International Journal of Research in Advent Technology, Vol.7, No.6S, June 2019
E-ISSN: 2321-9637
Available online at www.ijrat.org

Survey on Authentication Techniques for Web Applications

Mahith Madwesh, Sandeep Varma Nadimpalli

Abstract—Every application which has client and server, which has sensitive data security and privacy Is the major concern. This means any other user than authorized user can access the data no matter what happens and why. There are great deal of Authentication strategies which are created and utilized over years. Most frameworks use Cookies and sessions which make servers state full which gives obstructions for scaling and different issues. In this paper, survey on various previous authentication techniques with their advantages and disadvantages are done. Also a new authentication scheme Json Web Token(JWT) introduced by OAuth2.0 framework is also discussed and why to adopt this method other than the ones above mentioned are the key elements of the paper.

IndexTerms— authentication, authorization, OAuth, security, privacy, Json web Token(JWT).

I. INTRODUCTION

Technologies over the last few years have become very important in every company, educational institute, healthcare and everywhere else. For every application an account of the user needs to be created for storing user information, account information, transaction history and many more. Therefore a mechanism is needed to authenticate and authorize the user to access only their data and to prevent unauthorized access. In real world passport or identity card is used for authentication, but in web application whole new mechanisms are required.

Authentication in Information Technology is process of verifying who you are, are you the one pretend to be or not.

Regardless of the authentication method being used, there are several factors present that need to be taken care of, First, we need to have a particular or group of people to be authenticated. Next we need distinguishing characteristics that differentiate from others. Third, Mechanism to do the above mentioned from others and other systems. Fourth, we need a confirmation instrument to check the nearness of the distinctive trademark. Fifth, we concede some benefit when the verification prevails by utilizing an entrance control instrument and a similar system denies the benefit if confirmation falls flat.

Authentication is needed when API of an application is trying access located on a different server. It could be bank trying to complete a transaction on behalf of the client, or another third party application trying to access certain client information.

Authentication is additionally required for single sign-on a great deal and is the principle key for it since it manages client information. So as to get client acknowledgment token based confirmation is finished with Json Web Token(JWT) with customer based application for neighborhood and remote validation.

Present days websites like Amazon, eBay, Ali Express and a lot more would not be present at all if authentication did not work correctly.

For addressing the above issues mentioned, in this paper a complete survey on previous authentication techniques and present JWT scheme is discussed. Also a possible implementation with Json Web Encryption which serves both authentication and authorization is discussed.

II. LITERATURE SURVEY

This section provides a literature survey about the various authentication techniques used and security issues and challenges of Web applications.

Balaji has provided an excellent information about the basics of authentication. In this article he has explored several key concepts by providing complete evolution information examples which can were developed and the author has also explained about the merits and demerits of each method [1].

Miroslav Milinovi, Mijo Derek, Dubravko Penezi, Denis Stancer, Dubravko Voneina have discussed about the concerns of the consumers with respect to authentication. According to them security issues is one of the major reasons for large enterprises for web applicaions. Authors have provided their detailed analysis on authentication, authorization, identity management. They have also suggested some of the possible solutions for these issues [2].

Chris Sevilleja [3] stated complete authentication of API with Json Web Token(JWT). The author has explained the proposed scheme which ensures the correctness of user's data.

On jwt.io site [4] jwt has described in details with which alogorithm is used., contents of the JWT Token and how scalability is the benefits of using JWT as authentication for the web applications.

Arie, Nyoman and Linawati [5] has presented about the

Manuscript revised May 13, 2019 and published on June 5, 2019
***Mahith Madwesh**, Department of Information Science and Engineering, BMS College of Engineering, Bengaluru, India*
***Sandeep Varma** Nadimpalli, Department of Information Science and Engineering, BMS College of Engineering, Bengaluru, India*

doi: 10.32622/ijrat.76S201923

single sign-on systems using JWT for information system dashboard for government.

III. AUTHENTICATION TECHNIQUES WITH SECURITY ISSUES

Verification is the way toward deciding if a person or thing is, truth be told, who or what it proclaims itself to be. It is critical on the grounds that it empowers associations to keep their systems secure by allowing just validated clients (or procedures) to get to its ensured assets.

The first method of Authentication that was introduced was HTTP Basic Authentication, where client sends solicitation for a site page that requires Basic validation, the server reacts with a response that contains a 'WWW-verify' quality in its header. The client at that point enters a username and secret passphrase, which is sent to the server in Base64-encoded form. The fundamental disadvantages are: Base64 isn't an encryption however straightforward encoding, This can without much of a stretch be blocked and decoded; Basic verification utilizes content records to store usernames and passwords; There is no help for a Logout include. The server's character can't be verified.

