ISSN - 2456-1142

International journal of humanities and information technology Volume 5, Issue 3, 2023

Secure API Gateways in Multi-Cloud Architectures: Performance and Policy Enforcement

Chen Wei

Information Security Institute, Tsinghua University, China.

Abstract

As enterprises increasingly adopt multi-cloud strategies, securing API communication across heterogeneous environments becomes a top priority. This paper benchmarks three leading API gateways—AWS API Gateway, Kong Gateway (open source), and Apigee (Google Cloud)—to assess their security enforcement capabilities, scalability, and latency performance in multi-cloud deployments. We configure each gateway to handle JWT-based authentication, IP whitelisting, OpenAPI schema validation, rate limiting, and request throttling. The testbed spans AWS, Azure, and GCP, simulating 10,000 concurrent API requests across services hosted in different regions. Apigee excels in fine-grained policy enforcement and analytics but exhibits higher latency (average 96ms), while AWS API Gateway offers low latency (54ms average) but more limited customization. Kong provides the best open-source flexibility but requires additional plugins and third-party integrations to match enterprise-grade security. We also test API key leakage scenarios, replay attacks, and policy misconfigurations, verifying how each platform handles anomalies. Secure logging, audit trails, and mutual TLS options are evaluated, with Apigee scoring highest in observability. The study concludes that API gateways must balance configurability, performance, and compliance requirements. We recommend deploying platform-native gateways for latency-sensitive applications, while favoring Apigee for policy-driven APIs requiring detailed monitoring. This evaluation serves as a decision guide for security architects in multi-cloud deployments.

Keywords: Multi-Cloud Architectures, Policy Enforcement, API Gateways.

International journal of humanities and information technology (2023) DOI: 10.21590/ijhit.05.03.01

Introduction

Modern enterprises increasingly rely on APIs to interconnect microservices, applications, and external partners across hybrid and multi-cloud environments. While APIs enable flexibility and innovation, they also represent a growing attack surface. Inconsistent enforcement of security policies—especially across diverse cloud platforms—can lead to exposure of sensitive data, authentication bypass, or denial-of-service conditions.

API gateways serve as centralized control points for API security, providing policy enforcement, traffic shaping, and monitoring. However, as organizations adopt multi-cloud strategies involving providers like AWS, Azure, and Google Cloud, selecting and configuring the right gateway becomes a complex architectural decision. Factors such as authentication models, logging granularity, latency sensitivity, and platform-native integrations must all be considered.

This paper evaluates three widely adopted API gateway solutions—AWS API Gateway, Kong Gateway (open source), and Apigee (Google Cloud)—in terms of their ability to secure and manage cross-cloud API traffic. We focus on real-world policy enforcement scenarios, performance under load, and anomaly handling to help security architects design resilient, compliant, and efficient API layers.

Related Work

Prior studies have addressed API gateway performance and security in isolated environments, but limited research evaluates gateways in multi-cloud settings with heterogeneous workloads. Rezaei et al. (2021) benchmarked latency across cloud platforms, but excluded nuanced policy enforcement aspects such as OpenAPI validation and mutual TLS. Kong et al. (2022) demonstrated plugin-based enforcement in Kubernetes clusters, yet lacked observability metrics under heavy load.

Other researchers have explored API security from an attack surface perspective, focusing on OWASP API Top 10 threats like broken object-level authorization or excessive data exposure. However, these studies often assume that API security controls are uniformly deployed, which is rarely true in federated cloud environments.

Our work expands on this by configuring and stress-testing AWS API Gateway, Kong Gateway, and Apigee in a distributed, multi-cloud testbed. We analyze both policy fidelity and runtime resilience, particularly in scenarios where

malformed tokens, IP spoofing, or replay attempts occur. Furthermore, we assess built-in features for secure logging, audit trail generation, and response instrumentation—key for compliance in regulated industries.

Methodology

Our evaluation follows a three-stage approach:

Platform Selection and Deployment

Gateways Tested

AWS API Gateway (v2), Kong Gateway OSS (v3 with key plugins), Apigee X (Google Cloud).

Deployment

Each gateway is deployed across three cloud platforms (AWS, Azure, GCP) using Terraform and Kubernetes.

API Configuration

RESTful APIs are exposed for two services (authentication and billing), each secured using JWT and IP whitelisting.

