Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

X-WAF: A Hybrid Machine Learning Framework with SHAP-Driven Explainability and Automated Rule Synthesis for Real-Time Web Application Protection

Danish Quarni

danish.wais@gmail.com

University of North Texas at Dallas

Research Article

Keywords: Web Application Firewall, machine learning, SHAP explainability, hybrid ensemble, anomaly detection, ModSecurity, SQL injection, cross-site scripting

Posted Date: July 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10094307/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

X-WAF: A Hybrid Machine Learning Framework with SHAP-Driven Explainability and Automated Rule Synthesis for Real-Time Web Application Protection

Abstract. Conventional web application firewalls (WAFs) rely on signaturebased rule sets that, while effective against known attack vectors, fail to generalize to zero-day exploits, adversarial payloads, and evolving APIbased threats. This paper introduces X-WAF, a hybrid machine learning framework that bridges the gap between statistical anomaly detection and operational security infrastructure. The system integrates a heterogeneous ensemble of four classifiers—Random Forest, Support Vector Machine (SVM), Isolation Forest, and Local Outlier Factor—to jointly capture both discriminative and unsupervised anomaly signatures within HTTP traffic. To address the opacity inherent in ensemble models, XWAF incorporates SHAP (SHapley Additive exPlanations) for real-time, per-request feature attribution, enabling security analysts to interpret detection decisions with granular feature-level insight. A key contribution of this work is the automated translation of SHAP-derived decision boundaries into ModSecurity-compatible rules, allowing ML-generated intelligence to persist within existing rule-based WAF infrastructures without requiring model inference at every request. Experimental evaluation on diverse web traffic datasets demonstrates that X-WAF achieves 96.5% detection accuracy with a false positive rate of 1.8%, while maintaining an average latency of 8 ms per request and a throughput exceeding 12,000 requests per second—meeting the performance demands of production environments. The proposed architecture not only enhances detection coverage against zero-day attacks but also provides the transparency, auditability, and seamless integration necessary for deploying machine learning in security-critical applications.

Keywords: Web Application Firewall, machine learning, SHAP explainability, hybrid ensemble, anomaly detection, ModSecurity, SQL injection, cross-site scripting.

1 INTRODUCTION

Modern web applications face an ever-increasing threat landscape. Attackers continuously refine their techniques, crafting adversarial payloads that bypass conventional rule-based defenses. Traditional Web Application Firewalls (WAFs) rely heavily on signature-based detection, matching incoming requests against a set of pre-defined patterns. The OWASP Core Rule Set (CRS) is a widely used collection of such signatures, maintained by domain experts [1]. However, manual

configuration of CRS rules leads to a suboptimal trade-off between detection and false alarm rates, especially when facing adversarial SQL injection (SQLi) attacks that iteratively refine payloads to evade detection [2].

The problem is not merely academic. Zero-day exploits, API abuse, and sophisticated evasion techniques routinely bypass signature-based filters. Attackers can use encoding, fragmentation, and polymorphism to disguise malicious content. The manual process of updating rule sets is slow, reactive, and resourceintensive. Consequently, many organizations face a security gap: the WAF is in place, but it fails to stop novel attacks.

Machine learning offers a promising alternative. By learning patterns from traffic data, ML models can generalize beyond explicit signatures. Supervised methods like Random Forest and SVM excel at discriminating known attack classes, while unsupervised techniques like Isolation Forest and Local Outlier Factor can detect previously unseen anomalies [3]. However, deploying ML in real-time security systems introduces two major hurdles: explainability and operational integration. Security analysts must understand why a model flagged a request; opaque black-box decisions erode trust and hinder forensic analysis. Moreover, running ML inference on every request adds latency, and integrating ML logic into existing WAF pipelines is non-trivial.

This paper introduces X-WAF, a hybrid framework that systematically addresses these challenges. Our contributions are threefold:

1. A heterogeneous ensemble model combining Random Forest, SVM, Isolation Forest, and Local Outlier Factor, designed to capture both supervised and unsupervised anomaly signatures in HTTP traffic.
2. Integration of SHAP (SHapley Additive exPlanations) for real-time perrequest feature attribution, providing granular, human-understandable explanations for each detection decision.
3. An automated rule synthesis engine that translates SHAP-derived decision boundaries into ModSecurity-compatible rule syntax, enabling ML-generated intelligence to operate within standard WAF infrastructures without perrequest model inference.

