A Lightweight Automated SQL Injection Testing Tool with Integrated Discovery and Structured Reporting

Olusanjo Olugbemi Fasola[1], Ugochukwu Onwudebelu[2], Ibrahim Ismail [1], Nancy Chinyere Woods[3]

[1]Department of Cybersecurity, School of Information and Communication Technology, Federal University of Technology, Minna

[2]Department of Computer Science/Informatics, Alex Ekwueme Federal University Ndufu Alike (FUNAI), Abakaliki, Ebonyi State, Nigeria

[3]Department of Computer Science, University of Ibadan, Nigeria

sanjo.fasola@gmail.com,
ugochukwu.onwudebelu@funai.edu.ng,
ismail.ibrahim@futminna.edu.ng,
Chyn.woods@gmail.com
ugochukwu.onwudebelu@funai.edu.ng (Corresponding Author)

Abstract SQL injection (SQLi) remains one of the most persistent and high-impact vulnerabilities in web applications, consistently ranked among the OWASP Top 10 threats. Although mature tools such as SQLMap provide extensive detection and exploitation capabilities, their steep learning curve, complex configuration requirements, and command-line-driven interfaces limit accessibility for beginners, students, and small organizations with constrained technical resources. This paper presents the design, implementation, and evaluation of a lightweight, automated SQL injection testing tool that emphasizes usability, efficient detection, comprehensive endpoint discovery, and structured reporting. The proposed system integrates four classical SQLi detection techniques—error-based, union-based, boolean-blind, and time-based—within a modular scanning engine accessible through an intuitive graphical user interface. To improve assessment coverage, an automated discovery module identifies hidden and unlinked endpoints via robots.txt inspection, sitemap.xml parsing, and hyperlink crawling. A multi-format reporting framework generates human-readable and machine-processable outputs, including executive summaries, vulnerability evidence, and mitigation recommendations. Experimental evaluation was conducted using Damn Vulnerable Web Application (DVWA) and OWASP Juice Shop, representing standard testbeds for ethical and repeatable SQLi assessment. Results demonstrate detection success rates ranging from 79% to 93% across the implemented techniques, while integrated discovery increased endpoint coverage by more than 100%. Comparative benchmarking against SQLMap indicates that although SQLMap offers deeper exploitation capabilities, the proposed tool delivers superior accessibility, reduced configuration complexity, faster deployment, and structured reporting suitable for educational, developmental, and small-scale security testing environments. The findings validate that lightweight design can achieve effective vulnerability detection without sacrificing fundamental security capabilities, providing a practical entry point for SQL injection assessment.

Keywords: SQL injection, Web application Security, Vulnerability scanning, Endpoint discovery, Automated testing, Cybersecurity

1. Introduction

The exponential growth of web-based services has transformed modern society, enabling digital banking, e-commerce, healthcare delivery, academic management, and government operations. However, this expansion has also introduced significant security challenges, particularly vulnerabilities arising from improper handling of user input and backend database interactions. Among these threats, SQL injection (SQLi) remains one of the most critical and persistent attack vectors, enabling adversaries to manipulate database queries, bypass authentication mechanisms, extract sensitive data, and compromise system integrity. Despite over two decades of research and mitigation efforts, SQLi continues to rank prominently in the OWASP Top 10, underscoring its ongoing relevance and severity (OWASP, 2021).

Existing SQLi detection and exploitation tools, notably SQLMap, offer comprehensive functionality and extensive payload repositories, enabling deep vulnerability assessment and exploitation. Nevertheless, these tools are primarily command-line driven, demand advanced configuration knowledge, and impose steep learning curves, limiting their adoption among students, novice developers, and small organizations with limited cybersecurity expertise. Moreover, many lightweight scanners provide limited endpoint discovery, requiring users to manually specify URLs and parameters, thereby risking incomplete assessment coverage. Recent research has explored machine learning and deep learning approaches for SQLi detection, achieving high accuracy under controlled conditions (Lu et al., 2022; Leung et al., 2024; Khan et al., 2025). Akpojaro et al. (2014) evaluated unsupervised

learning algorithms for malware detection under scenarios involving known and unknown attackers, finding that the γ-algorithm outperforms other unsupervised techniques in terms of detection efficacy. Yang et al. (2024) introduced SWIDE (Successful Web Injection Detection Engine), designed to identify successful web injection attacks, including PHP command injection and SQLi. SWIDE leverages two key insights: successful attacks must comply with programming language syntax, and they inevitably produce observable effects such as execution results or backdoors. Through syntactic and semantic analysis, SWIDE detects malicious syntax in obfuscated payloads and recovers attack intent. Its two-stage design—attack identification and confirmation—enables accurate detection even under complex obfuscation. The system has been deployed and validated in real-world environments in collaboration with a cybersecurity firm, demonstrating practical applicability beyond proof-of-concept studies. However, these methods face challenges including labeled data scarcity, generalization limitations, interpretability concerns, and computational overhead, restricting their practical deployment in small-scale and educational environments.

