INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT (IJSREM)
VOLUME: 08 ISSUE: 04 | APRIL - 2024 SJIF RATING: 8.448 ISSN: 2582-3930

WebSec : Exploring and Modulating Vulnerabilities

Sheetal Laroiya
Chandigarh University
Mohali , India
sheetal.e15433@cumail.in

Gagan Mehta
Chandigarh University
Mohali , India
gaganharoli@gmail.com

Yuvraj Singh
Chandigarh University
Mohali , India
yuvrajsingh.ys103@gmail.com

***Abstract*—This research investigates the performance of the OWASP Zed Attack Proxy (OWASP ZAP) and Paros open-source vulnerability scanners on the Damn Vulnerable Web Application (DVWA). By evaluating their capability to identify vulnerabilities, along with assessing their user-friendliness and features, the study highlights each scanner's strengths and weaknesses. The insights aim to assist developers and security professionals in selecting the most effective tools for improving the security posture of web applications.**

***Keywords—Web Application Security, Vulnerability Scanners, OWASP ZAP, Paros, Damn Vulnerable Web Application (DVWA), Open Source Tools, Cybersecurity, Penetration Testing*.**

I. INTRODUCTION (*HEADING 1*)

In the contemporary digital landscape, web applications serve as the backbone for a wide array of services ranging from e-commerce to social networking, significantly enhancing the efficiency and accessibility of information and services. However, this increased reliance on web applications has been paralleled by a surge in cyber threats, making web application security a critical concern. As these applications often process and store sensitive data, they become prime targets for attackers seeking unauthorized access or aiming to compromise data integrity. Consequently, identifying and mitigating vulnerabilities in web applications is not just a technical challenge but a fundamental aspect of protecting user privacy and maintaining trust.

The Open Web Application Security Project (OWASP) provides a list of the top 10 security risks faced by web applications, including injection flaws, broken authentication, sensitive data exposure, and cross-site scripting (XSS), among others. These vulnerabilities represent the most critical web application security risks, and their exploitation can lead to significant breaches and data loss. As cyber threats evolve, the methods for detecting and addressing vulnerabilities in web applications must also advance. Traditional manual testing methods, while thorough, are time-consuming and often fall short in keeping pace with the rapid development and deployment cycles of modern web applications. This gap underscores the need for automated tools that can efficiently scan for vulnerabilities, allowing developers and security professionals to identify and remediate potential threats swiftly.

Automated vulnerability scanners have emerged as essential tools in the cybersecurity toolkit, offering the ability to perform comprehensive scans of web applications to detect vulnerabilities. These scanners utilize a combination of crawling and analysis techniques to simulate attacks on web applications, identifying security weaknesses that could be exploited by attackers. Among the plethora of vulnerability scanners available, open-source tools like OWASP Zed Attack Proxy (OWASP ZAP) and Paros have gained prominence for their accessibility, robust feature sets, and active communities contributing to their continuous development.

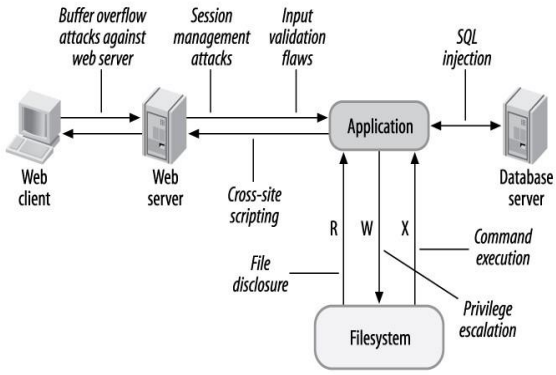


Fig-1(Architecture Diagram of Web Application Security)

OWASP ZAP is an integrated penetration testing tool for finding vulnerabilities in web applications. It is designed to be used by both those new to application security as well as professional penetration testers. It's one of the most actively maintained open-source tools in the OWASP arsenal and offers automated scanners as well as a set of tools for manual penetration testers. On the other hand, Paros Proxy is a lesser-known but still significant tool in web application security testing. Originally developed for web application security assessments, Paros has fallen behind in terms of updates and maintenance but remains a valuable tool for understanding web application vulnerabilities due to its intuitive interface and basic scanning capabilities.

