International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Automated Web Vulnerability Detection and Reporting Engine

CH. Narendra Kumar[1], SK. Riyan Azeem[2], Sk. Abdul Rahman[3], K. Siva Sai[4],

[1234] *B.TECH Student, Department Of Computer Science And Engineering(Cyber Security), Geethanjali Institute Of Science And Technology, Nellore, AP, India.*

Guide: Mr. V. Chaithanya, Assistant Professor,

Department Of Computer Science And Engineering (Cyber Security), Geethanjali Institute Of Science And Technology, Nellore, AP, India.

ABSTRACT

The widespread use of web applications in essential industries like banking, healthcare, e-commerce and education has created a major challenge for companies which need to protect their web applications from security threats. Manual penetration testing which belongs to traditional vulnerability assessment methods needs both high costs and extensive time commitments and becomes unmanageable when testing modern web applications which have complex security architectures. The project introduces an Automated Web Vulnerability Detection and Reporting Engine which solves the existing system limitations by providing a solution that detects and evaluates security flaws in web applications. The system automatically navigates through target websites by following links and form elements to gather input vectors which it uses to launch specific attack payloads for testing multiple security weaknesses including Cross-Site Scripting (XSS) and other typical web security issues. The engine checks application responses to find out whether vulnerabilities exist and their severity and their specific locations in the system. The scanning process produces a JSON-based report which organizes information about discovered vulnerabilities into a systematic format that includes their risk ratings and detailed descriptions and affected system parts and recommended solutions. The proposed engine uses automated processes to find vulnerabilities and create reports which decreases manual testing requirements while giving developers practical security recommendations for their applications. The lightweight tool operates without needing extra hardware resources so it becomes an ideal solution for developers and small organizations and academic institutions. The system successfully detects security vulnerabilities in testing conducted on purposely vulnerable web applications while offering effective guidance for remediation efforts. The research provides an efficient developer-friendly solution which scales better than traditional security testing methods to protect modern web applications from threats.

Key Words: Automated Web Vulnerability Scanner, SQL, Cross-Site Scripting (XSS) and Automated Security Testing and Vulnerability Detection.

I. INTRODUCTION

Businesses in diverse fields including banking, healthcare, e-commerce and educational institutions now depend on web applications to provide critical support. The increasing usage of these systems leads to rising Cyberattack threats swhich target them as potential attack points. The attacks exploit vulnerabilities which include get injected and cross-site scripting and cross-site request forgery and broken authentication. These vulnerabilities result in data leakage and financial loss and negative reputation damage and affect 1. The situation is bad because most high impact vulnerabilities occur because application input validation fails and developers create bad code and system configuration errors which become undetectable after extended application use, which has become more difficult because of modern application functionalities [3].

Traditional vulnerability assessment methods require both manual penetration testing and manual static analysis evaluation, which results in high testing costs and time requirements while making testing susceptible to human mistakes [4]. Web applications have become the global standard which disables all traditional methods from scaling, so security teams face growing challenges to maintain their defenses against new vulnerabilities, which keep appearing. The manual testing process to assess vulnerabilities mainly disregards critical vulnerabilities because the testing process handles too many attack points to manage effectively [5].

The manual testing process tests multiple systems which needs extensive human resource involvement to secure proper testing coverage, while the security vulnerability

scanning tools provide a more efficient method to handle testing requirements. Web applications use security testing tools to identify potential vulnerabilities in their systems through automatic scanning methods which create testing results. Most automated tools available today present usability challenges for non-experts, while the tools provide insufficient coverage to detect 17 A modern example Web application security Current WESA includes CSRF and advanced forms of XSS[7]. The demand for a user-friendly and efficient and affordable product for web vulnerability scanning and reporting has reached its highest point in the current market.

The solution of this research is an innovative one in the shape of an automated web vulnerability scanner. The tool provides a lightweight and user-friendly design which allows developers to intelligently crawl web applications, detect vulnerabilities across all security layers, and create reports that contain practical information. The scanner helps developers and small businesses enhance their security methods by delivering an affordable, effective solution which enables them to identify common web vulnerabilities and protect their systems against security threats.

II. PROBLEM STATEMENT

The essential nature of web applications in the current digital ecosystem creates vital operational functions which extend their use across all fields including banking and healthcare and e-commerce and additional business sectors. The cybersecurity risk to these applications increases as their operational complexity rises. Organizations and their users face financial losses and reputational damage through security breaches which occur from common weaknesses such as Cross-Site Request Forgery (CSRF) and SQL Injection (SQLi) and Cross-Site Scripting (XSS) and broken authentication mechanisms which reveal confidential information to attackers who execute these attacks [1][2]. The existing security testing tools which organizations possess do not suffice for identifying these vulnerabilities which stem from insecure design choices and inadequate input validation and configuration errors.

