Virtuoso Infotech Pvt. Ltd.

virtuoso

INFOTECH

OAuth 2.0

History

If a third party wanted access to an account, you’d give them your password.

Problems

- Apps store the user’s password.
- Apps get complete access to a user’s account.
- Users can’t revoke access to an app except by changing their password.
- Compromised apps might expose user’s password.
- Many services implemented things similar to OAuth 1.0.
- Each implementation was slightly different, certainly not compatible with each other.

What is OAuth?

- OAuth stands for “Open Authorization” .
- An open standard protocol that provides simple and secure authorization for different types of applications.
- A simple and safe method for consumers to interact with protected data.
- Allows providers to give access to users without any exchange of credentials Designed for use only with HTTP protocol.

Why OAuth?

- OAuth is created by studying each of the proprietary protocols.
- It is flexible, compatible and designed to work with all applications
- Provides a method for users to grant third-party access to their resources without sharing their credentials.
- Provides a way to grant limited access in terms of scope and duration.

Difference between OAuth 1.0 and OAuth 2.0

- More OAuth Flows to allow better support for non-browser-based applications.
- OAuth 2.0 no longer requires client applications to have cryptography.
- OAuth 2.0 signatures are much less complicated.
- OAuth 2.0 Access tokens are "short-lived".
- OAuth 2.0 is meant to have a clean separation of roles.