This study addresses these limitations by designing and implementing a lightweight, GUI-based SQL injection testing tool that integrates classical detection techniques, automated endpoint discovery, and structured reporting. The objectives are threefold: (i) enhance usability and accessibility, (ii) improve vulnerability coverage through integrated discovery, and (iii) deliver actionable and interpretable reporting. The proposed system aims to complement, rather than replace, comprehensive frameworks like SQLMap by providing a practical, low-overhead solution suitable for learning, development, and small organizational contexts.

2. Related Work

2.1 SQL Injection Detection Techniques

Security has become a fundamental requirement for individuals, organizations, and governments as reliance on digital systems increases. The rise in cyberattacks despite significant IT security investments underscores the persistent vulnerability of electronic information. Onwudebelu and Akpojaro (2012) emphasize the concept of an "electronic life" constantly targeted by social engineering and propose security-awareness strategies to mitigate such threats. Early approaches to SQLi detection focused on rule-based and lexical analysis methods. Hlaing and Khaing (2020) proposed a reserved-word lexicon and tokenization strategy for detecting malicious queries, demonstrating satisfactory performance under controlled conditions. Sheykhkanloo (2015) introduced a pattern-recognition neural network model employing URL generation and classification to detect and categorize SQLi attacks, later extended by Onwudebelu et al. (2025) to improve classification performance. With the maturation of artificial intelligence techniques, recent studies have leveraged deep learning and generative models to enhance SQLi detection and sample generation. Lu et al. (2022) employed GAN-based data augmentation to overcome data scarcity, while Leung et al. (2024) and Khan et al. (2025) demonstrated the effectiveness of language models and GAN-driven mutation engines in generating evasive SQLi payloads capable of bypassing modern web application firewalls (WAFs). Although effective, these approaches introduce challenges related to interpretability, computational cost, and real-world generalization. While there have been solutions proposed in the literature for SQLi attack detection using learning-based frameworks, the problem is often formulated as a binary, single-attack vector problem without considering the prioritization and prevention component of the attack (Paul et al., 2024). Research on Lightweight intrusion detection model based on Convolutional Neural Network (CNN) was discussed in Cao et al. (2025) while detection of SQL Injection and cross-site scripting based on multi-Model CNN combined with bidirectional GRU and multi-head self-attention (Hsiao & Wang, 2023). Furthermore, enhanced detection and prevention of SQL injection and cross-site scripting attacks in Web applications was carried out by Imran et al. (2024), where the analyzing algorithms and threat modeling approaches where used, Jahanshahi et al. (2020) mitigated SQL Injection attacks on legacy Web applications. Hwang et al. (2024) conducted a comparative study of decision and ensemble models for SQL injection detection. Evaluating Decision Tree, Random Forest, XGBoost, AdaBoost, Gradient Boosting Decision Tree (GBDT), and Histogram Gradient Boosting Decision Tree (HGBDT) on a comprehensive SQLi dataset, they found that Decision Tree, Random Forest, and AdaBoost achieved the highest performance, with 99.50% accuracy and an F1 score of 99.33%. Their results highlight the effectiveness of ensemble and boosting techniques in enhancing real-time SQLi detection and robustness against evolving attacks.

2.2 Web Application Firewalls and IDS-Based Approaches

Several studies focus on integrating SQLi detection within WAF and intrusion detection system (IDS) frameworks. Ikramaputra et al. (2025) combined ensemble learning models with lightweight heuristic filters, achieving detection accuracy exceeding 99% while maintaining low latency. Singh and Aksanli (2019) enhanced Snort-based IDS performance through deep packet inspection and real-time rule generation. Zaidan et al. (2024)

evaluated collaborative detection using SIEM platforms and heterogeneous WAF deployments, demonstrating improved situational awareness and centralized threat reporting.

2.3 Automated Discovery and Security Testing

Automated discovery remains a critical challenge in dynamic vulnerability scanning. Schlaubitz et al. (2025) proposed A2CT for automated detection of access control vulnerabilities, emphasizing broad request-type coverage. However, many SQLi-focused tools rely heavily on manually provided URLs, limiting assessment completeness. Integrated crawling, sitemap analysis, and robots.txt inspection have been shown to significantly enhance discovery coverage, particularly in dynamic web environments.