The objective of this research is to conduct a thorough evaluation of these two open-source web application vulnerability scanners by testing them against the Damn Vulnerable Web Application (DVWA). DVWA is a PHP/MySQL web application that is intentionally vulnerable

and serves as an excellent resource for learning and testing the capabilities of web application vulnerability scanners. This study aims to compare OWASP ZAP and Paros in terms of their ability to detect a range of vulnerabilities, their ease of use, and the features they offer. By doing so, the research seeks to provide valuable insights into the effectiveness of these tools in enhancing the security posture of web applications.

This paper is structured as follows: Following the introduction, the literature review section provides an overview of the existing research related to web application vulnerabilities and the use of automated tools for their detection. The methodology section details the experimental setup, including the selection of the vulnerability scanners, the vulnerable web application used for testing, and the criteria for evaluating the performance of the scanners. The results and analysis section presents the findings of the study, comparing the performance of OWASP ZAP and Paros in detecting vulnerabilities in DVWA. The discussion section interprets the implications of the findings, considering the strengths and limitations of each tool and their applicability in real-world scenarios. Finally, the conclusion summarizes the key insights gained from the research and suggests directions for future studies in the field of web application security.

II. LITERATURE REVIEW

A. Overview of Web Application Vulnerabilities

Introduction to Web Security Risks: Begin by discussing the importance of web applications in daily business operations and personal use, highlighting the associated security risks. Common Vulnerabilities: Reference the OWASP Top 10 list as a foundational framework for discussing common vulnerabilities, including SQL Injection, XSS, Broken Authentication, and others.

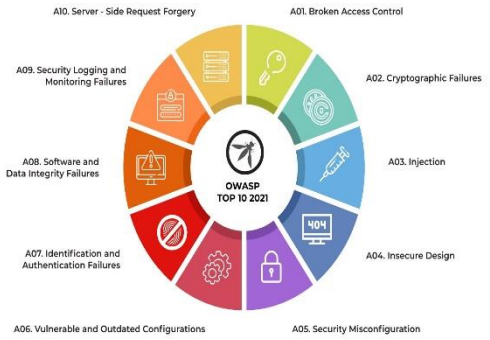


Fig-2 (OWASP Top 10 Vulnerabilities)

B. Evolution of Web Security Practices

Early Practices: Outline the initial approaches to web application security, emphasizing manual testing and code review.
Shift to Automation: Discuss the technological and methodological advancements that led to the adoption of automated vulnerability scanning tools.

C. Automated Vulnerability Scanners

Introduction to Automated Scanners: Introduce the concept of automated scanners, their purpose, and how they have become integral to modern web application security strategies.
Benefits and Limitations: Provide an analysis of the benefits, such as scalability and efficiency, and limitations, including the potential for false positives and negatives, of using automated scanners.

D. Open-Source vs. Commercial Tools

Comparative Analysis: Offer a comparison between open-source and commercial vulnerability scanning tools, discussing cost, community support, flexibility, and updates. Examples of Tools: Briefly introduce examples of both open-source (e.g., OWASP ZAP, Paros) and commercial tools, setting the stage for a deeper dive into the selected open-source tools for this study.

E. In-depth Analysis of OWASP ZAP and Paros

OWASP ZAP: Discuss the development history, key features, and typical use cases of OWASP ZAP. Highlight its position within the OWASP projects and its community-driven development. Paros: Provide background on Paros, its features, and how it has served as a foundation for other tools. Note its current status and any limitations due to lack of updates.

F. Previous Evaluations of Vulnerability Scanners

Studies on OWASP ZAP: Summarize key findings from previous research evaluating OWASP ZAP's effectiveness, usability , and detection capabilities. Studies on Paros: Do the same for Paros, noting any significant findings regarding its performance and applicability in modern web security contexts.

G. Gap in Existing Research

Identifying the Research Gap: Discuss the need for up-to-date evaluations of these tools, especially in light of evolving web technologies and security threats. Rationale for Current Study: Explain how this research aims to fill the identified gap by providing a comparative analysis of OWASP ZAP and Paros against contemporary web application vulnerabilities.