The manual penetration testing method has functioned as the main vulnerability detection method for web applications since its inception although this method possesses multiple drawbacks. The testing process requires security specialists who possess advanced expertise which drives up costs and prolongs testing duration while human errors during the process still lead to missed detection of security flaws [3]. The cost of manual testing prevents small businesses and organizations with limited budgets from implementing this testing method [4]. The traditional testing methods used to evaluate modern web applications face problems because their current methods cannot handle new vulnerabilities which have emerged since the existing methods were developed [5].

Automated vulnerability scanning tools have been developed to solve existing problems; however, most current solutions fail to meet minimum requirements. The security assessment tools OWASP ZAP and Burp Suite enable detection of specific vulnerability types; however, their ability to discover advanced security problems which entail advanced CSRF attacks and subtle XSS flaws remains limited. The tools demand extensive configuration work coupled with specialized security skills which creates obstacles for developers whose security understanding falls short and for small businesses [6][7]. The limited functionality of automated scanners results in their generation of security reports which lack specific guidance needed to fix the security issues detected in reports [8].

The rising dependence on web applications, combined with emerging cyber threats, demands immediate development of an automated system which can detect vulnerabilities with outstanding performance and efficient resource management and complete accessibility for users. The system needs to detect all web application vulnerabilities while delivering results which developers can use to create protective measures. This research resolves all requirements through an automated web vulnerability scanner which enables developers to quickly find web application vulnerabilities and offers them detailed information to strengthen security measures [9].

III. OBJECTIVES

The research aims to create an automatic web vulnerability scanner which will successfully identify standard security weaknesses found in web applications. These weaknesses include SQL Injection (SQLi) and Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) and broken authentication and directory traversal vulnerabilities. The security tools must provide easy deployment and usage for developers who have

different security skill levels because modern web applications become more intricate.

The research aims to design a fast, lightweight, and user-friendly scanner. The majority of vulnerability scanning tools create operational difficulties because users must configure complex settings which require security expertise. This situation restricts access to these tools for small organizations. The system design focuses on two main goals. The first goal establishes automatic vulnerability scanning access through direct deployment without the need for complex system setup.

The scanner detects vulnerabilities while producing complete structured reports which use JSON format as one of their output formats. The reports contain information about the discovered vulnerabilities which includes their severity level and the specific areas they affect and their potential damage to the system. The available actionable insights will assist developers in determining which tasks to complete first. This process will help them create better security measures for their software [1][5].

The scanner system implements both scalability and adaptability functions. These functions enable the scanner to work with multiple web technologies. The scanner system allows organizations to develop custom security measures which protect their specific web applications [6][8]. The tool decreases the need for manual penetration testing. It accelerates vulnerability discovery to help organizations find and fix their security weaknesses before they become major issues.

The research achieves its goals through the development of an effective security solution which maintains low resource requirements. This solution remains affordable and easy to use for modern web development environments [9].

IV. LITERATURE REVIEW

Web vulnerability scanners function as essential tools which help security experts find all existing security weaknesses present in web applications. The current market for vulnerability scanning tools shows strong diversity because different products deliver unique capabilities and performance characteristics plus their own sets of limitations. The section shows how different web vulnerability scanners perform by comparing their most common features with the scanner that this research introduces. The comparison includes popular open-source tools such as OWASP ZAP, Burp Suite, Nikto, as well as commercial solutions like Acunetix.

A) OWASP ZAP

The Zed Attack Proxy or OWASP ZAP functions as a penetration testing tool which security professionals use to test their web applications for security flaws. The system gives users the ability to conduct both automatic and manual scanning while providing complete security evaluation functions. The tool offers advanced expansion possibilities which allow users to create custom functions through plugins while the tool gains from active community development. The program's design presents difficulties for new users because of its advanced user interface and need for extensive system setup work. The program's advanced learning requirements make it difficult for organizations and developers with basic security knowledge to use the system effectively which leads to deployment challenges without previous experience [1].

TABLE 1: COMPARISON OF ZAP'S STRENGTHS AND WEAKNESSES.

Sr.	Pros	Cons
1	Open-source and free to use. Community support and constant updates.	Complex installation and usage, requires technical skills.
2	Full feature set, proxying, and scripting.	Mostly usable by security experts: steep learning curve for beginners.
3	Support automated scanning ang manual testing.	Slower scans compared to some commercial tools.

B) BURP SUITE

The Burp Suite system serves as a popular instrument which security professionals use to protect their web applications from security threats. The system has gained recognition as the top selection for professional penetration testers because of its precise results and broad range of functionalities. The complete Burp Suite package requires users to pay for a commercial license which costs more than its free version that offers limited advanced features. The tool

demands extensive technical knowledge for its operation, which creates a barrier that prevents non-experts and small development teams from using it effectively [2].