Messas et al. (2024) proposed sAIfe, a prescriptive threat modeling method for AI applications under development. By providing graphical references, remediation suggestions, and practical assessment steps, sAIfe simplifies security analysis. Evaluated on a real-world AI system and validated with academic developers, sAIfe effectively identified potential vulnerabilities while offering actionable mitigation options, demonstrating usability and efficiency in practical settings.

Dynamic, black-box testing tools simulate attacker behavior by submitting crafted inputs and analyzing responses. SQLMap is widely recognized as the de facto industry and research baseline due to its comprehensive payload database, support for multiple database management systems (DBMS), and extensive adoption in both academic studies and penetration testing practice (2022). SQLMap automates payload selection, injection point discovery, and data extraction, making it highly effective for in-depth assessments. However, practical reviews note several limitations: it is primarily command-line driven, often requires advanced configuration for complex targets, and can consume substantial computational resources during exhaustive scans. Comparative studies benchmark new methods against SQLMap, positioning it as a technical standard despite limited accessibility for beginners or constrained environments. A common limitation in dynamic testing workflows is inadequate endpoint discovery. Many tools require explicit target URLs and lack mechanisms to locate hidden or parameterized endpoints. Literature highlights complementary discovery techniques—such as robots.txt parsing, sitemap.xml analysis, internal link crawling, and JavaScript/AJAX endpoint enumeration—that can expand coverage. Omitting discovery risks leaving significant portions of the attack surface untested, particularly in educational settings and small organizations. Integrated discovery capabilities are therefore critical for comprehensive vulnerability assessment.

2.4 Research Gaps

Across existing literature, several gaps persist: (i) usability barriers in comprehensive tools, (ii) limited endpoint discovery in lightweight scanners, (iii) interpretability and deployment challenges in ML-based approaches, and (iv) absence of lightweight hybrids that balance efficiency, detection accuracy, and reporting clarity. These gaps motivate the development of the proposed tool.

3. Materials and Research Methodology

This study adopts a design science research (DSR) methodology, suitable for developing and evaluating technical artifacts. The process comprised iterative cycles of problem identification, artifact design, implementation, and evaluation. The identified problem centers on the limited accessibility and discovery coverage of existing SQLi tools despite their technical robustness. Figure 1 illustrates the automated workflow of the proposed system from URL input and endpoint discovery through payload execution, vulnerability analysis, and final report generation.

3.1 System Architecture

The proposed tool follows a modular architecture to ensure extensibility and maintainability. The system comprises six principal components (Figure 1): (i) graphical user interface (GUI), (ii) input validation and sanitization module, (iii) scanning engine, (iv) detection modules, (v) discovery system, and (vi) reporting module. This loosely coupled design enables independent evolution of detection, discovery, and reporting functionalities.

