https://doi.org/10.31891/2219-9365-2025-81-36
UDC 004

PYROH Mykola
Taras Shevchenko National University of Kyiv
https://orcid.org/0000-0003-2588-6066
TERESHCHUK Ganna
Taras Shevchenko National University of Kyiv
https://orcid.org/0000-0001-7573-9748
TOROSHANKO Oleksandr
Taras Shevchenko National University of Kyiv
https://orcid.org/0000-0002-2354-0187

AUTHENTICATION PRINCIPLES AS SECURITY ASPECTS OF WEB DEVELOPMENT

This article examines modern authentication and authorization methods, emphasizing their critical role in web systems security. It estimates traditional approaches, such as Basic Authentication, and compares them to more advanced techniques like OAuth and JSON Web Tokens (JWT). The authors focus on the growing complexity of web systems, necessitating stronger and more flexible security mechanisms.

Particular attention is paid to the weaknesses of password-based systems and the advantages of token-based solutions, with a strong emphasis on JWT as the most appropriate choice for the majority of current web development. The compact and self-contained nature of JWT, combined with its scalability and security, makes it a preferred method for both authentication and authorization in diverse web environments.

The article also highlights the risks associated with token management, such as interception and misuse, while offering practical strategies for mitigating these threats. The authors states that, for most modern web systems, JWT provides a more efficient and secure alternative to traditional methods, balancing security needs with performance and convenience in use.

In conclusion, the study recommends the adoption of JWT as a flexible and scalable solution that meets the security demands of evolving web systems. Practical recommendations are provided for the secure implementation of JWT to ensure the protection of user data and enhance overall cybersecurity.

Keywords: Authentication, Authorization, Token, Web development, Cybersecurity.

ПИРОГ Микола, ТЕРЕЩУК Ганна, ТОРОШАНКО Олександр
Київський національний університет імені Тараса Шевченка

ПРИНЦИПИ АВТЕНТИФІКАЦІЇ ЯК АСПЕКТИ БЕЗПЕКИ ВЕБ-РОЗРОБКИ

У цій статті розглядаються сучасні методи автентифікації та авторизації, підкреслюючи їх критичну роль у безпеці вебсистем. Оцінються традиційні підходи, такі як базова автентифікація, і порівнюються із більш просунутими методами, такими як OAuth і вебтокени JSON (JWT). Автори зосереджуються на зростаючій складності вебсистем, що вимагає більш потужних і гнучких механізмів безпеки.

Особливу увагу приділено слабким сторонам систем на основі паролів і перевагам рішень на основі токенів із сильним наголосом на JWT як найбільш прийнятному виборі для більшості сучасних веброзробок. Компактність і самодостатність JWT у поєднанні з його масштабованістю та безпекою робить його кращим методом як для автентифікації, так і для авторизації в різноманітних веб-середовищах.

У статті також висвітлюються ризики, пов'язані з керуванням токенами, такі як перехоплення та зловживання, а також пропонуються практичні стратегії пом'якшення цих загроз. Автори стверджують, що для більшості сучасних веб-систем JWT забезпечує більш ефективну та безпечну альтернативу традиційним методам, збалансовуючи потреби безпеки з продуктивністю та зручністю у використанні.

На завершення дослідження рекомендує прийняти JWT як гнучке та масштабоване рішення, яке відповідає вимогам безпеки веб-систем, що розвиваються. Надаються практичні рекомендації щодо безпечного впровадження JWT для забезпечення захисту даних користувачів і підвищення загальної кібербезпеки.

Ключові слова: автентифікація, авторизація, токен, веброзробка, кібербезпека.

STATEMENT OF THE PROBLEM IN A GENERAL VIEW AND ITS RELATIONSHIP WITH IMPORTANT SCIENTIFIC OR PRACTICAL TASKS

Authentication takes a key role in securing modern web applications and web systems by verifying user identity and controlling access to resources [10]. In the context of web development, this process aims to confirm that users are who they say they are by using a set of verification credentials such as a username and password, or through more modern mechanisms based on virtual or physical tokens. The complexity of authentication processes has increased significantly with the development of web-based systems, as remote access, mobile platforms, and various integrations with third-party services have opened up more opportunities for data compromise.

