**Container Image Vulnerability Dashboard – PRD**

**Author : Sahil Jain**

**Overview**

A security-focused dashboard that enables security engineers to identify, prioritize, and act on vulnerabilities across thousands of container images. This product automatically scans connected repositories, surfaces critical risks at scale, and empowers users to filter, investigate, and track remediation efficiently.

**Gap in Market**

The Modern Businesses building scalable apps , these days user Micro-service architecture instead of monolith architecture. While these sound easy but the pain point is while maintaining containers ,pods and Images . For Apps which is deployed there a lot of dependencies and there might be known vulnerabilities for the dependencies .  
As a Security Engineer maintaining and sifting though thousands of Images , maintaining wouldn’t be easy .Pinpoint the most critical issues and facilitate remediation could be quick and easy .  
Delays or blind spots can lead to breaches, compliance failures, and costly incidents.

**Goals**

**Business Goals**

* Decrease the average time to detect critical container vulnerabilities across customer environments.
* Increase product adoption among enterprise security teams by providing actionable, high-clarity vulnerability insights.
* Enhance customer retention by integrating the dashboard seamlessly into users’ existing security workflows.

**User Goals**

* Visualize vulnerability status across all container images at a glance.
* Rapidly identify which images contain critical or high-severity vulnerabilities.
* Filter, search, and drill into image-level and vulnerability-level details for targeted investigation.
* Track remediation actions and progress over time.
* Provide documentation for the known vulnerability and ways to fix it .

**Non-Goals**

* Automated patching or re-building of container images.

Runtime protection, this product focuses solely on pre-deployment image scanning.

* Support for vulnerability scanning in non-container environments.

**User Personas & User Stories**

**Primary Persona:**

* **Security Engineer / DevSecOps Engineer**
* **Lead Security Engineer**

**User Stories:-**

|  |  |
| --- | --- |
| **User Stories** | **Acceptance Criteria** |
| As a Security Engineer | 1. User can see a dashboard of all container images and their vulnerability summary, so they can quickly identify critical risks. 2. User can filter images by severity and search by image name, so they can focus on images that need immediate attention. 3. User can see full down into a specific image to view the full list of vulnerabilities, so they can assess impact and plan remediation. 4. User can mark vulnerabilities or images as fixed or acknowledged, so they can keep track of progress and avoid duplicate work. |
| As a DevSecOps Engineer / Admin Engineer | 1. User can see a dashboard of all container images and their vulnerability summary, so they can quickly identify critical risks. 2. User can filter images by severity and search by image name, so they can focus on images that need immediate attention. 3. User can see full down into a specific image to view the full list of vulnerabilities, so they can assess impact and plan remediation. 4. User can mark vulnerabilities or images as fixed or acknowledged, so they can keep track of progress. 5. User can export vulnerability reports, so they can document risk posture for audits and compliance. |

**Functional Requirements**

|  |  |
| --- | --- |
| Features | Requirement |
| Dashboard Overview | * Display a sortable, scrollable table of all container images from connected registries. * Show columns: Image Name, Tag, Critical, High, Medium, Low vulnerability counts. * Include: Last Scanned timestamp, Fix Status (Open, Fixed, Acknowledged). * Default sorting: by descending number of Critical vulnerabilities. |
| Filtering & Search | * Enable full-text search by image name and tag. * Allow filtering by Severity (Critical, High, etc.), Fix Status, and Scan Date. |
| Image Details Drilldown | * Each image row is clickable and opens a detail view. * Show list of vulnerabilities per image: CVE ID, Affected Package, Severity, Description. * Include: Fix version (if available), Remediation steps/documentation links. * Allow marking individual vulnerabilities or full image as Fixed or Acknowledged. * Allow adding optional notes to each vulnerability or image. |
| User Management & Access | * Allow export of filtered data (dashboard or detail view) as CSV or PDF. |
| Bulk Actions | * Implement Role-Based Access Control (RBAC). * Restrict actions (mark fixed, export, etc.) based on role (Admin, Viewer, Editor). |
| Registry Integration | * Connect to container registries (ECR, GCR, DockerHub). * Use credentials or service accounts to authenticate and fetch image lists. * Support both periodic (scheduled) and manual scans of images. |

**User Experience**

**Entry Point & First-Time User Experience**

* Users access the dashboard via secure login.
* On first use, users are prompted to connect their container registry (guided setup with authentication walkthrough).
* A short onboarding outline explains dashboard features (optional, dismissible).

**Core Experience**

* **Step 1:** Landing on the dashboard.
  + A responsive table lists all discovered images, sorted by highest number of critical vulnerabilities.
  + Red and orange badges visually flag images with critical/high risks.
* **Step 2:** Search and filtering.
  + User enters a search term (e.g., "api-svc") or selects a severity filter (e.g., "Critical").
  + Table instantly updates to reflect search/filter.
* **Step 3:** Drilldown into image details.
  + Clicking on an image row opens a detailed modal/page.
  + Shows vulnerability details in a tabular format, highlighting critical issues and fix availability.
  + User marks items as fixed/acknowledged and can add comments.
* **Step 4:** Bulk management.
  + Multi-select checkboxes enable users to apply actions to multiple images (e.g., mark as fixed).
* **Step 5:** Exporting data.
  + "Export" button available on main and detail screens to download filtered/current view as CSV or PDF.
* **Step 6:** Track remediation.
  + Fix/Acknowledgement status updates are logged for audit and compliance.

