

u2u U2U Online

online.u2u.be/courses/25896/modules/6/27314

Developer and IT Training

u2u

OAuth and OpenID Connect

26°C Mostly sunny

Search

1

The screenshot shows a web browser window with the following details:

- Title Bar:** U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/6/27314
- Left Sidebar (User Profile):** Shows "u2u Online" and "My profile" (Maxime@protonmail.com). Other options include "Your courses" and "Send this course invite".
- Page Content:** A slide titled "Agenda" with a red horizontal line below it. The agenda items are:
 - The internet and a way of sharing
 - Introducing OAuth2
 - OpenID Connect: Adding Sign-in to OAuth2
 - Other OAuth2 flows
 - OAuth2 and OpenID Connect web sign-in
- Right Sidebar (u2u logo):** A large "u2u" logo.
- Bottom Taskbar:** Shows the date (12/29/2023), time (3:19 PM), battery level (mostly charged), signal strength, and language (ENG).
- Bottom Weather Bar:** Shows the weather forecast: 26°C Mostly sunny.

u2u U2U Online

online.u2u.be/courses/25896/modules/6/27314

Authentication and Authorization on the Internet

- Simple fact: The internet is here to stay
 - And we need security
- A way of sharing things
 - We want to combine stuff, like post on my  wall whatever I 
 - How can I allow twitter to post on my Facebook wall??



26°C Mostly sunny 3:19 PM 12/29/2023

U2U Online

online.u2u.be/courses/25896/modules/6/27314

Home > Data Security Fundamentals Techniques > OAuth and OpenID Connect

OAuth and OpenID Connect

- OAuth and OpenID Connect
- Introducing OAuth2
- OAuth Client Flow
- OpenID Connect: Adding Identity to OAuth2

u2u

Password sharing?

Noooooo

- Simple idea – huge anti-pattern

The diagram shows two rectangular boxes labeled 'A' and 'B'. A small figure of a person is pointing from box 'B' up towards box 'A'. A red arrow originates from the bottom of box 'A' and points to the right. Above this arrow, the text 'I need to talk to B, can I have your password?' is written in red. Below the arrow, the text 'Sure, here it is...' is also written in red.

26°C
Mostly sunny

Search

3:19 PM 12/29/2023

u2u Online

online.u2u.be/courses/25896/modules/6/27314

Introducing OAuth2

- OAuth2 allows third parties to request and obtain access to your resources
 - On behalf of the user, who can grant or deny access to operations on those resources
 - As a server, OAuth2 allows you to expose your resources for delegated access
- OAuth2 allows an application to request access to a user's resources
 - Stored on another server
 - And allows an application to access those granted resources

26°C
Mostly sunny

Search

3:19 PM

U2U Online

online.u2u.be/courses/25896/modules/6/27314

Simplified OAuth2 flow between web applications **u2u**

The diagram illustrates the OAuth2 flow between two web applications, A and B, and an Authorization Server.

Participants:

- Application A:** Represented by a red-bordered box labeled 'A'.
- Application B:** Represented by a red-bordered box labeled 'B'.
- Browser:** Represented by a red-bordered box containing a person icon.
- Authorization Server:** Represented by a red-bordered box labeled 'Authorization Server'.

Flow:

1. Please do something: Application A sends a request to the Browser.
2. Redirect to AS: The Browser receives a redirect from Application A to the Authorization Server.
3. "Can I?" get code: The Browser sends a request to the Authorization Server asking for a code.
4. Code: The Authorization Server returns a code to the Browser.
5. Code + App Credentials: The Browser sends the code and application credentials to Application A.
6. Application A receives the code and app credentials.
7. Key: Application A uses the code and app credentials to unlock a chest (represented by a key icon) containing a treasure chest (represented by a chest icon).

Most common OAuth Grant Flows

- Authorization Code Grant
 - Useful for web app executing on the server and natively installed apps
- Proof Key for Code Exchange
 - Improved Security for Public Clients (JavaScript and Native applications)
- Resource owner credentials grant
 - Only useful if the client app is absolutely trusted with the user credentials
- Client credentials grant
 - Useful for unattended apps like services, server to server communication,...
- Refresh token grant
 - Used to refresh an access token when it expires

U2U Online

online.u2u.be/courses/25896/modules/6/27314

Authorization code flow

```

graph LR
    Native[Native] -- "1. Please do something" --> A[A]
    Native -- "2. Redirect to AS" --> AS[Authorization Server]
    AS -- "3. Auth code?" --> Native
    Native -- "4. Auth Code" --> A
    A -- "5. Auth Code + Credentials" --> B[B]
    B -- "7. Key" --> A
    
```

The diagram illustrates the Authorization code flow between three components: Application A, Native Client, and Authorization Server.

