Evaluating Web Application Vulnerability Scanners: Introducing the RD-Score for Comprehensive Performance Assessment

Rand Deeb

***Abstract*— Web application security is a critical aspect of modern internet services, as vulnerabilities can lead to data breaches, financial loss, and reputational damages. This study evaluates four prominent web application security tools—OWASP ZAP, BurpSuite Pro, Vega, and Wapiti—using the Damn Vulnerable Web Application (DVWA) as a testbed. We introduce a novel metric, the RD-Score, which combines detection accuracy and resource efficiency to provide a comprehensive assessment of each tool's performance. Our evaluation considers the number of HTTP requests sent during the scanning process, a crucial factor impacting scan duration, resource consumption, and network load. By normalizing HTTP requests and integrating them with the F1 Score, the RD-Score offers a balanced measure of algorithmic efficiency and detection capability. The results indicate that BurpSuite Pro achieves the highest average RD-Score, demonstrating superior balance between accuracy and resource usage, followed by Vega, OWASP ZAP, and Wapiti. This study highlights the importance of considering both detection accuracy and resource efficiency in the selection of web application security tools. The proposed RD-Score provides a robust metric for evaluating these tools, offering valuable insights for optimizing web application security. Future work should extend this evaluation to a broader range of tools and vulnerabilities, and explore real-world scenario testing to enhance the applicability of the findings.**

***Keywords*— Evaluation, OWASP, RD-Score, Web Application Vulnerability Scanners (WAVS), Web vulnerability.**

I. Introduction

Web applications are integral to modern internet-based services, ranging from online banking to e-commerce and social media platforms. Ensuring the security of these applications is paramount, as vulnerabilities can lead to data breaches, financial loss, and damage to an organization's reputation. Security vulnerability assessment tools, such as OWASP ZAP, Burp Suite, Wapiti and Vega, play a critical role in identifying and mitigating these risks.

While many studies focus on the accuracy of these tools in terms of false positives (FP) and true positives (TP), there is a significant aspect that often goes underexplored: the volume of HTTP requests sent during the scanning process. The number of HTTP requests can influence the efficiency and effectiveness of a tool, impacting scan time, resource utilization, and network load. Understanding this aspect is crucial for optimizing the use of these tools in various scenarios.

This study aims to investigate how contemporary tools used for assessing security vulnerabilities in web applications, such as Burp Suite, Wapiti and Vega, perform under varying volumes of HTTP requests.

II. Background

A. Web Vulnerability

Web application vulnerabilities are security flaws that can be exploited by attackers, compromising the application's security and granting unauthorized access. These vulnerabilities often arise from inadequate input validation, misconfigured web servers, and design flaws. They can lead to various malicious actions, such as unauthorized data access, code execution, or complete system takeover.

Key types of web vulnerabilities include:

- SQL Injection (SQLi): Allows attackers to execute arbitrary SQL code on a database, potentially gaining unauthorized access to sensitive data.
- Local File Inclusion (LFI): Exploits vulnerabilities to include files present on the target server, that can lead to information disclosure or remote code execution.
- Remote File Inclusion (RFI): Allows attackers to include and execute files from a remote server, often leading to remote code execution.
- Remote Code Execution (RCE): Enables attackers to execute arbitrary code on the target server, potentially gaining complete control over the application and underlying system.
- Cross-Site Scripting (XSS): Enables attackers to inject malicious scripts into web pages viewed by other users, that can lead to data theft, session hijacking, and other malicious activities

B. Black-Box Web Application Testing

Black-box testing evaluates web applications from an outsider's perspective without access to internal workings. This approach helps identify vulnerabilities by mimicking external attacker actions.

Active Scanners: Actively interact with the web application by sending payloads to detect vulnerabilities such as SQLi, XSS, LFI, RFI, and RCE. Tools like OWASP ZAP, Burp Suite, and Acunetix are prominent examples. They are thorough but can be time-consuming and generate high network traffic.
Passive Scanners: Monitor traffic between the client and server without injecting payloads. They analyze data flows for anomalies but might miss certain vulnerabilities.

C. *Black-Box Web Vulnerability Scanning Tools*

Web application security tools are essential black-box scanning tools for detecting and mitigating vulnerabilities in web applications. The tools covered in this study include:

- OWASP ZAP (Zed Attack Proxy): An open-source tool known for its ease of use and comprehensive scanning capabilities. It is widely adopted due to its integration with CI/CD pipelines.
- Burp Suite Pro: A popular commercial tool that offers extensive features for security testing, including a powerful proxy, scanner, and intruder module.
- Vega: An open-source web security scanner and testing platform to test the security of web applications.
- Wapiti: An open-source web application vulnerability scanner that allows you to audit the security of your web applications.

