Authentication and Authorization With OpenID Connect

Syamala Umamaheswaran


@shyamala_u


shyamz-22

Why OpenID Connect?

Enables

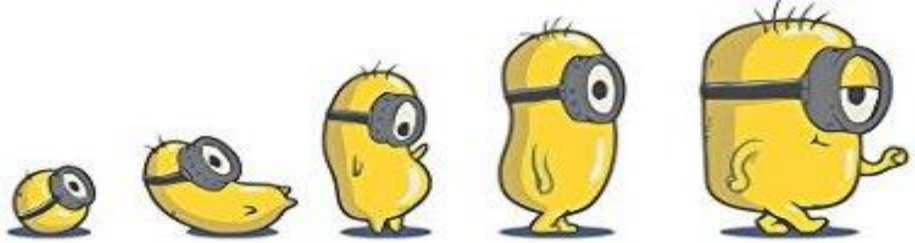
Users to share access to their data to third party application without sharing credentials (Ex:Username and password) with the application

Clients to verify end user authentication and obtain basic profile information of the User

OpenID Connect

- ➔ Social Graph
- ➔ Controlled Information sharing
- ➔ Supports Mobile, Single Page Application and Web apps
- ➔ Description for security and privacy considerations
- ➔ Managing a local authentication mechanism is not necessary
- ➔ Federated Identity management
- ➔ Application of OpenID Connect in IoT, Microservices

Why do you need to care as a developer?



OpenID Connect - Evolution

Enables

Users to share access to their data to third party application without sharing credentials (Ex:Username and password) with the application

OAuth2

- ✗ OAuth2 is not an authentication protocol
- ✗ No feedback about user authentication to client applications
- ✗ No standard for accessing user data resulting in complex code per provider

Why Not OAuth2?

OpenID Connect 1.0 is a simple identity layer on **top of the OAuth 2.0** [RFC6749] protocol. It enables Clients to **verify the identity of the End-User** based on the authentication performed by an Authorization Server, as well as to **obtain basic profile information** about the End-User in an interoperable and REST-like manner.