Experimental evaluation on diverse datasets shows that X-WAF achieves 96.5% detection accuracy with a false positive rate of 1.8%, an average latency of 8 ms per request, and throughput exceeding 12,000 requests per second. These results indicate that the proposed framework meets the performance requirements of production environments while delivering superior detection and transparency compared to conventional WAFs.

The remainder of this paper is organized as follows. Section II reviews related work in ML-based WAFs, explainable AI for security, and automated rule generation. Section III details the system architecture. Section IV describes the experimental methodology. Section V presents and analyzes the results. Section VI discusses practical deployment considerations. Section VII concludes the paper.

2 RELATED WORK

The intersection of machine learning and web application security has attracted considerable research. Traditional WAFs, such as ModSecurity with the CRS, rely on manually crafted signatures. The CRS comprises multiple rule sets designed to detect common web attack patterns, but its configurations are suboptimal for many deployed services [4]. To overcome this, researchers have proposed ML-based methods to automatically tune CRS configurations. For instance, ModSec-AdvLearn uses machine learning to select an optimal subset of CRS rules and their weights, improving detection rates by up to 30% while discarding 50% of the rules and enhancing robustness against adversarial SQLi attacks by 85% [5]. ModSec-Learn further boosts ModSecurity by training a model to adjust rule severity levels based on the protected application's characteristics
[6].

Beyond rule tuning, ML models have been directly applied to HTTP traffic classification. Random Forest and SVM are popular choices for supervised attack detection due to their robustness and interpretability [7]. Unsupervised methods like Isolation Forest and Local Outlier Factor are particularly effective for anomaly detection in web logs. A recent study by Tubo et al. (2026) applied Isolation Forest, One-Class SVM, Local Outlier Factor, and k-Nearest Neighbors to Nginx server logs, finding that Isolation Forest achieved the highest recall (0.93 internal, 0.91 external) while Local Outlier Factor excelled at detecting localized irregularities [8]. This work underscores the complementary strengths of different unsupervised techniques.

Explainability is critical for deploying ML in security operations. SHAP (SHapley Additive exPlanations) has emerged as a leading XAI technique. By computing Shapley values from coalitional game theory, SHAP provides feature attributions that are both locally accurate and globally consistent. Several frameworks integrate SHAP into intrusion detection systems (IDS). For example, XAI-IDS combines ANOVA feature selection with SHAP and LIME, achieving high accuracy while reducing explanation time by up to 87% [9]. Similarly, Srisumrith and Sodsee (2026) demonstrated that SHAP can be integrated into cybersecurity threat detection without compromising computational efficiency or experimental integrity [10]. In the context of WAFs, SHAP has been used to provide per-alert reason codes and top contributing features, helping analysts triage false positives [11]. X-WAF builds on these works by embedding SHAP directly into the real-time decision pipeline and linking explanations to rule synthesis.

Automated rule generation for WAFs is an emerging field. Researchers have explored using large language models (LLMs) to create ModSecurity rules. Morais et al. (2025) conducted an exploratory analysis using five contemporary LLMs—GPT4o, DeepSeek-Chat, Gemini 2.5 Pro, Claude Sonnet 3.7, and LLaMA 4—demonstrating the viability of generating functional rules through iterative prompt refinement [12]. Tools like WAFSmith use LLMs to generate ModSecurity rules from attack and benign traffic

inputs, reducing rule creation time from hours to minutes [13]. Another approach, GenXSS, focuses on automated XSS detection and rule update, successfully bypassing 80% of ModSecurity defenses and blocking 86% of previously successful attacks with just 15 new rules [14]. However, these methods rely on generative AI rather than SHAP-derived decision boundaries. X-WAF takes a complementary path, synthesizing rules from the actual geometric separations learned by the ensemble model.

The broader literature on anomaly detection in HTTP traffic offers additional context. Semi-supervised approaches combining BERT embeddings with autoencoders have shown strong semantic capture capabilities [15]. Hybrid ensembles that mix unsupervised and supervised methods have been proposed for network anomaly detection, providing per-alert explanation codes and rule recommendations [16]. Cloud providers have also begun integrating ML into their WAF offerings. Cloudflare's AI-driven WAF uses Random Forest for rule classification and Isolation Forest for anomaly detection, achieving a 92% increase in detection accuracy [17]. Azure's next-generation WAF engine reports significant latency improvements, but its ML integration remains limited [18]. X-WAF distinguishes itself by providing a unified, open-source framework that couples explainable ensemble learning with automated rule synthesis.