III. RELATED WORK

Ramadan Yacin Ibrahim and Marshima Mohd Rosli [1] assessed web application vulnerability scanners using SQL injection attacks , focusing on SQLMap, OWASP ZAP, and Skipfish. Their study compared the accuracy and response time of these scanners in detecting SQL injection vulnerabilities on predefined web applications, such as Damn Vulnerable Web App (DVWA). The results indicated that OWASP ZAP outperformed the other tools in terms of accuracy and performance, highlighting the need for continuous improvement of web application security scanners.

In a study by Karthik Anagandula and Pavol Zavarsky, four black-box scanners [2] (OWASP ZAP, BurpSuite Professional, Wapiti, and Nessus) were evaluated for their effectiveness in detecting stored XSS and SQL injection vulnerabilities. Using two testbeds, WackoPicko and Scanit, the research highlighted the need for improvements in attack vector insertion and multi-step attack detection. The study concluded that both commercial and open-source scanners need better functionality to effectively detect stored XSS and SQLI vulnerabilities.

Shafi Alassmi, Pavol Zavarsky, Dale Lindskog and Ron Ruhl [3] evaluated the effectiveness of black-box web application scanners in detecting stored XSS vulnerabilities using a custom testbed called SimplifiedTB. The study extended prior analyses by Bau et al. and Doupé et al. by focusing on the challenges posed by stored XSS attacks. They used scanners like Acunetix, N-Stalker, Rational AppScan, and Zed Attack Proxy (ZAP), and assessed their performance against three testbeds: PCI, WackoPicko, and SimplifiedTB. Their findings highlighted the scanners' low detection rates for stored XSS due to issues in response analysis and recommended improvements for better vulnerability detection

Yuan-Hsin Tung, Shian-Shyong Tseng, Jen-Feng Shih, and Hwai-Ling Shan [4] evaluated various vulnerability scanners with a focus on reducing redundant vulnerability alerts, which they identified as a significant issue. They introduced an advanced confusion matrix, incorporating true and false duplication metrics, to improve the evaluation accuracy of scanners. Using a testbed with web applications such as WebGoat, WordPress, and WackoPicko, they demonstrated their cost-effective evaluation approach, highlighting the importance of addressing false positives and redundant alerts in scanner assessments.

Sheetal Bairwa, Bhawna Mewara, and Jyoti Gajrani [5] conducted a comprehensive study on various vulnerability scanners to evaluate their effectiveness in detecting web application vulnerabilities. The authors used tools like Nmap, Nessus, Acunetix WVS, Nikto, and BurpSuite, assessing their performance based on detection rates of SQL Injection, XSS, and other common vulnerabilities. They emphasized the significance of using multiple scanners to cover a broader range of vulnerabilities and recommended integrating different tools for a more thorough security evaluation.

Malik Qasaimeh, Ala'a Shamlawi, and Tariq Khairallah [6] evaluated five leading web vulnerability scanners: Acunetix WVS, BurpSuite, NetSparker, Nessus, and OWASP ZAP. They assessed these tools based on their ability to detect a set of eight vulnerabilities derived from NIST and OWASP standards. The study highlighted the varying performance of scanners in terms of accuracy, false positives, and detection capabilities.

Alsaleh, Mansour, Alomar, Noura, Alshreef, Monirah, Alarifi, Abdulrahman, Al-Salman, AbdulMalik [7] performed a comparative evaluation of four web vulnerability scanners: Arachni, Wapiti, Skipfish, and two versions of Arachni. They measured the performance based on speed, crawler coverage, and detection accuracy using benchmarks like WAVSEP and Altoro Mutual test cases. The study found that Arachni 1.0.2 had the best crawling coverage, while Skipfish was the fastest scanner. They also noted significant variations in detection accuracy and recommended further research to understand these discrepancies.

S. Alazmi and D. C. De Leon [8] conducted a systematic literature review focusing on the characteristics and effectiveness of web application vulnerability scanners. They compared multiple tools including Arachni, OWASP

ZAP, Wapiti, and Skipfish, assessing their detection rates for SQL Injection and Cross-Site Scripting vulnerabilities. The review highlighted significant variability in detection rates among scanners and the need for comprehensive benchmarks to assess the scanners accurately.