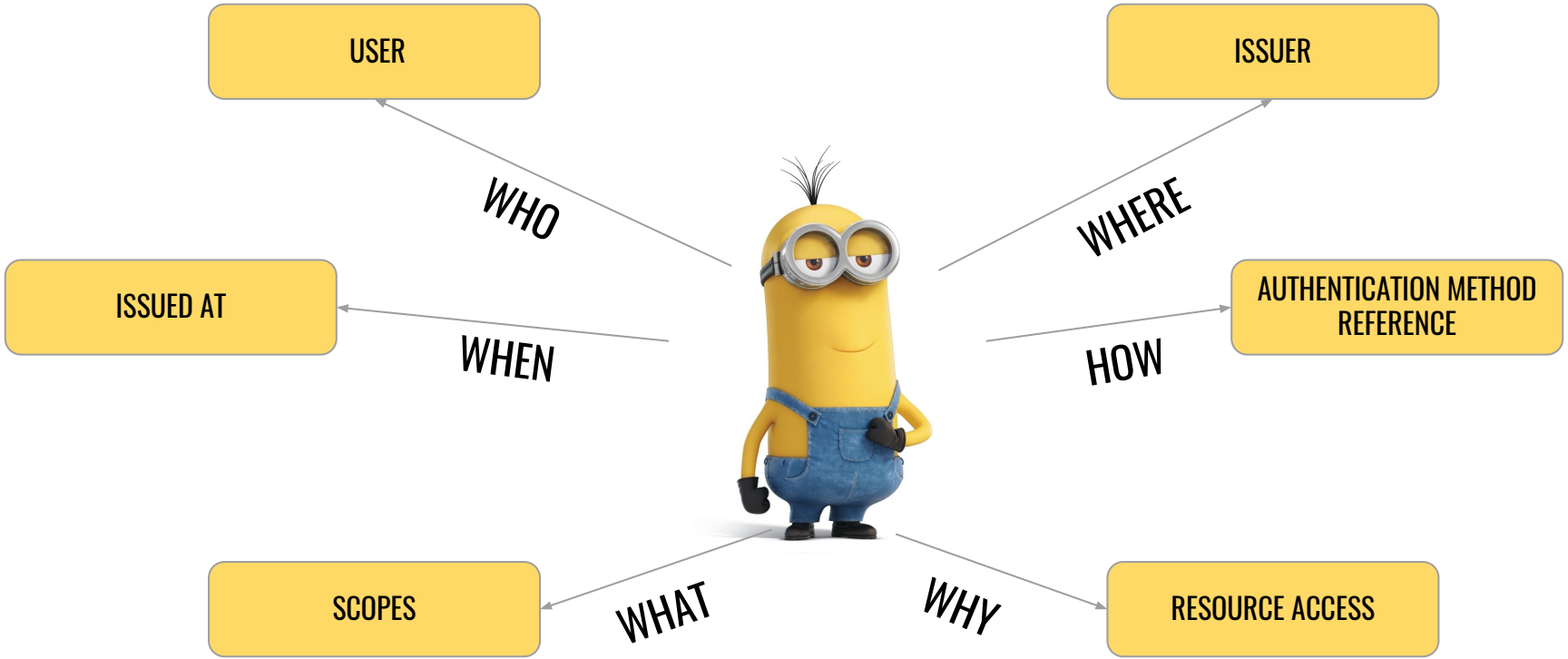
OpenID Connect

That is a mouthful !!!

Identity Layer on Top

Base Protocol

(Identity, Authentication)+ OAuth2 = OpenID Connect



OpenID Connect - Identity Layer

OpenID Connect - ID Token


```
{
  "azp":
"262806823822.apps.googleusercontent.com",
  "aud":
"262806823822.apps.googleusercontent.com",
  "sub": "193423423489515266",
  "email": "minion.bob@gmail.com",
  "email_verified": true,
  "at_hash": "OqvI6CGcsgbCGf2CILcfcQ",
  "nonce": "n-OS6_WzA2Mj",
  "iss": "https://accounts.google.com",
  "iat": 1492383446,
  "exp": 1492387046,
  "picture": "https://google.com/bob.jpg",
  "name": "Bob King",
}
```


- ➔ A digital identity card
- ➔ Authenticated end user data
- ➔ Self contained security token (JWT)
- ➔ Contains claims requested by the client

OpenID Connect

- Flows

```
+---------+                                      +---------+
|         |                                      |         |
|         |---------(1) AuthN Request-------->|         |
|         |                                      |         |
|         |  +---------+                         |         |
|         |  |         |                         |         |
|         |  |  End-   |<--(2) AuthN & AuthZ-->|         |
|         |  |  User   |                         |         |
|   RP    |  |         |                         |   OP    |
|         |  +---------+                         |         |
|         |                                      |         |
|         |<--------(3) AuthN Response--------|         |
|         |                                      |         |
|         |---------(4) UserInfo Request----->|         |
|         |                                      |         |
|         |<--------(5) UserInfo Response-----|         |
|         |                                      |         |
+---------+                                      +---------+
```

