# A Web Application Firewall Using Reflex Agents

Abbhinav Bharadwaj

December 2025

A

Synopsis

on

# A Web Application Firewall Using Reflex Agents

by

Abbhinav Bharadwaj (2300970100003)

Under the supervision of

Prof. Mukesh Kumar Singh

**Computer Science and Engineering**

**Galgotia's College of Engineering & Technology**

Greater Noida, Uttar Pradesh

India - 201306

Affiliated to

**Dr. A.P.J. Abdul Kalam Technical University**

Lucknow, Uttar Pradesh, India-226031

# ABSTRACT

Web applications form the backbone of modern digital services, making their security a critical concern in contemporary computing environments. Despite significant advances in secure development practices and defensive infrastructure, injection-based attacks such as SQL Injection and Cross-Site Scripting continue to persist as dominant and high-impact threats. Traditional Web Application Firewalls (WAFs), which primarily rely on static rule sets and signature-based detection, struggle to adapt to obfuscated, polymorphic, and zero-day attack patterns.

This project proposes an agent-based Web Application Firewall operating as a reverse proxy, employing reflex agents powered by machine learning techniques to enable real-time inspection and mitigation of malicious web requests. Incoming HTTP traffic is parsed, normalized, and transformed into numerical feature representations, which are evaluated by lightweight classification models to determine the threat level of each request. Based on this evaluation, the reflex agent takes immediate deterministic actions such as allowing or blocking requests, ensuring minimal latency overhead.

The proposed system is designed to balance detection accuracy with performance requirements suitable for high-throughput web environments. Its modular architecture allows for extensibility and integration of additional detection mechanisms in the future. Experimental evaluation using a controlled vulnerable web application demonstrates the effectiveness of the approach in detecting injection attacks while maintaining real-time responsiveness. The results indicate that agent-driven, machine learning–based WAFs can serve as a practical and adaptive complement to traditional rule-based security mechanisms.

**KEYWORDS:** *Reflex Agents, Reverse Proxy, SQL Injection, Cross Site Scripting, Web Application Firewall*

# 1 INTRODUCTION

Web application security is among the fastest evolving domains in computer science. The rapid layering of technologies—from early PHP and AJAX systems with relational databases to modern frameworks such as Node.js and React backed by NoSQL data stores—has not reduced security risks, but instead increased system complexity to a level where vulnerabilities often escape developer oversight [1]. As web applications continue to serve as the backbone of modern digital services, ensuring their security has become a critical challenge rather than a secondary concern.

In recent years, this challenge has intensified with the emergence of artificial intelligence–assisted software development. Prompt Driven Code Generators (PDCGs) and Coding Copilots (CCPs), often collectively referred to as *vibe coding*, enable developers to generate large volumes of code with minimal manual effort [2]. While studies indicate that AI agents are, on average, capable of producing code with security quality comparable to or better than human developers [3], their non-deterministic nature, dependence on prompt quality, and exposure to insecure training data introduce inconsistencies in security performance [4]. As a result, application security increasingly becomes probabilistic rather than guaranteed.

Simultaneously, the scale and accessibility of web applications have expanded dramatically. Businesses now prefer web-based solutions over platform-specific desktop or mobile applications due to lower development and maintenance costs, faster deployment cycles, and broader device compatibility [5]. However, this widespread accessibility also enlarges the attack surface, making web applications a primary target for adversaries. Despite improved security awareness, critical vulnerabilities such as SQL Injection (SQLi) and Cross-Site Scripting (XSS) continue to persist across real-world systems, consistently ranking among the most dangerous software weaknesses.

To address these threats, organizations employ multiple layers of defense, including secure development practices, vulnerability assessments, crowdsourced security testing, and security-focused infrastructure components. Among these, Web Application Firewalls (WAFs) play a central role by acting as reverse proxies that inspect, filter, and block malicious requests before they reach the application server. Traditional WAFs largely rely on signature-based or rule-based detection mechanisms, such as the OWASP Core Rule Set (CRS), which are effective against known attack patterns but struggle with evolving or obfuscated threats.

This project explores an alternative approach to web application security by incorporating machine learning–driven reflex agents into a Web Application Firewall architecture. By focusing on adaptive detection rather than static signatures, the proposed system aims to enhance responsiveness to malicious patterns in incoming requests. As an initial implementation, this work targets SQL Injection and Cross-Site Scripting vulnerabilities,

which remain among the most prevalent and severe threats in the modern web ecosystem. Through this approach, the project seeks to demonstrate how agent-based decision mechanisms can complement traditional defenses and contribute to more resilient web application security.

