An authentication based scheme for applications using JSON web token

Salman Ahmed
Department of Computer Science
Capital University of Science and technology
Islamabad, Pakistan
salman.ahmed@cust.edu.pk

Qamar Mahmood
Department of Computer Science
Capital University of Science and technology
Islamabad, Pakistan
qamar@cust.edu.pk

***Abstract*—Information and Communication Technology (ICT) encompasses many devices and applications that are inter linked with each other on internet for communication. Development of client server applications on technologies, like IoT based, cloud based, and smart homes systems are growing rapidly in the current technology era. User authentication is important concern in these applications. REST (Representational State Transfer) based architecture with JWT emerged enormously in recent development of client server applications. JWTs (JSON Web Token) is used for authentication of subsequent client requests without making frequent calls to the resource server or database. In this paper, we present an authentication technique for regeneration of JWT on each client request based on truly random timestamp values to enhance the authenticity of client on server. The working of proposed technique is elaborated with a case study which indicates the stronger authentication and security.**

Keywords—Authentication, Security, Client Authentication, JSON Web Token, REST call authentication

I. INTRODUCTION

User authentication and access management in Internet of Things (IoT)[1], mobile[2], web[3] and cloud based[4] applications is deemed as an important concern in terms of network security. In client server communication, servers maintain the identity of clients during interaction of clients with server. If appropriate security measures are not applied to protect the identity of a client, then it could lead to mis-usage of data to a great extent. Authentic access of data becomes more complex when a same user communicates with server from different devices using different platform like iOS, Android, Windows and Linux, such as mobile, tab, personal computer etc.

A JSON Web Token (JWT) is a JSON object that is defined in RFC 7519 [6] a secure method to represent a set of information between two parties. JWT has been used in various studies presented in the literature to maintain the authenticity of clients while interaction with server. Smart Home Environment Applications [5], Cloud SaaS Applications [7], Smartphone Management Applications [8] are using JWT mechanism for client authentication for utilizing server resources or accessing devices on IoT platform. These approaches have some authentication vulnerabilities which can be misused by attackers in order to again access to server resources. These approaches have a common vulnerability of using same token until the expiry of JWT or a user performs logout operation.

In this study, we present an approach which focuses on securing a server from JWT prediction attack during client and server interaction and to distinguish between valid and invalid client requests after user role has been modified by the super admin of application. Our proposed technique uses JWT based authenticity mechanisms [6] by reducing chance of vulnerability via enhancing security of client identification on sever. A case study is executed in which fresh JWT is generated based on each client request and server response timestamp. Addition of these two timestamps will generate random values which cannot easily be predicted by attackers. The proposed technique contributes to eliminate the prediction of JWT by attacker, preimage attack, and user role change problem vulnerabilities in different applications of IoT and cloud based systems.

II. USER AUTHENTICATION SCHEMES

A. Traditional Approach

The traditional methods maintain a session on server for each client for client authenticity on server. As long as the session is valid, a client sends a request and server responses to the request using a session key. Session keys are assigned to clients on first request to server after credentials verification from database server. A client maintains the assigned key in cookie for further interaction with server. Client sends same session key to server for authentication with each request. Sever will only check a valid session key coming from client for authenticity of client instead asking the credentials again. The weakness of this mechanism is easy hijacking of session keys by attackers. Attackers can use different tools for brute-force session key identification and gain access of current user sessions. Many tools and plugins are available which can transmit cookies data and keys to third party resources. The authors [9] present an approach that uses role-based access controls (RBACs) and Web session management to protect against network security breaches in the HTTP environment. SSL is used to reduce chance of hijacking of session keys. The CAPEC (Common Attack Pattern Enumeration and Classification) schemes are helping to resolve Session Fixation weaknesses [10].

B. JWT based authentication in applications

In-depth analysis of literature depicted that second mostly used approach for identification of client at server side is JSON Web Tokens (JWT). A JSON Web Token (JWT) is a JSON object that is defined in RFC 7519 [6] as a secure method to represent a set of information between two parties. Smart Home Environment Applications [5], Cloud SaaS Applications [8], Smartphone Management Applications [9] are using JWT mechanism for client authentication for utilizing server resources or accessing devices on IoT platform. These approaches have some authentication vulnerabilities which can be misused by attackers in order to again access of server resources. A common vulnerability that exist in these approaches is usage of a same token until expiry of JWT or a user performs logout operation.