Rawaa Mohammed [9] conducted an assessment of six open-source web vulnerability scanners by performing both manual and automatic testing on various testbeds. Tools like Paros Proxy, Wapiti, Skipfish, Nikto, Wfuzz, and Netsparker were evaluated for their precision, recall, and F-measure in detecting SQL Injection and Cross-Site Scripting vulnerabilities. The study found that scanner efficiency varied, with some tools performing better in certain types of attacks.

Mrs. M. Sridevi and Dr. K.V.N. Sunitha [10] reviewed common web vulnerabilities and the limitations of various web security scanners. The study highlighted the challenges of detecting vulnerabilities such as SQL Injection, XSS, and CSRF using scanners like W3AF, IronWASP, ZAP, Syhunt Dynamic, QualysGuard WAS, Wapiti, and Vega. It emphasized the need for more efficient scanners and proposed future enhancements to overcome the identified limitations.

Yuan-Hsin Tung, Shian-Shyong Tseng, Jen-Feng Shih, and Hwai-Ling Shan [11] developed W-VST, a testbed for evaluating web vulnerability scanners. They tested various tools, including OWASP ZAP and Skipfish, against W-VST to measure their effectiveness in detecting vulnerabilities. Their testbed provided a controlled environment to systematically compare different scanners' performance, focusing on metrics like detection accuracy, false positives, and scanning speed. The study emphasized the critical role of comprehensive benchmarking platforms in improving web application security by enabling detailed scanner evaluations.

IV. Proposed Metric: RD-Score

In our evaluation, we introduce a new metric, the RD-Score, designed to provide a comprehensive assessment of web application vulnerability scanners by considering both their detection accuracy and resource efficiency. The RD-Score is calculated using the formula:

$$RD\ Score = F1\ Score - 0.2 \times Normalized\ HTTP\ Requests$$

This metric integrates two crucial aspects of scanner performance:

1) Detection Accuracy: Represented by the F1 Score, which is a harmonic mean of precision and recall. The F1 Score balances the trade-off between false positives and false negatives, providing a single, cohesive measure of a scanner's accuracy in identifying true vulnerabilities without being misled by incorrect detections.
2) Resource Efficiency: Represented by the normalized number of HTTP requests. The number of HTTP requests a scanner makes during the scanning process directly impacts resource consumption, including bandwidth, server load, and scan duration. Normalizing these values ensures that the HTTP request count is scaled between 0 and 1, making it comparable to the F1 Score.

The RD-Score penalizes the F1 Score by a factor of 0.2 times the normalized HTTP requests. This weighting factor of 0.2 was carefully chosen to ensure that while resource efficiency is considered, it does not overshadow the primary goal of accurate vulnerability detection. The factor strikes a balance, acknowledging that while fewer HTTP requests are desirable to minimize resource usage, the primary function of a scanner is to detect vulnerabilities accurately.

By subtracting 0.2 times the normalized HTTP requests from the F1 Score, the RD-Score effectively highlights scanners that achieve high accuracy with lower resource consumption. This comprehensive metric allows for a more nuanced comparison of scanners, emphasizing the importance of both detection capabilities and operational efficiency. The RD-Score thus serves as a valuable tool for security professionals seeking to select the most effective and efficient web vulnerability scanners for their needs.

V. Methodology

1) Environment setup:
The first step is to initialize the environment. This involves setting up the following components:

- Testbed: we used the Damn Vulnerable Web Application (DVWA) configured at all security levels (low, medium, high),
- Operating system: the operating system used for the tools and testbed.
- Web Application Vulnerability Scanners (WAVS): we focused on OWASP ZAP, BurpSuite Pro, Vega, and Wapiti.

The architecture of the setup is illustrated in Figure 1, which includes three main components: The Web Application Vulnerability Scanners (WAVS), a proxy server (Burp Suite), and the DVWA

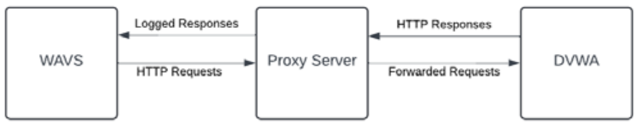

Fig 1 Architecture of the setup

2) Run the WAVs
Each security tool is executed to perform a comprehensive scan of the test web applications. This involves configuring the tools to interact with the DVWA through the proxy server, ensuring that all levels of vulnerabilities are tested. Multiple runs are conducted to account for variability and to ensure the robustness of the results.

3) Collect Data

During the scanning process, data is collected from each tool. This includes:

- WAVS Reports: Detailed reports generated by each tool, listing detected vulnerabilities and their severity levels.
- HTTP Logs: Logs of the number of HTTP requests sent by each tool during the scanning process. This helps in assessing the efficiency of the tools in terms of network traffic generated.