TABLE 2: COMPARISON OF BURP SUITE'S STRENGTHS AND WEAKNESSES.

Sr.	Pros	Cons
1	A high level of vulnerability detection accuracy and precision.	The professional version is costly.
2	Advanced features including scanning, crawling, and traffic interception.	Extremely high learning curve.
3	Extensible through plugins and custom scripts.	Requires significant configuration and technical expertise.

C) NIKTO

Nikto functions as a command-line vulnerability scanning tool which helps users identify outdated software and insecure settings and common web server security vulnerabilities. Nikto offers fast deployment yet it fails to identify advanced web application threats which include SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks. The tool's lack of graphical user interface design hinders its accessibility while making it less attractive to developers and users who possess limited security knowledge [3].

TABLE 3: COMPARISON OF NIKTO'S SUITE'S STRENGHTS AND WEAKNESSES.

Sr.	Pros	Cons
1	Simple and fast to use.	Limited coverage for modern vulnerabilities (e.g., SQLi, XSS).
2	Focuses on outdated software and misconfiguration.	Some users may struggle with the command-line interface(CLI).
3	Free and open-source.	Generates noisy results with possible false positives

D) ACUNETIX

Acunetix functions as a complete commercial web vulnerability scanning solution which identifies various security weaknesses through its ability to detect SQL Injection (SQLi) and Cross-Site Scripting (XSS) and other standard web application vulnerabilities. The system presents a user interface which both developers and security teams find appealing. Acunetix proves to be too costly for small companies and independent developers which results in reduced usage of the software. The scanning process requires a significant time investment while its results include false alarms which decrease efficiency for users who need to compare it with lighter scanning methods and specialized scanning solutions. [4]

TABLE 4: COMPARISON OF ACUNETIX'S STRENGTHS AND WEAKNESSES.

Sr.	Pros	Cons
1	Extensive scanning features and advanced detection capabilities.	Very costly, especially for small businesses.
2	User-friendly graphical interface for developers.	Can produce false positives in some cases.
3	Strong support for modern vulnerabilities.	Scanning process can be time-consuming.

E. NETSPARKER

Netsparker functions as a commercial web vulnerability scanning tool which provides high detection accuracy for security flaws in modern web applications that use complex systems. The tool is considered user-friendly and is well suited for enterprise environments. The enterprise-focused product Netsparker which operates as a paid solution provides effective results but its high licensing fees restrict user access. The absence of a complete free version throughout the market creates challenges for small organizations and individual developers who need accessible software solutions [5].

TABLE 5: COMPARISON OF NETSPARKER'S STRENGTHS AND WEAKNESSES

Sr.	Pros	Cons
1	High-quality vulnerability detection with fewer false positives.	Expensive compared to alternatives.
2	User-friendly interface.	Enterprise-oriented; not affordable for small organizations

3	Strong enterprise-level support.	Requires significant system resources for large scans

TABLE 6: COMPARATIVE ANALSIS OF WEB VULNERABILITY SCANNERS.

Tool	Strengths	Limitations	Ideal User
OWASP ZAP	Open-source, active community, extensive features	Complex setup; Difficult for beginners	Security experts and researchers
BURP SUITE	High accuracy, advanced features, well-documented	Expensive; steep learning curve for non-experts	Professional penetration testers
NIKTO	Fast, simple, free; detects server misconfigurations	Limited to outdated software.	Beginners and users needing fast scans
ACUNETIX	Comprehensive scanning, user-friendly interface,	Expensive; potential for false positives	Developers needing a user-friendly tool
NETSPARKER	Highly accurate, easy-to-use, strong enterprise-level support	Expensive; free version is very limited	Enterprise teams needing robust security tools

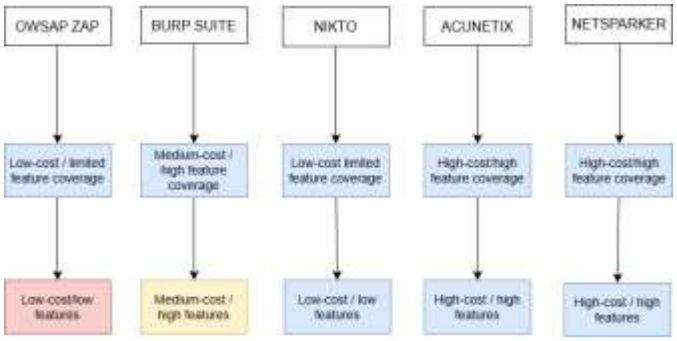

Figure 1: Cost vs. Feature Coverage Comparison of Automated Web Vulnerability Scanners

1) Cost vs. Feature Coverage Analysis

A Cost-to-Feature Ratio (CFR) needs to be established for web vulnerability scanning tools because it enables systematic and unbiased assessment of their operational costs against their functional capabilities. The metric enables quantitative assessment of each tool's value according to the findings from the literature review and the comparative analysis.

