

Graph-Based Probabilistic Approach for Automated Vulnerability Chain Detection in Web Security Scanning

Name Surname

School of Information Technology and Engineering

Kazakh-British Technical University

Almaty, Kazakhstan

Email: s_aman@kbtu.kz

I. INTRODUCTION

Web applications have become critical infrastructure for modern organizations, handling sensitive data and business-critical operations across e-commerce, healthcare, finance, and government sectors. However, the increasing complexity of web applications has introduced numerous security vulnerabilities that pose significant risks to data integrity, confidentiality, and availability. According to recent studies, the average web application contains multiple security flaws, with approximately 3.73% of tested applications exhibiting at least one instance of broken access control vulnerabilities alone [36]. While organizations invest substantial resources in security testing, the detection and remediation of web application vulnerabilities remain challenging due to the sophisticated nature of modern cyber attacks.

Current web application security testing relies heavily on automated vulnerability scanners such as OWASP ZAP [38], Burp Suite, and Nikto. These tools employ static analysis, dynamic analysis, and fuzzing techniques to identify security weaknesses in web applications [22], [24]. Comparative studies have demonstrated that modern vulnerability scanners can effectively detect individual vulnerabilities including SQL injection, cross-site scripting (XSS), and authentication flaws [1], [23]. However, these scanners operate under a fundamental limitation: they identify vulnerabilities in isolation, treating each security issue as an independent finding without considering potential interactions or dependencies between multiple vulnerabilities.

This isolated detection approach fails to capture a critical threat vector in web security: multi-stage attack chains. In real-world scenarios, attackers rarely exploit a single vulnerability to compromise a system. Instead, sophisticated attacks combine multiple vulnerabilities in sequence, where each vulnerability enables the exploitation of the next, ultimately leading to critical security breaches that individual vulnerability assessments fail to predict [29]. For instance, an attacker might leverage information disclosure to obtain session tokens, exploit XSS to bypass CSRF protection, and finally achieve administrative privilege escalation—a three-

stage attack chain that appears as three isolated medium-severity findings in traditional scan reports. Security analysts must manually review hundreds of vulnerability findings and mentally correlate them to identify such compound threats, a process that is time-consuming, error-prone, and requires significant expertise [21].

Existing research has explored various approaches to address vulnerability correlation and attack path analysis. Statistical correlation methods have been proposed to identify relationships between vulnerabilities based on temporal and spatial patterns [30]. Attack tree modeling provides theoretical frameworks for representing multi-stage attacks [46], and recent advances in automated penetration testing leverage machine learning and reinforcement learning techniques [14], [10]. However, these approaches suffer from several limitations: statistical methods require offline manual analysis and lack real-time processing capabilities; attack tree frameworks remain primarily theoretical without automated implementation for web applications; and ML-based penetration testing tools focus on network-level or IoT scenarios rather than web-specific vulnerability chains [48]. Furthermore, none of these solutions provide seamless integration with widely-adopted vulnerability scanners, creating a gap between vulnerability detection and chain analysis.

This paper presents a novel graph-based probabilistic approach for automated detection of multi-stage attack chains in web applications, directly integrated with OWASP ZAP. Our system addresses the limitations of existing approaches through four key contributions: (1) a probabilistic rule engine with 24 domain-specific chain rules that encode real-world attack patterns with confidence scores; (2) a graph-based detection algorithm using depth-first search to identify vulnerability chains of length 2-4 with sub-second performance; (3) an intelligent filtering mechanism that reduces tens of thousands of potential chains to critical actionable findings through on-the-fly deduplication and subchain removal; and (4) real-time integration with OWASP ZAP via REST API, enabling automated chain analysis immediately upon scan completion. For instance, in our experimental evaluation, the system analyzed 129 vulnerabilities and identified 44 critical

attack chains in 0.4 seconds, reducing 37,000 raw chain candidates to actionable security insights through intelligent filtering.

