

**School of Computing**

PROJECT REPORT

**SECURE CODING**

**20CYS302**

**Submitted by**

SRIDHARAN S VIGNESWARAN S SRISH N

CH.EN.U4CYS21080 CH.EN.U4CYS21090 CH.EN.U4CYS21081

Amrita Vishwa Vidyapeetham Chennai – 601 103, Tamil Nadu, India.

October 2023

## BONAFIDE CERTIFICATE

**University Register Numbers**:

CH.EN.U4CYS21080

CH.EN.U4CYS21081

CH.EN.U4CYS21090

This is to certify that this is a Bonafide record work done by **Mr.Sridharan S, Mr Srish N** and **Mr Vigneswaran S** studying BTech Computer Science with Cyber Security

**Internal Examiner1 Internal Examiner2**

Vulnerability Scanner for Node.js

1. Abstract

In today's rapidly evolving digital landscape, cybersecurity and the assessment of software vulnerabilities have become paramount concerns. Our project aims to develop a powerful and versatile vulnerability scanner using Node.js, a widely adopted JavaScript runtime environment. The NodeJS Vulnerability Scanner is a powerful security tool designed to identify and mitigate potential vulnerabilities in Node.js applications. Leveraging the flexibility and efficiency of Node.js, this project aims to provide a comprehensive solution for developers and system administrators to proactively detect and address security weaknesses in their Node.js codebases. Through automated scanning techniques, it analyzes dependencies, code patterns, and configurations to flag potential vulnerabilities, including known exploits and emerging threats. The scanner offers detailed reports, prioritized recommendations, and remediation guidance, empowering users to enhance the security posture of their Node.js applications, reduce risks, and fortify against cyber threats.

***Keywords—Cybersecurity, software vulnerabilities, Node.js, cyber threats.***

1. Introduction

In today's digitally connected world, cybersecurity is paramount. As the volume and sophistication of cyber threats continue to rise, it's crucial for organizations and individuals to proactively identify and mitigate vulnerabilities in their software and systems. To address this pressing need, we are embarking on an innovative project to develop a powerful and user-friendly Vulnerability Scanner using Node.js. Our Node.js Vulnerability Scanner aims to revolutionize the way we approach cybersecurity. Leveraging the flexibility and scalability of Node.js, we are creating a cutting-edge tool that will assist developers, IT professionals, and security experts in identifying potential weaknesses in their applications, libraries, and systems.

Features of our Vulnerability Scanner includes:

* Comprehensive Scanning: Our scanner will employ advanced algorithms and databases to conduct thorough scans for known vulnerabilities, covering a wide range of programming languages and libraries.
* Customizable Alerts: Users can configure alerts and notifications, ensuring timely responses to potential security risks.
* CVE ID Information Retrieval: A resourceful tool that provides immediate access to information about known security vulnerabilities (CVEs). This empowers users with the latest insights into potential threats.
* User friendly Interface: Our user-friendly web-based interface will make it easy for both beginners and experts to initiate scans, interpret results, and take action to enhance security.

Node.js is a popular JavaScript runtime environment that executes JavaScript code outside the web browsers such as being a web server to serve the client. Node.js ecosystem including millions of vulnerabilities, such as command injection. Our Node.js Vulnerability Scanner project embodies our commitment to making cybersecurity accessible and effective for everyone. By harnessing the power of Node.js, we are poised to deliver a state-of-the-art tool that safeguards digital assets and promotes a more secure online environment.

1. Literature Survey

The second resource, "A Comprehensive Review of Vulnerability Scanners," by Aneiba, Derhab, and Damiani, provides a broad overview of vulnerability scanners. This review categorizes vulnerability scanners into various types, including network-based, host-based, and web application scanners. Understanding these categories is vital for tailoring a Node.js vulnerability scanner to specific use cases and requirements. The paper emphasizes the significance of vulnerability databases in the context of vulnerability scanning. Keeping the scanner's database up-to-date with the latest vulnerabilities is critical for accurate assessments. The literature survey should explore existing vulnerability databases that the Node.js scanner can leverage. Automated and manual scanning techniques are discussed in the review, with insights into their respective strengths and weaknesses. Striking the right balance between automation and manual verification is crucial for the Node.js scanner's effectiveness. While automation speeds up the scanning process, manual validation ensures the accuracy of identified vulnerabilities.

