JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

WAF Rule Development and Experimental Evaluation for Detecting and Blocking Web-Based Attacks

[1]Naveena Koli, [2]Sufiyan Patel
[1]MSc CS specialization in Cybersecurity, [2]Assistant Professor
[1,2]Department of Advanced Computing
[1,2]Nagindas Khandwala College,
[1,2]University of Mumbai, Maharashtra, India

Abstract - Web applications are frequently exposed to security threats such as SQL injection, Cross-Site Scripting (XSS), command injection, path traversal, iframe injection, and other malicious HTTP requests. Web Application Firewalls (WAFs) provide an additional security layer by inspecting incoming requests and identifying patterns associated with known web-based attacks. However, effective WAF protection depends significantly on the quality and coverage of its security rules. This research presents the development and experimental evaluation of a rule-based Web Application Firewall using Mod Security integrated with the OWASP Core Rule Set (CRS) and a set of customized security rules. The proposed system is designed to inspect HTTP requests, detect predefined attack patterns, block malicious requests, and record security events for further analysis. Custom rules were developed to detect specific attack categories, including JavaScript protocol attacks, iframe injection, command injection, path traversal, and sensitive file access, while OWASP CRS rules were used to identify common attacks such as SQL injection and Cross-Site Scripting. An experimental web application was developed to generate both legitimate and malicious requests for evaluating the behaviour of the implemented rules. The detected events were logged and processed to provide information such as attack type, rule identifier, severity, attack category, request URL, payload, and blocking status. The experimental evaluation demonstrates how customized rule development can complement standard WAF rules and provide a practical approach for detecting and blocking different web-based attacks. The study also highlights the importance of rule testing, logging, and analysis in improving the effectiveness of WAF-based web application protection.

Keywords *- Web Application Firewall, ModSecurity, OWASP Core Rule Set, WAF Rule Development, Web Attack Detection, SQL Injection, Cross-Site Scripting, Command Injection, WAF Evasion, Web Application Security.*

1. INTRODUCTION

Web applications are frequently exposed to security threats because attackers can manipulate HTTP requests to deliver malicious inputs. Common attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), command injection, and other web-based attacks can exploit vulnerabilities in application inputs. Web Application Firewalls (WAFs) are widely used as an additional security layer to inspect HTTP traffic and identify potentially malicious requests using predefined security rules.

The effectiveness of a WAF largely depends on its ruleset and its ability to correctly identify malicious patterns. WAFs inspect request parameters and compare them with security rules designed to detect known attack patterns. When a request matches a rule, the WAF can block the request before it reaches the web application [1]. However, attackers may modify or manipulate malicious payloads to avoid detection, making systematic testing and evaluation of WAF rules important.

Previous research has demonstrated that WAFs can also be affected by evasion techniques caused by differences in request parsing and interpretation between the WAF and the protected web application. Such inconsistencies can potentially allow malicious

requests to bypass WAF inspection [1]. Research on automated WAF testing has further explored approaches for generating and modifying attack inputs to evaluate the effectiveness of WAF detection mechanisms.

In this work, a controlled experimental environment is developed for WAF rule development and evaluation using Apache2, ModSecurity, OWASP Core Rule Set (CRS), PHP, Python, and MariaDB. The proposed system combines existing security rules with custom ModSecurity rules for detecting selected web-based attacks, including JavaScript protocol attacks, iframe injection, command injection, and path traversal.

The developed system allows different attack payloads to be submitted through controlled web interfaces and evaluates how the WAF identifies and blocks them. ModSecurity audit logs are processed and stored in a database, allowing detected attacks to be categorized and monitored through a security dashboard. The study therefore provides an experimental framework for analysing WAF rule behaviour and detection effectiveness. The objective of this research is to develop and experimentally evaluate WAF rules for detecting and blocking representative web-based attacks, while providing detailed logging and monitoring of the resulting security events.

2. LITERATURE REVIEW

2.1 Existing Tools and Methodologies

Web Application Firewalls (WAFs) protect web applications by inspecting HTTP/HTTPS requests and detecting malicious traffic. Traditional WAFs primarily use predefined signatures and rules to identify attacks such as SQL Injection, Cross-Site Scripting (XSS), command injection, and path traversal. **ModSecurity** with the **OWASP Core Rule Set (CRS)** is a widely used open-source approach that also supports custom rule development.

Recent WAF methodologies extend rule-based detection through machine learning, fuzzing, adversarial testing, and automated evasion analysis. Techniques such as mutation-based fuzzing and reinforcement learning have been used to generate modified attack payloads and evaluate WAF robustness. These approaches show that effective WAF evaluation requires testing both known attacks and variations designed to bypass existing rules.