The Cost-to-Feature Ratio is expressed as: $CFR = \frac{\text{Cost}}{\text{Feature Coverage}}$

Where:

- The vulnerability scanning tool requires a financial expenditure which is referred to as its Cost.
- The Feature Coverage metric provides a quantitative assessment of the tool's detection abilities through its ability to identify multiple different types of vulnerabilities.

A lower CFR value indicates a more cost-effective tool which provides wider vulnerability detection abilities at its price point. The higher CFR value shows that the system offers minimal feature availability when compared to its cost.

V. METHODOLOGY

The automated web vulnerability scanner which the study introduces detects the most frequently occurring security weaknesses that affect web applications. The vulnerability assessment process employs a methodology that consists of several components which work together to achieve their designated assessment functions. The design implements a modular architecture which allows systems to become scalable while adapting to all web application types. The established framework supports efficient scanning operations which lead to precise vulnerability identification and organized reporting processes.

A) Scanner Workflow

The scanner workflow implements a systematic sequential design which enables effective task execution through its component operations. The general workflow of the proposed system is described below:

1. Target URL Input: The user provides the target web application URL to be scanned.

2. Session Initialization: The scanner initiates an HTTP session with the target application and configures the necessary request headers to enable proper communication.
3. Crawling: The scanner automatically traverses the web application to identify accessible links, forms, and other endpoints that may serve as potential attack vectors during the crawling and discovery process.
4. Security Testing: The testing process is executed on all discovered forms and endpoints by using specific attack payloads which detect vulnerabilities like SQL Injection (SQLi) and Cross-Site Scripting (XSS) and their associated threats.
5. Vulnerability Detection: The scanner analyzes server responses to the injected payloads and determines the presence of potential vulnerabilities by matching response patterns against known security signatures.
6. Reporting: The system produces a detailed PDF report which presents all identified vulnerabilities together with their severity levels and affected locations and descriptive details about each vulnerability after the scanning process ends.

B) Steps for Creating a Diagram

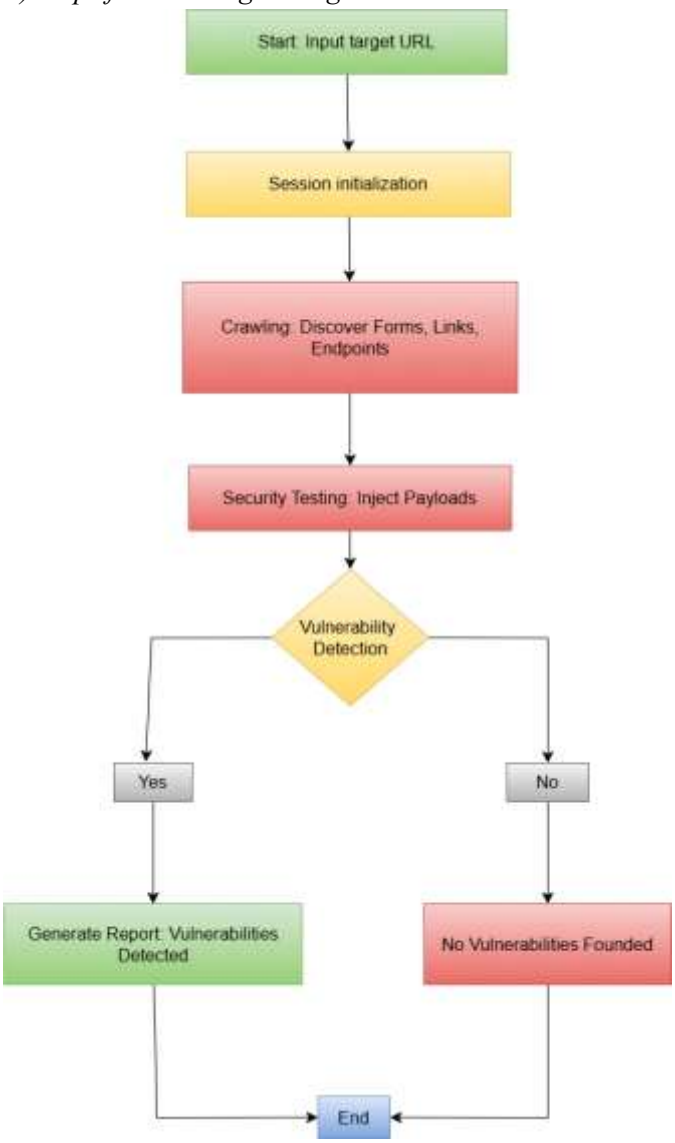

Figure 1: Scanning Process

The picture given above is expected to be a summary of the scanning done.

