Enhancing Web Application Security with OAuth 2.0 and JWT

Punit Sharma 12114945
Kuldeep Verma 12110095
Shraddha Gupta 12112893
Vikash 12116128

Abstract:

In today's digital age, web applications have become integral to communication, commerce, and information sharing, making robust security essential to protect user data and maintain trust. Despite their benefits, web applications are susceptible to security threats such as data breaches and malware. To mitigate these risks, OAuth 2.0 and JSON Web Tokens (JWT) are widely adopted to enhance web application security. OAuth 2.0 is a flexible authorization framework that allows third-party applications limited access to resources without sharing user credentials, while JWT serves as a standardised, token-based solution for secure communication and session management. This paper examines the role of OAuth 2.0 and JWT in safeguarding web applications, addressing key components, mechanisms, and use cases, including authentication and authorization, session management, and protection against prevalent vulnerabilities like cross-site scripting (XSS), SQL injection, and denial of service (DoS) attacks. By exploring the advantages and security practices associated with OAuth 2.0 and JWT, this study underscores their critical importance in building secure and resilient web applications.

Introduction:

As a primary form of transaction and communication, web applications are increasingly prevalent in this modern world where just about everything is done online. And like many online things these days, this comes at a cost. Even though web applications are so handy and do a lot of wonders, they face a number of security attacks like revenge, social engineering, hackers, and malicious through which private user data can be compromised and break trust in the services. There is a huge demand for the developers to stay one step ahead and secure user data from any vulnerabilities. To offer assistance to avoid such dangers, secure verification, and authorization strategies have ended up a standard in cutting edge web applications. OAuth 2.0 and JWT (JSON Web Token) are two such shapes that have picked up notoriety for improving the security of web applications. OAuth 2.0 is an authorization system that empowers a third-party application to get constrained to an HTTP benefit either on sake of an asset proprietor or by permitting the third party to regulate the HTTP benefit. It is used on the web and other potential uses to obtain limited access to a HTTP service without sharing the username and password.In contrast, JWT is a standard token format that contains claims that are securely transferred between parties as a means of communication. Despite the RFC 7519 of the standard token format.

Literature Survey:

The implementation of secure authentication and authorization mechanisms has been widely researched, especially with the rise of OAuth 2.0 and JWT for web application security. OAuth 2.0 was formally introduced in *The OAuth 2.0 Authorization Framework* by Hardt (2012), which provides a foundational explanation of the framework and its ability to enable secure, token-based access for third-party applications without requiring users to expose their credentials highlights the framework's adaptability through different

authorization flows, such as the Authorization Code and Client Credentials grants, making it a versatile solution for various web applications.

Further, *The OAuth 2.0 Authorization Framework: Bearer Token Usage* (2012) by Jones and Hardt discusses the role of bearer tokens within OAuth 2.0, emphasizing the significance of token usage in securing client-server communication. The authors explain how the Bearer Token allows repeated, authenticated access once issued, strengthening security without necessitating constant reauthentication . Sungch-Yeon Yoo, and Yoohwan Kim (2016) extended this work by introducing an authentication system for stateless RESTful services, proposing ID-Based Authentication (IBA) to reduce login overhead by eliminating session-based login states and enabling efficient, continuous access for users .

The comparisonbased and session-based authentication in *Token-Based vs Session-Based Authentication: A Survey* explores JWT's utility in replacing session tokens with self-contained tokens that can be securely transmitted across clients and servers, thus facilitating scalable, stateless communication. The survey highlights JWT's role in maintaining data integrity through cryptographic signing, which is critical for protecting against threats like token tampering and replay attacks .

Another significant conis Nagarjun and Shakeel's (2020) work on Cross-Site Scripting (XSS) vulnerabilities, *Cross-site Scripting Research: A Review,* which discusses XSS as a prevalent security issue in web applications and proposes input validation, output encoding, and Content Security Policy (CSP) as mitigation strategies . This research underlines the imposecure token handling to mitigate injection-based attacks when using JWT for session management in OAuth-based implementations.

