

Automatically benchmarking biases in ML-based wafs

Questa è la versione preprint della seguente opera:

Original

Automatically benchmarking biases in ML-based wafs / Hamza, X., Costa, G.. - 16900:(2027), pp. 5-21.
(ARES 2026 International Workshops Linköping, Sweden 24-27/08/2026) [10.1007/978-3-032-35576-8_1].

Availability:

This version is available at: 20.500.11771/44878

Publisher:

Springer Nature

Published

DOI:10.1007/978-3-032-35576-8_1

Terms of use:

This publication is made accessible in accordance with the terms for deposit in the institutional repository, as defined by the IMT School for Advanced Studies Lucca's Open Access Policy.
(https://library.imtlucca.it/sites/default/files/regolamento-policy-open-access-imtlib_0.pdf).

Si prega di consultare le pagine informative dell'editore relative alle politiche di autoarchiviazione.

(Article begins on next page)

Automatically Benchmarking Biases in ML-based WAFs

Hamza Sajjad¹[0000–0002–6954–938X]✉ and Gabriele Costa¹[0000–0002–9385–3998]

IMT School for Advanced Studies¹ hamza.sajjadahmed@imtlucca.it
gabriele.costa@imtlucca.it

✉ Corresponding author

Abstract. Web Application Firewalls (WAFs) are the first line of defense, inspecting incoming HTTP requests to detect *malicious* payloads. Distinguishing between malicious and benign payloads is complex, and universal criteria do not exist. In recent decades, Machine Learning (ML) has emerged as an alternative to rule-based WAFs, using trained models instead of manually crafted rules. However, the reliability of ML-WAFs is questionable, mainly due to recurrent biases and the lack of a standardized assessment framework. This paper introduces a benchmarking framework to systematically identify and measure biases in ML-WAFs, evaluating dataset diversity, training validity, and evaluation robustness. We conduct an in-depth case study on WafBrain and further validate the proposed framework by applying it to the Barracuda WAF, a widely deployed commercial product. Such biases determine inflated performances in controlled environments, but they make WAFs fragile in adversarial settings.

Keywords: Web Application Firewall · Web Security · Machine Learning.

1 Introduction

In today’s digital era, the rise of web applications is driven by widespread Internet access and diverse online services [5]. However, these applications are often exposed to various logical and technical vulnerabilities [1], require robust security measures to protect sensitive data and ensure reliable operation [15]. The diversity of programming languages, platforms, and scripting environments complicates efforts to defend against evolving threats [28]. Web Application Firewalls (WAF) have become integral to modern network infrastructures, protecting web applications exposed to the Internet [23]. In addition to blocking known attacks, WAFs play a critical role in maintaining the availability and integrity of web services by filtering malicious traffic before it reaches backend applications. A WAF typically inspects incoming HTTP requests and analyzes their *payloads*, which consist of the data or URL parameters transmitted by clients. Malicious requests are dropped to prevent attacks from reaching their targets [24, 22]. Accurate differentiation between benign and malicious payloads is crucial for the effective-

ness of a WAF. Traditionally, rule-based WAFs (RB-WAFs) rely on predefined patterns such as regular expressions to flag and block malicious payloads [4].

Modern RB-WAFs employ thousands of continuously updated rules maintained by experts [2]. The main limitation of RB-WAFs is their inability to quickly adapt to new or slightly modified attacks, requiring extensive manual rule updates [20]. Even minor changes, such as altering a single character, can evade detection. This sensitivity to obfuscation techniques highlights the inherent challenge of maintaining comprehensive rule sets against rapidly evolving attack strategies. To address this limitation, ML approaches have emerged as an alternative, enabling systems to generalize beyond predefined patterns and detect previously unseen attack payloads. Montes et al. [18] demonstrated the applicability of deep learning for web application attack detection, while Tadhani et al. [31] proposed a deep learning framework for detecting XSS and SQL injection attacks. Shaheed et al. [27] incorporated machine learning and feature engineering into a web application firewall to improve attack detection performance. In a broader cybersecurity context, Mohamed [17] reviewed recent advances in AI- and ML-based threat detection techniques. However, evaluating ML-WAFs remains difficult due to inconsistent training and testing setups. Furthermore, publicly available datasets often differ significantly in terms of attack types, traffic characteristics, and labelling procedures. Variations in data preprocessing, feature extraction, and train-test splitting strategies further complicate fair comparisons between proposed approaches [13]. As a result, reported performance metrics may not accurately reflect a model’s effectiveness in real-world deployment scenarios. Moreover, studies such as [3] identify several recurring biases that affect ML-based security systems, which we show in Table 1.

In this paper, we present a standardized testing methodology and a benchmarking framework to evaluate ML-WAFs under adversarial scenarios. The design and implementation of our framework start with a systematic review of the biases reported in the literature. We establish testing conditions for the biases summarized in Table 1, enabling a systematic assessment of ML-WAFs. For each bias, we propose a specific testing methodology. By combining built-in tests with support for pluggable modules, our methodology enables an extensible framework that can be expanded or modified to allow broader and deeper testing. Upon completion, the framework generates a report assessing the effectiveness of the target ML-WAF and exposing its potential biases. We demonstrate our framework through a case study on *WafBrain*¹, an open-source ML-WAF. Furthermore, we show its applicability by testing *Barracuda WAF*², a commercial WAF also relying on ML. Remarkably, even if we could access neither the training dataset nor the internals of Barracuda, our framework identified several undetected payloads, revealing a potential bias. Both the payloads and the report were shared with Barracuda through a responsible disclosure process. This work does not propose a new ML-WAF or novel detection techniques. Instead, it introduces a systematic, bias-aware benchmarking framework for evaluating existing

¹ <https://github.com/BBVA/waf-brain/tree/master>

² <https://www.barracuda.com/>

Table 1. Common Biases in Machine Learning Model Evaluation [3]

