iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.68953

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Secure API Gateway with Rate Limiting and JWT Authentication

Shivam (MAIT)[1], Mukti Gupta (IGDTUW)[2]

Abstract: ***APIs form the foundation of modern applications, yet they remain highly vulnerable to various attacks such as brute force, token abuse, and DDoS. In this paper, we present a layered approach to secure APIs using API gateways, JWT-based authentication, and rate limiting. Through comparative analysis of tools like AWS API Gateway, Kong, Apigee, and NGINX, we highlight key features, strengths, and trade-offs. Our findings suggest that integrating JWT with rate limiting significantly improves API security and scalability across architectures.***
Keywords: API Security, JWT, Rate Limiting, API Gateway, Microservices

I. INTRODUCTION

With the rise of digital transformation and the increasing reliance on cloud-native architectures, RESTful APIs have become the cornerstone of modern application development. These APIs allow seamless integration between services and support modular application designs through microservices. However, as APIs are often exposed publicly or semi-publicly, they introduce new attack surfaces. Securing these interfaces is thus of paramount importance to ensure data privacy, service availability, and system integrity. This paper argues that traditional access control mechanisms are insufficient for dynamic, stateless, and scalable API systems. The implementation of JWT and rate limiting via an API gateway introduces a security-centric, scalable approach that aligns with the needs of modern application environments.

With the explosion of microservices and interconnected platforms, APIs are now the primary mode of communication between services. However, this also makes them a prime target for malicious actors. Traditional security methods fall short when it comes to stateless, scalable communication. This paper explores the integration of API gateways, JSON Web Tokens (JWT), and rate limiting to construct a resilient and secure API architecture.

II. LITERATURE SURVEY

ElHejazi&Muragaa proposed a JWT-based secure mechanism specifically tailored for SDN controller REST APIs. While innovative, their work did not account for full SDN ecosystems or measure system performance under stress. On the other hand, Bucko et al. built a behavior-based JWT system that adapted token policies according to user behavior. This approach faced challenges with user fairness and browser compatibility, especially under mobile-first scenarios. Further, few existing studies have explored comprehensive systems that integrate authentication, authorization, rate limiting, and traffic filtering holistically. Our work aims to address these gaps by benchmarking production-ready tools in real-world contexts.

Previous studies have shown JWT to be a promising mechanism for authentication:

ElHejazi & Muragaa introduced JWT-based REST API security for SDN controllers but ignored other SDN components and lacked comparative analysis.

Bucko et al. focused on behavior-based JWT auth systems but faced issues with fairness for mobile and cross-browser users.

While effective in parts, both solutions lacked scalability, real-world testing, or full-system integration.

III. JWT AUTHENTICATION

JWTJSON Web Token (JWT) is a widely adopted method for stateless and secure data transmission in distributed systems. Designed as a compact and self-contained format, JWT allows information exchange between parties without requiring persistent server-side sessions.

1) A JWT consists of three components:

- Header: Specifies the token type and the cryptographic algorithm used.
- Payload: Contains claims or user-defined data such as user identifiers, roles, or permissions.
- Signature: Ensures the token's integrity by cryptographically signing the header and payload using a secret or private key.

©IJRASET: All Rights are Reserved

2) Key Advantages:

- Stateless and Scalable: Eliminates the need for server-side session storage, making it ideal for microservice and cloud-native architectures.
- Cross-platform Compatibility: Can be securely transmitted over various platforms and clients.

3) Potential Drawbacks:

- Token Theft: If intercepted, a token can be reused by attackers unless HTTPS, short expiries, and secure storage are enforced.
- Validation Complexity: Improper validation logic can lead to unauthorized access or token forgery.
- Expiry Management: Balancing token lifespan is challenging; short-lived tokens require frequent reauthentication, while long-lived tokens increase security risks.

In essence, JWT provides a decentralized and efficient mechanism for authenticating requests, but it requires rigorous implementation to ensure robustness against modern threats.

