Guarding the Gateways: Web Vulnerability Detection

1st Vibiksha Bharathi P K
(Department of Computer Technology)
(Madras Institute of Technology)
pkvibiksha@gmail.com

2nd Jincy J S
(Department of Computer Technology)
(Madras Institute of Technology)
jincyjs2003@gmail.com

3rd Priyasahaana M
(Department of Computer Technology)
(Madras Institute of Technology)
mspd3847@gmail.com

4th Nandhini V M
(Department of Computer Technology)
(Madras Institute of Technology)
nandhusuba621@gmail.com

***Abstract*—This paper presents an automated vulnerability detection tool for web applications, which aims to identify and mitigate common security issues, including SQL injection, XXE (XML External Entity) attacks, XSS (Cross-Site Scripting) attacks, and SSRF (Server Side Request Forgery) attacks. The tool leverages the 'requests' and 'BeautifulSoup' libraries to analyze web pages and assess potential vulnerabilities. Key components of the tool include form analysis for SQL injection vulnerability detection, testing for XXE vulnerabilities by sending XML payloads, identification of XSS vulnerabilities through payload injection, and SSRF vulnerability detection based on HTTP response status codes. The automated tests streamline the identification and mitigation of web application security issues. The tool can be customized and extended to enhance the security posture of web applications. This paper provides an overview of the tool's functionality and its potential to enhance web application security.**

***Index Terms*—Automated Vulnerability Detection, Web Application Security, SQL Injection, XXE Attack, XSS Attack, SSRF Attack, Security Assessment.**

I. Introduction

In today's digital landscape, web applications are integral to our online experiences, from e-commerce and social media platforms to enterprise systems. While they offer convenience and functionality, web applications are susceptible to a wide range of security vulnerabilities. Addressing these vulnerabilities is of paramount importance, as they can lead to data breaches, unauthorized access, and other security incidents.

To bolster the security of web applications, security professionals and developers employ various techniques and tools to identify and mitigate vulnerabilities. This paper introduces an automated vulnerability detection tool designed to streamline the process of identifying and addressing common web application security issues. The tool covers a spectrum of vulnerabilities, including SQL injection, XXE (XML External Entity) attacks, XSS (Cross-Site Scripting) attacks, and SSRF (Server Side Request Forgery) attacks.

The automated nature of the tool is a significant advantage in the context of rapidly evolving web applications. Manual testing for vulnerabilities can be time-consuming and error-prone, and it may not keep pace with the continuous deployment of web services. By automating the detection process, security professionals and developers can more efficiently identify and mitigate security risks.

This paper presents the tool's architecture and functionality, emphasizing its ability to automate testing procedures, provide rapid results, and enhance the overall security posture of web applications. It also highlights the key components of the tool, including form analysis, payload injection, and HTTP response analysis, which are crucial for identifying vulnerabilities. The tool is versatile and can be customized to cater to specific web application needs, offering a valuable resource for security professionals and developers seeking to secure their web applications effectively.

The remainder of this document will delve into the details of the tool's design, usage, and results, providing a comprehensive guide for leveraging this automated solution to bolster web application security.

II. Related Work

In recent years, web application security has been increasingly threatened by SQL injection attacks, a top concern per OWASP. These attacks continuously evolve, posing risks like data breaches and site disruptions. A novel approach presented in [1] offers an adaptive deep forest-based method to detect complex SQL injection attacks. It optimizes the model by enhancing deep forest structures and introducing an AdaBoost-based deep forest model, allowing dynamic feature weight assignment during training. Experimental results show that this approach outperforms traditional machine learning and deep learning methods, marking a significant advance in web security research.

In their study, H.-C. Chen, A. Nshimiyimana, C. Damarjati, and P.-H. Chang [2] address the persistent threat of Cross-Site Scripting (XSS) attacks on web applications. They employ a multi-pronged approach, utilizing machine learning

algorithms for XSS detection and classification. Additionally, they introduce Content Security Policy (CSP) for real-time detection, reporting promising results. Their novel approach, WAIDPFS, combines Web Application Firewall (WAF), Intrusion Detection System (IDS), and Intrusion Prevention System (IPS) to enable real-time detection and prevention of XSS attacks. Experimentation reveals the effectiveness of AI algorithms, with WAIDPFS emerging as a robust solution for enhancing web application security and privacy in the face of XSS threats.

