Authorization in HTTP using OAuth 2.0

Information Security – Lecture 11

Aadil Zia Khan

Authorization vs Authentication

- Authorization
  - What you can do
  - Achieved through access control
- Authentication
  - Who are you
  - Achieved through symmetric or asymmetric digital signatures

Trust Issues in Authorization

- A game app, “Ludo”, needs to access Bob’s Facebook friends list – how will he grant access?
- Insecure option
  - Ludo app will ask Bob to provide his Facebook login information
  - It will use those credentials to access his friends list on Facebook
- See any problems???
  - Bob is blindly trusting a third party app
  - It can sell our information/credentials to malicious users
  - It can access Bob’s pictures and other private information, or write on the wall, unauthorized
  - How will Bob cancel authorization later?

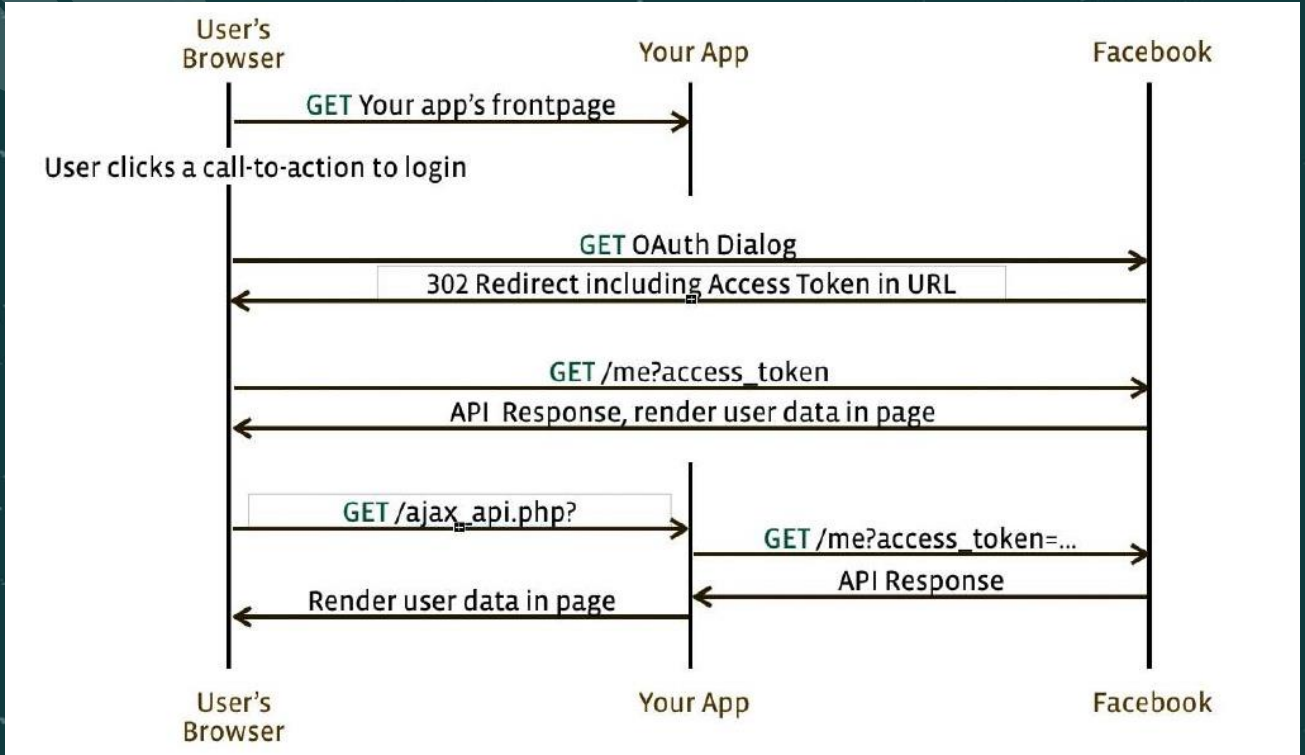
A More Secure Option

- Ludo will request Facebook to provide access to Bob's friends list
- Facebook will confirm with Bob
- Bob will allow Facebook to grant access to Ludo
- Facebook will share a JSON Web Token with the Ludo app
    - Each time it needs to access the list, it can send the token to Facebook