Biases	Summary
Dataset Preparation	
Cherry Pick	Selecting non-representative data that does not reflect real-world distributions leads to overly optimistic and misleading evaluation results.
False Flag	Incorrect or noisy labels introduce systematic errors that cause models to overfit and reduce their ability to generalize.
Training	
Data Dilemma	The presence of data leakage or irrelevant features during training allows the model to learn unintended patterns, reducing its reliability in unseen scenarios.
Pattern Pitfall	Models may capture spurious correlations instead of meaningful relationships, which degrades performance when deployed in real-world environments.
Preset Prejudice	Optimizing model parameters using test data biases the evaluation process and results in inflated and unreliable performance estimates.
Validation	
Foundation Fumble	The absence of proper baseline models prevents fair comparison and makes it difficult to assess the true effectiveness of a proposed approach.
Metric Muddle	Using inappropriate evaluation metrics, especially with imbalanced datasets, can distort performance interpretation and hide critical weaknesses.
Rate Roulette	Ignoring class imbalance leads to misleading performance metrics, often overestimating the model’s effectiveness on underrepresented classes.
Deployment	
Lab Trap	Evaluating models only in controlled environments fails to capture real-world variability, leading to unexpected performance drop after deployment.
Risk Radar	Overlooking adversarial or evolving threats results in models that are not robust, reducing their trustworthiness in practical applications.

ML-WAFs, aiming to help researchers and practitioners assess their reliability and real-world limitations rather than optimize individual models. Summing up, our framework answers three fundamental questions for system administrators:

1. Is my ML-WAF more effective than traditional RB-WAFs?
2. How reliable is my ML-WAFs under real-world adversarial conditions?
3. Do I have additional benefits from integrating RB and ML WAFs in my network?

This paper is organized as follows. Section 2 reviews the related literature, while Section 3 describes the design of our assessment framework. Section 4 presents the experimental setup and results for WafBrain, addresses the research questions, and discusses the limitations. Section 5 demonstrates the applicability of the proposed framework to a commercial ML-WAF under black-box access constraints. Finally, Section 6 concludes the paper.

2 Related Work

Identifying biases in ML is crucial, as they can significantly influence algorithmic outcomes and lead to misleading conclusions. Researchers have developed methods tailored to specific bias types and datasets to mitigate these effects. Mehrabi et al. [14] provide a comprehensive categorization of biases in ML pipelines and survey fairness-aware approaches applied across diverse domains such as healthcare, criminal justice, and finance, highlighting the broad relevance of bias-aware practices.

In the context of security applications, Daniel et al. [3] review the most common biases in ML-based security systems, demonstrating how these biases can distort performance measurements and produce unrealistic results. Skewed or mislabeled data are frequent sources of bias that can undermine model reliability. Mishra et al. [16] show that sampling methods such as SMOTE and Resampling help reduce data skew, improving classification accuracy. Similarly, Fredriksson et al. [11] explore challenges in data labeling and propose practical mitigation strategies, while Northcutt et al. [21] reveal that label errors are widespread and can cause lower-capacity models to outperform higher-capacity ones when datasets are noisy.

Class imbalance is another critical factor affecting model performance. Balla et al. [6] introduce novel sampling methods to alleviate imbalance, and Sen et al. [26] emphasize the importance of robust evaluation methodologies for real-world imbalanced datasets. Ceschin et al. [8] stress that model evaluation should focus on real-world impacts rather than solely on metrics, advocating for checklists to prevent common modeling mistakes. White [33] warns against data snooping, which can produce spurious results and overstate model effectiveness. Spurious correlations, arising from learning unreliable or non-generalizable features, present another challenge in ML systems [30]. Hays et al. [12] highlight the importance of transparency in data collection and labeling processes, cautioning against shortcuts that artificially inflate reported performance. Toyama et al. [32] examine how model misspecification and unreliable parameter estimates can impact reinforcement learning outcomes, further illustrating the consequences of overlooked biases.

In web security, biases in datasets are often underestimated, despite their potential to compromise ML-based tools. Doupé et al. [10] propose strategies to identify and mitigate these biases, improving detection accuracy and robustness. Technical vulnerabilities such as SQL injection (SQLi), cross-site scripting (XSS), remote file inclusion (RFI), and local file inclusion (LFI), often caused

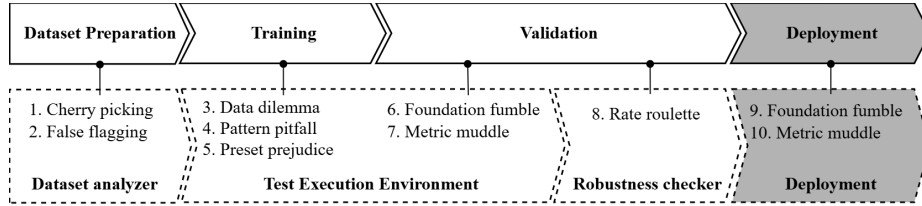


Fig. 1. Relationship between the ML lifecycle (upper layer), the bias categories considered in this study, and the corresponding assessment components of the proposed framework (lower layer), illustrating how bias categories are mapped to framework components.

by poor coding practices or outdated components, remain significant threats to web application integrity, emphasizing the importance of reliable, bias-aware ML systems in practical security contexts. To the best of our knowledge, existing studies focus on new WAF models under controlled conditions, but no prior work provides a reproducible, bias-aware benchmarking framework for ML-WAFs.

3 Assessment Framework

Our ML-WAF assessment framework adopts a three-stage approach, illustrated in Figure 1, to systematically evaluate ML-WAFs while addressing key dataset, system, and evaluation biases.