# 2 LITERATURE REVIEW

## 2.1 INTRODUCTION

SQL Injection (SQLi) and Cross-Site Scripting (XSS) remain among the most severe and persistent web application vulnerabilities. Despite decades of research and mitigation strategies, these attack vectors continue to appear prominently in vulnerability databases and industry reports. As web applications grow in complexity and scale, traditional security mechanisms increasingly struggle to detect obfuscated, polymorphic, and previously unseen attacks. Consequently, a substantial body of research has explored statistical, machine learning, and deep learning–based approaches to strengthen detection and prevention mechanisms.

This chapter reviews existing research related to SQL Injection and XSS detection, reflex agent–based security architectures, and the trade-offs between accuracy and performance in real-time web security systems such as Web Application Firewalls (WAFs).

## 2.2 STATE OF RESEARCH ON SQL INJECTION AND XSS

Several surveys highlight the dominance of learning-based approaches in detecting injection attacks. Habib (2024) conducted a comprehensive review of SQL Injection detection techniques and reported that approximately 90% of the proposed approaches rely on supervised learning, with the remainder exploring unsupervised or hybrid models [6]. Deep learning models, including CNNs and LSTM-based architectures, were shown to achieve high detection accuracy; however, they require large volumes of labeled data and are prone to overfitting in noisy or imbalanced datasets.

In contrast, traditional machine learning models such as Support Vector Machines (SVM), Random Forests, and Decision Trees demonstrate better robustness under limited or noisy data conditions. These models also offer significantly lower inference latency, making them more suitable for deployment in real-time systems.

XSS detection poses additional challenges due to its multi-language nature, involving HTML, JavaScript, CSS, and browser event models. This results in highly dimensional feature spaces and a wide variety of attack payloads. Consequently, classifiers capable of handling sparse and high-dimensional representations have shown greater effectiveness in XSS detection tasks.

## 2.3 REFLEX AGENT–BASED SECURITY ARCHITECTURES

The concept of reflex agents originates from classical artificial intelligence literature. Russell and Norvig define a simple reflex agent as a system that maps percepts directly to actions based on condition–action rules, without maintaining internal state. Such agents operate deterministically and react solely to the current input.

Recent research has explored the application of reflex agent principles to web security. Devi et al. (2025) proposed a reflex agent–based framework using Bidirectional Long Short-Term Memory (BiLSTM) networks to detect SQL Injection, XSS, and XML-based attacks [7]. Their work demonstrated that deep sequence models can outperform classical classifiers in detection accuracy. However, the authors also acknowledged the increased computational cost and latency introduced by deep learning models.

Within a WAF context, the reverse proxy itself can be modeled as a simple reflex agent, where incoming HTTP requests act as percepts, classification results form the condition, and the decision to allow or block constitutes the action. This architectural abstraction enables modular design and extensibility while maintaining conceptual clarity.

## 2.4 CHALLENGES WITH DEEP LEARNING–BASED CONDITIONS

While deep learning models often achieve superior accuracy, their use as decision-making conditions in reflex agents presents several challenges. SQLi and XSS datasets typically contain significant noise, syntactic variation, and class imbalance, which increases the risk of overfitting. Furthermore, deep learning models require large labeled datasets that may not always be available or representative of real-world traffic.

Performance is a critical constraint in WAF deployment. Hybrid architectures such as CNN-BiLSTM have been reported to incur inference latency 300–600 times higher than classical machine learning models such as Random Forests and Gradient Boosted Trees [8]. Such latency can severely degrade server throughput, making these models impractical for high-traffic production environments.

These limitations indicate a clear trade-off between detection accuracy and system performance, particularly for inline security mechanisms that operate on every incoming request.

## 2.5 ACCURACY–PERFORMANCE TRADE-OFF IN WEB SECURITY SYSTEMS

The balance between accuracy and computational efficiency is a recurring theme in the literature. Since database-backed web applications already exhibit inherent latency, security components must introduce minimal additional overhead. As a result, several studies advocate the use of ensemble and margin-based classifiers that offer competitive accuracy with significantly lower inference cost.

