Salesforce Integration Authentication Techniques

OAuth 2.0 & SOAP Login Authentication

The Context

- Secure authentication is essential for enterprise applications running on mobile devices. OAuth 2.0, the industry-standard protocol, enables secure authorization for access to a customer's data, without handing out the username and password. It is often described as the valet key of software access. A valet key restricts access to certain features of your car. For example, a parking attendant can't open the trunk or glove compartment using a valet key.

- Mobile app developers can quickly and easily embed the Salesforce OAuth 2.0 implementation. The implementation uses an HTML view to collect the username and password, which are then sent to the server. The server returns a session token and a persistent refresh token that are stored on the device for future interactions.

- A Salesforce connected app is the primary means by which a mobile app connects to Salesforce. A connected app gives both the developer and the administrator control over how the app connects and who has access. For example, a connected app can restrict access to a set of customers, set or relax an IP range, and so on.

Approach

- AUTHENTICATE YOUR API CALLS
    - The first step in an API-based integration is authenticating your calls. There are 2 ways to do it
        - OAUTH 2.0
        - SOAP LOGIN AUTHENTICATION

- WHAT IS OAUTH

- OAuth is an open protocol that authorizes a client application to access data from a protected resource through the exchange of tokens. OAuth tokens are essentially permissions given to a client application. The resource server can validate the tokens and allow the client application access to the defined protected resources. In Salesforce, you can use OAuth authorization to approve a client application's access to your org's protected resources.

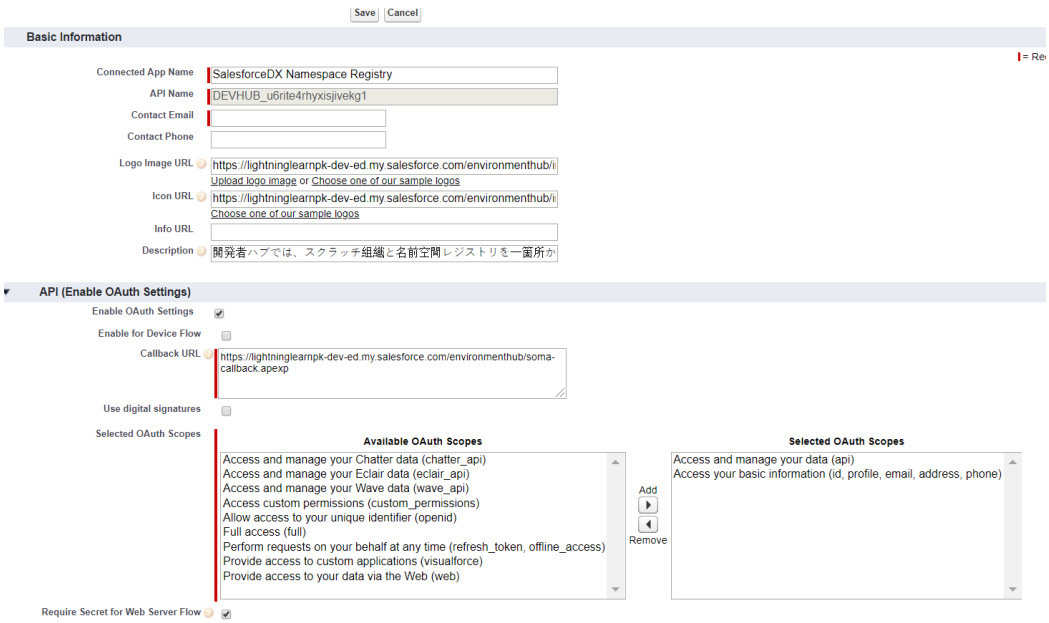
- OAUTH 2.0 FLOWS
  - OAuth authorization flows grant a client application restricted access to protected resources on a resource server. Each OAuth flow offers a different process for approving access to a client app, but in general the flows consist of three main steps.
  - To initiate an authorization flow, a client app requests access to a protected resource.
  - In response, an authorizing server grants access tokens to the client app.
  - A resource server then validates these access tokens and approves access to the protected resource.