RP - Relying Party or The Client

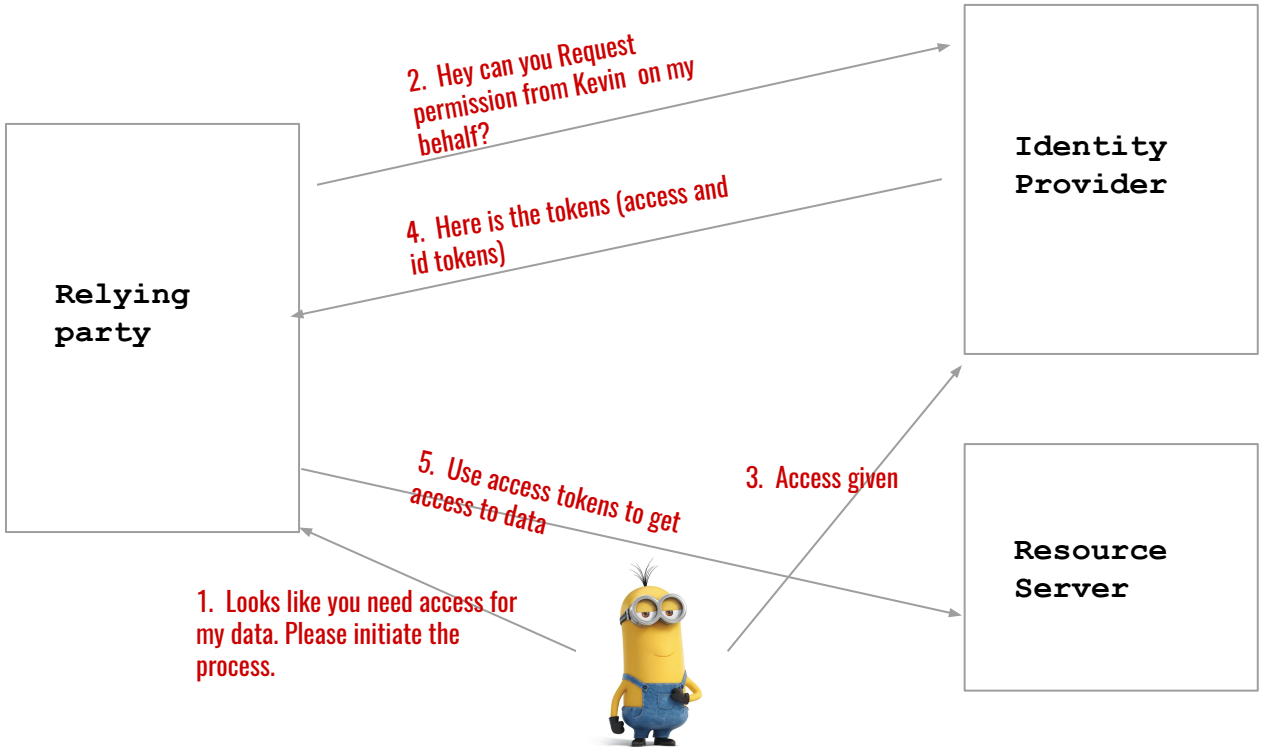
OP - OpenID Provider (like Google)