Random Forest classifiers have been widely adopted for SQL Injection detection due to their robustness to noisy features and ability to generalize across varied attack patterns. Similarly, Support Vector Machines have shown strong performance in high-dimensional feature spaces, making them well-suited for XSS detection.

Text-based feature extraction methods, particularly Term Frequency–Inverse Document Frequency (TF-IDF), remain a dominant choice for representing injection payloads. TF-IDF enables sparse, interpretable representations while retaining sufficient discriminatory power for classical classifiers.

## 2.6 INDUSTRY AND VULNERABILITY LANDSCAPE

Industry reports corroborate academic findings regarding the persistence of injection vulnerabilities. A 2024 survey by Aikido reported that 6.7% of vulnerabilities in open-source projects and 10% in proprietary software were attributed to SQL Injection [9]. Additionally, over 20% of projects were found vulnerable to SQLi at the onset of security scanning, often across dozens of code locations.

OWASP Top 10 (2025) draft statistics further indicate that XSS accounts for more than 30,000 known CVEs, while SQL Injection exceeds 14,000 CVEs [10]. Although injection vulnerabilities have shifted position in the rankings, their prevalence and potential impact remain significant. Real-world incidents, such as the 2023 breach involving over two million stolen email records through SQL Injection attacks, underscore their continued relevance [11].

## 2.7 LIMITATIONS OF TRADITIONAL WEB APPLICATION FIREWALLS

Traditional WAFs, including ModSecurity with the OWASP Core Rule Set, primarily rely on signature-based and rule-based detection. While effective against known attack patterns, these approaches struggle with zero-day vulnerabilities, polymorphic payloads, and context-dependent exploits.

Negative security models block known malicious patterns but remain vulnerable to novel attacks, whereas positive security models allow only predefined benign traffic at the cost of high maintenance overhead. These limitations have motivated research into adaptive, learning-based WAF architectures that can generalize beyond static rule sets.

## 2.8 SUMMARY AND RESEARCH GAP

The literature indicates that while deep learning approaches offer high detection accuracy, their computational overhead limits practical deployment in real-time WAFs. Classical machine learning models, when combined with effective feature extraction, offer a favorable balance between accuracy and performance. Reflex agent architectures provide a conceptual framework for modular, adaptive security systems but require careful selection of decision mechanisms.

This project builds upon these insights by implementing a reflex agent–based WAF that employs lightweight machine learning models for SQL Injection and XSS detection, aiming to achieve effective protection without compromising system performance.

# 3 PROBLEM FORMULATION

Traditional Web Application Firewall (WAF) systems primarily rely on static, rule-based detection mechanisms such as predefined signatures and regular expressions. While these approaches are effective against known attack patterns, they exhibit inherent limitations in handling obfuscated, polymorphic, and zero-day injection attacks. Updating rule sets in response to newly discovered vulnerabilities is often time-consuming and reactive, leaving systems exposed during the interim period.

Recent research has investigated the use of Machine Learning (ML) and Deep Learning (DL) techniques for detecting injection attacks within adaptive and agent-driven security architectures. Although deep learning models generally achieve higher detection accuracy, they suffer from significant drawbacks in real-time deployment scenarios, including susceptibility to overfitting, dependence on large labeled datasets, limited robustness to noisy inputs, and most critically, high inference latency. Studies indicate that deep learning models can be up to several hundred times slower per request compared to traditional machine learning classifiers, making them impractical for high-throughput reverse proxy systems such as WAFs.

Consequently, there exists a need for a web application security mechanism that balances detection accuracy with real-time performance, while retaining adaptability to evolving attack patterns. This motivates the formulation of a lightweight, agent-based Web Application Firewall that leverages classical machine learning models within a reflex agent framework to enable fast, deterministic decision-making for incoming web traffic.

## 3.1 PROBLEM STATEMENT

To design and implement an agent-based Web Application Firewall operating as a reverse proxy, which employs reflex agents powered by machine learning classifiers to detect and mitigate injection attacks, specifically SQL Injection and Cross-Site Scripting, in real time while maintaining acceptable system performance.

## 3.2 OBJECTIVES

The primary objectives of this project are as follows:

1. To study and analyze existing rule-based and learning-based approaches for detecting SQL Injection and Cross-Site Scripting attacks.
2. To design a reflex agent–based architecture suitable for real-time web traffic inspection in a reverse proxy setup.