- OAUTH 2.0 TOKENS
  - Authorization Code
    - An authorization code is a short-lived token representing the user's access grant, created by the authorization server and passed to the client application via the browser. The client application sends the authorization code to the authorization server to obtain an access token and, optionally, a refresh token.
  - Access Token
    - The access token is used by the client to make authenticated requests on behalf of the end user. It has a longer lifetime than the authorization code, typically on the order of minutes or hours. When the access token expires, attempts to use it will fail, and a new access token must be obtained via a refresh token.
  - Refresh Token

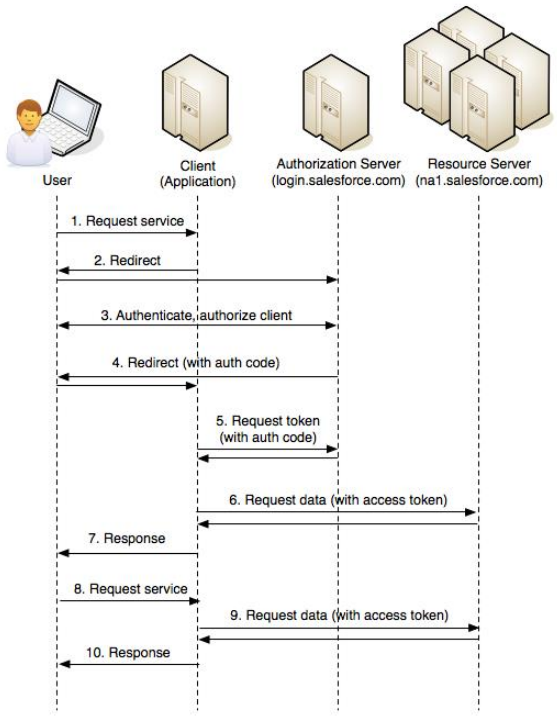
- The refresh token may have an indefinite lifetime, persisting until explicitly revoked by the end-user. The client application can store the refresh token, using it to periodically obtain fresh access tokens, but should be careful to protect it against unauthorized access, since, like a password, it can be repeatedly used to gain access to the resource server.
- Since refresh tokens may expire or by revoked by the user outside the control of the client application, the client must handle failure to obtain an access token, typically by replaying the protocol from the start.

- SALESFORCE CONNECTED APPS

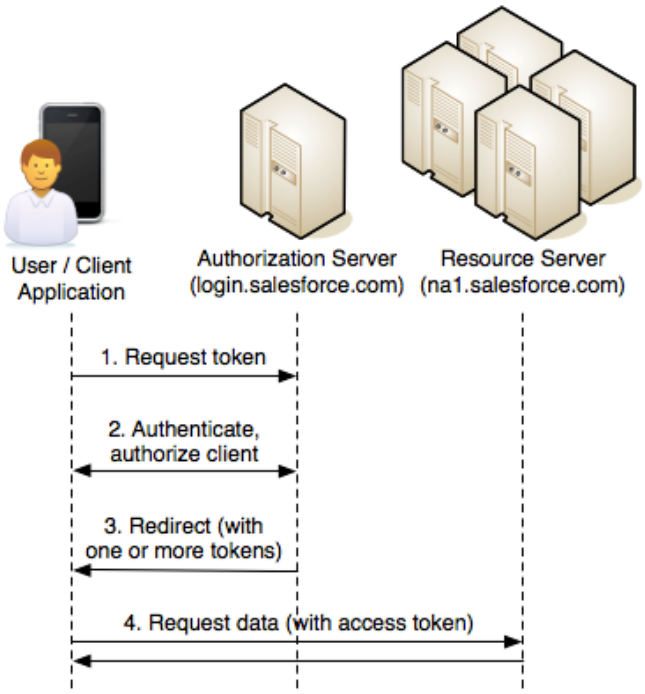
  - A connected app is a framework that enables an external application to integrate with Salesforce using APIs and standard protocols, such as SAML, OAuth, and OpenID Connect. Connected apps use these protocols to authenticate, authorize, and provide single sign-on (SSO) for external apps. The external apps that are integrated with Salesforce can run on the customer success platform, other platforms, devices, or SaaS subscriptions. For example, when you log in to your Salesforce mobile app and see your data from your Salesforce org, you're using a connected app.



