Authentication Services

Lakshay Kohli,B.Tech Student
Aman Ahmed, B.Tech Student
Bhavya Ratra,B.Tech Student
Shalaka Tyagi, Assistant Professor, ECE Deptt.
Delhi Technical Campus, Greater Noida

Abstract- Authentication Services play an important role in web applications and web pages. Authentication enables the developers to provide a safe and protected experience to its users. Most modern applications require users to authenticate their identity to safeguard their personal data. Verification is the process of verifying who you are. The user can interact with the web application using multiple actions. Access to specific actions or pages may be restricted using user levels. Authorisation is the process of controlling user access with assigned roles and rights.

I. INTRODUCTION

Cyber security stays in the perennial news these days. Every week it seems that a different company is the victim of hackers stealing sensitive data in which is why providing stronger network security for your clients is more important than ever. The top responsibility of the secure system is to ensure that only authorised users have access to the network. Security protocols need to allow legitimate users to enter while keeping cybercriminals out.*Network authentication* accomplishes this goal. There are a number of authentication methods and tools available, and it is important to understand how they work to choose the right method for your customers. In this article, we will survey a range of user authentication methods and how they can help customers secure their data. But first, let's clarify what authentication really is.

II. Authentication

Authentication is the process of verifying your identity. A unique identifier is associated with a user which is a username or username. Traditionally, we use a combination of username and password to authenticate the user. Verification logic should be kept in place so we will call it location verification. In addition to local authentication, we can use OpenID, Oauth and SAML and can also be used as Auth providers for an extra layer of security as the credentials provided by such techniques are refreshed periodically. For instance **one-time-passwords** are one of the most widely accepted and used methods to satisfy the customers.

We started this research in an attempt to secure our web application. During the course of our project we tried using different authentication. While some of the techniques were much more secure for our project, we choose JSON Web Token (JWT). While using the framework we realised that:

- JSON Web Tokens area good way to securely transmit information between parties.
- Once the user is logged in, each season will include token token transfer to access route, services and resources.

Since most of our users were to connect on a temporary basis, increasing the security to that level was not required since it would also require to give extra time both while developing. As for the users, generating new single use passwords for every login would cause confusion for the user.