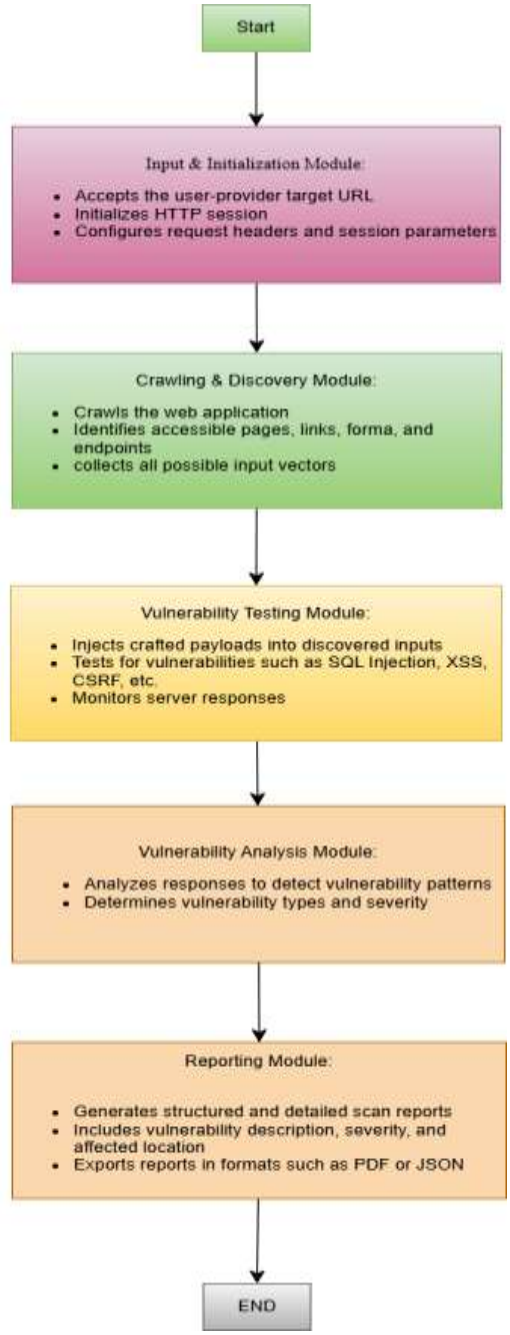

Figure 2: Web Vulnerability Scanner Workflow

The following is a Flowchart of Web Vulnerability Scanner Architecture. It starts with input and configuration of the session, followed by the endpoints scraped by the site.

Step-by-Step Breakdown of the Methodology:

1. Input Target URL: The scanning process begins when the user provides the URL of the target web application. The scanner waits in a state of inactivity until the user provides the valid target URL which starts the process of vulnerability assessment.

2. Session Initialization: The scanner starts an HTTP session after receiving the target URL to establish communication with the web application. The system prepares request headers and cookies along with session parameters which will be used to establish server connections throughout the current phase of operation.

3. Crawling: The crawler module explores all areas of the web application to find all available pages and URLs and forms and endpoints. The process of crawling enables the creation of an application structure diagram which identifies all possible input points that lead to security weaknesses in the system. The crawler collects testing points by following hyperlinks while using elements of forms to navigate through the web application.

4. Security Testing: The scanner uses customized attack payloads to test all endpoints and input fields which it has identified through its scanning process. The payloads are created to find common web security flaws which include SQL Injection (SQLi) and Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF) attacks. The scanner creates a simulation of actual attacks when it sends payloads to the system and monitors how the server reacts.

5. Vulnerability Detection: The scanner uses this stage to evaluate how the server responds to the payloads that have been inserted into the system. The scanner identifies insecure behavior through error messages and unexpected outputs and all other signs of abnormal application activity which it has found during its assessment. The process of detection requires comparing response patterns with a collection of established vulnerability signatures that have already been identified.

6. Reporting: The system creates an exhaustive vulnerability report which it presents in JSON format after completing the scanning process. The scanning report shows all detected vulnerabilities which include: The scanning report details the vulnerability type (e.g., SQLi, XSS) for every identified security issue. The report shows the severity level of each vulnerability which can be classified as high, medium, or low. The report provides an explanation of the problem. The report identifies the affected areas of the application through URLs and form fields.

7. The developers receive fast implementation of security recommendations through this structured method which helps them to fix problems quickly and to improve their security practices.

C) Modular Implementation

The proposed scanner uses a modular structure to create a system which enables easy maintenance and future enhancements and increased capacity to handle growing demands. All functional components function as separate units which make it possible to change or upgrade single modules without affecting the complete system. New attack payloads and reporting formats can be included in both the vulnerability testing and reporting modules without causing any disturbances to the other parts of the scanner.

