IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Jwt Token With Shared To Secret Key For Modern Web Application

[1] Mr. ANANDH S, [2] Mr. MARIMUTHU R,

[1] Mr. ANANDH S, M.sc CFIS, Department of Computer Science and Engineering, Dr. MGR UNIVERSITY, Chennai, India

[2] Mr. MARIMUTHU R, Centre for Cyber Forensic and Information Security, University of Madras, Chepauk, Chennai.

Abstract: A JWT (JSON Web Token) is a widely used technique for securely transmitting information between parties as a JSON object. These URL-safe tokens are used for authentication and information exchange. It is composed of three components: a header, a payload, and a signature, all encoded as a single string. The payload JSON Web Signature (JWS) format, or as the plaintext of a JSON Web Encryption (WE) format, is about a subject (typically the user) and extra the secret value information. The signature is generated by computing the encoded header, encoded payload, and a common secret key, which provides the token's integrity and authenticity. The salt value secret method supports encryption, key value resource and identification, authentication, and authorization. It includes attack prevention responses to web-based attacks like SQL injection, cross-site scripting (XSS), and Cross-Site Request Forgery (CSRF) for web security. From the results of the research, this idea is highly applicable to applications or information systems on various platforms sharing the same service. JWT tokens are also successfully stored in cookies in a browser. A shared secret value plays an important part in ensuring the integrity and security of the token. This delves into the importance of the shared secret within the process of signing and verifying JWTs, which is utilized for authentication and authorization in the current web application endpoint.

Index Terms **- JWT (JSON Web Tokens), Secret value, Authentication, Web Security, Token Integrity, Symmetric Encryption.**

I. INTRODUCTION

Applications or information systems are technology employed in most companies. By and large, each company possesses different types of systems or applications that are employed to assist workers in working methodically. The current systems or applications are still not integrated into each other, so most processes share the same purpose on other systems, such as the authentication process is designed based on the web service concept.

Interoperability or integration of information system software with multiple components, which can produce secret value that has the ability to break system security. Different means of minimizing threats to security in web services have been implemented in earlier research, such as JSON Web Token (JWT) for authentication in Shared to secret value Web Service-based Architecture Interoperability in research. The use of JSON Web Token authentication [1] has also been implemented through the inclusion of the Encryption and secret value of increases bite size by research.

Other research on the authentication and load balancing system with the use of JSON Tokens for Multi-Agent Systems. But based on some previous researches that investigated the utilization of JSON Web Token as a mechanism for authentication and data transfer, still utilized the RestFul API process [2] which is slow in the API process, this research will utilize another encryption, i.e. salt and number of bites size which utilizes a token storage mechanism.

JSON Web Tokens (JWTs) are a crucial aspect of contemporary web applications through providing secure and stateless user authentication. Following a successful login, a server gives a JWT to the user, which is incorporated into further requests to authenticate and preserve user identity. This eliminates server-side storage of sessions, enabling scalable stateless sessions. JWTs also assist in enforcing API authorization, permitting secure communication between microservices by checking the token's integrity and claims. JWTs also enable cross-service authentication, allowing various applications within the same environment to exchange authentication information securely utilizing a standardized common standard.

II. LITERATURE REVIEW

Heni Sulastri and Rizal Nugroho [3]. had proposed a study with and compared the base JWT against HMAC SHA-256 and optimized using HMAC SHA-512. The performance of the algorithm in terms of token size and execution time was used for evaluation. It was revealed through results that there was 1% enhancement in execution time using SHA-512, even though there was a 2% increase in token size against SHA-256. The security gap identified in the paper is the sufficiency of current security measures in RESTful Web Services. The research suggests filling this gap by using JSON Web Token (JWT) with the HMAC SHA-512 algorithm to improve the security of the web services.

Rohmat Gunawan [4]. had proposed the methodology entails crafting a JWT-based authentication system for a RESTful Web Service, with phases such as user login, token creation, and backend verification. PHP implementation, HTTP server response testing, and token data verification provide a secure and operational system. The paper does not discuss token storage security, especially in local storage (cookies), which introduces questions about its safety. There is a need for identifying more appropriate encryption algorithms for the use of JSON Web Token (JWT) in RESTful Web Service applications.