After some time as an Improvement to the basic method, HTTP digest verification was introduced which is like the Basic yet is more grounded as it utilizes 'hashes' while transmitting username and secret word to the server. Server sends back a reaction with a 'WWW-confirm' characteristic in the header and a 'nonce'. A 'nonce' is a string, which varies for each attempt. The customer utilizes a progression of hashes that include the username and secret key, the mentioned URL, the verification domain name and nonce, and sends the solicitation once more. Favorable circumstances are: The secret word is hashed with the dynamic nonce esteem; consequently shielding it in transmission and from replay assaults; The secret phrase can be put away on the server as a hash rather than as clear content; The server can likewise store the hash of the secret word alongside the nonce; This framework had real hindrances, for example, Although more grounded than Basic confirmation, it is powerless against a man in the middle assaults. A man in the middle assailant can trap customers to utilize Basic verification or use a digest in an inheritance decreased security mode; secret key must be put away in a content record; there is no direct method to log out the client; Digest validation likewise doesn't confirm the server identity.

Windows Integrated Authentication in the past known as NTLM validation or NTLAN Manager is verification conspire from Microsoft for a Windows dependent organization. NTLM is a challenge– reaction plot that utilizes a Cyclic Redundancy Check or message digest calculations. The Windows Integrated plan underpins NTLM and Kerberos. It is most appropriate for intranet applications that utilization dynamic index too IIS and IE program. In a Windows situation, the secret phrase isn't transmitted over the system; NTLMv2 and Kerberos address the shortcomings in NTLMv1 and counteract rainbow-breaking assaults; The server's personality is validated as well. Be that as it may, Integrated Windows verification does not work over HTTP intermediary associations. NTLM is helpless against various assaults.

Structure based confirmation alludes to any component that depends on variables outside to the HTTP convention for verifying the client. The application's designer has to manage the client certifications, checking them and choosing their realness. With this the engineer is allowed to actualize the Login page in an ideal way. All advancement structures and dialects bolster structure confirmation. Yet, Encryption or security isn't upheld naturally. The duty to actualize a protected arrangement has a place with the designer.

As attackers come up with more brilliant ways, applications needed to shield themselves against more up to date dangers like robotized secret word speculating and key lumberjacks. Assailants made their activity simple by composing contents that would continue using every permutation or combination of passwords on the Login page till a match was found. CAPTCHAs involve haphazardly produced content that is shown in a twisted way. The content can be perused by a human, however not a robotized program. They avert mechanized secret word speculating assaults, avoid computerized DoS assaults, basic and advantageous for a client.

A CAPTCHA that is not implemented in a proper way can make the application vulnerable to attacks. A visual CAPTCHA (distorted text) is not very user friendly for the visually weak.

The risk of key loggers was tended to by various destinations utilizing a virtual console. Since key-logging projects would live on the customer machine and catch every one of the keystrokes and email them to the attacker, virtual consoles take out the need to enter in the secret word. A graphical portrayal of the console is shown on the screen and the client utilizes the mouse to tap on the individual characters. Primary preferences are that it is a basic strategy that can be actualized effectively. A virtual console can likewise be utilized in pages with delicate data like Mastercard subtleties, and so forth. Shoulder surfing turns into an increasingly conceivable danger. There are progressed malevolent programs that catch the mouse clicks and dependent on the pixels, process the characters entered.

One Time Passwords are a type of Two-factor Validation (2FV). They imitate the sharing of a mystery on fly between two computerized devices utilizing an out-of-band correspondence. This brought about expanded trouble for the assailant – necessities to bargain SMSs/messages separated from the application. Validation relies upon insider facts from two unique frameworks. The term of the legitimacy of as far as possible the proceeded with trade off of the client account. Broadened accessibility for the attacker is constrained since OTP can be utilized only once for the particular exchange. The OTP relies upon the accessibility of the extra server and the outer framework (SMSs/messages). There can be delays in delivery of the message, which is outside the application's developer control.

A new type hardware tokens were introduced. The client needs to enter the digits showed on the token along with the secret phrase into the application. On the off chance that both the qualities entered are right, the client accesses the application. The token contains a calculation and a clock and

doi: 10.32622/ijrat.76S201923

a seed or a one of a kind number. Taking the time and the seed together, the calculation produces a number that is shown. Secure as you need the number created by the token to sign in. The token is anything but difficult to utilize. Be that as it may, the equipment token additionally had hindrances.