- Application A:** Represented by a red-bordered box labeled 'A'.
- Native Client:** Represented by a red-bordered box labeled 'Native'.
- Authorization Server:** Represented by a red-bordered box labeled 'Authorization Server'.

The flow of steps is as follows:

1. Please do something (Native to Application A)
2. Redirect to AS (Native to Authorization Server)
3. Auth code? (Authorization Server to Native)
4. Auth Code (Native to Application A)
5. Auth Code + Credentials (Native to Application A)
6. Auth Code + Credentials (Native to Authorization Server)
7. Key (Application A to Native)

Visual elements include a treasure chest icon near Application A and a key icon near the Native Client.

Authorization Code Grant (A)

- Redirect user to Authorization Endpoint

`https://service.domain/authorization_endpoint?` keyword

`response_type=code&` optional (registered url will be taken)

`client_id=<<my client id>>&`

`redirect_uri=https://my.local.domain/resource&`

`scope=api,profile,...&`

`state=<<CSRF Token>>` Depends on the application

optional (CSRF token stored in User Session state and checked in web app)

The screenshot shows a browser window with the URL `online.u2u.be/courses/25896/modules/6/27314`. The main content area displays a slide titled "Authorization Code Grant (A)" with a list of steps for OAuth and OpenID Connect. A red box highlights the URL parameters for the authorization endpoint, and red arrows point from the annotations to specific parts of the URL string.

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/6/27314. The page title is "Authorization Code Grant (B)". On the left, there's a sidebar with user information and course navigation. The main content area contains a section titled "Authorization Code Grant (B)" with a bullet point: "Redirect user back to Web App". Below this, a code snippet is shown in a red-bordered box:

```
https://my.Local.domain/resource?  
code=<<the authorization code>>&  
state=<<same CSRF Token>>
```

Two red arrows point from the text "Code to be used to fetch access tokens" and "CSRF token returned from Authorization Server to be checked in Web App" to the "code=" and "state=" lines respectively.

At the bottom of the screen, the Windows taskbar shows the date and time as "3:19 PM 12/29/2023".

Authorization Code Grant (C)

- Get the Access token from the Token Endpoint

```
POST https://service.domain/token_endpoint

grant_type=authorization_code&
client_id=<<client id>>&
client_secret=<<my client secret>>&
redirect_uri=<<same redirect uri as before>>&
code=<<the authorization code>>
```

- Returns JSON object with

```
{  
    "token_type" : 'Bearer',           // Most of the time Bearer is used  
    "expires_in" : <<TTL>>,          // integer representing the expiration time  
    "access_token": <<The access token>>,  
    "refresh_token": <<The refresh token>>  
}
```

U2U Online

online.u2u.be/courses/25896/modules/6/27314

Proof Key for Code Exchange Flow

The diagram illustrates the Proof Key for Code Exchange Flow. It shows the interaction between three components:

- SPA (Single Page Application):** Represented by a large red-bordered box containing a cartoon character.
- Authorization Server:** Represented by a red-bordered box containing the text "Auth endpoint" and "Token endpoint".
- Backend B:** Represented by a red-bordered box containing a treasure chest icon.

The flow of the proof key for code exchange is as follows:

1. Generate code challenge + code verifier
2. Auth request + code challenge
3. Authorization code
4. Token request (auth code + code verifier)

Arrows indicate the direction of data flow between the SPA and the Authorization Server. A key icon is shown near the SPA and Backend B boxes, representing the proof key used in step 4.