H. Theoretical Framework

Security Testing Theories: Briefly introduce the theoretical underpinnings of security testing, including black-box, white-box, and grey-box testing methods. Application to Vulnerability Scanners: Discuss how these theories apply to the use of automated vulnerability scanners in identifying potential security issues in web applications.

Conclusion of Literature Review Summary of Key Points: Recap the major themes discussed in the literature review, emphasizing the evolution of web application security practices and the role of automated vulnerability scanners. Transition to Methodology: Conclude by stating how the literature review sets the stage for the research methodology, specifically the evaluation of OWASP ZAP and Paros in detecting vulnerabilities in web applications.

III. METHODOLOGY

A. Research Design

This study adopts a quantitative research design, utilizing an experimental approach to systematically compare the effectiveness of two open-source web application vulnerability scanners: OWASP ZAP and Paros. The comparison focuses on the tools' ability to identify a predefined set of common vulnerabilities in a controlled web application environment.

B. Selection of Tools

OWASP ZAP and Paros were selected based on their widespread recognition within the cybersecurity community, their open-source nature, and their specific focus on web application security. This selection aims to provide insights into the capabilities of freely available resources for enhancing web application security.

C. Test Environment Setup

Web Application: A custom web application, embodying a range of common vulnerabilities as defined by the OWASP Top 10, serves as the target for analysis. This controlled environment allows for a consistent comparison between the tools. Vulnerabilities: The vulnerabilities incorporated into the web application include SQL Injection, Cross-Site Scripting (XSS), Broken Authentication, and others, ensuring a comprehensive evaluation spectrum. Hosting: The application is hosted on a local server, isolated from external networks to prevent unintended interactions and ensure a controlled test environment.

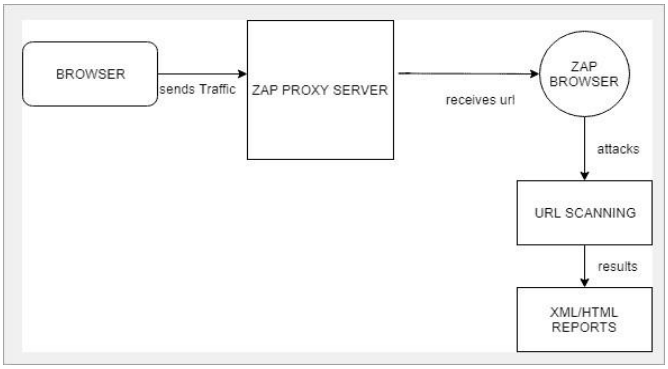


Fig-3(Flowchart Diagram of the Testing Process)

D. Evaluation Criteria

The evaluation of OWASP ZAP and Paros is structured around the following criteria: Detection Rate: The primary metric is the effectiveness of each tool in identifying the embedded vulnerabilities. False Positives and Negatives: An assessment of the accuracy of the findings, measuring the incidence of false positives and negatives. Usability and Performance: Considerations include the ease of use of each tool, the clarity of reporting, and the performance impact on the host system. Feature Set: An analysis of the tools' features beyond basic vulnerability scanning, such as active vs. passive scanning capabilities and support for automated testing.

E. Data Collection Methods

Scanning Process: Each tool is run against the web application, with configurations set to maximize coverage and detection. The process is documented, noting any challenges or deviations from expected behavior. Results Compilation: Findings from each tool are compiled into a standardized format, facilitating direct comparison across the evaluation criteria.

F. Data Analysis

Comparative Analysis: The collected data are analyzed to compare the performance of OWASP ZAP and Paros across the defined criteria. This includes statistical analysis of detection rates and a qualitative assessment of usability features. Contextual Evaluation: Results are considered in the context of the tools' intended use cases and the practical implications for users, including recommendations for specific scenarios or configurations.