The growing need to verify the identity of the person accessing web resources is a result of the steady increase in the number of online transactions, the storage of confidential information, including financial data, and the overall digitalization of business and public services. Authentication technologies must ensure not only accuracy and convenience for users, but also a high level of security. In particular, for critical services (such as banking or healthcare), ensuring the security of personal data and financial transactions is of paramount importance, as access to confidential resources is the main goal of attackers.

With the advancement of web technologies, the need to adapt authentication mechanisms to meet the challenges faced by modern systems has become essential. Traditional methods, such as password-based authentication, no longer provide adequate security, especially given the growing number of users, integration with third-party services, and the need for continuous system operation. The vulnerabilities of passwords, such as susceptibility to brute-force and phishing attacks, pose significant risks to entire web systems, prompting the development of modern approaches to authentication.

The main objective of this study is to enhance the security of web systems by implementing modern authentication methods, particularly JSON Web Tokens (JWT), which aim to balance security, usability, and scalability. A major challenge is achieving this balance, especially in the case of multi-factor authentication (MFA), which, while improving security, often requires additional steps from users, reducing convenience. Many users and organizations tend to prefer simpler, though less secure methods, which increases security risks.

Integrating web applications with external services, such as OAuth or APIs, also introduces new security demands. Without proper control over the storage and transmission of tokens, attackers can gain access to sensitive data or resources, creating additional threats. Tokens are frequently used for data exchange between services, making it critical to protect them from interception and modification.

Overall, the study aims to provide comprehensive security for the authentication process in the face of growing threats without compromising system usability and performance. This approach is unique in its comparison of traditional authentication methods with modern solutions like JWT, showcasing the benefits of token-based approaches and offering recommendations for enhancing token security in real-world scenarios where integration and performance demands are high.

ANALYSIS OF LATEST RESEARCH AND PUBLICATIONS

In the context of the study, we can note a wide range of scientific developments related to the issues of ensuring the cybersecurity of web systems. Web services, as the foundation of modern technologies [8], cannot ensure the security of user data without proper authentication and authorization, so scientists and practitioners are actively researching these issues, offering new approaches and solutions.

First of all, the literature focuses on traditional authentication methods [12], such as passwords, which have historically been the most common method of data protection. However, numerous studies show that this approach can no longer meet modern security requirements. Passwords are vulnerable to brute-force, phishing, and stealing attacks, which has been confirmed by a number of scientific papers. Accordingly, the authors of the article emphasize the need to move to more reliable methods, such as tokens and multi-factor authentication.

The latest technologies are analyzed in detail, in particular, the OAuth [3, 6, 10, 11] and JWT [2, 3, 4, 11, 14] protocols (JSON Web Tokens). The literature confirms the effectiveness of these approaches in ensuring the security of web systems. OAuth allows users to provide third-party applications with access to resources without transferring credentials, which significantly reduces the risk of password theft. JWT, in its turn, provides a compact format for transferring tokens, which simplifies integration with modern web and mobile platforms. At the same time, the authors emphasize the need to adhere to strict rules regarding the storage and exchange of tokens to avoid their interception or reuse, which is mentioned in several papers on token attacks and security methods.

The importance of centralized authentication through a single sign-on (SSO) mechanism is also mentioned. This approach, supported by the SAML (Security Assertion Markup Language) and OpenID Connect protocols [8, 9], significantly increases the convenience for users, as they do not need to enter credentials on each service. However, studies show significant risks associated with the compromise of a single account, which can endanger all related systems. Accordingly, researchers emphasize the importance of protecting such systems through multi-factor authentication and data encryption.

Cybersecurity research also addresses issues related to the integration of external services through APIs and OAuth [6, 10, 13]. The lack of proper control over data transmission and token use can lead to information leaks, which is confirmed by numerous cases of session attacks. Some researchers suggest the introduction of additional measures to protect tokens, such as regular key rotation and token expiration.