The first stage focuses on the adequacy of training and testing datasets. We quantify dataset diversity using Shannon entropy, originally proposed as a measure of information uncertainty [29]. Namdari and Li [19] provide a comprehensive review of entropy measures for uncertainty quantification, while Bereziński et al. [7] demonstrate the usefulness of entropy-based metrics for network traffic characterization and anomaly detection. Saraiva [25] further discusses the broad applicability of Shannon entropy across data analysis domains. To complement this assessment, we perform ground-truth validation using rule-based systems such as Snort³ and OWASP ModSecurity⁴. This dual approach ensures that the data are both representative and accurately labeled, reducing the risk of bias-induced overfitting or misestimation of model performance.

The second stage executes dynamic runtime testing in a standardized Test Execution Environment (TEE) built with Docker⁵. The TEE delivers realistic network traffic to the target WAF and integrates mutational fuzzing [9] to generate diverse, semantically valid variations of requests that probe hidden training or system biases. Both ML and RB WAFs process the same inputs, and adaptive feedback from the ML-WAF guides the fuzzing to craft increasingly evasive payloads. This stage enables robust evaluation of the WAFs’ resilience under

³ <https://www.snort.org/>

⁴ <https://owasp.org/www-project-modsecurity-core-rule-set/>

⁵ <https://www.docker.com/>

adversarial scenarios and mitigates biases related to limited or overly controlled evaluation setups.

In the third stage, we compute multiple standardized evaluation metrics such as precision-recall curves, ROC curves, and Matthews Correlation Coefficient (MCC) values to address class imbalance and other statistical biases. Aggregating results from all stages, the framework generates a comprehensive performance report highlighting detection effectiveness, generalization, and fairness. Each stage of the framework is explicitly designed to expose and measure specific bias categories summarized in Table 1.

Biases associated with deployment-specific conditions or fundamental ML security limitations remain outside the scope of this study. While both are important, our work focuses on controlled, reproducible assessment and abstracts from the operational and deployment-specific aspects unique to each installation. Nevertheless, the framework lays the foundation for evaluating these biases in future studies.

4 Experimental Evaluation

We implemented and evaluated our three-stage assessment framework using Docker containers on a Windows 11 Lenovo ThinkPad with an Intel[®] Core[™] i7-1165G7 (2.80 GHz). The setup integrates all components required for data, runtime, and performance assessments, ensuring reproducibility and comparability. The complete implementation, configuration files, and experimental results are publicly available at <https://github.com/imhamzasajjad/TEE-WAFs-Framework>.

4.1 Data Assessment

Entropy Analysis and Statistical Testing We used the HTTP Params dataset⁶, a parameter-focused version of the CSIC 2010 dataset⁷ that preserves original labels while exposing HTTP fields commonly used in ML-WAF research.

To quantify byte-level variability, Shannon entropy was computed over 64-byte chunks for each request, balancing sensitivity to subtle anomalies with statistical contrast. Average entropy was 5.576 bits/byte for normal and 5.629 bits/byte for anomalous traffic (Figure 2), with anomalous requests showing slightly higher standard deviation (0.2981 vs. 0.2713 bits/byte, Figure 3). Statistical tests confirmed significant differences (t-test $p=0.000017$; Mann-Whitney U $p=0.0065$), though the small mean difference indicates limited overall dataset heterogeneity.

These results suggest that, at the byte level, anomalous payloads are only marginally more variable than normal traffic, potentially limiting ML-WAF generalization. Combined with missing HTTP methods, this constrained diversity

⁶ <https://www.kaggle.com/datasets/evg3n1j/httpparamsdataset>

⁷ <https://www.kaggle.com/datasets/ispangler/csic-2010-web-application-attacks>

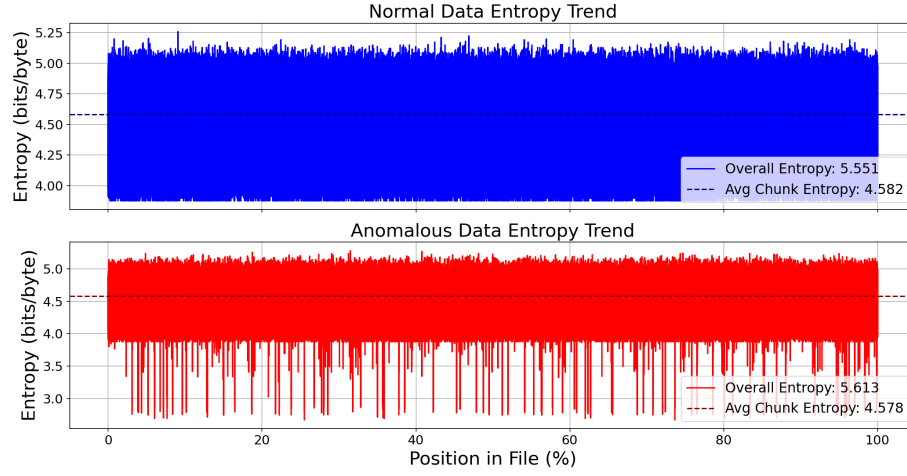


Fig. 2. Comparison of chunk-wise entropy for normal vs. anomalous HTTP requests, showing limited diversity in anomalous payloads.