PKCE Flow

- For public clients, Authorization Code Grant is vulnerable to an authorization code interception attack
- PKCE uses a code challenge and code verifier to prevent authorization code interception attacks

```

sequenceDiagram
    participant EndDevice as End Device (e.g., Smartphone)
    participant MaliciousApp as Malicious App
    participant LegitimateApp as Legitimate OAuth 2.0 App
    participant OS as Operating System/Browser
    participant AuthzServer as Authz Server

    Note left of EndDevice: End Device (e.g., Smartphone)
    Note left of MaliciousApp: Malicious App
    Note left of LegitimateApp: Legitimate OAuth 2.0 App
    Note left of OS: Operating System/Browser
    Note left of AuthzServer: Authz Server

    LegitimateApp->>OS: (1) Authz Request
    OS-->>AuthzServer: (2) Authz Request
    Note right of OS: Operating System/Browser
    Note right of AuthzServer: Authz Server

    MaliciousApp-->>OS: (3) Authz Code
    OS-->>MaliciousApp: (4) Authz Code
    Note right of OS: Operating System/Browser
    Note right of MaliciousApp: Malicious App

    AuthzServer-->>EndDevice: (5) Authorization Grant
    EndDevice-->>AuthzServer: (6) Access Token
    Note right of EndDevice: End Device (e.g., Smartphone)
    Note right of AuthzServer: Authz Server
  
```

The diagram illustrates the PKCE flow. It shows the interaction between an End Device (Smartphone), Malicious App, Legitimate OAuth 2.0 App, Operating System/Browser, and Authz Server. The flow starts with the Legitimate App sending an Authorization Request (1) to the Operating System/Browser. The OS/Browser then sends an Authorization Request (2) to the Authz Server. The Authz Server returns an Authorization Grant (5) to the End Device. The End Device then sends an Access Token (6) back to the Authz Server. A Malicious App is shown intercepting the Authorization Code (3) sent from the OS/Browser to the Malicious App, and the Malicious App then sends its own Authorization Code (4) back to the OS/Browser.

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/6/27314. The slide title is "PKCE: code challenge and code verifier". On the left, there's a sidebar with navigation links like "My profile", "My courses", and "Logout". The main content area contains a bulleted list explaining the PKCE process. At the bottom, there's a taskbar with weather information (26°C, Mostly sunny), a search bar, and various application icons.

PKCE: code challenge and code verifier

- Client generates a code verifier (random string) and code challenge (hash of code verifier)
- Client sends code challenge with auth request, Auth server keeps track of it
- When Client sends code verifier with token request, Auth server can generate a new code challenge based on it, and compare it to the existing code challenge
- If the received code challenge and the generated code challenge match, this proves ownership for the client

U2U Online

online.u2u.be/courses/25896/modules/6/27314

Resource Owner Credentials Grant Flow

1. Login

2.

3.

SPA

B

Authorization Server

15

The screenshot shows a web browser window with the following details:

- Title Bar:** U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/6/27314
- Left Sidebar:** A dark sidebar with a user profile picture, the text "U2U Online", and a "Logout" button.
- Content Area:**
 - A large title "Resource Owner Credentials Grant" is centered at the top.
 - To the right of the title is a "u2u" logo.
 - The main content area contains two bullet points:
 - Great for first-party trusted clients
 - Twitter app to use Twitter
 - Client app shows login screen to user
 - User logs in with credentials for the Resource
- Bottom Taskbar:** Shows system icons for weather (26°C, Mostly sunny), search, file explorer, and browser (Chrome). It also displays the date and time (3:19 PM, 12/29/2023).

Resource Owner Credentials Grant (A)

- POST login credentials from client app to authorization server

```
POST https://service.domain/login_endpoint

grant_type=password&
client_id=<<client id>>&
client_secret=<<my client secret>>&
scope=these,are,my,scopes&
username=<<my username>>&
password=<<my password>>
```