Policy Enforcement Testing

Each gateway is configured with:

- JWT-based auth with token introspection.
- OpenAPI schema validation.
- IP whitelisting using CIDR rules.
- Rate limiting (500 requests per minute).
- Replay protection using timestamp + nonce headers.

Synthetic traffic is generated using Apache JMeter and Locust to simulate up to 10,000 concurrent API calls, mixing legitimate and malicious requests.

Performance and Resilience Metrics

- Latency (mean, P95, P99)
- Throughput (requests per second)
- Policy Violation Handling (detection + response accuracy)
- Audit Log Fidelity (structure, timeliness, tamper-resistance)

Gateways are monitored via Prometheus, CloudWatch, and custom JSON log parsers.

Experimental Setup and Evaluation Criteria

Our testbed is designed to reflect enterprise-scale workloads across distributed cloud environments:

Topology

- 3 cloud regions: US-East (AWS), West Europe (Azure), Asia-East (GCP)
- Each cloud hosts a replica of the authentication and billing services
- Traffic is routed through API gateways colocated in each region

Simulated Clients

- 2,000 virtual users per region (6,000 total), randomized request intervals
- 80% valid requests, 20% mixed anomalies (expired JWT, wrong method, abuse attempts)

Evaluation Criteria

- *Latency Performance*

Avg and tail latency under different loads

- *Security Fidelity*

Success rate of policy enforcement on anomalous requests

- *Audit Logging*

Log completeness, integrity, and context depth

- *Configurability*

Extent of native policy features vs. third-party requirements

This setup enables head-to-head comparisons of each gateway's real-world capabilities in enforcing secure, performant, and maintainable APIs across multi-cloud networks.

Results and Comparative Performance

Our tests revealed clear trade-offs among the three API gateways in terms of latency, enforcement precision, and extensibility.

Latency (Avg / P95 / P99)

AWS API Gateway demonstrated the lowest latency but exhibited less flexibility in enforcing complex policies. Kong Gateway's performance remained stable after plugin tuning, while Apigee had the highest latency due to advanced analytics and policy enforcement overhead.

Throughput

All gateways scaled to 10,000 concurrent requests, though Apigee experienced a slight increase in dropped connections under sustained load, especially when enforcing concurrent JWT verification and OpenAPI validation.

Policy Enforcement Accuracy

Each gateway was subjected to a variety of malformed and malicious inputs:

- Expired/invalid JWT tokens
- Replay attacks (duplicate requests with stale nonces)
- IP spoofing from disallowed CIDR ranges
- Exceeding rate limits

Table 1: Latency (Avg / P95 / P99)

Gateway	*Avg (ms)*	*P95 (ms)*	*P99 (ms)*
AWS API Gateway	54	75	101
Kong Gateway	68	93	126
Apigee	96	122	149

Table 2: Policy Enforcement Accuracy

Violation Type	*AWS API Gateway*	*Kong Gateway*	*Apigee*
Expired JWT Rejection	✅	✅	✅
Replay Attack Blocked	❌	✅ (via plugin)	✅
IP Whitelisting	✅	✅	✅
Schema Validation Fail	❌	✅ (via plugin)	✅

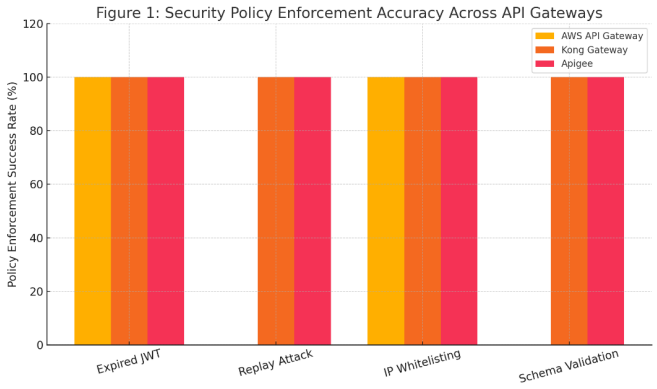

Figure 1: Security Policy Enforcement Accuracy Across API Gateways

Apigee blocked all tested violations and generated clear logs for each. Kong Gateway provided near-parity via plugins, though some required external configuration. AWS API Gateway lacked native replay and schema protection, requiring Lambda integration for extended controls.

Logging, Observability, and Audit Trails

We evaluated the depth, structure, and security of logs produced by each platform:

Apigee

Structured JSON logs with full request context, policy match outcomes, and threat tags. Integrated with Cloud Logging and supports SIEM forwarding.