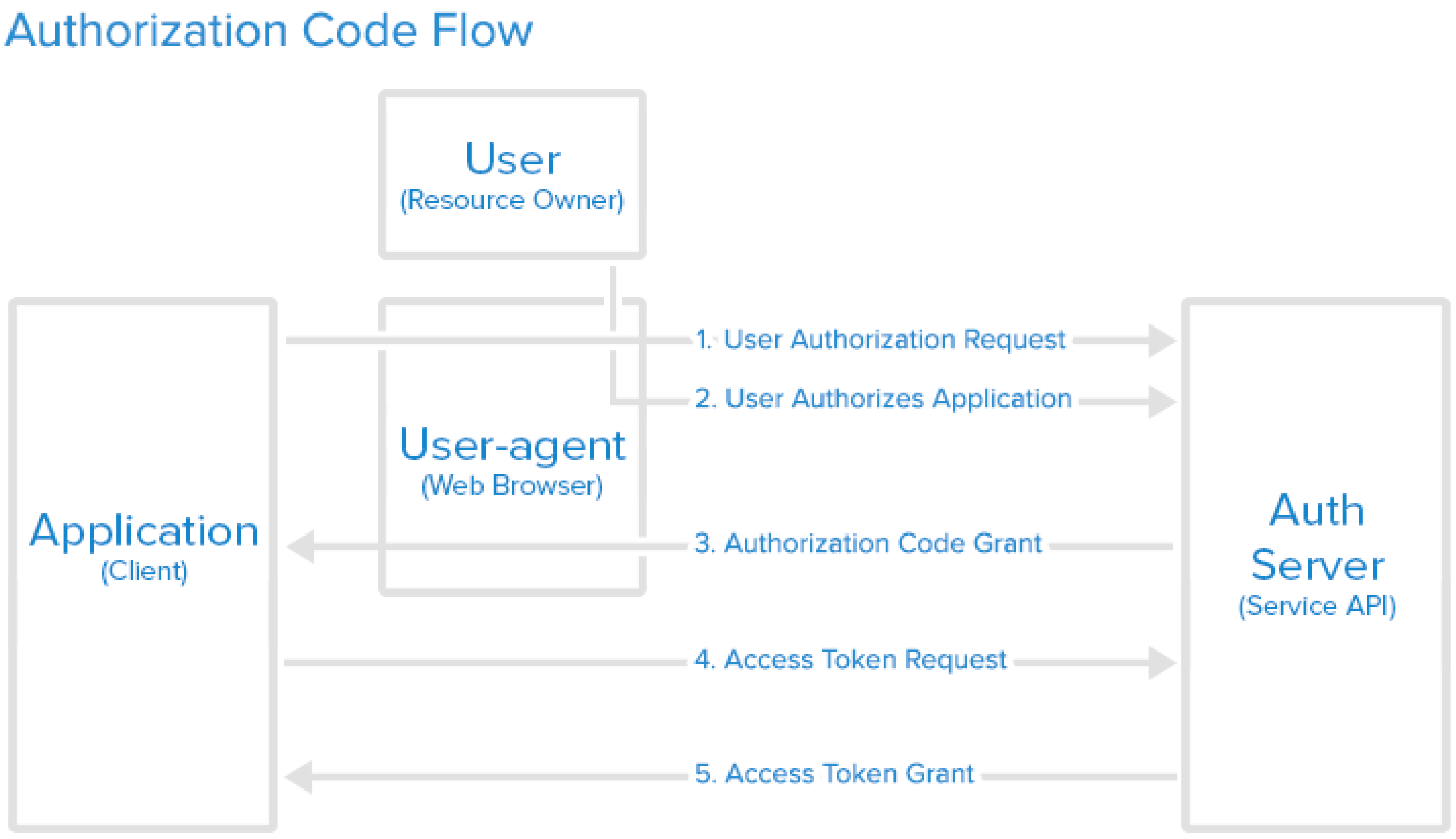
OAuth2

OAuth 2.0, which stands for "Open Authorisation", is a standard designed to allow a website or application to access resources hosted by other web applications on behalf of the user. It replaced OAuth 1.0 in 2012 and is now at the de facto online authorisation industry. OAuth 2.0 provides agreed access and limits the actions of client applications that can be used on behalf of the user, without sharing user information.

Although the web is a major OAuth 2 platform, the definition also describes how to handle this type of access to posts from other types of clients (browser-based applications, web server side applications, native / mobile applications, connected devices, etc.).

It is an authorisation protocol and not an authentication protocol. Its primary function is to grant access to a set of resources, for example, remote API’s or user’s data. OAuth 2.0 uses access tokens. An access token is a piece of data that represents the authorisation to access resources on behalf of the end-users. It doesn’t define a specific format for Access tokens. So JSON Web Tokens can be used in some contexts. This enables token issuers to include data in the tokens itself.

Access tokens can also save the expiration date to prevent hackers from decrypting data.



- **JSON Web Token:**

JSON Web Token or JWT ("jot") is a standard for securely sending requests in space-constrained environments. It found its way to all major web framework. Simplicity, compactness and ease of use are the key characteristics of the architecture. Although much more complex systems5 are still in use, the JWT has a wide range of applications.

Although this representation looks confusing, this format is actually a very compact, printable representation of a series of claim and an authenticating signature for the same.

```
{
  "alg": "HS256",
  "typ": “JWT"
}
{
"sub": "1234567890",
"name": "John Doe",
"admin": true
}
```

- Claims are definitions or assertions made about a certain party or object. JWT specifications contains the detailed meanings of these claims. Others are user defined. The magic behind JWTs is that they standardise certain claims that are useful in the context of some common operations. **So,** one of the standard **assertions** found in **the JWT** is **a sub-assert (in “topic”).**

- Another key aspect of **JWT** is the **ability to sign** using JSON Web **Signature** (JWS, RFC **75156) or encrypt** using JSON Web Encryption (JWE, RFC 75167). Together with JWS and JWE, **JWT provides robust and secure solutions to** a **variety of** problems.