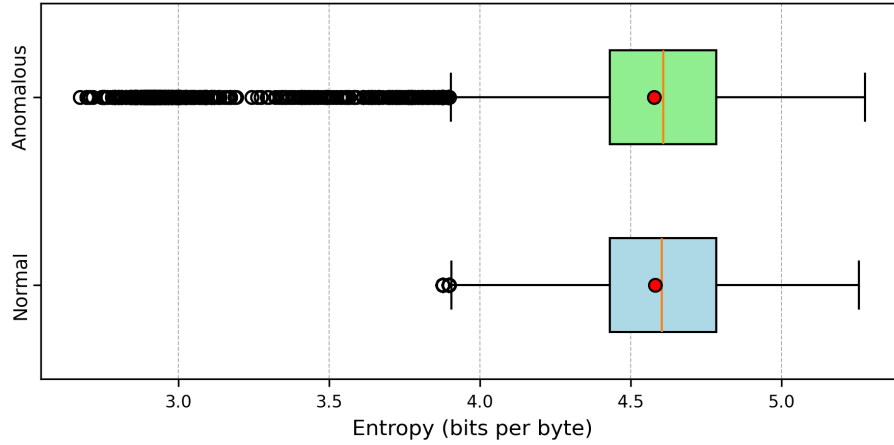


Fig. 3. Boxplot of chunk-wise entropy for normal and anomalous traffic, showing higher variability in anomalous requests.

represents a source of bias. To further assess labeling quality, all requests were processed through an RB-WAF (Snort/ModSecurity), allowing quantification of discrepancies between dataset labels and rule-based detection outcomes.

Labeling Validation Using Rule-Based WAF As a second step in the data assessment, we validated the labels of the web application requests dataset.

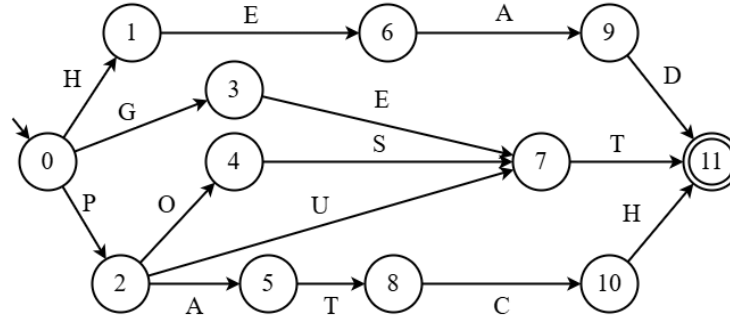


Fig. 4. FSA representing HTTP method usage in the CSIC 2010 dataset. Nodes correspond to HTTP methods, and edges represent observed transitions between request states.

All HTTP requests were processed through an RB-WAF (Snort/ModSecurity), which applies expert-defined rules to detect malicious requests. We then compared the RB-WAF outcomes with the dataset’s original labels and computed the ratio of requests where the labels disagreed with the RB-WAF predictions. A higher discrepancy ratio indicates potential labeling errors or inconsistencies, which could introduce bias during ML-WAF training and evaluation, while a low ratio suggests the labels are largely consistent with established detection rules.

4.2 Runtime Assessment

The runtime assessment stage evaluates the robustness of ML-WAFs under realistic and adversarial traffic conditions. We first analyzed the dataset for protocol coverage using finite state automata (FSAs), focusing on the HTTP `method` field (Figure 4). The FSA allowed us to verify which methods were present in the dataset and identify any missing but valid HTTP methods. For the CSIC 2010 dataset, only three methods appeared: GET (43,088 occurrences), POST (17,580 occurrences), and PUT (397 occurrences), while others, such as DELETE, HEAD, OPTIONS, PATCH, TRACE, and CONNECT, were entirely absent. This under-representation highlights a potential dataset bias, as real-world web traffic involves a wider variety of methods, and missing patterns could affect the generalization of ML models.

To test WAF performance under adversarial conditions, we employed mutational fuzzing using WAF-A-MOLE [9] to generate diverse, semantically valid variations of HTTP requests. Starting from an original payload selected from the dataset, the fuzzer applies incremental mutations to produce a sequence of payloads that increasingly deviate from the original input, challenging the WAFs’ detection capabilities and enabling evaluation of robustness against evasion techniques. In this stage, WafBrain is used as the target ML-based WAF under evaluation. Experiments were conducted in a Docker-based Test Execu-

tion Environment (TEE) to ensure reproducibility, isolation, and portability. The TEE consisted of four containers (Figure 5):

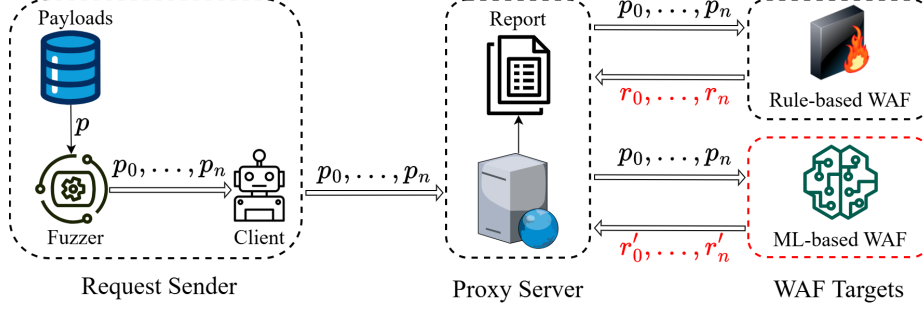


Fig. 5. Docker-based Test Execution Environment for runtime WAF evaluation, consisting of Client, Server, ML-WAF, and RB-WAF containers, enabling isolated and reproducible testing.

1. **Client:** Selects an initial payload P_0 from the dataset and generates mutated variants $P_1, \dots, P_n$ through successive fuzzing rounds, sending each payload to the server and logging all interactions.
2. **Server:** Receives requests and forwards them to both WAFs, recording their predictions and outcomes.
3. **ML-WAF:** Processes each payload using a trained machine learning model to detect malicious requests.
4. **RB-WAF:** Applies expert-defined rules (ModSecurity) for detection, serving as a baseline for comparison.

For each payload P_i , the corresponding responses r_i from the RB-WAF and r'_i from the ML-WAF are collected and logged. By comparing these response sequences across increasing fuzzing depths, we evaluate detection effectiveness, resilience to evasion, and the influence of dataset-related biases. This systematic logging enables detailed analysis of true positives, false positives, and false negatives across different payload types and fuzzing rounds.