The study also touches on physical [7] authentication methods. In modern research, these approaches are considered to be among the most reliable, as they are based on the user's physical characteristics, such as fingerprints or face recognition.

BASIC APPROACHES TO AUTHENTICATION

In web development, there are several approaches to authentication, each of which has evolved along with the development of technology. Each method has its own advantages and disadvantages, and their use depends on security requirements, usability, and integration features (Table 1). Let's take a look at these approaches from the oldest to the newest.

Basic Authentication is one of the simplest authentication methods based on the transmission of a username and password via HTTP headers. This data is encoded using Base64, but is not encrypted. Therefore, without the use of the secure HTTPS protocol, the data can be easily intercepted, making this method unsuitable for most modern web applications that require a higher level of security.

The advantage of this approach is its simplicity and support for the HTTP standard, which makes it easy to integrate into simple systems or APIs. However, the main disadvantage is the vulnerability to attacks in the absence of HTTPS.

Form-Based Authentication is a more traditional method where the user enters their data (username and password) into a web form that sends it to the server for verification. If the authentication is successful, the user receives a session or token for further interaction with the server.

This approach is easy to implement but has its risks. Using cookies to store a session ID can be vulnerable to CSRF and XSS attacks. In addition, storing data on the server can increase the load with a large number of users.

Session-Based Authentication. This approach is used to create a session on the server after successful user authentication. The server generates a unique session ID that is stored on the client via cookies and sent with each request. This identifier allows the server to track the user's status.

The main advantage is the support for saving the user's state between requests. However, the main disadvantage is the need to store session data on the server, which can cause scalability issues. In addition, sessions can be vulnerable to CSRF attacks.

API Keys are unique keys used for authentication when working with APIs. This method is simple and effective for controlling access to limited resources. The client transmits the key with each request, which allows the server to identify it.

The main advantage is ease of use and implementation. However, API Keys can be easily intercepted if used without HTTPS, and the lack of a mechanism for managing key expiration increases the risk of compromise.

Security Assertion Markup Language (SAML) is an XML-based standard for exchanging data between identity providers and services. It provides centralized authentication and is used to support a single sign-on (SSO) mechanism that allows users to log in to multiple applications using the same credentials.

This method is effective for enterprise systems, but its implementation is quite complex due to the use of XML and the voluminous data that is transmitted during authentication.

OAuth is an authorization protocol that is often used to provide access to third-party services without passing passwords. It allows third-party applications to gain limited access to resources through special tokens.

The main advantage of OAuth is its flexibility and support for scalable systems. However, it requires complex token management logic and, if not properly secured, can be vulnerable to token hijacking attacks.

JSON Web Tokens (JWT) is a compact token in JSON format that contains signed user information. The token is transferred to the client after authentication and is included in every subsequent request to the server.

The main advantage of JWT is that it does not require storing session data on the server, which makes this approach scalable. However, the tokens must be securely protected via HTTPS, otherwise they can be intercepted or reused.

Multi-Factor Authentication (MFA) enhances security by using multiple levels of authentication, such as a password and a one-time code via SMS or mobile app. This reduces the likelihood of credentials being compromised, even if the password is known to an attacker.

The main advantage is a significant increase in security. However, additional steps can reduce the usability of the system.

Biometric Authentication uses physical characteristics of the user, such as fingerprints or face recognition. Modern browsers support biometric authentication through the WebAuthn API, which allows you to integrate this method into web applications.

This approach is one of the most secure, but it requires the availability of appropriate equipment and can sometimes give false positives.

Single Sign-On (SSO) allows a user to log in to multiple applications using the same credentials. It often integrates with OAuth or OpenID Connect to provide a single sign-on across platforms.

The main advantage is the convenience for users, as they can use the same credentials across different services. However, a compromise of one account can put all connected systems at risk.

Table 1.