IV. RATE LIMITING

In the realm of API-driven systems, where scalability often invites abuse, rate limiting emerges as a fundamental defence mechanism. It controls the number of requests a client can make within a defined time window, thereby maintaining system stability and protecting against misuse.

1) Primary Objectives of Rate Limiting:

- Mitigating Brute Force Attacks: Repeated login attempts can be curtailed by limiting the number of authentication requests.
- Preventing Distributed Denial of Service (DDoS): Rate limits reduce the impact of malicious flooding, maintaining service availability.
- Blocking Automated Abuse: Bots or scripts attempting spam or excessive data scraping are effectively throttled.

Beyond security, rate limiting enforces fair resource usage, ensuring equitable access for all clients and protecting backend services from overload.

2) Popular Algorithms for Implementation:

- Token Bucket Algorithm: Tokens are added to a bucket at a fixed rate. Each request consumes a token; if the bucket is empty, the request is denied or delayed.
- Leaky Bucket Algorithm: Requests enter a queue and are processed at a constant rate, ensuring smooth flow and preventing sudden spikes.

Both techniques provide mechanisms to balance performance and protection, enabling APIs to handle legitimate traffic while shielding against anomalies.

In modern architectures, rate limiting is often implemented at the API Gateway level, integrating seamlessly with authentication and logging mechanisms to form a comprehensive security strategy.

V. API GATEWAY OVERVIEW

API Gateways serve as the critical intermediaries between client requests and backend services in modern distributed systems. By centralizing a range of functions — such as routing, load balancing, authentication, rate limiting, and logging —
API Gateways streamline communication, reduce overhead, and enhance security.

1) Leading API Gateways:

- AWS API Gateway: Fully managed service that integrates seamlessly with AWS services, providing high scalability and security.
- Kong: Open-source API Gateway with a rich plugin ecosystem, ideal for flexible deployment and customization in diverse environments.
- Google Apigee: An enterprise-grade solution designed for comprehensive API lifecycle management, offering analytics, monitoring, and security features.
- NGINX: Known for its high performance and flexibility, NGINX acts as a reverse proxy, load balancer, and API Gateway, with extensive support for custom configurations.

