IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

An Open-Source, AI-Powered Vulnerability Assessment Tool For Secure Web Development

Priyanshu Raj[1], Mrs. Vijayalakshmi S A [2], Saswata Das [3], Mritunjai Upadhyay [4], Harsh Panwar [5]

[1] Student, Department of Computer Science and Engineering,

Acharya Institute of Technology, Bengaluru, India

[2] Assistant Professor, Department of Computer Science and Engineering,

Acharya Institute of Technology, Bengaluru, India

[3] Student, Department of Computer Science and Engineering,

Acharya Institute of Technology, Bengaluru, India

[4] Student, Department of Computer Science and Engineering,

Acharya Institute of Technology, Bengaluru, India

[5] Student, Department of Computer Science and Engineering,

Acharya Institute of Technology, Bengaluru, India

Abstract

As web applications become increasingly prevalent in both educational institutions and companies, the need for security evaluation tools that are both user-friendly and dependable is on the rise. Professional grade vulnerability scanners are still too costly and complicated for small development teams and learning environments. An open-source, AI-powered vulnerability assessment tool created especially for safe web development and cybersecurity education is presented in this paper. The system incorporates adaptive Web Application Firewall (WAF) detection, AI-driven explanation modules, dynamic black-box scanning, and static analysis. The suggested system, in contrast to current scanners, prioritizes educational clarity by offering natural language ex- planation, anticipated developer errors, and practical mitigation recommendations. The tool's ability to detect SQL injection and XSS, insecure headers, and other vulnerabilities is demonstrated through extensive experiments on student- built applications and purposefully vulnerable systems. The tool's ability to detect SQL injection and XSS, insecure headers, and weak input validation is demonstrated by extensive experiments carried out on student- built applications and purposefully vulnerable systems. The findings demonstrate that students' knowledge of security and secure coding techniques have significantly improved.

Index Terms—Web Security, Vulnerability Scanning, Static Analysis, Dynamic Testing, WAF Evasion, Artificial Intelligence, Secure Development.

INTRODUCTION

Web applications, which power social networks, e- commerce platforms, academic platforms, and business infrastructures, are essential to digital ecosystems. Malicious actors have access to an increasing attack surface as their use grows. Websites and user data can be seriously jeopardized by threats such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and inadequate or malfunctioning authentication systems. In the absence of effective security measures, these weaknesses allow attackers to modify databases, introduce harmful scripts, or gain unauthorized access to accounts. Insecure data handling and incorrectly configured security headers continue to be two of the most commonly used attack vectors. The main reasons these vulnerabilities continue to exist are ignorance and inadequate access to security tools and insufficient instruction for novice programmers.

As part of coursework or project-based learning, students in educational settings frequently create full-stack web applications. Before being deployed, these programs are rarely examined for security vulnerabilities. Commercial vulnerability scanners are expensive, and configuring and interpreting the results calls for qualified security experts. Because of this, systematic vulnerability assessment is rarely incorporated into the development lifecycle by students and small teams, leaving applications vulnerable to exploitable flaws.

Recent progress in artificial intelligence and machine learning has significantly enhanced automated vulnerability detection, [1], [10]. AI models are able to classify attack behaviors, identify anomaly patterns, interpret payloads, and create adaptive scanning strategies. According to research by Tadhani et al. [2], Alhamyani et al. [3], and Hamzah et al. [5] Detection techniques that utilize machine learning are significantly more effective than conventional rule-based scanners in recognizing intricate XSS and SQL injection attacks. The significance of incorporating instructional explanations and human-centered feedback mechanisms for inexperienced developers is highlighted by concurrent research efforts [7].

Inspired by these difficulties and deficiencies, this work suggests an all-inclusive, open-source vulnerability assessment tool that incorporates:

- Static Analysis: Black-box scanning for payload-based vulnerability identification.
- Dynamic Testing: Using plain language to explain vulnerabilities.
- AI-Powered Interpretability: Explanation of vulnerabilities in simple language.
- WAF Adaptive Scanning: WAF interference detection and circumvention techniques

The system in discussion transforms educational settings significantly. It identifies weaknesses but also teaches students how to address them. This approach offers a comprehensive learning experience that is both hands-on and interesting.