2.2 Related Research Papers

Existing research has investigated WAF rule effectiveness, intelligent attack detection, and automated evasion techniques. Studies on ModSecurity and OWASP CRS have demonstrated the importance of evaluating existing rules and developing additional rules to improve attack coverage [1], [2]. Machine-learning approaches have been explored to enhance ModSecurity-based detection and improve resistance to adversarial inputs [3], [4]. Other studies have examined WAF architectures and application-layer protection mechanisms [5].

Adversarial testing has become an important research direction. WAF-A-MoLE uses mutation-based fuzzing to generate adversarial payloads for WAF evaluation [6], while AdvSQLi investigates automatically generated SQL injection payloads against WAF services [7]. Reinforcement-learning-based approaches have also been proposed to discover WAF bypass techniques for attacks such as SQL injection and XSS [8], [11]. Further research has examined automated rule testing, protocol-level evasion, and deep-learning-based WAF detection.

3. PROBLEM STATEMENT

Web Application Firewalls (WAFs) are widely used to protect web applications from attacks such as SQL Injection, Cross-Site Scripting, and other malicious requests. However, the effectiveness of rule-based WAFs depends on the coverage and accuracy of their detection rules. Recent research has identified gaps in OWASP CRS rules and demonstrated that custom rule development can improve detection of previously missed attack payloads [1]. Adversarial testing has also shown that modified payloads can be designed to evade WAF detection [2], while protocol-level parsing discrepancies can create additional WAF bypass vulnerabilities [3]. Therefore, there is a need for a practical WAF implementation that develops and evaluates custom rules against multiple attack categories and systematically records detected threats. This research addresses this problem by implementing a ModSecurity-based WAF with OWASP CRS, custom detection rules, attack logging, and controlled security testing.

4. OBJECTIVES OF THE RESEARCH

The primary objective of this research is to develop and evaluate a rule-based Web Application Firewall using ModSecurity and OWASP Core Rule Set (CRS) for detecting and blocking common web-based attacks. The study aims to develop custom detection rules to address gaps in existing rule coverage and improve the identification of malicious requests [1]. It further aims to evaluate the effectiveness of the developed rules against different attack categories and modified payloads, considering

the possibility of WAF evasion through payload manipulation [2]. The research also focuses on recording detected attacks and analyzing WAF performance in terms of detection and blocking effectiveness.

5. PROPOSED METHODOLOGY AND SYSTEM ARCHITECTURE

5.1 Overall Methodology

The proposed methodology implements a rule-based Web Application Firewall using Apache, ModSecurity, and OWASP CRS. Incoming HTTP requests are intercepted and inspected against predefined CRS rules and application-specific custom rules. Requests identified as malicious are blocked and recorded, while legitimate requests are forwarded to the web application. The generated security logs are processed and stored in a database for attack classification, severity analysis, and monitoring through an administrator dashboard. Custom rules are additionally tested using different attack payloads to evaluate their detection and blocking effectiveness. ModSecurity supports request inspection and rule-based actions, while OWASP CRS provides generic protection against common web attacks.

5.2 System Architecture

The proposed architecture consists of five major layers: User, Web Server/WAF Layer, Rule Detection Layer, Application Layer, and Monitoring Layer. The client sends HTTP requests to the Apache server, where ModSecurity intercepts and 814nalyses the traffic. The request is evaluated using OWASP CRS and custom detection rules. Malicious requests are blocked and logged, whereas legitimate requests are forwarded to the protected web application. Security events are subsequently processed and stored in the database and displayed through the administrator dashboard.