II. LITERATURE REVIEW

The detection of security vulnerabilities in web applications has been extensively studied from multiple perspectives. We organize related work into four primary research areas: web vulnerability scanners and testing methodologies, vulnerability correlation and prioritization approaches, attack modeling and chain detection techniques, and machine learning applications in security analysis.

A. Web Vulnerability Scanners and Testing Methodologies

Automated vulnerability scanners constitute the foundation of modern web application security testing. OWASP ZAP (Zed Attack Proxy) represents one of the most widely adopted open-source security testing tools, offering both passive and active scanning capabilities [38]. Recent benchmarking studies have evaluated ZAP's effectiveness in detecting OWASP Top 10 vulnerabilities, demonstrating its ability to identify SQL injection, XSS, and security misconfigurations with varying degrees of accuracy [1]. Comparative analyses of vulnerability scanners including ZAP, Burp Suite, Arachni, and Nikto have revealed that while each tool exhibits unique strengths, all share a common limitation: they detect vulnerabilities independently without correlating findings [22], [23], [24].

Automated testing methodologies have evolved to address specific vulnerability classes. Combinatorial testing approaches have been proposed for SQL injection detection, systematically generating attack vectors based on input grammars [6]. Studies comparing automated versus manual penetration testing have demonstrated that while automation improves efficiency, manual analysis remains necessary for identifying complex attack scenarios [7]. Recent advances in automated penetration testing include expert systems that guide security assessments through threat intelligence [9] and LLM-enhanced tools that leverage large language models for test case generation [10]. However, these approaches focus on individual vulnerability detection rather than identifying relationships between multiple security weaknesses. Furthermore, existing scanners lack context understanding between vulnerabilities, treating each finding as an isolated security issue without considering how multiple flaws might be chained together in a compound attack.

B. Vulnerability Correlation and Prioritization

The challenge of correlating vulnerability data from multiple sources has gained increasing attention in application security research. Statistical correlation methods have been developed to analyze relationships between vulnerability detection results and real-world attack patterns observed in Web Application Firewall (WAF) logs [30]. These approaches employ time-series analysis to identify correlations between vulnerability types and actual exploitation attempts, providing

insights into which combinations of vulnerabilities represent elevated risk [29].

Attack path prediction represents an advanced application of vulnerability correlation. Recent work has proposed using correlation analysis combined with attack tree modeling and multi-layer perceptrons to predict likely attack sequences based on vulnerability distributions across application layers [31]. This research demonstrates that vulnerabilities can be mapped to attack trees representing threat models, enabling simulation of potential attack paths. However, these correlation approaches require manual analysis and offline processing, lacking the real-time processing capability necessary for integration into security scanning workflows. Additionally, while statistical correlation can identify historical patterns, it does not incorporate domain knowledge about vulnerability exploitability and real-world attack probabilities.

Application vulnerability correlation (AVC) tools have emerged in the commercial security space to aggregate and normalize findings from multiple security testing tools [29]. These solutions address the problem of duplicate vulnerabilities reported by different scanners and attempt to prioritize remediation efforts. However, they focus primarily on deduplication and risk scoring of individual vulnerabilities rather than identifying multi-stage attack chains.

C. Attack Modeling and Chain Detection

Theoretical frameworks for modeling attacks have been established through attack tree and attack graph methodologies. Attack trees provide a formal notation for representing how attackers might achieve specific goals through combinations of actions [46]. This hierarchical representation captures attack sequences and alternative paths, enabling security analysts to reason about system vulnerabilities systematically. However, attack tree construction typically requires manual effort and domain expertise, limiting their application in automated security testing.

Threat modeling approaches have been developed for specific domains, particularly IoT and edge computing environments [47]. Expert systems for automated threat modeling and penetration testing have shown promise in IoT ecosystems, combining threat intelligence with automated testing frameworks [48]. These systems demonstrate the feasibility of automating security assessment through rule-based approaches. Nevertheless, they are focused on network-level and IoT-specific threats rather than web application vulnerability chains.