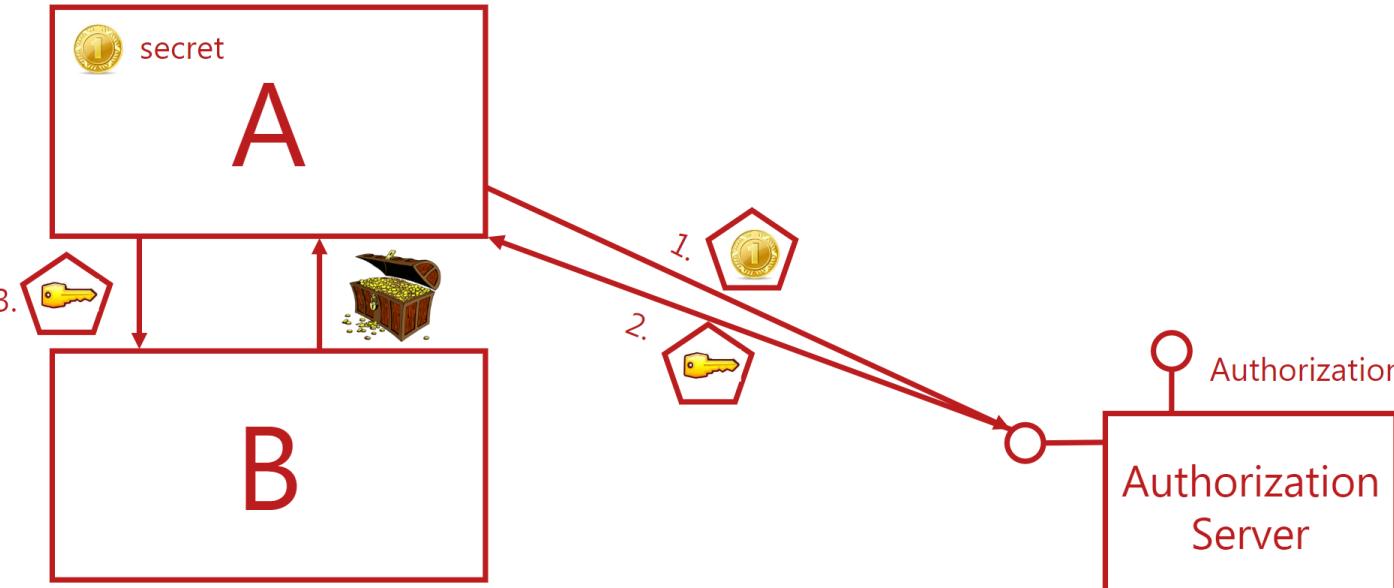
- Returns JSON

```
{
  "token_type" : 'Bearer',           // Most of the time Bearer is used
  "expires_in" : <<TTL>>,          // integer representing the expiration time
  "access_token": <<The access token>>,
  "refresh_token": <<The refresh token>>
}
```

u2u U2U Online

online.u2u.be/courses/25896/modules/6/27314

Client credentials grant flow (Server to Server) **u2u**



The diagram illustrates the Client credentials grant flow between two servers, A and B, and an Authorization Server.

- Server A:** Contains a "secret" (represented by a gold coin icon).
- Server B:** Contains a treasure chest icon.
- Authorization Server:** Contains an "Authorization" label.

The flow consists of three steps:

1. A gold coin icon (representing a token or access grant) is sent from the Authorization Server to Server A.
2. A key icon (representing client credentials) is sent from Server A to the Authorization Server.
3. A key icon (representing client credentials) is sent from Server B to Server A.

There is also a double-headed arrow between Server A and Server B, indicating a bidirectional communication channel.

26°C Mostly sunny 3:20 PM 12/29/2023

Client Credentials Grant

- Machine-to-machine authorization
 - No user consent necessary!

```
POST https://service.domain/login_endpoint

grant_type=client_credentials&
client_id=<<client id>>&
client_secret=<<my client secret>>&
scope=these,are,my,scopes
```

- Returns JSON

```
{  
    "token_type" : 'Bearer',           // Most of the time Bearer is used  
    "expires_in" : <<TTL>>,          // integer representing the expiration time  
    "access_token": <<The access token>>  
}
```

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/6/27314. The page title is "Refresh Token Grant". On the left, there's a sidebar with a user profile and navigation links. The main content area contains a list of bullet points:

- Access tokens expire – if the authorization server provides a refresh token, the refresh token can be used to get a new access token
- The Web App does **not** need user consent to refresh the token
 - It will do a post request in the background

The browser taskbar at the bottom shows various icons and the system tray on the right indicates it's 3:20 PM on 12/29/2023.

Refresh Token Grant (A)

- Post the refresh token to the Token Endpoint

```
POST https://service.domain/token_endpoint

grant_type=refresh_token&
refresh_token=<<the actual refresh token>>&
client_id=<<client id>>&
client_secret=<<my client secret>>&
scope=these,are,my,scopes
```

- Returns JSON

```
{  
    "token_type" : 'Bearer',          // Most of the time Bearer is used  
    "expires_in" : <<TTL>>,           // integer representing the expiration time  
    "access_token": <<The access token>>,  
    "refresh_token": <<The refresh token>> // New refresh token!  
}
```

Refresh Token Grant Flow

The diagram illustrates the Refresh Token Grant Flow. It features two main components: a box labeled 'A' at the top left and a box labeled 'B' at the bottom left. A red arrow points from 'A' to 'B'. To the right of 'B' is a box labeled 'Authorization Server'. A red arrow points from 'B' to the 'Authorization Server' box. Above the 'Authorization Server' box is a circle labeled 'Authorization'. Three pentagonal icons are shown: one with a key icon (labeled 3.), one with a user icon (labeled 1.), and one with a green circular icon (labeled 2.). A red arrow labeled 'Credentials' points from the three icons towards the 'Authorization Server' box. A small treasure chest icon is positioned between the 'B' box and the 'Authorization Server' box.

