# Case Study: AWS WAF & CloudFront Security Optimization (Anonymized)

**Category:** Cloud & Infrastructure Security
**Duration:** 2 Weeks | **Engagement Type:** WAF Tuning & Edge Security Review
**Tools:** AWS WAF, CloudFront, CloudWatch, Regex Pattern Sets, ElastAlert2, SNS, Slack

---

## Context

A global SaaS platform leveraging **AWS CloudFront** for API delivery and static asset caching noticed intermittent false-positive blocks from its WAF.

Some legitimate requests were being dropped under managed rule groups (particularly XSS and SQLi bodies), causing client-side 403 errors and degraded user experience.

The client requested a **complete AWS WAF and CloudFront optimization engagement** — aiming to reduce false positives, maintain strict security posture, and establish proactive alerting for future anomalies.

---

## Approach

The review followed the **AWS WAF Security Automation Reference Architecture** and your custom **traffic-aware tuning methodology**:

1. **Baseline Audit**
    - Extracted WAF metrics (BlockedRequests, AllowedRequests) from CloudWatch for 14 days.
    - Correlated logs with application access data to identify legitimate API blocks.
2. **Managed Rule Review**
    - Evaluated all active rule groups: AWS Core, Anonymous IP, SQLi, XSS, and Bot Control.
    - Analyzed each rule’s trigger patterns and match scopes (Body, URI, QueryString).

3. **Custom Rule Engineering**

    - Created targeted **regex pattern sets** to whitelist safe URIs (e.g., `/api/v1/crm/invoices`, `/api/v2/reviews/{uuid}/photo`).

    - Introduced conditional statements combining `LabelMatch` + `NotStatement` to bypass known false-positive paths.

4. **Alerting & Monitoring Setup**

    - Integrated **ElastAlert2** with Slack via SNS topic for near real-time blocked-request notifications.

    - Deployed rule-change tracking via CloudWatch alarms on WAF configuration updates.

---

## Key Findings

| **Severity** | **Count** | **Highlight** |
|---|---|---|
| High | 3 | Legitimate POST requests blocked due to aggressive `CrossSiteScripting_Body` inspection |
| Medium | 4 | Duplicate managed rules causing double evaluation latency (~2.3s delay) |
| Medium | 2 | Lack of granular geo-based restrictions for unused regions |
| Low | 6 | Unmonitored rule changes without SNS alerts |

---

## Remediation Summary

- Refined WAF rule priorities and merged overlapping AWS-managed sets.

- Created **custom regex allowlists** for verified safe API endpoints.

- Enabled **geo-based filtering** to restrict traffic from non-operational regions.

- Tuned default action behavior: switched from global “Block” → **Conditional Allow** (based on URI + LabelMatch).

- Integrated **CloudWatch + ElastAlert2 pipeline** for centralized visibility.

- Documented reusable **WAF template** for future deployments.

---

## Outcome

- False-positive request rate reduced by **~40%** within 48 hours
- Achieved **zero customer-facing 403 incidents** post-deployment
- Average CloudFront response latency improved by **31%**
- Real-time visibility through Slack-based blocked-request alerts
- Standardized deployment templates now used in both staging & production

---

## Executive Summary

This engagement combined **data-driven WAF optimization** with continuous monitoring and alert automation.

Through traffic correlation, regex tuning, and intelligent rule layering, the client achieved a **security posture that balanced protection and performance**.

By integrating alerts into their existing Slack workspace and maintaining versioned WAF configurations, the client now sustains a **self-healing security workflow** with minimal manual oversight.

---