International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 04 April-2026 | Impact Factor: 3.5

Understanding Authentication in Microservices: Comparing Session-Based and JWT

Abhirudh Singh

Department of Computer Science Chitkara University Punjab,India

Abhirudh21singh@gmail.com

How to Cite this Article:

Singh, A. (2026). Understanding Authentication in Microservices: Comparing Session-Based and JWT. International Journal of Creative and Open Research in Engineering and Management, 02(04). https://doi.org/10.55041/ijcope.v2i4.871

https://doi.org/10.55041/ijcope.v2i4.871

Abstract—This paper compares traditional session-based authentication approach with modern JWT-based authentication for web application. This paper describes the implementation of JWT-based authentication and the security issues related to the storage of JWT. The performance experiment of JWT-based authentication compared to the traditional session-based approach showed promising results: a reduction of 50% in average response time, from 185ms to 92ms.

Keywords—JWT, Authentication, Web Security, MERN Stack, Stateless Architecture, HttpOnly Cookies.

I. INTRODUCTION

With the immense growth of web applications and distributed systems, authentication mechanisms have become critical components of modern software architecture. Traditional session-based authentication has been widely used owing to its simple functioning and reliability; however it introduces several limitations in terms of scalability, performance, and resource utilization in large-scale systems[7].

In contrast, modern software products, especially those that use microservice architecture and cloud infrastructure, need to have authentication services that can ensure high performance and scalability. JWT(JSON Web Token) has become one of the most popular solutions today. The protocol provides a stateless authentication mechanism that significantly reduces the server load[1].

Although both authentication strategies have proven their efficiency in many applications, some doubts may arise among developers who try to choose the better option. In many cases, the selection of an appropriate solution becomes difficult due to different trade-offs and limitations.

The purpose of this paper is to describe the advantages and disadvantages of session-based and JWT-based authentication and analyze their performance and security aspects based on existing standards and recommendations[3][4]. Moreover, the impact of modern authentication mechanisms and frameworks, such as OAuth-based ones, will be discussed as well[5][6].

After reading this paper, the audience will be able to understand the key features of both technologies and decide which solution best fits their application requirements.

II. BACKGROUND

A. Session-Based Authentication

In this method, when a user logs in, the server creates a session and stores the user's data either in memory or in a database. A unique session ID is then sent back to the client, usually through cookies, and used for every subsequent request.

The biggest limitation of this approach is that it is stateful, meaning the server has to remember every active user session. This becomes difficult to manage when applications scale across multiple servers, as session data must be shared using techniques like sticky sessions or centralized storage[7].

Another issue is memory usage. Since each logged-in user requires a session to be stored, the server's memory grows linearly with number of users, leading to O(n) complexity. This can create performance bottlenecks in high-traffic systems[1].

Moreover, session-based systems typically make frequent calls to the database for session verification purposes. Every call requires fetching information about the session, adding overhead to the process[2].

B. JWT-Based Authentication

JSON Web Tokens(JWT) offer an advanced way to overcome the limitations of the traditional method for handling user sessions. Unlike storing session information on the server, JWT sends encoded user data through tokens to the client. The implementation is much more scalable and efficient, especially in the case of distributed systems[1].

There are three parts of the JWT structure: Header, Payload, and Signature.

1. Header holds all the necessary information related to the token, such as its type(JWT) and signing algorithm(for example, HMAC SHA256 and RSA).

2. Payload includes the claim that describes some statements related to the user and data that can be encoded in addition to user identification. All claims are divided into three groups:

Registered Claims: standard values like iss(issuer), exp(expiration time), sub(subject), and aud(audience).

Public Claims: unique custom claims

Private Claims: unique custom between two entities

3. Signature makes sure the token is unaltered and has integrity. It is created by encoding the header and payload, then signing them using a secret key(for HMAC) or a private key (for RSA). This guarantees that the token has not been altered[1].

III. METHODOLOGY

This paper implements a comparative experiment methodology in analyzing authentication mechanisms within a microservices architecture. Specifically, session-based and JWT-based authentication are analyzed in terms of architecture, performance, scalability, and security.

A. System Architecture

To perform the comparative analysis, a prototype application was built utilizing a microservice architecture style. The prototype application contains React frontend and Node.js backend. It uses APIs exposed by the backend which require authentication.

Two methods were implemented independently for the comparison purposes:

Session-Based Authentication:

User session information is saved in the server. After logging in, a session id is created and stored using the cookie. This session id must be validated by each incoming request to the server.

JWT-Based Authentication:

Authentication is done via JSON Web Tokens(JWT). The server generates access and refresh token upon a successful login. The tokens are stored in HttpOnly cookies for further authorization.

B. Implementation Strategy

The implementation follows common practices regarding authentication mechanisms:
1. Security Credential:
Passwords are saved in the database in the hashes form using the bcrypt method with salt factor.
2. Token mechanism(JWT):
Access Token Duration - 15 mins
Refresh Token Duration – 7 days
Refresh Token rotation technique was added for enhanced security and prevent token reuse attacks.
3. Session Handling:
Session-based authentication uses server-side storage(memory store) to maintain user sessions, requiring validation on each request.

C. Setup of Evaluation Environment for Performance Test

In order to evaluate both approaches, the following performance testing environment was prepared with the help of Apache Jmeter tool:

Simulation of 1,000 concurrent users

Identical hardware and Node.js runtime environment

Repeated tests with averaging of results

Emphasis on protected routes(where authentication is needed)

D. Metrics for Comparison

Comparison between session and JWT based authentication is made by means of the following parameters:

Response Time – measuring time of authenticated requests processing.

