SKUDONET

SKUDONET GUIDE

How to Evaluate WAFs in 2026

This guide does not explain what a WAF is or repeat basic concepts. Its goal is to provide a practical technical framework to evaluate WAFs in 2026, with measurable criteria applicable to real environments: cloud, APIs, encrypted traffic, high demand, and—especially—services consumed as SaaS.

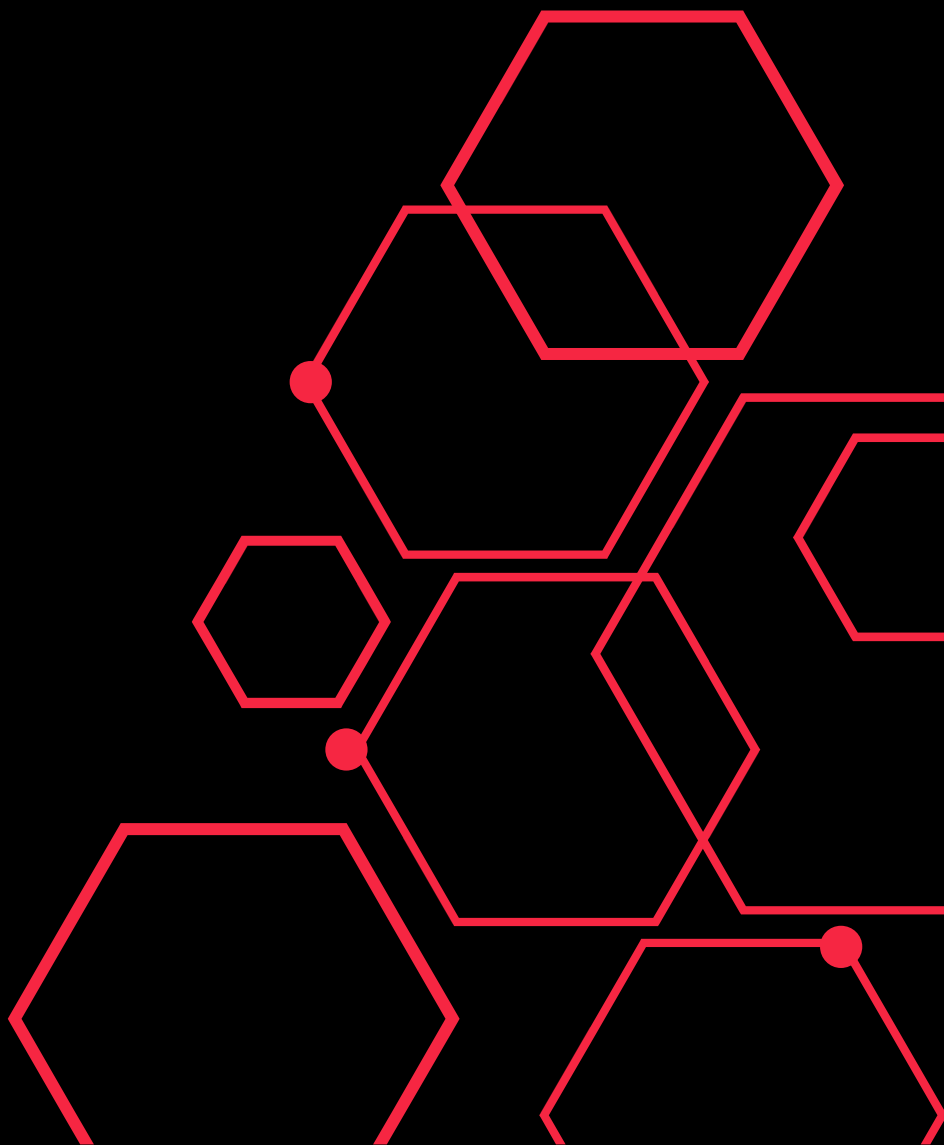

Table of Contents

What Evaluating a WAF Means in 2026

Many WAF solutions perform well at one extreme and fail at the other. That imbalance rarely appears in demos or superficial tests; it becomes visible only once the WAF is deployed in production.

