CyberCube Vulnerabilities and CVE Changes API Documentation

Overview

This API provides endpoints to fetch data from the vulnerabilities and cve_changes tables in the cybercube database. The API is built using Flask and MySQL, and it supports querying based on cve_id and product_id.

Prerequisites

- Python 3.x
- Flask
- MySQL database

Installation

1. Clone the repository or download the code.
2. Install the required packages using pip:

   Bash - pip install flask mysql-connector-python

3. Ensure you have a MySQL database running and the cybercube database with the required tables (vulnerabilities and cve_changes).

Configuration

Update the MySQL database connection parameters in the code as needed:

```
db_config = {
    'user': 'root',
    'password': 'admin',
    'host': 'localhost',
    'database': 'cybercube'
}
```

Running the API

To run the Flask application, execute the following command:

Bash - python app.py

The API will be available at http://127.0.0.1:5000.

API Endpoints

1. Get Vulnerabilities

URL

GET /vulnerabilities

Parameters

- cve_id (optional): Filter by specific CVE ID.
- product_id (optional): Filter by specific Product ID.

Example Requests

- Get all vulnerabilities:

  GET http://127.0.0.1:5000/vulnerabilities

- Get vulnerabilities by CVE ID:

  GET http://127.0.0.1:5000/vulnerabilities?cve_id=CVE-2021-12345

- Get vulnerabilities by Product ID:

  GET http://127.0.0.1:5000/vulnerabilities?product_id=Product-123

- Get vulnerabilities by both CVE ID and Product ID:

  GET http://127.0.0.1:5000/vulnerabilities?cve_id=CVE-2021-12345&product_id=Product-123

Response

The response is a JSON array of vulnerabilities. Each vulnerability contains various fields such as CVE ID, source identifier, published date, last modified date, etc.

Example Response


```
[
 {
  "CVE_ID": "CVE-2021-12345",
  "SOURCE_IDENTIFIER_CODE": 1,
  "PUBLISHED_DATE": "2021-05-15T00:00:00",
  "LAST_MODIFIED_DATE": "2021-05-20T00:00:00",
  "VULNERABILITY_STATUS_CODE": 2,
  "DESCRIPTION_EN": "Sample vulnerability description.",
  "CVSS_BASE_SCORE": 7.5,
  "ACCESS_VECTOR_CODE": 3,
```

```
   "CONFIDENTIALITY_IMPACT_CODE": 2,
   "INTEGRITY_IMPACT_CODE": 2,
   "AVAILABILITY_IMPACT_CODE": 2,
   "BASE_SEVERITY_CODE": 4,
   "EXPLOITABILITY_SCORE": 8.6,
   "IMPACT_SCORE": 6.4,
   "WEAKNESS_DESCRIPTION_CODE": 1,
   "CONFIGURATION_CRITERIA": "Sample criteria",
   "PRODUCT": "Sample Product",
   "PUBLISHED_DATE_DATE": "2021-05-15",
   "PUBLISHED_DATE_TIME": "00:00:00",
   "LAST_MODIFIED_DATE_DATE": "2021-05-20",
   "LAST_MODIFIED_DATE_TIME": "00:00:00",
   "RECORD_ADDED_DATE": "2021-05-15"
  }
]
```


2. Get CVE Changes

URL

GET /cve_changes

Parameters

- cve_id (optional): Filter by specific CVE ID.

Example Requests

- Get all CVE changes:

  GET http://127.0.0.1:5000/cve_changes

- Get CVE changes by CVE ID:

  GET http://127.0.0.1:5000/cve_changes?cve_id=CVE-2021-12345

Response

The response is a JSON array of CVE changes. Each CVE change contains various fields such as CVE ID, event name code, CVE change ID, source identifier code, created date, etc.

Example Response


```
[
  {
   "CVE_ID": "CVE-2021-12345",
   "EVENT_NAME_CODE": 1,
   "CVE_CHANGE_ID": "Change-001",
   "SOURCE_IDENTIFIER_CODE": 2,
```

```
   "CREATED_DATE": "2021-05-15T00:00:00",
   "CREATED_DATE_DATE": "2021-05-15",
   "CREATED_DATE_TIME": "00:00:00",
   "RECORD_ADDED_DATE": "2021-05-15"
  }
]
```


Severity Distribution

URL

- `http://127.0.0.1:5000/severity_distribution`

Response The response is a JSON array with the count of vulnerabilities grouped by their severity levels.

Example Response


```
[
  {
   "BASE_SEVERITY": "High",
   "count": 123
  },
  {
   "BASE_SEVERITY": "Medium",
   "count": 456
  },
  {
   "BASE_SEVERITY": "Low",
   "count": 789
  }
]
```


Worst Products, Platforms

URL

- `http://127.0.0.1:5000/worst_products_platforms`

Response The response is a JSON array with the top 10 products or platforms with the most number of known vulnerabilities.

Example Response


```
[

  {
   "PRODUCT_ID": "Product-123",
   "count": 50
```

```
  },
  {
    "PRODUCT_ID": "Product-456",
    "count": 45
  },
  // more products/platforms
]
```

Top 10 Vulnerabilities by Impact

URL

- `http://127.0.0.1:5000/top_vulnerabilities_impact`

Response The response is a JSON array with the top 10 vulnerabilities based on their impact score.

Example Response

```
[
  {
    "CVE_ID": "CVE-2021-12345",
    "IMPACT_SCORE": 9.8
  },
  {
    "CVE_ID": "CVE-2021-23456",
    "IMPACT_SCORE": 9.5
  },
  // more vulnerabilities
]
```

6. Top 10 Vulnerabilities by Exploitability Scores

URL

- `http://127.0.0.1:5000/top_vulnerabilities_exploitability`

Response The response is a JSON array with the top 10 vulnerabilities based on their exploitability score.

Example Response

```
[
  {
    "CVE_ID": "CVE-2021-34567",
    "EXPLOITABILITY_SCORE": 10.0
  },
  {
    "CVE_ID": "CVE-2021-45678",
    "EXPLOITABILITY_SCORE": 9.9
  },
```

```
  // more vulnerabilities
]
```

7. Top 10 Attack Vectors

URL

- `http://127.0.0.1:5000/top_attack_vectors`

Response The response is a JSON array with the top 10 attack vectors used in the vulnerabilities.

Example Response

```
[
  {
    "ACCESS_VECTOR": "Network",
    "count": 200
  },
  {
    "ACCESS_VECTOR": "Local",
    "count": 150
  },
  // more attack vectors
]
```

Helper Functions

get_db_connection()

Creates and returns a MySQL database connection.

convert_to_serializable(data)

Converts non-serializable data types (like datetime.date, datetime.datetime, and datetime.timedelta) to serializable formats.

Code Explanation

- The get_db_connection function establishes a connection to the MySQL database using the provided configuration.
- The convert_to_serializable function converts non-serializable data types to string or ISO format to ensure the data can be serialized to JSON.
- The /vulnerabilities endpoint retrieves data from the vulnerabilities table based on optional cve_id and product_id query parameters.
- The /cve_changes endpoint retrieves data from the cve_changes table based on the optional cve_id query parameter.

- Both endpoints build the SQL query dynamically based on the provided parameters, execute the query, and return the results as JSON.

Running the Application

To run the application, execute the following command in your terminal:

Bash - `python app.py`

The application will be available at `http://127.0.0.1:5000`.

Conclusion

This API provides endpoints to fetch data from the `vulnerabilities` and `cve_changes` tables in the `cybercube` database, supporting filtering by `cve_id` and `product_id`. The API returns the data in JSON format, making it easy to integrate with other systems or applications.