UDC 004.42

Methods of web application authorization

A. Dudak✉

Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russia

The article presents a comparative analysis of popular authorization methods in web applications: JSON Web Token (JWT), OAuth2, and Keycloak. It emphasizes that authorization is a critical process in access management, determining what a user can do after successful authentication. The characteristics of each method are examined in detail. A comparison based on several criteria – security, performance, scalability, flexibility, ease of integration, and management – demonstrates that each approach has unique advantages and disadvantages that define its applicability in different scenarios. The study supports the hypothesis that the choice of an authentication method should be based on the project's specifics, its scale, security requirements, and infrastructure capabilities.

Keywords: authorization, JWT, OAuth2, Keycloak, security, scalability.

Методы авторизации веб-приложений

А.А. Дудак✉

Томский государственный университет систем управления и радиоэлектроники, Томск, Россия

В статье проводится сравнительный анализ популярных методов авторизации в веб-приложениях: JSON Web Token (JWT), OAuth2 и Keycloak. Подчеркивается, что авторизация является важным процессом в управлении доступом к ресурсам, определяющим, что пользователь может делать после успешной аутентификации. Рассматриваются особенности каждого из методов. Сравнение по нескольким критериям – безопасность, производительность, масштабируемость, гибкость, простота интеграции и управление – позволяет сделать вывод, что каждый подход обладает уникальными преимуществами и недостатками, которые определяют их применимость в различных сценариях. Исследование обосновывает предположение о том, что выбор метода аутентификации должен основываться на специфике проекта, его масштабе, требованиях к безопасности и возможностях инфраструктуры.

Ключевые слова: авторизация, JWT, OAuth2, Keycloak, безопасность, масштабируемость.

Introduction

Modern web applications are characterized by security, reliability, and scalability of authorization mechanisms. In a context of increasing year-over-year digital service users, demands to raise performance coupled with data protection start to turn up as foreground for distributed systems. Token-based authorization models, such as JSON Web Token (JWT) and OAuth2, as well as more comprehensive solutions like Keycloak, are becoming increasingly relevant.

The development of Web technologies has created an ever-growing demand for unified and secure access control mechanisms able to provide user authentication and delegation of rights without heavy loads on the server infrastructure. On the other hand, the choice of a specific authorization method is related to a trade-off among security, performance, and easiness of integration. For instance, JWT allows for stateless authorization but involves a number of vulnerabilities regarding token storage security. OAuth2 is widely used in systems

with third-party service integrations, though its implementation requires detailed configuration and strict adherence to security protocols. Keycloak, in turn, is a strong means of centralized access management but is highly complex in its implementation.

This work is aimed at comparative analysis, strengths, and weaknesses of the listed authorization methods in order to outline optimal application scenarios. The study will allow a deeper understanding of how the mechanisms of modern authentication technologies function and influence the security and performance of web applications.

Main part. Theoretical foundations of authorization in web applications

Authorization is the most important process in access management to web application resources, defining what a user can do after successful authentication. Although the terms are used very frequently and almost interchangeably, one should understand the difference between authorization and authentication. Authentication confirms the identity of the user, while authorization regulates what resources or operations are available to the user. It is impossible to develop modern web applications without developing an authorization mechanism in line with security, performance, and scalability requirements, especially given the increasing pace of distributed systems and cloud technologies.

Among the most critical requirements of authorization is to ensure data security. The modern approaches include the usage of cryptographic methods that protect tokens and minimize the storage of confidential information on the client side. For instance, digital signature algorithms (RSA or HMAC) are widely used to confirm data integrity and authenticity during token transfer. Statistical evidence shows that a security breach of authorization systems, namely, due to improperly stored or intercepted transmission of tokens is among the most prevalent in web applications (fig.) [1].