The JWT is a string composed of a header, a payload, and a signature.

978-1-7281-4001-8/19/$31.00 ©2019 IEEE

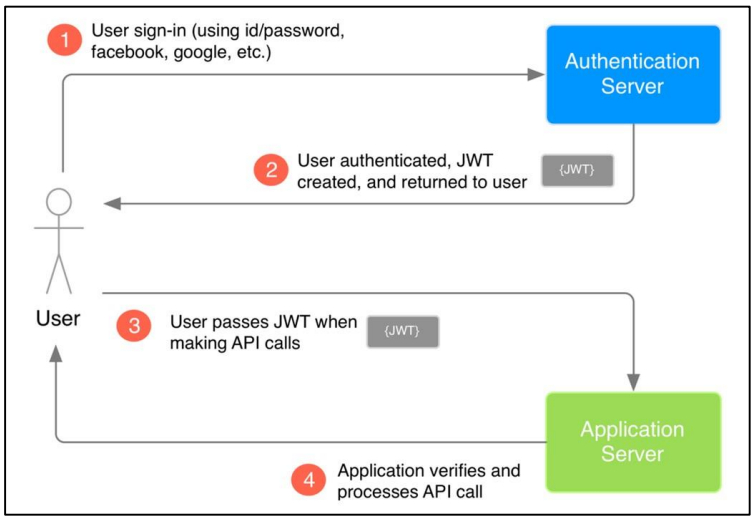

Fig. 1. How JWT work [11]

1) JWT header contains information about how the JWT signature should be computed. The "alg" in header represents any hash algorithm for formation of JWT signature.

{"typ":"JWT","alg":"HS256"}

2) JWT payload contains data stored inside the JWT which could be a user id or email or anything.

Payload = {"userId": "salman.ahmed@cust.edu.pk"}

3) JWT signature contains the signature of header and payload which is computed via following algorithm.

data = header + "." + payload
hashedData = hash(data)
signature = base64urlEncode(hashedData)

The basic JWT mechanism performs the following steps for authentic client recognition on server, as shown in Fig 1.

1) The client sends an authentication request by credentials, such as a username– password combination for login via username and password, Facebook login, Google login, etc.

2) On verifying the credentials of a user, the authentication service creates a JWT from the retrieved user authorization data. The created JWT retured to client for futher requests to the secured resources on server.

3) The client useses this token with the request to a secured resource on server.

4) The server receives the request, unpacks the user authentication data from JWT for recognation of authentic request. Server will send a proper response based on valid or invalid token.

JWT is a scalable solution with significant performance benefits for user access control in decentralized, large-scale distributed systems [11]. The key feature of JWT is that it allows the transition of state to the communication, thus a client cannot change the information contained in the token. The drawback of using this conventional approach is that server uses same JWT for the all requests until user logs out from system. Various approaches have been presented to overcome the said issue, but their scope is limited to specific applications. There is a need to propose a generalized solution for all type of applications. The generalized solution should secure JWT to eliminate vulnerability in client authentication.

III. Problem Statment

By considering JWT based authentication, this study focuses on the following problems.

1) User role update causes a significant loss after assigning the token to client. It can be misused to exploit the permissions user granted. A secured resourse on serve should not be accesable to the users, whose permessions are revoked.

2) Access token (JWT) remain same in most of the request and response between client and server interaction. Attacher can predict this vulnerability to misuse JWT.

IV. Proposed Solution

To solve above mentioned problems, a new and unique token is required with each client request on server for authentic client server interaction. This will make the attacker harder to predict the access key for misuse. On the other hand if role of a user is updated it will be reflected or predicted by server for any vulnerability in the very next request. In normal scenario, clients cannot force to forgot JWT old token and use the new token.

To generate a new JWT access token on each request CRT and SRT will be used. CRT is *Client Request Time* and SRT stands for *Server Response Time*. We will add these time stamps to create a random value T. Timestamp T is generated based on truly random values so it is harder to predict for attacker in a specific user session. To make the T more harder to guess for an attacker, a random integer value is added into the T on the server side to generate a new time `T. The value of `T will be placed in the payload of JWT format. When the signature is calculated it will always generate a unique random access token (JWT), because of CRT and SRT uniqueness. This token will be sent back to client in response against each client request.

