IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

WEB APPLICATION FIREWALLS: A COMPREHENSIVE SURVEY OF DETECTION TECHNIQUES, DEPLOYMENT MODELS, AND REAL-TIME MONITORING APPROACHES

[1]Drupad Dhamdhere, [2]Shardul Pharande, [3]Aashish Kondgire, [4]Sakshi Phutane

1 B.E. Student, 2 B.E. Student, 3 B.E. Student, 4 B.E. Student

[1,2,3,4,5]Department of Information Technology,

[1,2,3,4,5]PES Modern College of Engineering, Pune, India

Abstract: Web applications have quietly become the front door to almost everything we do online—banking, shopping, healthcare, government services—and unsurprisingly, attackers have taken notice. Industry reports now attribute roughly one in four data breaches to weaknesses in web-facing software, a number that keeps climbing year after year. Web Application Firewalls (WAFs) have stepped in as a vital line of defense, sitting between users and servers, examining every HTTP request to catch malicious traffic before it can do any damage. In this survey, we take a broad look at the WAF landscape as it stands today. We trace the evolution of these tools from the early days of simple rule-matching systems like Snort and ModSecurity, through the enterprise-grade commercial products that followed, all the way to the modern cloud-native offerings from AWS, Cloudflare, and Azure—and the exciting new wave of AI-powered firewalls that are just beginning to emerge. We organize the different detection approaches into a clear taxonomy covering signature-based, anomaly-based, machine-learning-driven, and hybrid methods, and we map them against the OWASP Top 10 (2021) along with their corresponding Common Weakness Enumerations. Beyond detection, we also dig into deployment architectures, real-time monitoring and alerting mechanisms, and the practical trade-offs organizations face around accuracy, latency, and operational complexity. Drawing on over 40 studies published between 2003 and 2025, we identify seven research gaps that continue to hold the field back—among them the shortage of public evaluation benchmarks, the opaqueness of commercial solutions, and the persistent difficulty of catching novel zero-day payloads. We close by outlining several promising future directions, including federated learning for distributed WAF deployments and the use of explainable AI to make security decisions more transparent and trustworthy.

Index Terms — Web Application Firewall, Survey, Intrusion Detection, SQL Injection, Cross-Site Scripting, DDoS Mitigation, Machine Learning, Anomaly Detection, Real-Time Monitoring, OWASP Top 10.

I. INTRODUCTION

It is hard to overstate just how central web applications have become to everyday life. Whether someone is transferring money, filing taxes, or booking a doctor's appointment, chances are they are interacting with an HTTP-based interface. This ubiquity, however, comes with a cost. The 2024 Verizon Data Breach Investigations Report [1] found that web applications served as the primary attack vector in 26% of all confirmed data breaches—and that figure has been creeping upward steadily over the past five years. Meanwhile, the OWASP Foundation's well-known Top 10 list [2] continues to highlight many of the same vulnerability categories—injection flaws, broken access control, security misconfigurations—that have plagued the web since the list was first published back in 2003.

This is where Web Application Firewalls come into the picture. Unlike traditional network firewalls that work at layers 3 and 4 of the OSI model, a WAF operates at layer 7, inspecting the actual content of HTTP requests and responses. This distinction matters enormously in practice. A network firewall, for instance, has no way to tell a legitimate login attempt apart from an SQL injection payload cleverly tucked inside a username field. A WAF can.

The WAF ecosystem has grown remarkably since the early 2000s. Open-source projects like ModSecurity [4] brought HTTP-level filtering to anyone willing to set it up. Commercial vendors such as Imperva [13], F5 Networks [14], and Barracuda [15] added polish and enterprise features—bot management, API protection, compliance dashboards. Then came the cloud giants—AWS [17], Cloudflare [18], Azure [19]—who made it possible to deploy a WAF without managing any hardware at all. And on the research side, academics have been pushing boundaries with machine-learning detectors, deep neural networks, and hybrid systems that blend the best of signatures and behavioral analysis [8, 27, 28].

Yet despite all this activity, we noticed something missing in the literature. Most existing surveys [8, 39, 40] tend to focus narrowly on detection algorithms, paying relatively little attention to deployment models, operational dashboards, and the messy real-world trade-offs that organizations actually wrestle with when choosing and configuring a WAF. There has not been, to our knowledge, a single survey that ties together the open-source, commercial, and academic perspectives while also covering the increasingly important dimension of real-time monitoring and visualization.