The screenshot shows a web browser window with the title bar "U2U Online". The address bar contains the URL "online.u2u.be/courses/25896/modules/6/27314". The main content area displays a presentation slide with a red header bar. The slide title is "OpenID Connect: Adding Sign-in to OAuth2". On the left, there is a sidebar with a navigation tree under "OAuth and OpenID Connect". The sidebar includes sections like "OAuth and OAuth2", "Introducing OAuth2", "OAuth Access Token", and "OpenID Connect: Adding Sign-in to OAuth2". A large "u2u" logo is in the top right corner. The slide content lists several bullet points:

- **OAuth2** is all about **authorization**
 - But what about authentication?
- Introducing **OpenID Connect**
 - Extension to OAuth2, formally adding Sign-in capabilities
 - Defines the ID-token to communicate client information
 - Defines a UserInfo endpoint to obtain information about the subject
- **OpenID Connect** extends and supersedes OAuth2
 - It is the superior protocol
 - Always use OIDC instead of OAuth2, even if you only require authorization

The bottom of the screen shows a taskbar with various icons, including a weather widget (26°C, Mostly sunny), a search bar, and system status indicators.

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/6/27314. The page title is "OpenID Connect Endpoints". On the left, there's a sidebar with a navigation tree under "OAuth and OpenID Connect". The main content area contains a list of five bullet points explaining different types of endpoints:

- OAuth2 and OIDC have endpoints, with standards that define how to consume them
- **Authorization endpoint** (IDP)
 - This is where the resource owner logs in
 - Grants authorization for a client application
- **Token endpoint** (IDP)
 - Where the client application exchanges a `clientid`, `clientsecret` and `code` for an access token
- **User Info endpoint** (IDP, OIDC only)
 - To request identity information about the resource owner
- **Redirect endpoint** (RP)
 - This is where the resource owner is redirected to after receiving authorization code

The browser taskbar at the bottom shows various icons and the date/time: 3:20 PM, 12/29/2023.

OpenID Connect Flows

- **Authorization Code Flow**
 - Authorization Code from Authorization Endpoint
 - Tokens come from Token Endpoint
 - For confidential clients
 - Give long-lived access
- **Implicit Flow**
 - Tokens come from authorization endpoint
 - For public clients
 - No long-lived access
- **Hybrid Flow**
 - Authorization Code from Authorization Endpoint
 - Tokens come from Authorization & Token Endpoint
 - For confidential clients
 - Give long-lived access



