Case Study: Comprehensive Web VAPT & WAF Rule Optimization for SaaS Platform (Anonymized)

Category: Web & API Security
Duration: 4 Weeks | **Engagement Type:** Web VAPT + AWS WAF Hardening
Tools: AWS WAF, Burp Suite, OWASP ZAP, Regex Pattern Sets, CloudWatch, ElastAlert2, Fluent Bit

---

Context

A SaaS company offering multi-tenant HR automation services had recently migrated its infrastructure to AWS CloudFront + ALB setup.

Following reports of **false-positive WAF blocks** and **intermittent 403s on legitimate API calls**, they sought an end-to-end **web application VAPT and WAF optimization engagement** to strike the right balance between security and usability.

The scope covered both the **frontend and backend APIs** across staging and production environments — ensuring strong protection while maintaining zero business disruption.

Approach

The engagement followed a combined **offensive + defensive security methodology**:

1. **Reconnaissance & Baseline Mapping** – Enumerated endpoints, parameters, and WAF rule coverage across environments.

2. **Web Application Penetration Testing** – Conducted manual and automated VAPT covering OWASP Top 10 and custom business logic flaws.

3. **False Positive Analysis** – Mapped CloudWatch metrics (`BlockedRequests`, `CountedRequests`) to user traffic to identify legitimate drops.

4. **WAF Rule Tuning** – Reviewed AWS Managed Rule Groups and refined custom regex pattern sets for API exceptions.

5. **Validation & Hardening** – Simulated attacks to validate rule precision while ensuring legitimate traffic flow.

6. **Alerting & Observability Setup** – Integrated ElastAlert2 + Slack for real-time detection of blocked or anomalous requests.

Key Findings

Category	Impact	Description
XSS (False Block)	Medium	`/api/v1/crm/invoices` blocked due to overly aggressive body inspection in `CrossSiteScripting_Body` rule
Broken Access Control	High	Unrestricted file upload endpoint allowed unsafe extensions before WAF interception
Misconfigured WAF Rules	Medium	Default action set to “Block” without IP-based whitelist for internal testing
Sensitive Data Exposure	Low	Verbose error stack traces visible in `500` responses during test failures

---

Remediation Summary

- Created **custom regex allowlists** for safe API endpoints (`/api/v2/reviews/{uuid}/photo`, `/api/v1/crm/invoices`).
- Enforced **strict MIME validation** and file extension controls at backend upload handlers.
- Modified **WAF default action** from global block → conditional allow based on request origin.
- Tuned AWS Managed Rules to ignore benign parameters (`utm_`, `ref_`, `source_`).
- Added **CloudWatch alarms** for anomaly spikes and Slack notifications through SNS + ElastAlert2.
- Deployed **versioned rule sets** for rollback capability during regression testing.

Outcome

- Reduced false-positive WAF blocks by **~40%**, improving production reliability
- Achieved **zero customer-facing 403s** post-deployment
- Strengthened application posture against XSS, LFI, and injection vectors
- Established **end-to-end observability** from WAF → CloudWatch → ELK → Slack
- Delivered reproducible **WAF rule templates** for staging and production parity

---

Executive Summary

This engagement demonstrated the critical importance of aligning **offensive testing** (VAPT) with **defensive optimization** (WAF tuning).

Through precise rule engineering, regex-based whitelisting, and log-driven analytics, the SaaS platform achieved a mature and adaptive web defense posture.

The client's internal DevOps team was trained to manage future rule iterations safely using **version-controlled WAF templates**, ensuring sustained visibility and operational agility.