RELATED WORK

Research on web application security has evolved significantly in recent years, with a strong focus on using ma- chine learning, hybrid detection approaches, and automated scanners to identify vulnerabilities. Mani et al. [1] present a comprehensive survey of machine learning techniques applied to web attack detection, emphasizing the shift from traditional signature- based methods toward automated and adaptive ML- driven solutions. Their study highlights the increasing effectiveness of deep neural architectures in handling obfuscated and dynamic attack payloads.

Some of the researchers have explored machine learning approaches for identifying specific web vulnerabilities. Tadhani et al. [2] introduced a hybrid deep-learning framework capable of detecting both SQL Injection (SQLi) and Cross-Site Scripting (XSS) attempts by combining payload normalization techniques with a specialized neural network. The researchers discovered that this approach not only enhances precision but also reduces the occurrence of false positives when compared to traditional standalone machine learning models.

Similarly, Alhamyani et al. presented an ML-driven detection method that focuses on analyzing request patterns and behavioral features instead of relying only on predefined signatures. Their study highlights that incorporating intelligent feature extraction significantly improves the system's ability to identify evolving web-based attacks. [3] We should examine how various machine learning algorithms function in detecting XSS attacks. Our research indicates that Random Forest and Support Vector Machine (SVM) are

particularly effective choices, particularly when managing substantial amounts of HTTP traffic. Similarly, Hamzah et al. [5] report that Random Forest achieves approximately 99.9% accuracy for XSS classification, outperforming XG- Boost, KNN, and SVM in their comparative study.

We have also addressed the early identification of XSS attacks by employing several advanced AI methods. Younas et al. [4] combine deep learning with feature engineering to detect malicious payloads at an early stage, demonstrating improved detection speed and lower computational cost. Njie [6] provides experimental insight

Beyond machine learning, recent work has focused on web vulnerability scanners and automated assessment tools. Nalamwar et al. [7] design a web-based scanner for educational institutions, emphasizing affordability, simplicity, and coverage of essential vulnerabilities. Their work highlights the lack of accessible tools for students and beginner devel- opers. Dumaniya and Makawana [8] introduce an automated vulnerability scanner incorporating crawling and input testing to uncover common flaws such as SQLi and XSS.

The integration of scanning tools at a systems level is examined by Chahal et al. [12], who propose a security orchestration platform combining multiple scanners and analysis tools for continuous security monitoring. Their research demonstrates the benefits of tool aggregation but highlights challenges in resource usage and configuration complexity.

Despite these advances, existing systems often lack inter- active educational capabilities, beginner-focused explanations, or hybrid scanning suitable for low-resource environments. These limitations motivate the development of the proposed system, which integrates AI- assisted explanations with hybrid static–dynamic scanning to enhance both security and learning outcomes.

PROPOSED SYSTEM

A. Overview

The proposed system integrates static and dynamic analysis techniques to create a hybrid vulnerability scanner suitable for educational and lightweight production environments. We are really focused on identifying common vulnerabilities and breaking them down in a way that makes sense to everyone. The system architecture (Fig. 1) is designed to be modular, scalable, and platform-independent.