This paper is our attempt to fill that gap. Specifically, we make four contributions:

1. A structured taxonomy of WAF detection techniques.
2. A comprehensive comparison of fifteen open-source, commercial, and cloud-based WAF solutions.
3. A systematic analysis of real-time monitoring architectures and how they affect threat response times.
4. An identification of seven open research gaps along with directions we believe are worth pursuing.

II. BACKGROUND: WEB APPLICATION THREATS

Before diving into how WAFs defend applications, it helps to understand what they are defending against. The OWASP Top 10 [2] remains the most widely adopted classification of web application risks, and it provides a useful framework for our discussion. Table 1 maps the 2021 edition to the relevant CWE identifiers and indicates which WAF detection approach is most commonly used for each category.

Table 1. OWASP Top 10 Mapped to CWE and Primary Detection

Rank	Category	CWE	Detection Approach
A01	Broken Access Control	CWE-284	Behavioral
A02	Cryptographic Failures	CWE-327	Out of scope
A03	Injection	CWE-89	Signature + ML
A04	Insecure Design	CWE-209	Out of scope
A05	Security Misconfiguration	CWE-16	Policy rules
A06	Vulnerable Components	CWE-1035	Virtual patching
A07	Authentication Failures	CWE-287	Rate limiting
A08	Data Integrity	CWE-502	Signature
A09	Logging Failures	CWE-778	Out of scope
A10	Server-Side Request Forgery	CWE-918	Signature + Anomaly

A few things jump out from this table. First, not every OWASP category is something a WAF can realistically address. Design-level flaws (A04) and logging deficiencies (A09), for example, need to be fixed in the code itself—no amount of traffic inspection will help. Second, injection attacks (A03) remain the sweet spot where WAFs provide the most direct, measurable protection. Third, newer categories like SSRF (A10) are still poorly handled by many deployed WAFs.

A. Attack Vector Taxonomy

To bring more structure to the discussion, we group the attack vectors into six families:

1. **Injection Attacks** (SQL Injection, Command Injection): The attacker slips executable statements into user-supplied input fields.
2. **Cross-Site Attacks** (XSS, CSRF): These exploit the trust relationship between a user's browser and the web application.
3. **Volumetric and Rate-Based Attacks** (Application-layer DDoS, Brute Force): Each request looks normal—the malice is in the sheer volume [10].
4. **File and Data Handling Attacks** (Malicious File Uploads, XXE, Deserialization).
5. **Server-Side Request Attacks** (SSRF): Tricking the server into making requests to internal resources [37].
6. **Authentication and Session Attacks** (Credential stuffing, session fixation).

III. TAXONOMY OF DETECTION TECHNIQUES

We organize the WAF detection literature into four categories.

A. Signature-Based Detection

Signature-based detection is the oldest and still the most widely deployed approach in practice. The system maintains a library of known-bad patterns—usually regular expressions—and scans every incoming request for matches.

ModSecurity and the OWASP CRS. The best-known example is ModSecurity [4]. The OWASP Core Rule Set (CRS) [5] ships with roughly 200 rules covering common attacks. Chaim and Oliveira [6] found that the CRS caught about 88% of standard attack payloads, but had a 12% false-positive rate.

Where they work and fall short. Signature engines are fast and deterministic. However, they cannot detect anything they have not already seen. Advanced evasion tricks can slip past signature sets with surprisingly high success rates [28].

B. Anomaly-Based Detection

Instead of looking for known-bad patterns, anomaly-based systems build a model of what "normal" looks like and flag anything that deviates.

Statistical profiling. The pioneering work by Kruegel and Vigna [7] modeled parameter length distributions and character frequencies during a training phase, managing a 93% detection rate.

The upside and downside. Anomaly detection can potentially catch zero-day attacks. However, it requires a clean, representative training set, and as applications evolve, baseline drift causes false positives to pile up [32].

C. Machine-Learning-Based Detection

Over the past decade, machine learning has made steady inroads into WAF detection.

Supervised learning. Researchers have trained random forests and SVMs on labelled HTTP request datasets, achieving F1 scores above 0.96 for SQL injection and XSS detection [27, 29].