The authors of the comprehensive review also delve into the Common Vulnerability Scoring System (CVSS), which is used to assess the severity of identified vulnerabilities. Understanding CVSS metrics is essential for prioritizing and addressing vulnerabilities in Node.js applications effectively. Challenges in vulnerability scanning are discussed, including issues like false positives, false negatives, and the impact of network complexity[1]. These challenges should be taken into account when designing and developing the Node.js-based scanner to minimize false alarms and ensure accurate results. Regarding, a recent work that has the same grounds with Node.XP, which attempts to mitigate injection vulnerabilities in Node.js applications. To accomplish that, during the installation process of a third-party Node.js module, a check is performed to detect and rewrite APIs that appear to be prone to injection. Static analysis of the possible input values that will be passed to the aforementioned APIs is performed; if the static analysis does not yield a definitive result.[ 2]. In a large-scale study about the runtime behavior of the eval() function in java Script application is presented. Their findings were very interesting as they concluded that the eval() function can not be always replaced exclusively by other functions. the results can be repeated and validated for server-side Node.js. Code injection attacks have been identified also in HTML% applications. The Author of [3] presented JS-SAN, a framework that multi gates the effects of java script code injection vulnerabilities in HTML5 based web application.

1. Methodology

The methodology behind our web application is rooted in providing users with a diverse array of choices, catering to their specific security needs. This approach empowers users to proactively address software vulnerabilities and bolster the security of their digital assets.

* XSS and SQL Injection Detection:

Specialized Scanners: To combat Cross-Site Scripting (XSS) and SQL injection vulnerabilities, we have dedicated scanners that identify websites prone to these attacks. These scanners employ advanced techniques to detect and mitigate potential threats, safeguarding web applications against malicious intrusions.

* CWE ID Information Retrieval:

Database Integration: Our application integrates a CWE (Common Weakness and Enumeration) ID information retrieval system that fetches data from the database. This feature equips users with up-to-date information on known security vulnerabilities, enabling them to stay informed and take timely action.

* Static and dynamic Analysis of the code:

Our application supports static and dynamic analysis over Node JS languages. We employ a range of open-source tools such as Synk, etc. and others to thoroughly scrutinize codebases for potential security vulnerabilities. This proactive approach assists developers in creating secure applications from the ground up.

* User-Friendly UI:

All these functionalities are seamlessly integrated into a user-friendly and intuitive user interface. This ensures that users, regardless of their technical expertise, can access and utilize the security tools with ease.

Our methodology centers around providing a comprehensive, accessible, and user-centric platform for addressing software vulnerabilities. By offering a wide spectrum of tools and leveraging state-of-the-art, our web application empowers users to take proactive measures in securing their digital assets, thereby contributing to a safer and more secure digital ecosystem.

**Static Analysis tool**

NodeJs Scan:

NodeJs Scan is a static security code scanner for Node.js applications. It is designed to identified the security vulnerabilities in the source code of Node.js applications. In python uses the libsast and semgrep libraries to perform static analysis of the code. And to use NodeJs Scan we need to install Postgres and configure SQLALCHEMY\_DATABASE. NodeJsScan module is used to find the insecure code pattern in the application and report that as a analyser.

Pattern Matcher:

Pattern Matcher scan through the code and find the matching pattern between the code and vulnerability data in CWE and it gives the vulnerability name, code, severity and description of the vulnerability.

Synk:

Synk is a tool it will scan and fix security vulnerability in our code. It analysis both static and dynamic analysis of the code, it check for deprecated package and modules used and suggest upgrades to mitigate the vulnerability. And it gives the result EPSS (Exploit Prediction Scoring System) percentage, and insights in to the vulnerability.