4) Compute Metrics

The WAVs metric is computed based on the collected data, considering factors like Precision, Recall, F1-Score, and our proposed metric (RD-Score).

VI. Results and Discussion

In this section, we present the results of our evaluation of the web application security tools (WAVS) using the proposed WAVs metric. The evaluation was performed on the Damn Vulnerable Web Application (DVWA) at three different security levels: low, medium, and high. For each level, we present the results for five types of vulnerabilities: SQL Injection (SQLi), Cross-Site Scripting (XSS), Remote Code Execution (RCE), Local File Inclusion (LFI), and Remote File Inclusion (RFI). The metrics included in the results are Precision, Recall, F1-Score, and our proposed WAVs metric, along with the counts of True Positives (TP) and False Positives (FP).

A. Low Security Level

Table 1. Summary Performance Metrics for Low Security Level

	Precision	Recall	F1-Score	RD-Score
OWASP ZAP	0.8571	0.6667	0.75	0.5844
BurpSuite Pro	1	0.7222	0.8387	0.6387
Vega	1	0.7222	0.8387	0.8373
Wapiti	1	0.5556	0.7143	0.7143

B. Medium Security Level

Table 2. Summary Performance Metrics for Medium Security Level

	Precision	Recall	F1-Score	RD-Score
OWASP ZAP	0. 6667	0.4	0.4444	0.4063
BurpSuite Pro	1	0.8	0.8	0.7514
Vega	1	0.6	0.6	0.5927
Wapiti	1	0.6	0.6	0.5962

C. High Security Level

Table 3. Summary Performance Metrics for High Security Level

	Precision	Recall	F1-Score	RD-Score
OWASP ZAP	0.222	0.2	0.2222	0.2222
BurpSuite Pro	1	0.9	0.9	0.8934
Vega	1	0.5	0.8333	0.8332
Wapiti	1	0.5	0.5556	0.3556

Table 4. Detailed Detection Results for Low Security Level

WAVS	SQLi		XSS		RCE		LFI		RFI		∑TP	∑FP	∑FN	Http Requests
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP				
OWASP Zap	3	2	6	0	1	0	1	0	1	0	12	2	6	29215
BurpSuite Pro	1	0	9	0	1	0	1	0	1	0	13	0	5	32842
Vega	3	0	8	0	0	0	1	0	1	0	13	0	5	14502
Wapiti	3	0	4	0	2	0	1	0	0	0	10	0	8	14289

Table 5. Detailed Detection Results for Medium Security Level

WAVS	SQLi		XSS		RCE		LFI		RFI		∑TP	∑FP	∑FN	Http Requests
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP				
OWASP Zap	1	3	4	0	0	0	1	0	0	0	6	3	9	28751
BurpSuite Pro	1	0	8	0	1	0	1	0	1	0	12	0	3	33817
Vega	1	0	6	0	0	0	1	0	1	0	9	0	6	15478
Wapiti	2	0	4	0	2	0	1	0	0	0	9	0	6	14522

Table 6. Detailed Detection Results for High Security Level

WAVS	SQLi		XSS		RCE		LFI		RFI		∑TP	∑FP	∑FN	Http Requests
	TP	FP	TP	FP	TP	FP	TP	FP	TP	FP				
OWASP Zap	0	5	2	2	0	0	0	0	0	0	2	7	8	12379
BurpSuite Pro	0	0	7	0	1	0	1	0	0	0	9	0	1	31409
Vega	0	0	5	0	0	0	0	0	0	0	5	0	5	12801
Wapiti	0	0	2	0	1	0	1	0	1	0	5	0	5	86418

Table [7] shows the average RD-Score across all the security levels. These average RD-Scores provide a clear picture of the overall performance of each tool.

Table 7. Average RD-Score for Each WAVS

WAVS	Average RD-Score
OWASP ZAP	(0.5844 + 0.4063 + 0.2222) / 3 = 0.4043
BurpSuite Pro	(0.6387 + 0.7514 + 0.8934) / 3 = 0.7612
Vega	(0.8373 + 0.5927 + 0.8332) / 3 = 0.7544
Wapiti	(0.7143.5962 + 0.3556) / 3 = 0.5554

VI Conclusion

In this study, we conducted a comprehensive evaluation of four widely used web application security tools—OWASP ZAP, BurpSuite Pro, Vega, and Wapiti—using the Damn Vulnerable Web Application (DVWA) as our testbed. Our evaluation introduced a novel metric, the RD-Score, which considers both detection accuracy and resource efficiency, providing a holistic measure of each tool's performance.