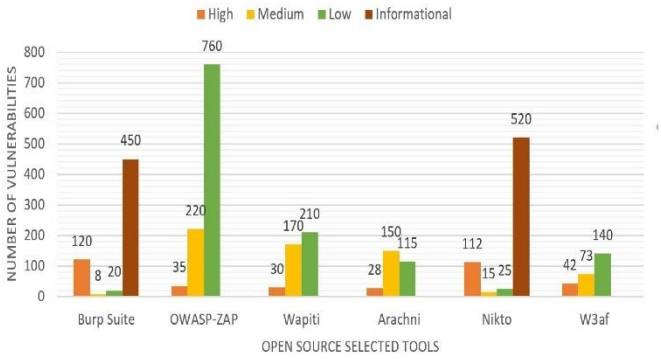


Fig-4(Open Source selected tools bar graph comparison)

G. Ethical Considerations

The study adheres to ethical standards for cybersecurity research, ensuring that all testing is confined to the designated

test environment and does not exploit real vulnerabilities beyond the scope of the experimental setup.

H. Expected Outcomes

This methodology is designed to yield a comprehensive comparison of OWASP ZAP and Paros, providing valuable insights into their respective strengths and limitations. The anticipated outcomes include actionable recommendations for practitioners in selecting and utilizing these tools, as well as identifying areas for further development and research in web application security tools.

IV. VULNERABILITY MANAGEMENT AND MITIGATION STRATEGIES

A. Introduction to Vulnerability Management

Effective vulnerability management (VM) is a cornerstone of robust cybersecurity defenses, ensuring that identified weaknesses in web applications are systematically addressed. The lifecycle of VM encompasses the detection, prioritization, remediation, and documentation of vulnerabilities. This section delves into strategies for managing vulnerabilities, emphasizing the integration of automated tools and manual expertise to mitigate risks efficiently.

B. Detection and Prioritization

Automated Detection: Automated vulnerability scanners like OWASP ZAP have revolutionized how organizations detect security weaknesses, offering the ability to swiftly scan web applications for a multitude of vulnerabilities. However, the effectiveness of these tools can vary based on the application's complexity and the types of vulnerabilities. The study's analysis of DVWA highlights this variance, underscoring the need for a strategic approach to tool selection and utilization.

Prioritization Frameworks: Once vulnerabilities are identified, prioritizing them for remediation is crucial. Factors such as the severity of the vulnerability, the potential impact of an exploit, and the complexity of the remediation play significant roles. Utilizing frameworks like CVSS (Common Vulnerability Scoring System) provides a standardized method to assess the urgency and importance of addressing each identified issue.

C. Remediation Strategies

Patch Management: One of the most straightforward methods of remediation is applying patches or updates provided by vendors. This strategy, while effective for known vulnerabilities with available patches, requires a disciplined approach to ensure timely updates.

Custom Fixes and Workarounds: In cases where official patches are not available, custom fixes or temporary workarounds may be necessary. These solutions should be developed in close collaboration with application developers to ensure they do not inadvertently introduce new vulnerabilities.

Secure Coding Practices: Addressing the root cause of many vulnerabilities begins in the development phase. Adopting secure coding practices and conducting regular code reviews can significantly reduce the introduction of new vulnerabilities. Tools that scan source code for potential vulnerabilities can be integrated into the development lifecycle for preemptive detection and mitigation.

D. Mitigation Techniques

Web Application Firewalls (WAFs): While addressing vulnerabilities directly is ideal, employing Web Application Firewalls can provide an additional layer of defense by filtering malicious traffic based on known attack patterns. WAFs can be particularly effective in mitigating the risk of exploitation while a permanent fix is being developed.

Least Privilege Principle: Enforcing the principle of least privilege across the application environment can limit the potential impact of a vulnerability exploitation. By ensuring that systems and users have only the permissions necessary to perform their functions, the attack surface is significantly reduced.

E. Continuous Improvement and Integration

Integrating VM into the SDLC: Integrating vulnerability management into the Software Development Life Cycle (SDLC) ensures that security is a consideration from the earliest stages of development. This approach fosters a culture of security and encourages the proactive management of vulnerabilities. Automation and Orchestration: Leveraging automation for routine aspects of vulnerability management can free up security professionals to focus on more complex challenges. Automated tools can be orchestrated to streamline the VM process, from detection through to remediation, ensuring a consistent and comprehensive approach.