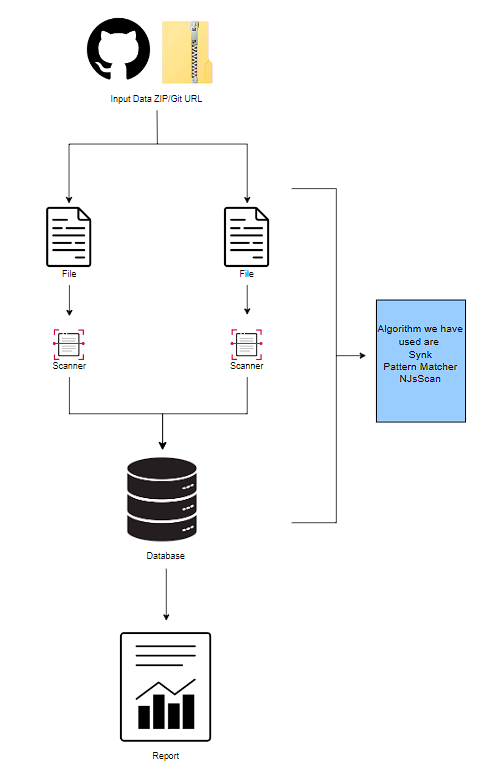


Fig. 1 System Architecture

Our user gives the input as a Zip File or github link or a file after uploading it and click the scan button after that it will analysis through NJJS scan module and keyword’s matching and it njsscan is a static application testing (SAST) tool that can find insecure code patterns in your node.js applications using simple pattern matcher from libsast and syntax-aware semantic code pattern search tool semgrep. And it fetch the details from the CWE database and after encountered the vulnerability it give the Scan Report to the email and also in our webpage.

Results and Analysis:

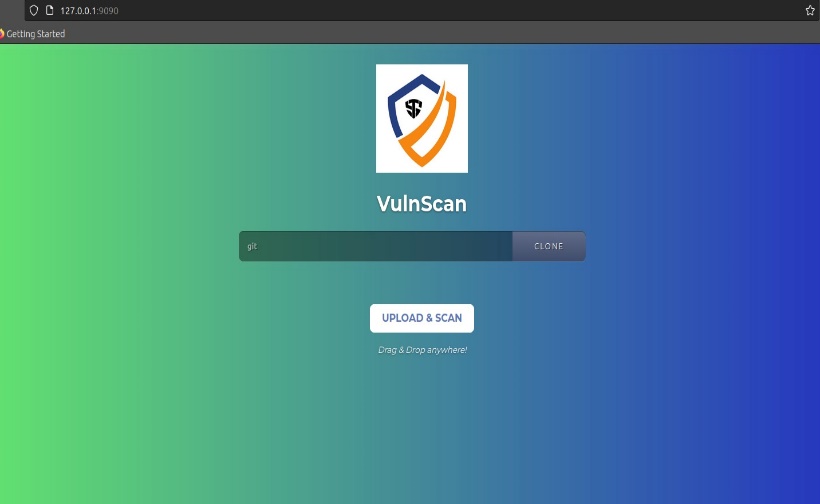


Fig.2 Website interface of the application

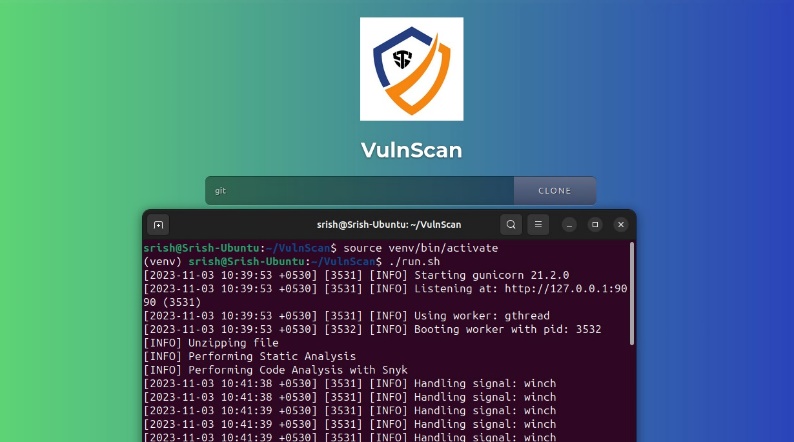


Fig.3 Background Code Analysis

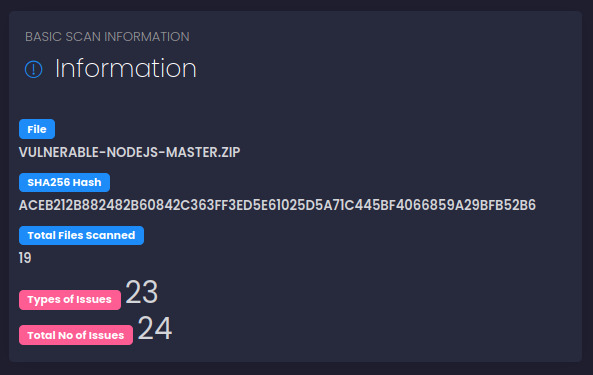


Fig.4 Summary of Analysis and graph

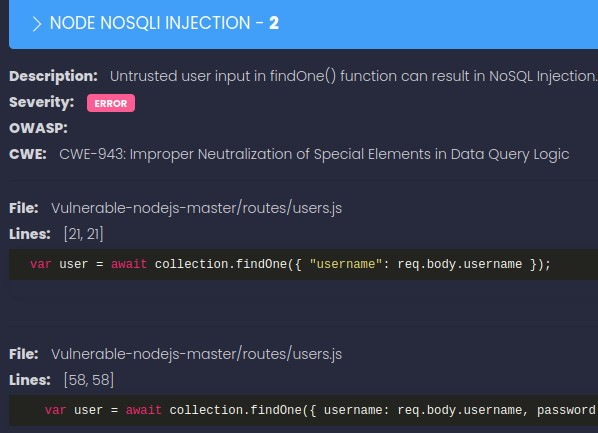


Fig.5 Issues Vulnerability found in the Input Code

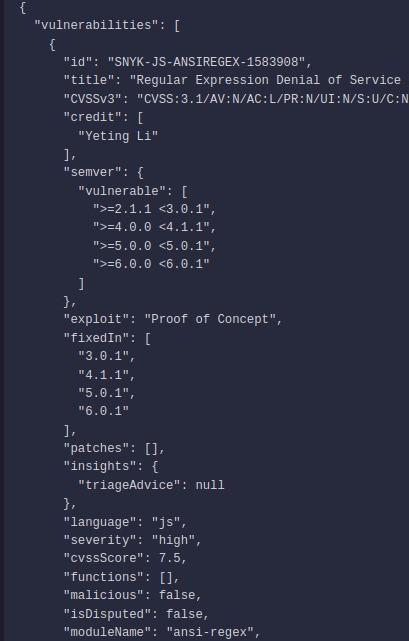


Fig.6 Synk Analysis of the vulnerability



Fig.7 EPSS Score



Fig.8 Upgrade Patches

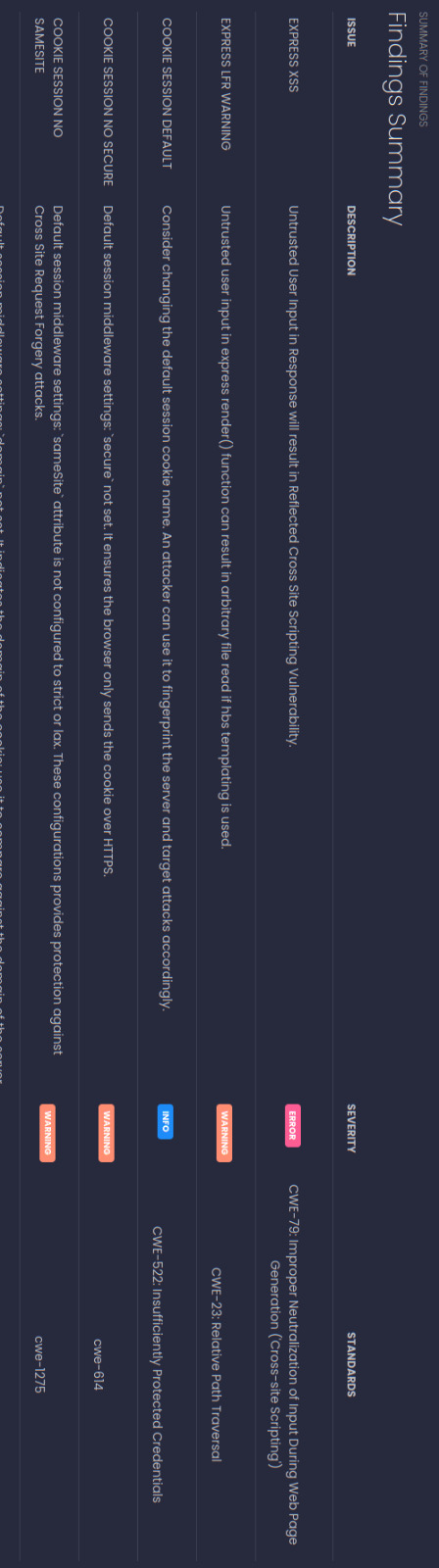


Fig.9 Summary of the Vulnerability Foun

Comparison:

Comparison of the VulScan and Vulnert

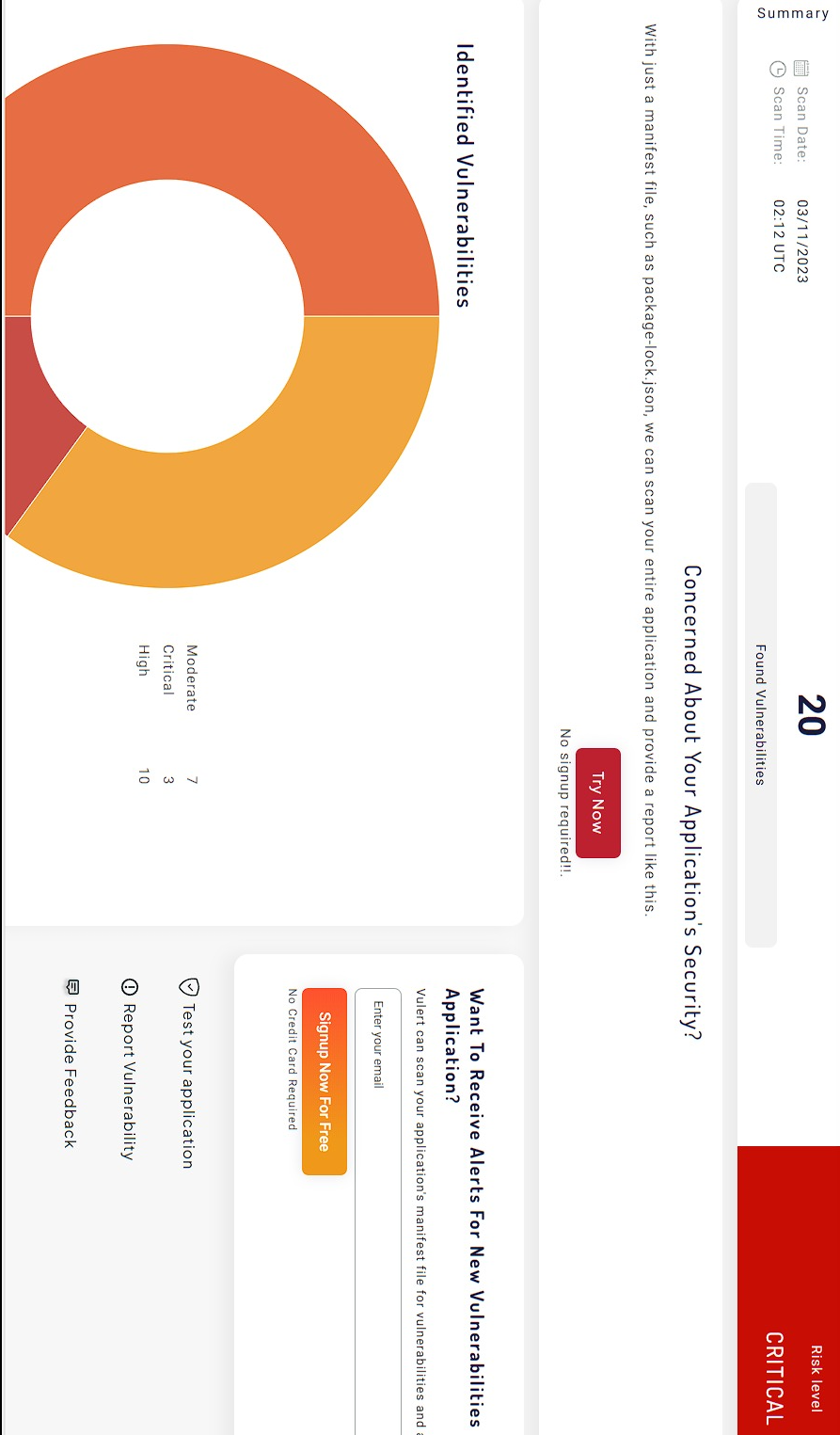


Fig.10 Vulnert Analysis for the given output

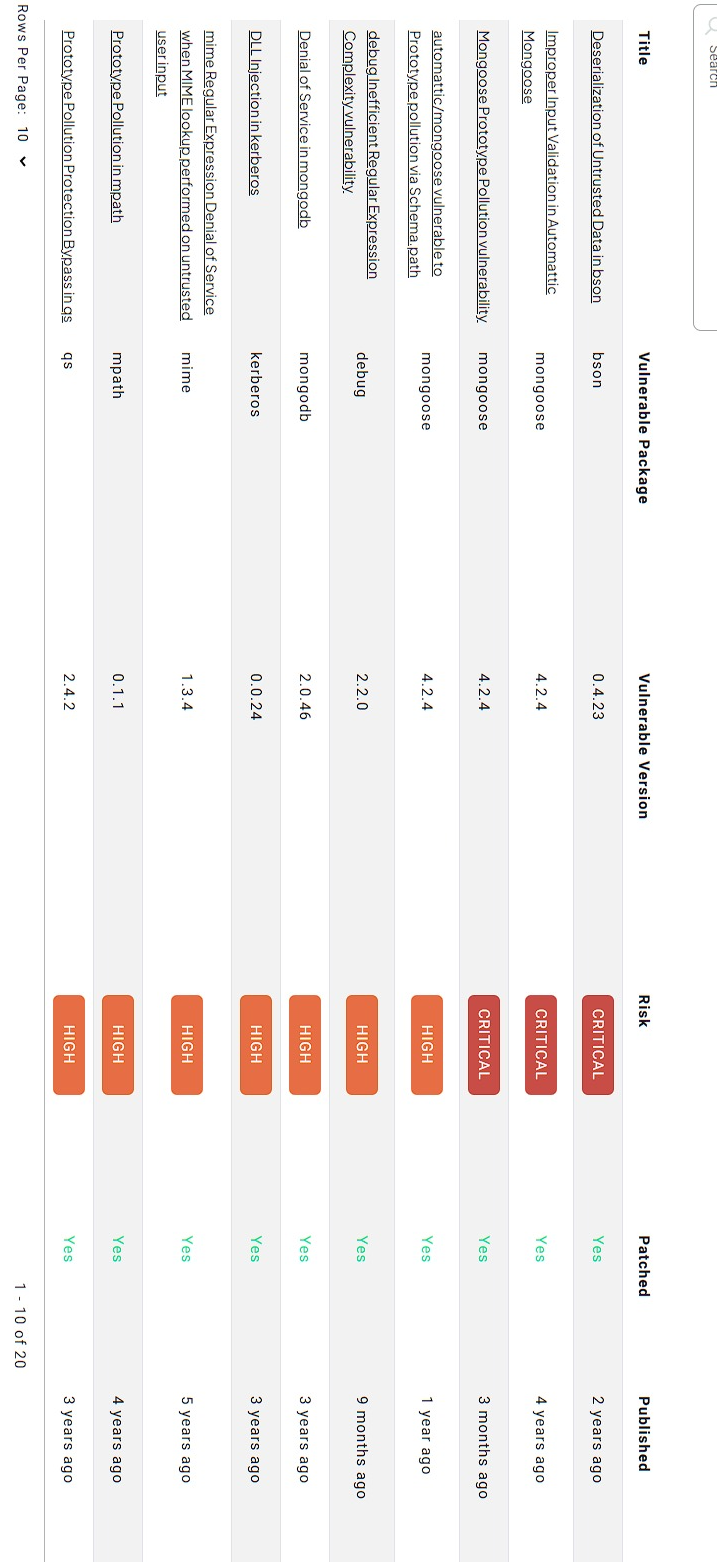


Fig .11 Summary of the vulnerability

Comparing Vulnert and Our VulnScan highlights several important aspects regarding their capabilities and capabilities in the context of static analysis for detecting code vulnerabilities. Let's explain the important points in more detail.

1. Detection sensitivity:

VulnScan appears to be more sensitive than Vulnert in detecting vulnerabilities. Detects numerous vulnerabilities in your code. This increased sensitivity helps ensure that potential issues don't go unnoticed.

2. Comprehensive Assessment:

VulnScan goes beyond mere detection and provides upgrade suggestions. This means that it not only identifies vulnerabilities but also offers actionable recommendations for fixing or improving the code. This comprehensive assessment can save developers valuable time and resources in addressing potential issues.

3. Detailed vulnerability report:

While Vulnert provides an overview of vulnerabilities, VulnScan provides a more detailed and detailed view of discovered vulnerabilities. This detailed information is invaluable to developers and security teams in understanding the nature and severity of the issue.