For example:
• Possibility of a mismatch can happen in the time leading to failure of authentication.
• The token is a very small entity which can be easily lost.
• A lot of extra infrastructure is required which is a cost overhead.
• The distribution and maintenance of the tokens is also an overhead.

After facing numerous problems with above mentioned session based authentication is being used. For each and every session a user logs into the server, server has to maintain user information to validate, authenticate and authorize the user. For this information stored on server a lot of memory is needed when too many user are using the resources at the same time. This makes the server stateful making it very difficult for scalability and migration. Every time an application has to authenticate with a new server, it has to again register a new session or the session from previous server has to be transferred, which results in performance degradation issues. For the above mentioned problems, Json Web Token(JWT) is introduced.

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** or **ECDSA**.
When to use JWT? JWT is used for Authentication, Authorization, and Information exchange.

Authorization: This is the most well-known situation for utilizing JWT. When the client is signed in, each consequent solicitation will incorporate the JWT, enabling the client to get to courses, administrations, and assets that are allowed with that token. Single Sign On is a component that generally utilizes JWT these days, on account of its little overhead and its capacity to be effectively utilized crosswise over various areas.

Information Exchange: JSON Web Tokens are a decent method for safely transmitting data between gatherings. Since JWTs can be marked—for instance, utilizing open/private key sets—you can make certain the senders are who they state they are. Furthermore, as the mark is determined utilizing the header and the payload, you can likewise confirm that the substance hasn't been altered.

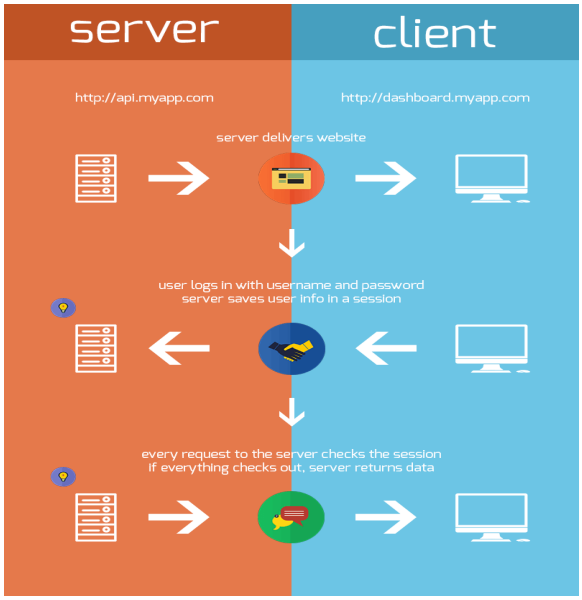

Figure 1: Problem with session based authentication

Problems faced with session based authentication are:
• **Sessions**: Each time a client is validated, the server should make a record some place on our server. This is normally done in memory and when there are numerous clients validating, the overhead on your server increments.
• **Scalability**: Since sessions are put away in memory, this gives issues adaptability. As our cloud suppliers begin repeating servers to deal with application load, having fundamental data in session memory will confine our capacity to scale.
• **CORS**: As we need to extend our application to give our information a chance to be utilized over numerous cell phones, we need to stress over cross-starting point asset sharing (CORS). When utilizing AJAX calls to get assets from another space (portable to our API server), we could keep running into issues with taboo solicitations.
• **CSRF**: We will likewise have assurance against cross-site demand forgery(CSRF). Clients are helpless to CSRF assaults since they would already be able to be validated with state a financial site and this could be exploited when visiting different locales.

So how does JWT work?.

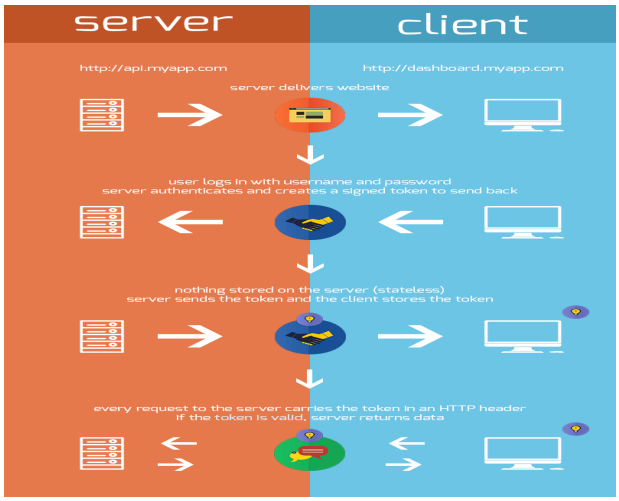

Figure 2: Working of JWT

doi: 10.32622/ijrat.76S201923

When a user logs into a server, the server authenticates the user and sends a token back with the response back to the user. The server stores no information about the user on the server. The client sends the token with every request as HTTP is stateless. This makes the server completely stateless which helps in scalability. Anytime when the user application has to access other domains or third party applications, the server checks whether the token has expired or not. If not the application can continue to access the resources or else the application will have re-authenticate with the server and then access the resources. With scheme one complete server can be placed for authentication where it handles all authentication for the users which minimises the architecture needed for an authentication server. The rest of application servers can separated which aids in scalability and better performance issues.