The context has changed significantly over the last few years. Applications are increasingly API-first, attack surfaces are distributed across microservices, gateways, CDNs, and edge layers, and most relevant traffic is encrypted. In parallel, widely adopted foundations —engines and rule sets—have evolved, meaning that “using the standard” no longer guarantees stable results, either in security or in false positives.

In SaaS models, these challenges are amplified. Technical teams have limited ability to compensate for architectural constraints or deep system behavior, and any degradation or false blocking quickly turns into an incident. As a result, evaluation must answer a concrete question:

How does this WAF behave when blocking real traffic, using complex legitimate datasets, without artificial tuning, and without relying on privileged system access?

WAF Architecture and Its Real-World Impact

A key architectural criterion is the difference between early inspection and late inspection. **Early inspection** filters traffic **before it consumes downstream resources**, improving resilience against spikes and attacks. **Late inspection** enables **more contextual decisions**, but may occur too late to prevent degradation.

In **SaaS**, this difference has an additional implication: if the **provider fully abstracts how inspection works**—placement, limits, degradation behavior—the risk is no longer only technical, but operational. A mature evaluation must demand observable signals about behavior under load and service limits.

From a technical standpoint, **architecture directly impacts:**

Dimension	Architectural impact
Latency	Early, optimized inspection reduces per-request processing time
Availability	Integrated architectures avoid single points of failure
Scalability	The scaling model determines how the WAF responds to traffic spikes
Behavior under load	Defines whether the system degrades in a controlled way or fails abruptly

Any evaluation that does not validate **prevention-mode behavior** and does not capture exportable, queryable results is measuring a state that does not reflect production.

Deployment Models Relevant in 2026

Most WAF evaluations today focus on solutions consumed as a service, although several distinct models exist within that category.

Cloud Managed WAF (SaaS)

A **cloud-managed WAF (SaaS)** enables rapid deployment and offloads infrastructure and operational management to the provider. This model reduces time-to-protection, but shifts critical decisions to the service layer. Evaluation in 2026 must focus on transparency: what traffic, rules, and decisions can be inspected; what can be tuned or overridden; and which elements remain abstracted from the operator.

Hyperscaler WAFs

Hyperscaler-native WAFs are deeply integrated into the cloud provider's ecosystem and align naturally with infrastructures concentrated in that environment. Their effectiveness depends on balancing protection accuracy, false positives, and operational cost. As traffic volume or advanced protections increase, tuning complexity and total cost of ownership can grow rapidly.

Load Balancer + WAF

Delivery platforms combining load balancing and WAF capabilities reduce architectural fragmentation and simplify policy enforcement across the traffic path. This model improves end-to-end visibility and consistency between delivery and security layers. When consumed as a managed service, however, the platform becomes a central dependency that must be evaluated for resilience guarantees, service limits, and observability depth.

Single-Tenant vs Multi-Tenant Architecture

In 2026, the **single-tenant vs multi-tenant** discussion is no longer philosophical: it is about operational isolation and predictability.

Multi-tenant

In **SaaS, multi-tenancy means shared infrastructure.** This can translate into limited customization, global policies, and controls designed to protect platform stability. In low-risk scenarios it may be acceptable, but **for high traffic, public APIs, or critical services it introduces real risk.**

Technically, multi-tenancy impacts three core areas:

Resource isolation. During traffic spikes, inspection consumption should not depend on other tenants.

Rules and tuning isolation. False positives happen due to specific combinations of routes, bodies, and business flows. Generic tuning does not work.

Auditability and traceability. In critical environments, you need to know what changed, when, and why.

Single-tenant

Single-tenant architectures address these constraints by providing **dedicated resources**, **isolated rule sets**, and **independent tuning per environment.**

This improves performance predictability, **reduces cross-tenant impact**, and simplifies incident investigation. **For critical services**, single-tenancy stops being a “premium” option and becomes a technical requirement.