Together, these studies provide a comprehensive understanding of OAuth 2.0 and JWT as security mechanisms for modern web applications, with each addressing various security concerns—from authorization and stateless session management to vulnerability mitigation strategies. The literature supports the premise that implementing OAuth 2.0 and JWT can significantly enhance security, given proper configuration and attention to potential vulnerabilities.

What is Authentication:

In recent years, technology has profoundly impacted every country, influencing companies, educational institutions, and nearly every other sector. Services once provided directly by people are now largely replaced by digital alternatives. Today, tasks like shopping and learning are easily accessible online, and users have accounts on numerous websites to store data or make purchases. As these online interactions often involve financial transactions, they attract individuals looking to exploit others by stealing data. Consequently, we must establish reliable methods to verify legitimate users and safeguard against imposters.

"The authentication in information technology is the process of verifying who you are, are you the one you pretend to be or not."[3]

Related Work:

The publication *"The OAuth 2.0 Authorization Framework"* [5] provides an overview of the OAuth framework, explaining how token-based authentication is achieved using JSON Web Tokens (JWT). Meanwhile, *"The OAuth 2.0 Authorization Framework: Bearer Token Usage"* [6] highlights the importance of the Bearer parameter, which includes a token added to the request header. This token is essential for each client-to-server request, allowing secure and continuous authentication across interactions.

OAuth 2.0 (Authorization Framework):

OAuth 2.0 is an open standard framework for authorization aimed at providing to users controlled access to their resources by third parties without needing to share their credentials such as passwords. It allows for the controlled delegation of authority to the clients by assigning access tokens with specific scopes to the resources. This strategy mitigates exposure of sensitive data, which generally raises security and improves the user experience.There are some actors involved in the OAuth 2.0 framework including the resource owner (user), the client (application requiring permission), the authorization server (the one providing authentication and token issuance) and the resource server (where important documents are stored). Different OAuth 2.0 flows are available such as: Authorization Code grant, Implicit grant, Password Credentials Extension Grant and Client Credentials Grant, hence it is flexible enough covering many use cases.With OAuth 2.0, issuing separate access tokens allows applications to make requests with just sufficient authorization without exposing any of the underlying user's secrets. Session control and assertions are generally conducted with the use of JSON Web Tokens (JWT) making OAuth 2.0 very essential in web application security.

OAuth or HTTP authentication holds a session from when the user logs in to when he logs out. Thus, every time the user gets logged out, he needs to log into AS which is the short term for authentication server or RS which is the short form for resource server for authentication[7]. On the other hand, IBA proposed in[7] does not have any such logging in or logging out concept. Thus, IBA which stands for ID based authentication eliminates the overhead of the login process.

JWT (Token Based Authentication):