In their study, Shahid et al. [3] address the critical issue of XML eXternal Entity (XXE) injection, a well-recognized vulnerability that poses a significant threat to web security. While much of the existing literature has focused on vulnerability testing for XXE, this research stands out for its emphasis on prevention strategies. The study involves the development of a vulnerable web application using a standard XML parser from the Java Development Kit (JDK) to execute and detect various XXE attacks. Furthermore, it offers solutions for preventing XXE attacks, thereby enhancing web application security. The research provides valuable insights into misconfigured XML parsers' vulnerabilities and effective measures to secure websites from XXE attacks. It also delves into the tools used for creating platforms for XXE attacks and methods to eliminate vulnerabilities. This study is a significant contribution to the field of web security, particularly in the context of XXE prevention and mitigation.

In the research, Alghawazi, Alghazzawi, and Alarifi [4] address SQL Injection, a prevalent cyberattack that jeopardizes web application security. They focus on using machine learning to detect SQL Injection, achieving an impressive accuracy of up to 99.6 percent with various models. Their work provides insights into safeguarding web applications from this threat, with Logistic Regression emerging as the top-performing model. This study is a valuable contribution to the field of SQL Injection detection and prevention.

In their research, Al-talak and Abbass [5] delve into the critical domain of Server-Side Request Forgery (SSRF) attacks, a security vulnerability that poses a significant threat to web applications. They explore various SSRF attack types, the importance of securing web applications, and the use of machine learning techniques for detection and mitigation. A key focus of their work is the application of deep learning techniques, specifically Long Short-Term Memory (LSTM) networks, to create an intelligent model capable of identifying SSRF attacks. The deep learning model they developed demonstrated an impressive accuracy rate of 96 percent, highlighting its effectiveness in SSRF detection. This research offers valuable insights into safeguarding web applications against SSRF vulnerabilities and showcases the potential of deep learning in enhancing security.

Banerjee et al. [6] address the critical concern of Cross-Site Scripting (XSS) attacks in web applications, a pervasive threat to cybersecurity. Their research employs machine learning classifiers to detect XSS attacks, focusing on two key features: URLs and JavaScript. Four machine learning algorithms, including SVM, KNN, Random Forest, and Logistic Regression, are utilized to categorize webpages as either malicious or benign. After thorough dataset analysis, the Random Forest Classifier emerges as the most accurate, boasting a low False Positive Rate of 0.34. This precision enhances the effectiveness of XSS threat evaluation, contributing to the seamless operation of web systems. Their work offers valuable insights into safeguarding web applications from XSS attacks, emphasizing the role of machine learning in enhancing security.

Kumar and Ponsam [7] delve into the prevalent and critical issue of Cross-Site Scripting (XSS) attacks in web applications, a pervasive cybersecurity concern. Their research focuses on the detection of XSS vulnerabilities, employing an array of deep learning and machine learning models. These models encompass diverse algorithms, including Long Short Term Memory (LSTM), Convolution Neural Networks (CNN), AdaBoost, Gradient Boosting, Logistic Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Random Forest (RF), Naïve Bayes (NB), and Decision Tree (DT). By conducting experiments on a dataset of real-world XSS attacks and non-attack web requests, their machine learning model demonstrates a high degree of accuracy in identifying XSS attacks, outperforming several baseline approaches. This study underscores the potential of machine learning to effectively detect XSS attacks, thereby enhancing web application security.

Li et al. [8] focus on countering SQL injection attacks, a top web security threat according to OWASP. Their adaptive deep forest-based method offers an innovative solution to the challenges posed by diverse attack loads and methods. By optimizing the deep forest structure and introducing an AdaBoost algorithm, they effectively handle multi-dimensional features and prevent overfitting. Experimental results highlight their method's superiority over traditional approaches. This research contributes to the evolving field of web security, providing a robust defense against complex SQL injection attacks.

They provide an overview of the SQL injection attack and identify the various attack sources, targets, and types, as well as classify the newly proposed detection and prevention solutions. They discuss and categorize the most interesting and recent proposed mitigation solutions, especially those based on ontology and machine learning[9].