3. To implement lightweight machine learning models, namely Support Vector Machine (SVM) and XGBoost, for detecting SQL Injection and XSS attacks.

4. To integrate the detection models into modular middleware components capable of taking immediate allow or block decisions.

5. To evaluate the proposed Web Application Firewall using a vulnerable test application (DVWA) to demonstrate effectiveness and practical applicability.

6. To assess the trade-off between detection accuracy and performance in comparison to heavier deep learning–based approaches.

# 4 METHODOLOGY

The methodology adopted in this project follows a modular, agent-oriented approach to web application security. The proposed system is designed as a reverse proxy–based Web Application Firewall (WAF) that inspects incoming HTTP requests in real time and applies machine learning–driven reflex agents to detect and mitigate injection attacks. The overall workflow is structured to ensure minimal latency while maintaining effective detection capability.

## 4.1 OVERALL SYSTEM ARCHITECTURE

The WAF operates as an intermediary between the client and the destination web server. All incoming requests are intercepted by the proxy layer, analyzed for malicious content, and conditionally forwarded or blocked based on the classification outcome. The architecture is composed of independent processing stages, enabling extensibility and isolation of responsibilities.

The major stages of the methodology are:

1. Request interception through a reverse proxy
2. Request parsing and normalization
3. Feature extraction using text vectorization
4. Machine learning–based classification
5. Reflex agent decision and enforcement

## 4.2 REVERSE PROXY–BASED TRAFFIC INTERCEPTION

The reverse proxy acts as the first point of contact for all client requests. It captures HTTP/1.1 traffic and preserves request semantics, including HTTP methods, headers, query parameters, and payloads. Hop-by-hop headers are excluded to maintain protocol consistency.

By operating inline with application traffic, the proxy ensures that every request undergoes inspection before reaching the backend server. This design allows the WAF to function transparently without requiring modifications to the protected application.

## 4.3 REQUEST PARSING AND NORMALIZATION

Each intercepted request is parsed to extract relevant components such as:

- URL path and query parameters
- HTTP headers
- Request body (form data or raw payload)

The extracted inputs are normalized into a unified textual representation. Normalization includes decoding encoded characters, removing redundant formatting, and concatenating relevant fields into a consistent structure. This step reduces variability caused by syntactic differences and improves the effectiveness of downstream feature extraction.

## 4.4 FEATURE EXTRACTION USING TF-IDF

To convert textual request data into numerical form, Term Frequency–Inverse Document Frequency (TF-IDF) vectorization is employed. TF-IDF captures the relative importance of tokens within a request while down-weighting commonly occurring terms across the corpus.

This vectorized representation produces sparse, high-dimensional feature vectors that are well-suited for classical machine learning classifiers. TF-IDF is selected due to its low computational overhead, interpretability, and proven effectiveness in text-based security classification tasks.

## 4.5 MACHINE LEARNING–BASED DETECTION

Two independent detection pipelines are implemented, each tailored to a specific class of vulnerability.

### *4.5.1 XSS Detection Using Support Vector Machine*

Cross-Site Scripting payloads often exhibit high dimensionality due to the mixture of HTML, JavaScript, and browser event syntax. A linear Support Vector Machine (SVM) is employed to classify XSS payloads in the TF-IDF feature space.

The SVM is trained on an open dataset and augmented with synthetically generated benign samples to address class imbalance. The linear kernel is selected to ensure fast inference and scalability in real-time request processing.

### *4.5.2 SQL Injection Detection Using XGBoost*

SQL Injection detection is handled using an XGBoost classifier, which leverages gradient-boosted decision trees to model complex token interactions while maintaining low inference latency.

The model is trained on an open dataset of SQL queries and augmented with synthetically generated benign queries. Tree depth and regularization parameters are constrained to prevent overfitting and ensure stable performance under noisy input conditions.

## 4.6 REFLEX AGENT DECISION MECHANISM

The decision-making component of the WAF follows a simple reflex agent model. For each request:

1. The request classification result acts as the condition.
2. A predefined action is mapped to the condition.

If a request is classified as malicious, the reflex agent blocks the request and optionally logs the event for monitoring. If the request is classified as benign, it is forwarded to the destination server without modification. This deterministic condition–action mapping ensures predictable behavior and minimal processing delay.