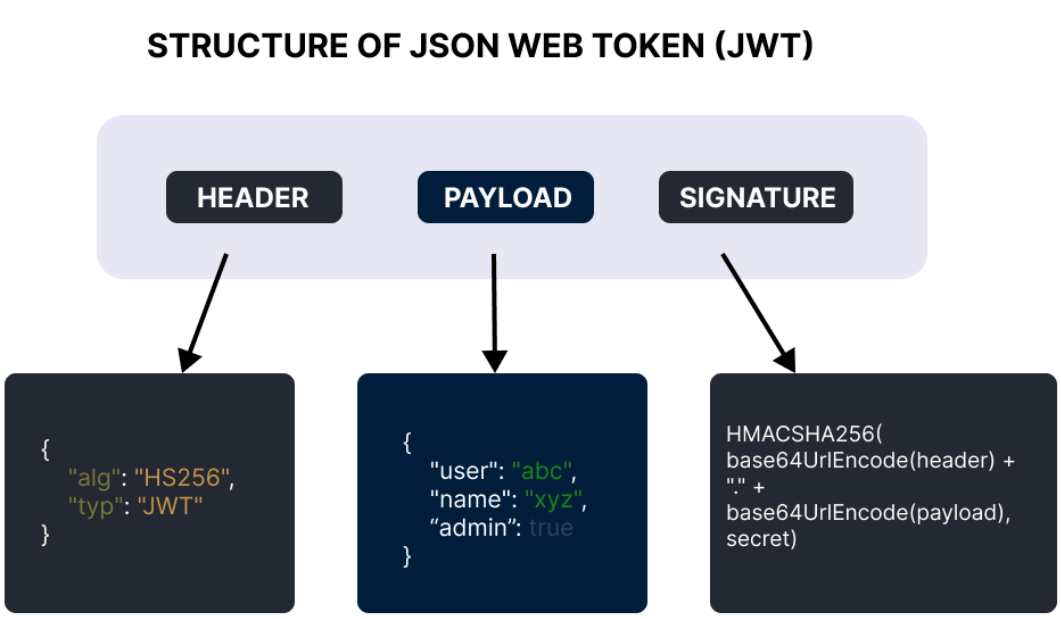
JSON Web Tokens (JWT), in the simplest terms, is a secure way to provide information in the form of a JSON object. The use of JWT is common for authentication and authorization also within contemporary web applications developed today. It mainly helps in passing claims between client and server caregivers in a most trustworthy and efficient way which, in most cases, is forms of digital signatures that enhance data authenticity.
A JWT consists of three major components:

1. **Header**: The header typically describes what type of token it is (e.g. JWT) and the algorithm used to sign that token (for example HS256 or RS256).
2. **Payload**: The payload contains the claims, which are facts about the user (such as userID, user roles and user expiration). This section is base64 encoded and holds the data that the server requires in order to handle the request.

3. **Signature**: The signature is formulated by taking the encoded header and payload plus a secret key or a pair of public/private keys. This prevents tampering by other parties since the token will be useless once the signature is missing, thus preserving its integrity.

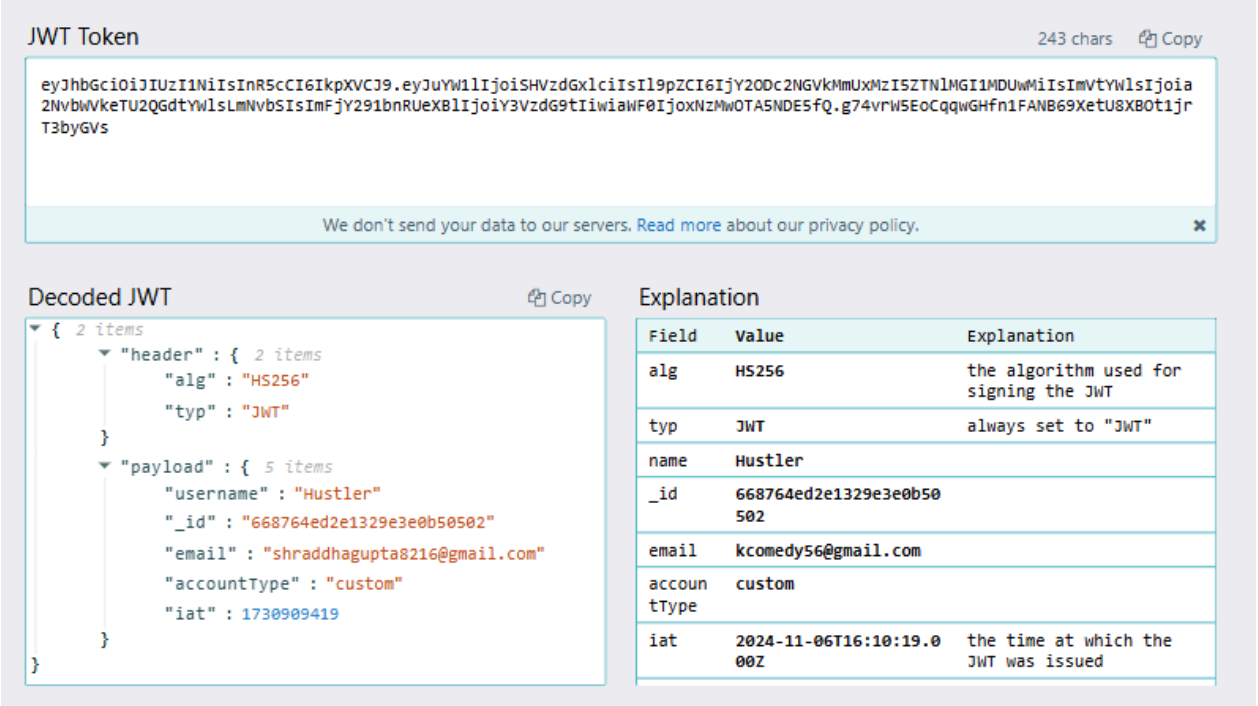
Creating JWT in NodeJS:

```
function generateToken(user) {
    const payload = {
        id: user.id,
        username: user.username,
    };

    const secretKey = 'yourSecretKeyHere';

    const options = {
        expiresIn: '1h'
    };

    const token = jwt.sign(payload, secretKey, options);
    return token;
}
```