Garima Singh et al. propose a m machine learning algorithm that detects not only SQL injection attacks but also unauthorised users by keeping an audit record[10].

They suggest a heuristic algorithm based on machine learning to avoid SQL injection attacks. To train and evaluate machine learning classifiers, we use a dataset of different SQL statements[11]. Pattern matching, which can be implemented using the machine learning paradigm, is capable of mitigating SQLIA requests via login, URL, and search. If required precautions are not taken, machine learning algorithm selection for model building may be arbitrary and biased[12].

They show how Autoencoders can be used to detect FDI attacks in a novel way. We take advantage of sensor data similarity in time and space, which can assist in the identification of falsified data. Furthermore, using the denoising audio encoder, the falsified data is cleaned[13].

Machine learning learns from previous attacks and blocks or bypasses the Web Application Firewall based on the previous performance. The paper focuses on SQL Injections and Phishing flaws, and how to prevent attackers from easily deceiving users[14].

III. Proposed Work

The proposed work outlines the development and implementation of an automated vulnerability detection tool for web applications. This tool is designed to identify and mitigate commonly encountered security vulnerabilities in web applications, with a primary focus on the following four critical areas: SQL injection, XXE (XML External Entity) attacks, XSS (Cross-Site Scripting) attacks, and SSRF (Server Side Request Forgery) attacks.

A. Automated Vulnerability Detection Framework The core of the proposed work involves the creation of an automated framework for web application vulnerability detection. This framework will incorporate various techniques and libraries for web scraping, form analysis, and HTTP communication. The automation of these processes is essential for the efficient analysis of web applications and the identification of potential security risks.

B. SQL Injection Vulnerability Detection SQL injection remains a prevalent security concern in web applications. The tool will implement a form analysis module to identify input fields susceptible to SQL injection. Subsequently, it will construct and send payloads to these fields to evaluate the application's response. Potential SQL injection vulnerabilities will be flagged based on response indicators.

C. XXE Attack Detection The tool will include a module for testing XXE (XML External Entity) attacks. It will send crafted XML payloads to web services and assess responses for signs of XXE vulnerabilities. Particular attention will be given to the identification of unexpected data leakage and system file access.

D. XSS Vulnerability Detection Cross-Site Scripting (XSS) attacks pose a significant threat to web applications. The tool will perform parameterized testing by injecting malicious payloads into web parameters and checking for their presence

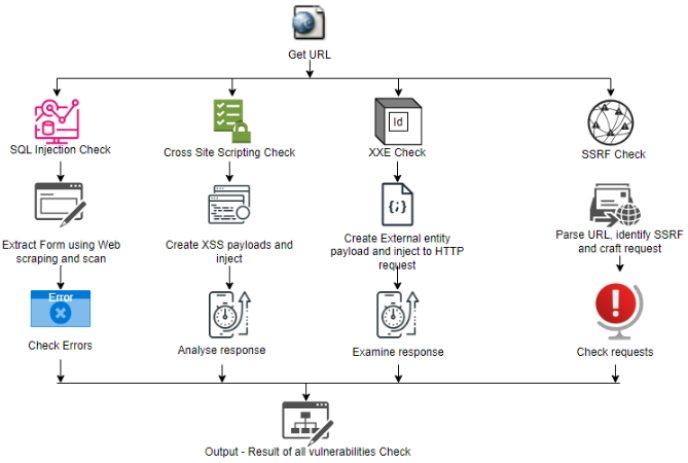

Fig. 1. WEB VULNERABILITY DETECTION

in the responses. Detected XSS vulnerabilities will be reported for further examination and mitigation.

E. SSRF Vulnerability Detection Server Side Request Forgery (SSRF) vulnerabilities will be detected through the analysis of HTTP response status codes. The tool will scrutinize responses and alert users to any SSRF vulnerabilities, thus preventing unauthorized requests to internal resources.

F. Customization and Reporting To accommodate the unique requirements of diverse web applications, the tool will be designed with flexibility in mind. Users will have the ability to customize payloads, adjust analysis parameters, and generate detailed vulnerability reports tailored to their specific applications.

G. Documentation and User Guide Comprehensive documentation will accompany the tool, offering clear instructions on its effective usage. The user guide will provide detailed explanations of the tool's features, customization options, and guidance on interpreting vulnerability reports.