Deep learning and unsupervised learning. Deep neural networks and gradient-boosted trees have shown over 99% accuracy on standard datasets like NSL-KDD [30]. Meanwhile, autoencoders trained solely on normal traffic can spot anomalies by measuring reconstruction error [31].

Where ML shines and struggles. These models can generalize beyond known patterns. But the biggest practical barrier remains the scarcity of high-quality, labelled datasets.

D. Hybrid Detection

Since no single technique works perfectly across all attack types, research has naturally shifted towards combining them.

Signature + ML. Appelt et al. [28] demonstrated that pairing ModSecurity rules with a random forest classifier brought the CRS evasion rate down from 67% to under 20%.

Multi-layer ensembles. Commercial vendors run multiple detection engines simultaneously and combine their outputs, creating a deeply resilient pipeline [18].

Table 2. Comparison of Detection Categories

Category	Zero-Day	FP Rate	Speed
Signature	Low	Medium	Fast
Anomaly	High	High	Medium
Machine Learning	High	Low	Varies
Hybrid	High	Low	Medium

IV. DEPLOYMENT ARCHITECTURES

Where a WAF sits in the network stack fundamentally defines its capabilities. We see four deployment models in industry practice.

Reverse-Proxy Deployment: The WAF sits between the client and the web server, forwarding only clean requests [4].

Embedded / In-Application: The WAF logic runs inside the application process itself. It has no extra network hop, but ties the WAF to the application stack [11].

Cloud-Based Deployment: Cloud WAFs run entirely at the provider's edge network (e.g., AWS WAF [17]). They offer global scale but introduce vendor lock-in.

Hybrid Deployment: Combines a cloud WAF at the edge for volumetric attacks and a local WAF for deeper inspection [45].

V. COMPARISON OF EXISTING WAF SOLUTIONS

Table 3 presents a side-by-side comparison across several solutions.

Table 3. Comparative Analysis of Prominent WAFs

WAF	Type	Detection	OSS	ML
ModSecurity [4]	Proxy	Signature	Yes	No
NAXSI [23]	Proxy	Scoring	Yes	No
SafeLine [26]	Proxy	ML + Sig.	Yes	Yes
Imperva [13]	Cloud	Hybrid	No	Yes
F5 BIG-IP [14]	Appliance	Hybrid	No	Yes
AWS WAF [17]	Cloud	Managed	No	Partial
Cloudflare [18]	Cloud	Hybrid + ML	No	Yes

Open-Source Solutions: ModSecurity is the most widely referenced open-source WAF. SafeLine combines traditional signatures with a semantic analysis engine using ML.

Commercial & Cloud Solutions: Enterprise WAFs come loaded with advanced bot management and proprietary intelligence feeds. Cloud WAFs are the fastest-growing segment due to single-click activations at massive scale.

VI. REAL-TIME MONITORING AND VISUALIZATION

Detection on its own is not enough. We identified three generations of monitoring approaches:

First Gen: Log-Based: WAFs wrote events to flat log files. The ELK stack became standard for aggregation, but introduces a delay of several minutes [44].

Second Gen: Polling Dashboards: Most cloud services offer dashboards that poll their backend at fixed intervals (5–30 seconds), creating a blind window.

Third Gen: Push-Based Streaming: The latest generation uses WebSockets or Server-Sent Events to deliver events the instant they occur, enabling genuine real-time situational awareness.

VII. EVALUATION METHODOLOGIES AND BENCHMARKS

Different studies use different datasets, payloads, and configurations. Common datasets include CSIC 2010 [46], CICIDS 2017 [47], and community repositories like PayloadsAllTheThings [12]. Standard metrics reported include precision, recall, latency overhead, and evasion resistance.

VIII. OPEN CHALLENGES

Our survey brought to light multiple persistent challenges:

1. **No standardized benchmarks.** Older datasets don't cover API abuse or GraphQL injection.
2. **Encoding and mutation evasion.** Automated fuzzing frequently bypasses basic signatures.
3. **Commercial opacity.** Cloud WAF algorithms are proprietary black-boxes.
4. **False positives from natural language.** English text frequently resembles SQL syntax elements.
5. **Not enough labelled data.** ML models lack high-quality, up-to-date attack datasets.
6. **Distributed attacks.** WAFs typically lack cross-request aggregation needed to identify low-and-slow threats.
7. **ML explainability.** Understanding why a black-box model blocked a request remains difficult.

