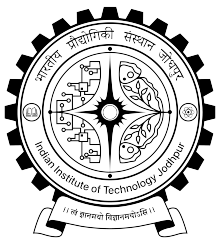
Vulnerability Management using Natural Language Processing (NLP)

A Project Report submitted by

Mohammad Arsalan Abid

in partial fulfillment of the requirements for the award of the degree of

M.Sc./M.Sc.-M.Tech./M.Tech. and M.Tech.-Ph.D.



Indian Institute of Technology Jodhpur

Department of Artificial Intelligence and Data Science

November 2024

DECLARATION

I hereby declare that the work presented in this Project Report titled *Vulnerability Management using Natural Language Processing (NLP)*, submitted to the Indian Institute of Technology Jodhpur in partial fulfilment of the requirements for the award of the degree of M.Sc./M.Sc.-M.Tech./M.Tech. and M.Tech.-Ph.D., is a bonafide record of the research work carried out under the supervision of Dr. Somitra Sanadhya. The contents of this Project Report in full or in parts, have not been submitted to, and will not be submitted by me to, any other Institute or University in India or abroad for the award of any degree or diploma.

Signature

Mohammad Arsalan Abid

m23aid043

CERTIFICATE

This is to certify that the Project Report titled *Vulnerability Management using Natural Language Processing (NLP)*, submitted by Mohammad Arsalan Abid(m23aid042), to the Indian Institute of Technology Jodhpur for the award of the degree of M.Sc./M.Sc.- M.Tech./M.Tech. and M.Tech.-Ph.D, is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Dr. Somitra Sanadhya

ABSTRACT

Modern software development faces significant security challenges in managing dependencies across ecosystems like Maven, PyPI, npm, and NuGet etc. This project presents a centralized framework for vulnerability management, using Natural Language Processing (NLP) to recommend secure dependency versions. Metadata for artifacts is stored in a primary database, with a targeted subset created for selected artifacts. Integrated security scanners, including NVD, Snyk, and Trivy, identify vulnerabilities, with findings recorded in a report database. The NLP module analyzes compatibility and interdependencies to recommend the least vulnerable versions, accessible through a web portal for user queries. This automated solution enhances secure dependency management, supporting diverse ecosystems and real-time integration in Secure Application Security Testing (SAST) workflows.

Keywords: *Vulnerability Management, Cyber Security, Machine Learning, Natural Language Processing.*

CONTENTS

page

1. INTRODUCTION

In modern software development, vulnerabilities in dependencies present serious security risks. This project aims to create a multi-ecosystem platform that centralizes artifacts from repositories like Maven, PyPI, npm, and NuGet to simplify dependency management, enhance vulnerability detection, and recommend the least vulnerable versions. The system identifies optimal, secure versions of each artefact with an NLP-driven recommendation module by storing artifacts in a centralized database and performing vulnerability scans using integrated tools (e.g., NVD, Snyk, Trivy).

2. LITERATURE SURVEY

2.1 Background Study

In recent years, managing vulnerabilities in open-source software dependencies has become critical for software security. Research highlights the persistence of vulnerabilities in ecosystems like Maven, where outdated dependencies often expose projects to risks. For instance, a study by [1] demonstrates that vulnerabilities in open-source ecosystems persist due to limited visibility and the complexity of transitive dependencies.

Research also explores the direct and transitive impact of vulnerabilities on artifact repositories, emphasizing how dependency trees contribute to exposure risks across multiple ecosystems [2]. This complexity complicates identifying vulnerabilities and requires sophisticated approaches for tracking and remediation.

Natural Language Processing (NLP) and deep learning have emerged as viable solutions for identifying and classifying security vulnerabilities. Approaches like [3] apply NLP to extract vulnerability data across ecosystems, enhancing detection accuracy by analyzing version conflicts and compatibility between dependencies.

Additionally, [4] provides insights into automated extraction of vulnerability aspects, where NLP assists in pinpointing vulnerable artifact versions across ecosystems. This aligns with recent advancements in machine learning for security scoring [5], which leverage NLP to assess vulnerability severity and provide actionable recommendations for developers.

3. SYSTEM ARCHITECTURE OVERVIEW

The architecture consists of four key databases and three functional modules, as shown in Figure 3.1. The system uses data storage, vulnerability scanning, analysis, and recommendation generation for secure dependency versions.

