Securing Your API

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

Coming Up

The authorization code flow with PKCE protection

Getting an access token, passing an access token to an API, validating it

- API scopes vs. API resources

Coming Up

Using access token claims

Including additional identity claims in an access token

Role-based authorization

The Authorization Code Flow + PKCE

Client application
(relying party)

IDP

create code_verifier

hash (SHA256)

code_challenge

authentication request + code_challenge

authorization endpoint

store code_challenge

user authenticates

(user gives consent)

code

code

token request (code, clientid, clientsecret, code_verifier)

token endpoint

The Authorization Code Flow + PKCE

Client application
(relying party)

IDP

token request (code, clientid, clientsecret, code_verifier)

token endpoint

hash code_verifier

check if it matches the
stored code_challenge

id_token

access_token

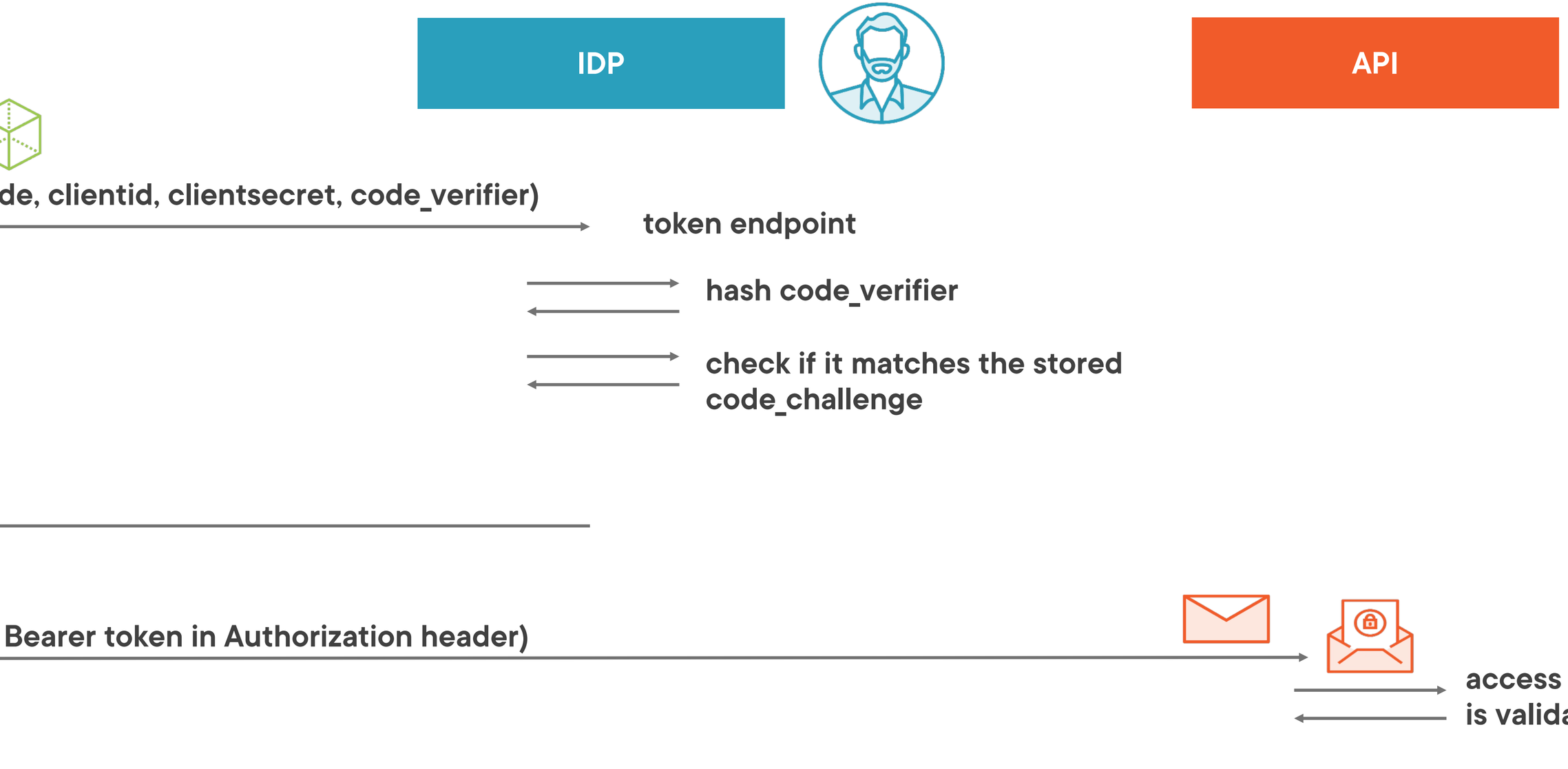
id_token

access_token

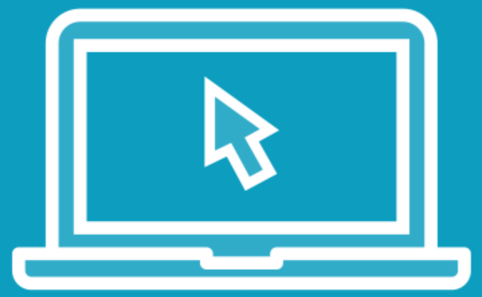
token is validated

access_token (as Bearer token in Authorization header)