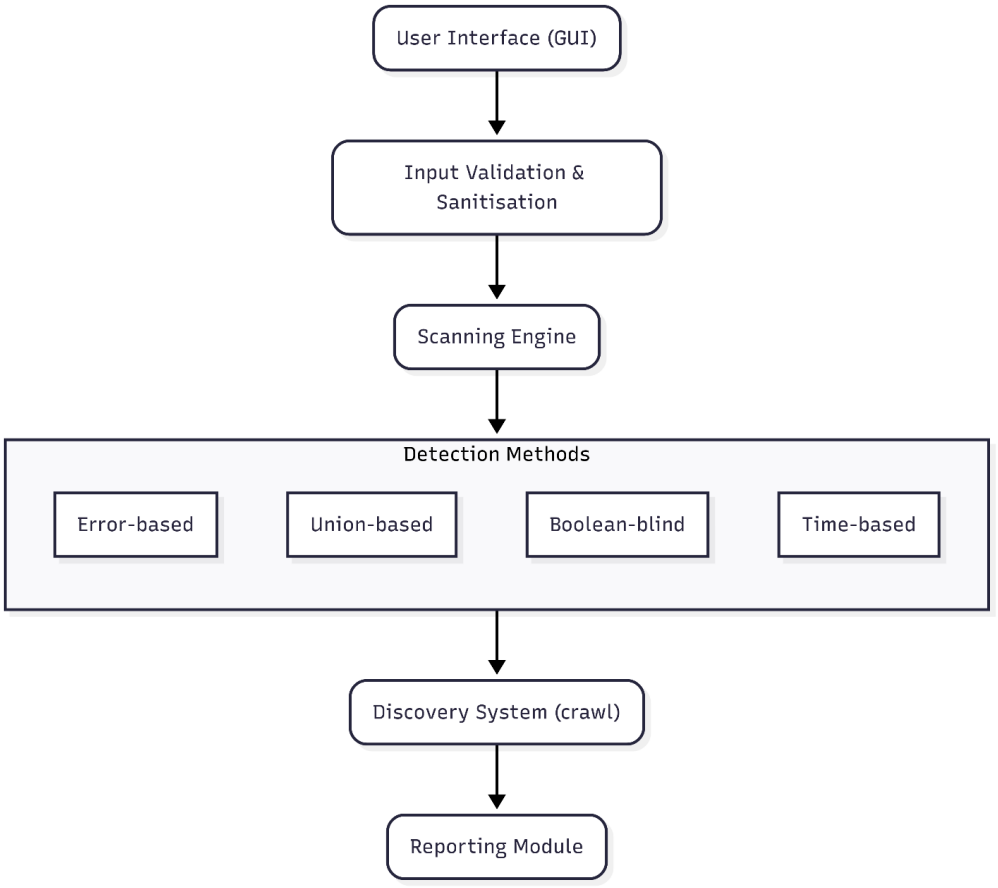

Figure 1: System Architecture Diagram

3.2 Detection Engine

Four classical SQLi detection strategies are implemented: error-based, union-based, boolean-blind, and time-based. Error-based detection injects payloads that provoke database errors, while union-based detection evaluates response manipulation via UNION SELECT queries. Boolean-blind detection infers vulnerability through differential response analysis, and time-based detection employs deliberate delays to confirm conditional execution. These methods are executed sequentially to balance detection coverage and scanning efficiency.

3.3 Discovery System and Reporting

The discovery module expands test coverage by automatically extracting endpoints from robots.txt, parsing sitemap.xml files, and crawling internal hyperlinks using BeautifulSoup4. The reporting module generates structured outputs in HTML, text, and JSON formats, providing executive summaries, evidence, severity classification, and mitigation recommendations.

3.4 Implementation and Evaluation Setup

The system was implemented in Python using Tkinter for the GUI, Requests for HTTP communication, and BeautifulSoup4 for HTML parsing (Figure 2). Evaluation was conducted on DVWA and OWASP Juice Shop, enabling controlled, ethical, and repeatable experimentation. Performance metrics included detection rate, false positive rate, scan time, and endpoint coverage, benchmarked against SQLMap.

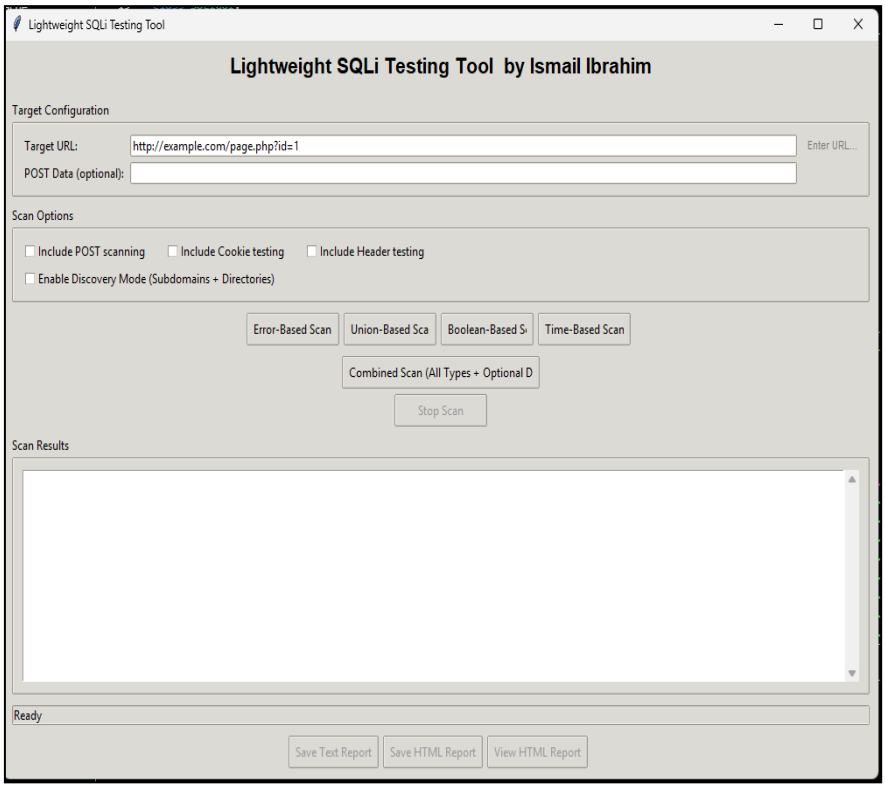

Figure 2: Basic GUI Interface showing main input screen with URL field, detection method checkboxes, and action buttons