Once a JWT is issued it is possible to log in and use it for resource access. The absence of session classification in JWTs means that there is no need to store information regarding particular sessions on the server, since all the information is encoded in the token itself. Commonly, JWT is used in combination with OAuth 2.0 where it is an access token for resources that require confidentiality.Being effective and effective while at the same time permitting no compromises to the confidentiality of information, makes the use of JWT very indispensable in the securing of web applications, where session management is made lighter without compromise of security.Authorization has to be strong and in this case.

Decoding JSON Web Token:



JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA [5][6].

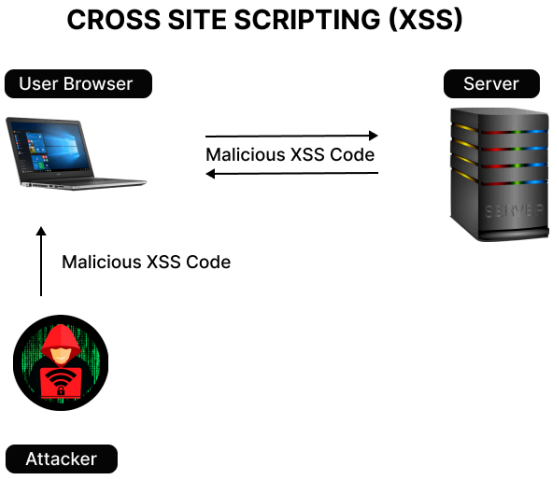
Security Issues In Website:

In this age of digitalization, websites are in sharp focus for multiple security threats of data breaches to malware infection. It may cause severe financial losses and reputation damage along with various legal consequences to an organisation as well as users. Some common security issues of the website security scan are SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF), which is mainly exploited due to weaknesses in the code or configuration. Since websites contain and process private information, such security problems have to be taken extremely seriously in order to safeguard privacy and integrity. Proper coding, encryption, and vulnerability assessments can protect the website from the ever-changing nature of the threats in cyberspace.

Types of security issues faced in website:

1. **Cross-site scripting(xss):** Cross-Site Scripting (XSS) is one of the most dangerous forms of attack on web applications and connected devices that perform many application-based activities. XSS can be defined as an attack in which the attacker manages to get executable scripts onto the trusted websites in question without

filtering input. In XSS attacks, the assailants may collect session ids, passwords or perform actions without the consent of the users. The chances of XSS attacks taking place are more than ever thanks to the rising number of web applications. To deal with such attacks, these applications should include measures such as input validation, output encoding, and other protective mechanisms such as Content Security Policy (CSP)[2].