Detection Capability: How to Measure It Correctly

Detection capability is often summarized through the **True Positive Rate (TPR)**—the percentage of attacks correctly blocked. While important, this metric alone does not provide a full picture.

Measuring TPR correctly in 2026 requires, at minimum:

- Coverage across attack families (SQLi, XSS, XXE, path traversal, command execution)
- Inclusion of modern payloads and known bypass techniques
- Validation that the WAF is actually blocking (prevention mode)

"Coverage" should **be treated as a set:**

- OWASP Top 10, yes—but with realistic payload variation and techniques
- APIs, with modern structures and behaviors
- Complex payloads, including large bodies and common formats
- Evasion techniques that typically break purely signature-based approaches

In SaaS models, evaluation must also consider **what aspects of inspection are configurable**, what remains fixed, and how blocking decisions are evidenced in exportable logs. Without traceable proof of blocking, detection is not being measured operationally.

False Positives and Detection Quality

The **False Positive Rate (FPR)** is one of the most critical production metrics. A WAF that blocks legitimate traffic has direct impact on business, user experience, and operational workload.

Even a **low FPR can generate a high number of incidents** when traffic volume is large. **In SaaS**, the issue is amplified: the ability to **resolve false positives depends** on the granularity **the provider** allows and the speed at which changes can be applied safely.

What causes false positives in production (and how to test it):

FPR source	Realistic example	Architectural impact
Complex bodies	Deep JSON, multipart forms	Real or recorded API/UI requests
E-commerce/ business flows	search/filter, cart, checkout	End-to-end navigation and actions
Uploads	PDFs, images, metadata attachments	Uploads on real routes
Atypical headers	Large cookies, custom headers	Real traffic captures (reverse proxy)

There is a direct relationship between precision, tuning effort, and operational burden. Overly **aggressive WAFs** without fine-grained control **tend to be relaxed over time or partially disabled.**

Balanced Metrics for Evaluating WAFs

If you only look at **TPR,** you may choose a solution that **“blocks a lot”** but breaks the business. If you only look at **FPR** (or its inverse, TNR), you may choose a **“friendly” solution that lets threats through.**

That is why serious lab-style evaluations use balanced metrics such as **Balanced Accuracy (BA)**, which combines the ability to block malicious traffic (TPR) and **allow legitimate traffic (TNR).**

A useful benchmark should expose three metrics:

- **Security Quality (TPR):** ability to block attacks
- **Detection Quality (TNR / inverse of FPR):** ability to allow legitimate traffic
- **Balanced Accuracy (BA):** arithmetic mean of TPR and TNR

How to interpret TPR/FPR/BA without falling into traps:

Metric pattern	Looks like	Means in production
High TPR + high FPR	“Very secure”	High operational risk: heavy tuning, outages, friction
Low TPR + very low FPR	“Doesn’t break anything”	High security risk: real attacks pass through
High BA (high TPR + high TNR)	“Balanced”	Best starting point: fewer surprises, less tuning

Behavior Against Volumetric Attacks and Layered Attacks

When **evaluating a modern WAF**, it is essential to distinguish attacks by the layer at which they operate. This distinction is especially relevant in **SaaS and cloud-managed models**, where the WAF is typically part of a broader protection chain.

Web applications and **APIs are primarily exposed to two categories** of attacks:

Volumetric attacks (L3/L4)

These aim to saturate network bandwidth or connection tables through massive TCP/UDP floods or SYN floods.

Application-layer attacks (L7)

These consist of HTTP(S) requests that appear legitimate at the network level but exhaust application or backend resources.

Expected WAF Behavior Under Traffic Spikes

The key question is **how the platform behaves** when **traffic increases** abruptly.

- Sustained **L7 inspection** and filtering under load, preventing malicious requests from exhausting backend resources.
- Progressive **defensive mechanisms**, such as adaptive rate limiting, behavioral challenges, and reputation- or anomaly-based blocking.
- Controlled degradation, **prioritizing application availability** over indiscriminate blocking or platform instability.
- **Tenant-level isolation**, ensuring that traffic spikes or attacks affecting other customers do not impact performance or security decisions for a given tenant.