The development and implementation of this automated vulnerability detection tool aim to offer a valuable resource to security professionals and developers tasked with fortifying the security of web applications. The automated nature of the tool will enable swift and systematic testing, thus facilitating the proactive identification and mitigation of security vulnerabilities.

The subsequent sections of this document will provide an in-depth exploration of the tool's architecture, functionality, and practical use cases, underscoring its potential to enhance web application security.

IV. Algorithm: Automated Web Vulnerability Detection

A. Input and Initialization

1) Accept website URL as input and fetch HTML content of given URL.
2) Parse HTML content using to extract form fields, input elements, and URLs.

B. SQL Injection Detection

1) Identify form fields and query parameters.
2) Inject SQL payloads into form fields and query parameters.
3) Analyze responses for error messages or delays indicating successful injection.

C. XXE Detection

1) Identify XML input fields.
2) Inject XXE payloads into XML input fields.
3) Analyze responses for error messages indicating XXE vulnerabilities.

D. XSS Detection

1) Identify input fields reflecting user input in the HTML content.
2) Inject XSS payloads into input fields.
3) Analyze HTML content for executing injected scripts indicating XSS vulnerabilities.

E. SSRF Detection

1) Identify URLs and request parameters.
2) Inject SSRF payloads into URLs and parameters.
3) Analyze responses for unexpected behavior or interactions indicating SSRF vulnerabilities.

F. Vulnerability Detection

1) Define a function, `vulnerable(response)`, to check if the server's response contains vulnerabilities.
2) If a vulnerability is detected, flag the form as potentially vulnerable and log the details.

G. Reporting

1) Generate reports or alerts for each identified vulnerability, including the URL of the tested web application, form details, and the nature of the potential attack.

H. Evaluation

1) Evaluate the tool's accuracy by comparing detected vulnerabilities with known vulnerabilities in the test applications.
2) Measure the time taken for each detection process, including payload injection and response analysis.

I. Conclusion

1) Conclude the detection process and present the final results, including identified vulnerabilities, false positives, accuracy rates, and detection times.

V. Result and Analysis

The result and analysis section presents the findings of applying the automated Web Vulnerability Detection tool to target web applications and provides a critical assessment of its performance and effectiveness.

A. Experimental Setup

To evaluate the tool's capabilities, a series of experiments were conducted using a diverse set of web applications known to have different vulnerabilities. The following details the experimental setup:

1) **Test Environments:** A variety of web applications with known web vulnerabilities were selected as test cases. These applications represent different domains and complexities.
2) **Tool Configuration:** The Web Vulnerability detection tool was configured to target each test application. The tool's settings, including payload variations and detection criteria, were optimized for accuracy.
3) **Data Collection:** For each test, data was collected on the tool's actions, detected vulnerabilities, and false positives, if any. This data includes the URL of the tested web application, form details, and detection results.

B. Detection Results

The results of the experiments are presented in this subsection. They include:

1) Identified Vulnerabilities: The tool successfully identified different web vulnerabilities in several web applications. Each detected vulnerability is detailed, including the specific form and payload that triggered the detection.

2) False Positives: Occurrences of false positives, where the tool incorrectly flagged a form as vulnerable, are documented. These cases are analyzed to determine the reasons behind false positives and potential areas for improvement.

C. Performance Evaluation

The performance evaluation section assesses the effectiveness and efficiency of the Web Vulnerability detection tool:

1) Accuracy: The tool's accuracy in identifying true web vulnerabilities is measured. The true positive rate and false positive rate are calculated and discussed.

2) Detection Time: The time taken by the tool to identify vulnerabilities is analyzed. This includes both the scanning time and the time required for payload injection.

D. Discussion

The discussion section provides a deeper analysis of the results and their implications:

1) Effectiveness of Payloads: The effectiveness of different payload variations is examined. Insights are provided into which types of payloads were most successful in triggering vulnerability alerts.

2) Challenges and Limitations: Challenges faced during the experiments, such as evasive techniques used by web applications, are discussed. The limitations of the tool and areas for future improvement are highlighted.

3) Real-World Applicability: The practical applicability of the tool in real-world scenarios is considered. Its potential impact on web application security and the prevention of web attacks is discussed.