4.3 Performance Assessment

We evaluated both RB- and ML-WAFs using multiple metrics: accuracy, precision, recall, F1 score, MCC, and specificity. Experiments used 50 payloads per run, with fuzzing iterations varying between 5 and 30 rounds. Each configuration was repeated 10 times, and the average values were computed to ensure reliable comparisons.

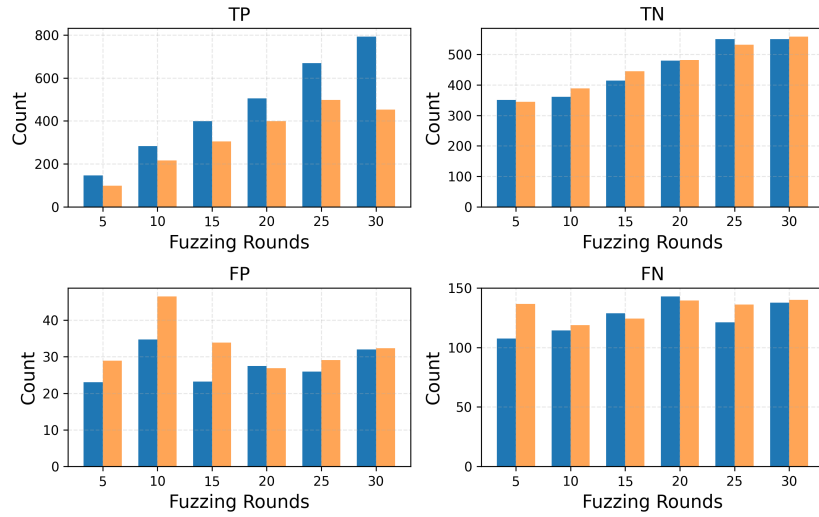


Fig. 6. Comparison of TP, TN, FP, and FN for RB-WAF (blue) and ML-WAF (orange) under identical evaluation conditions.

- **RB-WAF:** The rule-based WAF consistently outperformed the ML-WAF across all rounds. Accuracy ranged from 0.842 to 0.865, with precision between 0.866 and 0.910, recall from 0.853 to 0.897, and F1 scores between 0.859 and 0.883. MCC values (0.680–0.728) and specificity (0.821–0.874) confirmed its ability to effectively distinguish malicious from benign traffic.
- **ML-WAF:** In contrast, the ML-WAF showed higher variability. Accuracy dropped from 0.756 in earlier rounds to 0.557 in later rounds, with precision between 0.639 and 0.874, recall from 0.512 to 0.684, and F1 scores from 0.568 to 0.756. MCC values (0.128–0.535) and specificity (0.617–0.862) also fluctuated, indicating challenges in adapting to more complex or mutated traffic patterns.

Figures 6 and 7 compare detection behavior and performance metrics, highlighting RB-WAF’s stability and consistent superiority over the more variable ML-WAF under fuzzed traffic. These results show how dataset limitations and model biases affect ML-WAF performance, underscoring the need for standardized and reproducible evaluation. Combining RB and ML detection yielded only marginal gains (Figure 8). The RB Correct–ML Incorrect cases increased from 22.47% to 36.17%, confirming RB-WAF robustness, while fewer ML Correct–RB Incorrect cases indicate limited detection of novel attacks.

4.4 Potential Biases

Applying the proposed framework to the dataset and WAFBrain, we identified several potential biases, summarized in Figure 9. Following our classification

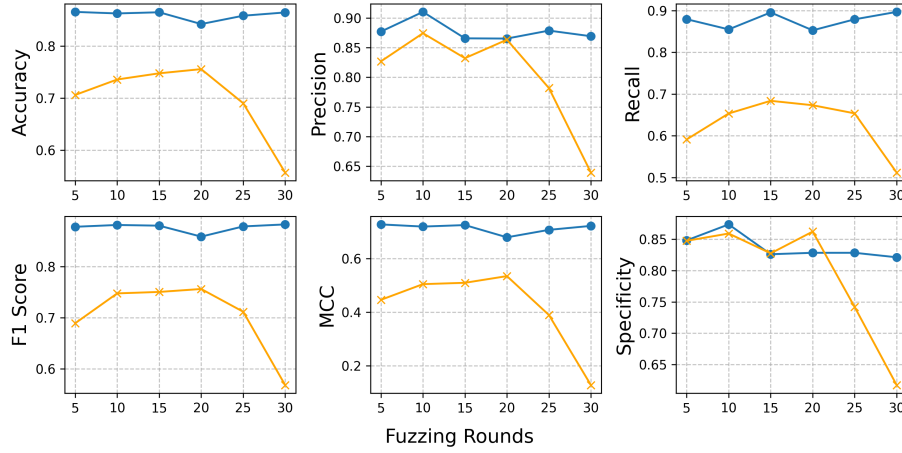


Fig. 7. Performance comparison of RB-WAF (blue) and ML-WAF (orange) under fuzzing-based adversarial testing. The figure contains six subplots showing different evaluation metrics plotted against fuzzing rounds.

criteria, each bias was labeled as present, partially present, or not present. Below, we briefly describe the rationale behind each classification.