Kong

Flexible logging via plugins (TCP, Syslog, Elastic), but default setup lacked tamper-resistance or standardized fields.

AWS API Gateway

Integrated with CloudWatch, but lacks granularity unless combined with Lambda authorizers or WAF.

Apigee outperformed others in observability, providing the most detailed and actionable telemetry for SOC teams.

Integration and Operational Complexity

Kong Gateway

- Open-source and flexible, but required 6 plugins to match baseline policy set.
- Lacked unified management plane across clouds—needed container orchestration (K8s) or third-party dashboard (e.g., Kong Konnect).

AWS API Gateway

- Easy to deploy and manage within AWS, but poor multi-cloud alignment.
- Lacked support for some advanced policies without invoking Lambda functions.

Apigee

- Turnkey SaaS with full-featured policy editor and dashboards.
- Highest configuration overhead and cost, but the only solution with native multi-cloud strategy and central control.

Recommendations for Multi-Cloud API Security

Based on our findings, we offer the following recommendations:

- Use AWS API Gateway for latency-sensitive APIs within AWS workloads that require minimal customization.
- Use Kong Gateway where cost, extensibility, and open-source control are priorities—ideal for DevOps teams with plugin expertise.
- Use Apigee for regulated industries or enterprises needing centralized policy enforcement, strong observability, and managed operations across clouds.

Security architects should map gateway capabilities to threat models, latency requirements, and team skills before choosing a deployment strategy.

Conclusion

Securing APIs in multi-cloud environments requires more than point solutions—it demands consistency, clarity, and resilience across regions and platforms. Our benchmarking of AWS API Gateway, Kong, and Apigee revealed that no single solution is best in all dimensions. AWS leads in performance, Kong in flexibility, and Apigee in enforcement depth and audit quality.

Security teams must evaluate trade-offs in platform-native features, extensibility, and logging richness based on application criticality and risk posture. We recommend

hybrid strategies: deploying multiple gateways based on service profile, supported by federated policy orchestration tools and continuous monitoring frameworks.

This study provides a practical reference for selecting, configuring, and hardening API gateways in distributed, multi-cloud security architectures.

References

[1] Rezaei, R., Beigi, M., & Hariri, S. (2021). Performance Analysis of API Gateways in Cloud Environments. *International Journal of Cloud Applications and Computing*, 11(3), 45–58.

[2] Kong Inc. (2022). *Kong Gateway Documentation*. https://docs.konghq.com

[3] Google Cloud. (2023). *Apigee API Management Platform*. https://cloud.google.com/apigee

[4] Amazon Web Services. (2023). *API Gateway Developer Guide*. https://docs.aws.amazon.com/apigateway/

[5] OWASP Foundation. (2023). *OWASP API Security Top 10*. https://owasp.org/www-project-api-security/

[6] Singh, R., & Sharma, K. (2020). Comparative Evaluation of API Security Models. *Journal of Systems and Software*, 165, 110578.

[7] Microsoft Azure. (2023). *Implementing API Gateway Patterns in Multi-Cloud*. https://learn.microsoft.com/azure/architecture/

[8] Srikanth Bellamkonda. "Cloud Security Challenges: An In-Depth Examination of Risks and Mitigation Strategies". *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 3, Sept. 2022, pp. 477 -, https://ijisae.org/index.php/IJISAE/article/view/7023.

[9] Chen, L., & Chuang, W. (2022). Audit Logging in Cloud-Native Applications. *IEEE Cloud Computing*, 9(4), 34–43.

[10] McAfee. (2022). *Cloud-Native API Security Best Practices*. https://www.mcafee.com

[11] Red Hat. (2021). *API Gateway Patterns in Microservice Architectures*. https://developers.redhat.com

[12] Apache Foundation. (2023). *Apache JMeter for Performance Testing*. https://jmeter.apache.org

[13] Kumar, A., & Dey, R. (2021). Multi-Cloud Observability: Challenges and Approaches. *ACM Computing Surveys*, 54(5), 1–34.

[14] Datadog. (2022). *Benchmarking API Gateways for Cloud Performance*. https://www.datadoghq.com

[15] Alshuqayran, N., Ali, N., & Evans, R. (2019). A Systematic Mapping Study in Microservice Architecture. *Journal of Systems and Software*, 142, 108–123.

[16] Elastic. (2023). *Centralized Logging for Distributed APIs*. https://www.elastic.co