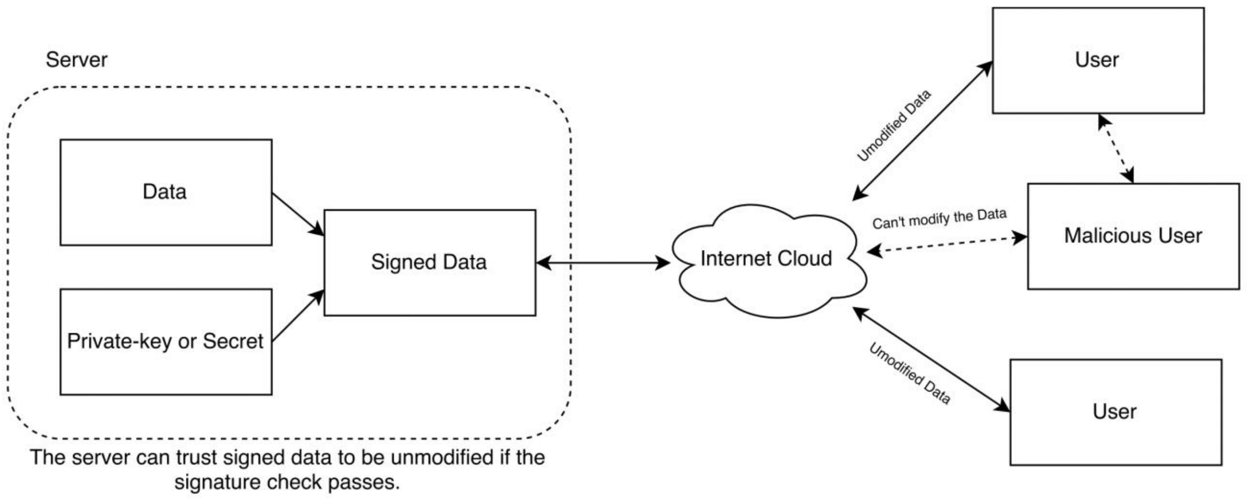
Although the main purpose of JWTs is to transfer claims between the two parties, undoubtedly the most important aspect of this is an attempt to establish a simple, voluntary and / or encrypted, container format. Ad hoc solutions to this same problem have been used both privately and publicly in the past. Older levels8 for receiving claims about certain organisations are also available. JWT provides a simple, useful and a standard container format for the data.

Some of it's applications involves:

- Authentication
- Authorization
- Federated identity
- Client-side secrets
- **Client-side sessions ("stateless" sessions) :**

The so-called random sessions are actually nothing but client-side data. A key feature of this application is the use of possible signature and encryption to secure and secure the content of the session. Client
side data is subject to interruption. It should therefore be handled with great care by the backend.JWT, thanks to JWS and JWE, can provide a variety of signatures and encryption. Sig-nature symbols are useful for validating data against interference. Encryption helps protect data from third party readings.

Most of the time the sessions only need to be signed. In other words, there is no security or privacy concern where the data stored is read by third parties. A common example of a claim that is commonly read by third parties is a sub-claim ("subject"). A title claim usually identifies one of the parties to another (think of user IDs or emails). It is not necessary for this claim to be separate. In other words, additional claims may be required to specifically identify the user. This is left to users to decide.



Client Side Session

STRUCTURE OF JWT

JWTs are structured into three different elements such as the header, the payload, and the signature data which is encrypted.
Header and payload are JSON objects of a structure. The third is dependent on the algorithm used for signing or encryption. JWTs can be encoded in a compact representation known as JWS/JWE Compact Serialization

Example data:

HEADER:

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

PAYLOAD{

```
 "id": "1234567890",
 "name": "Bhavya ratra",
 "course": "Btech"
}
```

The resulting sequence after encoding header and payload from above data :

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjEyMzQ1Njc4OTAiLCJuYW1lIjoiQmhhdnlhIHJhdHJhIiwiY291cnNlIjoiQnRlY2gifQ.BXfxP7LHtBS_Qg1vmgzRpqoUtiQhr-NrFzTHENVh5nE

- Every JWT has a header with claims about itself. These claims establish the algorithms used, whether the JWT is signed or encrypted, and how to parse the rest of the JWT.