OAuth 2.0

- OAuth 2.0 is the industry-standard protocol for authorization
- OAuth aims to simplify the process of providing authorization to clients to access secured resources
- OAuth is only for authorization – for authentication it uses an OpenId Connect extension

OAuth, Facebook, and 3rd Party Apps



Roles in OAuth

- User – owner of the resource (e.g., photos, friend list on Facebook)
- Resource Server – provides the API to access user's resources
- Client - application that needs user's permission to access user's resources
- Authorization Server – using permission from the user, allows the client to access the API to obtain user resources

Usecases

- Web server apps
- Browser based apps
- Username/password access
- Application access
- Mobile apps

OAuth 2.0 Grants

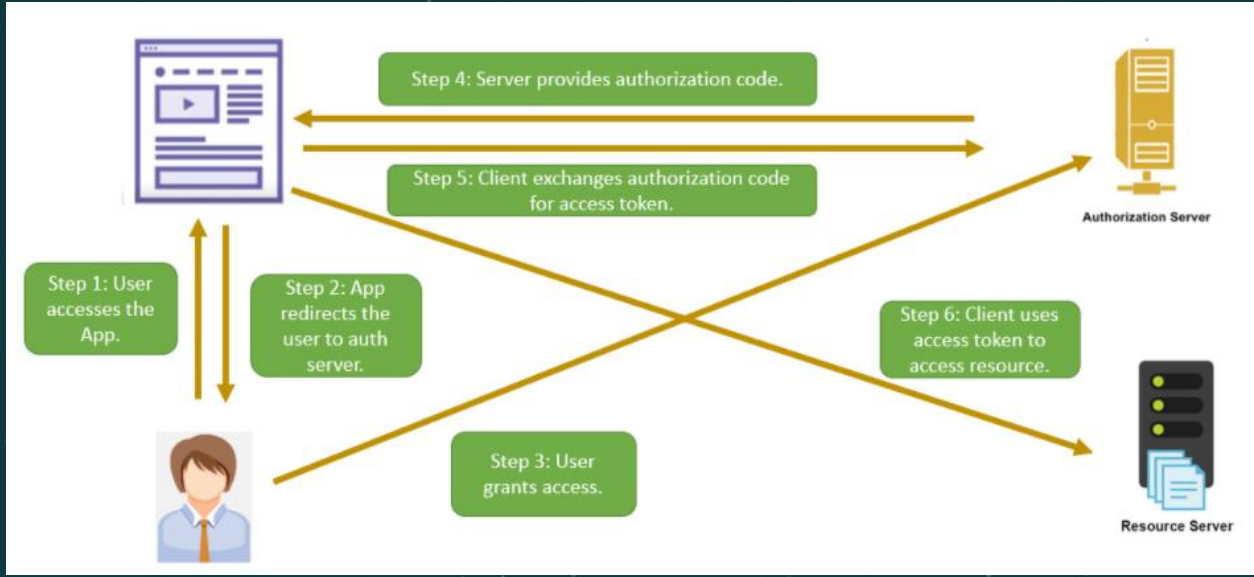
- Grants refer to the way an application gets an access token – different types of grants are suited for a particular use
- There are many kinds of grants including:
  - Authorization Code
  - Implicit
  - Password
  - Client Credentials
  - Device Code
  - Refresh Token

Login Link Format

https://facebook.com/dialog/oauth?
response_type={some value}
&client_id={some value}
&redirect_uri={some value}
&scope={some value}

