CVE Management System Documentation

Overview

The CVE Management System is a web-based application designed to fetch, store, and display CVE (Common Vulnerabilities and Exposures) data from the National Vulnerability Database (NVD). This system allows users to query CVE data via API endpoints and view CVE details through a user-friendly web interface. The system fetches CVE data periodically, stores it in a MySQL database, and provides sorting, filtering, and pagination features for CVE data.

Requirements

- Python 3.x
- Flask
- Requests
- MySQL Server
- MySQL Python Connector (`mysql-connector-python`)
- Jinja2 templates (for rendering HTML)

Setup Instructions

1. Install Dependencies

Start by installing the required Python libraries:

```
bash
CopyEdit
pip install flask requests mysql-connector-python
```

2. Set Up MySQL Database

Ensure that MySQL server is installed on your system. Then, create the database `cve_database` by running the following SQL command:

```
sql
CopyEdit
CREATE DATABASE cve_database;
```

3. Configure MySQL Connection

In the application's configuration file, ensure that the MySQL connection details are set correctly:

- **host**: `localhost` (or your database host)
- **user**: `root` (or your database user)
- **password**: `xxxxx` (your database password)
- **database**: `cve_database`

4. Create the Table

The `cve_details` table will be automatically created when the application is run for the first time. It stores the CVE data fetched from the NVD API.

5. Run the Application

To start the application, run the following command:

```
bash
CopyEdit
python app.py
```

By default, the application will be accessible at `http://localhost:5000`.

6. Fetching CVE Data

The application periodically fetches CVE data from the NVD API and stores it in the MySQL database. The fetching process runs in a separate background thread through the `sync_cves` function, which periodically calls the NVD API to fetch new data.

7. Web Interface

The web interface provides two main views:

- **Home Page**: The landing page that redirects to the CVE list.
- **CVE List**: Displays a table of CVEs that can be sorted and paginated.
- **CVE Details**: Displays detailed information about a specific CVE when clicked from the CVE list.

8. API Endpoints

The system exposes the following API endpoints:

- **GET `/api/cves/year/<int:year>`**
  Fetches all CVEs for a given year.
  Example: `/api/cves/year/2022`
- **GET `/api/cves/score/<float:score>`**
  Fetches all CVEs with a CVSS score greater than or equal to the specified score.
  Example: `/api/cves/score/7.5`
- **GET `/api/cves/modified/<int:days>`**
  Fetches all CVEs modified in the last N days.
  Example: `/api/cves/modified/30`

9. MySQL Schema

The database table `cve_details` is created with the following schema:

```
sql
CopyEdit
CREATE TABLE IF NOT EXISTS cve_details (
    cve_id VARCHAR(255) PRIMARY KEY,
    description TEXT NOT NULL,
```

```
    published_date DATETIME NOT NULL,
    modified_date DATETIME NOT NULL,
    cvss_score FLOAT NOT NULL,
    cvss_v2_score FLOAT,
    year INT NOT NULL,
    status VARCHAR(50) NOT NULL DEFAULT 'new'
);
```

10. Data Flow

- **Data Fetching**: The `fetch_cves` function fetches raw CVE data from the NVD API.
- **Data Cleaning**: The `clean_data` function processes the fetched data, extracting relevant information.
- **Data Insertion**: The `store_cve` function stores the cleaned data into the MySQL database. If the CVE already exists, it is updated.
- **Periodic Fetching**: The `sync_cves` function runs periodically to fetch and update CVE data.

11. Error Handling

- If any CVE entry contains missing or malformed data, it is skipped.
- If the CVE ID already exists in the database, the record is updated with the latest data.
- API calls to the NVD API are retried up to 5 times in case of an error.

12. Example Usage

- **Home Page**: `http://localhost:5000/` - Displays a list of CVEs.
- **CVE List**: `http://localhost:5000/cves/list?page=1&resultsPerPage=10&sort=published_date&direction=ASC` Displays a paginated list of CVEs sorted by the specified field (e.g., `published_date`).
- **CVE Details**: `http://localhost:5000/cves/CVE-2022-1234` Displays detailed information about a specific CVE.
- **API Endpoint**: `http://localhost:5000/api/cves/year/2022` Fetches all CVEs from the year 2022 in JSON format.

13. Enhancements and Future Work

- **Authentication**: Add authentication to restrict access to CVE data for certain users.
- **Search**: Implement search functionality based on CVE ID or description.
- **Advanced Filtering**: Provide additional filtering options for CVE attributes (e.g., severity, status).
- **Web Scraping**: In case the NVD API becomes unavailable, a fallback web scraping mechanism could be added to fetch data directly from the NVD website.