- B1. **Cherry Picking:** The data diversity analysis in Section 4.1 indicates partial presence of this bias. Although anomalous traffic shows higher variability (standard deviation 0.2981 vs. 0.2713 bits/byte, a $\sim 9.88\%$ increase), overall entropy values remain close (5.629 vs. 5.576 bits/byte, a $\sim 0.95\%$ difference). This suggests limited diversity in anomalous samples, which may not fully capture real-world attack variability and could reduce generalization to unseen attacks.
- B2. **False Flagging:** This bias is classified as present. Using ModSecurity with OWASP CRS as a proxy ground truth (Section 4.1), we observed a consistently high discrepancy where the RB-WAF was correct while the ML-WAF was incorrect, ranging from 16.22% to 36.17% across fuzzing rounds (Figure 8). This persistent gap indicates unreliable ML predictions, likely influenced by unstable or imperfect labeling, as discussed in Section 4.3.
- B3. **Data Dilemma:** Regular expression and FSA-based analysis revealed limited protocol coverage (Figure 4). Only GET (43,088), POST (17,580), and PUT (397) methods were present, with PUT appearing exclusively in anomalous traffic. All other standard HTTP methods (e.g., DELETE, HEAD, OPTIONS, PATCH) were absent. Such underrepresentation suggests dataset incompleteness and possible data snooping, reducing realism and robustness.
- B4. **Pattern Pitfall:** Evaluated using mutational fuzzing within the TEE, results show that the ML-WAF occasionally relies on spurious correlations learned during training. This behavior indicates partial presence of Bias 4

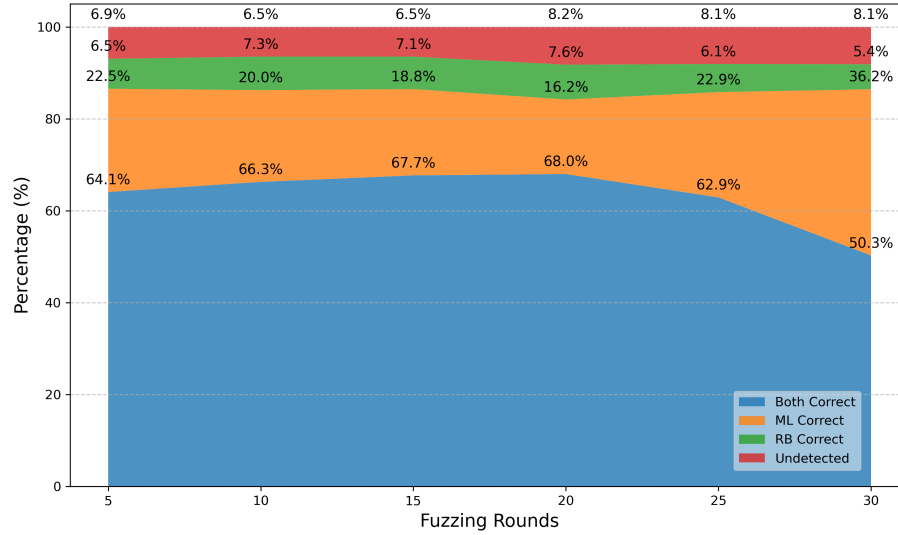


Fig. 8. Combined performance across fuzzing rounds. Blue: RB-WAF correct, green: both correct, orange: ML-WAF correct, red: both incorrect.

and highlights challenges in generalizing to diverse real-world inputs (Section 4.2).

- B5. **Preset Prejudice:** Using the TEE with fuzzing, we tested whether model parameters were overly influenced by prior data. While diverse inputs mitigated complete overfitting, residual false negatives and occasional false positives indicate partial presence of this bias, suggesting incomplete adaptation to novel payloads (Figure 6).
- B6. **Foundation Fumble:** This bias was identified in the original ML-WAF development due to the lack of standardized baseline comparisons. By benchmarking against an RB-WAF in a controlled Docker-based environment, our evaluation avoids this issue; however, the ML-WAF’s consistent underperformance across F1 and MCC scores (Section 4.2) highlights the consequences of insufficient baseline grounding during development.
- B7. **Metric Muddle:** Bias 7 was exposed through TEE-based fuzzing, which revealed weaknesses masked by reliance on accuracy alone. High false negatives and unstable F1 and MCC values under increasing fuzzing rounds indicate that simplistic metrics overestimated real-world effectiveness, confirming the presence of this bias (Section 4.2).
- B8. **Rate Roulette:** Class imbalance (e.g., 28,000 normal vs. 15,088 anomalous GET requests) contributed to this bias. While RB-WAF performance remained stable, the ML-WAF showed declining accuracy (0.7555 to 0.5570), fluctuating recall (0.5117–0.6837), and unstable MCC (0.1280–0.5347). High but unstable precision combined with low recall indicates difficulty detecting minority classes, confirming the presence of Bias 8 (Section 4.3).

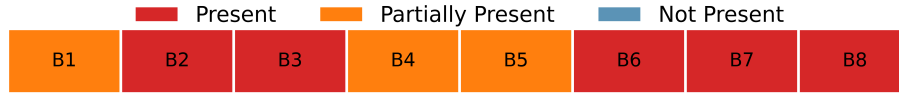


Fig. 9. Distribution of detected bias in WAFBrain during evaluation under adversarial traffic conditions.

4.5 Addressing the Questions

Our framework quantifies biases through threshold-based metrics, enabling systematic evaluation of detection accuracy, robustness, and consistency across experiments. These metrics form the basis for addressing the following research questions.

Q1: Is my ML-WAF more effective than traditional RB-WAFs? Overall, RB-WAF outperforms ML-WAF across all metrics (Figure 7), showing higher and more stable accuracy against fuzzing. However, as seen in Figure 8, the ML-WAF occasionally detects payloads that are missed by the RB-WAF, indicating potentially complementary behavior.

Q2: How reliable is my ML-WAF under real-world adversarial conditions? ML-WAF performance drops sharply with fuzzed or adversarial payloads (Figure 7), highlighting poor robustness. In contrast, RB-WAF degrades more slowly, confirming greater reliability under adversarial stress.

Q3: Do I have additional benefits from integrating RB and ML WAFs in my network? Integration yields only marginal improvement (5–8% gain; Figure 8) while inheriting ML-WAF’s instability and false positives. For the time being, the hybrid option offers limited benefit without significant noise reduction or robustness improvements.