26°C Mostly sunny

Search

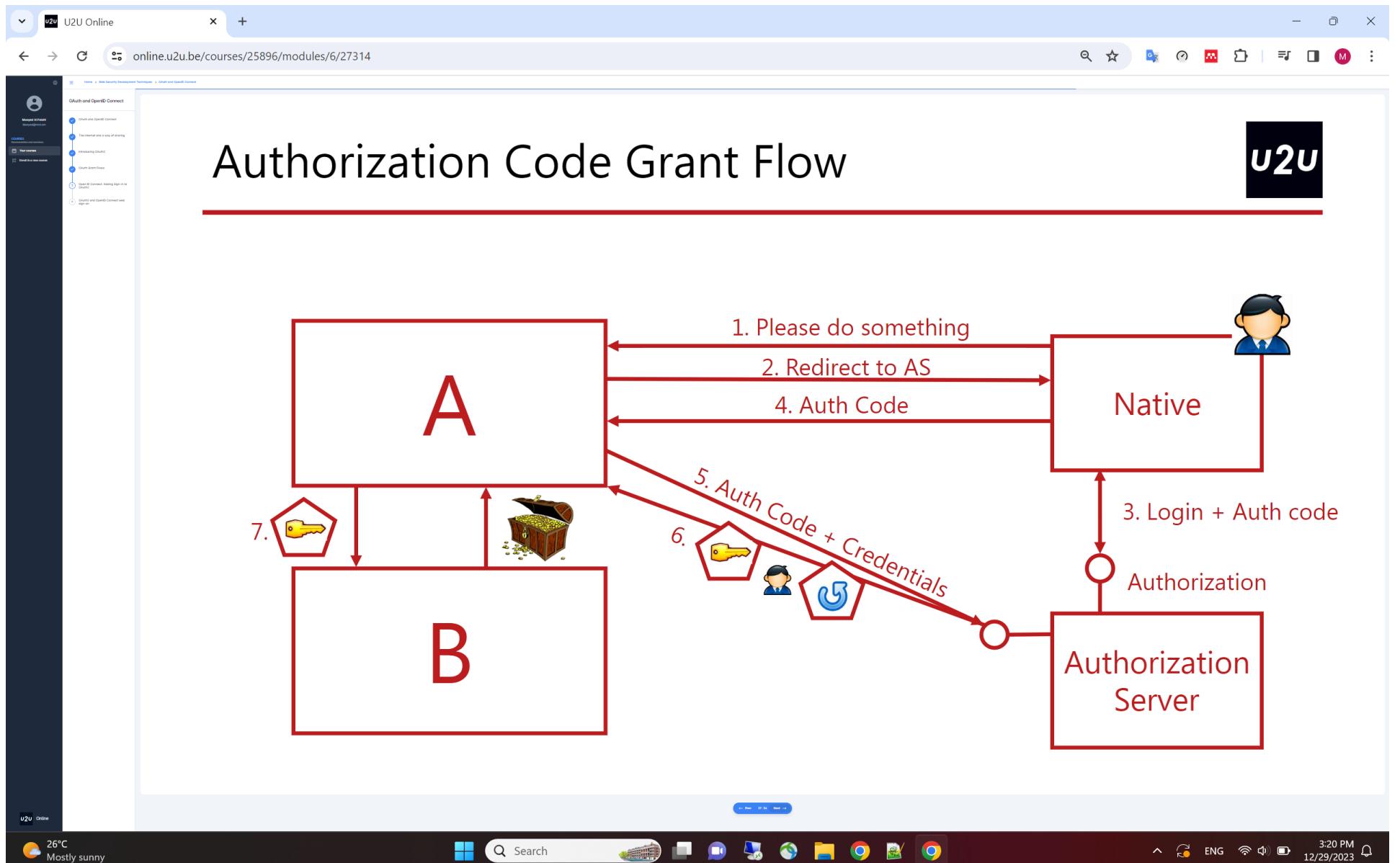
3:20 PM 12/29/2023

The screenshot shows a web browser window with the URL online.u2u.be/courses/25896/modules/6/27314. The page title is "OpenID Connect Flows". On the left, there's a sidebar with user information and course navigation. The main content area contains a list of OpenID Connect flows and a table comparing response types to grant flows.

OpenID Connect Flows

- The Response Types need to change depending on the flow
 - By setting the `response_type` parameter to one of these values
- `token` or `id_token` is used to serve as a user identification token (`jwt`)
- `code` is used to get the authorization code from the server

Response Types	Grant Flow
<code>code</code>	Authorization Code Flow
<code>id_token</code>	Implicit Flow
<code>id_token token</code>	Implicit Flow
<code>code id_token</code>	Hybrid Flow
<code>code token</code>	Hybrid Flow
<code>code id_token token</code>	Hybrid Flow



Implicit Grant Flow

The diagram illustrates the Implicit Grant Flow. It features three main components:

- SPA**: Represented by a red-bordered box containing a person icon.
- B**: Represented by a red-bordered box containing a treasure chest icon.
- Authorization Server**: Represented by a red-bordered box containing an 'A' icon.

The flow of the process is as follows:

- A user (person icon) interacts with the SPA.
- The SPA sends a request to the Authorization Server for an access token.
- The Authorization Server performs an *Authorization* step.
- The SPA receives the access token and uses it to interact with the Backend (B).
- The Backend (B) sends a response back to the SPA.

Key elements in the flow include:

- Login +**: Indicated by a red arrow pointing from the SPA to the Authorization Server.
- Key icons**: Representing the exchange of tokens or keys between the SPA and the Authorization Server.
- Treasure chest icon**: Associated with the Backend (B) component.