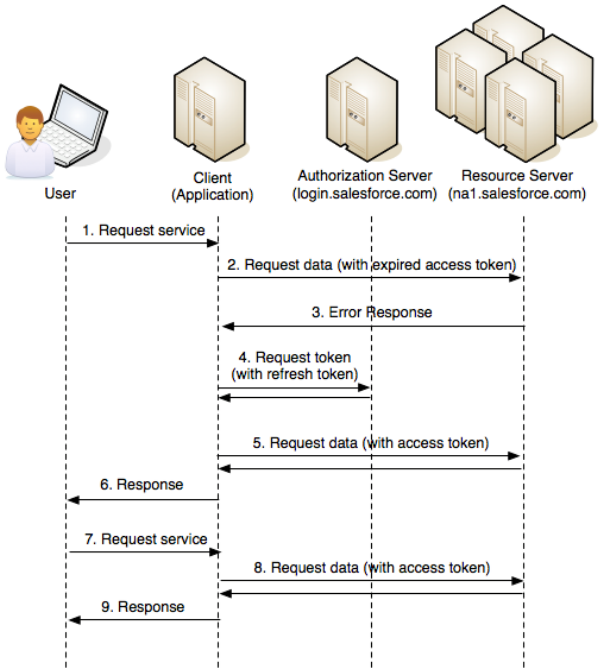
- OAUTH 2.0 WEB SERVER FLOW FOR WEB APP INTEGRATION
    - Most web applications will use the web server client profile and authorization code access grant type to obtain an access token on behalf of an end user.
    - The sequence starts (1) with a user requesting some service from the client
    - The client application responds by (2) redirecting the end user's browser to a URL of the following form: https://login.salesforce.com/services/oauth2/authorize?response_type=code&client_id=<your_client_id>&redirect_uri=<your_redirect_uri>
    - On successful authorization, the user's browser is redirected back to the redirect URI at the client application (4), with a URL of the form: https://app.example.com/oauth_callback?code= <Response Code>
    - (5) Request Token with Auth Code with a URL and payload of the form: https://login.salesforce.com/services/oauth2/token code=<Code>&grant_type=authorization_code&client_id=<your_client_id>&client_secret=<your_client_secret>&redirect_uri=<your_redirect_uri>
    - (6) In Response Access Token is provided
    - (10) Use the access token which can be used to access salesforce service.



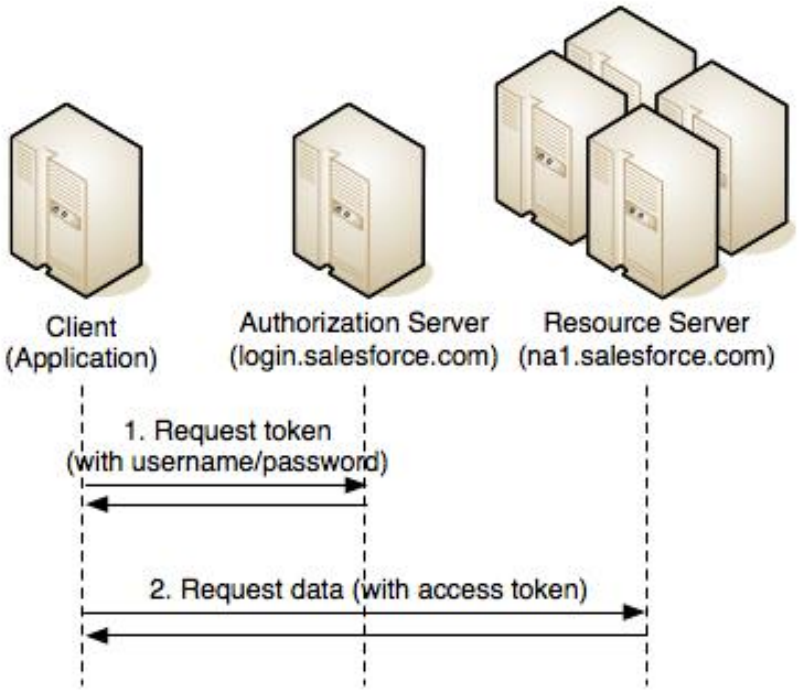
- OAUTH 2.0 USER AGENT FLOW
    - Client applications, for example, JavaScript running in the browser or native mobile or desktop apps, run on a user's computer or other device. Such apps are able to protect per-user secrets, but, since they are widely distributed, a common client secret would not be secure. The user-agent flow allows these applications to obtain an access token:
    - In this flow, the client application directs (1) the user to a URL at the authorization server of the form: https://login.salesforce.com/services/oauth2/authorize?response_type=token&client_id=<your_client_id>&redirect_uri=<your_redirect_uri>&display=touch&stat e=<some_state>
    - As in the web server flow, the user is authenticated and prompted to authorize the client application's access to resources (2):
    - Now, rather than sending an authentication code to the client and it retrieving the access token via a POST request, a redirect is returned (3) containing several parameters in a URL fragment myapp:oauth#access_token=<access token>&issued_at=<timestamp>&signature=<mysignature>&state=mystate
    - (4) Use the access token which can be used to access salesforce service.