The Authorization Code Flow + PKCE



Demo



Securing access to your API (part 1)

(API) scope

The scope of access that's requested by a client

API Scopes vs. API Resources

Typical API scopes:

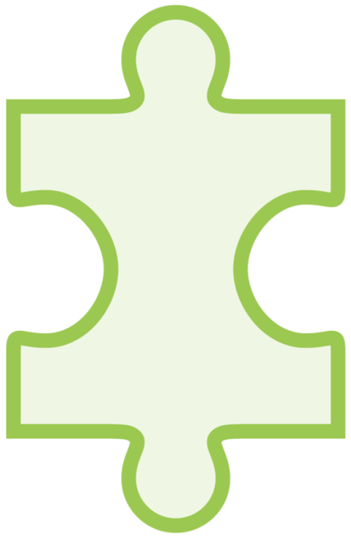
- read
- write
- ...

Use these scopes at level of the API to (dis)allow certain actions

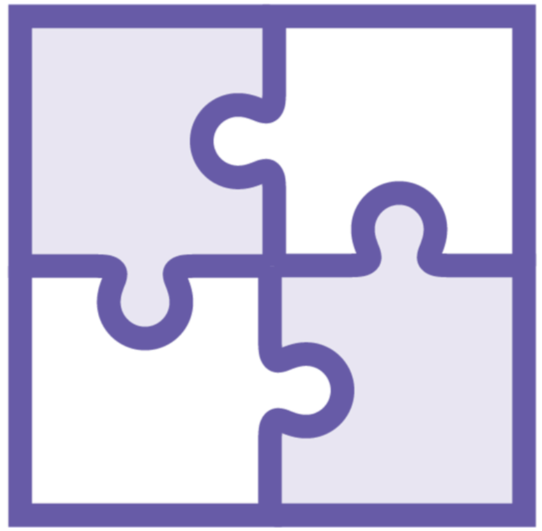
(API) resource

A physical or logical API

API Scopes vs. API Resources



API resource: imagegalleryapi
API scopes: imagegallery.read, imagegallery.write



API resources: employeeapi, customerapi, ...

API Scopes vs. API Resources

Web client
API scopes: imagegallery.read, imagegallery.write

Mobile client
API scopes: imagegallery.read

API Scopes vs. API Resources

Requesting a scope related to a resource results in:

- API resource(s) in audience (`aud`) claim list
- API scope(s) in scopes (`scopes`) claim list

```
{
    ...,
    "aud": ["imagegalleryapi"],
    "scopes": ["imagegalleryapi.read", "imagegalleryapi.write"]
    ...
}
```

Example Access Token

Requesting imagegalleryapi.read and imagegalleryapi.write scopes

```
{
    ...,
    "aud": ["imagegalleryapi"],
    "scopes": ["imagegalleryapi.read"]
    ...
}
```

Example Access Token

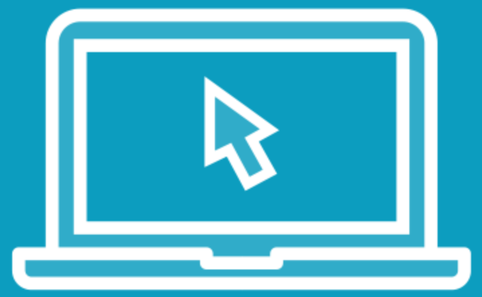
Requesting imagegalleryapi.read scope

API Scopes vs. API Resources

Upcoming demos:

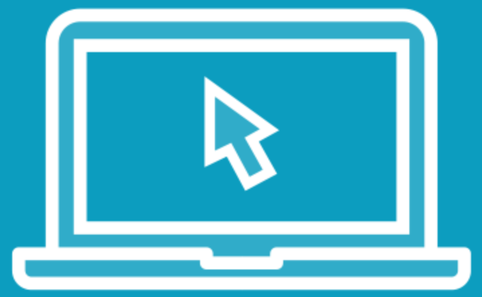
- API resource: `imagegalleryapi`
- API scope linked to resource: `imagegalleryapi.fullaccess`

Demo



Securing access to your API (part 2)

Demo



Passing an access token to your API

Demo

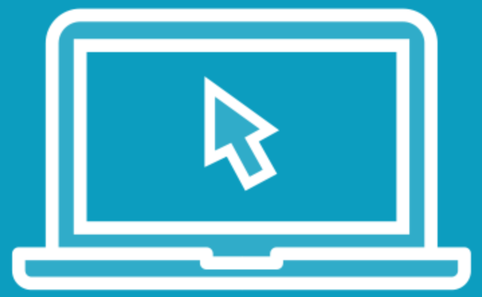
Using access token claims when getting resources

Including Identity Claims in an Access Token

Sometimes an API needs access to identity claims

- When defining an API resource, include the required claims in the claims list

Demo



Including identity claims in an access token

Demo

Protecting the API when creating a resource (with roles)

Summary

Access tokens are passed to the API as Bearer tokens

- IdentityModel.AspNetCore

`JwtBearerToken` middleware is used to validate an access token at level of the API

Using API resources allows for more complex scenarios than only using API scopes

- Results in audience + scope claim(s)

Summary

Configure the API resource to include additional identity claims in the access token

Role-based authorization is achievable through the `[Authorize]` attribute

Up Next:
Authorization Policies and Access Control