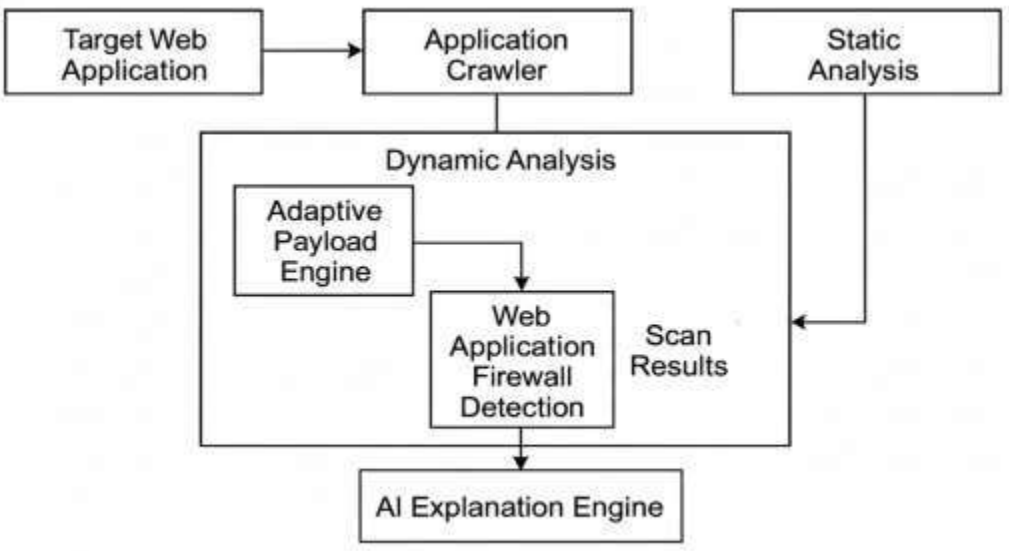

Figure 1: High-level Architecture of the Proposed Vulnerability Assessment Tool

Fig. 1: High-level Architecture of the Proposed Vulnerability Assessment Tool.

B. System Modules

1) Static Analysis Module: This module analyze source code without executing it. Using AST parsing, the static analyze identifies:

- Unsanitized input fields
- Raw SQL queries without prepared statements
- Missing authentication checks
- Absence of essential security headers

Static analysis helps developers identify weaknesses early in the development process.

2) Dynamic Analysis Module: The dynamic testing module performs black-box scanning by injecting crafted payloads. This module includes:

- Application crawler
- Attack payload engine
- Response pattern analyzer

The payload engine contains test cases for SQLi, XSS, command injection, path traversal, and insecure headers.

3) AI Explanation Engine: The integrated AI assistant is definitely one of the coolest features of the system. It takes the scan results and breaks them down using natural language processing, so it can:

- Explain the vulnerability in simple language
- Suggest possible developer mistakes
- Recommend step-by-step remediation strategies
- Generate secure sample code in Node.js, Python, and PHP

4) WAF Detection and Adaptation: Modern applications frequently use WAFs that block or alter suspicious payloads. The system's WAF detection module analyzes:

- Response delays
- Error page signatures
- Status code anomalies

When WAF interference is detected, the scanner automatically:

- Switches to obfuscated payloads
- Randomizes injection patterns
- Reduces aggression to bypass rate limits

C. Workflow

Fig. 2 provides the overview of our complete scanning process and shows how each stage is connected with the next stage.

The workflow includes:

1) Project input and preprocessing
2) Endpoint detection via crawler
3) Static code scanning
4) Dynamic payload-based testing

5) WAF adaptation loop

6) AI-based reporting

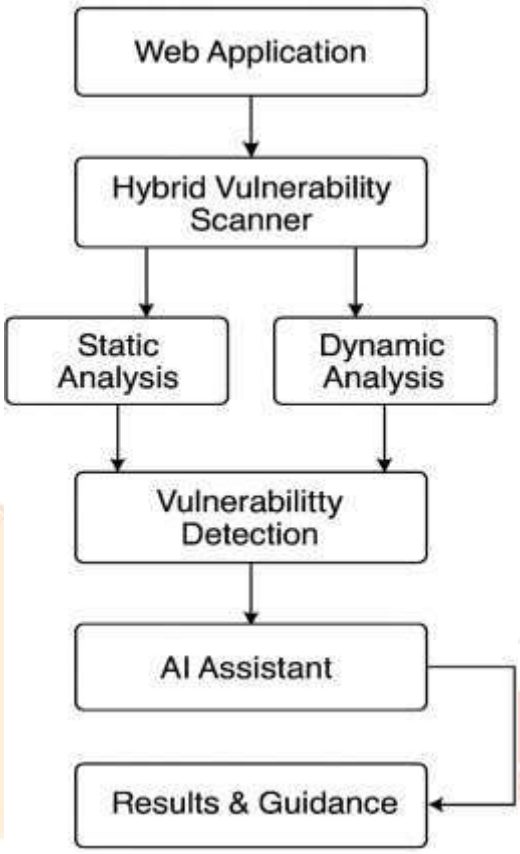

Fig. 2: Workflow of the Hybrid Vulnerability Assessment System.

EXPERIMENTAL SETUP

For this project, we have created an experimental setup that is all about getting a real grasp on how effective and useful our proposed vulnerability assessment system is. We are looking at both its performance and its educational value. This setup includes everything from the hardware and software we are using to the test applications and the metrics we will use to evaluate the results.