2. **Denial of service (DOS):**

Denial of Service attacks are malicious attempts to stop a server or a network from functioning normally. The objective of a Denial of Service attack is overloading the target with consumed resources, thereby making the system unavailable to the actual users. Well-known methods for carrying out DoS attacks include bandwidth and connectivity attacks in which too much traffic is forwarded to the system. Important types of DoS attacks include TCP-SYN floods, UDP floods, and ICMP floods[1]. Each method aims at certain weaknesses to overwhelm the server, network, or application. To reduce DoS risks, organisations use techniques such as traffic filtering, rate limiting, and load balancing to keep services running smoothly.

3. **SQL Injection :**

SQL injection is a type of cyberattack that takes advantage of weaknesses in web applications by changing SQL queries. Attackers add harmful SQL code into input fields, allowing them to access databases without permission, skip login checks, or steal sensitive information. This kind of attack can cause serious security problems, such as data theft, corruption, or deletion. SQL injection usually targets poorly cleaned user inputs in login forms, search fields, or URLs. Among measures to prevent malicious code execution are parameterized queries, prepared statements, and validation of input data. Strong database security significantly

reduces the threats that SQL injection comes with, thereby securing web applications.

Prevention of Security Risks:

1. **Input Integrity Validation and Data Sanitization:**

Validation checks what users feed to the system against certain rules, like format, type, size, or range, so only proper data can enter the system. That is to say, validation ensures that the input data falls within the expected limits of valid program inputs. Inputs must, therefore, follow certain type and number-range rules, as well as fixed conditions for class or subsystem. Validation keeps data accurate and prevents bad or harmful inputs from creating security problems or system mistakes. This helps keep the application stable and reliable.

2. **Secure authentication and authorization:**

Authentication is the process of ascertaining that a given person or system is actually who they say they are. Access control is the process of checking the credentials of the accessing user with saved credentials in some database or an authentication server that verifies if only authorized users are permitted to gain entrance into systems, applications and critical information so that enterprise data and processes may be secured. Using authentication will allow organizations to keep their systems away from unauthorized access and ensure that only the real users can avail their resources.
Types of authentication:

Something you know:

This method of authentication is based on knowledge that is exclusive to the user, including elements such as passwords, PINs, or a series of security questions. It represents the earliest form of authentication and is a commonly recognized strategy for managing access control.

Password: It is a combination of letters, numbers, and special symbols used to add complexity to it and increase security[1].
PIN: A shorter six-digit number, usually accompanied with other identification or through the use of kiosks such as ATMs[1].

Something you Have:

Authentication based on something you possess depends on a physical item or a digital device that is unique to the authorized user. This factor is crucial as it establishes a tangible or digital barrier, thereby hindering unauthorized access.

1. **OTPs:** One-Time Passwords (OTPs) are time-sensitive codes that are generated every 30 to 60 seconds. This approach is typically more secure

than SMS-based methods, as it is less susceptible to SIM-swapping attacks and necessitates physical access to the device[1].

2. **Hardware Security Keys:** Hardware security keys are tangible devices, such as USB or NFC keys, that employ cryptographic protocols to securely authenticate a user's identity. They offer robust defense against phishing attacks by functioning exclusively on approved websites and necessitating physical engagement for the authentication process.

3. **Smart Cards:** A smart card, containing a memory chip or microprocessor, enables secure data management. Widely used in government and corporate settings, it supports encrypted access, allowing secure logins and PIN-protected communications.

Something you are:

Something You Are authentication depends on unique biometric features like fingerprints, facial recognition, or voice patterns for secure access.

1. **Fingerprints:**

Fingerprint verification is the automatic identification and authentication process involving intelligent algorithms to identify a user based on unique features found in fingerprint images. Fingerprints are classified under types of biometric information sued to verify identity.