Recent advances in automated penetration testing have explored reinforcement learning and deep learning techniques. Deep reinforcement learning has been applied to automated penetration testing in dynamic network scenarios, learning optimal attack paths through interaction with target systems [14]. LLM-empowered penetration testing tools have demonstrated the potential of large language models to automate complex security testing workflows [15]. While these approaches show promising results in network penetration testing, they are not

specialized for web application chains and require substantial computational resources for training and execution.

D. Machine Learning and Deep Learning for Vulnerability Detection

Machine learning techniques have been increasingly applied to vulnerability detection and security analysis. Deep learning-based systems have been developed for source code vulnerability detection, employing neural networks to identify security flaws in software [41], [42]. Systematic literature reviews have examined the application of deep learning to web application security, finding that convolutional neural networks, long short-term memory networks, and deep feedforward networks are commonly used for vulnerability detection [16].

Specific vulnerability classes have been targeted with machine learning approaches. LSTM encoder-decoder architectures with word embeddings have been proposed for XSS attack detection [40]. These models learn patterns from training data to classify inputs as benign or malicious. However, machine learning approaches to web security face significant limitations: they are trained on individual vulnerability patterns rather than attack sequences, require substantial labeled training data, and produce results that lack interpretability for security practitioners [17]. Furthermore, ML-based vulnerability detection focuses on identifying instances of known vulnerability types rather than discovering relationships between multiple vulnerabilities that could form attack chains.

E. Research Gaps

Despite extensive research in web application security testing, several critical gaps remain unaddressed. First, existing vulnerability scanners detect security issues in isolation without identifying multi-stage attack chains that represent compound threats. Second, vulnerability correlation approaches require manual analysis and offline processing, lacking integration with real-time scanning tools. Third, attack modeling frameworks remain primarily theoretical or focused on non-web domains such as IoT and network security. Fourth, machine learning approaches target individual vulnerability detection rather than attack sequence identification. Our work addresses these gaps by introducing a probabilistic graph-based system that automatically detects vulnerability chains through real-time integration with OWASP ZAP, combining domain knowledge encoded in probabilistic rules with efficient graph traversal algorithms.

III. METHODS

A. Research Gaps

Contemporary web application security assessment faces four fundamental challenges that limit the effectiveness of vulnerability detection and remediation efforts.

Gap 1: Isolated Vulnerability Detection. Existing vulnerability scanners including OWASP ZAP, Burp Suite, and Nikto identify individual security weaknesses without analyzing potential relationships between findings [22], [23]. When a scanner reports 100-200 vulnerabilities in a typical

web application assessment, security analysts must manually review each finding independently. This approach fails to recognize compound attacks where multiple vulnerabilities combine to enable critical breaches. For example, an information disclosure vulnerability that leaks authentication tokens becomes significantly more dangerous when combined with a CSRF bypass, yet scanners report these as separate medium-severity issues rather than identifying the critical attack chain they form together.

Gap 2: Manual Correlation Analysis. Security analysts currently perform vulnerability correlation manually, a process that requires examining hundreds of findings to identify potential attack paths [21]. This manual approach is time-consuming, requiring hours or days for comprehensive analysis of large scan reports, and is highly error-prone, as analysts may overlook subtle relationships between vulnerabilities. The lack of automated correlation capabilities means that critical multi-stage attack chains may remain undetected until they are exploited by attackers who possess the time and expertise to identify these compound threats.

Gap 3: Absence of Probability Assessment. While statistical correlation methods have been proposed for vulnerability analysis [30], they fail to incorporate domain knowledge about vulnerability exploitability and real-world attack probabilities. Existing approaches treat all vulnerability combinations equally, without considering that certain chains (e.g., XSS leading to CSRF bypass) are significantly more likely to be exploitable than others. This lack of probability assessment results in either overwhelming analysts with false positive chain candidates or missing genuine attack paths due to overly conservative filtering.