AuthN - Authentication

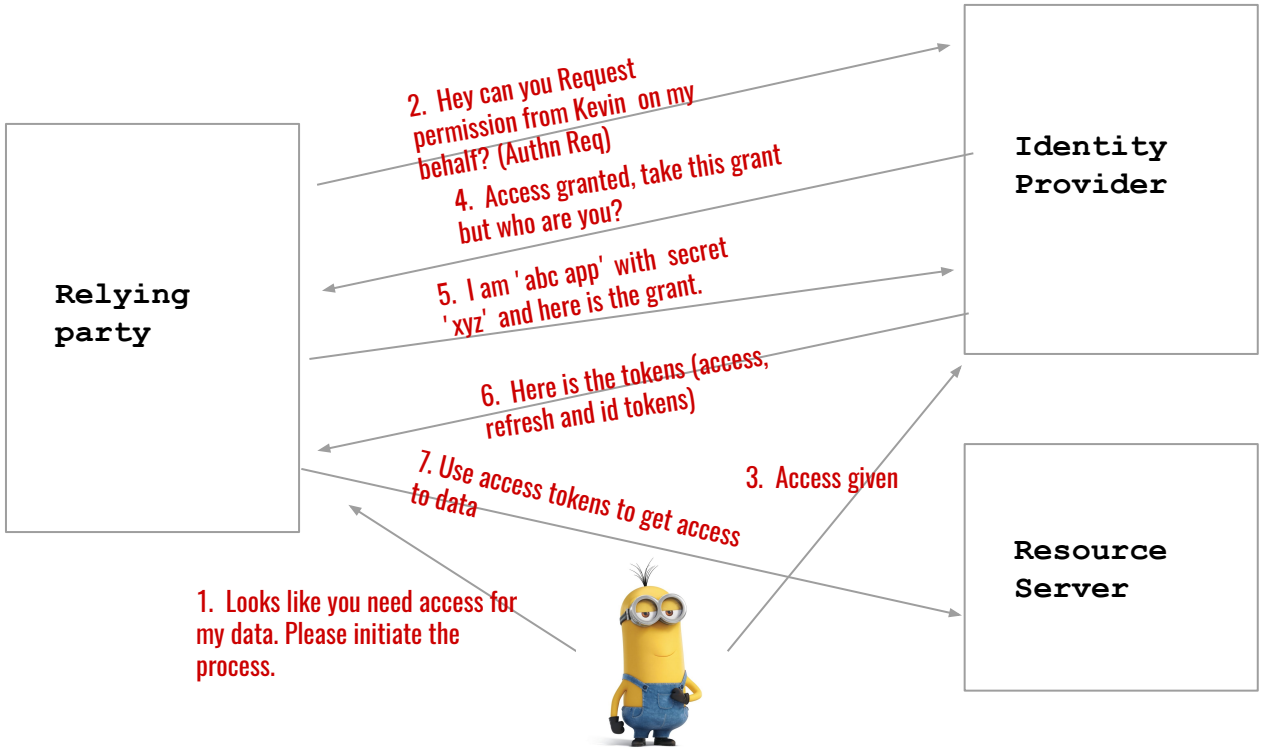
AuthZ - Authorization

End User - The Human participant

Vocabulary



Implicit Flow


Relying party
Identity Provider
Resource Server
2. Hey can you Request permission from Kevin on my behalf? (Authn Req)
4. Access granted, take this grant but who are you?
5. I am 'abc app' with secret 'xyz' and here is the grant.
6. Here is the tokens (access, refresh and id tokens)
7. Use access tokens to get access to data
3. Access given
1. Looks like you need access for my data. Please initiate the process.

Basic Flow

	BASIC	IMPLICIT	HYBRID
SUITABLE FOR	Traditional Webapp	Native app (Android/iOS) or single page app	Native app or single page app with backend
TOKENS REVEALED TO USER AGENT	NO	YES	YES
REFRESH TOKENS	YES	NO	YES

RESPONSE TYPES	`code`	`id_token`
		`id_token token`
			`code id_token`
			`code token`
			`code id_token token`

CLIENT AUTHENTICATION	YES	NO	YES
SECURE	YES	NO	YES

OpenID Connect - Flows

➔ Authorize Endpoint (Authentication Request)

```
Location: https://server.example.com/authorize?
    response_type=code
    &scope=openid%20profile%20email
    &client_id=<client_id>
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &prompt=<login none login consent select_account>
```

➔ Token Endpoint (Code exchange)

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
    grant_type=authorization_code
    &code=SplxlOBeZQQYbYS6WxSbIA
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

➔ Authentication Response (token endpoint)

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache
  {
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcd..."
  }
```

➔ Userinfo Endpoint (resource)

```
GET /userinfo HTTP/1.1
Host: server.example.com
Authorization: Bearer SlAV32hkKG

 HTTP/1.1 200 OK
Content-Type: application/json
{
 "sub": "248289761001",
 "name": "Jane Doe",
 "given_name": "Jane",
 "family_name": "Doe",
 "preferred_username": "j.doe",
 "email": "janedoe@example.com",
 "picture": "http://example.com/janedoe/me.jpg"
}
```

➔ Discovery Endpoint

```
https://<<Issuer Identifier>>/.well-known/openid-configuration
```

➔ Keys Endpoint


```
"keys": [{"kty": "RSA",
   "alg": "RS256",
   "use": "sig",
   "kid": "bc91576fc93df3adc59896c495cb6729dd5bc023",
   "n":"k4ar7LTlxvlL1ZfqwWIG0Hkphli3a4dqC_BIfFSJx-raiN…..",
   "e": "AQAB"
}]
```


OpenID Connect - Core Endpoints

OpenID Connect Playground

Let us see something in action

- Validate ID Tokens
- Do not omit `State` and `Nonce` parameters
- Choose the right flow
- Do not use access tokens of Idp to secure your application backend
- Exchange ID tokens to get app specific access tokens (Draft)
- Build IdP for redundancy

BEST PRACTICES

When token expires user authentication is required, this prevents SSO

- ➔ Use `refresh_tokens` to renew the tokens to prevent login prompt
- ➔ Use `prompt=none` or no prompt to achieve SSO behavior

Best Practices

OpenID connect Specification
http://openid.net/specs/openid-connect-core-1_0.html

JSON Web Tokens
https://tools.ietf.org/html/draft-ietf-oauth-json-web-token-32

Decoding JWT
https://jwt.io/

Sample Application
https://github.com/shyamz-22/spring-boot-google-openid-connect

REFERENCES

QUESTIONS???