2. **Facial recognition:**

Facial recognition uses distinct facial metrics, like eye distance, nose shape, and jawline, capturing an image for identity verification. It enables quick, contact-free access, functioning in low light, ideal for secure, high-traffic environments.

3. **Voice patterns:**

Voice authentication leverages unique vocal characteristics—pitch, tone, rhythm capturing speech samples to verify identity. With no need for contact or visuals, it’s ideal for secure, remote access, particularly in voice-activated environments.

Comparison of OAuth 2.0 and JWT:

OAuth 2.0	JWT
Primarily an authorization framework, OAuth 2.0 allows third-party applications to gain controlled access to user resources without exposing user credentials.	JWT is a compact, self-contained token format that securely encodes claims for authentication and information exchange between parties.
Delegated access control (e.g., social logins, access to APIs from third-party apps). Ideal for situations where user consent is required to grant access to specific resources.	Delegated access control (e.g., social logins, access to APIs from third-party apps). Ideal for situations where user consent is required to grant access to specific resources.
OAuth 2.0 is authorization-focused, allowing clients to request tokens for specific permissions through flows like Authorization Code and Client Credentials.	Primarily used for authentication, as tokens carry encoded user claims. Can serve as an access token in OAuth 2.0, but typically acts as a standalone authentication method.
OAuth 2.0 doesn't mandate a specific algorithm, but tokens are often signed using HMAC (symmetric) or RSA (asymmetric) for integrity. Commonly uses SHA-256 and other standard hashing.	JWT supports HMAC (symmetric) or RSA (asymmetric) signing for integrity and authenticity. Common algorithms include HS256 and RS256 for signing and SHA-256 for hashing.[8]
Implementation complexity can lead to vulnerabilities if flows and token handling are misconfigured. Often requires additional measures for stateless token revocation.	Inflexible revocation mechanisms due to stateless nature. Larger payloads can impact performance, especially for mobile apps with limited bandwidth.

JWT Algorithms comparison:

HS256:

HS256 is a symmetric hashing algorithm that uses the HMAC (Hash-based Message Authentication Code) construction combined with the SHA-256 hash function. It relies on a shared secret key for both signing and verifying tokens, making it fast and efficient. The algorithm is commonly used for signing JWTs and ensuring data integrity. However, the security of HS256 heavily depends on the secrecy of the shared key, which can be a

potential vulnerability if exposed. Despite this, it remains a popular choice due to its relatively simple implementation and good balance between speed and security.[9]

HS384:

HS384 operates similarly to HS256, but it uses the SHA-384 hash function instead of SHA-256, resulting in a larger output and higher security margin. This algorithm also employs a shared secret key for both signing and verification, maintaining symmetric encryption's efficiency. While providing better security than HS256, its increased output size and processing requirements lead to slightly higher execution times. HS384 is suitable for scenarios requiring a stronger hash output and additional security against potential attacks, though the reliance on the shared key introduces similar vulnerabilities. It strikes a balance between security and performance, making it a solid choice in specific use cases.