**Advanced Features & Edge Cases**

* Dashboard includes loading/error states for slow scans or integration failures.
* Notification banners highlight scan errors, registry connection problems, or scan completion alerts.
* Support for “no images found” and “no critical vulnerabilities” states, reassuring users when risk is low.

**UI/UX Highlights**

* High-contrast severity badges (Critical = Red, High = Orange, etc.) for instant visual cues.
* Responsive layout for use across desktops and 13-inch laptops.
* Accessible search field with keyboard shortcuts.
* Consistent table actions aligned to the right (View, Export).
* Tooltips and inline help for scan results and remediation advice.
* Pagination for large image sets (key for scale).

**Success Metrics**

**User-Centric Metrics**

* Percentage of container images scanned per week
* Reduction in mean time to detect and resolve critical vulnerabilities
* User engagement (dashboard sessions per user, search/filter/drilldown usage)
* User satisfaction/NPS after 30 and 90 days

**Business Metrics**

* Adoption rate (# of organizations actively using the dashboard)
* Customer retention and upsell/conversion rates
* Number of critical vulnerabilities detected and marked as resolved

**Technical Metrics**

* Scan completion time (average )
* Dashboard uptime and availability
* Scan and data integration error rates

**Tracking Plan**

* User logins and session counts
* Image scan triggers (manual and scheduled)
* Table search/filter actions
* Detail view clicks
* Bulk action and export events
* Fix/acknowledgment status updates

**Technical Considerations**

**Technical Needs**

* Integration APIs for connecting to third-party registries (authentication, fetch images, trigger scan).
* Image scanning pipeline—retrieves, scans, and analyzes container images for vulnerabilities asynchronously.
* Front-end web dashboard with dynamic data tables, search, filtering, and modal/detail components.
* Back-end services for user authentication, data storage, and reporting.

**Integration Points**

* AWS Elastic Container Registry, Google Container Registry, DockerHub, and other major registry APIs.
* Option for future integrations with CI/CD and alerting systems.

**Data Storage & Privacy**

* Store metadata about images and scan results, not full image contents.
* Secure handling of registry credentials and user accounts.
* Compliance with SOC2, GDPR, or other applicable standards.
* Data retention policy for historical scans and audit trails.

**Scalability & Performance**

* Large enterprise scalability: designed to handle 10,000+ images per account, updated nightly or on demand.
* Scanning optimized for parallelization and queueing to prevent backlogs.

**Potential Challenges**

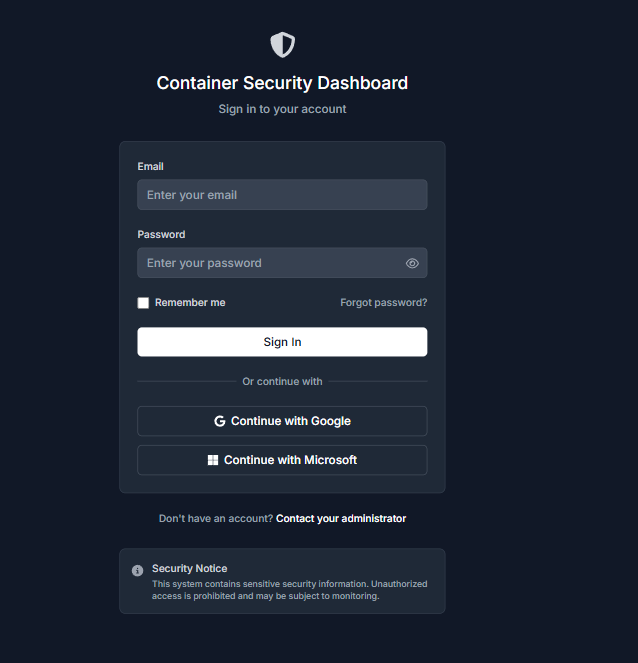
* Ensuring registry credentials are securely stored and rotated.
* Handling large repositories without performance bottlenecks.
* Surface integration or scan errors with actionable details.

**Action Items for Development Team**

* Define API and data model for image and vulnerability entities.
* Implement secure registry authentication workflow.
* Build initial dashboard UI with responsive, filterable table.
* Develop asynchronous image scanning pipeline and connect results to dashboard.
* Create image details view and add fix/acknowledge features.
* Set up bulk action and export endpoints.
* Implement basic analytics for user actions and scan completion.
* Test with a set of large image repositories and iterate on UI/UX.
* Conduct security review and user feedback loop before general availability.