4. Results and Analysis

Experimental evaluation demonstrated strong detection performance across all techniques. Error-based detection achieved the highest success rate (93%), followed by union-based (85%), boolean-blind (82%), and time-based detection (79%). Integrated discovery increased endpoint coverage by more than 100%, validating the effectiveness of automated endpoint identification. Comparative analysis against SQLMap revealed that while SQLMap achieved marginally higher coverage due to its extensive payload database, the proposed tool delivered faster scan times, simpler configuration, and superior usability (Table 1 and Figure 3). Resource consumption remained modest, enabling deployment in constrained environments. Structured reporting enhanced interpretability, providing actionable intelligence for developers and learners.

Table 1: Detection Success Rates across Four SQLi Detection Methods

Detection Method	Success Rate (%)	Average Scan Time (s)
Error-based	93	8
Union-based	85	12
Boolean-blind	82	26
Time-based	79	35

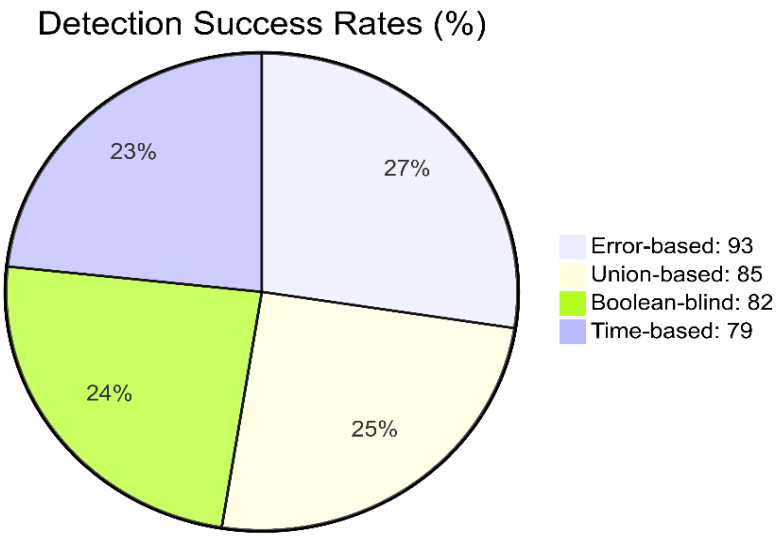

Figure 3: Pie chart showing distribution of SQLI Detection Method Success Rates

The developed tool maintained relatively low computational overhead during testing. Average scan times ranged from 12 to 40 seconds depending on the detection technique and endpoint complexity. Error-based and union-based methods completed more rapidly, while time-based detection introduced higher latency due to deliberate response delays. Compared with SQLMap, the proposed system demonstrated lower memory consumption and faster initialization, making it suitable for lightweight educational and small-scale deployment environments.

4.1 Comparative Analysis with SQLMap

SQLMap served as the baseline for comparative evaluation. While SQLMap achieved higher coverage due to its extensive payload database and advanced exploitation features, the developed tool offered distinct advantages in usability, lightweight performance, integrated discovery, and structured reporting (Table 2). This demonstrates a practical trade-off between accessibility and exhaustive coverage, particularly relevant for educational and small-scale security assessments.

Table 2: Comparative Feature Analysis: Developed Tool versus SQLMap

Feature	Developed Tool	SQLMap (Baseline)
Interface	GUI (Tkinter)	Command-line only
Detection Methods	4 integrated	4+ (comprehensive)
Discovery	Integrated (robots, sitemap, crawling)	Manual input required
Reporting	HTML, Text, JSON	Console output
Resource Usage	Lightweight	Heavy with full scan
Ease of Use	Beginner-friendly	Requires expertise
Configuration Complexity	Minimal	Extensive
Learning Curve	Low	High

While SQLMap offered higher extensibility and deeper exploitation capabilities through its comprehensive payload database, its complexity reinforces the importance of the proposed tool as an accessible alternative (Figure 4). The lightweight resource requirements of the developed tool enable deployment on modest hardware suitable for educational and small-scale assessments. The structured reporting module further distinguishes this system, ensuring results are interpretable and actionable by diverse audiences including non-security specialists.

