

CVE Project Documentation

1. Overall Scenario

This project aims to create a system for fetching, storing, and accessing Common Vulnerabilities and Exposures (CVE) data from the National Vulnerability Database (NVD) API. It consists of the following components:

- **Data Fetching:** Fetches CVE data from the NVD API and stores it in a MySQL database.
- **Database Management:** Sets up and manages the MySQL database, including data cleaning and validation.
- **API:** Provides API endpoints to access CVE data based on various criteria (CVE ID, year, score, modified date).
- **Web UI:** A basic web interface to display and search CVE data.
- **Unit Tests:** Tests to ensure the functionality of the application.
- **Centralized Execution:** A main.py script to orchestrate the execution of all components.

2. Project Structure

cve_project/

```
|— app2.py      # Flask application for the web UI
|— api_fetch.py # Flask application for the API endpoints
|— config.py   # Configuration settings
|— database.py # Database connection and setup
|— fetch_CVE_data.py # Fetches and stores CVE data
|— filter.py   # Flask application for the API endpoints for web UI
|— full_sync.py # Refreshes all CVE data.
|— main.py     # Central execution script
|— requirements.txt # Python dependencies
|— tests/      # Unit tests
|   |— unit_tests.py
|— templates/  # HTML templates for the web UI
|   |— cve_detail.html
|   |— cve_list.html
|       |— cve_list.html
|— mysql_code.sql # Mysql database creation and modification script.
```

3. File Explanations

3.1. config.py

- Purpose: Contains configuration settings for the entire project.
- Content:
 - DB_HOST, DB_USER, DB_PASSWORD, DB_NAME: Database connection details.
 - API_URL: NVD API endpoint URL.
 - RESULTS_PER_PAGE: Number of CVEs to fetch per API request.
 - LOG_LEVEL: Logging level (e.g., INFO, DEBUG).
 - LOG_FILE: Log file path.
- Functionality: Provides a central location for storing and accessing configuration parameters.

3.2. database/cve_database.sql

- Purpose: Creates and modifies the MySQL database schema and data.
- Content:
 - CREATE DATABASE cve_database;; Creates the database.
 - USE cve_database;; Selects the database.
 - CREATE TABLE cve_data (...): Creates the cve_data table.
 - SHOW TABLES;, DESC cve_data;, SELECT ...: Queries and displays table information.
 - TRUNCATE TABLE cve_data;; Clears the table.
 - SET SQL_SAFE_UPDATES = 0;; Disables safe update mode.
 - Duplicate removal, null value handling, CVE ID validation, description trimming, base score validation, date validation, future CVE ID validation.
 - SET SQL_SAFE_UPDATES = 1;; Re-enables safe update mode.
- Functionality: Sets up the database and performs data cleaning and validation.

3.3. database.py

- Purpose: Establishes and manages the database connection.
- Content:
 - get_db_connection(): Creates and returns a MySQL database connection using mysql.connector.connect().
 - Handles mysql.connector.Error exceptions.
- **Functionality: Provides a reusable function for database connection.**

3.4. fetch_CVE_data.py

- Purpose: Fetches CVE data from the NVD API and stores it in the MySQL database.

- Content:
 - `get_db_connection()`: Establishes a database connection.
 - `fetch_cve_data(start_index)`: Fetches CVE data from the API using `requests.get()`.
 - `insert_cve_data(cve_list, cursor)`: Inserts CVE data into the database.
 - `main()`: Orchestrates the fetching and insertion process with pagination and incremental commits.
- Functionality: Retrieves CVE data from the NVD API and populates the database.

3.5 filter.py

- Purpose: Provides API endpoints for accessing CVE data with web UI.
- Content:
 - `get_db_connection()`: Establishes a database connection.
 - `@app.route('/api/cve/id', methods=['GET'])`: Retrieves CVE details by CVE ID.
 - `@app.route('/api/cve/year', methods=['GET'])`: Retrieves CVE details by year.
 - `@app.route('/api/cve/score', methods=['GET'])`: Retrieves CVE details by score.
 - `@app.route('/api/cve/modified', methods=['GET'])`: Retrieves CVE details modified in the last N days.
- Functionality: Exposes API endpoints for CVE data retrieval in the web UI format.

