International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 07 Issue: 06 | June - 2023 SJIF Rating: 8.176 ISSN: 2582-3930

COMPARATIVE STUDY ON XML ENCRYPTION AND JSON ENCRYPTION

Aiswarya Jayadev (IMCA-194)
PgScholar, Department of Computer Applications SCMS School of Technology and Management
Kochi, India
IMCA-194@scmsgroup.org

Ms. Jisha Liju Daniel
Assistant Professor, Department of Computer Applications
SCMS School of Technology and Management
Kochi, India
jishaliju@scmsgroup.org

***Abstract*— JSON, which stands for JavaScript Object Notation, is a text-based format commonly used for storing and transmitting data. JSON encryption, also known as JSON web encryption (JWE), is an official standard established by the Internet Engineering Task Force (IETF). It defines a syntax for the secure exchange of encrypted data. XML, an acronym for Extensible Markup Language, is a markup language like HTML. XML Encryption, governed by recommendations from the World Wide Web Consortium (W3C), provides specifications for encrypting the contents of XML elements. The primary objective of this study is to conduct a comprehensive comparison between these two technologies to determine their relative merits. Furthermore, it aims to delve into the details of each technology to gain a deeper understanding.**

Keywords—; JSON; XML;JSON ENCRYPTION;XML ENCRY[TION;COMPARION TABLE.

I. INTRODUCTION

JSON (JavaScript Object Notation) is a widely adopted standard format that uses JavaScript object syntax to represent structured data in a textual form. Its primary purpose is to facilitate data transmission in web applications. For instance, JSON enables the exchange of data between the server and client, allowing the information to be displayed on web pages or vice versa. On the other hand, XML (Extensible Markup Language) is a straightforward text-based format designed to represent various forms of structured information. It encompasses documents, data, configurations, books, transactions, invoices, and much more. Originally derived from the SGML (ISO 8879) standard, XML was tailored to be more suitable for web usage. Today, XML stands as one of the most prevalent formats for sharing structured information across programs, individuals, computers, and networks, both locally and globally.

II. JAVASCRIPT OBJECT NOTATION

[5] JSON, which stands for JavaScript Object Notation, has emerged as a widely adopted standard for seamless exchange of JavaScript object data between different systems. It provides a convenient way to convert JavaScript objects into the JSON format and transmit them as JSON text to servers. When a recipient requests JSON text, it is received from the server and can be converted back into JavaScript objects. The JSON format represents data in the form of key-value pairs, and its simplicity has contributed to its popularity as a preferred data exchange format in numerous programming languages.

JSON is particularly utilized for asynchronous communication between web browsers (clients) and servers. In addition to objects, JSON also supports storing numbers, strings, Booleans, arrays, and null values.

III. XML

[5] In the field of computing, XML (Extensible Markup Language) documents have witnessed increasing popularity due to their ability to transmit data in a structured format through networks or portable storage devices. XML is an open standard language that offers mechanisms for sending, receiving, and storing data, which are independent of both hardware and software. XML files are known for their simplicity, ease of understanding, and versatility. They provide a more streamlined and general approach compared to other formats, resulting in faster processing. Furthermore, XML employs an encoding format that is straightforward for machines to interpret, while remaining human-readable.

IV. COMPARATIVE ANALYSIS BETWEEN JSON AND XML

XML	JSON
Relatively difficult for machines to parse.	Relatively easy for machines to parse.
Uses tags to define data, resulting in increased payload size.	Uses key-value pairs to define data, resulting in lesser payload size and better indexing.
Moderate performance in data exchange in web service applications.	High performance in data exchange.
Does not support data types and arrays.	Includes support for data types and arrays.
Not as lightweight and fast as JSON.	Faster and lighter compared to XML.
Focuses on the structural component of data as it is document oriented.	Data-oriented and considered as a more concise and streamlined version of XML.

Table 1. Comparison between JSON and XML

V. JSON WEB ENCRYPTION

[11] By default, JSON Web Tokens (JWTs) are encoded using base64url and represented as JSON objects. They are signed using a digital signing algorithm provided by JSON Web Signatures (JWS). JWS ensures integrity, authentication, and non-repudiation, but it does not provide confidentiality. This can be a concern if the token contains sensitive data, as anyone can read the payload.

[6] To address this issue, JSON Web Encryption (JWE) comes into play. JWE allows you to encrypt the payload of a JWT, ensuring that only the intended recipient can decrypt and access it. Along with providing integrity and authentication checks, JWE adds an extra layer of security. Combining JWE with JWS creates an encrypted token that is suitable for use as an access token or identity token in OpenID Connect.

The most commonly used format for encrypted JWTs is JWE Compact Serialization, which is more popular than the less common JWE JSON Serialization. JSON Web Encryption (JWE) utilizes JSON-based data structures to represent the encrypted content. [7] The specification for cryptographic algorithms and identifiers used in JWE is described separately in the JSON Web Algorithms (JWA) specification, along with the IANA registries defined by that specification. The JSON Web Signature (JWS) specification covers the related digital signature and Message Authentication Code (MAC) capabilities.

