Understanding Authentication With OpenID Connect

Kevin Dockx
ARCHITECT

@KevinDockx https://www.kevindockx.com

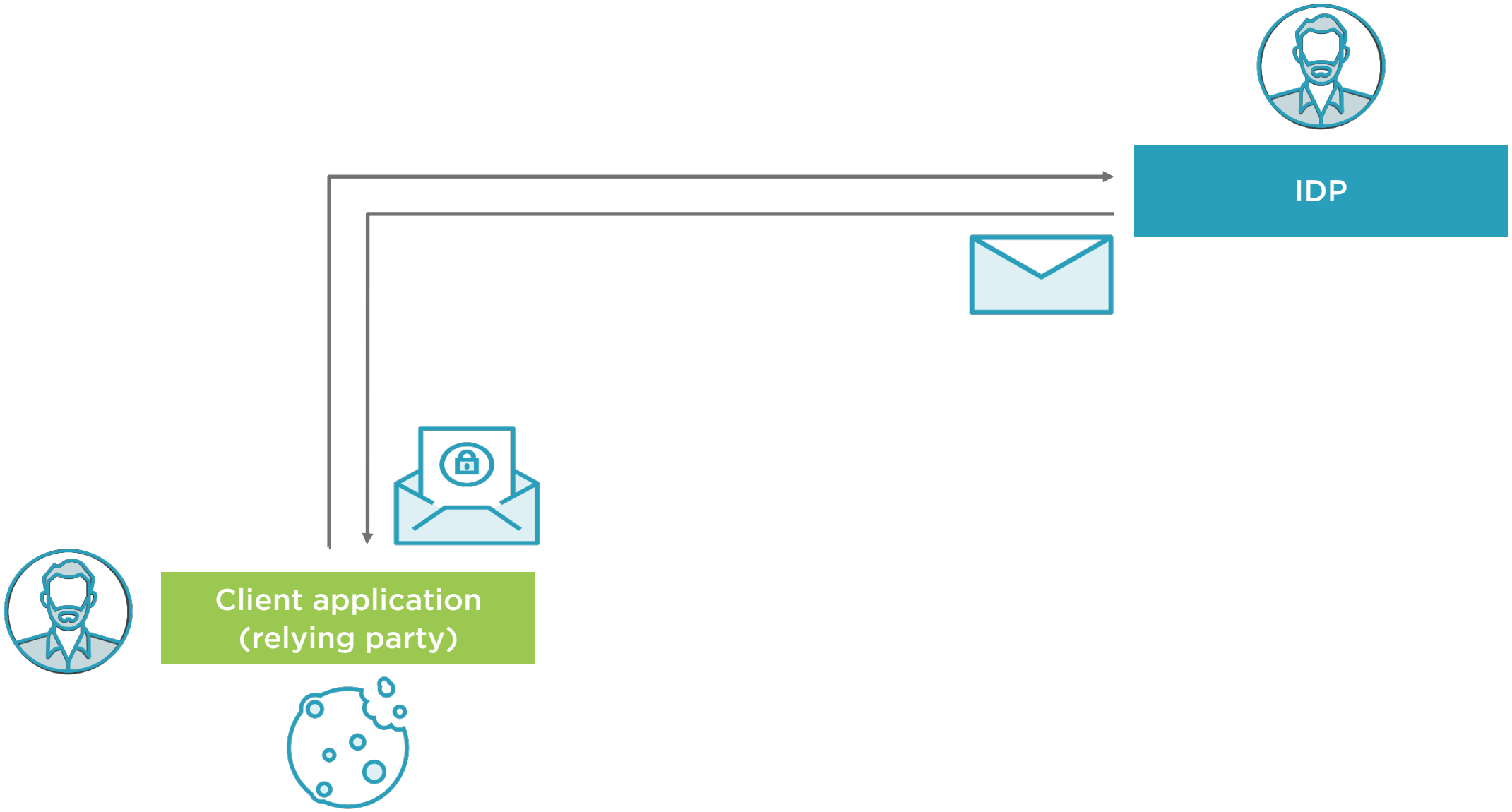
Coming Up

How OpenID Connect works

Clients, endpoints and flows

Setting up an identity provider: IdentityServer4

How OpenID Connect Works



Public and Confidential Clients

Confidential clients

Capable of maintaining the confidentiality of their credentials (e.g.: clientid, clientsecret)

Live on the server

These client applications can safely authenticate

E.g.: server-side web apps

Public clients

Incapable of maintaining the confidentiality of their credentials (e.g.: clientid, clientsecret)

Live on the device

These client application **cannot** safely authenticate

E.g.: JavaScript apps *(and mobile apps)*

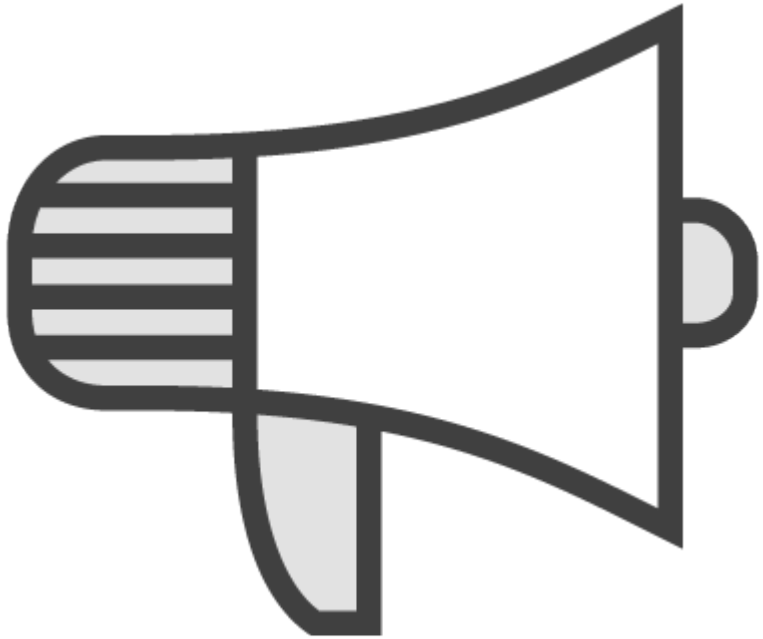
OpenID Connect Flows and Endpoints

The flow determines how the code and/or token(s) are returned to the client

- Depending on application type, requirements, ... we must use a different flow

Authorization endpoint (IDP level)