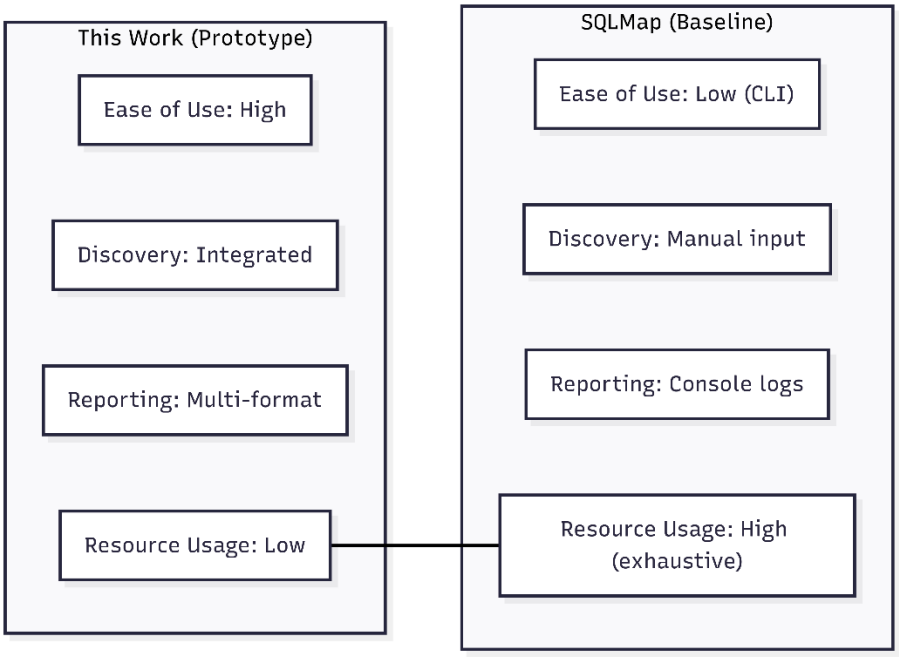

Figure 4: Comparison of Developed Tool and SQLMap across key dimensions (Interface accessibility, Resource usage, Configuration complexity, Reporting quality)

The comparative evaluation confirms that the proposed tool prioritizes accessibility, lightweight deployment, and structured reporting, while SQLMap and related frameworks remain more suitable for advanced exploitation-oriented assessments.

4.2 Reporting Module Output

The reporting module was tested by running complete assessments on DVWA vulnerable endpoints. Generated reports demonstrated the tool's capability to produce actionable outputs (Figure 5).

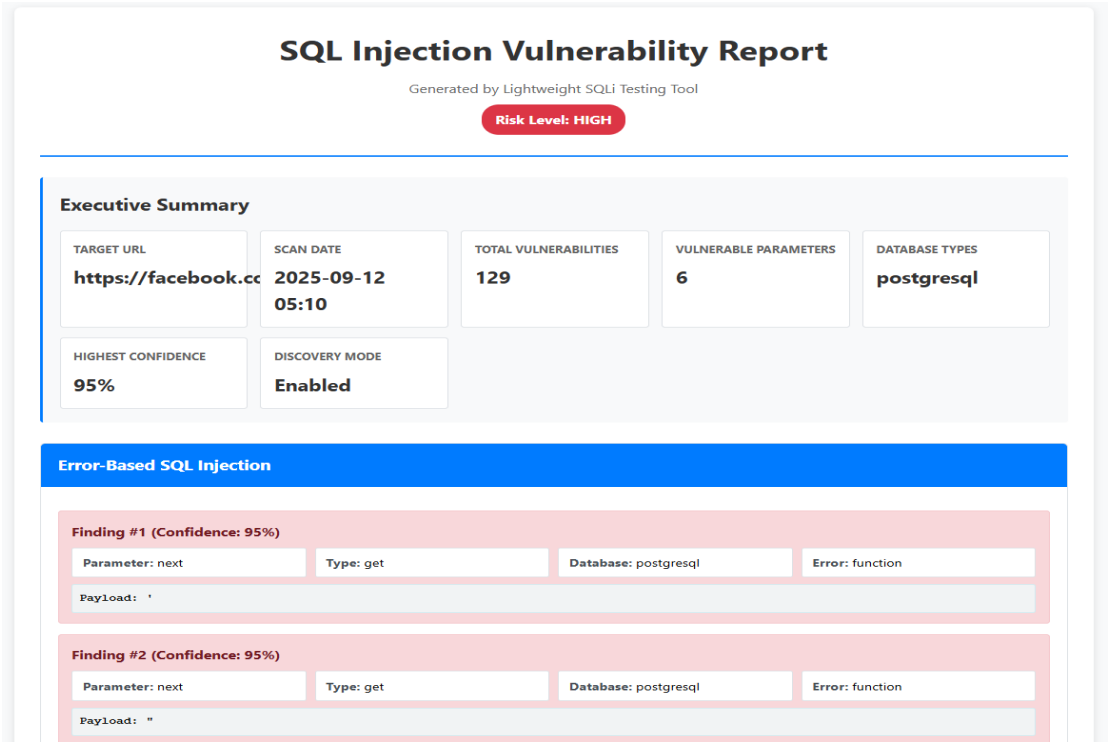

Figure 5: Sample HTML Report showing executive summary section with vulnerability overview, severity distribution, and key findings