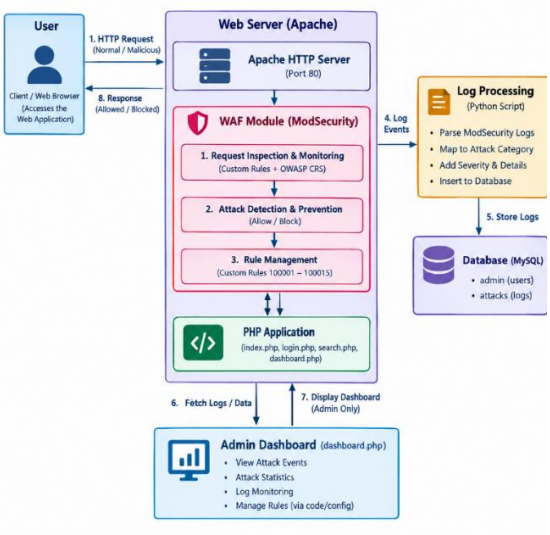

Fig 5.2.1 System Architecture

5.3 System Components

The proposed WAF system consists of several integrated components that work together to monitor and protect the web application. The client or user accesses the application through a web browser and sends HTTP requests to the Apache web server. Apache forwards the incoming requests to the ModSecurity WAF module, which inspects the requests using the OWASP Core Rule Set (CRS) and the project's custom security rules. The WAF detects attack patterns such as SQL injection, XSS, command injection, path traversal, PHP code injection, and other malicious requests, and either blocks or allows the request. The PHP web application, consisting of pages such as index.php, login.php, search.php, and dashboard.php, processes requests that are permitted by the WAF. Security events generated by ModSecurity and Apache are stored in log files and processed by the Python log-processing module, which extracts relevant information, identifies attack categories, assigns severity, and prepares the records for storage. The processed information is stored in the MySQL database, which maintains administrator information and attack/request records. Finally, the administrator dashboard retrieves the stored information and presents attack statistics, request logs, severity, attack categories, and security events, providing centralized monitoring of the WAF system.

6. IMPLEMENTATION DETAILS

The proposed WAF was implemented in an Ubuntu-based environment using Apache2, ModSecurity, OWASP Core Rule Set (CRS), PHP, Python, and MariaDB. Apache2 acts as the web server, while

ModSecurity is configured as the WAF inspection engine to intercept and analyze incoming HTTP requests. OWASP CRS provides predefined rules for common web attacks, and additional custom rules are implemented to detect attack patterns not sufficiently covered by the default rules.

The protected web application was developed using PHP and configured to operate behind the WAF. Incoming requests are inspected before being forwarded to the application. When a request matches a security rule, ModSecurity denies the request and generates a security event; legitimate requests are allowed to proceed. Custom rules were implemented for attack categories including SQL Injection, Cross-Site Scripting (XSS), JavaScript protocol injection, iframe injection, path traversal, command injection, and sensitive file access.

Security events generated by ModSecurity and Apache are processed using a Python-based log-processing module. The module extracts relevant information such as timestamp, IP address, attack type, URL, payload, rule ID, severity, and request status. The processed records are stored in a MariaDB database and presented through an administrator dashboard for monitoring and analysis. The implementation therefore provides an integrated process for request inspection, attack detection, blocking, logging, classification, storage, and security monitoring.

The implementation follows the rule-based WAF model in which requests are inspected before reaching the protected application, while custom rule development provides additional application-specific detection capability.

7. EXPERIMENTAL EVALUATION

The implemented WAF was evaluated in a controlled local environment to measure its ability to detect and block common web-based attacks. Test requests containing SQL Injection, Cross-Site Scripting (XSS), JavaScript protocol injection, iframe injection, path traversal, command injection, and sensitive file access were generated and submitted to the protected application. Both standard and modified attack payloads were tested to examine the effectiveness of the custom rules and their ability to identify variations of malicious requests. For each test case, the WAF response, triggered rule ID, attack category, severity, and logging status were recorded. Legitimate requests were also tested to verify that normal application traffic was not incorrectly blocked. The generated ModSecurity audit logs were processed and compared with the records stored in the database and displayed on the administrator dashboard. This evaluation provides a practical assessment of the proposed WAF's detection capability and the effectiveness of the developed custom rules.

7.1 Evaluation Setup

The experimental evaluation was conducted in a controlled local environment using Ubuntu, Apache2, ModSecurity, OWASP Core Rule Set (CRS), PHP, Python, and MariaDB. The WAF was deployed in front of the PHP-based web application to inspect incoming HTTP requests before they reached the application. Test cases were prepared for common web attacks, including SQL Injection, Cross-Site Scripting (XSS), JavaScript protocol injection, iframe injection, path traversal, command injection, and sensitive file access. Both legitimate requests and malicious payloads were submitted to evaluate detection and false-positive behaviour. For each request, the response status, triggered rule ID, attack type, severity, and logging information were recorded. The collected results were subsequently analysed to determine the detection and blocking effectiveness of the proposed WAF.

7.2 Test Cases

The WAF was tested against multiple categories of web-based attacks to evaluate the coverage of the predefined and custom rules. The test cases included SQL Injection, Cross-Site Scripting (XSS), JavaScript protocol injection, iframe injection, path traversal, command injection, and sensitive file access. Legitimate requests were also included to verify that normal application traffic was not incorrectly blocked.

