International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 05 Issue: 05 | May-2018 www.irjet.net p-ISSN: 2395-0072

# Secure External Login Based on Authorization Code Flow using JWT

**Dr. R. Kannan[1], G. Umasankar[2]**

[1] *Associate Professor, Department of Computer Science, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore-20*

[2]*M.Phil. Scholar, Sri Ramakrishna Mission Vidyalaya College of Arts and Science, Coimbatore-20*

---

**Abstract -** *Now a day's everyone use web and cloud based application using everywhere. It's given a quick solution and time consumption and communication for the users. In this place authentication and authorization have been a part of communication. It's based on the Identity. What is Identity means who is the user. Now a days we are widely using Single Sign.. In this SSO has ability for one application, known as an identity provider, to tell other applications, known as service providers, who you are? In this context, identity providers are systems that contain digital identity information about users.*

*This communication based HTTP Protocol. That base given a token based authenticate and authorize to Resource server. We can saw the flows and implementation external authentication. And important of JWT.*

***Key Words***: **Authentication, Authorization, Single-Sign-On, OAuth, JWT.**

## 1. INTRODUCTION

Web applications have become an essential component of business in today's world. In this place Security is very important. These applications can help target numerous client and customers at a time. And now its targets web and cloud based applications and multiple types of devices also, that time security is most important. Now we are using JWT (JSON WEB TOKEN) for Authentication purpose.

Is a JSON-based open standard (RFC 7519) for creating access tokens that assert some number of claims The tokens are designed to be compact, URL-safe and usable especially in web browser single sign-on (SSO) context.

Authentication is the process of verifying the users using credentials. In this process for create a user identity for the server and client. In this user identity has contains the roles and permissions.

In the authorization process its check the user identity based to control access rights by granting or denying specific permission to an authenticated user.

External authentication uses the Central Authentication Service (CAS), which enables Single Sign-On (SSO), and allows a user to authenticate with a CAS. External Authentication is commonly used away for Internet users to grant websites or web applications access to their information on other websites but without given them the passwords.

In now days external authentication using OAuth2 (Open Authentication). It's specifies a process for resource owner to authorize third-party access to their server resource without sharing their credentials.in the place we can saw the External Authentication and JSON WEB TOKEN Security. We can saw authorization code flow for identify the users.

## 2. SIGNLE SIGN ON

In this world all the things in web and cloud based. In this time users can't memory all the passwords. In this situation. We can use the Single sign on concept. It's used authenticate the user without user credentials on the resource server.

In this application identify the users using external server. Its common directory service for the enterprise applications either a social login's. That give the access token that based identity the user in server. These token based access other protected resources without having to re-authenticate.

In Enterprises applications are using active directory or IAM (Identity and Access Management) service for authenticate to the user. It verify and given the access permission their resources. If authentication succeed, it will establish a relationship of trust that grant user the access to all web resources for which he/she have permissions.

Wed-based SSO is a widely deployed single sign-on technology sometimes also called web access management. In this trend commonly used external authentication for social network login's like Google, Facebook, Microsoft Outlook, twitter and etc.,

## 3. OAUTH (OPEN AUTHENTICATION)

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. The latest iteration of OAuth, formalized in 2012, is the version 2.0.

As one can imagine, it is much more accommodating to current trends and needs in the industry, OAuth includes the notion of Access Token as the mechanism of choice for allowing access to restricted resources. In other words, an Access Token is the authorization issued to a client

In an OAuth has four type generating access token. In this place we can saw the server to server communication based authorization code grant flow.

### 3.1. OAUTH SERVER AND CLIENT CONFIGUATION

OAuth is an authentication and authorization protocol that is widely used on the Internet. It's based on client server communication model. In authorization code flow required client information for the authorization server.

Before using OAuth with your application, you must register your application with the OAuth service provider. This is done through a registration form in the "developer" or "API" portion of the service's website, where you will provide the following information (and probably details about your application)

- Application Name
- Application Website
- Redirect URI or Callback URL

The redirect URI is where the service will redirect the user after they authorize (or deny) your application, and therefore the part of your application that will handle authorization codes or access tokens.