The RD-Score effectively integrates the F1 Score, which balances precision and recall, with the normalized number of HTTP requests to account for resource consumption. This metric highlights tools that achieve high accuracy in vulnerability detection while maintaining low resource usage. By focusing on the number of HTTP requests, the RD-Score provides a language-agnostic comparison, making it a fair and effective measure across different implementation languages. Additionally, it emphasizes algorithmic efficiency, guiding improvements for better performance in large-scale applications and resource-constrained environments.

Our results demonstrated that BurpSuite Pro achieved the highest average RD-Score across all security levels, indicating its superior balance of accuracy and efficiency. Vega followed closely, also showing strong performance. OWASP ZAP and Wapiti, while effective, scored lower on average, suggesting room for improvement in resource management without compromising detection capabilities. The study further revealed that high volumes of HTTP requests could lead to longer scan times, increased resource consumption, and network load, impacting the overall performance and efficiency of the tools.

Our findings underscore the importance of considering both detection accuracy and resource efficiency when selecting web application security tools. The RD-Score proved to be a valuable metric, providing a comprehensive assessment that helps security professionals make informed decisions.

References

[1] R. Y. Ibrahim and M. M. Rosli, "Evaluation of Web Application Vulnerability Scanners using SQL Injection Attacks," 2023 IEEE 8th International Conference on Recent Advances and Innovations in Engineering (ICRAIE), Kuala Lumpur, Malaysia, 2023, pp. 1-6, doi: 10.1109/ICRAIE59459.2023.10468295.

[2] K. Anagandula and P. Zavarsky, "An Analysis of Effectiveness of Black-Box Web Application Scanners in Detection of Stored SQL Injection and Stored XSS Vulnerabilities," 2020 3rd International Conference on Data Intelligence and Security (ICDIS), South Padre Island, TX, USA, 2020, pp. 40-48, doi: 10.1109/ICDIS50059.2020.00012.

[3] Alassmi, S., Zavarsky, P., Lindskog, D., Ruhl, R., Alasiri, A., & Alzaidi, M. (2012). An Analysis of the Effectiveness of Black-Box Web Application Scanners in Detection of Stored XSSI Vulnerabilities.

[4] Yuan-Hsin Tung, Shian-Shyong Tseng, Jen-Feng Shih and Hwai-Ling Shan, "A cost-effective approach to evaluating security vulnerability scanner," 2013 15th Asia-Pacific Network Operations and Management Symposium (APNOMS), Hiroshima, 2013, pp. 1-3

[5] Bairwa, Sheetal & Mewara, Bhawna & Gajrani, Jyoti. (2014). Vulnerability Scanners-A Proactive Approach To Assess Web Application Security. International Journal on Computational Science & Applications. 4. 10.5121/ijcsa.2014.4111.

[6] Qasaimeh, Mo'Nes & Shamlawi, A. & Khairallah, T.. (2018). Black box evaluation of web application scanners: Standards mapping approach. Journal of Theoretical and Applied Information Technology. 96. 4584-4596.

[7] Alsaleh, Mansour, Alomar, Noura, Alshreef, Monirah, Alarifi, Abdulrahman, Al-Salman, AbdulMalik, Performance-Based Comparative Assessment of Open Source Web Vulnerability Scanners, Security and Communication Networks, 2017, 6158107, 14 pages, 2017. https://doi.org/10.1155/2017/6158107

[8] S. Alazmi and D. C. De Leon, "A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners," in IEEE Access, vol. 10, pp. 33200-33219, 2022, doi: 10.1109/ACCESS.2022.3161522.

[9] Rawaa Mohammed . Assessment of Web Scanner Tools. International Journal of Computer Applications. 133, 5 (January 2016), 1-4. DOI=10.5120/ijca2016907794

[10] Sridevi, M., & Sunitha, K. (2017). A Study on Different Scanners and Their Limitations for Web Application Vulnerabilities.

[11] Y. -H. Tung, S. -S. Tseng, J. -F. Shih and H. -L. Shan, "W-VST: A Testbed for Evaluating Web Vulnerability Scanner," 2014 14th International Conference on Quality Software, Allen, TX, USA, 2014, pp. 228-233, doi: 10.1109/QSIC.2014.50.

Rand Deeb - ITMO University, Saint Petersburg, Russia (email: rand.sec96@gmail.com)