Securing Your User Authentication Processes

Kevin Dockx
Architect

@KevinDockx https://www.kevindockx.com

Coming Up

The authorization code flow with PKCE protection

- Logging in and logging out

Best practice for returning identity claims

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Authorization Code Flow

Authentication request to the authorization endpoint

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Authorization Code Flow

Authorization endpoint at IDP level

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Authorization Code Flow

Identifier of the client

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Authorization Code Flow

Redirection endpoint at client level

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Authorization Code Flow

Requested scopes by the client application

```
https://idphostaddress/connect/authorize?
client_id=imagegalleryclient
&redirect_uri=https://clientapphostaddress/signin-oidc
&scope=openid profile
&response_type=code
&response_mode=form_post
&nonce=63626...n2eNMxA0
```

The Authorization Code Flow

The requested response_type determines the flow

Response Type Values

`code` **Authorization Code**	`id_token` **Implicit**	`id_token token` **Implicit**
`code id_token` **Hybrid**	`code token` **Hybrid**	`code id_token token` **Hybrid**

Response Type Values

`code` **Authorization Code**	`id_token` **Implicit**	`id_token token` **Implicit**
`code id_token` **Hybrid**	`code token` **Hybrid**	`code id_token token` **Hybrid**

The Authorization Code Flow

Client application
(relying party)

IDP

authentication request

authorization endpoint

user authenticates

(user gives consent)

code

code

token request (code, clientid, clientsecret)

token endpoint

id_token

id_token

token is validated

Authorization Code

A very short-lived token that provides proof of authentication, linked to the user that just signed in to the IDP

Communication Types

Front channel communication

Information delivered to the browser via URI or Form POST (response_mode)

In our current flow: authorization endpoint

Back channel communication

Server to server communication

In our current flow: token endpoint

Defence in depth

Implement different types of protection against the same vulnerability.
If one mechanism fails, (an)other mechanism(s) is/are still in place.

The Authorization Code Flow

Client application
(relying party)

IDP

authentication request

authorization endpoint

user authenticates

(user gives consent)

code

code

token request (code, clientid, clientsecret)

token endpoint

id_token

id_token

token is validated

Demo

Configuring IdentityServer to log in with the authorization code flow

Demo

Logging in with the authorization code flow

Authorization Code Injection Attack

Authorization code grant is vulnerable to authorization code injection attacks

- A leaked authorization code (linked to the victim) is used by the attacker to swap the attacker's session for the victim's session
- The attacker now has the privileges of the victim

Authorization Code Injection Attack

Full description of the attack

- https://nat.sakimura.org/2016/01/25/cut-and-pasted-code-attack-in-oauth-2-0-rfc6749/
- https://tools.ietf.org/html/draft-ietf-oauth-security-topics-13#page-19

Proof Key for Code Exchange (PKCE)

Mitigate with the PKCE (Proof Key for Code Exchange) approach

- https://tools.ietf.org/html/rfc7636
- For each request to the auth endpoint, a secret is created
- When calling the token endpoint, it's verified

Proof Key for Code Exchange (PKCE)

Code injection is mitigated because the attacker doesn't have access to the per-request secret

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

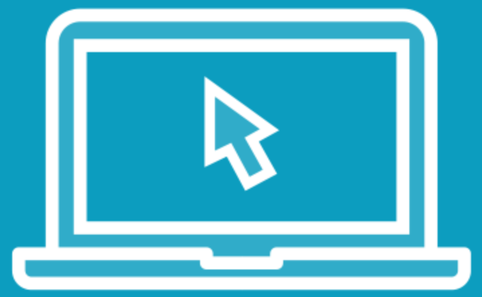
check if it matches the
stored code_challenge