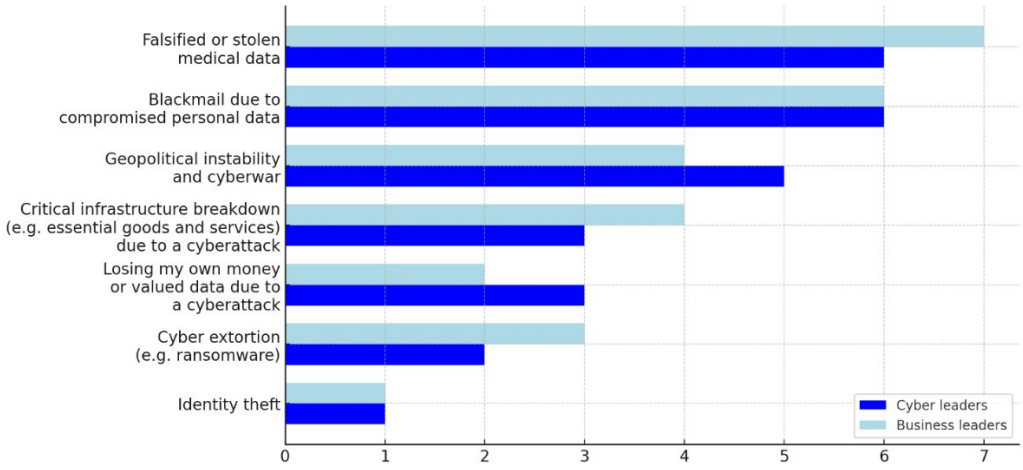

Figure. Cyber-risk concerns between business leaders and cyber leaders

In addition, authorization must ensure scalability of the system. As the number of users increases, traditional session management mechanisms based on server memory (e.g., server session storage) become a bottleneck. An alternative is stateless authorization mechanisms, such as JWT, which allow the server not to store information about the session state. This is especially important for distributed systems, where each server can process requests independently [2].

Another important aspect is the ease of integration and flexibility of solutions, as web applications often interact with multiple third-party systems, such as API or microservices. In

such scenarios, support for modern protocols is important, including OAuth2 and OpenID Connect. These protocols provide standardized authorization and authentication mechanisms, which simplifies the implementation of single sign-on (SSO) and delegation of rights.

Particular attention should be paid to protection against vulnerabilities such as replay attacks and token compromise. To prevent such threats, timestamps, unique token identifiers (nonces), and token expiration dates are implemented. It is also recommended to use transport encryption (e.g., HTTPS), which reduces the likelihood of data interception.

Thus, authorization in web applications is an integral component of modern IT infrastructure. It combines data protection mechanisms, scalability and ease of integration, which is especially important in the context of the global spread of web technologies. Innovation-oriented companies are actively implementing modern authorization methods, such as JWT and OAuth2, to ensure the sustainability and competitiveness of their systems. However, despite the advantages, the implementation of these approaches requires strict adherence to security protocols and adaptation to specific operating conditions.

Overview of authorization methods

Web application authorization is based on various mechanisms, the most common of which are JWT, OAuth2, and centralized access management systems such as Keycloak. These technologies have their own implementation features that affect their use in different scenarios, including distributed systems, third-party service integration, and data security.

Is JWT an open standard for transferring data between parties in the form of tokens signed using the HMAC or RSA algorithms. JWT stands out for its lightweight nature and independence from server state (stateless), which allows scaling applications without the need to store session data on the server. The basic structure of the token consists of three parts: a header, a payload, and a signature, combined into a string and encoded in Base64 format. This makes JWT convenient for use in distributed systems where servers process requests in parallel. However, JWT vulnerabilities include the risk of token compromise when stored on the client side. For example, when using Local Storage in a browser, the likelihood of data leakage through XSS (cross-site scripting) attacks increases. According to the OWASP report, about 14% of attacks on web applications are related to token security flaws [3].

One of authorization protocols is OAuth2, which is being hugely used today since it allows access by other third-party applications without necessarily sharing user credentials. It comes with major flexibility in the form of configurations for various access levels through tokens; support for an authorization flow like an authorization code for web applications or the implicit one targeting client applications-strict consideration on security protocols. As one example, the obligatory use of HTTPS reduces the possibility of token intercept. Even in these cases, implementation errors-for example, badly configured redirect URI-can lead to data disclosure. In the USA, OAuth2 protocol is utilized for integrations-for example, from the Stripe platform to third-party applications-in cases when access to user data is granted without having to share API keys [4].

Yet one of the known weaknesses within OAuth2 is poor token check-up while integrating the systems, and attackers use those to perform a replay attack. A signature verification feature of the token should be part of it while employing a private key to develop minimal risks for these kinds.

Keycloak is an all-in-one authorization and authentication manager that comes with OAuth2 and OpenID Connect support. Among its most significant advantages is the fact that it can manage users centrally through interfaces and API. Keycloak is particularly handy for organizations that have to deal with hundreds of users who require different levels of access. Furthermore, the system allows for SSO mechanisms that minimize the headache for users

while accessing various applications. The complexity of Keycloak is in the implementation and further maintenance of high-load systems. According to the 2023 report from Gartner, the average time required to set up Keycloak reaches up to 120 man-hours, together with integration with existing systems [5].

It is interesting to note here that Keycloak performs access management for internal corporate systems across large US corporations like IBM. Keycloak adds some extra advantage-integration with Active Directory, multi-factor authentication-thus, making it preferable for large enterprise customers.

Therefore, JWT, OAuth2, and Keycloak offer a wide range of options for the developer regarding authorization according to project needs-from JWT, which is suitable for scalable and distributed systems; OAuth2 for the integration of third-party applications; to Keycloak, offering more flexibility in the form of centralized management. At the same time, all these technologies require a thoughtful configuration and strict adherence to security protocols to reduce vulnerability and increase system resilience.