The HTML reports provided styled, professional presentations suitable for stakeholders and educational contexts. Text reports offered lightweight alternatives for quick sharing and archival. JSON outputs enabled seamless integration with security information and event management (SIEM) systems and automated workflows. Each report included payload evidence, response indicators, and specific remediation recommendations, providing comprehensive documentation for remediation efforts.

5. Conclusion

This study presents a lightweight, automated SQL injection testing tool designed to enhance usability, detection coverage, and reporting clarity. By integrating four classical detection techniques with automated endpoint discovery and multi-format reporting, the system achieves detection rates between 79% and 93% while doubling assessment coverage. The findings demonstrate that lightweight design does not compromise fundamental detection capabilities, offering a practical alternative for educational, developmental, and small-scale security testing contexts. The results confirm that classical SQLi detection techniques, when systematically integrated, remain highly effective for vulnerability assessment. The modular architecture and GUI-based interface significantly lower operational barriers, enabling broader adoption among students and small organizations. Integrated discovery addresses a major limitation of existing lightweight scanners, while structured reporting enhances practical usability. Although SQLMap remains superior for advanced exploitation, the proposed tool successfully balances technical effectiveness with accessibility and efficiency. Future enhancements include integration of explainable machine learning models, extension to API and microservices architectures, development of standardized benchmarking datasets, and incorporation of SIEM interoperability for automated response workflows. These directions will further strengthen detection capability, interpretability, and real-world applicability.

Acknowledgement

The authors acknowledge the support of the Department of Computer Science and colleagues who provided technical feedback during the development and testing of the system.

Ethical Statement: This research was conducted in controlled and authorized testing environments using deliberately vulnerable applications, including Damn Vulnerable Web Application (DVWA) and OWASP Juice Shop. No unauthorized penetration testing or access to third-party systems was performed during this study. The developed tool was evaluated strictly for academic and defensive cybersecurity purposes. No human participants, personal data, or sensitive organizational information were involved in the research.

Ethical Approval: Not applicable. This study does not involve human participants, animals, or identifiable personal data.

Consent to Participate: Not applicable.

Consent for Publication: Not applicable.

Data Availability: No datasets were generated or analyzed during the current study.

Conflict of Interest: The authors declare no conflict of interest.

Use of Artificial Intelligence Tools: AI-based language models were used as editorial and drafting support tools during manuscript preparation. All content has been critically reviewed, validated, and approved by the authors.

References

Akpojaro, J., Onwudebelu, U. and Aigbe, P., 2014. Unsupervised machine learning techniques for detecting malware applications in wireless devices. Transactions on Machine Learning and Artificial Intelligence, 2(3), pp.20–29. Available at: http://dx.doi.org/10.14738/tmlai.23.206

Baklizi, M., Atoum, I., Abdullah, N., Al-Wesabi, O.A., Otoom, A.A. and Hasan, M.A.-S., 2022. A technical review of SQL injection tools and methods: A case study of SQLMap. International Journal of Intelligent Systems and Applications in Engineering, 10(3), pp.75–85. Available at: https://ijisae.org/index.php/IJISAE/article/view/2141

Cao, H., Tang, J. and Zhu, Q., 2025. Research on lightweight intrusion detection model based on convolutional neural network. In: Proceedings of the 5th International Conference on Computer Network Security and Software Engineering (CNSSE 2025), pp.1–5. https://doi.org/10.1145/3732365.3732375

Hlaing, Z.C.S.S. and Khaing, M., 2020. A detection and prevention technique on SQL injection attacks. In: 2020 IEEE Conference on Computer Applications (ICCA), Yangon, Myanmar, pp.1–6. https://doi.org/10.1109/ICCA49400.2020.9022833

Hsiao, W. and Wang, C., 2023. Detection of SQL injection and cross-site scripting based on multi-model CNN combined with bidirectional GRU and multi-head self-attention. In: 2023 5th International Conference on Computer Communication and the Internet (ICCCI), pp.142–150. https://doi.org/10.1109/ICCCI59363.2023.10210155

Hwang, Y., Le, T., Wardhani, R.W., Putranto, D.S. and Kim, H., 2024. Enhancing SQL injection detection using ensemble learning and boosting models. In: 2024 International Conference on Platform Technology and Service (PlatCon), pp.101–106. https://doi.org/10.1109/PlatCon63925.2024.10830720