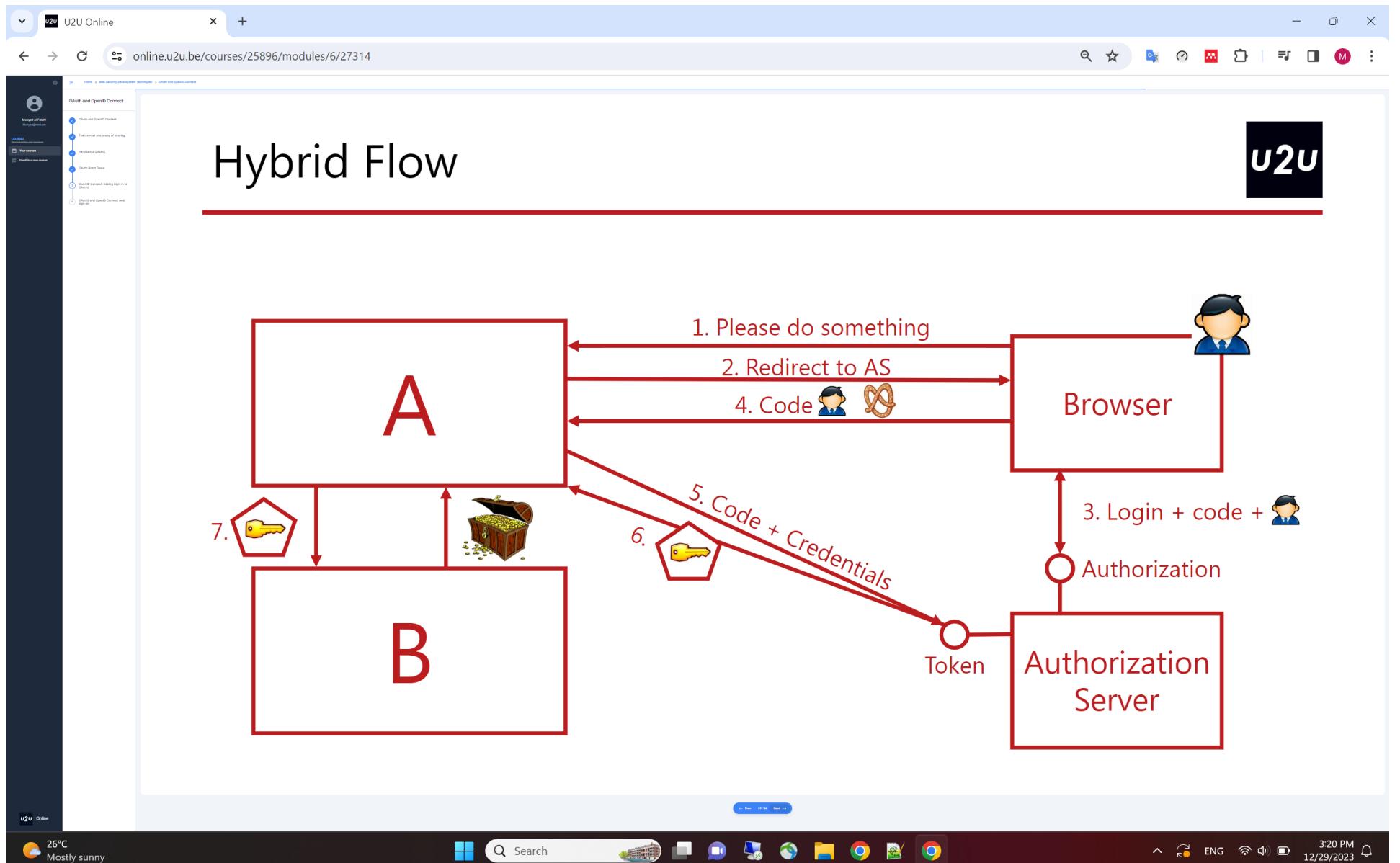
On the left side of the slide, there is a sidebar with the following navigation items:

- Logout (u2u)
- My profile
- My courses
- Your courses
- Send me a message

At the bottom of the slide, there is a navigation bar with the following items:

- Home
- API Security Best Practice Techniques
- OAuth and OpenID Connect
- Implicit Grant Flow
- Implicit Grant Flow - Using OAuth 2.0
- Implicit Grant Flow - Using OpenID Connect
- Implicit Grant Flow - Using OAuth 2.0 and OpenID Connect

On the right side of the slide, there is a large **u2u** logo.



