Kerberos Grant

Kerberos is a security protocol that has support built into various operating systems and open-source distributions (e.g., Ubuntu, Windows, RedHat, Open Solaris, etc). Additionally, a majority of browsers support some Kerberos functions as well. As WSO2 Identity Server (WSO2 IS) uses the OAuth 2.0 protocol, the Kerberos-OAuth2 grant type allows organizations to exchange a Kerberos ticket for an OAuth 2.0 token. This allows organizations to re-use their existing Kerberos infrastructure and adopt OAuth 2.0.

- Kerberos-OAuth2 grant flow
- Configuring Kerberos Grant with Identity Server

Kerberos-OAuth2 grant flow

The following section describes the flow involved in exchanging a Kerberos ticket for an OAuth2 token.

1. The Kerberos client requests the Kerberos Service Ticket from the Kerberos Key Distribution Center (KDC) to invoke the service.
   The Kerberos Key Distribution Center can be any Kerberos Server.
2. The Kerberos Key Distribution Center sends a response with the Kerberos Service Ticket. If the client and the requested service is valid, the Key Distribution Center (KDC) sends a Kerberos ticket encrypted with the service owners private key. The API handles the exchanging of the Ticket Granting Ticket (TGT), Service Granting Ticket (SGT), and all other low level Kerberos details.
3. The Kerberos client requests the OAuth2 token. For more information on how to generate the Kerberos token, see step 6 given below. The message format of the OAuth2 token request should be as follows:

**cURL Request Format**
You can use one of the following two cURL commands to request for the OAuth2 token.

```bash
```

```bash
```

**cURL Response**

**Example**

```
POST /oauth2/token HTTP/1.1
Host: idp.example.com:9443
Content-Type: application/x-www-form-urlencoded
Authorization: Basic MW91TDJmTzZTeGxmRDJMRHcxMjVjVG8wdlFrYTp1UV0bTg5dFk2UVp1WlVtcVpmTDkyQkRGZUFh
grant_type=kerberos&kerberos_realm=example.com&kerberos_token=YII1...
```

4. The Kerberos client receives the OAuth2 token.
The Kerberos Grant validates the received token with the provided Identity Provider (IDP) credentials and if it is a valid token, it issues an OAuth2 token to the client.

**Example**

```
{
    "access_token":"636ce45f-c7f6-3a95-907f-d1f8aca28403",
    "refresh_token":"831271d9-16ba-3bad-af18-b9f6592a8677",
    "scope":"my_scope",
    "token_type":"Bearer",
    "expires_in":521
}
```

**Configuring Kerberos Grant with Identity Server**

Follow the instructions below to configure Kerberos Grant with WSO2 IS:

1. Download the Keberos-grant JAR (*kerberos-grant-1.0.0.jar*).
2. Copy the JAR into the `<IS_HOME>/repository/components/lib` directory.
3. Add following entry under `<SupportedGrantTypes>` in the `<IS_HOME>/repository/conf/identity/identity.xml` file.
4. Configure OAuth2 with IWA as an allowed grant type.
   a. Sign in to the WSO2 IS Management Console.
      https://<Server-Host>:9443/carbon
   b. Navigate to the Main menu, click Add under the Service Providers menu.
   c. Add a new Service Provider and configure OAuth2 for your client application with the kerberos grant type.
To enable OAuth support for your client application, you must first register your application. Follow the instructions below to add a new OAuth2 application.

i. Expand the **OAuth/OpenID Connect Configuration** and click **Configure**.

ii. Fill in the form that appears. For the **Allowed Grant Types** you can disable the ones you do not require or wish to block. Select the **kerberos** grant type as an allowed grant type.

iii. Click **Add**. The following information is added to your service provider.

- **OAuth Client Key** - This is the client key of the service provider, which will be checked for authentication by the Identity Server before providing the access token.
- **OAuth Client Secret** - This is the client secret of the service provider, which will be checked for authentication by the Identity Server before providing the access token. Click the **Show** button to view the exact value of this.
- **Actions** -
  - **Edit**: Click to edit the OAuth/OpenID Connect Configurations
  - **Revoke**: Click to revoke (deactivate) the OAuth application. This action revokes all tokens issued for this application. In order to activate the application, you have to regenerate the consumer secret.
  - **Regenerate Secret**: Click to regenerate the secret key of the OAuth application.
  - **Delete**: Click to delete the OAuth/OpenID Connect Configurations.

5. Configure the Service Principal Name (**SPNName**) and Service Principal Password (**SPNPassword**).

a. Navigate to the **Main** menu, click **Add** under the **Identity Providers** menu.

b. Add a new Identity Provider (IDP). Enter the basic information as follows.

   - **Identity Provider Name**: example.com
   - **Alias**: https://192.168.53.12:9443/oauth/token

   The IDP name should be the name of the realm as specified in the token request in step 3. Based on this example, it should be `example.com`. 
c. Expand the **Federated Authenticators** tab, and then the **IWA Kerberos Configuration** tab. Enter the required details as follows.
   - **Server Principal Name**: HTTP/idp.example.com@EXAMPLE.COM
   - **Server Principal Password**: <password>

6. Generate the kerberos token.
   a. Git clone the KerbClientProject.

```
git clone https://github.com/erandacr/KerbClientProject
```

b. Run KerbClient.cs using an IDE.
   You can run it using Visual Studio by downloading and installing the following required libraries and programs.
   - Visual Studio sdk (.NET Core 2.1)
   - Microsoft Visual Studio (Professional Edition)
   - Install the System.Net.Http.dll and define the path in the 'KerbClientProject.csproj' file.

   **Tip:** Note that you can also use any other IDE to run this project.

c. Configure the following parameters in the project according to your setup.
d. Run the project by selecting the **Start without Debugging** option on the Visual Studio editor.

This project generates a Kerberos ticket and a kerberos token is generated using the ticket. The generated token can be used to get the OAuth token.

7. Invoke the token endpoint using the message format discussed in **step 3**.

```csharp
// Service Name goes here
static string serviceName = "HTTP/apimserver.example.com@EXAMPLE.COM";

// Token endpoint URL
static string URI = "https://idp.example.com:9443/oauth2/token";

// Client ID
static string username = "1ouL2fO6SxlfD2LDw125cTo0vQka";

// Client Secret
static string password = "uUEtm89tY6QZuZUmqZfL92BDFeAa";

// Kerberos realm name
static string realm_Name = "example.com";
```