3.5. full_sync.py

- Purpose: Clears the existing CVE data and re-fetches all data from the NVD API.
- Content:
 - `get_db_connection()`: Establishes a database connection.
 - `fetch_cve_data(start_index)`: Fetches CVE data from the API.
 - `insert_cve_data(cve_list)`: Inserts CVE data into the database.
 - `refresh_data()`: Clears the table and re-fetches data.
 - Main execution.
- Functionality: Performs a complete data refresh.

3.6. api_fetch.py

- Purpose: Provides API endpoints for accessing CVE data.
- Content:
 - `get_db_connection()`: Establishes a database connection.
 - `@app.route('/api/cve/id', methods=['GET'])`: Retrieves CVE details by CVE ID.
 - `@app.route('/api/cve/year', methods=['GET'])`: Retrieves CVE details by year.
 - `@app.route('/api/cve/score', methods=['GET'])`: Retrieves CVE details by score.

- `@app.route('/api/cve/modified', methods=['GET'])`: Retrieves CVE details modified in the last N days.
- Functionality: Exposes API endpoints for CVE data retrieval.

3.7. app2.py

- Purpose: Provides a web UI for displaying and searching CVE data.
- Content:
 - `get_db_connection()`: Establishes a database connection.
 - `@app.route('/cves/list')`: Displays a list of CVEs with pagination and search functionality.
 - `@app.route('/cves/<cve_id>')`: Displays details of a specific CVE.
- Functionality: Provides a web interface for CVE data.

3.8. main.py

- Purpose: Acts as a central execution script.
- Content:
 - `run_script(script_path, description)`: Executes a Python script using `subprocess.run()`.
 - `main()`: Parses command-line arguments and calls the appropriate scripts.
- Functionality: Orchestrates the execution of all components.

3.9. tests/unit_tests.py

- Purpose: Contains unit tests for the Flask application.
- Content:
 - `FlaskTestCase`: A test class that inherits from `unittest.TestCase`.
 - `setUp()`: Sets up the test database and data.
 - `tearDown()`: Cleans up the test database.
 - Test methods: Tests for various API endpoints and functionalities.
- Functionality: Ensures the functionality of the application.

3.10. requirements.txt

- Purpose: Lists the Python dependencies for the project.
- Content:
 - Flask
 - mysql-connector-python
 - requests
- Functionality: Specifies the required packages.

4. Execution Flow

1. Run `python main.py --all` to execute all scripts in order.
2. `database.py` sets up the database.
3. `fetch_CVE_data.py` fetches and stores CVE data.
4. `full_sync.py` refreshes all CVE data.
5. `Filter.py` starts API server and web UI server
6. `api_fetch.py` starts the API server.
7. `app2.py` starts the web UI server.
8. `unit_tests.py` runs the unit tests.
9. Access the API endpoints and web UI through a browser or API testing tool.

5. API Endpoints

To get Web Ui the base URL: `http://127.0.0.1:5000/`

Base URL: `http://127.0.0.1:5000/api/`

- **5.1. Get All CVEs**
 - Endpoint: `/cves`
 - Method: GET
 - Description: Returns all stored CVEs.
- **5.2. Get CVE by ID**
 - Endpoint: `/cve/id`
 - Method: GET
 - Description: Returns details of a specific CVE.
 - Example Request: `http://127.0.0.1:5000/api/cve/id?cve_id=CVE-2023-1234` (Replace CVE-2023-1234 with an actual CVE ID from your database.)
- **5.3. Get CVEs by Year**
 - Endpoint: `/cve/year`
 - Method: GET
 - Description: Returns CVEs from a specific year.
 - Example Request: `http://127.0.0.1:5000/api/cve/year?year=2023` (Replace 2023 with a year from your database.)
- **5.4. Get CVEs by Score**
 - Endpoint: `/cve/score`
 - Method: GET

