1. Product Requirements Document (PRD)

Title: Container Image Vulnerability Scanner

1.1 Overview

The Container Image Vulnerability Scanner is designed to identify and report vulnerabilities in container images stored in a repository. The tool will scan all container images, prioritize the vulnerabilities based on their severity, and provide actionable insights for fixing critical issues.

1.2 Objectives

- Help users identify which container images have vulnerabilities.
- Prioritize vulnerabilities based on severity (Critical, High, Medium, Low).
- Enable users to fix critical and high-severity vulnerabilities by giving them detailed information about the affected images.
- Support large repositories with thousands of images by providing scalable and efficient scanning.

1.3 Key Features

- **Dashboard Overview**: Display a summary of vulnerabilities across all container images.
- **Severity-Based Filtering**: Allow users to filter images based on vulnerability severity (Critical, High, Medium, Low).
- **Search and Sort**: Provide search functionality by image name and the ability to sort results based on severity, image name, or last scan date.
- **Detailed Image Report**: Show detailed information about each image, including:
    - Affected packages
    - Vulnerability description
    - Available patches or remediation steps
    - Last scanned date
- **Bulk Action**: Allow users to initiate scans for multiple images and fix vulnerabilities across many images at once.
- **Alerts/Notifications**: Notify users when new vulnerabilities are discovered in critical or high-severity categories.

1.4 User Personas

- **Security Analyst**: Needs to know which images are vulnerable and prioritize based on severity.
- **DevOps Engineer**: Responsible for maintaining the repository and ensuring that images are vulnerability-free.
- **Developer**: Works on fixing vulnerabilities in application code and dependencies.

1.5 User Stories

1. **As a user**, I want to scan all container images in my repository to identify vulnerabilities.
2. **As a user**, I want to filter images based on the severity of the vulnerabilities (e.g., Critical or High).
3. **As a user**, I need a clear indication of which vulnerabilities need immediate attention.
4. **As a user**, I want detailed information about each vulnerability, including potential fixes.
5. **As a user**, I want to receive notifications when new critical vulnerabilities are found.

1.6 Functional Requirements

- **Scanning**:
    - The system should scan container images for known vulnerabilities using a database of vulnerabilities (such as CVEs).
    - Scans should be scheduled or run on-demand.
- **Filtering & Sorting**:
    - The system should allow users to filter container images by vulnerability severity.
    - Users should be able to sort images by the severity of vulnerabilities, last scan date, or image name.
- **Notifications**:
    - Users should be notified when new critical vulnerabilities are discovered in any of their images.
- **User Interface**:
    - The dashboard should display an overview of total images scanned, total vulnerabilities found, and the distribution by severity.
    - A detailed report for each container image should be accessible by clicking on the image.

1.7 Non-Functional Requirements

- **Performance**: The system should be able to scan thousands of images in a scalable and efficient manner.
- **Security**: All communication between the scanner and the user interface must be encrypted.
- **Usability**: The interface should be intuitive, allowing users to quickly understand and act upon scan results.

1.8 Success Metrics

- Reduced number of critical vulnerabilities in container images over time.
- Decreased time taken by users to identify and fix vulnerabilities.

2. Low-Fidelity Wireframes

Below is a description of the low-fidelity wireframes for the UI:

2.1 Dashboard Overview

- **Top Navigation**: Links to "Dashboard," "Images," "Settings," and "Notifications."
- **Summary Cards**:
    - Total container images scanned.
    - Total vulnerabilities found.
    - Breakdown of vulnerabilities by severity (Critical, High, Medium, Low) in a pie chart or bar chart.
- **Recent Critical Findings Table**: A table showing the most recent critical/high vulnerabilities with columns for:
    - Image name
    - Number of vulnerabilities
    - Last scanned date
    - Severity

2.2 Image List View

- **Filter and Search Bar**: Options to filter by severity and search by image name.
- **Image Table**: A table listing all images with columns for:
    - Image name
    - Total vulnerabilities
    - Critical vulnerabilities
    - Last scan date
    - Fix available (Yes/No)
- **Bulk Actions**: A checkbox next to each image and buttons to "Rescan" or "Fix Vulnerabilities."

2.3 Detailed Image View

- **Image Header**: Image name, last scanned date, and total vulnerabilities.
- **Vulnerabilities List**: A list of vulnerabilities for the image with details like:
    - Affected package
    - Vulnerability severity
    - Vulnerability description
    - CVE link
    - Fix available (Yes/No)
- **Fix Action**: A button to apply patches or download instructions.

I can also create these wireframes using tools like Figma or hand sketches if needed.

3. Development Action Items

Here are a few technical items that can be discussed with the development team:

1. **Integrate a vulnerability scanning tool** (e.g., Clair, Trivy) into the system to scan container images.
2. **Build a scheduling service** to automatically scan images at regular intervals.
3. **Design scalable storage** for storing the results of scans, especially when dealing with large repositories.
4. **Implement a notification system** to send alerts for critical vulnerabilities.
5. **Create APIs** to allow users to initiate scans, retrieve scan results, and trigger vulnerability fixes via the frontend.
6. **Security considerations**: Secure the communication between the scanner, the repository, and the UI using encryption.