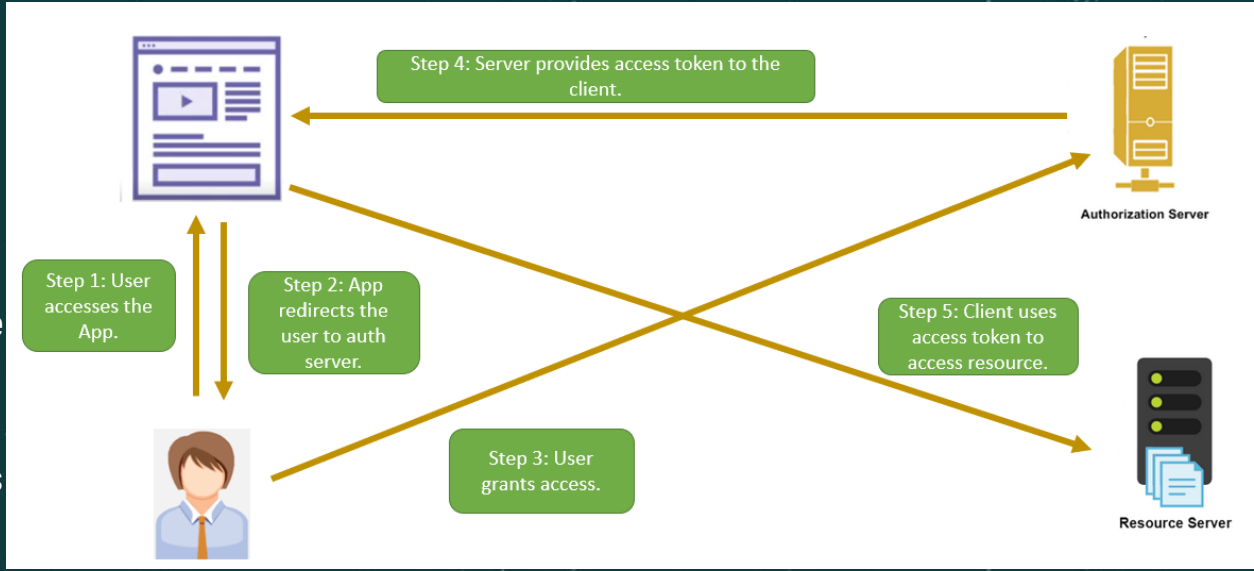
OAuth 2.0 Grants: Authorization Code Grant