id_token

id_token

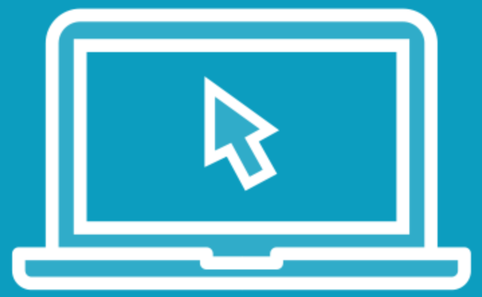
token is validated

Demo



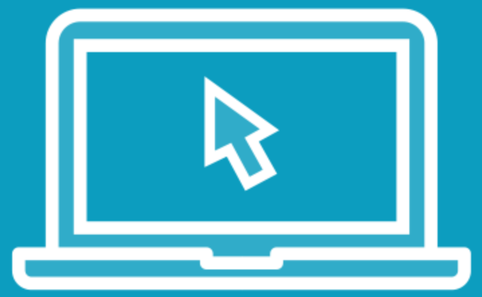
Logging out of our web application

Demo



Logging out of the identity provider

Demo



Redirecting after logging out

The UserInfo Endpoint

Not including the claims in the id_token

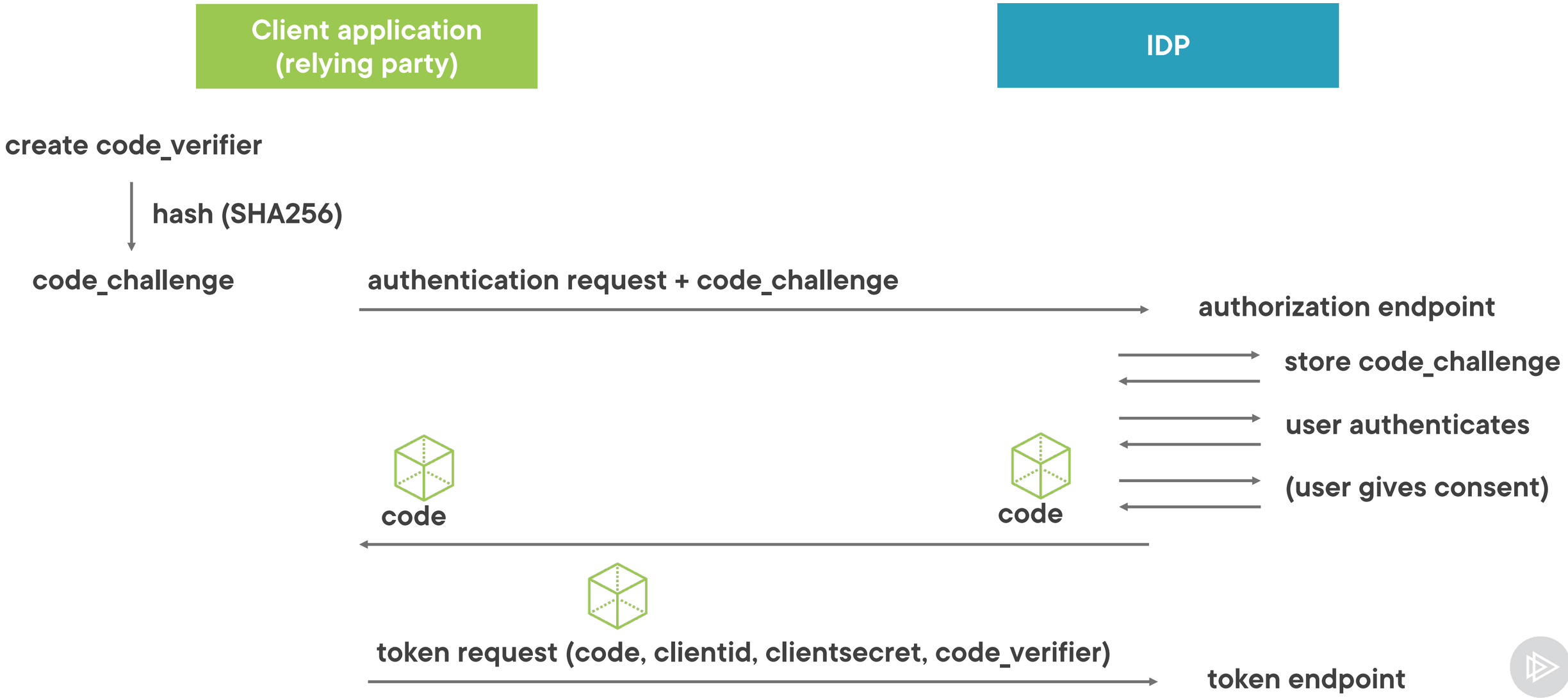
- Keeps the token smaller, avoiding URI length restrictions
- Decreases the potential gains of an attack in case of token interception