- Used by the client application to obtain authentication and/or authorization, via redirection



TLS is a requirement for OIDC

Redirection endpoint (client level)

- Used by the IDP to return code & token(s) to the client application

Token endpoint (IDP level)

- Used by the client application to request tokens (without redirection) from the IDP

OpenID Connect Flows

Authorization Code

Tokens from token endpoint

Confidential clients

Long-lived access

Implicit

Tokens from authorization endpoint

Public clients

No long-lived access

Hybrid

Tokens from authorization endpoint & token endpoint

Confidential clients

Long-lived access

Reasoning towards a Flow with an Example

Choosing the wrong flow is a potential security risk

- Long lifetimes should only be allowed to authenticated clients
- For that, clients must be able to safely store their credentials

Reasoning towards a Flow with an Example

Choosing the wrong flow is a potential security risk

- ...
- Authorization code and hybrid flows should not be allowed for those clients

Reasoning towards a Flow with an Example

Choosing the wrong flow is a potential security risk

- ...
- ~~Authorization code and hybrid flows should not be allowed for those clients~~

Reasoning towards a Flow with an Example

Choosing the wrong flow is a potential security risk

- ...
- Authorization code and hybrid flows can be allowed for those clients, as long as we don't return a refresh token from an unauthenticated token request

The thing with security is that a lot of approaches will work, but most of them are not a good idea

The most important statement of the entire course

What IS a good idea changes over time

The second most important statement of the entire course

OpenID Connect Flow for ASP.NET Core

ASP.NET Core MVC

- Confidential client (server-side web app)
- We require long-lived access

OpenID Connect Flow for ASP.NET Core

Authorization code with PKCE protection is the current best practice

Hybrid flow (response_type="code id_token") is still a valid and secure option

IdentityServer4

- http://docs.identityserver.io/

IdentityServer4 is an OpenID Connect and OAuth2 framework for ASP.NET Core

- Part of the .NET Foundation

Demo

Setting up IdentityServer4

Demo

Adding a user interface for IdentityServer4

Demo

Adding users to test with

Summary

A confidential client can safely store secrets

A public client can't safely store secrets

A flow can be seen as how an application can achieve authentication (and authorization)

Summary

Authorization endpoint (IDP)

- Used by the client application to obtain authentication and/or authorization

Token endpoint (IDP)

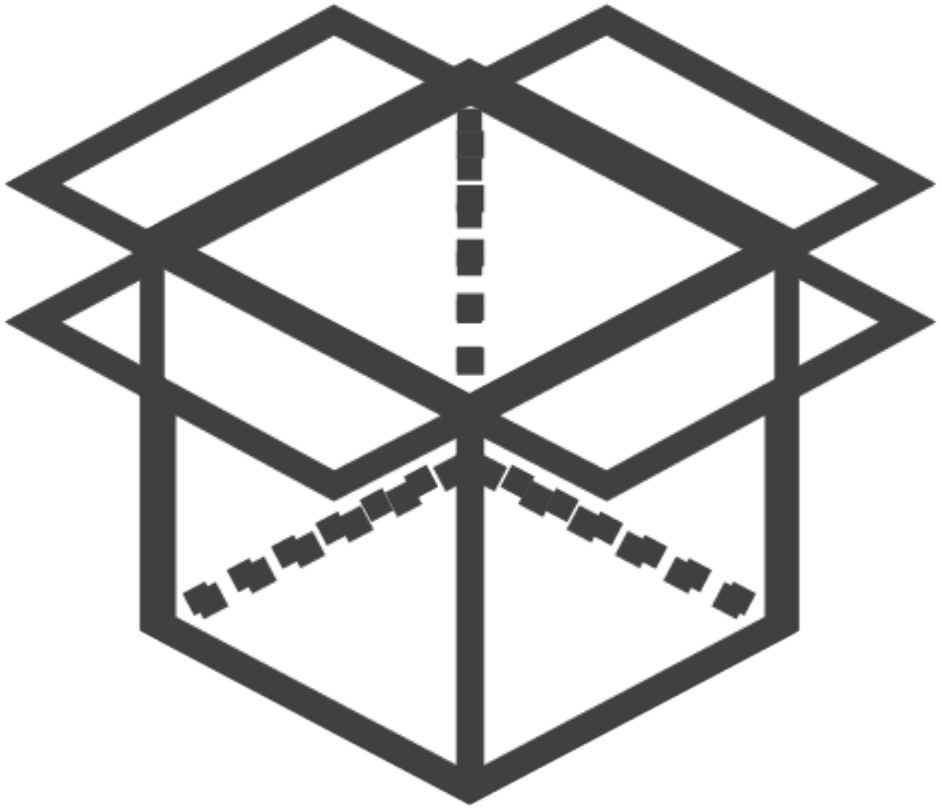
- Used by an application to programmatically request tokens

Summary

Redirection endpoint (client application)

- Where the tokens are delivered to from the authorization endpoint

TLS is a requirement!



It's our responsibility to keep the holes in this box as small as possible