Performance Analysis of API Key vs. Basic Authentication in Intra-Network Spring Boot Microservices

Azem Kakitaeva * and Mekia Shigute Gaso

Posted Date: 12 May 2025

doi: 10.20944/preprints202505.0825.v1

Keywords: microservices; spring boot; API KEY authentication; basic authentication; bcrypt; intranetwork; performance; security

 doi:10.20944/preprints202505.0825.v1

Article

Performance Analysis of API Key vs. Basic Authentication in Intra-Network Spring Boot Microservices

Azem Kakitaeva [1,*] and Mekia Shigute Gaso [2]

[1] Student, Department of Computer Science, Ala-Too International University
[2] Senior Lecturer, Department of Computer Science, Ala-Too International University
* Correspondence: azem.kakitaeva@alatoo.edu.kg

Abstract: Microservice architecture has become a prevalent paradigm in modern software development, emphasizing modularity, scalability, and efficient service communication. Communication between microservices that is secure and of high performance is of paramount importance. This study evaluates two commonly used authentication methods such as API key authentication and basic authentication using bcrypt hashing within a trusted intra-network environment. The primary objective of this study is to compare their performance characteristics, specifically response time and throughput in order to determine which is more suitable for high-load scenarios. Two Spring Boot microservices were developed and tested using Apache JMeter under varying load conditions. The results indicate that API Key Authentication outperforms Basic Authentication, exhibiting reduced latency and increased throughput. However, due to the limitations of the testing setup, the findings should be considered approximate. This study was conducted as part of the "Basics of Scientific Research Methods" course, demonstrating foundational research skills rather than offering definitive conclusions.

Keywords: microservices; spring boot; API KEY authentication; basic authentication; bcrypt; intra-network; performance; security

Classifier: UDC 004.056

---

1. Introduction and Literature Review

The microservices architecture has become a cornerstone in modern software development for building distributed, maintainable, and scalable applications [1]. With this evolution, securing communication between services while maintaining performance has become a central concern. Authentication mechanisms play a critical role in ensuring secure access, yet their computational requirements can impact overall system performance. Two common authentication schemes: API Key Authentication [2,3] and Basic Authentication with bcrypt [4] hashing offer different security and performance trade-offs. Additionally, [5] this research on data security in Spring Boot REST APIs underscores the significance of leveraging security features like bcrypt for password encryption and Spring's built-in authentication mechanisms. Basic authentication is a simple one. However, it should be configured with proper encryption. Bcrypt offers better security, but at a cost of computational overhead. API Key Authentication, utilizing personal tokens, guarantees greater performance and scalability. However, it also comes with its own security concerns.Previous research [6] has explored the effectiveness of web service authentication; however, comparisons specifically focused on Spring Boot microservices within a single network remain limited. The hypothesis of this paper is that API Key Authentication is more efficient than Basic Authentication with bcrypt hashing in high-load intra-network microservices environments. This is due to its lower computational overhead, which is critical in scenarios where performance and scalability are paramount.

 doi:10.20944/preprints202505.0825.v1

2. Methodology

This study uses a quantitative approach to compare the performance of API Key and Basic Authentication in a controlled environment. Two microservices with different authentication techniques were implemented using Spring Boot [7] framework and Apache JMeter [8] was used to simulate varying loads and measure response times and throughput.

2.1. Experimental Setup

The experiment was conducted within single private network. The experiment was designed to measure the response time and throughput under different loads and implemented using Spring Security functions [9]. Scalability was measured by incrementally adding client threads in JMeter.

2.2. Configuration Details

1. Basic Authentication with bcrypt: Implemented using Spring Security's httpBasic() method, combined with BCryptPasswordEncoder for secure password hashing during verification.
2. API Key Authentication: Implemented through a custom filter (ApiKeyAuthenticationFilter) that intercepted HTTP requests and validated a token passed in the X-API-KEY header. Requests without a valid key received a 401 Unauthorized response.