4. Dynamic analysis:

A notable advantage of VulnScan is that it can perform dynamic analysis of your code. Dynamic analysis runs your code to identify vulnerabilities that static analysis alone might not be able to detect. This ensures that all areas of your code, including runtime behavior, are covered and improves your overall security score.

5. User-friendly interface:

VulnScan's user experience and interface will be more user-friendly and accessible as it provides a comprehensive overview of vulnerabilities and their possible solutions. This makes it easier for developers to use the tool effectively.

6. Real-time Feedback:

The dynamic analysis component of VulnScan may offer real-time feedback during code execution, which can be extremely valuable for identifying and addressing vulnerabilities as they are encountered, rather than waiting for post-analysis reports.

7. Customizability:

Depending on the specific needs of the development and security teams, VulnScan may offer options for customizing the scanning process and defining the scope of the analysis. This adaptability can be crucial for tailoring the tool to the project's requirements.

In summary, VulnScan appears to have a broader and more feature-rich set of capabilities compared to Vulnert. It not only detects a greater number of vulnerabilities but also offers detailed information, upgrade suggestions, and dynamic analysis. These features make VulnScan a valuable tool for developers and security professionals looking to enhance code security and address vulnerabilities efficiently. However, the choice between the two tools may also depend on factors like ease of integration, cost, and specific project requirements.

Conclusion:

In this paper, we have presented a web application platform that offers a suite of essential security tools, including file and website scanning, CWE ID information retrieval, static and dynamic code analysis, Cross-Site Scripting (XSS) scanning, and web server vulnerability scanner.

Our project's significance lies in its ability to empower users, ranging from web developers to security professionals, with a set of tools that enable them to assess, mitigate, and fortify. By consolidating these tools into a single, unified platform, we streamline the security assessment process, making it more accessible and efficient for a wider range of users. Additionally, we aim to provide regular database updates for CWE information and maintain compatibility with the latest web technologies.

In conclusion, our web application present a valuable resource for the web development and cybersecurity communities. By combining a wide array of security tools into a unified platform, we contribute to the ongoing efforts to create a more secure and resilient digital environment. We look forward to the continued development and refinement of our project, with the ultimate goal of fostering a safer online ecosystem for all users.

**Reference**

[1] "A Comprehensive Review of Vulnerability Scanners," by Aneiba, Derhab, and Damiani

[2] Gauthier F, Hassanshahi B, Jordan A. AFFOGATO: Runtime detection of injection attacks for node.js. In: Companion proceedings for the ISSTA/ECOOP 2018

[3] Gupta S, Gupta B. JS-SAN: defense mechanism for HTML5-based web applications against javascript code injection vulnerabilities. Secur Commun Netw 2016.

[4] Kim, H.Y., Kim, J.H., Oh, H.K. et al. DAPP: automatic detection and analysis of prototype pollution vulnerability in Node.js modules 2021.

[5] Alfadel M, Costa D, Shihab E and Adams B. (2023). On the Discoverability of npm Vulnerabilities in Node.js Projects. ACM Transactions on Software Engineering and Methodology. 32:4. (1-27). Online publication date: 31-Oct-2023.

[6] Lau T. (2022). A Class of Software-Layer DoS Attacks in Node.js Web Apps 2022 6th International Conference on Cryptography, Security and Privacy (CSP). 0.1109/CSP55486.2022.00028. 978-1-6654-7975-2. (108-113).

[7] A. Al Anhar and Y. Suryanto, "Evaluation of Web Application Vulnerability Scanner for Modern Web Application," 2021 International Conference on Artificial Intelligence and Computer Science Technology (ICAICST), Yogyakarta, Indonesia, 2021, pp. 200-204

[8] Anupam, Aman, et al. "Analysis of Open Source Node. js Vulnerability Scanners." International Research Journal of Engineering and Technology (IRJET) e-ISSN (2020): 2395-0056.

[9] Maurel, Héloïse, Santiago Vidal, and Tamara Rezk. "Comparing the Detection of XSS Vulnerabilities in Node. js and a Multi-tier JavaScript-based Language via Deep Learning." ICISSP 2022-8th International Conference on Information Systems Security and Privacy. 2022.

[10] Brito, Tiago, et al. "Study of JavaScript Static Analysis Tools for Vulnerability Detection in Node. js Packages." arXiv preprint arXiv:2301.05097 (2023).