Fig 2 explains step by step process of generation a new token on each request. This process will enhance the authenticity of a client on sever, which is the solution of our second problem statement.

Step 1. Client sends a login https request to server with CRT and login credentials.

Step 2. Server receives https request and verifies credentials from database server. If credentials verified go to step 3 otherwise move to step1.

Step 3. Server will generate JWT by combining CRT, SRT and a rand value. Note: The value of `T is placed in payload section of JWT.

Step 4. Server will save email from credentials, role as 1, SRT and Rand value from system against JWT.

Step 5. Server sent this JWT, `CRT and a login success message to client as a response. Note: `CRT value is same as CRT in request, it will change in next request from same client.

Step 6. Client will ask a query from server with `CRT, JWT and new CRT for new request.

Step 7. Server will receive the request and access the SRT and rand value against received JWT. Server will also check status of role status change. If role is 1, the server will generate `JWT using `CRT, SRT, and the rand value.

Step 8. If the JWT and `JWT is not matched or role is 0 server will reject the request and move to step1.

Step 9. If the JWT and `JWT is matched then server will accept the request to answer query and move to setp3.

For the solution of first problem statement, database server will generate trigger for JWT server to make role status of specific email as 0 in case of role of user is changed by admin of system. Verifying the role status on each client request is a performance overhead but from the indexed table, information of valid role can be retrieved the with time complexity of $O(1)$.

V. Benefit of Time Based JWT Generation

The primary advantages of the proposed approach are presented below:

1) Two different countries time stamps will be added when client and server are in different regions. It will be difficult for attacker to find the value of T.

2) Change of token on each request will be a barrier for brute force attack in this minimum time slot between two requests.

3) On new token generation recent user role will be picked to avoid further secured resource access.

4) Generation of unique JWT signature in each request will resist preimage attacks. The generation of unique signature depends on the `T value placed in payload section of JWT. The value of `T is turly random and unique as we discussed in setion IV.

VI. Conclution

In this study, we have presented an enhancement in the JWT access control solutions for different applications developed on the platforms, such as IoT and cloud systems. Rapid generation of fresh tokens on each client request with a negligible time complexity of $O(1)$ on server can resist attacker to identify the signature of a client. In future we will implement a framework which will be used of application developers for securing the server resources from unauthenticated requests.