A. JSON Object Signing and Encryption (JOSE)

JSON Object Signing and Encryption (JOSE) refers to a standardized set of software technologies introduced by the IETF (Internet Engineering Task Force). These technologies are designed to represent encrypted and/or signed content in the form of JSON (JavaScript Object Notation) data. The key components of JOSE include JSON Web Signatures (JWS), JSON Web Encryption (JWE), JSON Web Key (JWK), and JSON Web Algorithms (JWA). Together, these technologies form a framework that enables secure transfer of claims, such as authorization information, between different parties. The primary objective of JOSE is to establish a reliable method for securely exchanging data using JSON-based representations.

B. Attacks

[4] Ethernet is a widely adopted method for connecting computers or nodes within a Local Area Network (LAN). It has been the predominant approach for interconnecting computers within LAN environments. The fundamental concept behind its design is to allow multiple computers to have simultaneous access to the network, enabling them to send data at any given time. Ethernet provides a reliable and efficient means of communication within a LAN, facilitating the exchange of information between connected devices.

- JWT Attacks:

  JWT attacks involve a user manipulating or modifying JSON Web Tokens (JWTs) and sending them to the server with malicious intent. The primary objective of such attacks is often to bypass authentication and access controls by impersonating another user who has already been authenticated. By tampering with the contents of a JWT, an attacker can potentially gain unauthorized access to protected resources or perform actions on behalf of another user. The impact of JWT attacks can be severe as they undermine the integrity and security of the authentication mechanism. It can lead to unauthorized access, data breaches, and compromised systems, posing significant risks to the affected individuals or organizations.

- JSON Injection:

  JSON injection is a vulnerability that arises when data originating from an untrusted source is not properly sanitized by the server and is directly incorporated into a JSON stream. This type of vulnerability is known as server-side JSON injection. Similarly, client-side JSON injection occurs when data from an untrusted source is not appropriately sanitized and is directly parsed using the JavaScript eval function.

  In server-side JSON injection, if the server fails to sanitize or validate the input before including it in the JSON stream, an attacker can inject malicious content that may lead to various security issues. This can include executing arbitrary code, modifying data, or accessing unauthorized information.

  On the other hand, client-side JSON injection occurs when the client-side JavaScript code utilizes the eval function to directly parse JSON data received from an untrusted source. If the data is not properly validated or sanitized before being evaluated, it can enable an attacker to execute arbitrary code on the client-side, leading to security risks.

  Both server-side and client-side JSON injection vulnerabilities can have serious consequences, such as data breaches, unauthorized access, and potential manipulation of the system. To mitigate these vulnerabilities, it is crucial to implement proper input validation and sanitization techniques to ensure the integrity and security of JSON data handling.

C. UTF Support in JSON.

UTF-8 is a character encoding system designed for representing Unicode characters. It allows the translation of any Unicode character into a corresponding unique binary string and vice versa. The name "UTF" stands for "Unicode Transformation Format." In contrast, ASCII (American Standard Code for Information Interchange) is an encoding system that represents characters using seven bits of binary data. With seven bits, there are 128 possible combinations, allowing for the representation of a limited set of characters. Each binary number in the ASCII encoding can be converted to a decimal number ranging from 0 to 127.For example, the binary number 1000001 in ASCII corresponds to the decimal number 65, which represents the uppercase letter "A" in the ASCII character set.

VI. XML ENCRYPTION

[1] The Extensible Markup Language (XML) is a text-based format that is used to represent structured information, such as documents, data, configuration settings, books, transactions, invoices, and more. It was developed as a successor to the Standard Generalized Markup Language (SGML) with the aim of being more suitable for web usage. If you are already familiar with HTML, you will find XML to be quite similar. However, XML has stricter syntax rules compared to HTML. XML tools and parsers are designed to detect and report any errors in XML files, ensuring that they conform to the specified syntax. This allows computer software to process XML documents reliably. XML Encryption employs two encryption algorithms, namely AES-CBC and RSA PKCS1. It is commonly used in web service penetration testing frameworks, such as WS-Attackers. These frameworks automatically analyze web service interfaces and execute

attacks specifically targeting XML Encryption vulnerabilities. The purpose of XML encryption is to provide data confidentiality for transmitted messages. It allows you to encrypt an entire message or selectively encrypt specific elements within the message. However, it's important to note that the use of XML encryption, either independently or in conjunction with XML digital signatures, can introduce potential security risks and implications. XML Signature, on the other hand, defines the syntax and processing rules for creating, representing, and verifying digital signatures in XML-based data. The flexibility of the XML Signature specification allows for the signing of various types of digital data, ensuring their integrity and authenticity.