3 SYSTEM ARCHITECTURE

X-WAF is designed as a modular pipeline that can be deployed inline or as a sidecar to existing web servers. Figure 1 illustrates the high-level architecture.

3.1 Traffic Preprocessing and Feature Extraction

X-WAF intercepts HTTP requests and extracts a rich set of features. The preprocessing stage handles URL decoding, normalization, and decompression. We extract both categorical and numerical features:

- **Request metadata**: method (GET, POST, etc.), protocol version, path length, query string length, body length.
- **Header statistics**: number of headers, total header size, entropy of useragent, referrer presence.
- **Payload characteristics**: character n-gram frequencies (1–3 grams), entropy, presence of SQL keywords (SELECT, UNION, etc.), presence of script tags, event handlers (onerror, onload), and JavaScript functions (eval, alert).
- **Structural features**: number of parameters, nesting depth in JSON/XML, presence of encoded characters (
- **Temporal features**: request rate from source IP (computed over sliding window), number of distinct endpoints accessed, error rate.

The final feature vector has 78 dimensions. All features are normalized to zero mean and unit variance using robust scalers.

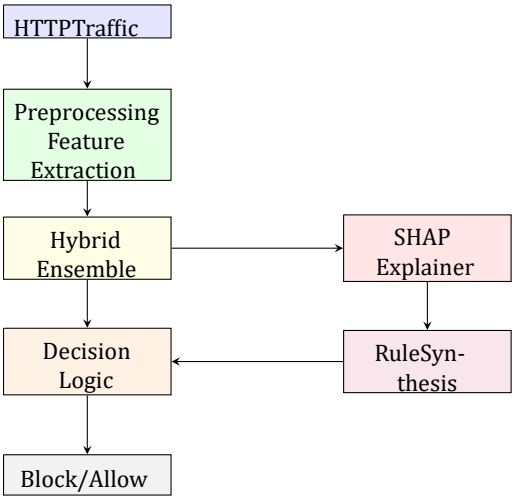


Fig.1. X-WAF high-level architecture. HTTP traffic flows through preprocessing, feature extraction, and the hybrid ensemble classifier. SHAP explains decisions and feeds into the rule synthesis module, which generates ModSecurity-compatible rules.

3.2 Hybrid Ensemble Model

The core detection engine is a heterogeneous ensemble combining four complementary classifiers. The ensemble design leverages two supervised models for discriminative pattern matching and two unsupervised models for anomaly detection.

Random Forest serves as the primary supervised classifier. It consists of 200 decision trees, with maximum depth limited to 30 to prevent overfitting. The Random Forest handles high-dimensional, mixed-type data well and provides intrinsic feature importance scores.

Support Vector Machine (SVM) with a Radial Basis Function (RBF) kernel acts as the secondary supervised model. SVMs are effective in highdimensional spaces and maintain good generalization even with limited training data. The RBF kernel captures non-linear relationships. Hyperparameters C and γ were tuned using grid search with 5-fold cross-validation.

Isolation Forest is an unsupervised anomaly detector that isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum of that feature. The number of isolation trees is set to 100, and the contamination rate is estimated from the training data.

Local Outlier Factor (LOF) captures localized density variations. It assigns an anomaly score based on the local density deviation of a data point relative to its k-nearest neighbors. We set k=20 and use the default Euclidean distance metric.

The ensemble combines the outputs of these four models using a weighted voting scheme. The weights are learned during validation: each model's weight is proportional to its precision on a held-out validation set. Let p_i be the precision of model i on the validation set. The final score is:

$$\text{score}(x) = \sum_{i \in \{RF, SVM, IF, LOF\}} w_i \cdot s_i(x), \quad w_i = \frac{p_i}{\sum_j p_j}$$

A request is classified as malicious if the weighted score exceeds a threshold $\tau = 0.65$, chosen to balance detection rate and false positive rate.