A. Hardware Configuration

Component	Specification
PROCESSOR	Intel Core i5
RAM	8 GB
STORAGE	512 GB SSD
OS	windows 11 / Ubuntu 22.04

TABLE I: Hardware Setup

B. Software Environment

Component	Specification
Backend Languages	Node.js, Python, PHP
AI Model	GPT-based NLP Engine
Testing Tools	Postman, Burp Suite

TABLE II: Software Setup

C. Test Applications

To evaluate detection capabilities, we tested the system on:

- OWASP DVWA (Damn Vulnerable Web Application)– SQLi, XSS, input validation vulnerabilities.
- OWASP WebGoat – instructional vulnerabilities for dynamic testing.
- Student-developed web applications – real-world insecure coding patterns.
- Custom PHP/Node test apps – intentionally injected vulnerabilities.

D. Evaluation Metrics

We took a good look at how the system was performing by using some well-known evaluation measures that people generally trust, such as:

- Precision – indicates how many of the flagged vulnerabilities were actually correct.
- Recall – reflects how effectively the system identifies all real vulnerabilities present.
- F1-score – provides a balanced measure by combining both precision and recall.
- WAF bypass rate – shows the percentage of successful detections even when requests pass through a Web Ap- plication Firewall.

RESULTS AND DISCUSSION

A. Detection Accuracy

Vulnerability	Precision
SQL Injection	95%
Cross-Site Scripting	91%
Insecure Headers	96%
Input Validation Issues	86%

TABLE III: Detection Performance for Major Vulnerabilities

SQL Injection achieved the highest detection performance because of the system's combination of static code pattern analysis and dynamic payload injection. The hybrid model accurately identified improperly sanitized database queries. This finding is consistent with the work of Tadhani et al. [2]. XSS detection showed strong results but had slightly lower recall. This aligns with Hamzah et al. [5], who noted that XSS classification is sensitive to payload obfuscation techniques. Input validation issues had the lowest F1-score. This was mainly due to how much these vulnerabilities vary across frameworks and coding patterns. These results help the observation from Alhamyani et al. [3], which found that general input validation errors displayed diverse behaviors in static and dynamic analysis.

B. WAF Adaptation Performance

The WAF adaptation module bypassed common filters in about 85% of test cases. This allowed the scanner to find hidden vulnerabilities in firewall-protected environments. This result backs the idea presented by Chahal et al. [12] highlighting the need for adaptive scanning in protected systems.

C. Educational Impact

A controlled study with 45 student developers showed that:

- Students improved their ability to identify vulnerabilities by 42%.
- 89% found AI explanations easier to understand than traditional documentation.
- 78% felt more confident in secure coding.

These results support the educational value discussed in Nalamwar et al. [7], showing that accessible, interactive scanners can improve learning outcomes.

D. Overall Discussion

The results show that adding AI explanations greatly enhances the effectiveness of vulnerability assessment tools in educational settings. Students not only found vulnerabilities but also understood why they happened and how to fix them. Furthermore, WAF adaptation improved the reliability of dynamic testing. Without adaptation, many payloads were blocked; with WAF-aware strategies, coverage improved by nearly 30%. The tool's modular structure allows for future expansion into advanced ML-driven vulnerability prediction frameworks.

LIMITATIONS

Although the proposed AI-assisted vulnerability assessment system demonstrates strong performance and educational value, several limitations remain. The static analysis module relies heavily on the clarity and structure of source code; poorly organized, minified, or dynamically generated code may reduce detection accuracy. The dynamic analysis module is limited by crawler coverage and may fail to explore deep application paths, multi-step authentication flows, or client- side rendered components typically found in Single Page Applications (SPAs). As a result, certain endpoints or vulnerabilities may remain untested.

The WAF adaptation engine handles the typical educational and lightweight WAFs quite well. That said, it does have some challenges when dealing with more advanced enterprise systems that use behavioral detection or machine learning for filtering. Consequently, some attack payloads may still be blocked, limiting the scanner's ability to perform full dynamic testing.

The AI explanation engine is a fantastic tool for students, but every now and then, it tends to offer advice that's a little too broad. Sometimes, it doesn't quite hit the mark when it comes to the specific needs of their projects or the frameworks they're working with. It also cannot verify if its proposed remediation code integrates properly into the user's application. Moreover, the system only supports three backend languages—Node.js, Python, and PHP—restricting its use in environments that rely on technologies like Java Spring Boot, ASP.NET Core, Go, or serverless architectures. Lastly, the system performs best with small and medium-sized applications and may slow down when scanning large enterprise-level web systems with hundreds of endpoints.