- Step 5 requires a client secret key so that token can be exchanged securely



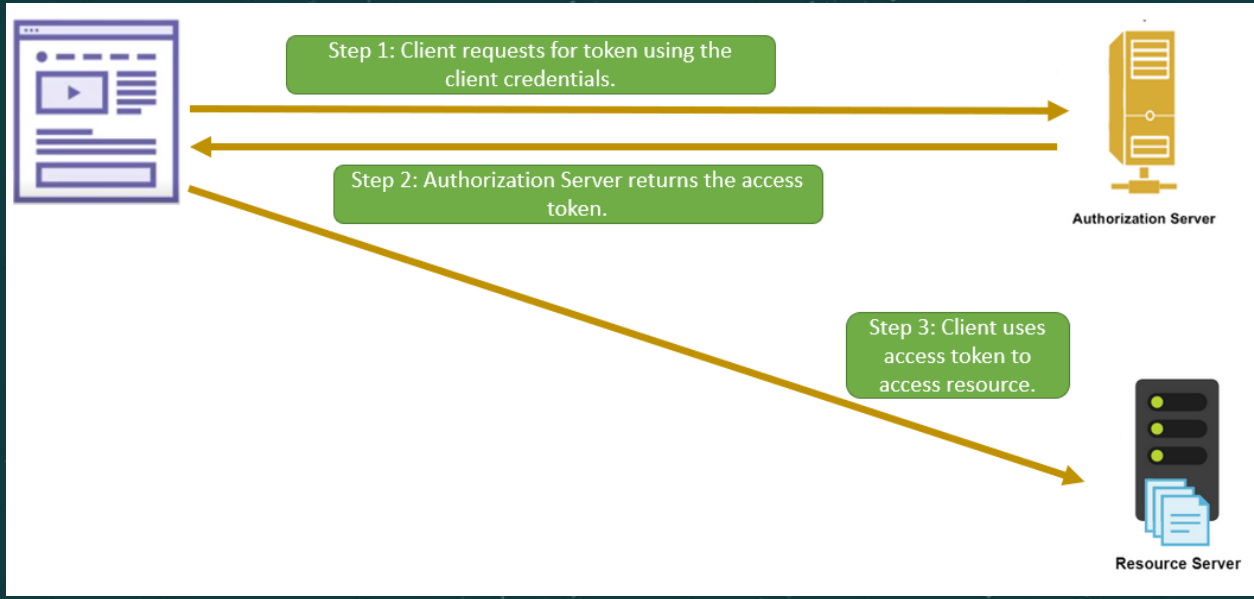
OAuth 2.0 Grants: Implicit Grant

- This is different from Authorization Code Grant - step 5 which required a client secret key is missing here
- Why???
  - This approach is used where the client secret key can not be stored securely at the client – e.g., client is a mobile app and is can be easily decompiled to determine the key's value



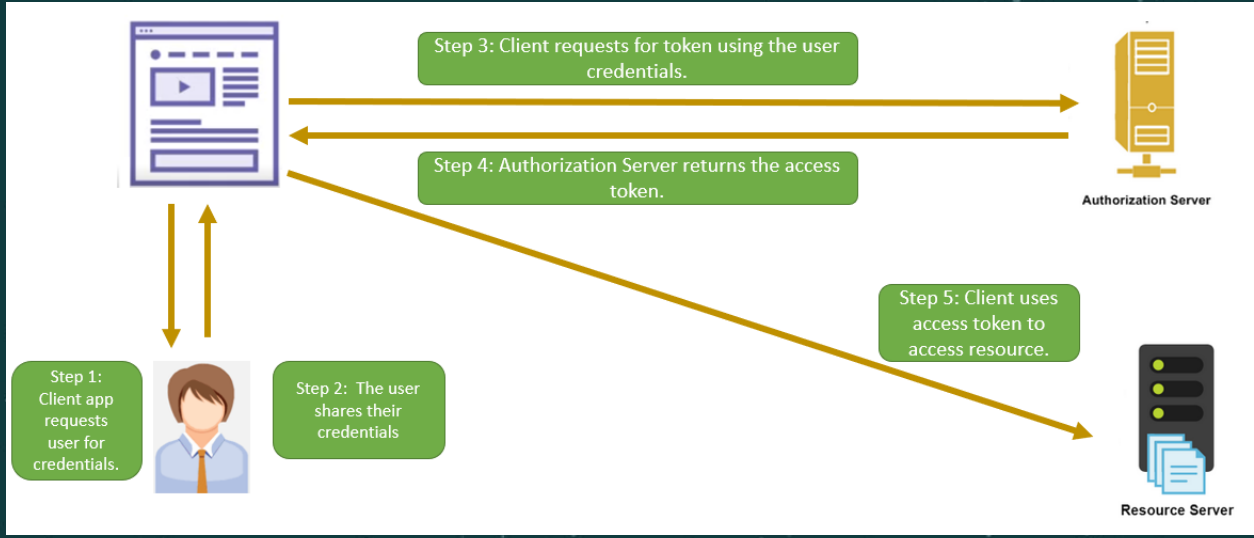
OAuth 2.0 Grants: Client Credentials Grant

- The app already has access rights so it doesn't need to involve the user
  - For example when the app and resources belong to the same company