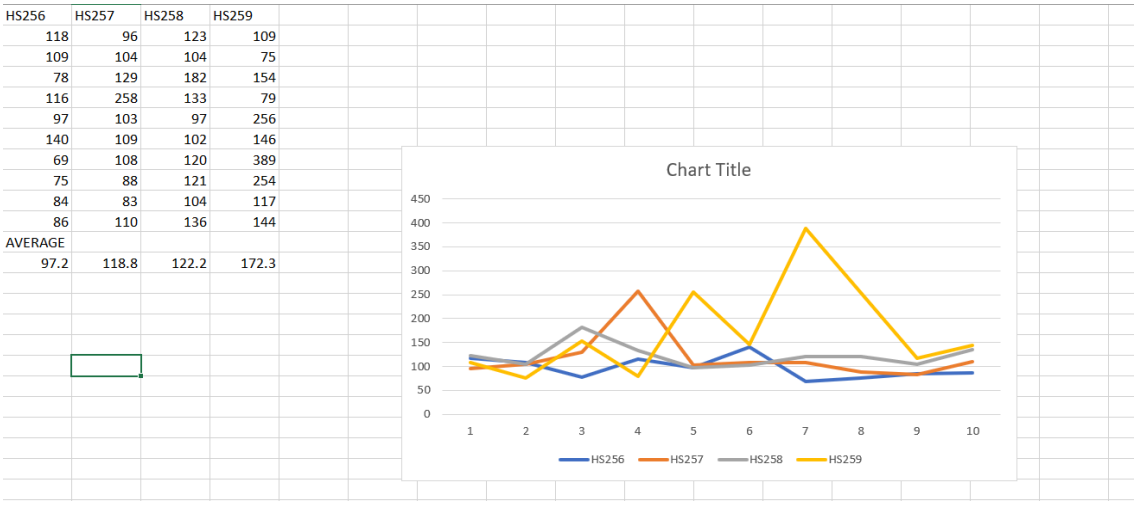
HS512:

HS512 is the most secure of the HMAC-based algorithms in the HS family, utilizing the SHA-512 hash function for even stronger security. This algorithm delivers a longer hash output, resulting in better protection against brute force and collision attacks. However, the increased complexity of the hashing function leads to slightly slower performance compared to HS256 and HS384. Like other HMAC algorithms, HS512 relies on a shared secret key, which remains a point of concern for security if the key is compromised. Despite the performance trade-offs, HS512 is ideal for high-security applications where data integrity and confidentiality are paramount.

RS256:

RS256 is an asymmetric encryption algorithm that uses the RSA algorithm for public-key cryptography, coupled with the SHA-256 hash function for signing. Unlike symmetric algorithms, RS256 relies on a private key to sign tokens and a public key for verification, ensuring that no sensitive data is exposed in the process. This separation of keys significantly enhances security, especially in distributed systems. RS256 tends to be slower than symmetric algorithms like HS256 due to the computational cost of asymmetric encryption, but its security benefits make it a preferred choice for systems that require strong authentication mechanisms. It is widely used in OAuth 2.0 and other protocols where secure, scalable, and tamper-resistant token generation is crucial.

Algorithm	HS256	HS384	HS512	RS256
Average time	97.2	118.8	122.2	172.3
Type	Symmetric	Symmetric	Symmetric	Asymmetric
Output Size	256 bits	384 bits	512 bits	256 bits
Performance	Fast	Slightly slower than HS256	Slower than HS256 and HS384	Slower due to asymmetric encryption
Key type	Shared secret key	Shared secret key	Shared secret key	Public/Private key pair
Use Cases	Small-scale	Small-scale	Small-scale	Large-scale



The chart shows the response times of four different JWT algorithms: HS256, HS257, HS258, and HS259. Ten iterations were used for this chart. HS259 shows the longest response times in all ten iterations. HS256 and HS257 show similar and shorter response times. HS258 shows a moderate increase in response time compared to the other algorithms.

Outcomes:

In my research on API performance testing using various hashing algorithms with Thunder Client, a comparative analysis was conducted under the diff-check algorithm. The results highlighted notable execution time differences among the algorithms. HS256 took 2.53 seconds, while HS384 and HS512 demonstrated significantly reduced times of 129 ms and 208 ms, respectively. Interestingly, RS256, an asymmetric cryptographic algorithm,

outperformed the others with a remarkable 63 ms execution time. Such findings emphasize that the choice of hashing algorithms plays a crucial role in optimizing API performance.