Authentication methods in web development

Authentication methods	Main characteristics	Advantages	Disadvantages	Usage
Basic Authentication	Uses HTTP headers to transmit the username and password. The data is encoded in Base64	Easy to implement. HTTP standard	Vulnerability without HTTPS. Easy to intercept without encryption	Easy access to APIs or internal systems
Form-Based Authentication	The user enters data on a web form, which is transmitted to the server for verification. After authentication, a session or token is created	Easy to implement. Supported by all web applications	Vulnerability to XSS and CSRF attacks. Dependence on cookies or tokens	Web applications that don't use complex authentication mechanisms
Session-Based Authentication	After authentication, the server creates a session with a unique identifier that is transmitted via cookies.	Supported by browsers. Convenient status management	Server load due to session storage. Vulnerability to CSRF	Older web applications and services with cookies
Token-Based Authentication	It uses tokens (for example, JWT) to transfer data between the client and the server without saving sessions on the server.	Stateless authentication. Flexibility in data transfer	Need to protect tokens. Vulnerability without HTTPS	Scalable web applications, API services
JSON Web Token (JWT)	A JSON token containing signed user information is transmitted with each request.	Easy to transfer. No need to store sessions	Tokens can be stolen or reused	Scalable web applications, mobile applications, APIs
OAuth	An authorization protocol that allows third-party applications to access resources without passing passwords.	Secure integration with third-party services. Scalability	Complexity of implementation. Vulnerability to token hijacking attacks	Integration with third-party resources
Security Assertion Markup Language (SAML)	Uses XML to transfer authentication data between the identity provider and the service.	Single sign-on (SSO). Centralized authentication	Complex implementation. Large amount of transmitted data	Corporate systems, SSO for large companies
Multi-Factor Authentication (MFA)	Requires several levels of verification, for example, a password and a one-time code via SMS or an app.	Increased level of security. Minimize the risk of compromise	Lower user experience. Implementation and support costs	Highly secure systems, financial services
Biometric Authentication	Use of biometric data (fingerprints, face recognition) for login.	High security. Convenience for users	Required hardware support. Possibility of authentication errors	Mobile applications, modern browsers via WebAuthn
Single Sign-On (SSO)	Single sign-on to multiple sites or applications using the same credentials.	Simplifies access to multiple services. Convenience for the user	If credentials are compromised, all systems are at risk	Large companies, integration with OAuth or OpenID Connect

THE IMPORTANCE OF SECURITY IN MODERN WEB SYSTEMS

In today's environment, the number of web systems and the amount of confidential data they process continues to grow. The security of web systems is becoming increasingly important due to the growing complexity of attacks and the sophistication of cybercriminals. One of the main aspects of protection is to minimize the risks of compromising authentication mechanisms, which can lead to massive data leaks, identity theft, financial losses, and even damage to companies' reputations.

In particular, improper implementation of authentication can allow attackers to carry out:

Intercept sessions or tokens (Man-in-the-Middle attacks) when data is transmitted over unsecured network channels.

Attacks on sessions using CSRF, XSS, or Clickjacking vulnerabilities that can allow session ID theft.

Attacks on user databases where accounts and passwords are stored. The use of weak hashing methods or the lack of password protection can lead to massive data leaks.

Ensuring the security of web systems includes not only the use of proper authentication methods, but also constant monitoring of potential threats, implementation of multi-factor authentication (MFA), use of HTTPS, control of token and session expiration, and the use of high-level encryption algorithms.

Authentication in web systems must not only meet security requirements, but also be efficient in terms of performance and risks associated with credential compromise. To evaluate the effectiveness of each authentication method, you can use mathematical models that include the probability of a successful attack, session duration, number of requests, and number of attacks (Table 2). Below is the mathematical justification for each method.

Basic Authentication is one of the least secure authentication methods due to the lack of encryption. Its mathematical assessment is the probability of a successful data interception attack in the absence of HTTPS.

$$P_{success} = 1-e^{-\lambda t} \tag{1}$$

where $P_{success}$ is the probability of a successful attack, λ is the intensity of attacks (the average number of attacks per unit time), and t is the time of data transmission without encryption. In the absence of HTTPS, the probability of a successful attack increases proportionally with time.

In this method, the main risk is vulnerability to XSS and CSRF attacks. The system vulnerability can be assessed through the probability of session compromise:

$$P_{attack}=PXSS+PCSRF-PXSS\cdot PCSRF \tag{2}$$

where PXSS is the probability of an XSS attack, PCSRF is the probability of a CSRF attack. The system is vulnerable if at least one of these attacks successfully compromises the user's session.