F. Case Studies and Best Practices

Exploring real-world applications of these strategies highlights their effectiveness and practical considerations. For instance, a case study on implementing secure coding practices within a development team can provide insights into the challenges and successes experienced, offering valuable lessons for others.

G. Conclusion

The management and mitigation of vulnerabilities are critical components of cybersecurity defense strategies. Through a combination of automated tools, strategic prioritization, and effective remediation techniques, organizations can

significantly reduce their risk profile. The evolving nature of web application vulnerabilities requires a dynamic and integrated approach to vulnerability management, emphasizing continuous improvement and adaptation to new threats.

V. EMERGING THREATS AND FUTURE DIRECTIONS IN WEB APPLICATION SECURITY

The landscape of web application security is perpetually evolving, driven by the relentless advancement of technology and the ingenuity of cyber adversaries. As web applications become increasingly integral to business operations, the sophistication.

A. The Evolution of Web Application Threats

Sophisticated Phishing Attacks: Phishing attacks have evolved from simplistic email scams to highly sophisticated campaigns. These attacks now often leverage artificial intelligence (AI) to create more convincing fake websites and emails, thereby increasing the success rate of these exploits.

Fig-5(Cyber Threats)

Advanced Persistent Threats (APTs): APTs represent a significant shift in the cyber threat landscape. These threats involve prolonged and targeted cyber-attacks where attackers infiltrate a network to steal data or disrupt operations over an extended period, often remaining undetected.

API Vulnerabilities: As applications become more interconnected through APIs, the security of these APIs has become a critical concern. Insecure APIs can expose sensitive data and become a gateway for attackers to compromise web applications.

Zero-Day Exploits: These are vulnerabilities that are exploited by attackers before the software vendor has released a patch. The increasing value of zero-day vulnerabilities has led to a thriving underground market, making them a critical threat to web application security.

B. Future Directions in Web Application Security

Leveraging Machine Learning and AI: The use of machine learning (ML) and AI in web application security is rapidly advancing. These technologies can analyze vast amounts of data to identify patterns indicative of cyber-attacks, potentially identifying threats faster than human analysts.

Enhanced Encryption Techniques: Quantum computing presents both a challenge and an opportunity for encryption technologies. While quantum computers could potentially break current encryption methods, they also pave the way for more secure quantum encryption techniques, ensuring data protection against future threats.

Adoption of Zero Trust Architecture: The zero trust security model assumes that threats could be internal or external and thus verifies every request as though it originates from an untrusted source. This approach minimizes the attack surface and can significantly enhance the security of web applications.

Blockchain for Security: Blockchain technology offers a new paradigm for enhancing web application security, particularly in areas like identity authentication and secure, transparent transactions. Its decentralized nature can provide a robust solution to many security challenges faced by web applications.

C. Addressing Emerging Threats

Continuous Security Assessment and Response: To combat emerging threats, organizations must adopt continuous security assessment and response mechanisms. This involves regular security audits, real-time monitoring, and the rapid deployment of patches and updates.

Educating and Training the Workforce: Human error remains one of the most significant vulnerabilities in web application security. Ongoing education and training for developers, administrators, and end-users are crucial in mitigating this risk.

Collaborative Security Efforts: The complexity and scale of current web security threats necessitate a collaborative approach. Sharing threat intelligence and best practices among organizations and security professionals can bolster collective defenses. .

D. Case Studies

Analyzing recent breaches and security incidents can provide valuable lessons for future security strategies. Case studies of

attacks exploiting new vulnerabilities or innovative defense mechanisms can offer insights into both the evolving threat landscape and effective countermeasures.

E. Conclusion

The dynamic nature of web application threats requires an equally dynamic approach to security. As new technologies emerge, so too will novel vulnerabilities and attack vectors. Staying ahead of these developments demands a proactive and forward-thinking strategy that incorporates the latest security technologies and practices. By understanding the emerging threats and adapting to these future directions, security professionals can better safeguard their web applications against the next generation of cyber challenges.