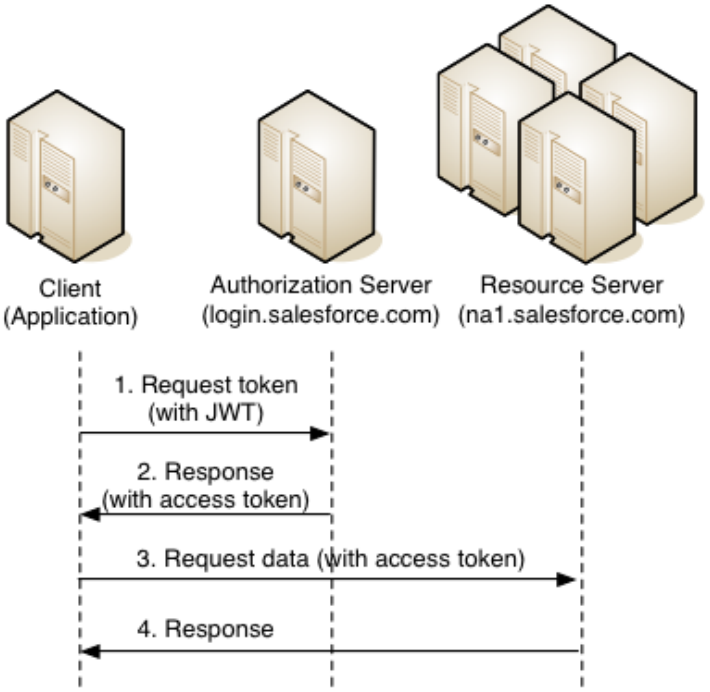
- REFRESH TOKEN FLOW
  - Refresh token is available only for Web Server and User Agent flows.
  - The lifetime of an access token obtained by the above mechanisms is limited to the session timeout configured in salesforce Session Settings.
  - In this situation, the client application can use the refresh token to obtain a new access token. The refresh token represents the user's access grant to the application, and is valid until explicitly revoked by the user, via Setup ➤ My Personal Information ➤ Remote Access.
  - In this flow, The client application obtains a new access token by POSTing another request (4) to https://login.salesforce.com/services/oauth2/token, this time with payload of the form:
    grant_type=refresh_token&client_id=<client id>&client_secret=<client secret>&refresh_token=< previous access grant >



- OAUTH 2.0 USERNAME PASSWORD FLOW
    - Client applications, for example, JavaScript running in the browser or native mobile or desktop apps, run on a user's computer or other device. Such apps are able to protect per-user secrets, but, since they are widely distributed, a common client secret would not be secure. The user-agent flow allows these applications to obtain an access token:
    - An autonomous client can obtain an access token by simply providing username, password and (depending on configuration) security token in an access token request. Again the request is POSTed (1) to: https://login.salesforce.com/services/oauth2/token, but the payload now has the form: grant_type=password&client_id=<your_client_id>&client_secret=<your_client_secret>&username=<your_username>&password=<your_password>
    - (2) Use the access token which can be used to access salesforce service.



- OAUTH 2.0 JWT TOKEN FLOW
    - Sometimes you want to authorize servers to access data without interactively logging in each time the servers exchange information. For these cases, you can use the OAuth 2.0 JSON Web Token (JWT) bearer flow. This flow uses a certificate to sign the JWT request and doesn't require explicit user interaction. However, this flow does require prior approval of the client app.
    - With the OAuth 2.0 JWT bearer token flow, the client posts a JWT to the Salesforce OAuth token endpoint. Salesforce processes the JWT, which includes a digital signature, and issues an access token based on prior approval of the app.
    - Step of this authorization flow.
        - Create a JWT
        - Request Access Token
        - Salesforce Grants Access Token
        - Access Protected Data



- SOAP AUTHENTICATION
  - Use the login() call to log in to the login server and start a client session. The client app logs in and obtains a sessionId and server URL before making other API calls.
  - When a client app invokes the login() call, it passes in a username and password as credentials. Upon invocation, the API authenticates the credentials. It then returns the sessionId, the user ID associated with the logged-in username, and a URL that points to the Lightning Platform API to use in all subsequent API calls.
  - After logging in, make sure that your client app performs these tasks.
  - Sets the session ID in the SOAP header so that the API can validate subsequent requests for this session.
  - Specifies the server URL as the target for subsequent service requests. The login server supports only login calls.
  - The limit is 3,600 calls to login() per user per hour. Exceeding this limit results in a "Login Rate Exceeded" error. After reaching the hourly limit, Salesforce blocks the user from logging in. Users can try to log in again an hour after the block occurred.

References

- https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_quickstart_intro.htm
- https://developer.salesforce.com/docs/atlas.en-us.api.meta/api/sforce_api_calls_login.htm