## 4.7 LOGGING AND MONITORING

Detected events are logged using a lightweight data store to enable monitoring and analysis. Logged data includes request metadata, classification outcome, and timestamps. A simple Security Information and Event Management (SIEM) interface is provided to visualize system status and security events.

## 4.8 EVALUATION ENVIRONMENT

The proposed WAF is evaluated using the Damn Vulnerable Web Application (DVWA), which provides a controlled environment for testing SQL Injection and XSS attacks. DVWA enables safe validation of detection accuracy and system behavior without exposing real-world applications to risk.

## 4.9 SUMMARY

This methodology combines a reverse proxy architecture with lightweight machine learning models and reflex agent decision-making to address injection attacks in real time. By prioritizing low-latency classification and modular design, the proposed system aims to achieve an effective balance between security, performance, and extensibility.

# 5 ACTIVITY CHART

Table 5.1: Activity Chart

| **Week** | **Activities** |
|---|---|
| Week 1 | Problem definition, planning, and research |
| Week 2 | SQL Model Planning |
| Week 3 | SQL Model Implementation |
| Week 4 | XSS Model Planning |
| Week 5 | XSS Model Implementation |
| Week 6 | Integration with AIOHTTP |
| Week 7 | Development of SIEM |
| Week 8 | Final review, documentation, and submission |

# REFERENCES

[1] E. Iannone, R. Guadagni, F. Ferrucci, A. D. Lucia, and F. Palomba, "The secret life of software vulnerabilities: A large-scale empirical study," *IEEE Transactions on Software Engineering*, vol. 49, no. 1, pp. 44–63, Jan. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9672730

[2] A. Mohsin, H. Janicke, A. Wood, I. H. Sarker, L. Maglaras, and N. Janjua, "Can We Trust Large Language Models Generated Code? A Framework for In-Context Learning, Security Patterns, and Code Evaluations Across Diverse LLMs," *arXiv preprint arXiv:2406.12513*, 2024.

[3] O. Asare, M. Nagappan, and N. Asokan, "Is github's copilot as bad as humans at introducing vulnerabilities in code?" *Empirical Software Engineering*, vol. 28, no. 6, p. 129, 2023. [Online]. Available: https://doi.org/10.1007/s10664-023-10380-1

[4] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, W. Ye, Y. Zhang, Y. Chang, P. S. Yu, Q. Yang, and X. Xie, "A survey on evaluation of large language models," *arXiv preprint arXiv:2307.03109*, 2023. [Online]. Available: https://arxiv.org/abs/2307.03109

[5] Amplitude Labs, "2022 App vs. Website Trend Report," https://amplitude.com/guides/2022-app-vs-website-report, April 2022, accessed: [Insert Date You Accessed the Report].

[6] U. Habib, "A survey on implication of artificial intelligence in detecting sql injections," https://www.researchgate.net/publication/378496266_A_Survey_on_Implication_of_Artificial_Intelligence_in_detecting_SQL_Injections, 2023, preprint or article hosted on ResearchGate, accessed 2025-12-15.

[7] S. D. R. N Anitha Devi *et al.*, "Web application security enhancement through automated form analysis and ai-driven attack detection," *International Research Journal on Advanced Engineering Hub*, vol. 3, no. 4, pp. 1460–1464, 2024. [Online]. Available: https://www.researchgate.net/publication/393988516_Web_Application_Security_Enhancement_Through_Automated_Form_Analysis_and_AI-Driven_Attack_Detection

[8] K. A.-A. Vandit R Joshi *et al.*, "Hybrid ai intrusion detection: Balancing accuracy and efficiency," *Sensors*, vol. 25, no. 24, p. 7564, 2025. [Online]. Available: https://www.mdpi.com/1424-8220/25/24/7564

[9] M. Jackson, "The state of sql injections," https://www.aikido.dev/blog/the-state-of-sql-injections, 2024.

[10] “A05:2025 Injection - OWASP Top 10:2025,” https://owasp.org/Top10/2025/A05_2025-Injection/, OWASP Foundation, 2025, oWASP Top 10:2025 Release Candidate 1, accessed 2025-12-15.

[11] SecurityWeek, “Millions of user records stolen from 65 websites via sql injection attacks,” *SecurityWeek*, February 2024, accessed: 2025-12-15. [Online]. Available: https://www.securityweek.com/millions-of-user-records-stolen-from-65-websites-via-sql-injection-attacks/