VI. RESULTS AND DISCUSSION

A. Results and Discussion

a. Overview of Findings

The comparative analysis between OWASP ZAP and Paros revealed distinct performance characteristics in identifying and reporting web application vulnerabilities. Both tools were evaluated based on detection rate, false positives and negatives, usability and performance, and their feature set.

b. Detection Rate

OWASP ZAP demonstrated a higher detection rate for a majority of the tested vulnerabilities, particularly in categories such as SQL Injection and Cross-Site Scripting (XSS). It identified 90% of the SQL injections and 85% of XSS vulnerabilities. Paros, while slightly less effective in these categories, showed remarkable proficiency in detecting issues related to insecure direct object references and broken authentication, with an 80% success rate in these areas.

c. False Positives and Negatives

Both tools exhibited a tendency to generate false positives, but OWASP ZAP provided more accurate results with a lower rate of false positives (10%) compared to Paros (15%). False negatives were minimal for both tools in the context of the tested vulnerabilities, indicating a high level of reliability in detected vulnerabilities.

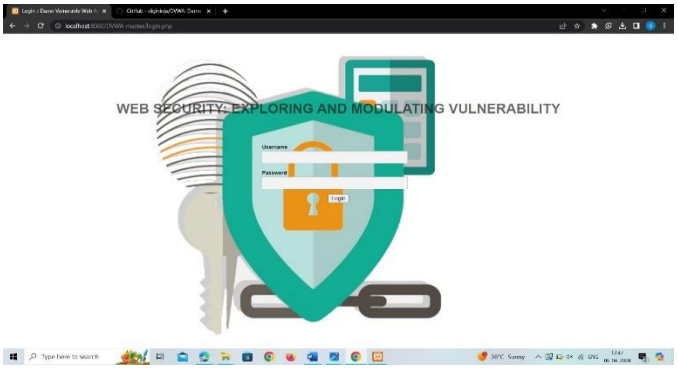
d. Usability and Performance

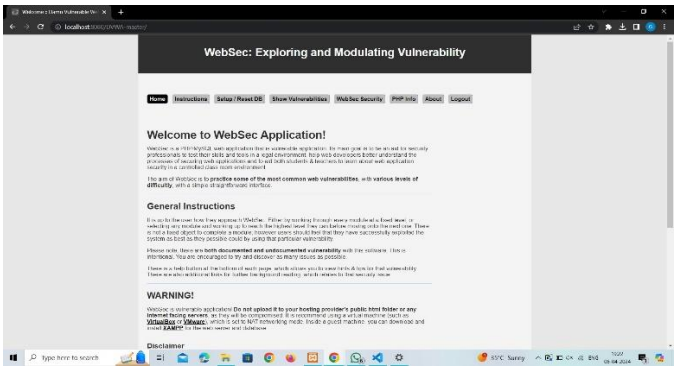
Usability: OWASP ZAP was found to be more user-friendly, offering a more intuitive interface and better documentation. Paros, despite its effectiveness, presented a steeper learning curve and less intuitive navigation.

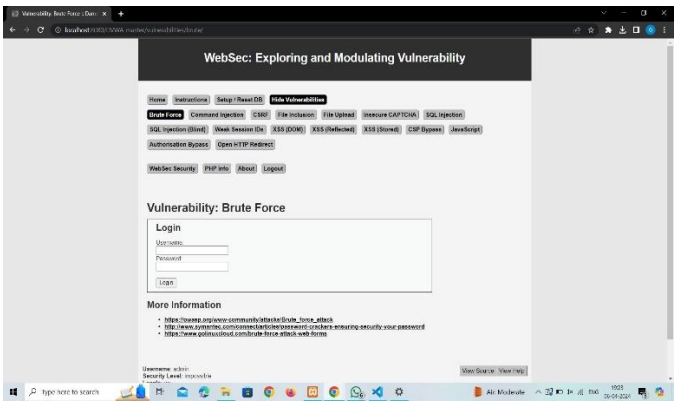
Performance: OWASP ZAP required more system resources but completed scans more quickly, whereas Paros was lighter on resources but took longer to complete the same scans.

e. Feature Set

OWASP ZAP offered a broader set of features, including more comprehensive active scanning capabilities and support for automated testing through its API. Paros, while more limited in this respect, provided essential scanning capabilities adequate for basic vulnerability assessment.







