Enhancing API Security: A Hybrid Approach Combining OAuth2, JWT, and Session-Based Authentication for Scalable and Secure Applications

1 Abstract

With the increasing reliance on RESTful APIs for web and mobile applications, authentication and security mechanisms have become critical. OAuth2, JWT, and Session-Based Authentication are widely used techniques, but each has its own strengths and limitations. This paper introduces a hybrid authentication model that integrates these three methods to enhance security, performance, and scalability. We analyze real-world scenarios, benchmark efficiency, identify vulnerabilities, and explore mitigation strategies. Additionally, we discuss the role of Zero Trust principles and blockchain technology in securing API authentication.

2 Introduction

APIs serve as the backbone of modern applications, enabling seamless data exchange and integration. However, securing APIs is challenging due to evolving cyber threats. Traditional authentication mechanisms such as session-based authentication face scalability issues, while JWT and OAuth2 introduce token-related vulnerabilities. This research proposes a hybrid model combining these approaches to achieve robust security while maintaining performance and scalability.

3 Background and Related Work

3.1 OAuth2 Authentication

OAuth2 provides delegated authorization, enabling secure third-party access. However, it is susceptible to token leakage and phishing attacks.

3.2 JSON Web Token (JWT)

JWT offers a stateless authentication method, improving performance. However, long-lived tokens pose security risks such as replay attacks and token theft.

3.3 Session-Based Authentication

Session authentication is a secure, server-side approach but lacks scalability due to session storage requirements.

3.4 Zero Trust Security Model

A security paradigm that assumes no implicit trust, enforcing strict identity verification at all levels.

3.5 Blockchain for Authentication

Blockchain technology introduces decentralized authentication mechanisms that prevent token forgery and replay attacks.

4 Security and Performance Benchmarking

4.1 Experimental Setup

API implemented using Spring Boot with PostgreSQL. Performance tested using Apache JMeter under varying authentication loads. Security vulnerabilities assessed using OWASP ZAP.

4.2 Performance Analysis

- **Latency and Throughput:** JWT performs better under high loads, while session-based authentication slows down due to server-side storage.
- **Scalability:** OAuth2 excels in multi-service applications but requires additional token validation processing.
- **Resource Utilization:** Session-based authentication consumes more memory due to state persistence.

5 Proposed Hybrid Authentication Model

We propose a hybrid model that integrates:

- Session-based authentication for initial login, ensuring secure user verification.
- JWT for API-to-API communication, reducing server load and enhancing performance.

- OAuth2 for third-party integrations, enabling secure delegated access.
- Blockchain-based identity verification, preventing token forgery and replay attacks.

6 Security Vulnerability Analysis and Mitigation

Vulnerability	**Affected Method**	**Proposed Mitigation**
Token Theft	JWT, OAuth2	Short-lived tokens, refresh tokens, HTTPS enforcement
Replay Attacks	JWT, Session	Nonces, timestamp validation, blockchain signatures
Session Hijacking	Session	Secure cookies, SameSite flag, IP-based validation

Table 1: Security vulnerabilities and mitigations.

7 Implementation and Real-World Use Case

Implemented in a Spring Boot WebSocket chat application.

- Secure login via session-based authentication.
- API requests authenticated using JWT.
- OAuth2 used for third-party logins (Google, GitHub, etc.).
- Blockchain records token hashes to prevent forgery and unauthorized reuse.

8 Conclusion and Future Work

Our hybrid authentication model enhances security while maintaining performance and scalability. Future research will focus on AI-driven anomaly detection for API security and quantum-resistant cryptography for token security.

9 References

References

[1] OAuth2 Specification - `https://oauth.net/2/`

[2] JSON Web Tokens - `https://jwt.io/introduction/`

[3] OWASP API Security Guidelines - `https://owasp.org/www-project-api-security/`

[4] Zero Trust Security - https://csrc.nist.gov/publications/detail/sp/800-207/final