IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Mitigating Website Vulnerabilities: A Comprehensive Comparative Review Of Security Mechanisms And Detection Approaches

Mr. MATHEW SOMY, Miss. LIBINA ROSE SEBASTIAN

Student, Professor

Department of Computer Science and Engineering (Cyber Security)

St Joseph's College of Engineering and Technology, Palai, India

Abstract: Web applications are constantly targeted by attackers due to their role in delivering interactive services and processing sensitive information. Vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), and insecure coding practices can compromise both data integrity and user trust. This review synthesizes findings from representative studies, highlighting detection and mitigation approaches including AI-driven models, static and dynamic analysis, automated remediation tools, and platform-specific scanners. By examining objectives, methods, strengths, and limitations, the paper identifies patterns and practical lessons for embedding robust security practices throughout the software development lifecycle.

Index Terms - Web Application Security, SQL Injection, Cross-Site Scripting, Detection Frameworks, Secure Coding, Artificial Intelligence

1. INTRODUCTION

Modern web applications provide critical functionality for users but are also inherently exposed to security threats. Client-side scripts, if improperly validated or encoded, can be exploited to compromise data and disrupt services. Common attack vectors include XSS, SQLi, cookie and session theft, and other browser-based exploits. Injection techniques remain a leading cause of vulnerabilities, consistently appearing among the most exploited weaknesses in web platforms.Studies of real-world web applications, including e-commerce and banking platforms, indicate that more than 85% of sites contain exploitable flaws. Securing web applications therefore requires a combination of preventive coding, runtime monitoring, and adaptive detection mechanisms.

2. LITERATURE REVIEW

This section presents a synthesis of eight key studies, summarizing their approaches, methodologies, and notable outcomes.

2.1 Comprehensive Web Vulnerability Mitigation Framework

The foundational work develops a taxonomy covering SQLi, XSS, CSRF, and Remote Code Execution (RCE). The authors advocate a defence-in-depth approach spanning all SDLC stages, combining preventive coding, input validation, robust error handling, and encryption for both data in transit and at rest. AI-assisted intrusion detection complements static and dynamic scanning, while DevSecOps practices embed security checks directly into CI/CD pipelines. Ongoing developer training and structured dependency management are emphasized to ensure holistic protection.

2.2 Hybrid Deep Learning for XSS and SQLi Detection

A hybrid CNN-LSTM model addresses XSS and SQLi vulnerabilities by learning both spatial and sequential patterns of potentially malicious input. CNN layers capture local patterns such as keyword structures, while LSTM layers consider context and sequence. This approach demonstrates high accuracy with low false positives, though it requires large, continually updated datasets and integration into live monitoring systems to remain effective.

2.3 Reducing False Positives in SQLi Detection Using Data Mining

Static taint analysis often produces false positives when detecting SQLi vulnerabilities. By combining static analysis with data mining classifiers, this approach discriminates between genuine vulnerabilities and safe code. Evaluated on real-world PHP applications, it significantly reduces false positives compared to tools like RIPS and Pixy. Regular retraining ensures adaptability to new attack patterns.

2.4 Taxonomy of Cross-Site Scripting Attacks

XSS attacks are categorized into stored, reflected, DOM-based, induced, and meta-information XSS. The study highlights evasion techniques such as encoding variations, whitespace insertion, and payload fragmentation, demonstrating why simple filters are often insufficient. Context-aware output encoding and comprehensive input validation remain essential mitigation strategies.

2.5 Automated XSS Remediation Tool: saferXSS

saferXSS statically analyses Java applications to trace data from user inputs to output sinks. Detected vulnerabilities are automatically sanitized using pattern-based encoding. While effective in reducing manual intervention, the tool is limited to Java environments and requires regular updates to address emerging attack vectors.

2.6 Platform-Specific Security for ASP.NET

A hybrid static-dynamic scanner for ASP.NET applications detects SQLi, XSS, cookie poisoning, and session hijacking vulnerabilities. Static analysis examines code and configuration files, while dynamic testing validates exploitability. Although highly accurate for ASP.NET, the methodology lacks portability to other frameworks.

2.7 Hierarchical XSS Defence Model

This model enforces security at design, development, and runtime stages, employing architectural patterns, secure coding, automated scanning, anti-CSRF tokens, and HTTP-only cookies. Effectiveness depends on adherence to best practices, and the model focuses primarily on XSS mitigation.

2.8 MITM and Session Hijacking Vulnerabilities

This work examines transport-layer threats, including MITM and session hijacking attacks. Techniques such as ARP spoofing, DNS poisoning, and SSL stripping are discussed. Recommended mitigations include strict TLS configuration, HSTS enforcement, certificate pinning, and robust session management.

