Product Requirement and Low-Fidelity Wireframes

Product Requirements Document (PRD)
Product Name: VulneraDock
Customer Name: Xyz LTD (Confidential)
Document Owner: Product Management Team, AccuKnox
Date: 12 March 2025

Revision History

Version	Date	Author	Changes	Approval
1.0	DD/MM/YYYY	Prajwal D	Initial PRD based on customer req	

---

1. Objective
Enable Xyz LTD to have a Single Pane of Glass (SPoG) dashboard that identifies, prioritizes, and remediates vulnerabilities in container images at scale (1,000+ images) with minimal manual effort. The solution will adhere to Zero Trust principles by enforcing least privilege access, ensuring secure interactions, and delivering actionable insights to help drive remediation.

2. Background
Xyz LTD team struggles to:
- Track vulnerabilities across 1,000+ container images.
- Prioritize critical/high-severity risks.

3. User Requirements Collected during Day-0
- A dashboard that lists all container images along with vulnerability counts and severity levels, so that they can quickly identify risky images.
- Filter and sort images by severity, scan date, or repository so that they can prioritize remediation tasks.
- Detailed views of each image's vulnerabilities (including CVE IDs, descriptions, CVSS scores, and remediation recommendations) to plan fixes.

4. Functional Requirements
- Image Inventory & Scanning
  Repository Integration:
  - Connect to container registries (e.g., Docker Hub, AWS ECR, GCP Artifact Registry) to pull image metadata.
  - Automatically trigger vulnerability scans (using tools like Trivy/Clair or Docker Scout) for new or updated images.

Scan Scheduling & Automation:

- Schedule regular scans and support on-demand scanning.
- Maintain historical scan data for trend analysis.

Remediation Guidance;

- Link CVEs to patches/base image upgrades.

- Vulnerability Data & Prioritization
  - Collect vulnerability data, including severity (Critical, Medium, Low), CVE identifiers, and remediation recommendations.
  - Display aggregated risk scores for each image.
- Filtering & Sorting
  - Filtering Options: By severity (Critical, High, etc.), By image repository, tag, or name, scan date.

5. Non-Functional Requirements

- The system should load lists quickly even when handling thousands of images.
- The product must support dynamic scaling as the number of images grows.
- Optimize database queries for filtering and sorting.
- Secure API integration with registries and scanning tools.
- Alerting mechanism; Integration with email/Slack/incident management tools.

6. Technical Specifications

Architecture

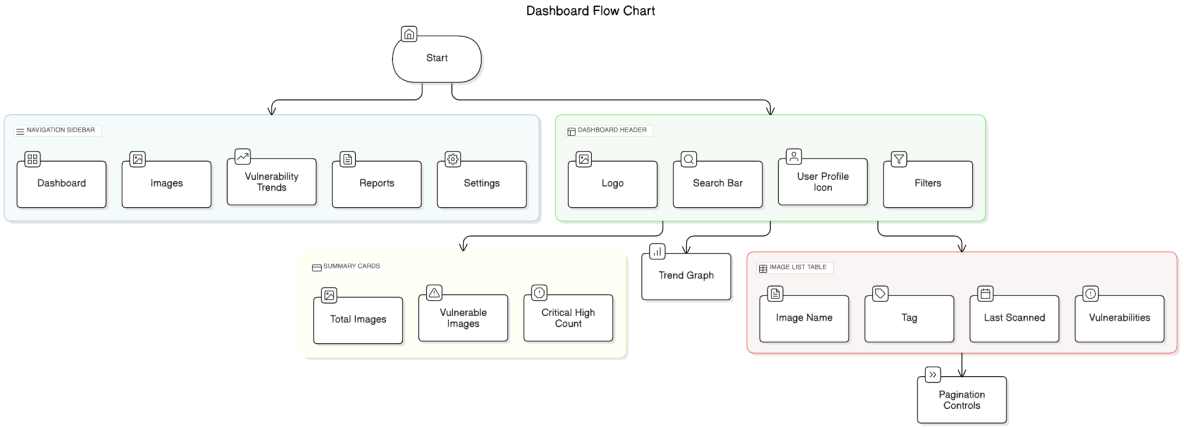
- Scanner Integration: Trivy (default) + Clair (optional).
- Backend:
  REST APIs for registry connections.
  Golang microservices with Kubernetes orchestration.
- Frontend: React-based dashboard with real-time updates.
- Database: PostgreSQL (metadata) + Elasticsearch (CVE search).

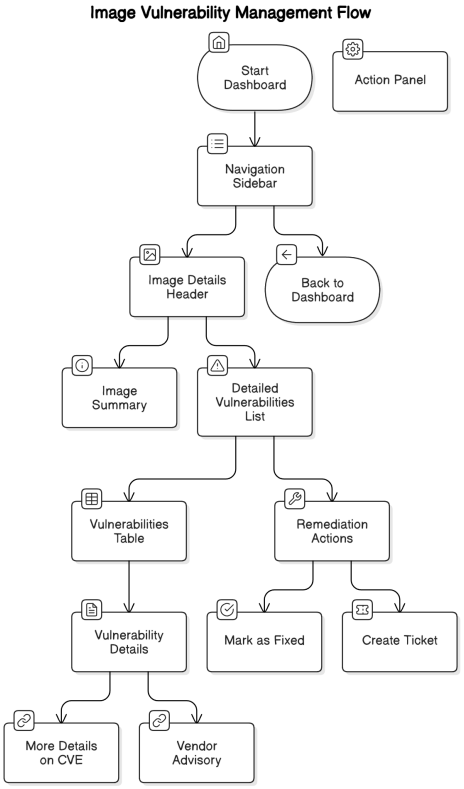
Integrations

- CI/CD: Jenkins, GitHub Actions, GitLab.
- Ticketing: Jira, ServiceNow (auto-create tickets for critical CVEs).

7. Implementation Roadmap

1. Phase 1 – MVP:
- Core dashboard with image inventory, vulnerability scanning integration, and basic filtering.
- Detailed view per image with vulnerability listing.
2. Phase 2 – Advanced Features:
- Integration with ticketing/CI systems.
3. Phase 3 – Optimization:
- Performance enhancements and scalability improvements.





8. Team Member Responsibilities

Role	Responsibilities
Product Management	PRD, roadmap, KPIs
Engineering	Backend, APIs, UI
Security	CVE validation, RBAC
QA	Load testing, bug tracking

9. Assumptions & Dependencies

- Customer uses AWS ECR/Docker Hub as primary registries.
- Customer has existing CI/CD pipelines (Jenkins/GitLab).
- Compliance requirements include NIST and CIS benchmarks.

10. Out-of-Scope Items

- Real-time scanning during image build.
- Custom plugins for non-standard registries.

11. Timeline

Phase	Activities	Start Date	End Date

12. Success Metrics

13. Next Steps - Action items

- Approve PRD
- Kick off MVP development with Engineering.
- Discussed the securities tools to add in the product as per the user's environment
- Ensure our solution connects seamlessly with the client's registry.
- Develop a Single Pane of Glass dashboard for centralized monitoring of image vulnerabilities.
- How can we integrate our workflow with CI/CD pipelines and ticketing systems?
- Also, can we integrate Kubescape works via an API with no graphical user interface + ARMO.

Above Points can be discussed with the development team

This document aligns with AccuKnox's Zero Trust CNAPP strategy and addresses customer requirements for scalable container security.