D) Vulnerability Detection Algorithm

The process of vulnerability detection starts with the server response analysis which determines if the system has security weaknesses based on known vulnerability patterns. The mathematical formula can formalize how a scanner detects a vulnerability by comparing the server response to known attack signatures

Detection of Vulnerability

$$V = \sum_{i=1}^{n} f(\text{payload}_i, \text{response})$$

Where:

- V= Total number of vulnerabilities detected
- n= Number of payloads tested
- $f(\text{payload}_i, \text{response})$= A function that evaluates whether a server response matches predefined vulnerability patterns
- Response = The function evaluates server responses against established vulnerability patterns. The server generates output results after the payload injection process.

Explanation:

The function evaluates server responses against established vulnerability patterns. The server generates output results after the payload injection process.

E) Vulnerability Severity Calculation

The scanner uses a weighted scoring model to determine vulnerability severity levels which classify vulnerabilities according to risk.

$$S = w_1 \cdot \text{exploitability} + w_2 \cdot \text{impact} + w_3 \cdot \text{likelihood}$$

Where:

1. S= Severity score (range: 0–10)
2. Exploitability = Ease with which the vulnerability can be exploited
3. Impact = Potential damage caused by exploitation
4. Likelihood = Probability of the vulnerability being exploited
5. w_1, w_2, w_3= Weighting factors assigned based on importance

F) Conceptual Diagram Ideas

The scanner starts its operation when it gets a target URL which it uses to create a communication link. The system crawls through the web application to test its security by injecting specific payloads and examining how the server responds. The system creates complete reports which display all found vulnerabilities or show that there are no major security challenges.

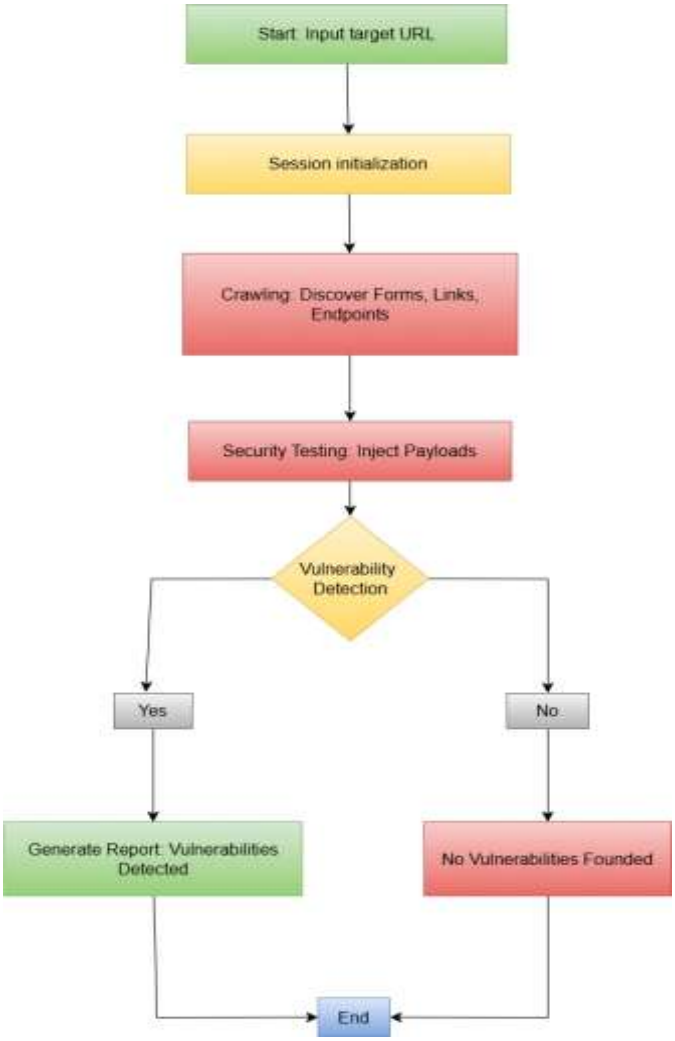

Figure 3: The Graphic of the Automated Web Vulnerability Scanner More Workflow.

VI. RESULT AND ANALYSIS

The researchers conducted laboratory tests to determine how effectively their web vulnerability scanner detected SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and other common web application security vulnerabilities. The testing occurred on applications that had been designed with security weaknesses and used both Damn Vulnerable Web Application (DVWA) and other specialized testing environments as testing platforms.

The scanner successfully detected a wide range of vulnerabilities with a high level of accuracy. The system identified SQL Injection flaws in login forms and Cross-Site Scripting (XSS) vulnerabilities in search input fields. The SQLi payloads ' OR 1=1-- successfully executed in multiple test scenarios. The system successfully detected reflected XSS vulnerabilities when attackers used malicious scripts to execute scripts through search parameters.