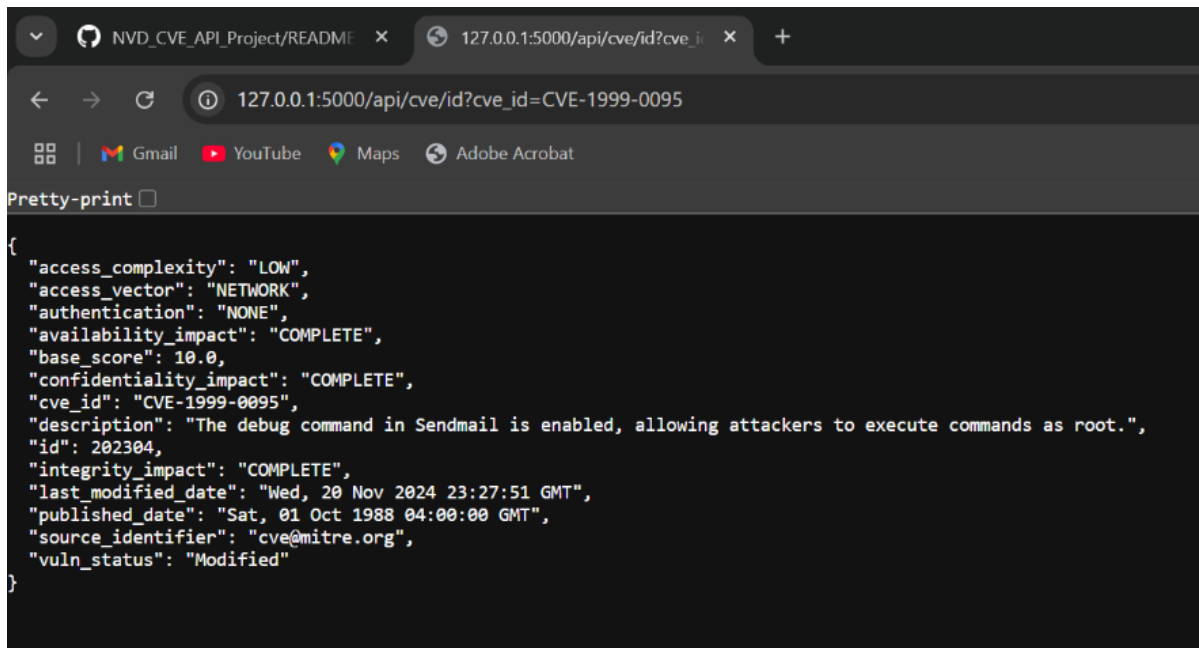
- Description: Filters CVEs with a CVSS base score above a threshold.
- Example Request: `http://127.0.0.1:5000/api/cve/score?score=7.0` (Replace 7.0 with a score from your database.)
- **5.5. Get Recently Modified CVEs**
 - Endpoint: `/cve/modified`
 - Method: GET
 - Description: Fetches CVEs modified in the last N days.
 - Example Request: `http://127.0.0.1:5000/api/cve/modified?days=30` (This will return CVEs modified in the last 30 days.)

6. Web User Interface (UI)

Base URL: `http://127.0.0.1:5000/cves/list?page=1&resultsPerPage=10`

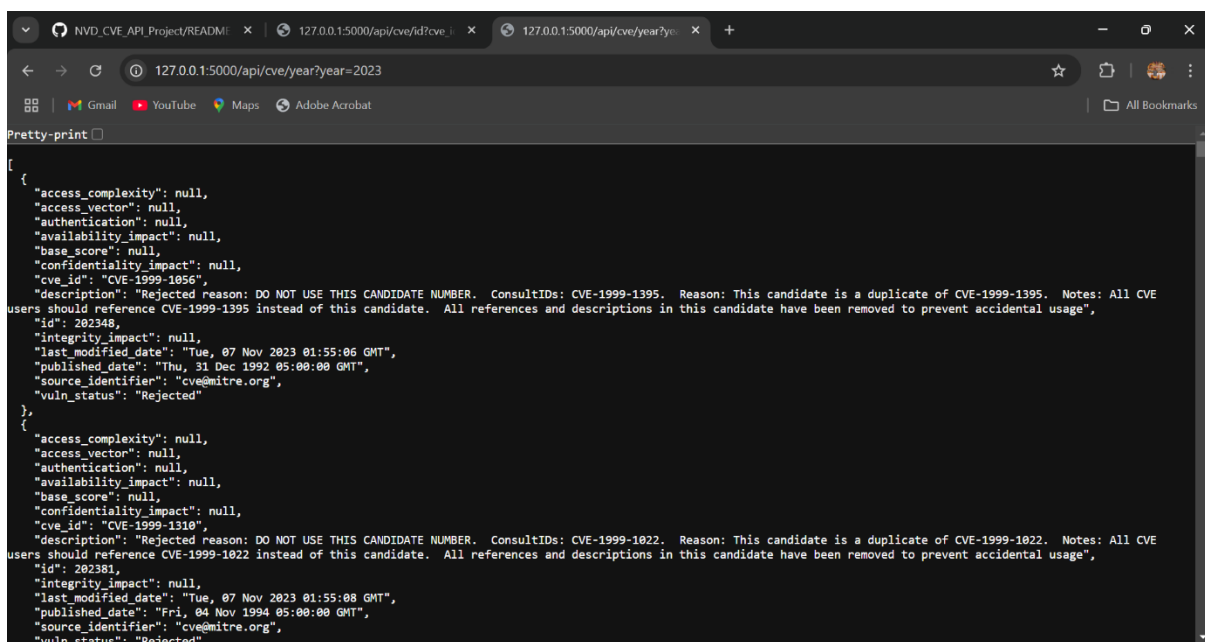
- **6.1. CVE List Page**
 - Endpoint: `/cves/list`
 - Description: Displays a paginated list of CVEs.
 - Parameters:
 - `page` (optional, integer): The page number to display. Default is 1.
 - `resultsPerPage` (optional, integer): The number of CVEs to display per page. Default is determined by the application.
 - Example Request: `http://127.0.0.1:5000/cves/list?page=1&resultsPerPage=10`
 - Functionality:
 - Presents CVE data in a user-friendly table format.
 - Allows users to navigate through pages of CVEs.
 - Provides search and filtering capabilities (if implemented).
 - Displays CVE details when a specific CVE is clicked.