### 3.2 Client ID and Client Secret

Once your application is registered, the service will issue "client credentials" in the form of a client identifier and a client secret. The Client ID is a publicly exposed string that is used by the service API to identify the application, and is also used to build authorization URLs that are presented to users.

The Client Secret is used to authenticate the identity of the application to the service API when the application requests to access a user's account, and must be kept private between the application and the API.

### 3.3 OAUTH 2.0 Roles

OAuth defines the four roles.

1. **Resource owner**

   Resource owner is the Person it is referred to the end user.

2. **Resource server**

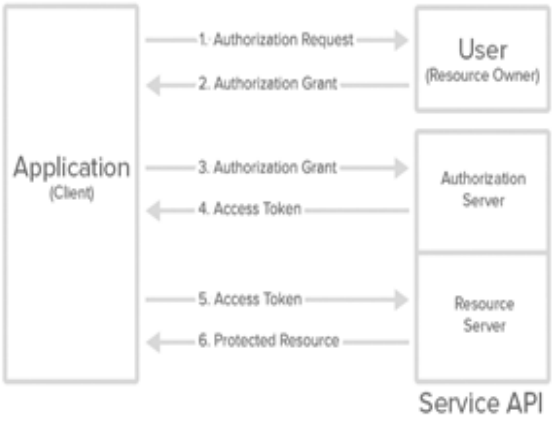
   The server hosting the protected resources, capable of accepting and responding to protected resource requests using access token

3. **Client**

   An Application making protected resource request on behalf of the resource owner and with is authorization.

4. **Authorization server**

   The server issuing access token to the client after successfully authenticating the resource owner and obtaining authorization.




### 3.4 OAUTH COMMUNCIATION PROCESS

The **authorization code** grant type is the most commonly used because it is optimized for *server-side applications*, where source code is not publicly exposed, and *Client Secret* confidentiality can be maintained. This is a redirection-based flow, which means that the application must be capable of interacting with the *user-agent* (i.e. the user's web browser) and receiving API authorization codes that are routed through the user-agent.

In my analysis OAuth has three stages. The first stage is the user is requested to the External Login in our application server or resource server. In this server has redirect to the authorization server (or) External Server.

In this redirection contains some query string in a URL part. This is important for authentication for external server. Because that based only identify which client is request for the user information. If the client id and redirect URL. Doesn't match it's say it's not valid client. This client is not registered is our system.

In an Authorization server find the client. And in that client is valid then only it's redirected to the Login Page. In that Login Page is used for the End User (or) Resource owner Pass the Credentials. If the credentials is valid the server set the cookie in response to the user agent. And the same time it's redirected to the user browser for next stage.

In a second stage that redirect URL pass to resource server that server has taken the information using the query string. In that query string contains the authorization code.

In this Authorization code has Contains the user information and the client information. That code generation is authorization server logic. That authorization code has some validity and it's verify for subsequent request in authorization server. In this authorization code validate

from the client application and firewall and redirect URL based on OAuth Server configuration of Client Application

In the authorization code is very important for identity the client and the user for the external authorization server. In the code based perform the post action from the resource server to the external or authorization server. That communication required for the client information like client id, secret, redirect URL and authorization code.

In the External Authorization server check the client information and request domain and firewall and redirect url based it's valid then check the code it's given by this authorization server or not. If code also valid then its fetch the user information and user scopes and permissions and it's generate the JWT. And its token encoded the Base64 format. That token send to the resource server.

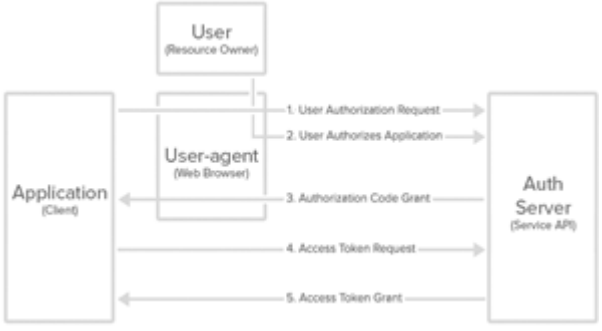
In the Response is the third stage of the Process. In the token is decoded and get the JWT. That JWT claims based identify the system user whether it's a new user or existing user that based fetch the user that based generate the new access token for resource server.