5 Applicability to Commercial ML-WAFs

To demonstrate the applicability of the proposed framework under realistic deployment constraints, we evaluated a commercial ML-based WAF, *Barracuda WAF*⁸, using the same Test Execution TEE and adversarial testing methodology employed throughout this study. As Barracuda WAF is a proprietary system, we had no access to its training data, internal models, or feature representations. The evaluation therefore, relied exclusively on black-box testing through controlled traffic injection and observation of runtime behavior. HTTP requests derived from the HTTP Params dataset were used solely as input traffic for fuzzing and adversarial runtime assessment. Consequently, data assessment tests were not applicable, while runtime robustness and adversarial evaluation remained fully supported by the framework.

⁸ <https://www.barracuda.com/products/application-protection/web-application-firewall>

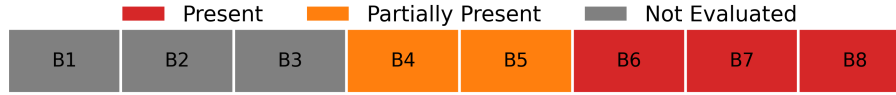


Fig. 10. Distribution of detected bias in Barracuda WAF during evaluation under adversarial traffic conditions.

We evaluated the ML-WAF using both benign and malicious payloads under adversarial conditions, varying fuzzing rounds from 5 to 30. Across all experiments, the ML-WAF processed a total of 7,050 requests. Overall, 20.16% of requests were misclassified. The majority of these errors were false positives (18.10%), which primarily impact availability by incorrectly blocking benign traffic. More critically, 2.06% of all requests were false negatives, corresponding to malicious payloads that bypassed detection. Although fewer in number, these false negatives pose a significant security risk because they allow attack traffic to reach the protected application.

As shown in Figure 10, the first three biases (B1–B3), which relate to dataset construction and labeling, cannot be assessed for Barracuda WAF due to the lack of access to its training data and are therefore marked as not evaluated. Bias 4 (Pattern Pitfall) is partially present, as mutational fuzzing revealed that certain payload transformations could bypass detection, indicating reliance on fragile or spurious patterns. Bias 5 (Preset Prejudice) is also partially present, as repeated fuzzing exposed residual false negatives, suggesting limited adaptation to novel inputs under black-box constraints. Bias 6 (Foundation Fumble) is present and observable through comparison with the RB-WAF baseline: although the ML-WAF produced fewer false negatives (145 vs. 743), this improvement came at the cost of substantially higher false positives (1,276 vs. 342), demonstrating how the absence of baseline-grounded evaluation could lead to an inflated perception of effectiveness. Bias 7 (Metric Muddle) is present, as accuracy-oriented assessment would obscure the security impact of false negatives, which reached up to 6.97% under attack-only conditions. Finally, Bias 8 (Rate Roulette) is present, as class imbalance and adversarial traffic caused unstable detection behavior, highlighting the sensitivity of the ML-WAF to skewed traffic distributions. The results of evaluation indicate that biases similar to those observed in open-source ML-based WAFs can also manifest in commercial solutions. In particular, the presence of false negatives under adversarial fuzzing suggests potential blind spots in the learned detection logic.

Responsible Disclosure All false-negative payloads and corresponding reports were shared with the vendor via responsible disclosure, conducted with their explicit permission. Aggregated results and representative payloads are hosted in a repository to support transparency and reproducibility, without revealing proprietary details.

6 Conclusion

This paper presented a general and reproducible framework for benchmarking ML-based WAFs under realistic conditions. The main goal is to provide the community with a unified environment where ML-WAFs can be systematically tested and evaluated. We implemented the framework and applied it to Waf-Brain, showing that the performance advantage of ML-WAFs over rule-based approaches is marginal under adversarial evaluation. We further demonstrated the applicability of the framework to a commercial ML-WAF (Barracuda) in a black-box setting, where it successfully identified false negatives indicative of potential blind spots. By capturing detailed execution logs and generating structured evaluation reports, the framework provides deeper insights into system behavior beyond aggregate performance metrics. Overall, our results highlight the critical impact of dataset bias, labeling quality, and evaluation methodology on reported performance, underscoring the need for standardized and transparent assessment practices. In addition to validating the framework’s effectiveness, the experiments emphasize the importance of addressing biases at every stage of the ML lifecycle, including data collection, labelling, testing, and evaluation. In the future, our focus will be on expanding the framework with additional testing methodologies and applying it to a broader range of ML-WAFs, including industrial and commercial-grade systems.

7 Acknowledgments

This work was partially supported by RDS PTR 25-27 Project 2.1 cybersecurity.

Disclosure of Interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Alhogail, A., Alkahtani, M.: Automated extension-based penetration testing for web vulnerabilities. *Procedia Computer Science* **238**, 15–23 (2024). <https://doi.org/10.1016/j.procs.2024.05.191>
2. Applebaum, S., Gaber, T., Ahmed, A.: Signature-based and machine-learning-based web application firewalls: a short survey. *Procedia Computer Science* **189**, 359–367 (2021). <https://doi.org/10.1016/j.procs.2021.05.105>
3. Arp, D., Quiring, E., Pendlebury, F., Warnecke, A., Pierazzi, F., Wressnegger, C., Cavallaro, L., Rieck, K.: Dos and don’ts of machine learning in computer security. In: *Proc. USENIX Security*. pp. 3971–3988 (2022)
4. Asad, H.U., Gashi, I.: Diversity in open source intrusion detection systems. In: *Proc. SAFECOMP*. pp. 267–281 (2018). https://doi.org/10.1007/978-3-319-99130-6_18