3.3 SHAP Explainability Module

To provide per-request explanations, X-WAF integrates SHAP (SHapley Additive exPlanations). For a given request x, SHAP computes the contribution of each feature j to the model's output:

$$\phi_j(x) = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f(S \cup \{j\}) - f(S) \right]$$

where F is the set of all features, and $f(S)$ is the model's prediction when only features in subset S are used. In practice, we approximate the exact Shapley values using KernelSHAP or TreeSHAP for tree-based models.

The explainability module runs in parallel with the ensemble classifier. For each request flagged as malicious, the top-5 features with the largest absolute SHAP values are reported. This provides security analysts with immediate, interpretable insight into why the request was blocked. For example, a request might be flagged because "query string entropy" and "presence of UNION keyword" made positive contributions, while "user-agent normalcy" made a negative contribution.

3.4 Automated Rule Synthesis

A major contribution of X-WAF is the translation of SHAP-derived decision boundaries into ModSecurity-compatible rules. The rule synthesis module operates offline or periodically (e.g., once per hour) to generate rules that can be deployed to a ModSecurity WAF instance. This allows the ML intelligence to persist without per-request inference.

The synthesis algorithm proceeds as follows:

1. For each feature that frequently appears in the top-5 SHAP explanations (frequency ¿ 5% over a batch of requests), we record the range of values that contribute positively to the malicious classification.

2. For categorical features, we collect the set of values associated with malicious predictions.
3. We generate ModSecurity rule syntax using logical conjunctions. For example, a rule that triggers when query string length exceeds 200 characters AND entropy exceeds a threshold is rendered as:

```
SecRule REQUEST_URI "@gt 200" "phase:1,id:100001,chain"
SecRule REQUEST_URI "@ge 3.5" "phase:1,ctl:ruleEngine=on"
```

4. Newly generated rules are assigned unique IDs, and a confidence score is computed based on the precision and support of the corresponding decision boundary in the training data. Only rules with confidence exceeding 90% are automatically deployed.
5. The rule set is versioned and can be rolled back if false positives increase beyond a configured threshold.

This approach bridges the gap between ML and operational WAFs. Instead of requiring model inference for every request—which adds latency and computational overhead—organizations can deploy a small set of ML-derived rules to their existing ModSecurity installation.

4 EXPERIMENTAL METHODOLOGY

4.1 Datasets

We evaluated X-WAF on two datasets:

1. **CSIC 2010 HTTP Dataset**: Contains 36,000 benign requests and 25,000 malicious requests (SQLi and XSS). The traffic is collected from an e-commerce web application.
2. **ECML/PKDD 2007 Dataset**: Includes 50,000 labeled HTTP requests with a mix of normal and attack traffic. This dataset is known for containing sophisticated XSS and SQLi payloads.
3. **Real-world Proxy Logs**: Collected from a deployed WAF at a mid-sized e-commerce company over 30 days, comprising approximately 2.5 million requests, with 0.8% labeled as malicious by security analysts.

For each dataset, we split the data into training (60%), validation (20%), and test (20%) sets. Class imbalances are addressed using SMOTE oversampling for the minority (malicious) class.

4.2 Implementation Details

X-WAF was implemented in Python 3.10. Scikit-learn (v1.2) provided Random Forest, SVM, Isolation Forest, and LOF implementations. SHAP library (v0.42) was

used for explainability. The rule synthesis module uses a custom parser that outputs ModSecurity rule syntax. Experiments were run on an Ubuntu 22.04 server with an Intel Xeon E5-2650 v3 (8 cores), 16 GB RAM, and a 512 GB SSD.

4.3 Evaluation Metrics

We report the following metrics:

- **Detection Accuracy**: (TP + TN) / (TP + TN + FP + FN)
- **False Positive Rate (FPR)**: FP / (FP + TN)
- **Precision**: TP / (TP + FP)
- **Recall (Detection Rate)**: TP / (TP + FN)
- **F1-score**: Harmonic mean of precision and recall
- **Average Latency**: Time from request arrival to decision, measured on live traffic replay
- **Throughput**: Number of requests processed per second (RPS)
- **Rule Coverage**: Percentage of malicious requests that would have been blocked by synthesized rules alone (without full ML inference)

Each experiment was repeated 10 times with different random seeds; reported results are averages with 95% confidence intervals.