OAuth 2.0 flow

Basic Concepts

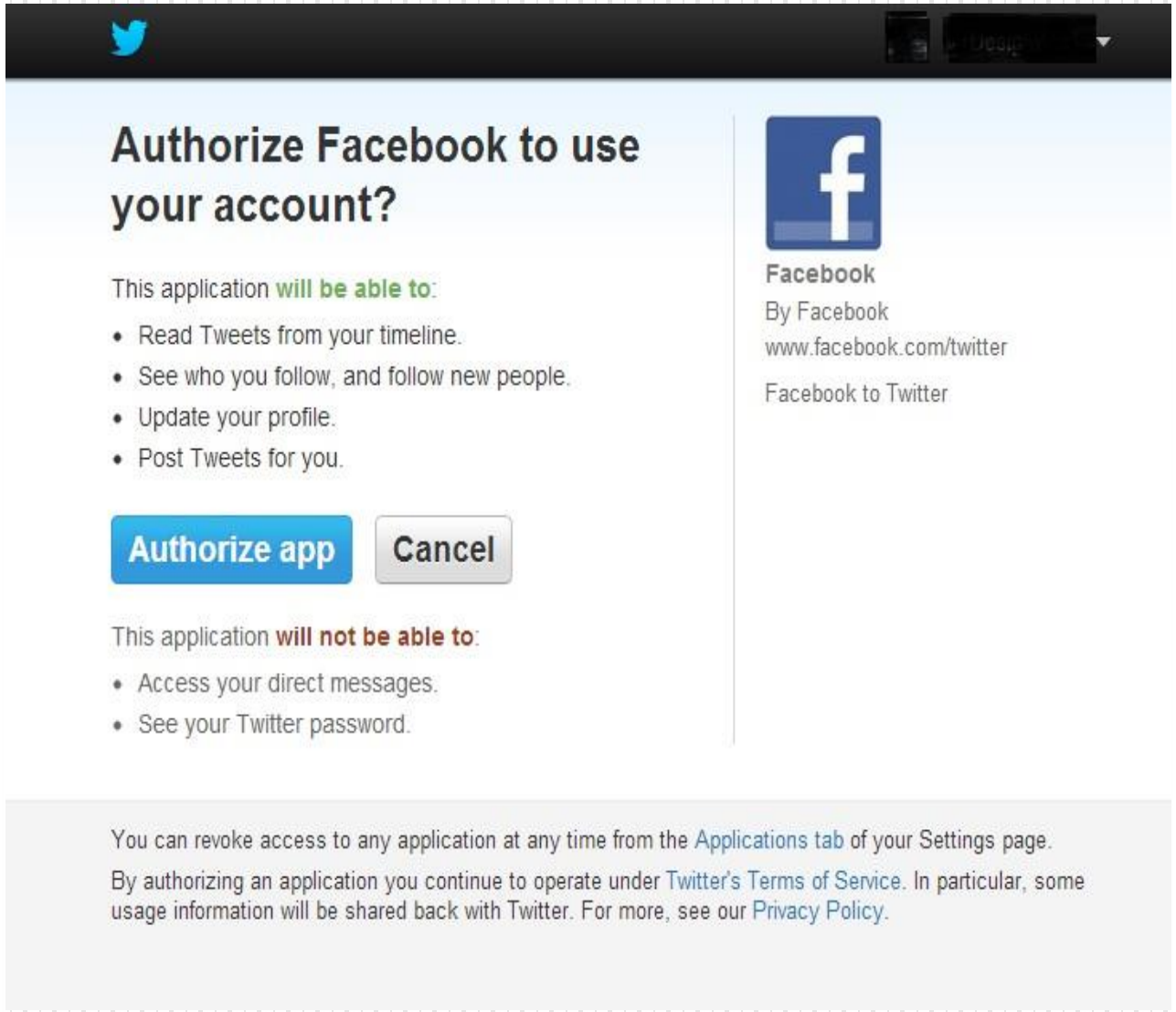
Roles

OAuth defines four roles:

- Resource owner (the user)
- Resource server (the API):must be able to accept and validate access tokens and grant the request.
- Authorization server: Shows Auth prompt, grants access token etc.
- Client (the third-party app):
  1. Confidential Clients(web apps)
  2. Public Clients

Scopes

Permissions asked by client when requesting a token.



Tokens

Access Token (Required)

- Short- lived token used by Client to access Resource Server (API)
- No client authentication required (Public Clients)
- Usually can’t be revoked

Refresh Token (Optional)

- Long- lived token that is used by Client to obtain new access tokens from Authorization Server.
- Usually requires Confidential Clients with authentication