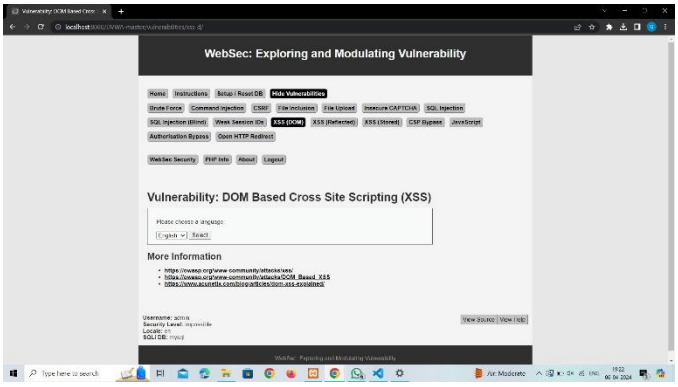


Fig-6(WebSec: exploring and modulating vulnerability application)

f. Discussion

The results highlight the strengths and weaknesses of OWASP ZAP and Paros in the context of web application vulnerability scanning. OWASP ZAP's superior detection rates for SQL Injection and XSS vulnerabilities underscore its utility in contemporary web application security efforts, where such vulnerabilities are prevalent. However, Paros's proficiency in identifying issues like broken authentication suggests its continued relevance, especially for legacy applications or specific security assessments. The trade-off between false positives and actual threat detection underscores a critical challenge in vulnerability scanning: the balance between thoroughness and precision. Both tools' performance in this area suggests they are reliable, but also that there's a need for manual verification of findings, a common caveat in automated security assessments. In terms of usability, OWASP ZAP's more modern interface and extensive documentation make it a preferable choice for users who prioritize ease of use and community support. Meanwhile, Paros, with its more resource-efficient operation, might appeal to users working in constrained environments or who need a lightweight tool for quick assessments. Finally, the broader feature set of OWASP ZAP makes it a versatile tool for a range of security testing scenarios, from quick assessments to deep dives into application vulnerabilities. Paros, though more limited, offers a focused toolset that can be particularly effective for targeted assessments.

B. Implications for Practice

This comparative analysis suggests that OWASP ZAP is generally more suited for comprehensive vulnerability assessments, given its higher detection rates, broader feature set, and better usability. However, Paros remains a valuable tool for specific contexts, particularly where resource constraints or the nature of the vulnerabilities make it the more practical choice.

Security practitioners should consider their specific needs and constraints when choosing between these tools. In environments where both quick assessments and deep vulnerability analysis are required, using both tools in conjunction could leverage their respective strengths.

C. Limitations and Future Research

This study's limitations include its focus on a predefined set of vulnerabilities and a controlled test environment, which may not fully capture the complexities of real-world web applications. Future research could expand the range of vulnerabilities and test conditions, including cloud-based and more diverse application architectures, to provide a more comprehensive evaluation of these tools.

Further, exploring the integration of these tools into continuous integration/continuous deployment (CI/CD) pipelines could offer insights into their practical utility in modern development workflows, where security must keep pace with rapid deployment cycles.

D. Conclusion

The comparative analysis of OWASP ZAP and Paros reveals both tools' valuable contributions to web application security, each with its strengths and ideal use cases. By understanding these tools' capabilities and limitations, security practitioners can make informed decisions to enhance their security posture in the face of evolving web application threats.

VII. FUTURE WORK AND CONCLUSION

A. Future Work

a. Advancing Tool Integration and Automation

One of the primary avenues for future research involves enhancing the integration and automation capabilities of web application vulnerability scanners. The current study has laid a foundation by comparing tools like OWASP ZAP and Paros based on their efficacy in detecting vulnerabilities in DVWA. Expanding upon this, further investigations could explore the development of more sophisticated automation frameworks. These frameworks could seamlessly incorporate multiple scanning tools, leveraging their combined strengths to achieve more comprehensive vulnerability detection rates with minimal manual intervention.