U2U Online

online.u2u.be/courses/25896/modules/6/27314

Using Microsoft Entra ID for web authentication

- Step 1 : Register your application with Microsoft Entra ID
- Step 2 : Add and configure OAuth2 middleware

26°C Mostly sunny

Search

3:20 PM

Step 1 : Register your application with Entra ID

- Open Azure portal, and open Microsoft Entra ID
- Create a new application registration

* Name i

Application type i

Web app / API

* Sign-on URL i

https://localhost:44350/signin-oidc

26°C Mostly sunny 3:20 PM 12/29/2023

OWIN Pipeline Processing Concept

The diagram illustrates the OWIN Pipeline Processing Concept. It shows a horizontal flow labeled "Pipeline" containing four rectangular boxes labeled "Middleware". A red arrow labeled "Request" enters from the left and passes through the first middleware box. From the output of the first box, a red arrow points to the second box. This pattern repeats for the third and fourth boxes. After the fourth box, a red arrow labeled "Response" exits to the left. The pipeline is represented by a thick red line connecting the outputs of one middleware box to the inputs of the next. The entire process is enclosed within a large red rectangle labeled "Pipeline".

Request

Response

Middleware

Middleware

Middleware

Middleware

Pipeline

The screenshot shows a Microsoft Edge browser window with the URL online.u2u.be/courses/25896/modules/6/27314. The page title is "Step 2 : Add and configure OAuth middleware". On the left, there's a sidebar with a navigation tree under "OAuth and OpenID Connect". The main content area contains two sections: "Add OAuth middleware to your application" and "Configure OAuth middleware (order is important!)". Below these are two code snippets in C#.

- Add OAuth middleware to your application
 - Microsoft.AspNetCore.Authentication.JwtBearer** by Microsoft
ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.
 - Microsoft.Identity.Web** by Microsoft
This package enables ASP.NET Core web apps and web APIs to use the Microsoft identity platform (formerly Azure AD...).
 - Microsoft.AspNetCore.Authentication.OpenIdConnect** by Microsoft
ASP.NET Core middleware that enables an application to support the OpenID Connect authentication workflow.
 - Microsoft.Identity.Web.UI** by Microsoft
This package enables UI for ASP.NET Core Web apps that use Microsoft.Identity.Web.
- Configure OAuth middleware (order is important!)

```
// Add this to the Startup.ConfigureServices method
services.AddAuthentication(OpenIdConnectDefaults.AuthenticationScheme)
    .AddMicrosoftIdentityWebApp(Configuration.GetSection("AzureAd"));

// Add this to the Startup.Configure method
app.UseAuthentication();
```

Configure the Application

- Copy to `appsettings.json`
 - Application ID
 - Directory ID

```
"AzureAd": {  
    "Instance": "https://Login.microsoftonline.com/",  
    "Domain": "peteru2u.onmicrosoft.com",  
    "TenantId": "301492ba-4fbf-4114-be09-2b346e16849f",  
    "ClientId": "33f2b982-a441-4c3a-bf02-ac27a56ccc4c",  
    "CallbackPath": "/signin-oidc"  
},
```

Directory properties

* Name:

Country or region: Belgium

Location: EU Model Clause compliant datacenters

Notification language: English

Directory ID: [\[Copy\]](#)

OpenIdConnectLab [\[X\]](#)

Registered app

[Settings](#) [Manifest](#) [Delete](#)

Display name: OpenIdConnectLab	Application ID: 33f2b982-a441-4c3a-bf02-ac27a56ccc4c
Application type: Web app / API	Object ID: 04fd64ca-e4fe-4a2d-a89e-d616d6009861
Home page: https://localhost:44350/	Managed application in local directory: OpenIdConnectLab

26°C Mostly sunny 3:20 PM ENG 12/29/2023

What does the CookieAuthentication do?

- The Cookie middleware persists sessions (with claims)
 - Deserializes claims from cookie on entry
 - Serializes claims as cookie on return
 - Clears cookie after logout

The screenshot shows a Microsoft Edge browser window with the following details:

- Title Bar:** U2U Online
- Address Bar:** online.u2u.be/courses/25896/modules/6/27314
- Page Content:**
 - A sidebar on the left displays a navigation tree under "OAuth and OpenID Connect".
 - The main content area features a large title "OpenID Connect middleware" and a bulleted list describing its function.
- Right Side:** A large "u2u" logo is visible.
- Taskbar:** Shows the Windows Start button, a search bar, pinned icons for File Explorer, Google Chrome, and others, and system status icons.