2.2.1. Basic Authentication with Bcrypt

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    private UserDetailsService userDetailsService;

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .httpBasic();
    }

    @Autowired
    public void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

}
```

2.2.2. API Key Authentication

```
@Configuration
@EnableWebSecurity
public class ApiKeySecurityConfig extends WebSecurityConfigurerAdapter {
```

 doi:10.20944/preprints202505.0825.v1


```
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .authorizeRequests()
            .anyRequest().authenticated()
            .and()
            .addFilterBefore(new ApiKeyAuthenticationFilter(),
                UsernamePasswordAuthenticationFilter.class);
    }
}

@Component
public class ApiKeyAuthenticationFilter extends OncePerRequestFilter {

    private static final String API_KEY_HEADER = "X-API-KEY";
    private static final String API_KEY = "your-api-key";

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain)
            throws ServletException, IOException {

        String apiKey = request.getHeader(API_KEY_HEADER);

        if (apiKey == null || !apiKey.equals(API_KEY)) {
            response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
            response.getWriter().write("Unauthorized: Invalid or missing API Key");
            return;
        }

        filterChain.doFilter(request, response);
    }
}
```

3. Results

The results, summarized in Table 1, demonstrate that API Key Authentication significantly outperformed Basic Authentication with bcrypt in terms of response time and throughput.

Table 1.

Method	Response (ms)	Throughput (req/sec)
Basic Authentication (bcrypt)	150	120
API Key Authentication	80	300
Difference	-70	180
Ratio (API Key/Basic)	0.53 (47% reduction)	2.5 (150% increase)

4. Discussion

The performance disparity between Basic Authentication with bcrypt and API Key Authentication can be explained by the computational demands of each method. Bcrypt is a hashing algorithm, which performs multiple rounds of hashing to slow down brute-force attacks. Thereby, it leads to increased

 doi:10.20944/preprints202505.0825.v1

authentication time—especially under high-load conditions. While this makes it more secure, it can noticeably degrade performance in applications handling many concurrent authentication requests. In contrast, API Key Authentication simply checks a string value in the request header, requiring far less CPU and memory. This efficiency results in faster response times and higher throughput. It makes API Key Authentication scheme better suited for internal services where performance is critical. However, though it has performance benefits, it is important to remember to use the API Key Authentication by following its best practices such as secure storage, key rotation, limit scope and rate limiting to ensure security and minimize possible attacks.

5. Conclusions

In this study, performances of two common authentication methods such as API Key Authentication and Basic Authentication using bcrypt in intra-network Spring Boot microservices [10] were explored. This study outlines that API Key Authentication is a better choice for such scenario, particularly for high-load services communication. API Key Authentication offers faster response time and higher throughput due to its lower computational demands compared to the bcrypt hashing method. However, it comes with some potential security concerns. Under improper management, it can impose greater security risks, particularly if keys are not well managed, rotated, or stored securely. This study was conducted as part of an academic course on research methods and is intended for educational purposes. The findings are approximate and should not be considered exhaustive or conclusive. Future research could expand on these results by incorporating a broader range of authentication methods, real-world deployment scenarios, and more rigorous performance benchmarking.

6. References

1. Nadareishvili, I., Mitra, R., McLarty, M., & Amundsen, M. (2016). *Microservice architecture: Aligning principles, practices, and culture.* O'Reilly Media. https://www.oreilly.com/library/view/microservice-architecture/9781491956274/
2. LinkedIn. (n.d.). *What are the best practices for securing and rotating API keys and secrets?* LinkedIn. https://www.linkedin.com/advice/3/what-best-practices-securing-rotating-api-keys-secrets
3. Baeldung. (n.d.). *Spring Boot API key and secret authentication.* Baeldung. https://www.baeldung.com/spring-boot-api-key-secret
4. Auth0. (n.d.). *Hashing in action: Understanding bcrypt.* Auth0. https://auth0.com/blog/hashing-in-action-understanding-bcrypt/
5. Batyrbekova, S., & Esenalieva, G. (2025). *Research on data security approaches in Spring Boot for REST API. Preprints*. https://www.preprints.org/manuscript/202501.0908/v1
6. Stack Overflow. (n.d.). *How to secure REST APIs in Spring Boot web application.* Stack Overflow. https://stackoverflow.com/questions/42870489/how-to-secure-rest-apis-in-spring-boot-web-application
7. Spring. (n.d.-b). *Microservices.* Spring.io. https://spring.io/microservices
8. GeeksforGeeks. (n.d.). *Apache JMeter: An introduction.* GeeksforGeeks. https://www.geeksforgeeks.org/apache-jmeter-an-introduction/
9. Spring. (n.d.-a). *Spring Security.* Spring.io. https://spring.io/projects/spring-security
10. Raj, R. V. (2016). *Spring microservices.* Packt Publishing. https://books.google.kg/books?id=pwNwDQAAQBAJ