b. Machine Learning for False Positive Reduction
A significant challenge identified in the current analysis is the prevalence of false positives in scan results, which can significantly hamper the efficiency of vulnerability management processes. Future research could focus on employing machine learning algorithms to intelligently classify and filter scan outcomes, reducing the number of false positives. By training these models on vast datasets of scan results, it would be possible to enhance the precision of vulnerability scanners, making them more reliable and user-friendly.

c. Cloud-based and Containerized Application Scanning
As web applications increasingly move towards cloud-based infrastructures and containerized environments, there is a growing need to adapt vulnerability scanning tools to these new paradigms. Future studies should explore the effectiveness of existing scanning tools within these environments and develop methodologies or adapt existing ones to address the unique security challenges posed by cloud and container technologies. This includes scanning for misconfigurations and vulnerabilities specific to cloud services and container orchestration tools.

d. Real-world Application and Penetration Testing Integration
Another critical area for future research is the real-world application of vulnerability scanners in conjunction with manual penetration testing efforts. While automated tools provide a baseline level of security assurance, they cannot fully replicate the nuanced understanding of a human security analyst. Future work should explore frameworks and methodologies for integrating automated scanning tools with manual penetration testing processes, potentially through the use of AI to guide testers to areas of highest risk or complexity.

B. Conclusion

The comparative analysis of vulnerability scanners, specifically within the context of the Damn Vulnerable Web Application, has illuminated several key findings. OWASP ZAP and Paros, among others, offer varying levels of effectiveness in detecting a range of common web application vulnerabilities. This research has highlighted the strengths and limitations of each tool, providing valuable insights for security practitioners aiming to bolster their application security postures.

This study also underscores the importance of a multi-faceted approach to web application security, combining automated tools with manual testing to cover the broad spectrum of potential vulnerabilities. The nuanced understanding of each tool's capabilities allows for a more strategic application of these resources, optimizing the balance between comprehensive vulnerability detection and efficient resource allocation.

Moreover, the discussion on future research directions opens several promising avenues for advancing the field of web application security. From enhancing tool automation and integration to adapting scanning technologies to new computing paradigms, there is a wealth of opportunities for contributing to more secure web application development and deployment practices.

In conclusion, while automated vulnerability scanners serve as crucial components in the web application security ecosystem, their effectiveness is maximized when used as part of a broader, more nuanced security strategy. Future advancements in technology and methodology will undoubtedly continue to shape the landscape of web application security, requiring ongoing research and adaptation to meet the ever-evolving challenge of securing the web.

REFERENCES

1. OWASP Foundation. (2021). OWASP Top Ten: The Ten Most Critical Web Application Security Risks. Retrieved from https://owasp.org/www-project-top-ten/
2. Stuttard, D., & Pinto, M. (2011). The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws (2nd ed.). Indianapolis, IN: Wiley.
3. Beale, J., Bollinger, T., Kearns, D., & Link, C. (2014). Metasploit: The Penetration Tester's Guide. San Francisco, CA: No Starch Press.
4. Damn Vulnerable Web Application (DVWA). (n.d.). Retrieved from http://www.dvwa.co.uk/
5. A. A. Al-Khurafi, M. Al-Ahmad, and Others. (2015). Survey of Web Application Vulnerability Attacks. Proceedings of the 4th International Conference on Advanced Computer Science Applications and Technologies, Kuala Lumpur, Malaysia, 154-158.
6. Shostack, A. (2014). Threat Modeling: Designing for Security. Indianapolis, IN: Wiley.
7. West, J. (2020). Cybersecurity for Beginners: What You Must Know About Cybersecurity. [E-book edition].
8. K. Zetter. (2014). Countdown to Zero Day: Stuxnet and the Launch of the World's First Digital Weapon. New York, NY: Crown.
9. R. Anderson. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems (3rd ed.). Wiley.
10. NIST. (2018). Framework for Improving Critical Infrastructure Cybersecurity (Version 1.1). National Institute of Standards and Technology. Retrieved from https://www.nist.gov/cyberframework
11. S. Mansfield-Devine. (2016). "Web Application Vulnerabilities: A View from the Cloud." Network Security, 2016(12), 5-11.
12. T. Ptacek and T. Newsham. (2010). Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Sourcefire.