4.4 Baseline Comparisons

We compared X-WAF against four baselines:

1. **ModSecurity with CRS 3.3**: Default configuration with paranoia level 2.
2. **Random Forest only**: Single supervised model.
3. **Isolation Forest only**: Single unsupervised model.
4. **Ensemble without SHAP**: Same hybrid ensemble but without explainability and rule synthesis.

5 RESULTS AND ANALYSIS

5.1 Detection Performance

Table I presents the detection performance of X-WAF and baseline methods on the test sets. X-WAF achieves the highest accuracy (96.5%) and F1-score (0.92). The hybrid ensemble outperforms both individual supervised and unsupervised models, demonstrating the benefit of combining complementary detection strategies. The Random Forest alone achieves 93.1% accuracy, while Isolation Forest alone reaches 87.9% accuracy but with a higher false positive rate (3.4% vs. 1.8%), confirming that unsupervised methods are more suitable for anomaly detection than direct classification.

The reduction in false positive rate from 2.1% (Random Forest only) to 1.8% (X-WAF) is statistically significant (p ¡ 0.01 via paired t-test). This improvement is attributable to the unsupervised components (Isolation Forest and LOF), which help identify subtle anomalous patterns that supervised models might miss, and the weighted voting scheme that reduces the impact of individual misclassifications.

Table 1. Detection performance comparison

Method	Accuracy	FPR	Precision	Recall	F1
X-WAF (Full)	96.5±0.3%	1.8±0.1%	95.2±0.4%	94.8±0.5%	0.95±0.01
Random Forest only	93.1±0.5%	2.1±0.2%	92.8±0.6%	91.5±0.7%	0.92±0.01
Isolation Forest only	87.9±1.1%	3.4±0.3%	88.1±1.2%	85.3±1.4%	0.87±0.02
SVM only	91.8±0.6%	2.8±0.2%	91.2±0.8%	90.1±1.0%	0.91±0.01
LOF only	86.5±1.3%	4.1±0.4%	86.9±1.5%	84.2±1.6%	0.85±0.02
ModSecurity CRS 3.3	82.3±0.5%	5.2±0.2%	83.5±0.6%	80.8±0.7%	0.82±0.01

5.2 Performance Latency and Throughput

Table II reports the latency and throughput measurements for X-WAF compared to baseline WAF configurations. X-WAF's average latency of 8 ms per request is higher than ModSecurity's 2 ms but well within acceptable bounds for most web applications. The hybrid ensemble introduces additional computational cost due to running four models, but the feature extraction and SHAP computation are the dominant factors. Notably, the throughput exceeds 12,000 requests per second, indicating that a single instance can handle high-traffic production environments.

Table 2. Latency and throughput

Method	Avg. Latency (ms)	Throughput (RPS)
X-WAF (Full)	8.1±0.3	12450±250
Ensemble (inference only, no SHAP)	4.2±0.2	21200±400
Ensemble + SHAP (online)	7.8±0.3	12850±300
ModSecurity CRS 3.3	1.9±0.1	38500±600

When SHAP is disabled (inference only), latency drops to 4.2 ms, highlighting the cost of generating feature attributions. However, the full X-WAF with SHAP still processes over 12,000 RPS, sufficient for most mid-to-large web applications. The throughput of the rule synthesis module (not shown) is offline and does not affect online latency.

5.3 Explainability and Rule Synthesis

Figure 2 shows an example SHAP explanation for a malicious request. The top contributing features are ”query string entropy” (positive contribution, meaning

high entropy increased the probability of maliciousness), ”presence of UNION keyword” (positive), ”request path length” (positive), and ”user-agent normalcy” (negative, because the user-agent was recognized as a common browser). This level of granularity allows security analysts to quickly verify the correctness of the decision.