IX. FUTURE RESEARCH DIRECTIONS

Modern Benchmarks: We need up-to-date datasets with API-specific attacks and full HTTP request contexts.

Adversarial Testing: Fuzzing and reinforcement-learning-based mutation should become a routine part of how WAFs are evaluated [50].

Explainable AI: Integrating SHAP or LIME into ML-based WAFs would help operators trust the automated systems.

Federated Learning: Training models collaboratively across WAF instances without sharing raw traffic addresses data privacy concerns [51].

Context-Aware Detection: Multi-request correlation would uncover slow, distributed attacks.

Adaptive Thresholds: WAFs that automatically adjust their sensitivity based on traffic patterns could reduce false positives during quiet periods [52].

X. CONCLUSION

Web Application Firewalls have evolved into sophisticated, multi-layered defense systems blending signatures, anomaly detection, and machine learning. No single detection technique is a silver bullet: hybrid approaches consistently outperform standalone methods. Furthermore, the shift from log-based to push-based real-time monitoring directly affects an organization's incident response time. Moving forward, the integration of explainable AI, federated learning, and modern benchmark datasets will be crucial to securing the next generation of web applications.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to our guide, Prof. Supriya Jagtap, Department of Information Technology, PES Modern College of Engineering, Pune, for her invaluable guidance, constant encouragement, and thoughtful feedback throughout the course of this work. Her expertise and support were instrumental in shaping this survey paper.

REFERENCES

[1] Verizon, "2024 Data Breach Investigations Report," Verizon Business, 2024. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/

[2] OWASP Foundation, "OWASP Top 10 — 2021," 2021. [Online]. Available: https://owasp.org/Top10/

[3] M. Roesch, "Snort — lightweight intrusion detection for networks," in Proc. USENIX LISA Conf., 1999, pp. 229–238.

[4] I. Ristic, ModSecurity Handbook. Feisty Duck, 2010.

[5] OWASP Foundation, "OWASP ModSecurity Core Rule Set," 2024. [Online]. Available: https://coreruleset.org/

[6] R. Chaim and J. Oliveira, "Evaluating ModSecurity and the OWASP CRS against real-world attacks," in Proc. IEEE ISCC, 2019, pp. 1–6.

[7] C. Kruegel and G. Vigna, "Anomaly detection of web-based attacks," in Proc. ACM CCS, 2003, pp. 251–261.

[8] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," IEEE Commun. Surveys Tuts., vol. 18, no. 2, pp. 1153–1176, 2016.

[9] W. G. Halfond, J. Viegas, and A. Orso, "A classification of SQL injection attacks and countermeasures," in Proc. IEEE ISSSE, 2006, pp. 13–23.

[10] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service flooding attacks," IEEE Commun. Surveys Tuts., vol. 15, no. 4, pp. 2046–2069, 2013.

[11] VMware Tanzu, "Spring Boot Reference Doc 3.5," 2025.

[12] swiss krepo, "PayloadsAllTheThings," GitHub, 2024.

[13] Imperva, "Web Application Firewall (WAF)," 2024.

[14] F5 Networks, "BIG-IP Application Security Manager," 2024.

[15] Barracuda Networks, "Barracuda Web Application Firewall," 2024.

[16] Fortinet, "FortiWeb — Web Application Firewall," 2024.

[17] Amazon Web Services, "AWS WAF," 2024.

[18] Cloudflare, "Cloudflare WAF," 2024.

[19] Microsoft, "Azure Web Application Firewall," 2024.

[20] Google Cloud, "Cloud Armor," 2024.

[21] Akamai Technologies, "Kona Site Defender," 2024.

[22] Sucuri, "Sucuri Website Firewall," 2024.