A) Performance Evaluation:

1. *Speed:*

The scanner operated at a much higher speed compared to manual testing methods because it completed scans within a very short period. A basic scan of a small web application consisting of 10–15 pages required only a few minutes whereas manual penetration testing of the same application would typically take several hours.

2. *Accuracy:*

The scanner demonstrated a high level of accuracy with very few false positives. Security header misconfigurations created minor issues that resulted in most false alerts. The detection rate of the system achieved better results than popular open-source tools such as Nikto and OWASP ZAP because it used more specific payloads that minimized unnecessary alerts.

3. *Scalability:*

The scanner's modular architecture enables efficient scaling to larger web applications. The system successfully detected all vulnerabilities in major application components without any performance issues when tested on applications containing 50 pages and multiple interdependent elements.

Test Case	**Target URL/Form**	**Vulnerability Detected**	**Severity**	**Descri**
SQL Injection	/login.php	SQL Injection(High)	High	SQL injecti via 'userna field(' 1=1--)

Test Case	**Target URL/Form**	**Vulnerability Detedted**	**Severity**	**Description**
Broken Auth	/login.php	Insecure login form(High)	High	Login form does not enforce HTTPS and password autocomplete is enabled
CSRF	/transfer.php	Missing CSRF Token(Medium)	Medium	Form submission lacks CSRF token
Directory Traversal	/download?file=../../	Directory Traversal(High)	High	Sensitivity files like /etc/passwd accessible via path traversal
XSS	/search.php?query=	Reflected XSS(High)	High	Malicious script payload execute in search query
Security Misconfig	/debug/	Detailed error page(Medium)	Medium	Stack trace information exposed in error response

The scanner shows effective results because it detects all major web application vulnerabilities which are most commonly found in web applications. The scanner achieves its goal to enhance web application security through its successful detection of these vulnerabilities.

B) False Positive Rate (FPR) and False Negative Rate (FNR):

The proposed vulnerability scanner accuracy evaluation process uses False Positive Rate (FPR) and False Negative Rate (FNR) as standard performance metrics. These metrics are used in automated security testing to measure how well detection algorithms work.

Formula 1: False Positive Rate (FPR)

$$\text{FPR} = \frac{FP}{FP + TN}$$

Where:

- FP (False Positives): Vulnerabilities incorrectly reported by the scanner.
- TN (True Negatives): Secure components correctly identified as safe.

Formula 2: False Negative Rate (FNR)

$$\text{FNR} = \frac{FN}{FN + TP}$$

Where:

- FN (False Negatives): Actual vulnerabilities that were not detected by the scanner.
- TP (True Positives): Vulnerabilities correctly identified by the scanner.

VII. FUTURE ENHANCEMENTS

The current version of the proposed scanner successfully detects standard web application vulnerabilities but its performance can be improved through multiple proposed enhancements which will increase its operational capabilities and detection accuracy and user interface experience.

A) Incorporation of Machine Learning:

The application of machine learning (ML) methods serves as the most effective way to advance vulnerability detection systems which need to identify advanced unknown (zero-day) security threats. ML-based anomaly detection models require training to develop skills for identifying abnormal behaviour patterns which do not correspond with established signatures or known attack payloads. The models learn to recognize hidden security weaknesses through their analysis of historical scanning data which includes both visible and undetectable security flaws that existing rule-based detection systems fail to identify [13].

Machine learning enables substantial reductions in false positive rates through its capacity to implement intelligent response classification. The scanner uses supervised learning techniques to study prior scan outcomes which enable it to identify valid vulnerabilities and harmless server response anomalies. The model develops its ability to identify new attack methods through continuous learning which leads to better detection accuracy.

B) Continuous Monitoring and Cloud Integration:

The enhancement process needs to start with better cloud system compatibility. The scanner must achieve cloud infrastructure compatibility which enables seamless operation with both cloud environments and Infrastructure-as-Code (IAC) deployment processes because web applications are now shifting towards cloud-based systems. The development process will benefit from this enhancement which delivers continuous vulnerability scanning via its capacity to operate in real-time throughout the entire development cycle. The dynamic IP allocation and auto-scaling architectures used by cloud platforms such as AWS and Azure and Google Cloud create obstacles for conventional vulnerability assessment methods. The introduction of cloud-aware scanning modules enables the scanner to modify its operation according to the various changes in the dynamic environments [14].

The scanner needs to become part of Continuous Integration and Continuous Deployment (CI/CD) pipelines which will create a system for automatic vulnerability scanning that activates with every code update. The system detects security weaknesses at an early development stage and it maintains security

monitoring for production systems throughout their operational period.

