Project Name: Vulnerability Scanning Dashboard

Document Owner: Amit Singh | Last Update: 29-09-2024

| Jira Epic | [Link for epic link for reference-helpful to track engineering team’s progress] |
| --- | --- |
| Delivery Date | TBD |
| Document Status | On Time |
| Product Lead | Amit Singh |
| Engineering Lead | [Your Tech Lead’s name] |
| Designer Lead | [Your Product designer’s name] |
| Quality Assurance (QA) Lead | [Your Testing/QA name] |
| Relevant Documents | [Link to any related documents,, confluence or wiki pages,etc] |

### **1. Overview**

#### **Why are we doing this?**

* With the rise in the use of containerized applications, ensuring security across container images is a priority. Many vulnerabilities are tied to components within container images, leading to potential security risks. This product addresses this critical need.

#### **How do we know this is a real problem and worth solving?**

* Increasing number of reports and incidents due to vulnerabilities in container images. Security teams struggle to keep track of thousands of images with various risk levels.

#### **How does this fit into the overall company objectives?**

* Enhances the security posture of the company by mitigating risks in the CI/CD pipeline. A more secure infrastructure increases trust and decreases the chance of breaches, aligning with long-term growth strategies.

#### **What problem is this solving?**

* Identifying, highlighting, and prioritizing container images with vulnerabilities.

#### **Who are we solving for?**

* Security teams, DevOps engineers, and developers managing containerized applications.

#### **What are the user benefits?**

* A streamlined and efficient process to scan, identify, and prioritize fixes for vulnerable images.

### **2. Goals**

* Provide users with a clear dashboard of vulnerabilities per container image.
* Enable users to prioritize remediation based on the severity (critical, high, medium, low).
* Ensure scalability to handle large repositories with thousands of images.
* Provide easy-to-understand visualizations of findings to help users make informed decisions.

### **3. Success Metrics**

To measure the success of this solution, we will track the following metrics:

* **Metric 1:** Percentage reduction in the time taken to identify vulnerable images.
* **Metric 2:** Number of resolved vulnerabilities post-scan per sprint.
* **Metric 3:** User satisfaction score based on ease of use and efficiency.

### **4. Assumptions & Dependencies**

* **Assumptions:**
  + Users will have large image repositories (thousands of images).
  + The product will integrate with existing CI/CD pipelines.
  + Users will prioritize vulnerability scanning as part of their deployment process.
* **Dependencies:**
  + Integration with vulnerability databases (e.g., CVE database).
  + Dependency on the container registry or image repository for data ingestion.
  + Collaboration with the DevOps and security teams for access and infrastructure support.

### 

### **5. Requirements**

| **S.No** | **Title** | **User Story** | **Acceptance Criteria** | **Priority** | **Jira Link** | **Notes** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Vulnerability Scanning Dashboard | As a user, I want to see a dashboard listing all scanned container images along with their vulnerabilities and severity levels, so that I can prioritize fixes. | - The dashboard must display a list of container images.  - Vulnerabilities should be categorized by severity (Critical, High, Medium, Low). | Must Have | TBD | Discuss UI details with designer and dev team. |
| 2 | Vulnerability Drilldown | As a user, I want to drill down into a specific container image to view details of its vulnerabilities, so that I can understand the risk of each issue. | - Each container image must display a list of associated vulnerabilities.  - Each vulnerability must include a description, severity, and fix. | Must Have | TBD | Requires integration with vulnerability databases. |
| 3 | Bulk Image Scanning | As a user, I want to scan thousands of container images at once, so that I can manage large repositories effectively. | - The system should support bulk scanning of large image repositories.  - Users should be notified of scan progress and completion. | Should Have | TBD | Requires testing for scalability. |
| 4 | Remediation Suggestions | As a user, I want the system to provide actionable suggestions for fixing vulnerabilities, so that I can efficiently mitigate security risks. | - Each critical and high-severity vulnerability must provide remediation steps.  - Users must be able to export a list of suggested fixes for developer action. | Must Have | TBD | Requires collaboration with security team. |

### **6. UX Mocks**

* Low-fidelity wireframes will show:
  1. A dashboard view with all images and their vulnerability statuses.
  2. A detailed view of individual images showing vulnerability details.
  3. Filters to sort by severity or image date.

### **7. Questions**

| **Questions** | **Outcomes** |
| --- | --- |
| How should we prioritize vulnerabilities when several images have critical issues? | Will require a discussion with security teams to determine prioritization strategy. |
| Should the product handle automatic remediation? | Needs to be discussed with development and security teams—may add complexity but could be valuable long term. |

### **8. Out of Scope**

* **Automatic vulnerability fixes:** While suggestions will be provided, this version will not automate the remediation process.
* **Integration with all CI/CD tools:** Focus will be on commonly used tools; others will not be part of the first release.