Comparative analysis of methods

Modern web application authorization methods such as JWT, OAuth2, and Keycloak provide different approaches to access control. JWT provides a high level of security when implemented correctly, but is vulnerable when storing tokens in Local Storage, which can lead to XSS attacks. OAuth2 offers a more sophisticated security protocol with the ability to use multi-factor authentication, while Keycloak integrates OAuth2 and OpenID Connect, offering additional layers of security such as SSO and centralized access rights management (tabl.).

Table

Comparative analysis of authorization methods

Criterion	JWT	OAuth2	Keycloak
Security	High when implemented correctly, vulnerable to XSS	Supports HTTPS, MFA	SSO, integration with OpenID Connect
Performance	High (stateless)	Medium (delays in token validation)	Low under high load
Scalability	Excellent for distributed systems	Good (can become complex with many apps)	Good with clustering support
Flexibility	Limited (fixed token payload)	High (supports different flows)	Very high (extensive configuration)
Ease of integration	Simple, good documentation	Medium (requires precise configuration)	Complex (server and API setup required)
User management	No built-in management	Managed via external systems	Centralized management, SSO

Thanks to its principle of stateless authorization, JWT demonstrates high performance, making it particularly suitable for high-load systems. OAuth2 requires additional resources to work with refresh tokens, and Keycloak creates a load on the server, especially when integrating with external systems [6].

For scalable distributed systems, JWT is suitable due to its stateless architecture. OAuth2 and Keycloak also demonstrate good scalability, but Keycloak may suffer from performance degradation under high load without caching settings [7]. OAuth2 is more flexible due to its support for various authorization flows, and Keycloak offers additional customization options for organizations with high access control requirements.

Due to its ease of integration, JWT is an ideal choice for developers working with RESTful API. OAuth2 is more complex to set up, especially when using refresh tokens or implementing multi-factor authentication. Keycloak requires significant effort during the implementation phase, including setting up servers, configuring access rights, and training employees [8].

Applications with high performance and scalability requirements can benefit from using JWT, though it has limitations in flexibility and potential security risks. OAuth2 demonstrates versatility and reliability, but the complexity of implementation makes it less accessible for small projects. Keycloak, although difficult to configure, is a powerful tool for large organizations with distributed systems and strict access control requirements. Each of the solutions has its own optimal use cases and depends on the needs and capabilities of a particular project.

Conclusion

A comparative analysis of the JWT, OAuth2, and Keycloak authentication methods showed that each of them has unique strengths and weaknesses that determine their applicability in different scenarios. JWT is an excellent choice for high-load systems that require scalability and performance, but its security depends on proper configuration and protection of tokens. OAuth2 provides high flexibility and security, especially in integration with external services, but requires complex configuration and protocol compliance. Keycloak, despite its complexity, offers powerful centralized access management and support for various security mechanisms such as SSO and MFA, making it ideal for large organizations. Ultimately, the choice of authentication method should be based on the specifics of the project, its scale, security requirements, and infrastructure capabilities, which determines its effectiveness and long-term stability.

REFERENCES

1. Global Cybersecurity Outlook 2023: Insight Report [Electronic resource]. – URL: https://www3.weforum.org/docs/WEF_Global_Security_Outlook_Report_2023.pdf [Accessed 9th January 2025].

2. Перспективы развития информационной безопасности: глобальные вызовы и стратегии защиты / А. Яковишин, И. Кузнецов, И. Дроздов [и др.] // Информационные ресурсы России. – 2024. – № 2 (197). – С. 93–103.

3. OWASP Top Ten [Electronic resource] // OWASP® Foundation. – URL: https://owasp.org/www-project-top-ten/ [Accessed 9th January 2025].

4. OAuth 2.0 [Electronic resource] // Stripe Documentation. – URL: https://docs.stripe.com/stripe-apps/api-authentication/oauth [Accessed 10th January 2025].

5. Gartner Magic Quadrant for Access Management [Electronic resource] // Gartner. – URL: https://www.gartner.com/en/documents/4936631 [Accessed 10th January 2025].

6. Advanced Security Mechanisms in the Spring Framework: JWT, OAuth, LDAP and Keycloak / N. Dimitrijević, N. Zdravković, M. Bogdanović [et al.] // BISEC'23: 14th International Conference on Business Information Security, November 24, 2023, Niš, Serbia: CEUR Workshop Proceedings. – 2024. – P. 64–70.

7. Norimatsu T. Policy-Based Method for Applying OAuth 2.0-Based Security Profiles / T. Norimatsu, Yu. Nakamura, T. Yamauchi // IEICE Transactions on Information and Systems. – 2023. – Vol. E106.D, No. 9. – P. 1364–1379.

8. Sidorov D. Cross-browser compatibility issues and solutions in web development / D. Sidorov // ISJ Theoretical & Applied Science. – 2024. – Vol. 11, No. 139. – P. 18–21.

INFORMATION ABOUT THE AUTHOR

Dudak Aleksei, specialist degree, Tomsk State University of Control Systems and Radioelectronics, Tomsk, Russia.
e-mail: aleksei.dudak@rambler.ru