- Can be revoked

Client ID

- The client_id is a public identifier for apps.
- It’s best that it isn’t guessable by third parties.
- Implementations use something like a 32-character hex string.
- It must also be unique across all clients.

Client Secret

- The client_secret is a secret known only to the application and the authorization server.
- It must be sufficiently random to not be guessable.
- Generate a secure secret by using 256-bit value and converting it to a hexadecimal representation.

Grant Types

- Web-server apps – authorization_code
- Browser-based apps – implicit
- Username/password access – password
- Application access – client_credentials
- Mobile apps – implicit

Web Server Apps - Authorization Code Grant

Create a “Log In” link

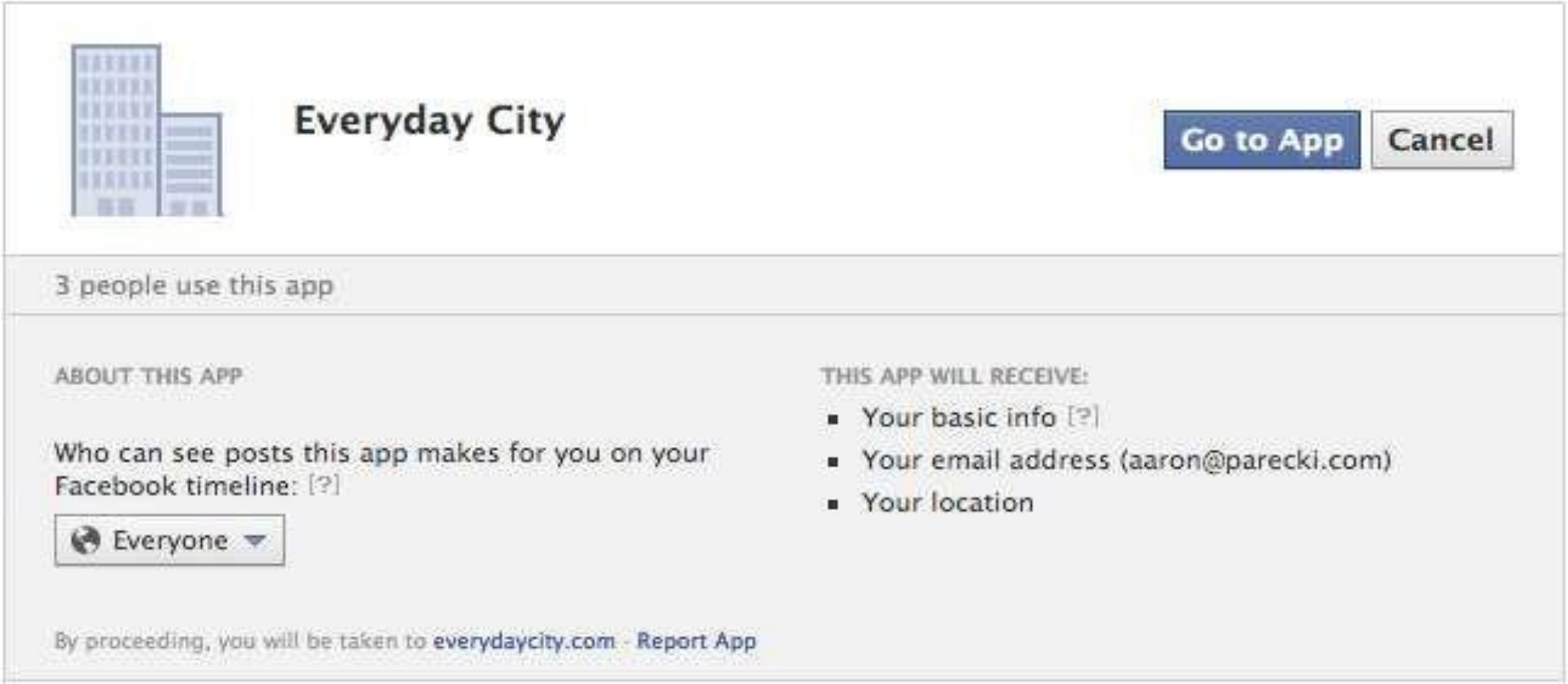
Link to:

https://facebook.com/dialog/oauth?response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=REDIRECT_URI&scope=email

Authorization Grant Parameters

- client_id:It is the identifier for your app
- response_type: is set to code indicating that you want an authorization code as the response.
- redirect_uri (optional):This is the URL to which you want the user to be redirected after the authorization is complete.
- scope (optional):Include one or more scope values to request additional levels of access.
- state (recommended):The state serves as a parameter.

User visits the authorization page



Continue..

- On success, user is redirected back to your site with auth code.

  https://example.com/auth?code=AUTH_CODE_HERE

- On error, user is redirected back to your site with error code.

  https://example.com/auth?error=access_denied

Server exchanges auth code for an access token

- Your server makes the following request

POST

https://graph.facebook.com/oauth/access_token

Post Body:

grant_type=authorization_code&code=CODE&redirect_uri=REDIRECT_URI&client_id=YOUR_CLIENT_ID
&client_secret=YOUR_CLIENT_SECRET

Exchanging code for an access token

- Your server gets a response like the following

{
"access_token":"RsT5OjbzRn430zqMLgV3Ia", "token_type":"bearer", "expires_in":3600, "refresh_token":"e1qoXg7Ik2RRua48lXIV"
}

- or if there was an error

{
"error":"invalid_request"
}

Browser-Based Apps - Implicit Grant

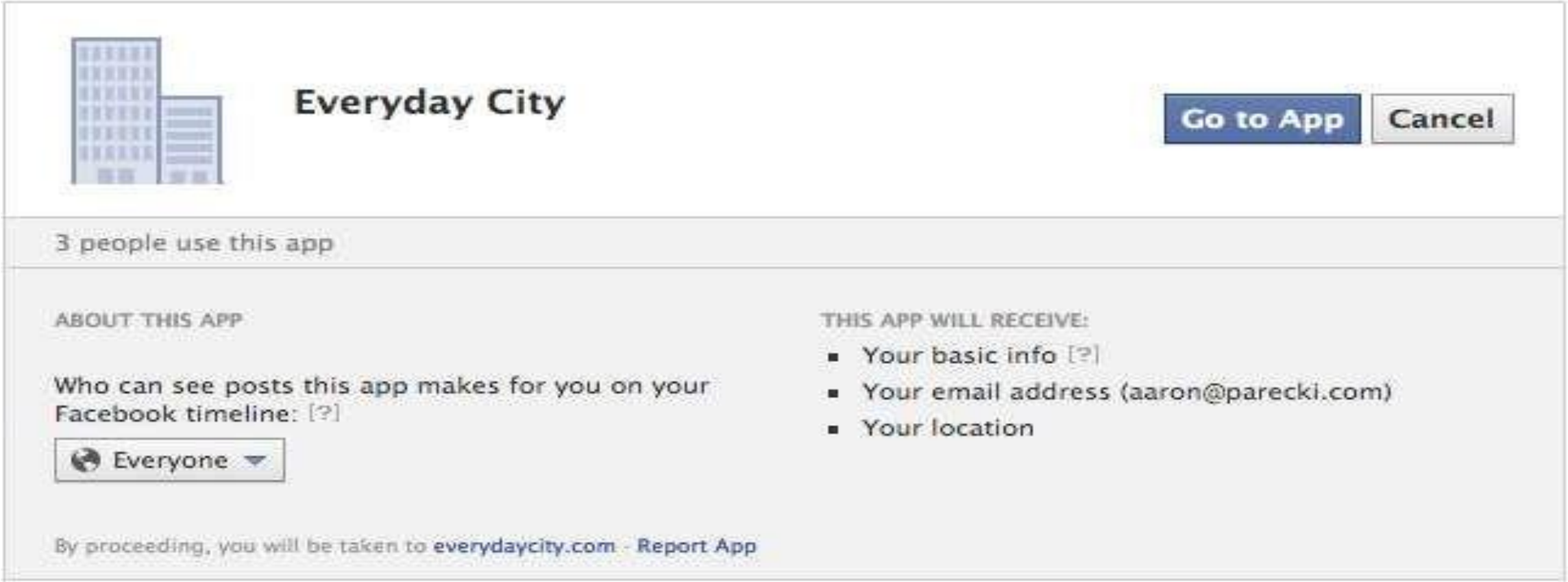
Create a “Log In” link

Link to:

https://facebook.com/dialog/oauth?response_type=token&client_id=CLIENT_ID_&redirect_uri=REDIRECT_URI&scope=email

User visits the authorization page

https://facebook.com/dialog/oauth?response_type=token&client_id=2865368247587&redirect_uri=everydaycity.com&scope=email



Continue..

- On success, user is redirected back to your site with the access token in the fragment