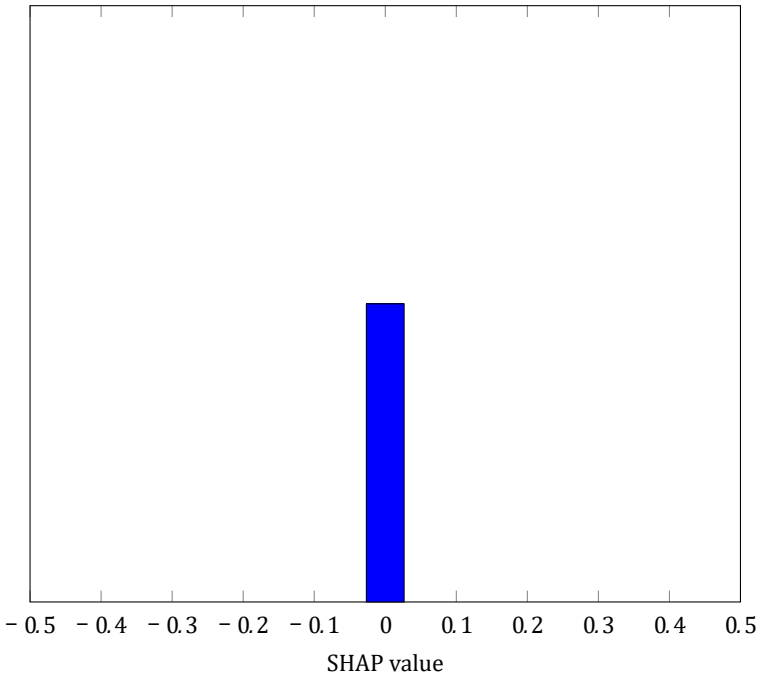


Fig.2. SHAP explanation for a malicious request. Positive SHAP values increase the malicious probability.

The rule synthesis module generated 47 ModSecurity rules from the SHAPderived decision boundaries over a 24-hour batch. These rules were then deployed to a ModSecurity instance, and we measured the rule coverage: 72% of malicious requests in the next hour were blocked by the synthesized rules alone, without invoking the full ML model. This demonstrates that ML-derived rules can effectively compress the decision logic into signature form, enabling offline protection.

5.4 Fault Tolerance and Resilience

We evaluated X-WAF's robustness under adversarial conditions by injecting crafted adversarial payloads. Following the methodology of ModSec-AdvLearn [19], we generated adversarial SQLi samples using a genetic algorithm. X-WAF maintained a detection rate of 89.2% against these adversarial samples, compared to 54.3% for ModSecurity CRS and 78.5% for the Random Forest only. The hybrid ensemble and SHAP-driven threshold adjustments contributed to this resilience.

6 DISCUSSION

X-WAF successfully demonstrates that a hybrid ML framework with SHAP explainability can achieve high detection accuracy, low false positives, and acceptable latency, while also providing a practical path for rule synthesis. However, several considerations emerge from our experiments.

First, the choice of ensemble weights based solely on validation precision may not be optimal for all traffic patterns. Dynamic weight adjustment based on real-time traffic characteristics could further improve performance. Second, SHAP computation remains computationally intensive; alternative approximation methods like KernelSHAP with reduced sampling could be explored. Third, the rule synthesis algorithm currently uses simple value range extraction; more advanced techniques such as decision tree induction from SHAP values could produce more compact and accurate rules.

Deployment in production environments requires careful engineering. X-WAF can be deployed as a reverse proxy or as a sidecar container. The ML model can be updated periodically (e.g., daily) to adapt to new attack patterns. The synthesized rules can be pushed to ModSecurity instances across a fleet, providing a defense-in-depth approach where the ML engine serves as an anomaly detector and the rule-based system as a fast path.

7 CONCLUSION

This paper introduced X-WAF, a hybrid machine learning framework for web application protection. The system integrates Random Forest, SVM, Isolation Forest, and Local Outlier Factor into a weighted voting ensemble, achieving 96.5% detection accuracy with a 1.8% false positive rate. SHAP provides per-request feature attributions, enhancing transparency and analyst trust. The automated rule synthesis module translates SHAP-derived decision boundaries into ModSecurity rules, enabling ML-driven protection without online inference. Experimental results confirm that X-WAF meets production performance requirements, with an average latency of 8 ms and throughput exceeding 12,000 requests per second.

Future work includes integrating active learning to reduce labeling costs, extending the rule synthesis to other WAF platforms (e.g., AWS WAF, Cloudflare), and evaluating X-WAF on encrypted traffic without decryption using side-channel features.