E. Conclusion

The result and analysis section concludes by summarizing the key findings of the experiments, evaluating the tool's performance, and discussing its significance in enhancing web security through vulnerability detection.

REFERENCES

[1] Q. Li, W. Li, J. Wang and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," in IEEE Access, vol. 7, pp. 145385-145394, 2019, doi: 10.1109/ACCESS.2019.2944951.

[2] H. -C. Chen, A. Nshimiyimana, C. Damarjati and P. -H. Chang, "Detection and Prevention of Cross-site Scripting Attack with Combined Approaches," 2021 International Conference on Electronics, Information, and Communication (ICEIC), Jeju, Korea (South), 2021, pp. 1-4, doi: 10.1109/ICEIC51217.2021.9369796.

[3] R. Shahid, S. N. K. Marwat, A. Al-Fuqaha and G. B. Brahim, "A Study of XXE Attacks Prevention Using XML Parser Configuration," 2022 14th International Conference on Computational Intelligence and Communication Networks (CICN), Al-Khobar, Saudi Arabia, 2022, pp. 830-835, doi: 10.1109/CICN56167.2022.10008276.

[4] M. Alghawazi, D. Alghazzawi, and S. Alarifi, "Detection of SQL Injection Attack Using Machine Learning Techniques: A Systematic Literature Review," Journal of Cybersecurity and Privacy, vol. 2, no. 4, pp. 764–777, Sep. 2022, doi: 10.3390/jcp2040039.

[5] Al-talak, Khadejah, and Onytra Abbass. "Detecting server-side request forgery (SSRF) attack by using deep learning techniques." International Journal of Advanced Computer Science and Applications 12.12 (2021).

[6] R. Banerjee, A. Baksi, N. Singh and S. K. Bishnu, "Detection of XSS in web applications using Machine Learning Classifiers," 2020 4th International Conference on Electronics, Materials Engineering Nano-Technology (IEMENTech), Kolkata, India, 2020, pp. 1-5, doi: 10.1109/IEMENTech51367.2020.9270052.

[7] J. Harish Kumar and J. J Godwin Ponsam, "Cross Site Scripting (XSS) vulnerability detection using Machine Learning and Statistical Analysis," 2023 International Conference on Computer Communication and Informatics (ICCCI), Coimbatore, India, 2023, pp. 1-9, doi: 10.1109/ICCCI56745.2023.10128470.

[8] Q. Li, W. Li, J. Wang and M. Cheng, "A SQL Injection Detection Method Based on Adaptive Deep Forest," in IEEE Access, vol. 7, pp. 145385-145394, 2019, doi: 10.1109/ACCESS.2019.2944951.

[9] nes Jemal, Omar Cheikhrouhou, Habib Hamam and Adel Mahfoudhi," SQL Injection Attack Detection and Prevention Techniques Using Machine Learning", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 15, Number 6 (2020) pp. 569-580.

[10] Musaab Hasan ; Zayed Balbahaith ; Mohammed Tarique," Detection of SQL Injection Attacks: A Machine Learning Approach,International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 2019, pp. 1-6.

[11] Musaab Hasan ; Zayed Balbahaith ; Mohammed Tarique," Detection of SQL Injection Attacks: A Machine Learning Approach,International Conference on Electrical and Computing Technologies and Applications (ICECTA), Ras Al Khaimah, United Arab Emirates, 2019, pp. 1-6.

[12] Akinsola, Jide E. T, Awodele, Oludele and 3 Idowu, Sunday A," SQL Injection Attacks Predictive Analytics Using Supervised Machine Learning Techniques", International Journal of Computer Applications Technology and Research Volume 9–Issue 04, 139- 149, 2020.,

[13] Mariam M. N. Aboelwafa, Karim G. Seddik, Mohamed H. Eldefrawy, Yasser Gadallah, and Mikael Gidlund, A Machine Learning-Based Technique for False Data Injection Attacks Detection in Industrial IoT", IEEE,2020.

[14] J.Jagadessan, Akshat Shrivastava, Arman Ansari, Laxmi Kanta Kar, Mukul Kumar," Detection and Prevention Approach to SQLi and Phishing Attack using Machine Learning", International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-8, Issue-A, April 2019