Test Case	Attack Type	Expected
TC01	SQL Injection	Block
TC02	Cross Site Scripting	Block
TC03	Javascript Protocol Injection	Block
TC04	Iframe Injection	Block
TC05	Path Traversal	Block
TC06	Command Injection	Block

TC07	Sensitive File Access	Block

Table 7.2.1 Test Case

7.3 Results and Observations

The experimental evaluation demonstrated that the implemented ModSecurity WAF successfully inspected incoming requests and applied the configured security rules. Malicious requests matching OWASP CRS or custom rule patterns were denied, while legitimate requests were forwarded to the protected application. Each detected security event generated corresponding log information containing details such as the request URL, payload, triggered rule ID, attack type, severity, and status. The results also showed that custom rules can extend the detection coverage of the default rule set for application-specific attack patterns.

Test Case	Attack Type	Expected	Result
TC01	SQL Injection	Block	Pass
TC02	Cross Site Scripting	Block	Pass
TC03	Javascript Protocol Injection	Block	Pass
TC04	Iframe Injection	Block	Pass
TC05	Path Traversal	Block	Pass
TC06	Command Injection	Block	Pass
TC07	Sensitive File Access	Block	Pass

Table 7.3.1 Test Results

7.4 Discussion

The evaluation indicates that a combination of OWASP CRS and custom ModSecurity rules provides an effective approach for detecting common web attacks. The use of custom rules improves the ability of the WAF to identify application-specific malicious patterns, while centralized logging enables systematic monitoring and analysis of detected requests. However, rule-based WAF protection remains dependent on rule coverage and may be affected by modified or obfuscated payloads. Previous research has similarly demonstrated that adversarial payloads and protocol-level differences can be used to evade WAF detection [1], [2]. Therefore, continuous rule testing and refinement are necessary to maintain effective protection.

8. CONCLUSION

This research presented the design and implementation of a rule-based Web Application Firewall using ModSecurity and OWASP Core Rule Set. Custom rules were developed to detect multiple web attack categories, and the implemented system was evaluated using controlled malicious and legitimate requests. The system integrates request inspection, attack blocking, security logging, log processing, database storage, and administrator monitoring. The experimental approach demonstrates the practical applicability of custom WAF rule development for improving web application security. The study also highlights the importance of continuous testing against modified attack payloads to identify potential limitations in rule-based protection.

9. FUTURE SCOPE

Future work can focus on improving the WAF through automated rule generation, machine-learning-based anomaly detection, and larger and more diverse attack datasets. Advanced evasion techniques can also be incorporated to evaluate the robustness of the developed rules. Further improvements may include real-time log analysis, automated rule updates, and integration with threat-intelligence sources. These enhancements can help reduce dependence on manually maintained rules and improve the ability of the WAF to detect previously unseen attacks.

10. REFERENCES

[1] Anuvarshini Mk, Kommuri Sai Suhitha Bala, S. Sonti, Jevitha Kp, “An Empirical Study on the Evaluation and Enhancement of OWASP CRS (Core Rule Set) in ModSecurity,” 2026.

[2] M. Curipallo Martínez, A. Guevara-Vega, A. Reyes Narváez, G. Raura, and H. Barba Molina, “Web Application Protection Optimization Through Coraza WAF: Performance Assessment Against OWASP Risks in Reverse Proxy Configurations,” 2025.

[3] Christian Scano, Giuseppe Floris, Biagio Montaruli, Luca Demetrio, "ModSec-Learn: Boosting ModSecurity with Machine Learning," 2025.
[4] G. Floris et al., "ModSec-AdvLearn: Countering Adversarial SQL Injections With Robust Machine Learning," 2025.

[5] Dr. A. Shaji George and A. S. Hovan George, "A Brief Study on the Evolution of Next Generation Firewall and Web Application Firewall," 2021.

[6] L. Demetrio, A. Valenza, G. Costa, and G. Lagorio, "WAF-A-MoLE: Evading Web Application Firewalls Through Adversarial Machine Learning," 2020.

[7] Z. Qu, X. Ling, T. Wang, X. Chen, S. Ji, and C. Wu, "AdvSQLi: Generating Adversarial SQL Injections Against Real-World WAF-as-a-Service," 2024.

[8] M. Amouei, M. Rezvani, and M. Fateh, "RAT: Reinforcement-Learning-Driven and Adaptive Testing for Vulnerability Discovery in Web Application Firewalls," 2023.

[9] D. Appelt, D. C. Nguyen, and L. Briand, "Behind an Application Firewall, Are We Safe from SQL Injection Attacks?" 2015.

[10] Rizki Agung Muzaki, Obrina Candra Briliyant, Maulana Andika Hasditama, Hamzah Ritchi, "Improving Security of Web-Based Application Using ModSecurity and Reverse Proxy in Web Application Firewall," 2020.