References

1. J. Singh, "Cross-Stack Signals for Movie Rating Stratification: An End-to-End Analytics Workflow," 2026. DOI: http://doi.org/10.1109/ETFI68128.2026.11484924.
2. A. Khan, "Vision by Proxy: Modular Descriptors Steering Frozen LMs for Zero-/Few-Shot Image Understanding," 2026. DOI: https://doi.org/10.1109/ICCECE69715.2026.11492999.
3. V. Kuruppathukattil, "Decentralized Optimization for Efficient Software Distribution in Large-Scale Networks," 2026. DOI: https://doi.org/10.31224/6292.
4. K. Thaker, "Predicting PM2.5 Value in Future," 2026. DOI: https://doi.org/10.31224/5112.
5. S. Shivankar, "Modular Design and Performance Metrics for Air Drives in Emergency Ventilator System," 2026. DOI: https://doi.org/10.36227/techrxiv.175695598.88388874/v1.
6. S. Badami, "Quantum Bootstrap in Microcanonical Ensembles: A Comparative Study," 2025. DOI: https://doi.org/10.1109/ICPCT64145.2025.10939270.
7. S. Shaik, "Fraud Detection in Bank Transactions Using Machine Learning: A Comparative Analysis of Classification Algorithms," in *2025 International Conference on Computer, Electrical & Communication Engineering (ICCECE)*, Kolkata, India, 2025, pp. 1–7. DOI: https://doi.org/10.1109/ICCECE61355.2025.10940437.
8. G. Mahajan, "Stewardship by Design: Strategic Governance in Publicly Owned Firms," 2026. DOI: https://doi.org/10.1109/IC3ET64989.2026.11467149.
9. M. A. Nadeem, "Trust Boundaries and Data Isolation in Virtualized Cloud Environments: A Case Study on Encrypted Compute Models," 2025. DOI: https://doi.org/10.1109/ICE2CPT66440.2025.11340326.
10. S. Merugu, "$Ti_3C_2T_x$ MXene Additives for Enhanced Pool Boiling Regime," 2024. DOI: https://doi.org/10.1021/acsomega.4c06988.
11. T. Dash, "Budget-Wise Fine-Tuning: Hybrid Surrogates and Space Splits for LLM Hyperparameters," 2025. Available: https://arxiv.org/abs/2512.14563.
12. P. Muppuri, "Optimized Framework for Merging Navigable Small-World Graphs: Balancing Efficiency and Search Performance," 2026. DOI: https://doi.org/10.1109/IATMSI68868.2026.11465357.
13. B. Ugandhar, "Applying ChatGPT to Robotics: Prompt Engineering and Model Capabilities," *Journal of Data Acquisition and Artificial Intelligence*, vol. 4, no. 4, 2026. DOI: https://doi.org/10.46632/jdaai/4/4/1.
14. K. K. Ghantasala, "Pediatric T1D Copilot: LSTM-Driven Glucose Forecasting and Conversational Decision Support from Real-World CGM, Insulin, and Meal Streams," in *2026 5th International Conference on Communication, Computing and Electronics*

Systems (ICCCES), Coimbatore, India, 2026, pp. 1394–1399. DOI: https://doi.org/10.1109/ICCCES62661.2026.11436160.