A. *XML Signature*

[8] An XML digital signature, also known as XML DSIG, is an electronic cryptographic stamp that provides authentication and integrity assurance for digital information, including messages. The purpose of a digital signature is to verify that the information originated from the signer and that it remained unchanged during transmission. When using public key cryptography in digital signatures, the identity of the sender can be traced, satisfying non-repudiation requirements.XML digital signatures can be used independently or in conjunction with XML encryption. However, it's important to note that the use of XML digital signatures, either separately or combined with encryption, can introduce potential security implications. Careful implementation and adherence to security best practices are necessary to mitigate these risks effectively. Compared to XML digital signatures, XML encryption generally has fewer potential security issues. XML encryption primarily focuses on ensuring data confidentiality, whereas XML digital signatures are concerned with authentication and integrity. However, both technologies play crucial roles in securing XML- based data and should be implemented with proper consideration of the associated security implications.

B. *Attacks*

- XML External Entity attack.

  An XML External Entity (XXE) attack is a type of security vulnerability that targets applications parsing XML input. This attack takes advantage of a weakly configured XML parser when processing XML input that includes a reference to an external entity. If the XML parser is not properly configured or hardened, an attacker can exploit this vulnerability to carry out various malicious activities.

- XML Bombs (Denial of Service).

  XML Bomb Attack, also known as Billion Laughs Attack, is a specific type of Denial-of-Service (DoS) attack that targets XML parsers. It exploits the vulnerability in XML parsing by sending a small but maliciously crafted XML file that contains nested data entities. When the XML parser attempts to process the file, the nested entities start expanding exponentially, consuming excessive memory and computational resources. This can lead to the crash or slowdown of the server or application. It is important to note that XML Bomb Attacks are not limited to crashing servers that parse XML files; they can also affect other systems or applications that process XML data. Regarding UTF-16 encoding, it represents Unicode characters using two or four bytes, depending on the character's code point. UTF-16 uses the distinction between two-byte and four-byte representations to accommodate the full range of Unicode characters. On the other hand, UTF-8 encoding uses a variable-length format, where the size of the encoded character can vary from one to four bytes. In UTF-8, ASCII characters are represented by a single byte (eight bits), while other characters may require multiple bytes based on their code points. Both UTF-8 and UTF-16 are widely used encoding schemes for representing Unicode characters, each with its own advantages and considerations depending on the requirements of the system or application.

C. *UTF Support in XML*

Supports UTF-8 and UTF-16 encodings.UTF-8 encodes a character into a binary string of one, two, three, or four bytes.

VII. COMPARISON BETWEEN JSON ENCRYPTION AND XML ENCRYPTION

	JSON ENCRYPTION	XML ENCRYPTION
DETECTED ATTACK	JWT attacks, JSON injection	XML External Entity attack. XML Bombs (Denial of Service).
KEY	Public key and Private Key	Public key and Private Key
DATA STORAGE	Data is stored like a map with key- value pairs in JSON format.	data is stored as a tree structure.
SPEED	Fast	Slow compared to JSON
ALGORITHM	AES (Advanced Encryption Standard), Recursive, Translating	AES-CBC and RSA-PKCS
ENCRYPTION AND DECRYPTION TECHNIQUE	JSON Object Signing and Encryption (JOSE)	XML Signature

Table 2. Comparison between JSON encryption and XML encryption

VIII. CONCLUSION

The choice between JSON encryption and XML encryption depends on the specific use case and requirements. Both formats can be encrypted using standard encryption algorithms such as AES or RSA, and both have their advantages and disadvantages.

JSON is a lightweight and easy-to-read format that is commonly used for data exchange in web applications. It is also supported by most programming languages and platforms. JSON encryption is generally faster than XML.encryption because it has a smaller payload, making it more suitable for real-time data exchange.

On the other hand, XML is a more flexible and powerful format that allows for more complex data structures and metadata. XML encryption is often used in enterprise systems that require high security and compliance with industry standards such as XML Encryption Syntax and Processing.

Ultimately, the choice between JSON and XML encryption will depend on the specific requirements of your application. If you need to exchange real-time data quickly and easily, JSON encryption may be the better choice. If you require complex data structures and metadata, or need to comply with industry standards, XML encryption may be more appropriate.

REFERENCES

[1] Gaurav Goyal,Karanjit Singh,K R Ramkumar ,"A detailed analysis of data consistency concepts in data exchange formats (JSON & XML)",2017,IEEE

[2] Munir Šabanović , Muzafer Saračević , Emruš Azizović ," Comparative analysis of AMF, JSON and XML technologies for data transfer between the server and the client ",2016

Gaurav Goyal,Kamal Deep Garg,Rupali Gill," A Key based Distributed Approach for Data Integrity and Consistency in JSON and XML(Hierarchical Data Exchange Formats)",2020,IEE

[3] Dennis Detering , Juraj Somorovsky , Christian Mainka ," On The (In-)Security Of JavaScript Object ", November 2017 Article No.: 3Pages 1–11

[4] Ms. Boda Jagruti , Ms. Patel Nidhi , Prof. Dhatri Pandya ,"A Survey on Webservice Security Techniques ",ICCA ,2018

[5] M.Jones,"JSON Web Encryption(JWE)",May 2015 [7]M.jones,"JSON Web Algorithms(JWA)",May201

[6] Juraj Somorovsky* , On the insecurity of XML Security , DE GRUYTER OLDENBOURG it – Information Technology 2014; 56(6): 313–317