Benefits of using JWT are:

- Stateless and scalable:
  - Completely stateless and ready to be scaled
  - Without tokens user would have to be sent the same server repeatedly called Session infinity
  - Would result in high spot traffic
- Security:
  - Since cookie is not sent, would prevent CSRPF attacks
  - No session based information to manipulate
  - Token revocation allows to validate token based on expiry time.
- Extensibility:
  - Provide selective permission to third-party apps
  - Can build our own api and handout special permission token allowing application to access user data.
- Multiple platforms and Domain:
  - As applications expand to handle different mobile devices and domains, CORS has to be handled.
  - As long as user has a valid token, interoperability can be maintained.

A Json Web Token consists of a Header, Payload and signature. The header *typically* consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA

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

Figure 3: Header of JWT

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims.

- **Registered claims**: These are a set of predefined claims which are not mandatory but recommended, to provide a set of useful, interoperable claims. Some of them are: iss (issuer), exp (expiration time), sub (subject), aud (audience), and others.
- **Public claims**: These can be defined at will by those using JWTs. But to avoid collisions they should be defined in the IANA JSON Web Token Registry or be defined as a URI that contains a collision resistant namespace.
- **Private claims**: These are the custom claims created to share information between parties that agree on using them and are neither registered or public claims

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

Figure 4: Payload of JWT

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm is specified in the header, and sign that.

For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way.

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Figure 5: Signature of JWT

.

IV. CONCLUSION

The rate at which authentication for web applications is increasing, which requires scalability, quick access of resources and cheaper cost. It provides various benefits to customers. Users are getting to know about this technology from various sources. Many consumers have the assumption that authentication is not as easy as building UI, though some are finding it much more secure than other security policies. Many organizations are holding back to the Session based authentication because, they feel it is unsafe as newer techniques may not protect their data. So, if the customers want to accept the JWT technology with confidence, there is a need to develop some skilled standardization of security. For the third parties, there should be a certification done for ensuring that standards are properly met. JWT comprises three encoded parts: Header, Payload, and Signature. It can be passed as a URL or POST parameter, or in an HTTP header. Due to JWT's lightweight, self-containing, and versatile strucutre, it remains a popular tool for information exchange and authentication.

doi: 10.32622/ijrat.76S201923

REFERENCES

[1] Balaji: Evolution of Authentication in web Applications:‖ https://www.paladion.net/blogs/evolution-of-authentication-in-web-applications‖Oct 29 ,2011.

[2] Miroslav Milinovi, Mijo erek, Dubravko Penezi, Denis Staner, Dubravko Vonina, ―On Evolution of an Authentication and Authorisation Infrastructure in Academic and Research Community: AAI@EduHr – From RADIUS Hierarchy to an AAI Federation‖

[3] Chris Sevilleja, Authentication of Node.js API with JSON WEB TOKENS: ―https://scotch.io/tutorials/authenticate-a-node-js-api-with-json-web-tokens‖ Aprill 22nd , 2015

[4] Introduction to JWT: ―https://jwt.io/introduction/‖ 2011

[5] Arie Pratama, Linawati Linawati, Nyoman Putra Sastra, Token-based single sign-on with JWT as information system dashboard for government‖, Aug 2018

[6] V Beltran. Characterization of Web Single Sign-On Protocols. *IEEE Communications Magazine-Communications Standards Supplement. 2016*; 54(7): 24

[7] Y Tie Jun. *Method of Single Sign-on for Independent Web Systems Based on AJAX.* Proceedings of 2013 3rd International Conference on Computer Science and Network Technology. 2013*:* 310–314.

[8] Kubovy J., Huber C., Jäger M., Küng J. (2016) ―A Secure Token-Based Communication for Authentication and Authorization Servers‖. In: Dang T., Wagner R., Küng J., Thoai N.

[9] Hardt, D.: ―The OAuth 2.0 Authorization Framework.‖ RFC 6749, RFC Editor, October 2012.

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

doi: 10.32622/ijrat.76S201923