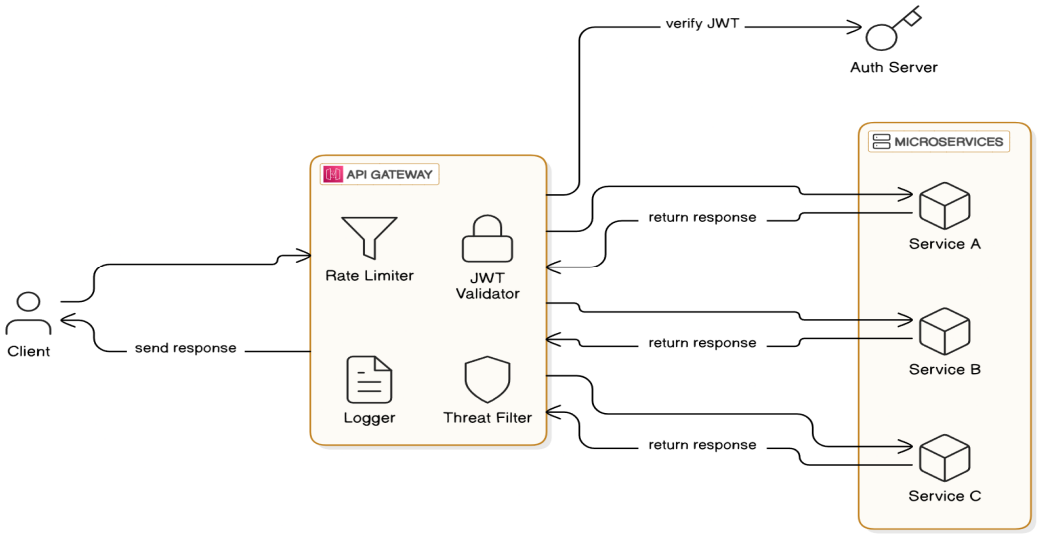


Fig. API Gateway Security Architecture

VI. COMPARATIVE ANALYSIS OF TOOLS

Feature	Kong	AWS API Gateway	Apigee	NGINX
JWT Support	Yes	Yes	Yes	Manual
Rate Limiting	Yes	Yes	Yes	Yes
Open Source	Yes	No	No	Yes
Ease of Setup	Medium	Easy	Medium	Hard
Performance	High	High	High	Very High
Custom Plugins Support	Yes	No	Yes	Yes
Dashboard/GUI	Yes	Yes	Yes	No
Community Support	Strong	Strong	Strong	Strong
Deployment Options	Self-hosted, Cloud	Cloud	Cloud	Self-hosted
Pricing	Free &Paid	Pay-as-you-go	Premium	Free &Paid
OAuth2 Support	Yes	Yes	Yes	Manual
gRPC Support	Yes	Yes	No	Yes
GraphQL Support	Partial	Yes	Yes	Manual
API Analytics	Basic	Advanced	Advanced	Manual
Security Features	Advanced	Advanced	Advanced	Customizable
Scalability	High	High	High	High
Load Balancing	Yes	Yes	Yes	Yes
Logging & Monitoring	Yes	Yes	Yes	Manual
Caching	Yes	Yes	Yes	Yes
Authentication Methods	JWT, OAuth2, Key	JWT, OAuth2, IAM	JWT, OAuth2, API Keys	Custom Scripts

VII. REAL-WORLD USAGE

Gateway	Company
Kong	Yahoo
AWS API Gateway	Netflix, Airbnb
Google Apigee	Citibank
NGINX	Dropbox, Github, Adobe

These companies validate the scalability and reliability of modern API security platforms.

VIII. THREAT MODEL

In real-world deployments, APIs are constantly under threat from adversaries attempting to exploit authentication flaws, inject malicious payloads, or bring down services through brute force and denial-of-service techniques. JWTs, while secure, must be implemented carefully with encryption (JWE) or proper signing (JWS), and strict validation of claims such as expiration, audience, and issuer. Token replay attacks occur when a previously valid token is intercepted and reused. Rate limiting counters this by capping request frequency, making brute-force or automated attacks infeasible. DDoS attacks target availability, overwhelming the infrastructure. An API gateway can serve as the first line of defense, applying IP blacklisting, anomaly detection, and limiting resource-intensive endpoints. These mechanisms create layered security, enhancing both resilience and response time.

The paper considers major attack vectors such as:

- Brute-force login attempts
- Token replay attacks
- DDoS (Distributed Denial of Service)
- SQL injection and payload tampering

Implementing gateway-level policies neutralizes many of these threats before they reach the application layer.