Yogiswara and Wijono Dahlan [5]. had proposed the methodology encompasses designing and applying a data integration model that utilizes XML-RPC, SOAP, and REST web service approaches. Performance assessment encompasses the measurement of transaction rate and user-perceived latency with emphasis on determining the ideal configuration—emphasizing the "Apache" client, "Nginx" server, and the "REST" approach—for effective implementation of data integration. The paper discusses a data integration model developed using XML-RPC, SOAP, and REST web service approaches, analyzing their performance on various web servers. The findings reveal that the REST approach, which is implemented in the client mode by Apache and in the server mode by Nginx, offers the optimal latency for data integration.

Digvijaysinh M Rathod [6]. had proposed developed a prototype system, coupled with a mobile client, to experiment on comparing the performance and scalability of SOAP/WSDL and RESTful web services, demonstrating better results for RESTful services in mobile settings. The paper is short on an essential discussion regarding the security aspects of opting between RESTful and SOAP/WSDL web services, addressing mostly performance factors.

KV Kanmani, P.Sc and S. Smitha [7]. had proposed multiple methods in Restful Web Services, highlighting the combination of REST and OAuth 2.0. But it misses a thorough research methodology like experimental design or data gathering methods, which restricts the level of scrutiny. The gap found is in the lack of real-world examples or empirical verification of the envisaged REST and OAuth combination. The work would be enriched by highlighting the implementation of ideas discussed in case studies, thus increasing the practical applicability and relevance of techniques offered.

S. Dalimunthe, E. Hasri Putra, and M. A. Fadhly Ridha [8]: Information system software integration or interoperability between different components can leave holes that can affect the system security. Security in this research has been applied in web services through JSON Web Token (JWT) using the HMAC-SHA512 algorithm that is saved in browser cookies. This idea is very appropriate to be used in applications or information systems across different platforms that utilize the same service based on the findings obtained. JWT tokens are also stored successfully in browser cookies. Apart from that, a comparison was also performed between the HMAC-SHA512 and HMACSHA-256 algorithms, and in the ultimate result, it was observed that the total time gap was 185 ms and the average time gap was 6.17 ms. Thus, it can be concluded that the HMAC-SHA512 algorithm is 0.9861% faster than the HMAC-SHA256 algorithm.

Bucko, A.; Vishi, K.; Krasniqi, B.; Rexha, B [9]: JSON Web Tokens (JWTs) following RFC 7519 are commonly used as a standard for authentication and authorization of users. But these tokens don't keep track of the user's behavior history. For this, this paper proposes a solution to provide greater trust in user authentication in web applications based on behavior history. The solution takes into account the count of password attempts, consistency of IP address, and type of user agent and assigns a weight or percentage value to each. These weights are added up and saved in the user's account and get updated after every transaction. The suggested method was implemented using the .NET framework, C# programming language, and PostgreSQL database. The findings indicate that the proposed solution efficiently enhances the level of trust in user authentication. The paper concludes by summarizing the advantages and weaknesses of the proposed solution.

Adhitya Bhaviyuga, Mahendra Data and Andri Warda and et al., [10]. had proposed research implements a token-based authentication mechanism for MQTT in resource-constrained IoT devices, with a conditional token generation process, storage, and usability/performance testing. Token-based authentication on resource-constrained IoT devices can compromise system efficiency due to limitations in processing power and memory.

III. PROPOSED METHODOLOGY

A JWT is a small, tokenized presentation containing a header, payload, and signature. The payload contains claims or user information that are digitally signed with a shared secret key. The key should be known only to authorized servers. When a token is presented, it is signed with this secret key to guarantee the integrity and authenticity of information. Upon receiving the token in a request, the server verifies the same key on seeing its signature. If genuine, the user gets authenticated without requesting the database or storing client-side sessions.

This section describes a formal method for implementing and managing the shared secret value in JWT-based authentication in contemporary web applications. The suggested method guarantees secure generation, verification, and storage of tokens while solving potential security risks related to shared secrets. A token is then transmitted to the client, usually stored in an HttpOnly cookie or a secure storage solution, to accompany requests in the future for authentication. The server authenticates the token by validating its signature using the same shared secret key to ensure that the request is being made by an authenticated user.

