DEPENDENCY GUARD: A VULNERABILITY DETECTION SYSTEM USING OSV API

Alka Devidas Dhile, Renuka Anna Jadhav

JSPM's Jayawantrao Sawant Polytechnic, Pune Survey No. 58, Indrayani Nagar, Handewadi Road, Hadapsar, Satav Nagar, Hadapsar, Pune, 411028, Maharashtra , India.

Email id :- Alkadhile2406@gmail.com, Jrenuka583@gmail.com

Guide Name :- MS.P.A.Garad

HOD Name :- Ms. Z.S.Sajjade

ABSTRACT

Modern software development relies heavily on third-party dependencies, which often introduce security vulnerabilities. This paper presents Dependency Guard, a web-based vulnerability detection system that utilizes the OSV API to identify known security issues in software packages. The system allows users to input package details and retrieve real-time vulnerability data including CVE IDs, severity levels, and CVSS scores. Built using Flask, the system provides a lightweight and efficient solution for developers and students to enhance software security. Experimental results show that the system accurately detects vulnerabilities and provides timely insights.

KEYWORDS

Cybersecurity, Vulnerability Detection, Dependency Analysis, OSV API, CVE, CVSS, Flask, Web Application, Software Security

1. INTRODUCTION

In modern software development, developers widely use third-party libraries and open-source dependencies to build applications efficiently. While these components save time and effort, they often contain known vulnerabilities that can be exploited by attackers. Such vulnerabilities may lead to serious security issues like data breaches, unauthorized access, and system compromise.

Many developers, especially students, are not aware of the security risks present in the dependencies they use. Checking vulnerabilities manually is difficult and time-consuming. Therefore, there is a need for a simple and effective solution to identify these risks quickly.

To solve this problem, this paper introduces **Dependency Guard**, a web-based vulnerability detection system that uses the OSV API to scan software dependencies. The system allows users to enter package details and provides information such as CVE IDs, severity levels, and risk scores.

The proposed system is built using Flask and offers a lightweight and user-friendly interface. It helps users detect vulnerabilities easily and promotes secure software development practices.

2. SYSTEM ARCHITECTURE

The **Dependency Guard** system is designed as a simple web-based application that detects vulnerabilities in software dependencies using real-time data. The system is divided into four main modules that work together to provide accurate results.

2.1 Input Module

This module allows the user to enter the **package name and version** through a web interface. It collects user input and sends it to the backend for processing.

2.2 Processing Module

The processing module is implemented using the Flask framework. It receives the user input and prepares a request for vulnerability scanning. It handles the logic of communicating with the API and processing the response.

2.3 API Integration Module

This module connects the system to the **OSV (Open Source Vulnerabilities) API**. It sends the package details to the API and retrieves information such as:

- CVE IDs
- Severity level
- CVSS score

2.4 Output Module

The output module displays the results on the web interface. It shows:

- Number of vulnerabilities
- Risk level (Low, Medium, High)
- Detailed vulnerability information

3. METHODOLOGY

The **Dependency Guard** system follows a simple and efficient process to detect vulnerabilities in software dependencies using real-time data.

First, the user enters the **package name and version** through the web interface. This input is sent to the backend developed using the Flask framework. The backend processes the request and prepares it for vulnerability scanning.

Next, the system sends a request to the **OSV API**, which contains a database of known vulnerabilities. The API returns detailed information such as **CVE IDs, severity levels, and CVSS scores** related to the given dependency.

The received data is then analyzed by the system. Based on the severity and number of vulnerabilities, the system categorizes the risk level as **Low, Medium, or High**.

Finally, the results are displayed to the user on the web interface in a clear and structured format. This allows users to quickly understand the security risks and take necessary action.

4. MATERIALS AND TECHNOLOGIES

Materials (Software Components)

- **Python** – Used as the main programming language for backend development
- **Flask Framework** – Used to build the web application and handle user requests
- **OSV API** – Provides real-time data about known vulnerabilities
- **Web Browser** – Used to access and interact with the system

Technologies Used

- **HTML** – For creating the structure of the web interface
- **CSS** – For designing and styling the user interface
- **REST API** – For communication between the system and OSV database

5. IMPLEMENTATION

The **Dependency Guard** system is implemented as a web-based application using the Flask framework in Python. The system is designed to be simple, lightweight, and easy to use for students and developers.

The implementation begins with creating a user interface using **HTML and CSS**, where users can enter the **package name and version**. This data is sent to the Flask backend through HTTP requests.

The backend processes the input and sends a request to the **OSV API** using Python libraries such as `requests`. The API returns vulnerability data in JSON format, which includes details like **CVE IDs, severity levels, and CVSS scores**.

The system then processes this data and categorizes the risk level based on the severity of vulnerabilities. Finally, the results are displayed on the web interface in a clear and structured format.

Key Features of Implementation

- Real-time vulnerability scanning
- Simple and user-friendly interface
- Fast API response handling
- Lightweight system without database

Tools Used

- Python (Flask)
- HTML, CSS
- OSV API
- Requests Library

6. RESULTS AND DISCUSSION

The system successfully detects vulnerabilities in software dependencies and provides accurate and reliable results. It retrieves detailed information including CVE IDs, severity levels, and CVSS scores in real time. The results demonstrate that the system is efficient, fast, and useful for identifying potential security risks in applications.

7. CONCLUSION

The **Dependency Guard** system provides a simple and effective solution for detecting vulnerabilities in software dependencies. By using the OSV API, the system is able to identify known security issues and present accurate results in real time. The user-friendly interface makes it easy for students and developers to analyze the security of their applications.

The system helps in increasing awareness about dependency-related risks and promotes secure coding practices. It reduces the effort required for manual vulnerability checking and improves overall software security.

In the future, the system can be enhanced by adding features such as automatic project scanning, integration with GitHub, and advanced risk analysis.

.8. REFERENCES

https://osv.dev
https://cve.mitre.org
https://flask.palletsprojects.com
OWASP Dependency Check