Gap 4: Integration Challenges. Most research tools for vulnerability correlation and attack path analysis operate as standalone systems that are disconnected from the vulnerability scanning workflow [29]. OWASP ZAP, despite being one of the most popular open-source security scanners, lacks built-in chain detection capabilities. This integration gap forces security teams to export scan results, manually process them through separate correlation tools, and reconcile findings across multiple systems—a workflow that introduces delays and reduces the likelihood that chain analysis will be performed systematically.

B. Proposed Approach

We propose a graph-based probabilistic approach that addresses these gaps through automated, real-time vulnerability chain detection integrated directly with OWASP ZAP. Our system models vulnerabilities and their potential relationships as a directed graph, where nodes represent individual security findings and edges encode probabilistic rules defining how one vulnerability can enable exploitation of another.

The core of our approach consists of four interconnected components. First, a **graph representation** transforms vulnerability scan results into a structured format where each vulnerability becomes a node annotated with attributes including severity, exploitability, and context information. Second, a

probabilistic rule engine encodes 24 domain-specific chain rules that capture real-world attack patterns, such as "XSS enables CSRF bypass with 85% probability" or "SQL injection leads to privilege escalation with 90% probability." Each rule specifies source and target vulnerability types, transition probability, and contextual constraints that must be satisfied for the chain to be viable.

Third, a **depth-first search chain detection algorithm** traverses the vulnerability graph to identify all potential attack chains of length 2-4, applying probability thresholds to filter out unlikely paths. The algorithm employs path pruning to avoid exploring chains with cumulative probability below a configurable threshold, ensuring computational efficiency even for large vulnerability sets. Fourth, a **smart filtering mechanism** processes the raw chain candidates to eliminate duplicates and remove subchains—if chain A→B→C is detected, the system removes the shorter chain A→B since it represents a subset of the longer attack path.

The system integrates with OWASP ZAP through its REST API, enabling automated chain analysis immediately upon scan completion. When a ZAP scan finishes, our system retrieves vulnerability findings, constructs the graph, executes chain detection, and generates an enriched report highlighting critical attack chains alongside individual vulnerabilities. This real-time integration eliminates the workflow gap between vulnerability detection and chain analysis, ensuring that compound threats are identified as part of the standard security assessment process.

IV. CONCLUSION

[To be completed with experimental results]