3.1. Javascript Object Notation (JSON)

Javascript Object Notation (JSON) is a client-side to server-side communication of this request and identifies the users. JSON is derived from a subset of JavaScript programming and is standardized in the ECMA-263 document. JSON is an id random format since it employs a shared secret of key value. The JSON key and value pairs are JSON objects that are separated using commas shared to secret methods [11].

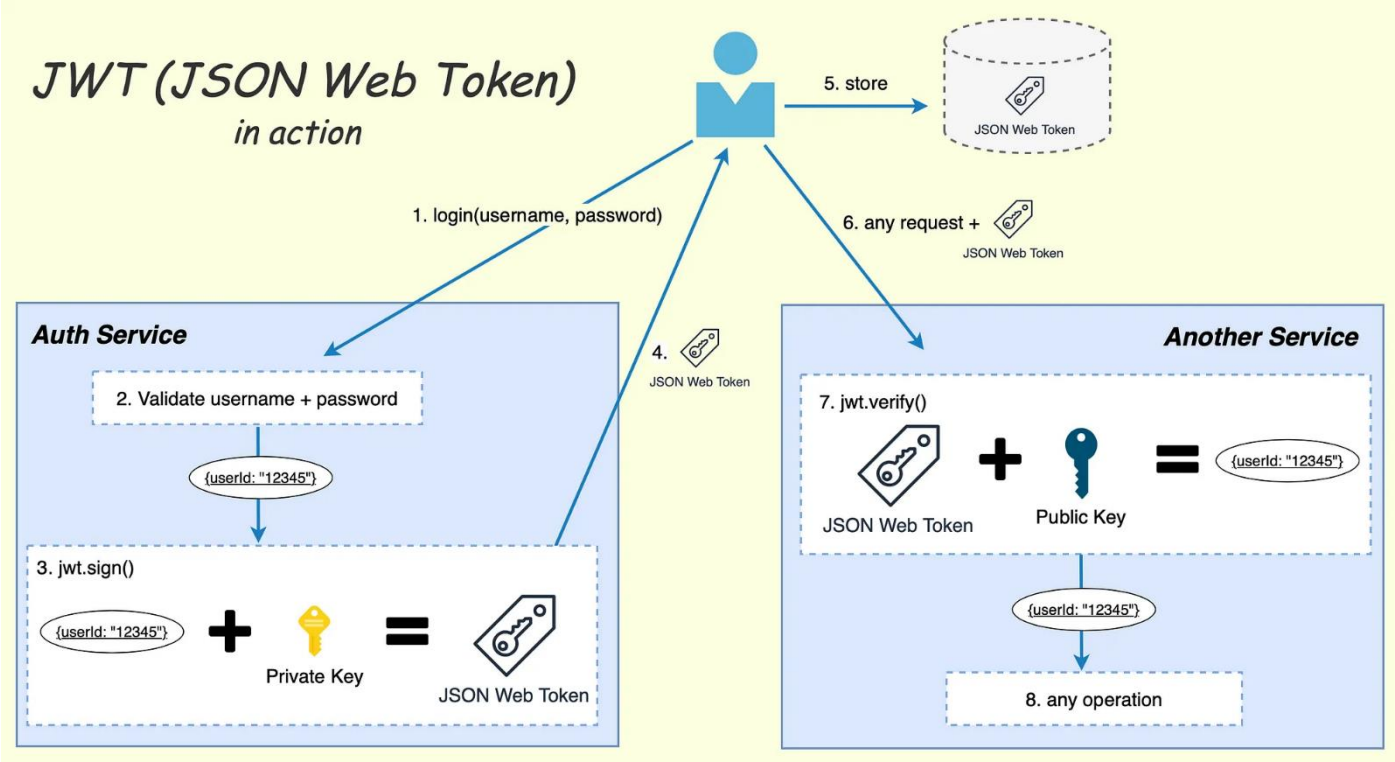


Figure - 3.1.1 JSON Web Token Authentication Methods.

3.2. Shared Secret Generation

Use a hash to secure a pseudo-random number generator to generate a strong shared secret key (e.g., 256-bit or 512-bit key for Shared to secret key, HS256 or HS512 algorithms). Select a suitable MAC algorithm (e.g., HS256) and ensure the length of the key matches security best practices to withstand brute-force attacks [12]. Environment Variables keep the shared secret in environment variables instead of hard-coding it within the application code. The Secret Management Tools Use secure secret management tools (e.g., AWS Secrets Manager, Azure Secrets Manager) to store and retrieve the shared secret securely. An Access Control Limit access to the shared secret to only authorized security, e.g., authentication server and resource server.