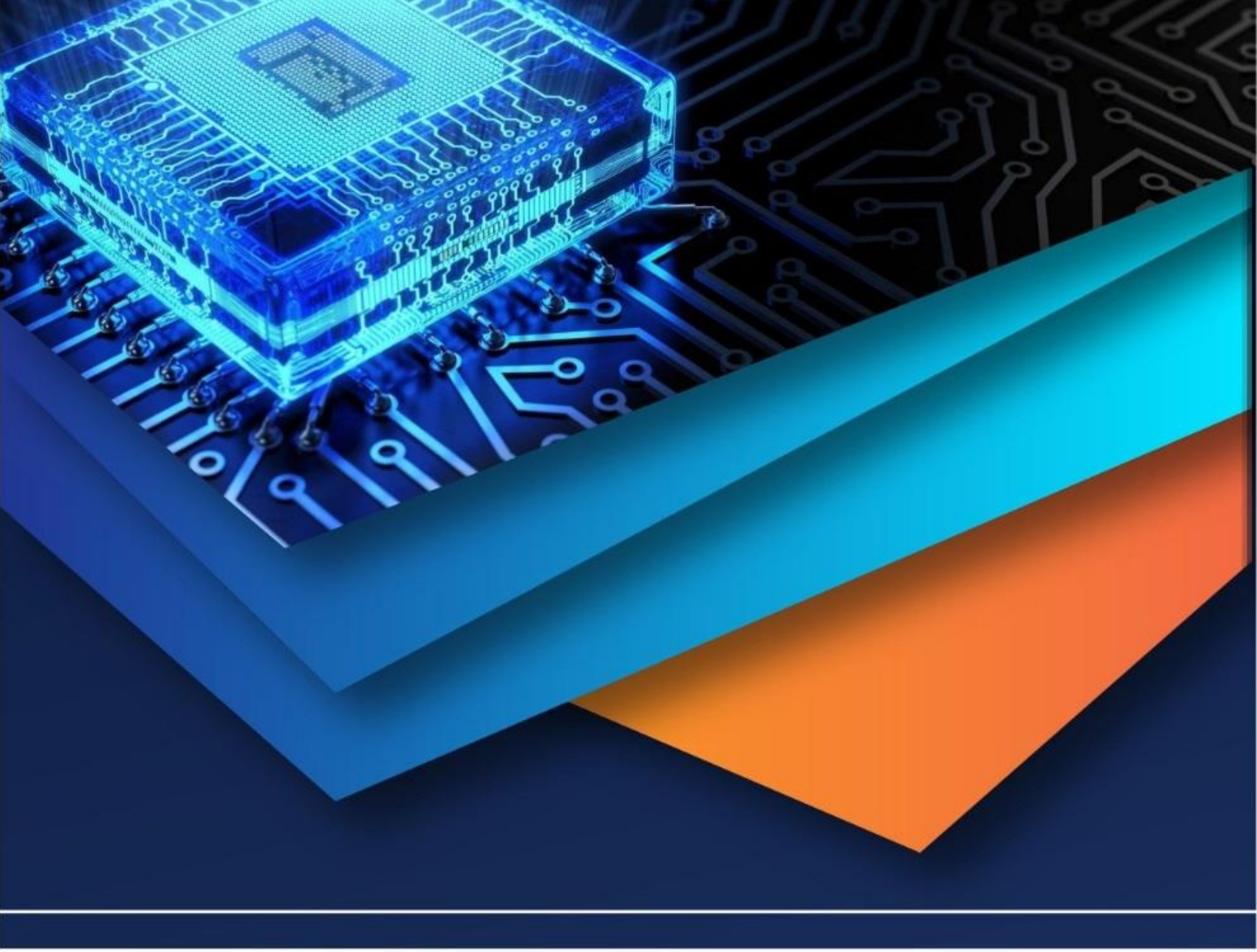
IX. CONCLUSION

The modern internet infrastructure relies heavily on secure, scalable, and performant APIs. Our research shows that employing JWT-based authentication ensures stateless and scalable validation across microservices, while rate limiting protects against high-frequency attacks and promotes fair API usage. Integrating these within an API gateway abstracts complexity from individual services and allows central policy management. The comparative analysis shows that tools like Kong and AWS Gateway provide rich feature sets that are production-ready. Although Apigee offers enterprise-grade capabilities, its cost might be restrictive for startups. NGINX, though powerful, demands more manual configuration. Overall, our work emphasizes that robust API security requires proactive design choices—layered defenses at the gateway, token hygiene, and intelligent traffic controls. Future exploration could include adaptive rate limiting using machine learning, bot behavior prediction, and automated threat response mechanisms at the gateway level.

Securing APIs is critical in the current application ecosystem. This research demonstrates that combining JWT authentication with rate limiting inside a well-configured API gateway significantly enhances both security and scalability. While tools like AWS and Kong offer out-of-the-box features, the right choice depends on project needs and budget. Future work may explore AI-based adaptive rate limiting and anomaly detection in API traffic.

REFERENCES

[1] JWT.io Documentation
[2] Kong, AWS, Apigee, NGINX Official Docs
[3] ElHejazi M.F. et al., “Improving the Security and Reliability of SDN Controller REST APIs…” IEEE
[4] Bucko A. et al., “Enhancing JWT Authentication and Authorization in Web Applications…” IEEE



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)