The UserInfo Endpoint

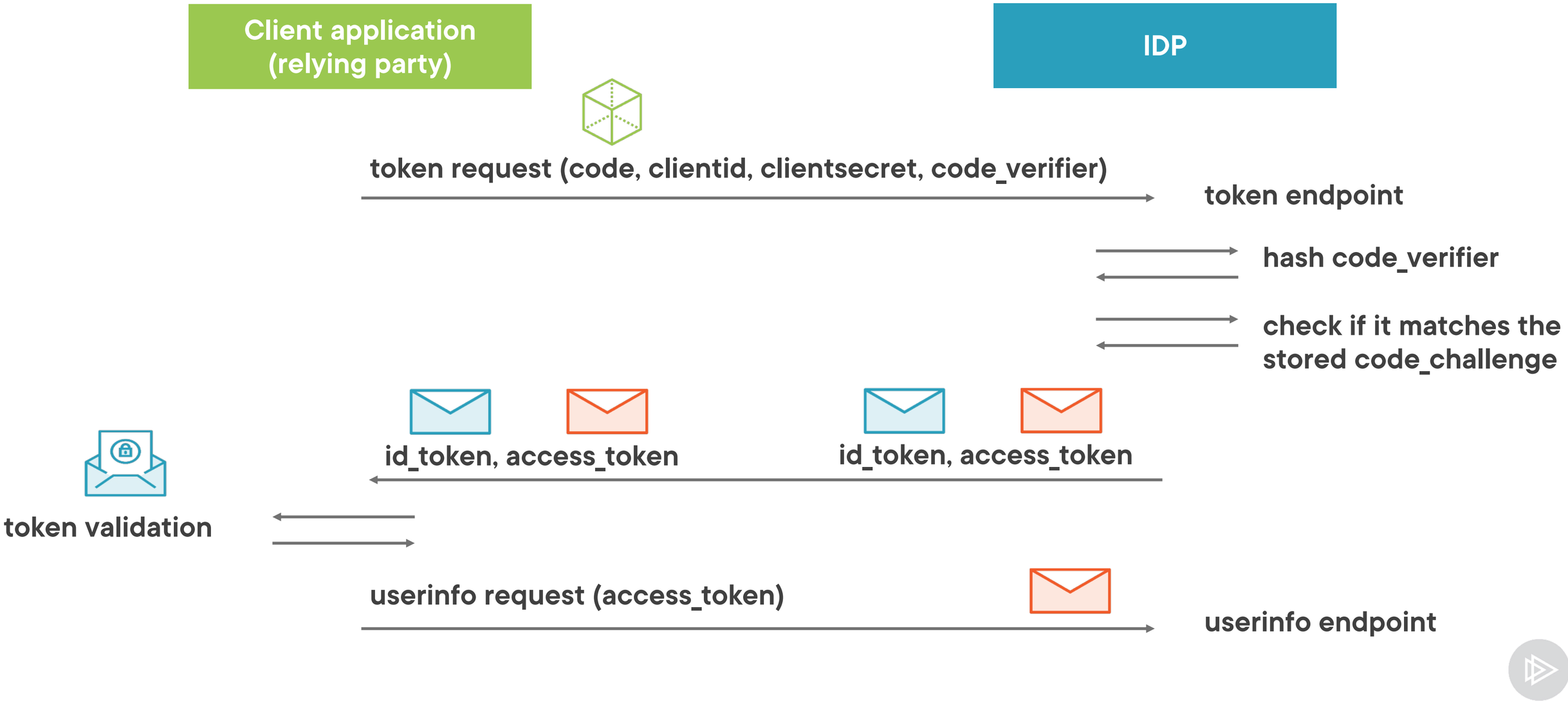
UserInfo endpoint (IDP level)

- Used by the client application to request additional user claims
- Requires an access token with scopes related to the claims that have to be returned

The Authorization Code Flow + PKCE + UserInfo



The Authorization Code Flow + PKCE + UserInfo



The Authorization Code Flow + PKCE + UserInfo

Client application
(relying party)

IDP

userinfo request (access_token)

userinfo endpoint

access token is
validated

user claims

user claims

Demo

Returning additional claims from the UserInfo endpoint

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Emma",
  "iss": "https://localhost:5001",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Identity tokens are JWTs (Json Web Token)

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Emma",
  "iss": "https://localhost:5001",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Subject: the user's identifier

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Emma",
  "iss": "https://localhost:5001",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Optional user claims related to the requested scopes

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Emma",
  "iss": "https://localhost:5001",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Issuer: the issuer of the identity token

```
{
  "sub": "b7539694-97e7-4dfe-84da-b4256e1ff5c7",
  "given_name": "Emma",
  "iss": "https://localhost:5001",
  "aud": "imagegalleryclient",
  …
}
```

Inspecting an Identity Token

Audience: the intended audience for this token

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Issued at: the time at which the JWT was issued

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Expiration: the expiration time on or after which the identity token must not be accepted for processing

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Not before: the time before which the identity token must not be accepted for processing

```
{ …
  "iat": 1490970940,
  "exp": 1490971240,
  "nbf": 1490970940,
  "auth_time": 1490970937,
  …
}
```

Inspecting an Identity Token

Authentication time: the time of the original authentication

```
{ …
  "amr": ["pwd"],
  "nonce":
"63…200.ZjMzZ…5YzFlNWNiN2Mw…AtNGYyZi00MzYzNmZh",
  "at_hash": "90V_c-PO0kdoP-IOERlkdi"
}
```


Inspecting an Identity Token

Authentication methods references: identifiers for authentication methods

```
{ …
  "amr": ["pwd"],
  "nonce":
"63…200.ZjMzZ…5YzFlNWNiN2Mw…AtNGYyZi00MzYzNmZh",
  "at_hash": "90V_c-P00kdoP-IOERlkdi"
}
```

Inspecting an Identity Token

Number only to be used once

```
{ …
  "amr": ["pwd"],
  "nonce":
"63…200.ZjMzZ…5YzFlNWNiN2Mw…AtNGYyZi00MzYzNmZh",
  "at_hash": "90V_c-PO0kdoP-IOERlkdi"
}
```

Inspecting an Identity Token

Access token hash: Base64 encoded value of the left-most half of the hash of the octets of the ASCII representation of the access token

Summary

Current best practice: authorization code flow with PKCE protection

Flow has a front channel and back channel part

- Front channel communication goes via the browser
- Back channel communication is server to server communication

Summary

`ClaimsIdentity` is created from a validated id_token

Claims can be returned from the UserInfo endpoint to avoid issues with URL length restrictions & decrease the gains of a potential attack

When logging out, remember to log out of the IDP if required

Up Next:

Working with Claims in Your Web Application