3. COMPARATIVE ANALYSIS

Key observations include:

- Broad frameworks cover multiple vulnerabilities but are resource-intensive.
- Automation improves efficiency but requires careful management of false positives.
- Platform-specific tools offer precision but limited portability.
- Machine learning approaches depend on large, up-to-date datasets.

Table 1: Comparison of Security Approaches for Web Applications

Method	Vulnerabilities	Key Features	Strengths	Limitations
Comprehensive Framework	SQLi, XSS, CSRF, RCE	Secure coding, encryption, AI detection, DevSecOps	Wide SDLC coverage	Resource-intensive
Hybrid CNN-LSTM	SQLi, XSS	Deep learning payload classification	High accuracy, low false positives	Needs large datasets
Taint Analysis + Data Mining	SQLi	Reduces false positives	Improved precision	Computationally heavy
XSS Taxonomy	XSS	Classification of variants	Clear understanding	No automation
saferXSS	XSS	Automated static analysis	Automatic remediation	Java-specific
ASP.NET Scanner	SQLi, XSS, Session	Platform-specific scanning	High accuracy	Framework-specific
Hierarchical XSS Model	XSS	Design-to-runtime controls	Multi-layer protection	XSS-focused
MITM & Session Analysis	MITM, Session	Network-level mitigation	Strengthens transport security	Lacks app-layer measures

The comparative analysis highlights the diversity of techniques developed to secure web applications against common vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Cross-Site Request Forgery (CSRF), Remote Code Execution (RCE), and session-level attacks. The Comprehensive Framework offers the widest protection, integrating secure coding, encryption, AI-based detection, and DevSecOps practices throughout the software lifecycle, though it is resource-intensive. The Hybrid CNN-LSTM approach achieves high accuracy and low false positives for SQLi and XSS detection but depends heavily on large, continuously updated datasets. Taint Analysis combined with Data Mining improves precision by reducing false positives in SQLi detection, yet it is computationally demanding. The XSS Taxonomy study enhances understanding of attack variants, but it lacks automation. saferXSS automates static analysis and remediation in Java applications, although its scope is limited to that platform. The ASP.NET Scanner delivers high accuracy in detecting SQLi, XSS, and session vulnerabilities, but is restricted to ASP.NET environments. The Hierarchical XSS Model applies multi-layered controls from design to runtime for stronger protection, albeit focused mainly on XSS. Finally, the MITM and Session Analysis approach strengthens network-level defenses through secure transport mechanisms but provides minimal coverage at the application layer. Overall, these methods demonstrate a trade-off between breadth of protection, platform dependence, automation, and computational cost.

4. CONCLUSION

Securing web applications demands a layered and continuous approach. Vulnerabilities like SQLi, XSS, CSRF, RCE, MITM, and session hijacking evolve rapidly, requiring integration of preventive coding, AI-assisted detection, automated remediation, and runtime monitoring. Cross-study insights highlight trade-offs between coverage and specificity, automation and accuracy, and generality versus framework dependence. Human factors, including developer training and adherence to best practices, remain vital. Future research should aim to unify these approaches into adaptive, cross-platform security frameworks that balance operational efficiency with comprehensive protection.

5. REFRENCES

[1] R. K. Gupta, S. Sharma, and M. Kumar, "Mitigating Website Vulnerabilities: A Comprehensive Analysis of Security Mechanisms and Emerging Technologies," ICCCIS, pp. 102–110, 2024.
[2] A. Prasad, V. Kumar, and S. K. Singh, "Securing Web Applications Against XSS and SQLi Using Deep Learning," *Int. J. of Computer Applications*, vol. 184, no. 36, pp. 1–8, 2023.
[3] R. D. Patil and S. P. Patil, "An Approach to Minimize False Positive in SQLI Detection Techniques through Data Mining," *ICAC3*, pp. 215–220, 2021.
[4] P. S. Kumar and N. V. Kumar, "Analysis of Cross-Site Scripting Attack," *IJCSIT*, vol. 6, no. 2, pp. 1183–1187, 2022.
[5] S. Mehta and R. Gupta, "Automated Removal of Cross-Site Scripting Vulnerabilities in Web Applications," *IJACSA*, vol. 14, no. 5, pp. 200–207, 2023.
[6] A. Sharma, P. Verma, and M. Singh, "Discovering Security Vulnerabilities and Leaks in ASP.NET Websites," *SmartTechCon*, pp. 450–456, 2022.
[7] R. Nair, V. S. Rao, and P. Thomas, "XSS Vulnerability Assessment and Prevention in Web Application," *Procedia Computer Science*, vol. 185, pp. 712–719, 2021.
[8] K. Singh and M. Chauhan, "Unveiling Vulnerabilities of Web Attacks," *Cyber Security Conf.*, pp. 370–375, 2022.