Product Requirements Document (PRD) - Container Image Vulnerability Scanner

**1. Introduction**

This document outlines the product requirements for a security product designed to scan container images and present vulnerability findings to users. The primary goal is to help users understand, prioritize, and remediate vulnerabilities within their container image repositories, especially given the presence of thousands of images.

**2. Goals**

* Enable users to quickly identify which container images have vulnerabilities.
* Clearly indicate the severity of identified vulnerabilities (e.g., critical, high, medium, low).
* Help users prioritize remediation efforts by highlighting images with critical or high vulnerabilities.
* Provide a scalable solution capable of handling thousands of container images in a repository.
* Offer clear information to assist users in fixing identified vulnerabilities.

**3. Background/Task**

Container images are fundamental components of modern application deployment, containing applications and their dependencies. These components can have known vulnerabilities that pose security risks. Users need a product that helps them understand these findings and take corrective action.

**4. User Stories / Features**

As a user:

* I need to understand which container images have vulnerabilities and how severe they are, so I can assess my overall security posture.
* If there are any critical or high vulnerabilities, I need to fix them and thus need to identify which images have to be fixed, so I can prioritize my remediation efforts.
* I have thousands of images in my repository, so the product needs to efficiently manage and display information for a large number of assets.
* I want to see a clear list or dashboard of my container images, indicating their vulnerability status at a glance.
* I want to be able to filter and sort images based on vulnerability severity to quickly find critical issues.
* I want to view detailed information for each vulnerability, including its description, associated CVE, and potential remediation steps.
* I want to know when a vulnerability has a known fix or a recommended patched version of a component.
* I want to track the remediation progress of vulnerabilities over time.

**5. Low-Fidelity Wireframe Concepts**

*(Note: These are conceptual descriptions for low-fidelity wireframes. Actual wireframes would involve sketching out these layouts.)*

**5.1. Dashboard View**

* **Purpose:** Provide an at-a-glance overview of the overall security posture of container images.
* **Elements:**
  + **Vulnerability Summary Card:** Displays counts of Critical, High, Medium, and Low vulnerabilities across all scanned images.
  + **Top 5 Most Vulnerable Images List:** A short list of images with the highest number of critical/high vulnerabilities.
  + **Scan Activity Chart:** A simple graph showing recent scan history or newly identified vulnerabilities over time.
  + **Quick Search Bar:** For immediate lookup of specific image names or vulnerability IDs.

**5.2. Image List View**

* **Purpose:** Allow users to browse and filter their container images based on vulnerability status.
* **Elements:**
  + **Table Layout:**
    - **Columns:** Image Name/Tag, Repository, Last Scanned, Total Vulnerabilities, Critical, High, Medium, Low.
    - **Sorting:** Clickable column headers to sort by name, scan date, or any vulnerability count.
    - **Filters:** Dropdowns or checkboxes for:
      * Severity (Critical, High, Medium, Low)
      * Repository Name
      * Image Status (e.g., "Needs Fix" for critical/high vulnerabilities)
      * Search bar for keyword search (image name, tag).
  + **Pagination/Infinite Scroll:** To handle thousands of images efficiently.
  + **Call to Action:** "Scan New Image" button (if applicable).

**5.3. Image Detail View**

* **Purpose:** Provide in-depth information about vulnerabilities found within a specific container image.
* **Elements:**
  + **Image Header:** Image Name/Tag, Repository, Last Scanned, Overall Vulnerability Summary (e.g., "Contains 3 Critical, 5 High").
  + **Vulnerability List (Table):**
    - **Columns:** CVE ID, Severity, Component/Package, Vulnerability Description (brief), Fixed Version (if available), Recommended Action.
    - **Sorting/Filtering:** Ability to sort by severity, component, etc.
  + **Detailed Vulnerability Pane (on click/expand):** When a user clicks on a specific vulnerability in the list, a side panel or expanded row appears with:
    - Full Description
    - CVSS Score
    - References (links to NVD, vendor advisories)
    - Impact
    - Exact Remediation Steps/Patched Version

**6. Future Considerations / Bonus Task: Development Action Items**

* **API Design:** Define RESTful APIs for image submission, scanning triggers, and retrieving scan results.
* **Data Model:** Design the database schema for storing container image metadata, scan jobs, and vulnerability findings (CVEs, severity, affected components, fixed versions).
* **Integration Points:**
  + Identify potential integrations with CI/CD pipelines (e.g., Jenkins, GitLab CI) for automated scanning.
  + Consider integration with vulnerability databases (e.g., NVD, vendor-specific feeds).
  + Explore integration with repository managers (e.g., Docker Hub, Artifactory, Quay) for pulling images.
* **Scanning Engine Selection:** Research and potentially integrate with existing open-source or commercial container image scanning tools (e.g., Trivy, Clair, Anchore).
* **Scalability & Performance:** Plan for a microservices architecture to handle concurrent scans and large data volumes. Implement efficient indexing and caching strategies for the thousands of images.
* **Notification System:** Implement alerts for critical/high vulnerabilities (e.g., email, Slack).
* **Reporting:** Develop capabilities for generating compliance reports or historical trend analysis.
* **Authentication & Authorization:** Secure access to scan results and sensitive image information.