C) Interactive Dashboards and Visualization:

An interactive dashboard will provide visual tools for displaying scan results which will enhance the software's accessibility and usability. Developers have to manually interpret static output formats of scan results which currently exist in JSON format. The interactive visualization interface enables users to examine vulnerabilities through graphical representation while they can use filtering tools to sort results according to severity levels and determine which issues should be fixed first. The researchers can use heat maps and severity charts and vulnerability trend graphs to display vulnerabilities which occur frequently and have a major impact on their work.

D) Expanded Vulnerability Coverage:

The scanner already detects most web application vulnerabilities but it needs new capabilities to identify more complex and emerging threats. The future versions will introduce detection capabilities which can identify Server-Side Request Forgery (SSRF) attacks and insecure deserialization activities and security weaknesses which affect specific APIs through improper authentication methods and endpoint configuration mistakes. The introduction of a dedicated API vulnerability scanning module will enable the scanner to become more valuable for current web applications because REST and GraphQL APIs have become essential components of modern web architecture [15].

VIII. CONCLUSION

The research study introduces an automated web vulnerability scanner which scientists developed to safeguard web applications against rising security threats. The tool identifies both common and high-risk vulnerabilities while executing security assessments which include checking for SQL Injection (SQLi) and Cross-Site Scripting (XSS) and finding configuration mistakes and other common security dangers. The scanner uses risk-based assessment to identify web application vulnerabilities which matter most and could cause the most damage.

The scanner demonstrates its ability to detect vulnerabilities through experimental testing under controlled conditions because it can identify vulnerabilities with both speed and accuracy. The automated testing method proved to be superior to manual testing because it delivered better results while saving time and reducing the likelihood of human mistakes. The scanner functions through a modular system which allows it to expand to handle various web application needs from simple websites to advanced systems.

The scanner produces comprehensive reports which contain JSON formatted data to give developers precise information about the vulnerabilities which have been found. The organized reporting system speeds up the process of fixing security problems while it helps security teams make better security choices. The scanner achieved strong performance results which confirmed its value to developers and security experts after detecting a few false positives during minor configuration fault cases.

The system will undergo future development which will involve adding machine learning capabilities to enhance its ability to detect vulnerabilities together with new cloud-based features which will allow continuous security monitoring across modern development platforms. The enhanced detection capabilities of the scanner will benefit from these improvements while the scanner will also become useful for future web technology and dynamic infrastructure detection.

The research study presents a web vulnerability scanning solution which delivers powerful performance at an affordable price and offers scalable capabilities which will help protect web applications from advanced cyber threats.

REFERENCES

1. Mohaidat, A. I., & Al-Helali, A. (2024). Web vulnerability scanning tools: A comprehensive overview, selection guidance, and cybersecurity recommendations. International Journal of Research Studies in Computer Science and Engineering, 10(1), 8–15.
2. Bazzoli, E., Criscione, C., Maggi, F., & Zanero, S. (2014). XSS Peeker: A systematic analysis of cross-site scripting vulnerability scanners. Politecnico di Milano. arXiv:1410.4207.

3. Rajan, A., & Erturk, E. (2017). Web vulnerability scanners: A case study. Eastern Institute of Technology. arXiv:1706.08017.
4. Shamunesh, P., Vinoth, S., & Srinivas, L. N. B. (2023). Cyber Check – OSINT & web vulnerability scanner. In Proceedings of the Second International Conference on Edge Computing and Applications (ICECAA 2023).
5. Al Anhar, A., & Suryanto, Y. (2021). Evaluation of web application vulnerability scanners for modern web applications. In Proceedings of the International Conference on Artificial Intelligence and Computer Science Technology (ICAICST 2021).
6. Ibrahim, R. Y., & Rosli, M. M. (2023). Evaluation of web application vulnerability scanners using SQL injection attacks. In Proceedings of the 8th IEEE International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE 2023).
7. Sandberg, M., & Gunnarsson, E. (2024). Web vulnerability scanner: Cybersecurity (Bachelor's thesis). KTH Royal Institute of Technology.
8. Sarpong, P. A., Larbi, L. S., Korsah, D. P., Abdulai, I. B., Amankwah, R., & Amponsah, A. (2021). Performance evaluation of open-source web application vulnerability scanners based on OWASP benchmark. International Journal of Computer Applications, 174(18), 15–22.
9. Sharma, R., & Gupta, A. (2023). Enhancing vulnerability scanning with machine learning integration. Computers & Security, 98, 102–114.
10. Lee, S., & Kim, J. (2022). A comparative study of web vulnerability scanners for identifying XSS and SQL Injection. Journal of Information Security, 18(2), 115–130.
11. Dr. Venkatesh Shankar, Ramprasad Kulkarni, Shridhar Bhovi, Varun Chiniwalar, Chinmay Betageri, (2025). Ai-Based Vulnerability Scanner for Detecting Common Website Security. (IJSREM 2025)