5. Awang, N.F., Manaf, A.A., Zainudin, W.S.: A survey on conducting vulnerability assessment in web-based application. In: *Advanced Machine Learning Technologies and Applications*. pp. 459–471 (2014)
6. Balla, A., Habaebi, M.H., Elsheikh, E.A.A., Islam, M.R., Suliman, F.M.: The effect of dataset imbalance on the performance of scada intrusion detection systems. *Sensors* **23**(2), 758 (2023)
7. Bereziński, P., Jasiul, B., Szpyrka, M.: An entropy-based network anomaly detection method. *Entropy* **17**(4), 2367–2408 (2015). <https://doi.org/10.3390/e17042367>
8. Ceschin, F., Botacin, M., Bifet, A., Pfahringer, B., Oliveira, L.S., Gomes, H.M., Grégio, A.: Machine learning (in) security: A stream of problems. *Digital Threats* (2023). <https://doi.org/10.1145/3617897>
9. Demetrio, L., Valenza, A., Costa, G., Lagorio, G.: Waf-a-mole: Evading web application firewalls through adversarial machine learning. In: *Proc. ACM SAC*. pp. 1745–1752 (2020). <https://doi.org/10.1145/3341105.3373962>
10. Doupé, A., Cova, M., Vigna, G.: Why johnny can’t pentest: An analysis of black-box web vulnerability scanners. In: *Proc. DIMVA* (2010)
11. Fredriksson, T., Mattos, D.I., Bosch, J., Holmström Olsson, H.: Data labeling: An empirical investigation into industrial challenges and mitigation strategies. In: *International Conference on Product-Focused Software Process Improvement* (2020), <https://api.semanticscholar.org/CorpusID:227130095>
12. Hays, C., Schutzman, Z., Raghavan, M., Walk, E., Zimmer, P.: Simplistic collection and labeling practices limit the utility of benchmark datasets for twitter bot detection. In: *Proc. ACM Web Conf*. pp. 3660–3669 (2023). <https://doi.org/10.1145/3543507.3583214>
13. Kan, Z., McFadden, S., Arp, D., Pendlebury, F., Jordaney, R., Kinder, J., Pierazzi, F., Cavallaro, L.: Tesseract: Eliminating experimental bias in malware classification across space and time (extended version) (2025), <https://arxiv.org/abs/2402.01359>
14. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A survey on bias and fairness in machine learning. *ACM Comput. Surv.* **54**(6), 115:1–115:35 (jul 2021). <https://doi.org/10.1145/3457607>, <https://doi.org/10.1145/3457607>
15. Mirjalili, M., Nowroozi, A., Alidoosti, M.: A survey on web penetration test. *Advances in Computer Science: An International Journal* **3**, 107–121 (2014)
16. Mishra, S., Mallick, P.K., Jena, L.: Optimization of skewed data using sampling-based preprocessing approach. *Frontiers in Public Health* (2020). <https://doi.org/10.3389/fpubh.2020.00274>
17. Mohamed, N.: Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms. *Knowledge and Information Systems* (2025). <https://doi.org/10.1007/s10115-025-02429-y>
18. Montes, N., Betarte, G., Martínez, R., Pardo, A.: Web application attacks detection using deep learning. In: *Proc. CIARP*. pp. 227–236 (2021). https://doi.org/10.1007/978-3-030-93420-0_22
19. Namdari, A., Li, Z.: A review of entropy measures for uncertainty quantification of stochastic processes. *Advances in Mechanical Engineering* **11**(6) (2019). <https://doi.org/10.1177/1687814019857350>
20. Nguyen, T.C.H., Nguyen, M.K.L., Le, D.T., Nguyen, V.H., Ton, L.P., Nguyen-An, K.: Improving web application firewalls with automatic language detection. In: *SN Computer Science*. vol. 3 (2022). <https://doi.org/10.1007/s42979-022-01327-2>
21. Northcutt, C.G., Athalye, A., Mueller, J.: Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv preprint arXiv:2103.14749* (2021)

22. Palka, D., Zachara, M.: Learning web application firewall - benefits and caveats. In: Proc. ARES Workshops. pp. 295–308 (2011)
23. Pingale, S.V., Sutar, S.R.: Analysis of web application firewalls, challenges, and research opportunities. In: Proc. ICDSMLA. pp. 239–248 (2022)
24. Prandl, S., Lazarescu, M., Pham, D.S.: A study of web application firewall solutions. In: Proc. ICISS. pp. 501–510 (2015). https://doi.org/10.1007/978-3-319-26961-0_29
25. Saraiva, P.: On shannon entropy and its applications. Kuwait Journal of Science **50**(3), 194–199 (2023). <https://doi.org/10.1016/j.kjs.2023.05.004>
26. Sen, S., Singh, K.P., Chakraborty, P.: Dealing with imbalanced regression problem for large dataset using scalable artificial neural network. New Astronomy **99**, 101959 (2023). <https://doi.org/10.1016/j.newast.2022.101959>
27. Shaheed, A., Kurdy, M.H.D.B., Megias, D.: Web application firewall using machine learning and features engineering. Security and Communication Networks **2022** (2022). <https://doi.org/10.1155/2022/5280158>
28. Shahid, J., Hameed, M., Javed, I., Qureshi, K., Ali, M., Crespi, N.: A comparative study of web application security parameters: current trends and future directions. Applied Sciences (2022)
29. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal **27**, 379–423 (1948)
30. Sreekumar, G., Boddeti, V.N.: Spurious correlations and where to find them. arXiv preprint arXiv:2308.11043 (2023)
31. Tadhani, J.R., Vekariya, V., Sorathiya, V., Alshathri, S., El-Shafai, W.: Securing web applications against xss and sqli attacks using a novel deep learning approach. Scientific Reports **14**(1), 1803 (2024). <https://doi.org/10.1038/s41598-023-48845-4>
32. Toyama, A., Katahira, K., Kunisato, Y.: Examinations of biases by model misspecification and parameter reliability of reinforcement learning models. PsyArXiv (2022). <https://doi.org/10.31234/osf.io/jz5rv>
33. White, H.: A reality check for data snooping. Econometrica **68**(5), 1097–1126 (2000)