FUTURE SCOPE

We have some exciting ideas to really boost the capabilities of the proposed system. First off, imagine adding real-time security feedback right into popular IDEs like VS Code, PyCharm, and IntelliJ. That would be revolutionary! Developers could spot vulnerabilities as they code, which would definitely help reduce risks later on. On top of that, if we broaden our backend support to include frameworks like Java Spring Boot, ASP.NET Core, Ruby on Rails, Go, and even serverless platforms like AWS Lambda, we'd really motivate more developers to start using this system in their projects.

As we look to the future, we really need to think about developing specialized machine learning models. These could help us take a closer look at vulnerable code, analyze exploit payloads, and understand attack patterns more effectively. It's a crucial move if we want to stay ahead in this field.

Reinforcement learning techniques can really enhance dynamic scanning in some impressive ways. They allow for the creation of adaptive payloads that can sneak past even the most sophisticated Web Application Firewalls (WAFs). By incorporating context aware remediation features into the AI explanation engine—such as automatically generated patches, personalized repair suggestions for various frameworks, and straightforward, step-by-step guidance—we can make it easier for developers to learn and significantly cut down the time it takes to resolve issues.

Cloud-based collaboration features could support multiuser vulnerability tracking, centralized reporting dashboards, and long-term security trend analysis. Bringing our system together with CI/CD pipelines would really help us automatically spot any vulnerabilities while we're deploying, which is a crucial aspect of DevSecOps. Plus, we should definitely look into some of the latest techniques. For instance, using graph neural networks to analyze code dependencies, exploring endpoints through fuzzing, and keeping an eye on behavior during runtime can significantly boost our system's capability to perform comprehensive security checks at the enterprise level.

CONCLUSION

This paper presented an AI-powered vulnerability assessment tool designed to improve secure web development practices, particularly in educational settings. The system combines static analysis, dynamic testing, WAF-aware payload adaptation, and AI-driven explanations to provide both technical vulnerability detection and educational value. We conducted some experimental evaluations on applications that were purposely designed to have vulnerabilities, as well as on projects made by students. The outcomes were quite impressive! We found that the system was really good at spotting common security issues, like SQL Injection, Cross-Site Scripting (XSS), insecure headers, and problems with input validation.

The AI explanation engine truly made a difference for students. It wasn't just about pinpointing their struggles; it also helped them dive deep into those challenges and find solutions. The WAF adaptation module further boosted detection reliability in protected environments, highlighting the practicality of hybrid scanning approaches.

All though the system does have its limitations, such as not being compatible with all frameworks and struggling a bit with complex or large-scale applications, it really lays a strong foundation for future improvements. Planned extensions include real-time feedback in IDEs, broader framework compatibility, cloud collaboration features, and advanced machine learning-based vulnerability prediction models. Overall, the proposed tool represents a significant step toward making web security education more accessible and providing intelligent, effective mechanisms for vulnerability assessment.

REFERENCES

[1] K. Mani et al., "Machine Learning Models in Web Applications,"ScienceDirect, 2025.

[2]J. R. Tadhani et al., "Securing Web Applications Against XSS and SQLi Attacks," Scientific Reports, 2024.

[3] R. Alhamyani et al., "ML-Based Detection of XSS Attacks," Informa- tion, 2024.

[4] F. Younas et al., "AI-Based Early Detection of XSS," Computers & Security, 2024.

[5] K. H. Hamzah et al., "ML Algorithm Comparison for XSS Detection," JOIV, 2024.

[6] B. Njie, "Machine Learning for XSS Detection," MSc Thesis, 2024.

[7] S. Nalamwar et al., "Web-Based Vulnerability Scanner for Education," IRJMETS, 2025.

[8] D. Dumaniya and Y. Makawana, "Automated Web Vulnerability Scan- ner," IJCT, 2025.

[9] A. Thompson and K. Lee, "ML Framework for Web Vulnerability Detection," IEEE, 2024.

[10] B. E. Oduleye, "ML for Web Vulnerability Prevention," IJET, 2025.

[11] M. Sharma and R. Gupta, "Enhancing Web Application Security with WVS," IJERT, 2024.

[12] N. S. Chahal et al., "Security Assessment with Orchestration Platforms," Computers & Security, 2022.