CVE Collector Core

Dies ist die Kernanwendung zur automatisierten Sammlung und Speicherung von Sicherheitsinformationen aus verschiedenen Quellen.

Architektur

Die Anwendung besteht aus zwei Docker-Containern:

1. **collector**: Ein Python-Dienst, der periodisch eine Liste von RSS-Feeds abruft und neue Einträge in einer Datenbank speichert.
2. **db**: Ein PostgreSQL-Datenbankserver, der die gesammelten Daten persistent vorhält.

Die gesamte Anwendung wird über docker-compose.yml verwaltet.

Setup

1. **Voraussetzungen**: Docker und Docker Compose müssen installiert sein.
2. **Konfiguration erstellen**: Kopieren Sie die Vorlagedatei für die Umgebungsvariablen.
   cp .env.example .env

3. **Konfiguration anpassen**: Öffnen Sie die .env-Datei und passen Sie die Werte an. Ändern Sie **unbedingt** das POSTGRES_PASSWORD zu einem sicheren Passwort. Sie können hier auch das POLL_INTERVAL und die RSS_FEEDS nach Ihren Wünschen anpassen.
4. **Anwendung starten**: Führen Sie den folgenden Befehl im Hauptverzeichnis des Projekts aus:
   docker-compose up --build -d

   - --build erstellt die Docker-Images beim ersten Start.
   - -d startet die Container im Hintergrund (detached mode).
5. **Logs überprüfen**: Sie können die Logs des Collectors einsehen, um sicherzustellen, dass alles korrekt funktioniert:
   docker-compose logs -f collector

Wie es funktioniert

- Der collector-Dienst startet und verbindet sich mit der db-Datenbank.
- Er stellt sicher, dass eine Tabelle namens vulnerabilities existiert.
- In einer Endlosschleife durchläuft er alle konfigurierten RSS-Feeds.
- Für jeden neuen Eintrag in einem Feed wird eine Zeile in der Datenbank angelegt. Ein UNIQUE-Constraint auf der entry_id verhindert Duplikate.
- Neben den extrahierten Feldern (Titel, Link etc.) wird der komplette Original-Eintrag als JSONB in der Spalte raw_data gespeichert. Dies ist extrem wertvoll, da so keine Informationen verloren gehen und zukünftige Auswertungen auf den Rohdaten

aufbauen können.
- Nach jedem Durchlauf pausiert der Dienst für die im POLL_INTERVAL festgelegte Zeit.

Nächste Schritte (Ausbaustufen)

Diese robuste Basis ist der perfekte Ausgangspunkt für weitere Automatisierungen:

1. **Daten anreichern**: Erstellen Sie einen weiteren Service, der neue Einträge aus der DB liest, mithilfe von regulären Ausdrücken nach CVE-Nummern sucht und den CVSS-Score von einer API (z.B. NVD) abfragt und in die Datenbank zurückschreibt.
2. **KI-Zusammenfassung**: Ein Service könnte täglich die Einträge der letzten 24 Stunden aus der DB holen, sie an eine LLM-API senden und eine Management-taugliche Zusammenfassung erstellen, die dann per E-Mail oder in einem Chat-Tool gepostet wird.
3. **Vektorisierung & Semantische Suche**: Für jeden neuen Eintrag können Embeddings (Vektoren) erstellt und in einer Vektordatenbank wie Qdrant (aus Ihrer ursprünglichen Idee) gespeichert werden. Dies ermöglicht es, nach inhaltlich ähnlichen Schwachstellen zu suchen, auch wenn die Schlüsselwörter nicht übereinstimmen.
4. **Alarmierung**: Ein "Watcher"-Service könnte die Datenbank auf Einträge mit hohen CVSS-Scores oder bestimmten Schlüsselwörtern (z.B. Namen Ihrer eingesetzten Software) überwachen und bei einem Treffer sofort eine Alarmierung via Slack, Matrix oder E-Mail auslösen.

Sie haben jetzt ein solides Fundament, das die Daten zuverlässig sammelt und strukturiert ablegt. Alle weiteren intelligenten Funktionen können nun auf diese saubere Datenquelle zugreifen, ohne sich selbst um die Datenbeschaffung kümmern zu müssen.