**CVE DATA PROCESSING**

**Section 1:** Introduction and Problem Statement

**Introduction**

**CVE Data Processing** directs the significant need for organizations and security professionals to successfully manage and query vulnerabilities using CVE (Common Vulnerabilities and Exposures) data. By integrating real-time data fetching from the National Vulnerability Database (NVD) API and a structured database, this project confirms quick access to accurate and validated vulnerability data.

**Problem Statement**

The increasing occurrence of cyberattacks and the rapid evolution of software vulnerabilities take a robust system for tracking and exploring vulnerabilities. Current challenges include:

1. **Data Availability**: Difficulty in fetching, updating, and managing CVE data in real time.
2. **Search Restrictions**: Limited qualifications to filter vulnerabilities by severity, dates, or descriptive keywords.
3. **Data Integrity**: The need for proper validation and deduplication of fetched data.
4. **Scalability**: The ability to handle growing volumes of vulnerability data efficiently.

This project associates these gaps by providing a Python-based solution that fetches CVE data, validates inputs, stores data in a MySQL database, and offers flexible search facilities.

**Section 2:** Project Block Diagram and Data Flow

**Block diagram:**

Search queries

Insert data into database

NVD Data processing

NVD API

**Data flow:**

Process API response

Python Script

(Extract API response)

Process data using re, datetime module

Python code validation

(cve\_id, severity, date)

MySQL Database (Storage & Retrieval)

Insert data

Retrieve data

User Interface

(Search & Query)

**Section 3:** Python Packages Details

**1. requests**

* + Handles HTTP requests to the NVD API for fetching CVE data.
  + Sends parameters such as API keys and request headers to fetch JSON data.
  + Provides methods like get(), post(), and json() for working with API responses.

**2. mysql.connector**

* + Establishes a connection to the MySQL database.
  + Executes SQL queries for creating tables, inserting records, and performing searches.
  + Provides cursor objects for database operations.

**3. re**

* + Performs input validation using regular expressions.
  + Ensures the correct format for CVE IDs (e.g., CVE-YYYY-NNNNN).
  + Matches patterns in strings using re.match or re.search.

**4. datetime**

* + Validates and parses date inputs (e.g., published and modified dates).
  + Handles date comparisons for querying data within a range.
  + Provides tools like datetime.strptime for converting strings to datetime objects.

**Section 4:** Performance/Results Overview

* + The script fetches **5 CVE entries** from the NVD API within **~9-10 seconds**.
  + Inserts **5 CVE entries** into the database in **~13-15 seconds**, including validation and deduplication checks.
  + The system skips duplicate entries efficiently by querying the database before insertion.
  + Search by CVE ID, severity, Date range, keyword: Retrieves specific entries in **~0.1 seconds**.

**Section 5:** Summary and Extensions

**Summary:**  
The CVE Data Management and Search Application was successfully developed to:

* + Fetch CVE data from the NVD API.
  + Store and manage vulnerabilities in a structured MySQL database.
  + Flexible Search criteria querying with quick response time.