https://example.com/auth#token=ACCESS_TOKEN

- On error, user is redirected back to your site with error code

https://example.com/auth#error=access_denied

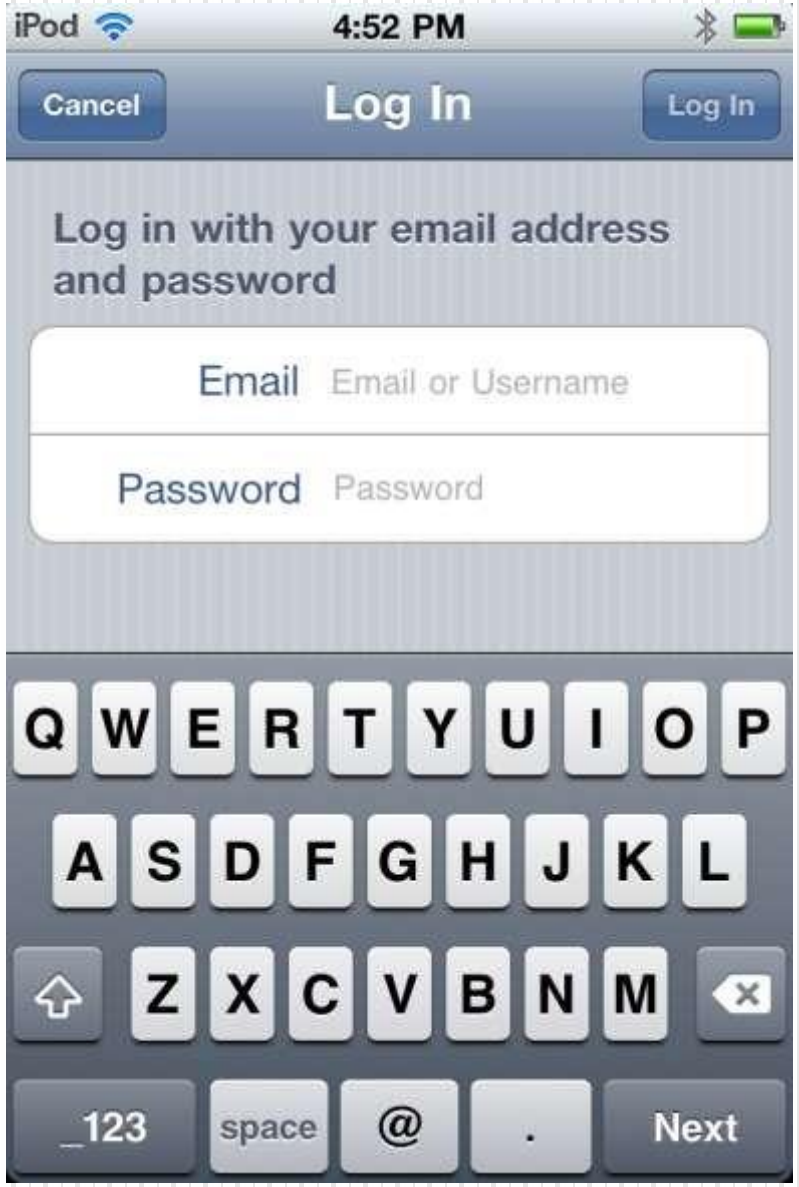
Browser-Based Apps

- Use the “Implicit” grant type
- No server-side code needed
- Client secret not used
- Browser makes API requests directly

Username/Password - Password Grant

Password Grant

- For trusted clients only (first-party apps).
- Only appropriate for your service's website or your service's mobile apps.



Continue..

POST

https://api.example.com/oauth/token

Post Body:

grant_type=password&username=USERNAME&password=PASSWORD&client_id=YOUR_CLIENT_ID&client_secret=YOUR_CLIENT_SECRET

Response:

{
"access_token":"RsT5OjbzRn430zqMLgV3Ia","token_type":"bearer,
"expires_in":3600, "refresh_token":"e1qoXg7Ik2RRua48lXIV"
}

Application Access - Client Credentials Grant

Client Credentials Grant

POST
https://api.example.com/1/oauth/token

Post Body:
grant_type=client_credentials&client_id=CLIENT_ID&client_secret=YOUR_CLIENT_SECRET
Response:


```
{
   "access_token":"RsT5OjbzRn430zqMLgV3Ia",
   "token_type":"bearer",  "expires_in":3600,
   "refresh_token":"e1qoXg7Ik2RRua48IXIV"
}
```

Mobile Apps - Implicit Grant


Carrier
9:05 AM
Everyday City
Automatically post your current city to Facebook
Login with Facebook
Photo credit
flickr:kennymatic


Sprint
8:14 PM
100%
Cancel
facebook
Okay
Everyday City
This app may post on your behalf, including cities you arrived in and more.
About this app
You are logging into this app as Aaron Parecki.
By proceeding, you also agree to Everyday City's Terms of Service and Privacy Policy.

Redirect back to your app

- Facebook app redirects back to your app using a custom URI scheme.
- Access token is included in the redirect, just like browser-based apps.

fb2865://authorize/#access_token=BAAEEmo2nocQBAFFOeRTd


Carrier
9:10 AM
Everyday City
Current City
Portland
Everyday City runs in the background and automatically posts your current city to Facebook!
Even though you will see the location services icon in your phone's menu, the app is monitoring your location in a battery-safe way.
Automatic Updating
Off
On
Logout
Powered by
Geoloqi
History
Current City

Mobile Apps

- Use the “Implicit” grant type
- No server-side code needed
- Client secret not used
- Mobile app makes API requests directly

Making Authenticated Requests

- There are two ways API servers may accept Bearer tokens.
  1. As a Header parameter.
  2. As a Body parameter.
- Passing in the access token in an HTTP header:

```
POST /resource/1/update HTTP/1.1
Authorization: Bearer RsT5OjbzRn430zqMLgV3Ia"
Host: api.authorization-server.com
description=Hello+World
```

Continue..

- If the service accepts access tokens in the post body, then you can make a request like the following:

```
POST /resource/1/ HTTP/1.1
Host: api.authorization-server.com
access_token=RsT5OjbzRn430zqMLgV3Ia
&description=Hello+World
```

Common OAuth 2.0 Security Issues

- Too many inputs that need validation

  1. Token hijacking with CSRF
     - Always use CSRF token with state parameter .Leaking authorization codes or tokens through redirects
     - Always whitelist redirect URIs and ensure proper URI validations

  2. Token hijacking by switching clients
     - Bind the same client to authorization grants and token requests

- Leaking client secrets

Thank You!

Virtuoso InfoTech Pvt. Ltd.
4th Floor, Victory Landmark, Opp. D-Mart, Behind Dominos Pizza,
Baner, Pune.

+91 80870 81318
support@virtuositech.com

www.virtuositech.com