7. Results



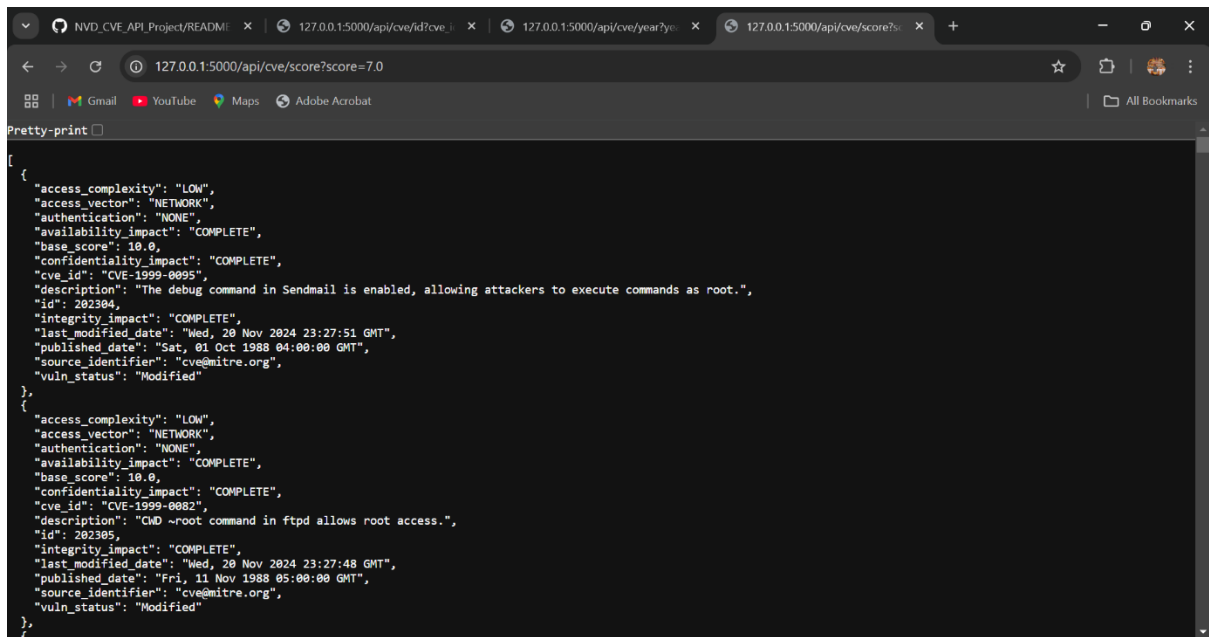
```
{
  "access_complexity": "LOW",
  "access_vector": "NETWORK",
  "authentication": "NONE",
  "availability_impact": "COMPLETE",
  "base_score": 10.0,
  "confidentiality_impact": "COMPLETE",
  "cve_id": "CVE-1999-0095",
  "description": "The debug command in Sendmail is enabled, allowing attackers to execute commands as root.",
  "id": 202304,
  "integrity_impact": "COMPLETE",
  "last_modified_date": "Wed, 20 Nov 2024 23:27:51 GMT",
  "published_date": "Sat, 01 Oct 1988 04:00:00 GMT",
  "source_identifier": "cve@mitre.org",
  "vuln_status": "Modified"
}
```