Ikramaputra, A., Sudarsono, A. and Ningsih, N., 2025. Enhancing web application firewall with ensemble machine learning to detect SQL injection attacks. In: 2025 International Electronics Symposium (IES), pp.521–526. https://doi.org/10.1109/IES67184.2025.11160726

Imran, H.M., Abdullah, K., Habib, M.A. and Ahmad, M., 2024. Enhanced detection and prevention of SQL injection and cross-site scripting attacks in web applications: Analyzing algorithms and threat modeling approaches. In: 2024 18th International Conference on Open Source Systems and Technologies (ICOSST), pp.1–6. https://doi.org/10.1109/ICOSST64562.2024.10871142

Jahanshahi, R., Doupé, A. and Egele, M., 2020. You shall not pass: Mitigating SQL injection attacks on legacy web applications. In: Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (Asia CCS '20), pp.1–13. https://doi.org/10.1145/3320269.3384760

Khan, L.M., Hoang, H.D., Ngo-Khanh, K., Duy, P.T. and Pham, V., 2025. GSQLi: A GAN-based approach for adversarial SQL injection sample generation against WAF. In: Proceedings of the 11th International Conference on Computing and Artificial Intelligence (ICCAI), pp.788–793. https://doi.org/10.1109/ICCAI66501.2025.00122

Leung, D., Tsai, O., Hashemi, K., Tayebi, B. and Tayebi, M.A., 2024. XploitSQL: Advancing adversarial SQL injection attack generation with language models and reinforcement learning. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), pp.1–8. https://doi.org/10.1145/3627673.3680102

Lu, D., Fei, J., Liu, L. and Li, Z., 2022. A GAN-based method for generating SQL injection attack samples. In: 2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), pp.1827–1833. https://doi.org/10.1109/ITAIC54216.2022.9836726

Messas, G.E., Miani, R.S. and Zarpelão, B.B., 2024. sAIfe: Towards a lightweight threat modeling approach to support machine learning application development. In: Proceedings of the XXIII Brazilian Symposium on Software Quality (SBQS 2024), pp.1–10. https://doi.org/10.1145/3701625.3701640

Onwudebelu, U. and Akpojaro, J., 2012. Electronic security issues: Protecting our electronic life from social engineering attacks. International Journal of Advanced Research in Computer Science, 3(4), pp.1–7.

Onwudebelu, U., Ugah, J.O., Eze, S.E. and Fasola, O.O., 2025. Developing a security-driven hybrid model for detecting malicious URLs. In: Proceedings of the Society for the Advancement of ICT and Comparative Knowledge Conference (SOCTHADICKconf'25), University of Ibadan, Nigeria, 16–19 November.

OWASP, 2021. OWASP Top 10: The ten most critical web application security risks. Open Web Application Security Project. Available at: https://owasp.org/Top10/

Paul, A., Sharma, V. and Olukoya, O., 2024. SQL injection attack: Detection, prioritization and prevention. Journal of Information Security and Applications, 85, p.103871. https://doi.org/10.1016/j.jisa.2024.103871

Schlaubitz, M., Veyisoglu, O. and Rennhard, M., 2025. A2CT: Automated detection of function and object-level access control vulnerabilities in web applications. In: Proceedings of the International Conference on Information Systems Security and Privacy. https://doi.org/10.5220/0013092700003899

Sheykhkanloo, N.M., 2015. SQL-IDS: Evaluation of SQL injection attack detection and classification based on machine learning techniques. In: Proceedings of the 8th International Conference on Security of Information and Networks, pp.258–266. https://doi.org/10.1145/2799979.2800011

Singh, T. and Aksanli, B., 2019. Real-time traffic monitoring and SQL injection attack detection for edge networks. In: Proceedings of the 15th ACM Symposium on QoS and Security for Wireless and Mobile Networks (Q2SWinet '19), pp.1–8. https://doi.org/10.1145/3345837.3355952

Yang, R., Wang, X., Luo, K., Lei, X., Li, K., Xin, J. and Lau, W.C., 2024. SWIDE: A semantic-aware detection engine for successful web injection attacks. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), pp.1–15. https://doi.org/10.1145/3658644.3670304

Zaidan, M.N., Sukarno, P. and Wardana, A.A., 2024. Collaborative detection of SQL injection attacks using SIEM, multi-Wazuh agents, and diverse web application firewalls. In: 2024 5th International Conference on Communications, Information, Electronic and Energy Systems (CIEES), pp.1–6. https://doi.org/10.1109/CIEES62939.2024.10811420