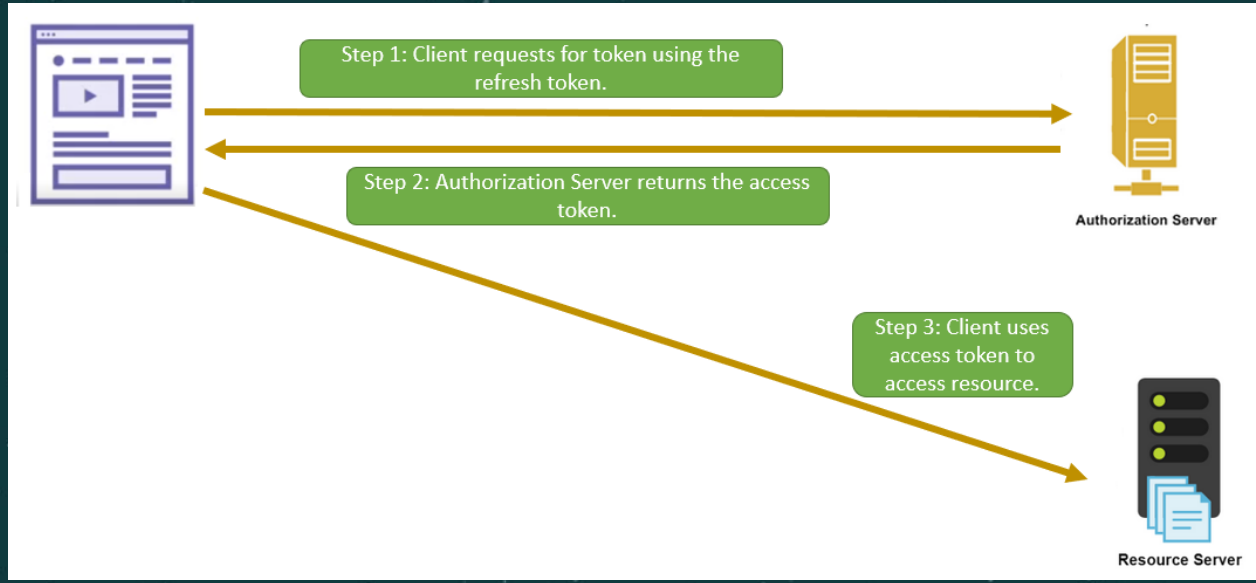
OAuth 2.0 Grants: Resource Owner Credentials Grant

- User shares his username/password with the application – very insecure as the party can use the password in whatever way it chooses
  - Should not be used with 3rd party apps



OAuth 2.0 Grants: Refresh Token Grant

- A token needs to be refreshed on expiry - since the application already has a valid token, refreshing it does not involve the user



Using the Access Token to Access Resources

- There are two ways of using the access token
  - Place it inside the HTTP header as follows

    GET https://api.testing.com/me

    Authorization: Bearer {access token value}
  - Place it in the URL as a query string as follows

    https://api.testing.com/me?access-token= {access token value}

OAuth Scopes

- Scope is a mechanism in OAuth 2.0 to limit an application's access to a user's account
- An application can request one or more scopes, this information is then presented to the user in the consent screen, and the access token issued to the application will be limited to the scopes granted
- OAuth does not define any particular values for scopes, since it is highly dependent on the service's internal architecture and needs.

Scopes Example - FitBit

Scope	Description
activity	It includes activity data and exercise log features, such as steps, distance, calories burned, and active minutes
heartrate	It includes the continuous heart rate data and related analysis
location	It includes the GPS and other location data
nutrition	It includes calorie consumption and nutrition related features, such as food/water logging, goals, and plans
profile	It is the basic user information
settings	It includes user account and device settings, such as alarms
sleep	It includes sleep logs and related sleep analysis
social	It includes friend-related features, such as friend list, invitations, and leaderboard
weight	It includes weight and related information, such as body mass index, body fat percentage, and goals

OpenID Connect

- OAuth is only for authorization – for authentication it uses an OpenId Connect extension
- OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol
  - It allows verification of the identity of an end-user based on the authentication performed by an authorization server
  - It also allows end nodes to obtain basic profile information about the end-user
- In technical terms, OpenID Connect specifies a RESTful HTTP API, using JSON as a data format