A JWT Token Header, Signature, and payload, including (issuer), exp (expiration time), and sub (subject), to validate tokens correctly. Signing Process Sign the JWT with the shared secret using the chosen MAC algorithm (e.g., HS256) to protect token integrity and authentication. Token Expiry: Have short-lived tokens to reduce the risk of the token, with expiration times usually between a few minutes to an hour.

3.3. Token Validation and Verification

The way JSON Web Token works in verifying incoming data starts with receiving the token. Then the token is validated and detected whether the token is valid. If yes, then the signature is validated, if not, then the token is invalid, and an error message is given. After that, the validation time is corrected whether it is still valid or expired. If yes, it will check the time before it expires, and an invalid token and an error message will be displayed. The time does not exceed the expiration or expiration of a session. If not, the token has expired and an error message appears. After the session is created, services will be provided to the information system according to previously verified data. JSON Web Token verification workflow [13] .

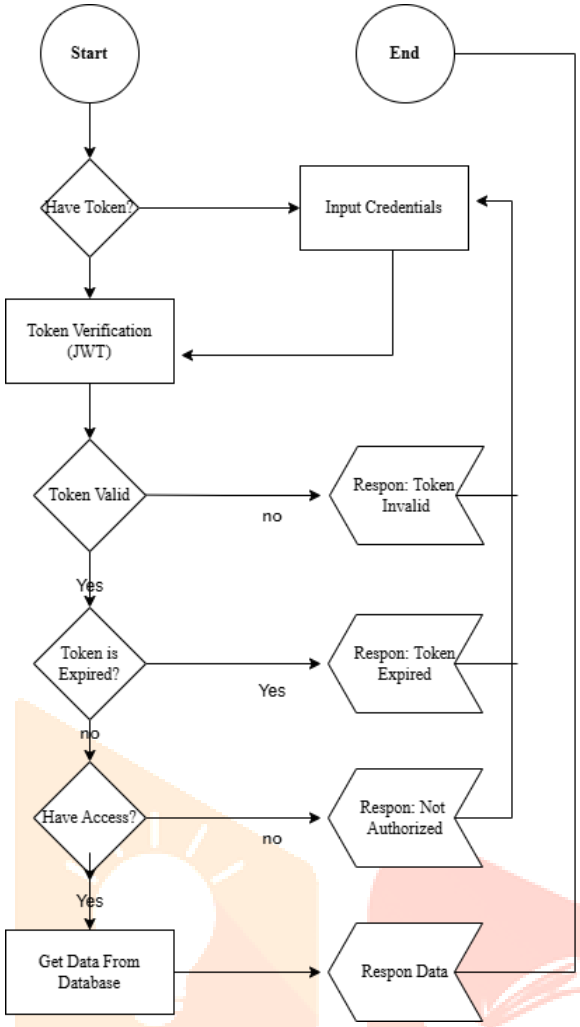


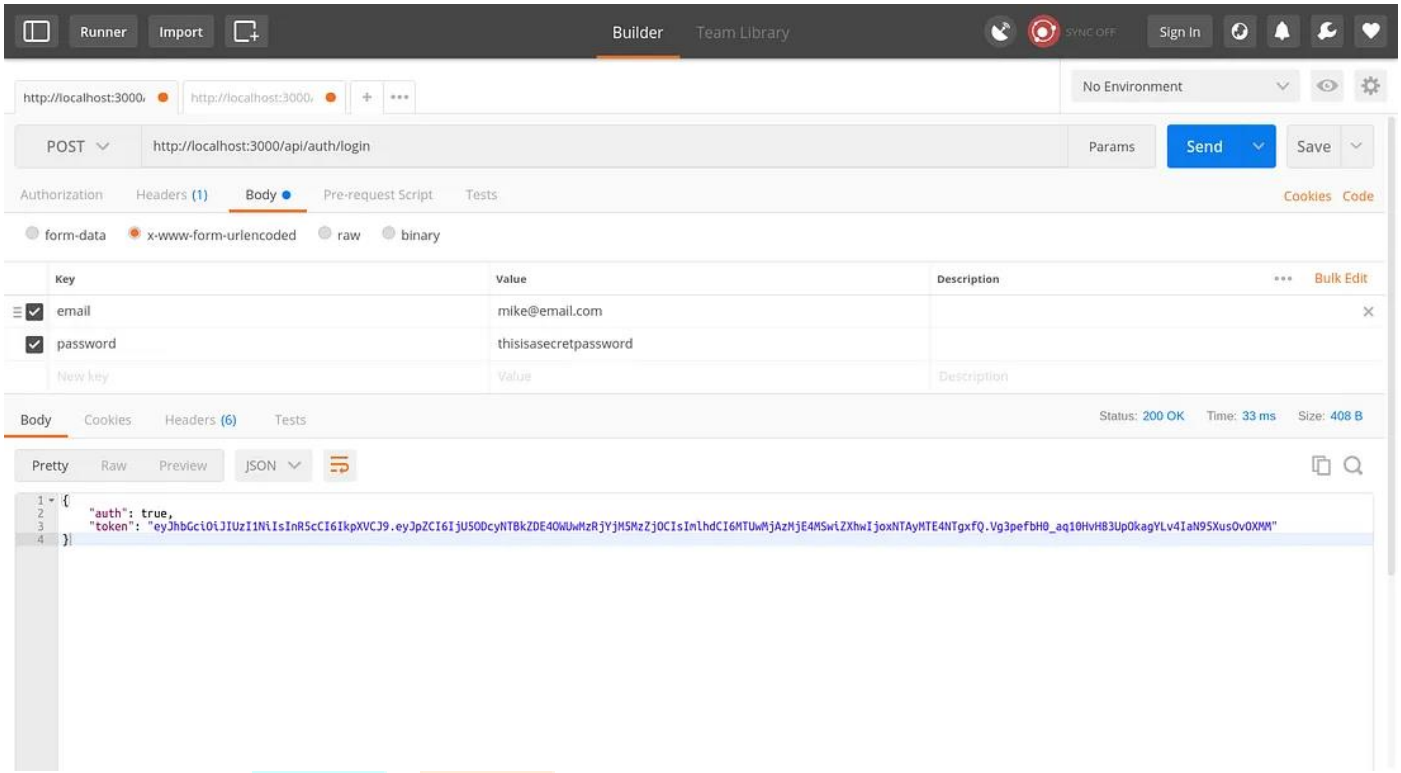
Figure - 3.3.1 JSON Web Token Authentication Mechanism.

IV. FINDINGS

An application with a shared secret value in JWT authentication for contemporary web apps offers security. A sufficient, hash secure secret (minimum 256 bits) is required for the brute-force attacks to generate tokens without authorization. To the advantage, shared secret-derived JWT authentication is easy to deploy, computationally light, and ideal for stateless, elastic systems such as microservices and single sign-on (SSO) systems [14].

A solution also poses substantial challenges, especially regarding securely storing and delivering the shared secret. Once the secret is compromised, the entire authentication system is at risk, which emphasizes the necessity of strong key storage, rotation, and access control mechanisms. The stateless nature of JWTs also complicates token revocation, requiring extra steps like short token lifetimes or blacklisting. JWT token verification, and secure key management, shared secret-based JWT authentication for modern web applications.

In a Node.js web application, one may use a secret key to authenticate APIs using authentication mechanisms like JWT (JSON Web Tokens) or API keys [15]. The secret key is generally kept secure in a .env file and then accessed via environment variables so that it is not hard-coded into the source code. While using Postman to test the API, the client has to include the token or API key in the request headers. An obtained JWT token after successful login can be passed in the Authorization token to access protected routes. This process guarantees that access to sensitive information or the execution of specific actions is limited to authorized users, which makes secret key management a critical component of web application secure development.