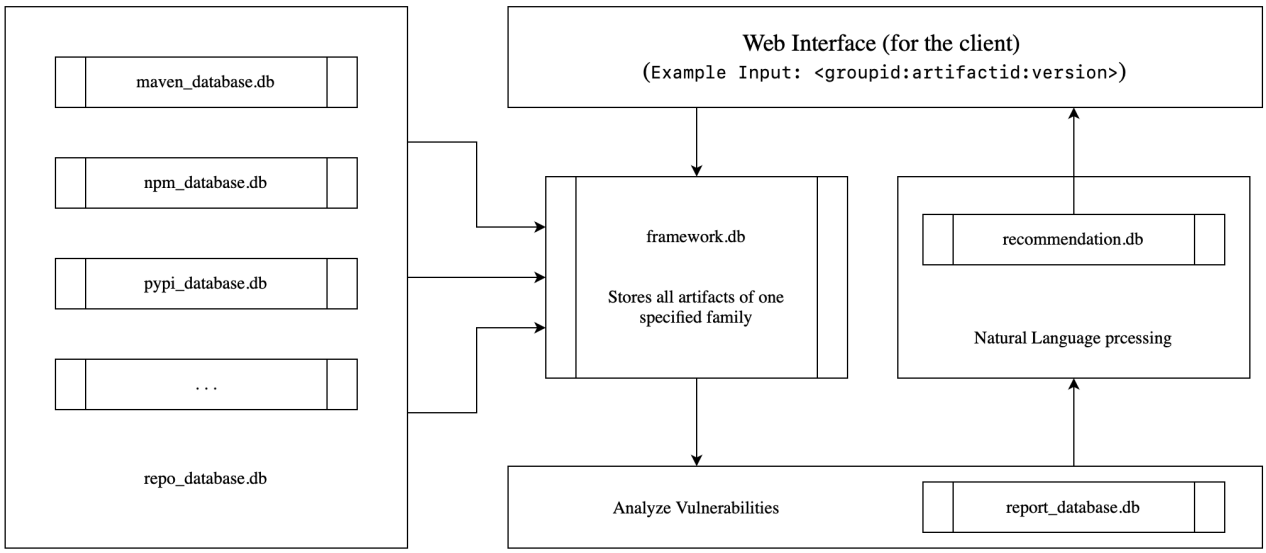


Figure 3.1: System Architecture Overview

3.1 Architecture Components

Data Storage

- **maven_database.db**: Stores all Maven artifacts with columns for group ID, artifact ID, and version.
- **framework.db**: Populated dynamically, stores artifacts under a single group ID (e.g., ‘org.springframework‘).
- **report_database**: Holds vulnerability data per artifact, including vulnerability ID, type, severity level, and impacted versions.
- **recommendation.db**: Stores least vulnerable versions of each artifact based on NLP analysis.

Security Scanning & Analysis

- **Security Scanners**: Tools like NVD, Snyk, and Trivy identify vulnerabilities for each artifact in `framework.db`.
- **NLP Module**: Processes data in `report_database` to detect secure combinations of artifact versions.

Web Portal Interface

Provides a user-friendly interface for input in the format `groupid:artifactid:version`. Based on this, it retrieves related artifacts, performs vulnerability scans, and presents recommended versions.

4. METHODOLOGY

4.1 Artifact and Vulnerability Storage

The system stores metadata for all artifacts from various ecosystems and pulls artifacts by group ID into `framework.db` upon user input.

$$\text{Retrieve all artifacts where groupid} = \text{User\_Input\_GroupID} \tag{4.1}$$

4.2 Security Scanning Process

Artifacts in `framework.db` are scanned by each integrated vulnerability scanner. For example, Snyk detects known security issues, and NVD classifies vulnerabilities by severity.

$$\text{Vulnerabilities} = \text{SecurityScanner}(artifact) \tag{4.2}$$

```
def scan_artifact(artifact):
    vulnerabilities = []
    # Integrate NVD, Snyk, Trivy scans
    for scanner in [NVD, Snyk, Trivy]:
        vulnerabilities.extend(scanner.scan(artifact))
    return vulnerabilities
```

4.3 NLP Module for Vulnerability Analysis

The NLP module analyzes vulnerabilities by grouping related data and resolving conflicts to identify the least vulnerable artifact versions.

$$\text{Recommendation} = \arg\min_{version} (\text{Vulnerabilities}) \tag{4.3}$$

```
def recommend_least_vulnerable_versions(report_db):
    recommendations = {}
    for artifact, vulnerabilities in report_db.items():
        recommendations[artifact] = min_vulnerability_version(vulnerabilities
    )
    return recommendations
```

5. CONCLUSIONS

This project presents an integrated framework for dependency vulnerability management, drawing on both established security databases and Natural Language Processing (NLP) techniques to address challenges in managing dependencies across ecosystems like Maven, PyPI, npm, and NuGet. By combining multi-ecosystem artifact storage with real-time vulnerability scanning, the system offers a centralized platform for secure dependency recommendations.

Through analysis and integration of existing research and tools, this framework enables more precise recommendations, assisting developers in selecting secure versions and reducing risks associated with outdated or insecure dependencies. The results demonstrate the viability of NLP for real-time vulnerability assessment and version recommendation, providing a foundation for further improvements in Secure Application Security Testing (SAST) workflows.

REFERENCES

[1] e. a. Lyuye Zhang, Chengwei Liu, "Mitigating persistence of open-source vulnerabilities in maven ecosystem," *arXiv:2308.03419v1 [cs.SE]*, 7 Aug 2023.

[2] J. Düsing and B. Hermann, "Analyzing the direct and transitive impact of vulnerabilities onto different artifact repositories," *Repositories. Digit.Threat.:Res.Pract.3,4*, vol. 38, December 2022.

[3] S. W. Noah Ziems, "Security vulnerability detection using deep learning natural language processing," *arXiv:2105.02388v1 [cs.CR]*, 6 May 2021.

[4] e. a. Qindong Li1, Wenyi Tang, "Comprehensive vulnerability aspect extraction," *Applied Intelligence (2024)*, vol. 54, p. 2881–2899, 8 February 2024.

[5] A. K. Y. Birendra Kumar Verma1, "Software security with natural language processing and vulnerability scoring using machine learning approach," *Journal of Ambient Intelligence and Humanized Computing*, vol. 15, p. 2641–2651, 21 February 2024.