Figure 1: API filtering based on CVE_ID



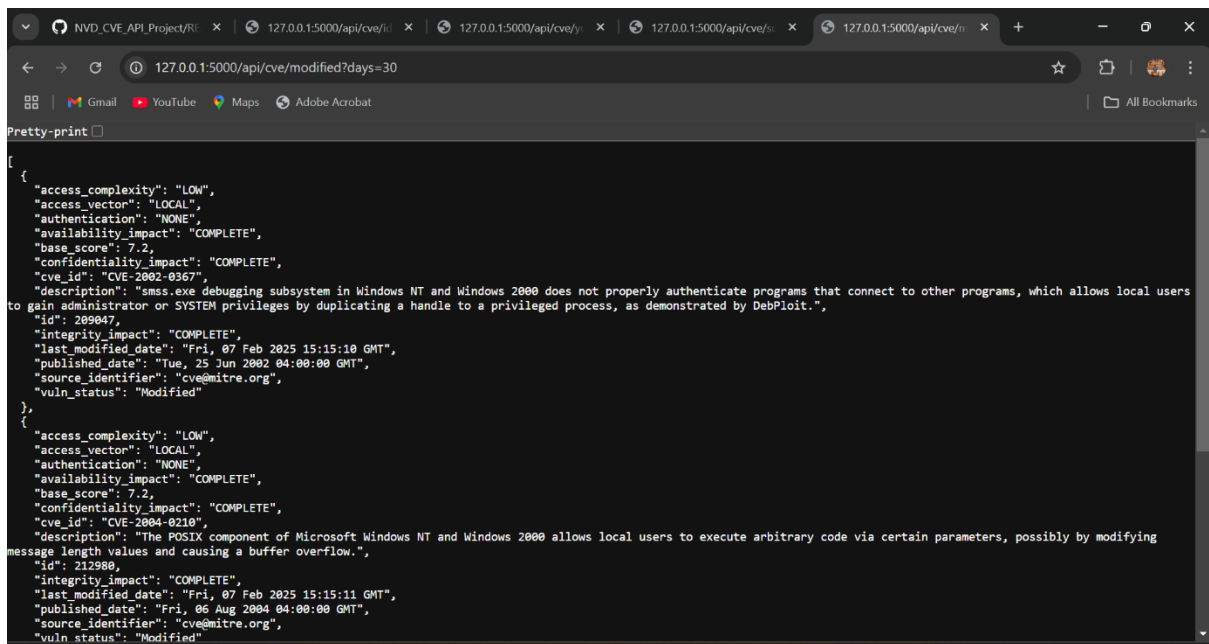
```
[
  {
    "access_complexity": null,
    "access_vector": null,
    "authentication": null,
    "availability_impact": null,
    "base_score": null,
    "confidentiality_impact": null,
    "cve_id": "CVE-1999-1056",
    "description": "Rejected reason: DO NOT USE THIS CANDIDATE NUMBER. ConsultIDs: CVE-1999-1395. Reason: This candidate is a duplicate of CVE-1999-1395. Notes: All CVE users should reference CVE-1999-1395 instead of this candidate. All references and descriptions in this candidate have been removed to prevent accidental usage",
    "id": 202348,
    "integrity_impact": null,
    "last_modified_date": "Tue, 07 Nov 2023 01:55:06 GMT",
    "published_date": "Thu, 31 Dec 1992 05:00:00 GMT",
    "source_identifier": "cve@mitre.org",
    "vuln_status": "Rejected"
  },
  {
    "access_complexity": null,
    "access_vector": null,
    "authentication": null,
    "availability_impact": null,
    "base_score": null,
    "confidentiality_impact": null,
    "cve_id": "CVE-1999-1310",
    "description": "Rejected reason: DO NOT USE THIS CANDIDATE NUMBER. ConsultIDs: CVE-1999-1022. Reason: This candidate is a duplicate of CVE-1999-1022. Notes: All CVE users should reference CVE-1999-1022 instead of this candidate. All references and descriptions in this candidate have been removed to prevent accidental usage",
    "id": 202381,
    "integrity_impact": null,
    "last_modified_date": "Tue, 07 Nov 2023 01:55:06 GMT",
    "published_date": "Fri, 04 Nov 1994 05:00:00 GMT",
    "source_identifier": "cve@mitre.org",
    "vuln_status": "Rejected"
  }
]
```