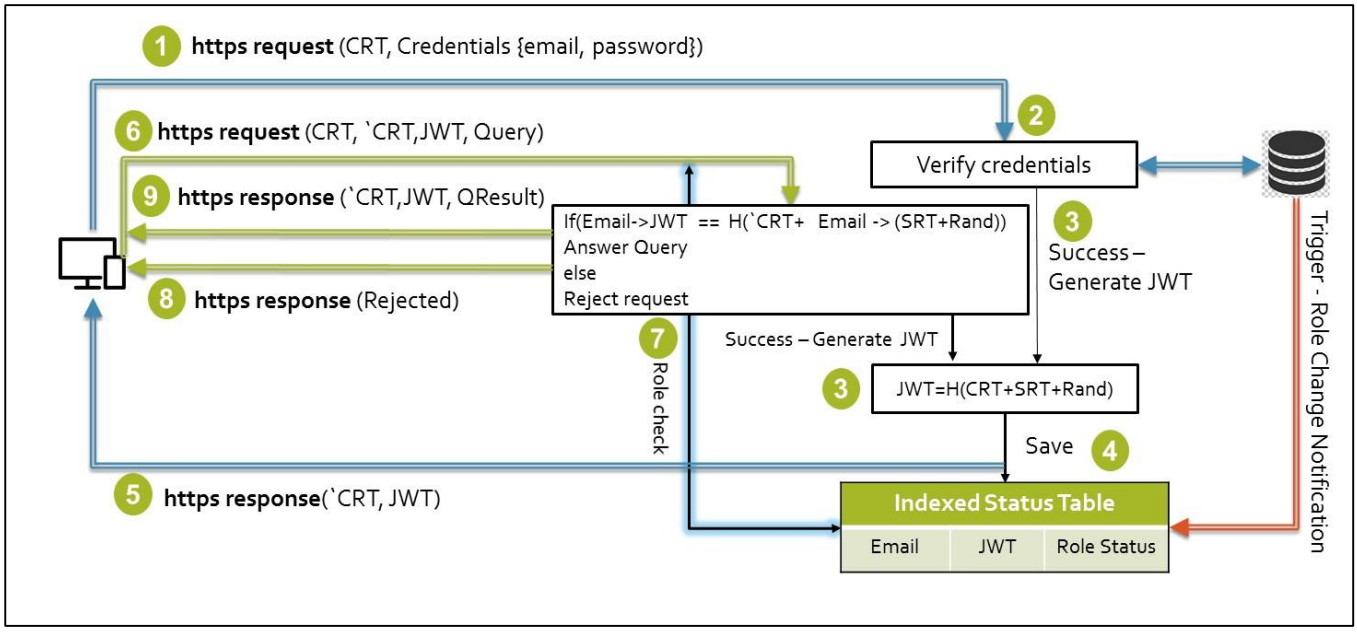

Fig. 2. How proposed technique works using JWT

References

[1] P. Kumar, A. Gurtov, J. Iinatti, M. Ylianttila and M. Sain, "Lightweight and Secure Session-Key Establishment Scheme in Smart Home Environments," in IEEE Sensors Journal, vol. 16, no. 1, pp. 254-264, Jan.1, 2016. doi: 10.1109/JSEN.2015.2475298.

[2] A. Janardanan, C. Ajil Paul, P. Anju, V. Eldiva Thomas and D. Davis, "Android Application for Car Wash Services," 2018 International Conference on Emerging Trends and Innovations In

Engineering And Technological Research (ICETIETR), Ernakulam, 2018, pp. 1-3. doi: 10.1109/ICETIETR.2018.8529025

[3] Z. Liu and B. Gupta "Study of Secured Full-Stack Web Development," Proceedings of 34th International Conference on Computers and Their Applications, vol. 58, pp. 317-324, 2019 doi: 10.29007/jpj6

[4] O. Ethelbert, F. F. Moghaddam, P. Wieder and R. Yahyapour, "A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications," 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, 2017, pp. 47-53. doi: 10.1109/FiCloud.2017.29

[5] N. Hong, M.Kim, M. Jun, J. Kang, "A Study on a JWT-Based User Authentication and API Assessment Scheme Using IMEI in a Smart Home Environment," in jornal of sustainability, vol. 9, no. 7, June 2017

[6] M. Jones, J. Bradley, and N. Sakimura, “JSON Web Token (JWT) RFC 7519”, http://www.rfc-edi tor.org/rfc/rfc7519.txt, RFC Editor 2015.

[7] O. Ethelbert, F. F. Moghaddam, P. Wieder and R. Yahyapour, "A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications," 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), Prague, 2017, pp. 47-53. doi: 10.1109/FiCloud.2017.29

[8] B. Chifor, S. Arseni, I. Matei and I. Bica, "Security-Oriented Framework for Internet of Things Smart-Home Applications," 2019 22nd International Conference on Control Systems and Computer Science (CSCS), Bucharest, Romania, 2019, pp. 146-153. doi: 10.1109/CSCS.2019.00033

[9] K. Gutzmann, "Access control and session management in the HTTP environment," in IEEE Internet Computing, vol. 5, no. 1, pp. 26-35, Jan.-Feb. 2001. doi: 10.1109/4236.895139

[10] X.Yuan, E. Borkor, S. Beal, H. Yu. "Retrieving relevant CAPEC attack patterns for secure software development," In Proceedings of the 9th Annual Cyber and Information Security Research Conference (CISR '14), ACM, pp. 33-36, 2014 doi: https://doi.org/10.1145/2602087.2602092

[11] L. Viktor Jánoky, J. Levendovszky, P. Ekler, "An analysis on the revoking mechanisms for JSON Web Tokens," International Journal of Distributed Sensor Networks, vol. 14, September 2018, doi: https://doi.org/10.1177/1550147718801535