**Wireframes**

Login Page UI



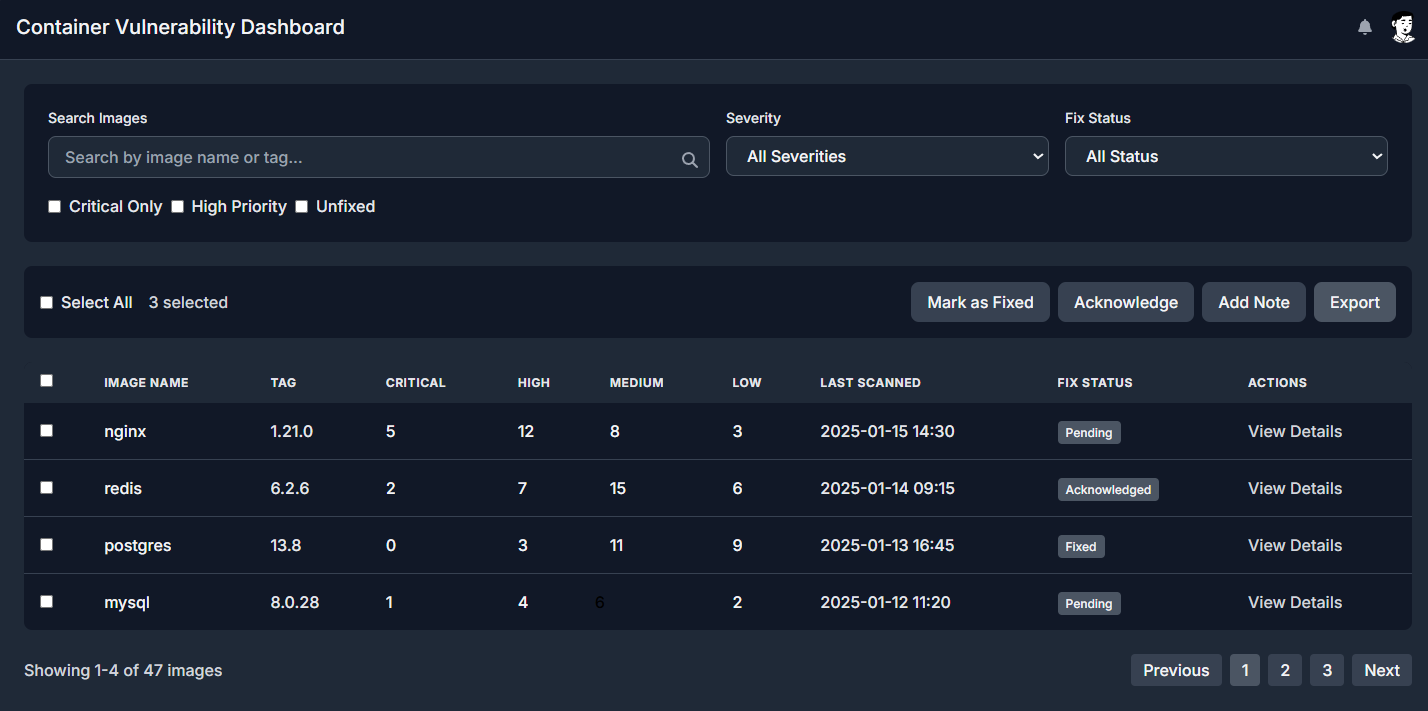
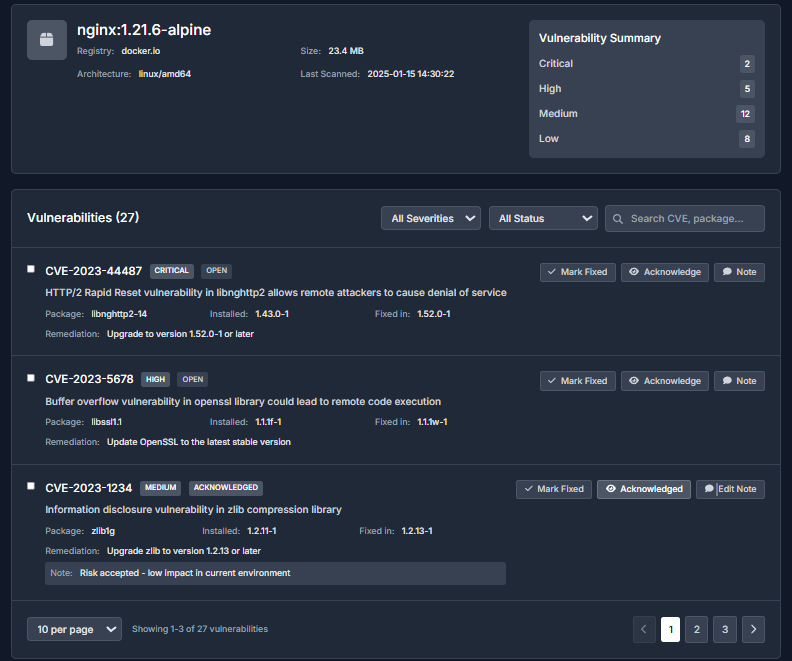
Dashboard   


Image Level View



**User Flow for Using the Application**

This flow represents how security engineers interact with the dashboard to triage, filter, and manage container image vulnerabilities.