[23] NBS System, "NAXSI — Nginx Anti XSS & SQL Injection," 2024.
[24] Z. Pohlmann, "Shadow Daemon," 2024.
[25] Coraza Project, "Coraza WAF," 2024.
[26] Chaitin Technology, "SafeLine WAF," 2024.
[27] T. D. Nguyen et al., "Machine learning-based detection of SQL injection and cross-site scripting attacks," in Proc. IEEE RIVF, 2019, pp. 1–6.
[28] D. Appelt, C. D. Nguyen, and L. Briand, "Behind an application firewall, are we safe?" in Proc. IEEE/ACM ASE, 2018, pp. 279–290.
[29] C. Torrano-Gimenez, A. Perez-Villegas, and G. Alvarez, "A self-learning anomaly-based web application firewall," in Proc. CISIS, 2015, pp. 85–92.
[30] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," Neural Comput. Appl., vol. 31, no. 4, pp. 955–965, 2019.
[31] Y. Mirsky et al., "Kitsune: an ensemble of autoencoders for online network intrusion detection," in Proc. NDSS, 2018.
[32] R. Sommer and V. Paxson, "Outside the closed world: on using machine learning for network intrusion detection," in Proc. IEEE S&P, 2010, pp. 305–316.
[33] I. Corona, G. Giacinto, and F. Roli, "Adversarial attacks against intrusion detection systems," Inf. Sci., vol. 239, pp. 199–225, 2013.
[34] A. Sadeghian, M. Zamani, and A. A. Manaf, "A taxonomy of SQL injection detection and prevention techniques," in Proc. IEEE ICIT, 2018, pp. 674–679.
[35] I. Hydara et al., "Current state of research on cross-site scripting (XSS) — a systematic literature review," Inf. Softw. Technol., vol. 58, pp. 170–186, 2015.
[36] C.-A. Staicu and M. Pradel, "Freezing the web: a study of ReDoS vulnerabilities in JavaScript-based web servers," in Proc. USENIX Security, 2018, pp. 361–376.
[37] B. Jabiyev et al., "Preventing server-side request forgery attacks," in Proc. ACM SAC, 2021, pp. 1626–1635.
[38] K. Thomas et al., "Protecting accounts from credential stuffing with password breach alerting," in Proc. USENIX Security, 2019, pp. 1556–1571.
[39] K. L. Ingham and H. Inoue, "Comparing anomaly detection techniques for HTTP," in Proc. RAID, 2007, pp. 42–62.
[40] V. Clincy and H. Shahriar, "Web application firewall: network security models and configuration," in Proc. IEEE COMPSAC, 2018, pp. 835–836.
[41] J. Singh and J. Singh, "A survey on machine learning–based malware detection in executable files," J. Syst. Archit., vol. 90, pp. 1–20, 2018.
[42] M. Ring et al., "A survey of network-based intrusion detection data sets," Comput. Secur., vol. 86, pp. 147–167, 2019.
[43] I. Fette and A. Melnikov, "The WebSocket Protocol," RFC 6455, IETF, 2011.
[44] Elastic, "The Elastic Stack: Elasticsearch, Kibana, Logstash," 2024. [Online]. Available: https://www.elastic.co/elastic-stack
[45] D. M. Divakaran et al., "Evidence gathering for network security and forensics," Digit. Investig., vol. 20, pp. S56–S65, 2017.
[46] National Institute of Standards and Technology, "Guidelines on Securing Public Web Servers," NIST SP 800-44, 2023.
[47] Spanish National Research Council, "HTTP Dataset CSIC 2010," 2010.
[48] Canadian Institute for Cybersecurity, "CICIDS 2017 Dataset," 2017.
[49] M. Tavallaee et al., "A detailed analysis of the KDD CUP 99 data set," in Proc. IEEE CISDA, 2009, pp. 1–6.
[50] C. Torrano-Gimenez et al., "The HTTP dataset CSIC 2012," Spanish National Research Council, Tech. Rep., 2012.
[51] L. Demetrio et al., "WAF-A-MoLE: Evading web application firewalls through adversarial machine learning," in Proc. ACM SAC, 2020, pp. 1745–1752.
[52] D. L. Marino et al., "An adversarial approach for explainable AI in intrusion detection systems," in Proc. IEEE IECON, 2018, pp. 3237–3243.

[53] V. Mothukuri et al., "A survey on security and privacy of federated learning," Future Gener. Comput. Syst., vol. 115, pp. 619–640, 2021.

[54] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," IEEE Trans. Neural Netw. Learn. Syst., vol. 34, no. 8, pp. 3779–3795, 2021.