Figure 2: API Filtering based of year



```
127.0.0.1:5000/api/cve/score?score=7.0
[
  {
    "access_complexity": "LOW",
    "access_vector": "NETWORK",
    "authentication": "NONE",
    "availability_impact": "COMPLETE",
    "base_score": 10.0,
    "confidentiality_impact": "COMPLETE",
    "cve_id": "CVE-1999-0095",
    "description": "The debug command in Sendmail is enabled, allowing attackers to execute commands as root.",
    "id": 202304,
    "integrity_impact": "COMPLETE",
    "last_modified_date": "Wed, 20 Nov 2024 23:27:51 GMT",
    "published_date": "Sat, 01 Oct 1988 04:00:00 GMT",
    "source_identifier": "cve@mitre.org",
    "vuln_status": "Modified"
  },
  {
    "access_complexity": "LOW",
    "access_vector": "NETWORK",
    "authentication": "NONE",
    "availability_impact": "COMPLETE",
    "base_score": 10.0,
    "confidentiality_impact": "COMPLETE",
    "cve_id": "CVE-1999-0082",
    "description": "CMD ~root command in ftpd allows root access.",
    "id": 202305,
    "integrity_impact": "COMPLETE",
    "last_modified_date": "Wed, 20 Nov 2024 23:27:48 GMT",
    "published_date": "Fri, 11 Nov 1988 05:00:00 GMT",
    "source_identifier": "cve@mitre.org",
    "vuln_status": "Modified"
  }
]
```

Figure 3: API filtering based on score



```
127.0.0.1:5000/api/cve/modified?days=30
[
  {
    "access_complexity": "LOW",
    "access_vector": "LOCAL",
    "authentication": "NONE",
    "availability_impact": "COMPLETE",
    "base_score": 7.2,
    "confidentiality_impact": "COMPLETE",
    "cve_id": "CVE-2002-0367",
    "description": "smss.exe debugging subsystem in Windows NT and Windows 2000 does not properly authenticate programs that connect to other programs, which allows local users to gain administrator or SYSTEM privileges by duplicating a handle to a privileged process, as demonstrated by DebPloit.",
    "id": 209047,
    "integrity_impact": "COMPLETE",
    "last_modified_date": "Fri, 07 Feb 2025 15:15:10 GMT",
    "published_date": "Tue, 25 Jun 2002 04:00:00 GMT",
    "source_identifier": "cve@mitre.org",
    "vuln_status": "Modified"
  },
  {
    "access_complexity": "LOW",
    "access_vector": "LOCAL",
    "authentication": "NONE",
    "availability_impact": "COMPLETE",
    "base_score": 7.2,
    "confidentiality_impact": "COMPLETE",
    "cve_id": "CVE-2004-0210",
    "description": "The POSIX component of Microsoft Windows NT and Windows 2000 allows local users to execute arbitrary code via certain parameters, possibly by modifying message length values and causing a buffer overflow.",
    "id": 212900,
    "integrity_impact": "COMPLETE",
    "last_modified_date": "Fri, 07 Feb 2025 15:15:11 GMT",
    "published_date": "Fri, 06 Aug 2004 04:00:00 GMT",
    "source_identifier": "cve@mitre.org",
    "vuln_status": "Modified"
  }
]
```