4.1. JSON Web Token Shared to Secret Key Authentication Application.

V. CONCLUSION

The shared secret key for JWT tokens provides an easy and effective way to implement secure authentication in contemporary web applications. Though its statelessness and support for HMAC algorithms make it suitable for scalable systems, issues such as secure key management and token revocation need to be resolved. Through adherence to best practice—secure storage, key rotation, and enforcement of HTTPS—developers are able to utilize shared secret-based JWTs in constructing resilient, high-performance authentication mechanisms capable of responding to the requirements of contemporary web applications. JSON Web Token (JWT) utilizing a shared secret key (HMAC-based signature, e.g., HS256) [16] continues to be a common practice in the implementation of securing contemporary web applications as it is both simple and effective. The secret key shared among parties means that only parties in possession of the key can create and authenticate tokens, offering a secure method to authenticate users and send data from client to server. As JWTs are stateless, they are lightweight on server-side session storage, thereby suitable for large-scale, distributed systems. But key management is also important—once the secret key is compromised, an attacker will be able to create fake tokens and defeat security restrictions. Therefore, developers need to impose strict key rotation policies, employ HTTPS to avoid token interception, and employ short-lived tokens with refresh mechanisms to counter threats.

VI. REFERENCES

[1] I. Darmawan, A. P. A. Karim, A. Rahmatulloh, R. Gunawan and D. Pramesti, "JSON Web Token Penetration Testing on Cookie Storage with CSRF Techniques", 2021 International Conference Advancement in Data Science E-learning and Information Systems (ICADEIS), pp. 1-5, 2021.

[2] Chatterjee, A., & Prinz, A. (2022). Applying Spring Security Framework with KeyCloak-Based OAuth2 to Protect Microservice Architecture APIs: A Case Study. Sensors, vol. 22, no. 1703.

[3] A. Neumann, N. Laranjeiro, and J. Bernardino, “An Analysis of Public REST Web Service APIs,” IEEE Trans Serv Comput, vol. 14, no. 4, pp. 957–970, Jul. 2021, doi: 10.1109/TSC.2018.2847344.

[4] R. Bandara, M. Fernando, and S. Akter, “Privacy concerns in ecommerce: A taxonomy and a future research agenda,” Electronic Markets, vol. 30, no. 3, pp. 629–647, 2020.

[5] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Stateful versus stateless selection of edge or cloud servers under latency constraints," in 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2022. doi: 10.1109/WoWMoM54355.2022.00022 pp. 110–119.

[6] Li, W., Jia, S., Liu, L., Zheng, F., Ma, Y., Lin, J.: CryptoGO: automatic detection of go cryptographic API misuses. In: Annual Computer Security Applications Conference, pp. 318–331 (2022).

[7] E. Gashi, B. Rexha and A. Rexhepi, "Trust establishment between OAuth 2.0 resource servers using claims-based authorization", *Electron. Gov. Int. J.*, vol. 17, pp. 3, 2021.

[8] M. Lodder, "Token based authentication and authorization with zero knowledge proofs for enhancing web api security and privacy," Dissertation, Dakota State University, Department of Computer Science, 2023, doctor of Philosophy in Cyber Operations (PhDCO).

[9] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Stateful versus stateless selection of edge or cloud servers under latency constraints," in 2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2022. doi: 10.1109/WoWMoM54355.2022.00022 pp. 110–119.

[10] Akanksha and A. Chaturvedi, "Comparison of Different Authentication Techniques and Steps to Implement Robust JWT Authentication", 2022 7th International Conference on Communication and Electronics Systems (ICCES), pp. 772-779, 2022.

[11] P. Mahindrakar and U. Pujeri, "Insights of json web token," International Journal of Recent Technology and Engineering (IJRTE), vol. 8, no. 6, 2020.

[12] Michaelides, M., Sengul, C., Patras, P.: An experimental evaluation of MQTT authentication and authorization in IoT. In: WiNTECH'21: Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & CHaracterization, New Orleans, LA, USA, 4 February 2022, pp. 69–76. ACM (2021).

[13] Ami, A.S., Cooper, N., Kafle, K., Moran, K., Poshyvanyk, D., Nadkarni, A.: Why crypto-detectors fail a systematic evaluation of cryptographic misuse detection techniques. In: 2022 IEEE Symposium on Security and Privacy (SP), pp. 614–631. IEEE (2022).

[14] E. Gashi, B. Rexha and A. Rexhepi, "Trust establishment between OAuth 2.0 resource servers using claims-based authorization", *Electron. Gov. Int. J.*, vol. 17, pp. 3, 2021.

[15] A. Bucko, K. Vishi, B. Krasniqi and B Rexha, "Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History", *Computers*, vol. 12, pp. 78, 2023.

[16] Sheffer, Y., Hardt, D., Jones, M.: RFC 8725: JSON web token best current practices. Tech. rep, Internet Engineering Task Force (2020).