Trivy Offline Vulnerability Scanning – Setup & Workflow

Setup Overview

1. **Online Build Server (Ubuntu/MacBook/Windows)**
- Runs a script that builds the **latest Trivy offline DB bundle** (`trivy-offline.db.tgz`).
- The DB includes both **OS vulnerabilities** and **language package advisories**.
- Output bundle contains: `trivy.db` and `metadata.json`.

2. **Air-gapped RHEL Server**
- Receives the offline DB tarball via Ansible automation.
- DB is unpacked into `~/.cache/trivy/db/`.
- Local images are built/stored with Podman and scanned using the offline DB.

3. **Bulk Scanning Script**
- Discovers all Podman images on the server.
- Runs Trivy in **offline archive mode** (reliable even without Podman socket).
- Generates JSON reports into timestamped folders.

Dashboard & Reporting

- Reports can be uploaded to the **Trivy Dashboard HTML viewer**.
- Provides:
- Weekly trend of vulnerabilities across all images.
- Per-image severity breakdown (Critical, High, Medium, Low, Unknown).
- Drill-down vulnerabilities viewer with filters (CVE, package, fix availability).
- Export options available: `weekly_summary.csv`, `latest_week_per_image.csv`, and `vulnerabilities_filtered.csv`.

Functionality Highlights

- **Offline compatible** → no internet required on RHEL servers.
- **Configurable severity** → supports all severities, or limit (e.g., HIGH,CRITICAL).
- **Supports both OS & language packages** (or OS-only).
- **Automated workflow** with Ansible for syncing DB and triggering scans.
- **Scalable** → works for any number of Podman images.

Next Steps

- Use the automation playbook to regularly update the DB and run scans.
- Review results in the HTML dashboard or CSV exports.
- Begin remediation by prioritizing **Critical/High vulnerabilities**.
- Optionally integrate into CI/CD or centralize reports across multiple servers.

Architecture Diagram