Figure 4: API Filtering based of N number of days modifications made

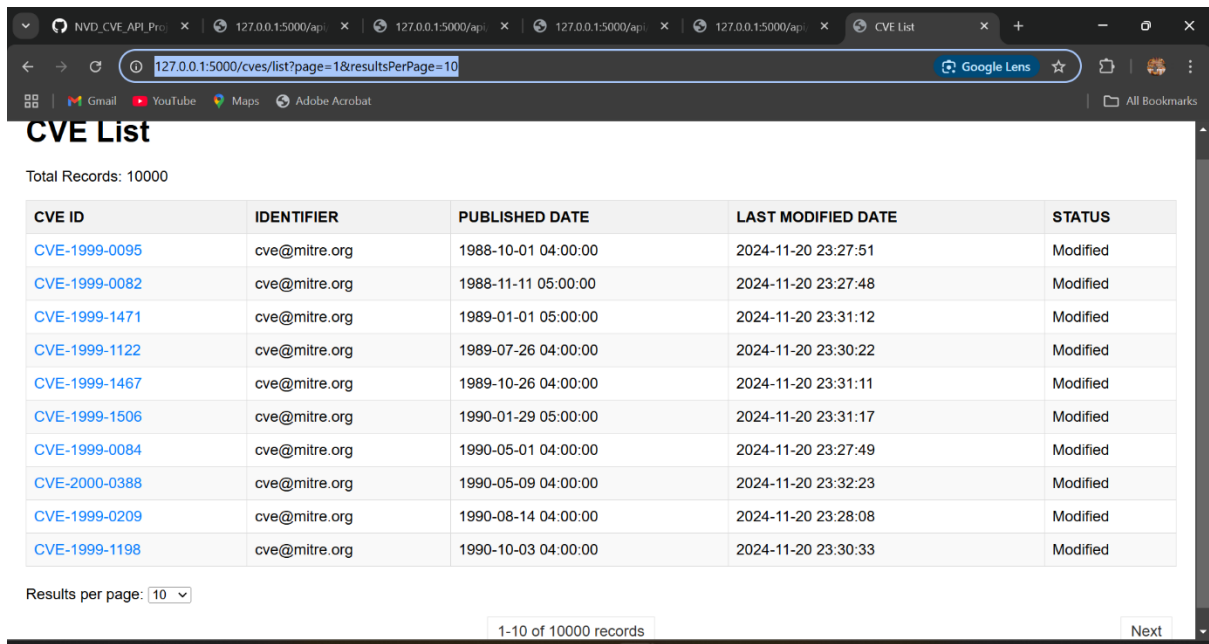


Figure 5: UI interface

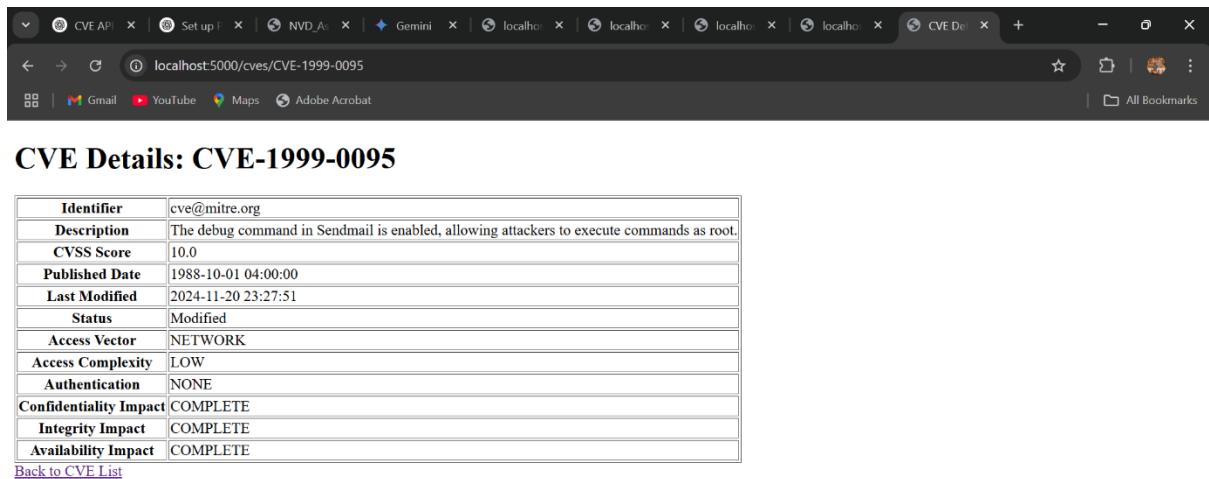


Figure 6: UI interface details of CVE

```
Command Prompt
AssertionError: 404 != 200
=====
FAIL: test_multiply (tests.unit_tests.FlaskAppTestCase.test_multiply)
=====
Traceback (most recent call last):
  File "C:\Users\gnani\OneDrive\Desktop\NVD_CVE_API_Project\tests\unit_tests.py", line 29, in test_multiply
    self.assertEqual(response.status_code, 200) # Check status code
    ~~~~~^~~~~~
AssertionError: 404 != 200
=====

Ran 4 tests in 0.026s

FAILED (failures=3)

C:\Users\gnani\OneDrive\Desktop\NVD_CVE_API_Project>python -m unittest tests.unit_tests
.....
Ran 5 tests in 0.290s

OK

C:\Users\gnani\OneDrive\Desktop\NVD_CVE_API_Project>
```

Figure 7: Output after executing unit tests