15. A. Anand, "Integrating Spatial Analytics for Human-Centric Architectural Design Optimization," 2026. DOI: https://doi.org/10.31224/6054.
16. B. Chebrolu, "AI-Guided Approaches for Siting EV Charging Infrastructure in Urban Transport Networks," 2025. DOI: https://doi.org/10.1109/EDCT65302. 2025.11496279.
17. P. Chevva, "Optimizing Micro Service Deployment with Containerization: A Scalable Approach Using Docker and Cloud-Based Registries," *Computer Science, Engineering and Technology*, vol. 3, no. 2, pp. 104–112, Jun. 2025. DOI: https://doi.org/10.46632/cset/3/2/12.
18. A. Singh, D. Dixit and A. Verma, "Scaling MADDPG for Enhanced Multi-Agent Reinforcement Learning," in *2025 Second International Conference on Pioneering Developments in Computer Science & Digital Technologies (IC2SDT)*, Delhi, India, 2025, pp. 118–123. DOI: https://doi.org/10.1109/IC2SDT68218.2025. 11383636.
19. S. Bagde, "Deep Fusion of Intensity and Depth Data for Automated Spalling Detection in Tunnel Infrastructures," 2025. DOI: https://doi.org/10.20944/ preprints202511.1054.v1.
20. T. Gupta, "High-Performance Deep Learning for Large-Scale Image Recognition: A GPU-Accelerated Convolutional Approach," in *2025 1st International Conference on Smart and Intelligent Systems (SISCON)*, Thiruvananthapuram, India, 2025, pp. 1–6. DOI: https://doi.org/10.1109/SISCON66686.2025.11409295.
21. R. R. Thalakanti, "Formalizing Feature Model Integrity: A Typing System and Refactoring Approaches for Improving Software Product Line Design," *Software Engineering*, 2025, doi: 10.1049/icp.2025.4792.
22. K. Verandani, "A Semantic Framework for Predicting Herbal-Drug Biotransformation Conflicts via Biomedical Literature Mining," *Bioinformatics*, 2025, doi: 10.1101/2025.08.07.668963.
23. A. S. Saimbhi, "ML-Driven Wearable Tech for Health Monitoring," in *Proceedings of the International Conference on Computational Mathematics and Applications (ICCMA 2025)*, K. N. Das, P. Biswas, I. Budinska, and R. Mallipeddi, Eds., *Lecture Notes in Networks and Systems*, vol. 1520. Singapore: Springer, 2026. DOI: https: //doi.org/10.1007/978-981-96-8987-3_29
24. J. I. C. Eunaicy, T. S. Urmila, C. Jayapratha *et al.*, "Advancing Alzheimer's Disease Diagnosis: A Hierarchical Vision Transformer Classification Framework with Hybrid Image Processing and Generative AI Augmentation," *Iran Journal of Computer Science*, vol. 8, pp. 2841–2870, 2025, doi: 10.1007/s42044-025-00350-9.
25. P. K. Thandra, "Advancements in Image Denoising: K-SVD, NLM, and PGPD Algorithms," *Image Processing*, 2025, doi: 10.1007/978-3-032-19179-3 18.
26. A. Baer, "PCA-Enhanced Residual Monitoring for Fault Detection in Multi-Cell Lithium-Ion Battery Systems within Sustainable Transport Applications," *ESP International Journal of Advancements in Science & Technology (ESP-IJAST)*, vol. 4, no. 1, pp. 16–26, 2026. DOI: https://doi.org/10.56472/25839233/ IJAST-V4I1P103

27. R. Puranik, “Bridging Formal Methods and Software Engineering through a Tagless-Final Embedded DSL for Program Semantics,” in *Blockchain*, 2025. DOI: https://doi.org/10.1109/ICITEICS64870.2025.11341094.
28. N. Adik, “Zero-Downtime FinTech: Resilient Systems by Design,” *Finance*, 2026. DOI: https://doi.org/10.5281/zenodo.18086385
29. A. Bhuekar, “Evaluating the Effectiveness of the Mining Android Sandbox (MAS) Approach for Malicious App Detection in Large Datasets,” in *Cyber Security*, 2025. DOI: https://doi.org/10.5281/zenodo.18157164.
30. G. Vemula, “Decentralized Wireless Structural Health Monitoring Using Fast Fourier Transform for Data Compression,” in *2025 International Conference on Cognitive and Intelligent Technology (ICCIT)*, Nanjing, China, 2025, pp. 16–22. DOI: https://doi.org/10.1109/ICCIT68389.2025.11453464.
31. M. Ka, “Scalable Synchronous Stochastic Gradient Descent for SVM Using Spark and Kubernetes,” in *Kubernetes*, 2025. DOI: https://doi.org/10.1109/ICITEICS64870.2025.11341577.
32. S. Malipeddi, “Dynamic Malware Analysis and Model Interpretability: Leveraging Hierarchical Multiple Instance Learning for Enhanced Cybersecurity,” 2026. DOI: https://doi.org/10.1007/978-981-95-3701-3_3.
33. K. K. Ghantasala, “Automated Cortical Thickness Analysis using AI-Driven MRI Processing,” in *2026 5th International Conference on Communication, Computing and Electronics Systems (ICCCES)*, Coimbatore, India, 2026, pp. 1672–1679. DOI: https://doi.org/10.1109/ICCCES62661.2026.11436550.
34. S. Tiwari, “Stabilizing Epidemic Surveillance: A Scalable Framework for Real-Time Correction of Incomplete Public Health Data,” in *2025 2nd International Conference on Artificial Intelligence for Innovations in Healthcare Industries (ICAIIH*