REFERENCES

- [1] U.-S. Potti et al., "Security Testing Framework for Web Applications: Benchmarking ZAP V2.12.0 and V2.13.0 by OWASP as an Example," *arXiv preprint arXiv:2501.05907*, Jan. 2025.
- [2] O. R. Laponina, "Using the ZAP Vulnerability Scanner to Test Web Applications," in *2017 IEEE Conference on Application of Information and Communication Technologies*, 2017.
- [3] A. Lathifah, F. B. Amri, and A. Rosidah, "Security Vulnerability Analysis of the Sharia Crowdfunding Website Using OWASP-ZAP," in *2022 10th International Conference on Cyber and IT Service Management (CITSM)*, pp. 1–5, 2022.
- [4] M. Alfarizi et al., "Vulnerability Analysis and Effectiveness of OWASP ZAP," *Repository UIR*, 2024.
- [5] A. Jakobsson and I. Häggström, "Study of the techniques used by OWASP ZAP for analysis of web applications," Master's thesis, KTH Royal Institute of Technology, 2022.
- [6] J. Bozic, B. Garn, D. E. Simos, and F. Wotawa, "Automated Combinatorial Testing for Detecting SQL Vulnerabilities in Web Applications," in *Proceedings of the 14th International Workshop on Automation of Software Test*, pp. 55–61, 2019.
- [7] M. A. Khan et al., "Automated versus Manual Approach of Web Application Penetration Testing," in *2020 IEEE International Conference on Systems, Man, and Cybernetics*, 2020.
- [8] L. Feldmann et al., "Overview and Open Issues on Penetration Test," *Journal of the Brazilian Computer Society*, vol. 23, no. 1, pp. 1–16, 2017.
- [9] D. Granata, M. Rak, and G. Salzillo, "Advancing ESSecA: A Step Forward in Automated Penetration Testing," in *Proceedings of the 19th International Conference on Availability, Reliability and Security*, 2024.
- [10] G. Deng et al., "PentestAgent: Incorporating LLM Agents to Automated Penetration Testing," in *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*, 2024.
- [11] A. Armando, R. Carbone, and L. Compagna, "Semi-Automatic Security Testing of Web Applications from a Secure Model," in *2012 IEEE Sixth International Conference on Software Security and Reliability*, pp. 253–262, 2012.
- [12] A. Singh and S. Sharma, "Vulnerability Assessment and Penetration Testing of Web Application," in *2018 4th International Conference on Computing Communication and Automation (ICCCA)*, pp. 1–6, 2018.
- [13] X. Zhang et al., "Towards Automated Penetration Testing for Cloud Applications," in *2014 IEEE 7th International Conference on Cloud Computing*, pp. 156–163, 2014.
- [14] Q. Li et al., "DynPen: Automated Penetration Testing in Dynamic Network Scenarios Using Deep Reinforcement Learning," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 1–14, 2024.
- [15] G. Deng et al., "PENTESTGPT: An LLM-Empowered Automatic Penetration Testing Tool," in *Proceedings of the 33rd USENIX Conference on Security Symposium*, 2024.
- [16] R. L. Alaoui and E. H. Nfaoui, "Deep Learning for Vulnerability and Attack Detection on Web Applications: A Systematic Literature Review," *Future Internet*, vol. 14, no. 4, p. 118, 2022.
- [17] M. S. Chughtai, I. Bibi, S. Karim, S. W. A. Shah, A. A. Laghari, and A. A. Khan, "Deep Learning Trends and Future Perspectives of Web Security and Vulnerabilities," *Journal of High Speed Networks*, 2024.
- [18] M. Tahir et al., "Deep Learning and Web Applications Vulnerabilities Detection," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 7, 2024.
- [19] A. Iqbal et al., "Web Application Security Vulnerabilities Detection Approaches: A Systematic Mapping Study," in *2015 IEEE/ACIS 16th International Conference on Software Engineering*, pp. 1–6, 2015.
- [20] A. Singh and A. Sharma, "Deep Analysis of Attacks and Vulnerabilities of Web Security," in *Advances in Data and Information Sciences*, pp. 1085–1095, Springer, 2022.
- [21] S. Kumar, R. Mahajan, N. Kumar, and S. K. Khatri, "A Study on Web Application Security and Detecting Security Vulnerabilities," in *2017 6th International Conference on Reliability, Infocom Technologies and Optimization*, pp. 451–455, 2017.
- [22] Y. Makino and V. Klyuev, "Evaluation of Web Vulnerability Scanners," in *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems*, vol. 1, pp. 399–402, 2015.
- [23] A. I. Mohaidat and A. Al-Helali, "Web Vulnerability Scanning Tools: A Comprehensive Overview, Selection Guidance, and Cyber Security Recommendations," *International Journal of Research Studies in Computer Science and Engineering*, vol. 10, no. 1, pp. 8–15, 2024.
- [24] S. M. Srinivasan and R. S. Sangwan, "Web App Security: A Comparison and Categorization of Testing Frameworks," *IEEE Software*, vol. 34, no. 1, pp. 99–102, 2017.
- [25] A. Alzahrani, A. Alqazzaz, Y. Zhu, H. Fu, and N. Almarshfi, "Web Application Security Tools Analysis," in *2017 IEEE 3rd International Conference on Big Data Security on Cloud*, pp. 237–242, 2017.
- [26] M. C. Ghanem and T. M. Chen, "Enhancing Web Application Security through Automated Penetration Testing with Multiple Vulnerability Scanners," *Computers*, vol. 12, no. 11, p. 235, 2023.
- [27] Y. Zhang et al., "An Automatic Vulnerability Scanner for Web Applications," in *2020 IEEE Conference on Communications and Network Security*, 2020.
- [28] P. Touseef, "Analysis of Automated Web Application Security Vulnerabilities Testing," in *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, 2019.
- [29] S. Kasturi, X. Li, J. Pickard, and P. Li, "Prioritization of Application Security Vulnerability Remediation Using Metrics, Correlation Analysis, and Threat Model," *American Journal of Software Engineering and Applications*, vol. 12, no. 1, pp. 5–13, 2024.
- [30] S. Kasturi et al., "Understanding Statistical Correlation of Application Security Vulnerability Data from Detection and Monitoring Tools," in *2023 IEEE International Conference on Big Data*, pp. 1–6, 2023.
- [31] S. Kasturi, "Predicting Application Security Attack Paths Using Correlation Analysis, Attack Tree, and Multi-Layer Perceptron," Ph.D. dissertation, Indiana State University, 2024.
- [32] M. Vieira et al., "Web Application Security through Comprehensive Vulnerability Assessment," *Procedia Computer Science*, vol. 230, pp. 77–86, 2024.
- [33] J. Fonseca, M. Vieira, and H. Madeira, "Evaluation of Web Security Mechanisms Using Vulnerability and Attack Injection," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 440–453, 2014.