The effectiveness of Session-Based Authentication depends on the security of session cookies and the amount of data that needs to be processed on the server. The probability of a session attack can be calculated using the following formula:

$$P_{attack} = 1-e^{-\lambda n} \tag{3}$$

where n is the number of active sessions, λ is the intensity of attacks on each session. A larger number of active sessions increases the probability of system compromise.

API Keys provide a static authentication mechanism. Without HTTPS, the probability of key interception can be described similarly to Basic Authentication:

$$P_{success} = 1-e^{-\lambda t} \tag{4}$$

However, if you use key rotation, you can modify the formula:

$$P_{success} =1-e^{-\lambda t}\cdot P_{Rotation} \tag{5}$$

where $P_{Rotation}$ is the probability of key rotation before the attack.

SAML provides single sign-on (SSO) for various services. However, due to the volume of data transmitted, there is a risk of DoS attacks due to the increased volume of requests.

The probability of system compromise due to the volume of data:

$$P_{DoS}=1-e^{-\mu q} \tag{6}$$

where μ is the intensity of requests, q is the amount of data transmitted in requests.

OAuth 2.0 uses access tokens, which reduces the risk of password compromise. However, the probability of a token being compromised is determined by the probability of a token interception attack:

$$P_{attack} = 1-e^{-\lambda t} \tag{7}$$

here, t is the time the token is valid before it is rotated.

JWT also uses tokens, but unlike OAuth, the token contains more information, which increases the risk of reusing the token after an attack. Probability of token reuse:

$$P_{recurrence} = \frac{t_{live}}{t_{rotation}} \tag{8}$$

where t_{live} is the lifetime of the token, $t_{rotation}$ is the frequency of token rotation.

SSO reduces the user load by using the same credentials for several services. However, the compromise of one account provides access to all related systems. Probability of system compromise:

$$P_{compr} = 1-(1-P_{attack})^n \tag{9}$$

where n is the number of services accessed through a single account.

Table 2.

Table comparing the effectiveness of authentication methods

Method	Probability of attack	Vulnerability	Resistance to attack	Additional factors
Basic Authentication	$1-e^{-\lambda t}$	No encryption	Low	Need for HTTPS
Form-Based Authentication	$P_{XSS}+P_{CSRF}-P_{XSS}\cdot P_{CSRF}$	XSS, CSRF	Average	Protection against XSS, CSRF
Session-Based Authentication	$1-e^{-\lambda n}$	CSRF, compromising sessions	Average	Server load
API Keys	$1-e^{-\lambda t}$	Key interception	Low	Using key rotation
SAML	$1-e^{-\mu q}$	DoS-attacks	High	Large volumes of data
OAuth	$1-e^{-\lambda t}$	Token interception	High	Securing tokens via HTTPS
JWT	$\frac{t_{live}}{t_{rotation}}$	Reuse of tokens	Average	Token lifespan
SSO	$1-(1-P_{attack})^n$	Compromise of a single account	High	Protecting a single account

EXPERIMENT

For the experimental part, a comparison of the test of obtaining access to a specially created test system and data providing a broader statistical reporting base, which is relevant for 2024 [1, 5] on methods and attempts to obtain unauthorized access to web resources, was used.

In order to get a clear idea of the protection methods and choose the most optimal of them, we have considered:

The probability of a successful attack for each authentication method.

Attack intensity of attacks per unit time for non-commercial websites.

Time to compromise authentication methods.