In this new Access token has contains the user identity for perform the subsequent request in the resource server access for the protected resource. That based perform the authorization and secure communication.

Using Social Login IDP's (Identity Provider) JWT contain the identity provider user id claim that based getting the user from the local database based on internal user mapping and generate the new access token for the local issuer. Either Enterprise Application are using the common IAM (Identity and Access Management) issuer.

In Our Application is user Social Login's Like Google Plus, Facebook, etc., we can identity the user using External Authentication and we generate a new access token. For our domain based issue the new access token. Otherwise, we using Enterprise Login like Azure Active Directory or something like we using that IDP provider given token for our application. In our application either using cookie or token based send the response for authorize to sub sequent requests.

Authorization Code Flow Diagram:




## 4. JSON WEB TOKEN

JWTs are an encoded representation of a JSON object. The JSON object consists of zero or more name/value pairs, where the names are strings and the values are arbitrary JSON values. JWT is useful to send such information in the clear (for example in an URL) while it can still be trusted to be unreadable (i.e. encrypted), unmodifiable (i.e. signed) and url-safe (i.e. Base64 encoded).

### 4.1 JSON WEB TOKEN STRUCTURE

The Encode **JWT** format contain the Header, Payload and Signature these three part separated with (.) for example: **Header (.) Payload (.) Signature**.

1. Header: The first part of a JWT is an encoded string representation of a simple JavaScript object which describes the token along with the hashing algorithm used.

2. Payload: The second part of the JWT forms the core of the token. Payload length is proportional to the amount of data you store in the JWT. General rule of thumb is: store the bare minimum in the JWT.

3. Signature: The third, and final, part of the JWT is a signature generated based on the header (part one) and the body (part two) and will be used to verify that the JWT is valid.

The payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed or integrity protected with a Message Authentication Code (MAC) and/or encrypted.

JWTs can have different usages: authentication mechanism, url-safe encoding, securely sharing private data, interoperability, data expiration, etc. Regardless of how you will use your JWT, the mechanisms to construct and verify it are the same.

`Sample JWT Encode Format is **eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJPbmxpbmUgSldUIEJ1aWxkZXIiLCJpYXQiOjE1MjM3MDk5MTAsImV4cCI6MTU1NTI0NTkxMCwiYXVkIjoid3d3LmV4YW1wbGUuY29tIiwic3ViIjoianJvY2tldEBleGFtcGxlLmNvbSIsIkdpdmVuTmFtZSI6IkpvaG5ueSIsIlN1cm5hbWUiOiJSb2NrZXQiLCJFbWFpbCI6Impyb2NrZXRAZXhhbXBsZS5jb20iLCJSb2xlIjpbIk1hbmFnZXIiLCJQcm9qZWN0IEFkbWluaXN0cmF0b3IiXX0.xSnUao6u6YHlKPDKjl9ZoSyTy6vYJXsq0iXyXTvu8A**

**Sample JWT Decoded format is**

```
{
 typ: "JWT",          //Header part
 alg: "HS256"
}.
```

```
{                       //Payload part
 iss: "menporul",
 iat: 1523942501,
 exp: 1555478501,
 aud: "www.menporul.in",
 sub: "jwt-token",
 GivenName: "Umasankar",
 Surname: "Govindaraj",
 Email: "umasankar@menporul.in",
 Role: [
  "Admin",
  "Customer"
 ]
}.
[signature]          // signature part
```

What are the benefits of using a token-based approach? [4]

- **Cross-domain / CORS:** cookies + CORS don't play well across different domains. A token-based approach allows you to make AJAX calls to any server, on any domain because you use an HTTP header to transmit the user information.
- **Stateless (a.k.a. Server side scalability):** there is no need to keep a session store, the token is a self-contained entity that conveys all the user information. The rest of the state lives in cookies or local storage on the client side.
- **CDN:** you can serve all the assets of your app from a CDN (e.g. JavaScript, HTML, images, etc.), and your server side is just the API.
- **Decoupling:** you are not tied to any particular authentication scheme. The token might be generated anywhere, hence your API can be called from anywhere with a single way of authenticating those calls.
- **Mobile ready:** when you start working on a native platform (iOS, Android, Windows 8, etc.) cookies are not ideal when consuming a token-based approach simplifies this a lot.
- **CSRF:** since you are not relying on cookies, you don't need to protect against cross site requests (e.g. it would not be possible to sib your site, generate a POST request and re-use the existing authentication cookie because there will be none).
- **Performance:** we are not presenting any hard performance benchmarks here, but a network roundtrip (e.g. finding a session on database) is likely to take more time than calculating an HMACSHA256 to validate a token and parsing its contents.