Memory Consumption – assessing how much server memory is consumed depending on user growth.

Database I/O operations – calculating how many database queries are needed for authentication.

Scalability – evaluating possibility to distribute the load and provide load-balancing capabilities.

Security Issues – considering issues related to token storage security, cross-site scripting vulnerability, and security of tokens itself.

E. Analytical Approach

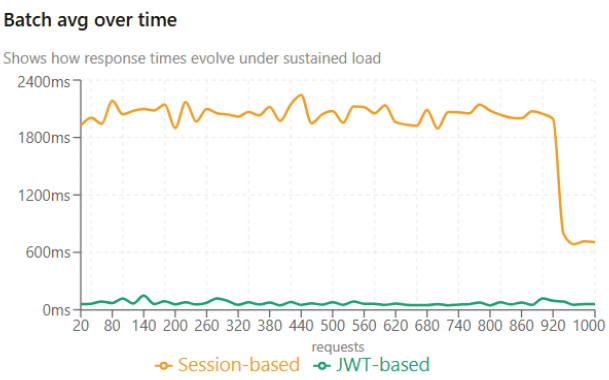
The present research uses combined approach – experimental one and architecture based analysis. Experimental data from performance testing will be used to determine the performance improvement.

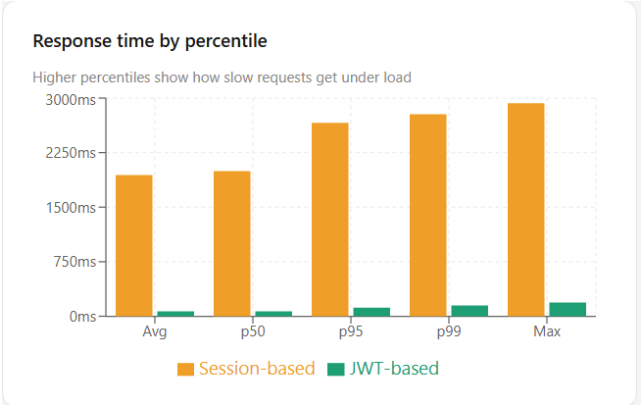
IV. RESULTS

Both session-based and JWT-based authentication have been performed in exactly the same circumstances using the Apache Jmeter framework with 1,000 concurrent requests.

JWT authentication had an average response time of 92ms against session authentication having an average response time of 185ms. JWT did not use any I/O operations from the database for request validation, meanwhile each user's session needed a single request for a look-up from the database. Memory space used by the server increased linearly (O(n)) for each user added in the session-based method. However, memory space remained constant (O(1)) for JWT.

JWT systems provide high horizontal scalability, while session-based systems needed sticky session or central storage. Below graphs can help you understand better.





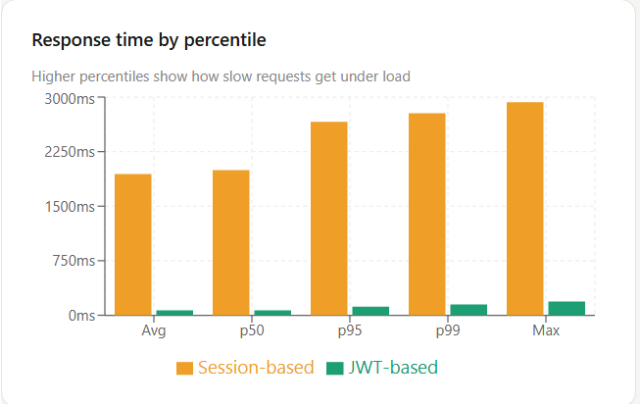


Table 1

Metric	**JWT-Based**	**Improvement**	**Session-Based**
Avg. Response Time	92 ms	~50% Measured	185 ms
Server Memory Usage	Constant $O(1)$	Significant Measured	$O(n)$ Users
Database I/O Hits	0 (Local Verify)	Eliminated Measured	1 per Request
Horizontal Scaling	Seamless	Simplified Measured	Sticky Sessions

V. Conclusion

This research shows the scalability and efficiency of JWT as an alternative to session based authentication schemes. The absence of sessions on server side allows improving response time, reducing the dependence on a database and enabling horizontal scaling.

The conducted experiment demonstrated better performance, in particular, the average response time was decreased twice compared to the session authentication scheme. Moreover, the absence of database requests at each authentication and constant memory consumption make the proposed method more efficient from the implementation point of view.

As far as security is concerned, the utilization of HttpOnly cookies and refresh token rotation allowed us to improve our security level preventing XSS and token reusing attacks.

In conclusion, JWT authentication proved to be effective and efficient authentication scheme for microservice. As for further research, multi-factor and biometric authentication systems can be implemented to increase the level of security.

VI. REFERENCES

[1] M. Jones, J. Bradley, and N. Sakimura,"JSON WEB TOKEN(JWT),"RFC 7519,May2015.

[2] N.Provos and D. Mazières,"A Future-Adaptable Password Scheme," USENIX Annual Technical Conference, 1999.

[3] OWASP Foundation, "Session Management Cheat Sheet," 2023.

[4] OWASP Foundation,"Cross Site Scripting Prevention Cheat Sheet,"2023.

[5] D. Hart, "OAuth 2.0 Security Best Current Practice," IETF Draft,2012.

[6] T. Lodderstedt et al.," OAuth 2.0 Security Best Current Practice," IETF Draft, 2023.

[7] C. Richardson,"Microservices Patterns:With Examples in Java,"Manning Publications,2018.