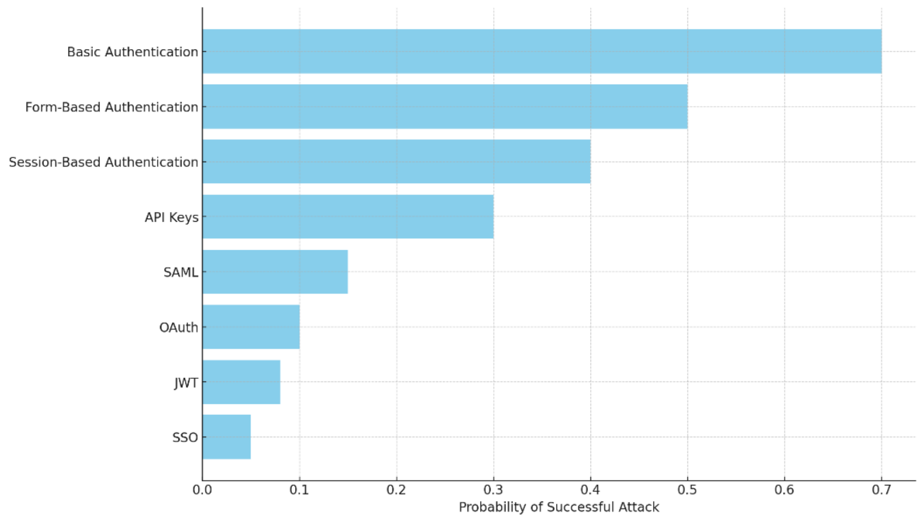

Fig. 1. Comparison of probability of successful attack across authentication methods

As we can see, JSON Web Tokens offer a powerful solution for authentication and authorization in modern web development, particularly for non-financial web applications that need a balanced approach to scalability, security, and integration flexibility. The following highlights why JWT is an optimal choice for such use cases.

A key advantage of JWT is its stateless structure, which enables scalability across distributed systems without requiring server-side session storage. Unlike session-based authentication, JWT encapsulates all necessary information within the token, which is passed with each request. This eliminates the need for centralized session management, reducing server load and supporting applications that anticipate high user traffic or integration with multiple services. For web applications requiring efficient scaling across instances, JWT's stateless nature provides a significant benefit.

JWTs are compactly encoded in JSON format, making them lightweight and easy to transfer between systems. This efficiency reduces communication overhead and makes JWT an excellent choice for mobile and IoT devices with limited bandwidth and computing power. The standard structure of a JWT—with its header, payload, and signature—ensures that essential information is securely contained within the token, supporting efficient and streamlined authentication.

Regarding security, JWT provides strong protection through cryptographic signing with HMAC or RSA/ECDSA keys, ensuring that token content cannot be modified once issued. Any unauthorized alteration will cause signature verification to fail, making JWT an effective safeguard against tampering. JWT also supports expiration claims (exp), which allows developers to set token lifetimes and mitigate risks associated with token reuse. Additional claims, such as "issued at", "not before", and JWT ID, give developers greater control over the token's validity and usage, enhancing the security profile of JWT-based authentication.

Another notable benefit is JWT's flexibility. Its payload can include various data types, supporting authentication and authorization needs. Custom claims for roles, permissions, or user-specific data can be embedded in the token, allowing the backend to determine user privileges efficiently. This minimizes the need for frequent database queries, reducing latency and improving application performance, particularly in high-traffic environments.

JWT's adaptability across different programming languages and support within numerous frameworks make it a versatile solution for a wide range of web applications, from API authentication to mobile sessions. For non-financial sectors, JWT's balance of security, scalability, and integration ease makes it an ideal choice, providing the necessary protection without compromising on performance or usability.

JWT is significantly more advantageous compared to methods based on SSO (Single Sign-On) because it does not require centralized storage of sessions and allows easy scaling in distributed systems. This is especially true for web applications with high traffic and numerous integrations, where it is important to reduce the load on the server and ensure uninterrupted operation. Also, JWT provides the ability to configure individual attributes of the token, which reduces the number of requests to the database and increases the speed of the system.

CONCLUSIONS

The article emphasizes the need to improve authentication methods in web systems, taking into account modern cyber threats and technological challenges. The use of outdated approaches, such as passwords, no longer provides an sufficient level of protection. Modern token-based technologies, such as OAuth and JWT, are more efficient, but they also require additional security measures to prevent interception or reuse of tokens. An important aspect is the introduction of multi-factor authentication and secure data transfer protocols. The authors conclude that to improve the security of web systems, it is necessary to combine different authentication methods, taking into account their advantages and disadvantages, to ensure maximum protection of user data without sacrificing performance or usability.