### 4.2 HOW TO AUTHORIZE USING JWT

In a JWT Authorization Process is suitable for web and mobile and other device based application. In JWT Encoded access token client storage like Local Storage in Browser and any Variables. In the JWT token in passed all the request of the resource server in request header.

This header name is Authorization. In this Header we passed the value for Bearer {Access Token} this format based pass the token in resource Server. In this Server validate the token.

The Token based get the User and Claims in authorization server for access the protected resource. If expired or any malfunction it's detected in this token its throw the error in response.

## 5. RELATED WORK

I was Analyze the OAuth Server and Client Process and implemented the Social Login Google, Facebook, and Azure Active Directory for Enterprise Common Login.

I was written the custom OAuth server using OWIN Middleware in ASP.Net Web API Application Angular JS1. In this server based access the API server on various clients on Web, and Android and IOS Application. It's Working Perfectly. And JWT Token Handle in web and Mobile for the storage and if expired we using he refresh token Technology in Application.

These Implementation and Analysis based I getting knowledge for how to work implement the OAuth Server and Client and Communicate securely for the client and server using JWT.

## 6. CONCLUSION

In this paper can we saw the authentication, authorization single sign-on external authentication server to server communication, client server communication (Authorization Code flow) and JSON WEB Token and its process and advantages and how to implement the JWT in our web and mobile based application.

OAuth2 is a security framework. It's a details ways of authenticating multiple types of applications in various different scenarios. Because of this there is a lot of material to learn. This is not a fast process. JWT on the other hand is relatively light on conceptual understanding. After a day of reading the specs I felt comfortable starting an implementation.

In many situations it is convenient to let user's authenticate using a pre-existing account they have on other larger websites.

If you expect your users to be using an OAuth provider like Facebook or Gmail, OAuth2 can be a pretty fast and pain free way of adding authentication, by using an existing library.

In this work we can cover the External Authentication based on the JWT and OAuth 2.0 Server side process. And we know how to work on OAUTH 2.0 Authorization flow and JWT based Security implementation.

I was gathered what is JWT? How to add the custom claims? And how to maintain the token, how to decode and extract the claim and authorize the user and that based access the protected resource. If expire how to generate the new token use refresh token technology. That knowledge based on writing this paper.

## REFERENCE

[1] https://ac.els-cdn.com/S1877050913009423/1-s2.0-S1877050913009423-main.pdf?_tid=678fb538-c968-4f09-86a1-11cdc4b34948&acdnat=1522582485_c62edef04c38c724f52a976df6928249

[2] https://en.wikipedia.org/wiki/OAuth

[3] https:/yourstory.com/2015/07/web-application

[4] https://stackoverflow.com/questions/1592534 /what-is-token-based-authentication

[5] https://docs.microsoft.com/en-us/aspnet/web-api/overview/security/external-authentication-services

[6] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.5661&rep=rep1&type=pdf

[7] https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[8] http://bitoftech.net/2014/06/01/token-based-authentication-asp-net-web-api-2-owin-asp-net-identity/

[9] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.403.5661&rep=rep1&type=pdf

[10] http://www.seedbox.com/en/blog/2015/06/05/oauth-2-vs-json-web-tokens-comment-securiser-un-api/

## BIOGRAPHIES

**Dr. R. Kannan. MCA, M. Phil.,** Associate Professor, Department of Computer Science. Sri Ramakrishna Mission Vidyalaya College Of Arts and Science, Coimbatore, TamilNadu

**G. Umasankar,** He was complete MCA and currently studying in M. Phil., in Sri Ramashina Mission Vidyalaya College of Arts and Science. Coimbatore. TamilNadu