These insights align with the broader context of enhancing web application security through OAuth 2.0 and JWT. OAuth 2.0 offers delegated resource access without exposing client-side user credentials, addressing vulnerabilities related to direct credential sharing. Its flexibility, with various grant types like Authorization Code and Client Credentials, allows applications to implement fine-grained authorization controls tailored to specific access scenarios. On the other hand, JWT provides lightweight, self-contained tokens with cryptographic signing, ensuring token integrity and resistance to tampering. JWT's scalability is particularly advantageous for distributed systems and microservices, facilitating efficient session management.

Applications built on OAuth 2.0 and JWT significantly reduce inappropriate requests and demonstrate strong resilience against session hijacking, XSS, and SQL injection attacks. However, while JWT's stateless authentication model resists replay attacks through secure transmission protocols like HTTPS and short-lived tokens, it remains susceptible to token leakage. To address this, implementing mechanisms like token revocation lists and enhancing token lifecycle management is essential for highly sensitive applications. This study confirms that integrating OAuth 2.0 and JWT provides robust mechanisms for authentication and authorization, substantially bolstering web application security.

Conclusion

This paper provided an in-depth analysis of cryptographic algorithms (HS256, HS384, HS512, and RS256) in the context of API security and web application development. The performance testing revealed a clear distinction in execution times, with asymmetric algorithms like RS256 offering faster response times and superior security compared to symmetric algorithms. The study also highlighted the integral role of OAuth 2.0 and JWT in securing modern web applications, focusing on their effectiveness in authentication, authorization, and securing user data.

OAuth 2.0’s ability to securely delegate access without exposing credentials, coupled with JWT’s lightweight, tamper-resistant tokens, enables efficient and scalable security mechanisms. The findings suggest that combining these technologies reduces vulnerabilities, such as session hijacking, XSS, and SQL injection attacks, thereby enhancing overall web application security. However, token leakage remains a concern in stateless authentication systems, requiring careful implementation of token expiration and revocation mechanisms for optimal protection.

Overall, this research confirms that OAuth 2.0 and JWT, when used alongside suitable cryptographic algorithms, significantly improve both security and performance in web applications. The adoption of these technologies facilitates a scalable, robust, and secure i environment, making them essential components of modern application architectures.

References:

[1]. Ahmet Bucko, Kamer Vishi, Bujar Krasniqi,Blerim Rexha Article Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History, pp 3-5, 2023

[2]. Nagarjun, PMD; Shaik, Shakeel Ahamad, “Cross-site Scripting Research: A Review” International Journal of Advanced Computer Science and Applications; West Yorkshire, pp 626-627,2020

[3]. Yjvesa Balaj, Faculty of Electrical and Computer Engineering Prishtina, Kosovo, Token Based vs Session Based Authentication, PP 1, Sep 2017

[5]. Hardt, D.: “The OAuth 2.0 Authorization Framework” RFC 6749, RFC Editor, October 2012

[6]. Jones, M.B., Hardt, D.:”The OAuth 2.0 Authorization Framework: Bearer Token Usage.” RFC 6750, RFC Editor, October 2012

[7] Sungchul Lee, Ju-Yeon Yo,Yoohwan Kim “Authentication System for Stateless RESTful WebService”, originally published in November 2016 in Computer Science, vol 10018. Springer, Cham

[8] Jones, M.; Bradley, J.; Sakimura, N. JSON Web Token (JWT), RFC 7519, 2015. Available online: http://www.rfc-editor.org/info/rfc7519 (accessed on 23 June 2017).

[9] Manish Rana,Ayush pandey, Ankit mishra A Comprehensive Study on the Efficacy of JSON Web Token (JWT) and HMAC SHA-256 Algorithm for Web Application Security pp 4414-4415 september 2024.

[10]R. Priyatna and S. Waluyo, “Implementasi Restful Dengan Jwt Untuk Booking Barang Di Primajaya Multisindo,” Semin. Nas. Mhs. Fak. Teknol. Inf. Jakarta-Indonesia, no. September, pp. 1040–1047, 2022.