The adoption of JWT as a standard for authentication and authorization in web development provides numerous benefits, including scalability, security, efficiency, and interoperability. Its stateless nature and ability to carry payloads that include claims relevant to both authentication and authorization make JWT a robust solution for modern web applications. By eliminating the need for server-side session management, JWT facilitates the development of scalable, high-performance systems that can securely integrate with third-party services and APIs.

When implemented with best practices such as HTTPS encryption, token expiration, and proper key management, JWT offers a highly secure and flexible method for managing user identity and access control in web development. As the complexity and scale of web applications continue to grow, JWT stands out as a reliable and efficient solution for meeting the authentication and authorization needs of most systems.

References

1. 2024 Data Breach Investigations Report. Verizon Business. https://www.verizon.com/business/resources/reports/dbir/ (n.d.). Accessed 10 Nov 2024
2. Bucko, A., et al.: Enhancing JWT authentication and authorization in Web applications based on user behavior history. Computers. 12(4), 78 (2023). https://doi.org/10.3390/computers12040078
3. Dimitrijević, N., et al.: Advanced security mechanisms in the Spring framework: JWT, oauth, LDAP and keycloak. Fourteenth Int. Conf. Bus. Inf. Secur. (BISEC'2023). 3676, 64–70 (2024). https://ceur-ws.org/Vol-3676/short_09.pdf
4. ELHejazi, M.F., Muragaa, W.H. A.: Improving the security and reliability of SDN controller REST APIs using JSON Web Token (JWT) with OpenID and Auth2.0. In: 2024 IEEE 4th international maghreb meeting of the conference on sciences and techniques of automatic control and computer engineering (MI-STA), Tripoli, Libya, 19–21 May 2024. IEEE (2024). https://doi.org/10.1109/mi-sta61267.2024.10599643
5. IBM Security X-Force Threat Intelligence Index 2024. IBM. https://www.ibm.com/reports/threat-intelligence (n.d.)
6. Innocenti, T., et al.: OAuth 2.0 redirect URI validation falls short, literally. In: ACSAC '23: annual computer security applications conference, Austin TX USA. ACM, New York, NY, USA (2023). https://doi.org/10.1145/3627106.3627140
7. Lakhno, V., et al.: A model developed for teaching an adaptive system of recognising cyberattacks among non-uniform queries in information systems. Eastern-European J. Enterp. Technol. 4(9(82)), 27 (2016). https://doi.org/10.15587/1729-4061.2016.73315

8. Navas, J., Beltrán, M.: Understanding and mitigating OpenID Connect threats. Comput. & Secur. 84, 1–16 (2019). https://doi.org/10.1016/j.cose.2019.03.003
9. Primbs, J., Menth, M.: OIDC2: open identity certification with OpenID connect. IEEE Open J. Commun. Soc. 1 (2024). https://doi.org/10.1109/ojcoms.2024.3376193
10. Rushdy, E., Khedr, W., Salah, N.: Framework to secure the OAuth 2.0 and JSON Web Token for REST API. J. Theor. Appl. Inf. Technol. 99 (9), 2144–2161 (2021)
11. Seo, J.: The future of digital authentication: blockchain-driven decentralized authentication in Web 3.0. J. Web Eng. 611–636 (2024). https://doi.org/10.13052/jwe1540-9589.2351
12. Venčkauskas, A., et al.: Enhancing microservices security with token-based access control method. Sensors. 23(6), 3363 (2023). https://doi.org/10.3390/s23063363
13. Wiefling, S., Hönscheid, M., Lo Iacono, L.: A privacy measure turned upside down? Investigating the use of HTTP client hints on the web. In: ARES 2024: the 19th international conference on availability, reliability and security, Vienna Austria. ACM, New York, NY, USA (2024). https://doi.org/10.1145/3664476.3664478
14. Xu, B., et al.: JWTKey: automatic cryptographic vulnerability detection in JWT applications. In: Computer security – ESORICS 2023. Pp. 263–282. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-51479-1_14