In SaaS environments, traffic spikes occur frequently, and customers lack direct control over the underlying infrastructure.

Layer	Attack Type	What to Evaluate
L3/L4	Volumetric DDoS	Upstream mitigation, integration clarity
L7	HTTP request floods	L7 filtering under load, latency impact
L7	Low-rate high-cost attacks	Behavioral detection, anomaly thresholds
L7	SQL Injection (SQLi)	Rule transparency, tuning depth
L7	Cross-Site Scripting (XSS)	Context awareness, encoding handling
L7	API abuse	Per-endpoint controls, API visibility
L7	Business logic abuse	Custom rules, false-positive handling
L7	Structured payload abuse	Parsing depth, configurable limits
L7	Encoding & evasion techniques	Normalization process, payload visibility
Cross-layer	TLS-encrypted traffic	TLS termination model, performance impact
L7	Bot & automation abuse	Behavioral signals, mitigation flexibility
SaaS-specific	Cross-tenant impact	Tenant isolation guarantees

Technical Validation and Evaluation Checklist

Before adopting a WAF, it is essential to perform real technical testing. This includes **detection tests, latency measurements, load scenarios, and false-positive analysis.**

In 2026, **evaluating a WAF**—especially in SaaS or cloud-managed models—means validating not only what it blocks, but how it behaves **under realistic operating conditions.**

What to validate in 2026:

Component	What it must include	Why it matters
Legitimate dataset	Real requests (UI + APIs), complex JSON, uploads, full flows	Reveals false positives and real operational cost
Malicious dataset	Varied payloads + modern collections and bypasses	Measures real coverage, not “demo blocking”
Operation mode	Prevention mode (blocking)	Detection behavior changes significantly under enforcement
Result logging	Request/response + decision, exportable	Enables reproducibility and fair comparison
Metrics	TPR + FPR/TNR + Balanced Accuracy	Avoids decisions biased toward one extreme
Configuration	Default profile (no tuning) + strict profile if available	Measures the real cost of increasing security

Technical Checklist 2026

- **Architecture:** Position in the traffic path, early vs late inspection, integration with routing and load balancing.
- **Detection:** Effective coverage against representative and up-to-date malicious datasets.
- **False positives:** Impact on complex legitimate traffic and full application flows.
- **APIs:** Real support for REST, GraphQL, and SOAP; JSON/XML parsing; handling of large payloads.
- **Performance:** Latency impact and stability under sustained load.
- **Visibility:** Actionable logs, decision traceability, and change auditing.
- **Operations:** Rule control, automation, APIs, and safe change workflows.
- **Costs:** Predictable cost model that does not penalize security or observability.

Operational Capacity and Limits

In SaaS WAFs, beyond the points above, you must also confirm:

- Whether **true isolation** exists (traffic, resources, and rules) or only logical separation.
- Which **operational limits** apply (payload size, number of rules, domains, rate limits, log retention).
- Whether **logs are complete** and exportable, and whether export involves cost or restrictions.
- The real level of **granular tuning** available (per service, route, host, or API).
- Behavior under **traffic spikes:** controlled degradation, defensive blocking, or service failure.
- The level of **support during incidents:** response times, scope, and technical visibility.

Closing

Evaluating a WAF in 2026 requires going beyond feature lists or promises. Only a **rigorous technical analysis**, representative testing, and balanced metrics can determine whether a solution is ready to **protect modern applications** without compromising availability or evolution.

And if the **chosen model is SaaS**, the bar is not lower. Delegating infrastructure means the **technical team must gain**, in exchange, isolation, observability, operational control, and predictability. A **WAF consumed as a service** is only a real advantage if it can maintain that **technical level** in a verifiable way.

Further Technical Resources

Documentation, configuration guides, and deployment examples are available in our → Knowledge Base.

Technical Inquiry

If you would like to discuss specific performance requirements or architectural considerations, our engineering team can provide technical guidance → info@skudonet.com