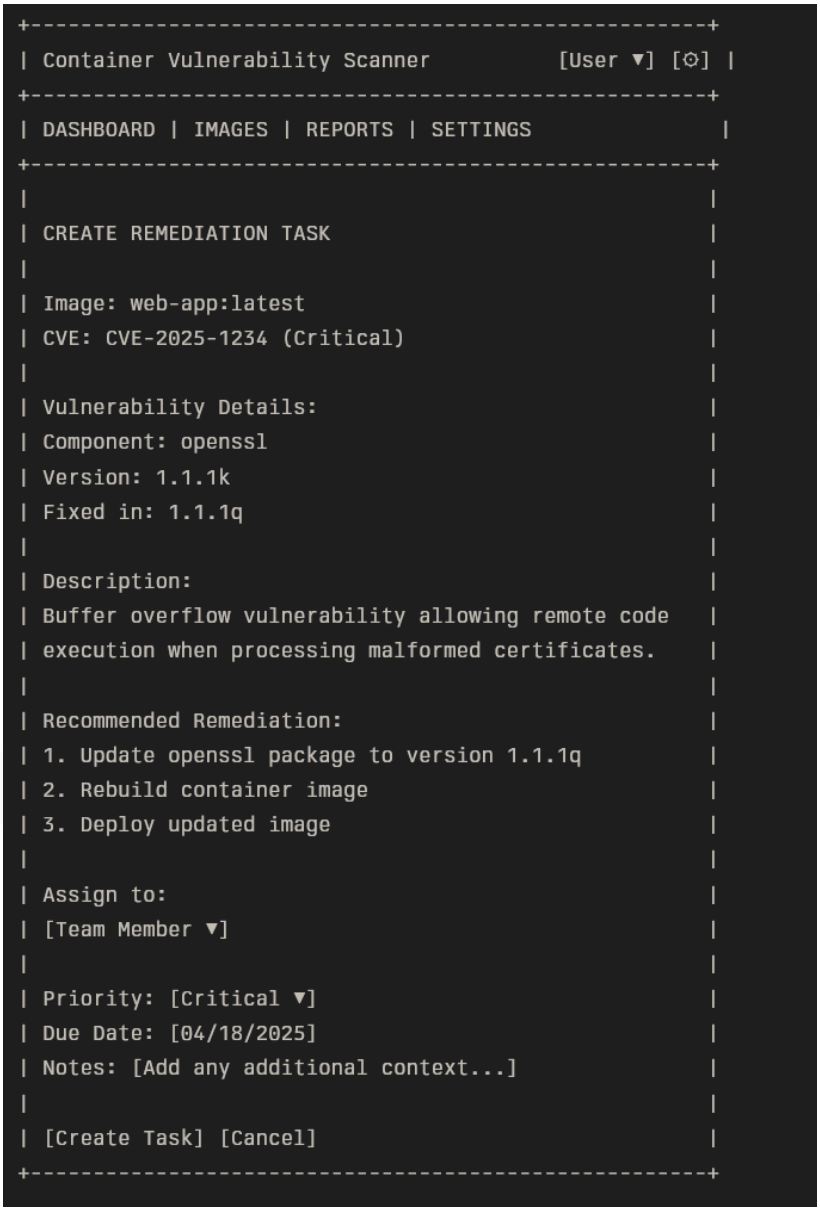


Image Detail View



Remediation View



---

12. Development Action Items

1. **Backend Development:**

- Create container image scanning engine using industry-standard tools
- Develop vulnerability database integration with CVE tracking
- Build RESTful API for frontend and third-party integration
- Implement efficient data storage for large image repositories
- Create notification system for alerts and reports

2. **Frontend Development:**
    - Build responsive dashboard with data visualization components
    - Implement advanced filtering and search functionality
    - Create detailed image and vulnerability views
    - Develop remediation workflow components
    - Implement user preference settings and configuration

3. **Infrastructure:**
    - Design scalable architecture to handle large repositories
    - Implement secure API authentication and authorization
    - Create deployment pipeline for continuous integration
    - Develop performance testing framework

4. **Security:**
    - Implement secure scanning methodology
    - Ensure vulnerability database is regularly updated
    - Create secure data handling procedures
    - Perform security review of the application itself

---

13. Open Questions

1. What is the optimal scanning frequency for large repositories?
2. How should we handle container images that cannot be easily updated?
3. What metrics will best demonstrate security improvement over time?
4. Should we integrate directly with vulnerability management tools?

END