- [34] H. C. Huang, Z. K. Zhang, H. W. Cheng, and S. P. Shieh, “Web Application Security: Threats, Countermeasures, and Pitfalls,” *Computer*, vol. 50, no. 6, pp. 81–85, 2017.
- [35] OWASP Foundation, “OWASP Top 10:2021,” 2021. [Online]. Available: <https://owasp.org/Top10/>
- [36] OWASP Foundation, “OWASP Top 10:2025,” 2025. [Online]. Available: <https://owasp.org/Top10/2025/>
- [37] OWASP Foundation, “OWASP Benchmark Project,” 2024. [Online]. Available: <https://owasp.org/www-project-benchmark/>
- [38] OWASP Foundation, “OWASP Zed Attack Proxy (ZAP),” 2024. [Online]. Available: <https://www.zaproxy.org/>
- [39] A. Kieyzun, P. J. Guo, K. Jayaraman, and M. D. Ernst, “Automatic Creation of SQL Injection and Cross-Site Scripting Attacks,” in *2009 IEEE 31st International Conference on Software Engineering*, pp. 199–209, 2009.
- [40] R. L. Alaoui and E. H. Nfaoui, “Cross Site Scripting Attack Detection Approach Based on LSTM Encoder-Decoder and Word Embeddings,” *International Journal of Intelligent Systems and Applications in Engineering*, vol. 11, pp. 277–282, 2023.
- [41] Z. Li et al., “VulDeePecker: A Deep Learning-Based System for Vulnerability Detection,” in *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.
- [42] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, and Z. Chen, “SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2244–2258, 2022.
- [43] N. Antunes and M. Vieira, “Enhancing Penetration Testing with Attack Signatures and Interface Monitoring for the Detection of Injection Vulnerabilities in Web Services,” in *2011 IEEE International Conference on Services Computing*, pp. 104–111, 2011.
- [44] M. Felderer, M. Büchler, M. Johns, A. D. Brucker, R. Breu, and A. Pretschner, “Security Testing: A Survey,” *Advances in Computers*, vol. 101, pp. 1–51, 2016.
- [45] B. Garn, I. Kapsalis, D. E. Simos, and S. Winkler, “On the Applicability of Combinatorial Testing to Web Application Security Testing: A Case Study,” in *Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing*, pp. 16–21, 2014.
- [46] S. Mauw and M. Oostdijk, “Foundations of Attack Trees,” in *International Conference on Information Security and Cryptology*, pp. 186–198, 2006.
- [47] M. Ficco, D. Granata, M. Rak, and G. Salzillo, “Threat Modeling of Edge-Based IoT Applications,” in *Quality of Information and Communications Technology*, pp. 282–296, Springer, 2021.
- [48] M. Rak, G. Salzillo, and D. Granata, “ESSecA: An Automated Expert System for Threat Modelling and Penetration Testing for IoT Ecosystems,” *Computers and Electrical Engineering*, vol. 99, p. 107721, 2022.