- The payload is the element where the user data is stored. The payload is also a JSON object like the header

Sample code

Sign:

```
function  getSignedToken() {
  return jwt.sign({
 "id": "1234567890",
 "name": "Bhavya ratra",
 "course": "Btech"
}, process.env.JWT_SECRET, {
    expiresIn: 10 * 60 * 60,
  }),
{ algorithm: 'RS256'}
};
```

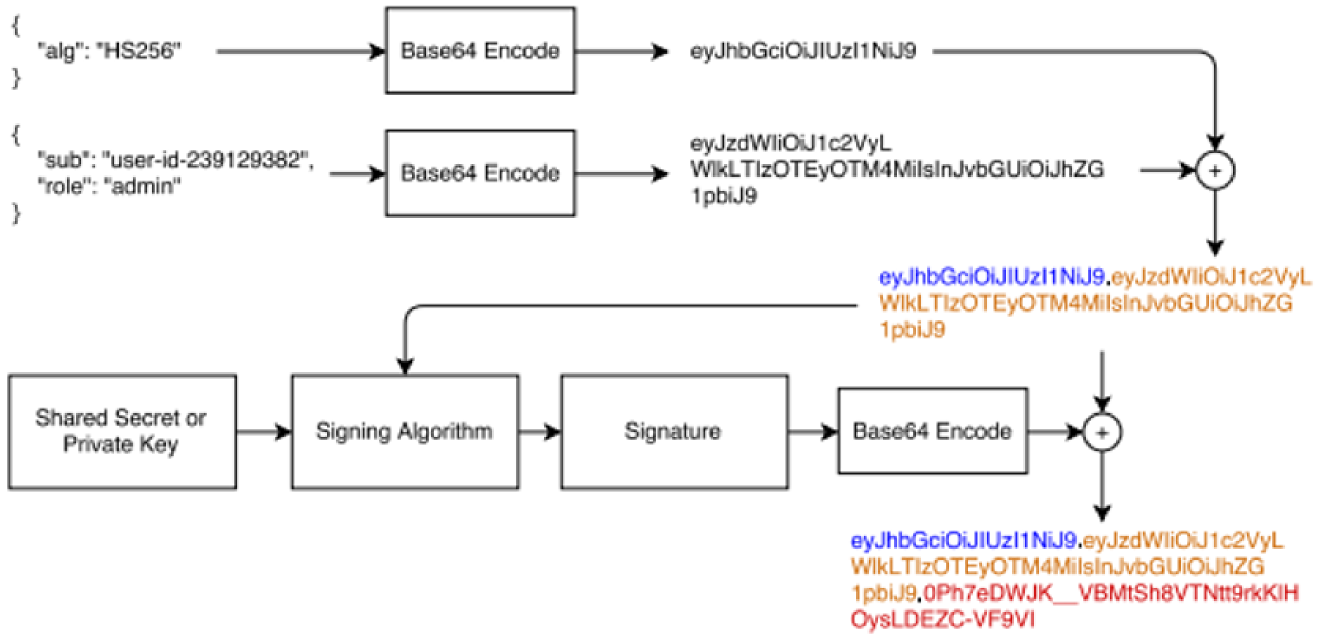
JWT secret is a string which is used for signing and needed for encryption/decryption of data.

Verify:

```
function verifyToken(){
                return jwt.verify(token, process.env.JWT_SECRET);
}
```

Returns the decrypted payload

JWT Compact Serialization


{ "alg": "HS256" }
Base64 Encode
eyJhbGciOiJIUzI1NiJ9
{ "sub": "user-id-239129382", "role": "admin" }
Base64 Encode
eyJzdWIiOiJ1c2VyL WlkLTIzOTEyOTM4MiIsInJvbGUiOiJhZG 1pbiJ9
+
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyL WlkLTIzOTEyOTM4MiIsInJvbGUiOiJhZG 1pbiJ9
Shared Secret or Private Key
Signing Algorithm
Signature
Base64 Encode
+
eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ1c2VyL WlkLTIzOTEyOTM4MiIsInJvbGUiOiJhZG 1pbiJ9.0Ph7eDWJK__VBMtSh8VTNtt9rkKIH OysLDEZC-VF9VI

References:

- JWT Handbook, Sebastián Peyrott
- Introduction to JSON Web Tokens,(https://jwt.io/introduction)
- Official OAuth Website,(https://oauth.net/2/)
- (https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2)