The

OpenID Connect

Protocol

@clementoudot

Clément OUDOT
http://sflx.ca/coudot

LLNG

LSC
LDAP Synchronization Connector

LTB Project

Savoir-faire LINUX

- Founded in 1999
- >100 persons
- Montréal, Quebec City, Ottawa, Paris
- ISO 9001:2004 / ISO 14001:2008
- contact@savoirfairelinux.com

Microsoft Certified Partner
redhat. Advanced Business Partner
odoo Gold Partner
LIFERAY Community Excellence 2014 • 2013 2012 • 2011
BlueMind Premium Partner
Alfresco Gold Partner
ORACLE PARTNER
talend Gold Partner
Acquia

GET /summary

```
{
   "part1":"Some words on OAuth 2.0",
   "part2":"The OpenID Connect Protocol",
   "part3":"OpenID Connect VS SAML",
   "part4":"OpenID Connect in Open Source"
}
```


OAUTH
2
OAUTH

RFC 6749

The OAuth 2.0 authorization **framework** enables a **third-party** application to obtain **limited** access to an **HTTP service**, either on behalf of a **resource owner** by orchestrating an approval interaction between the resource owner and the **HTTP service**, or by allowing the **third-party** application to obtain access on its own behalf. This specification replaces and obsoletes the OAuth 1.0 protocol described in RFC 5849.

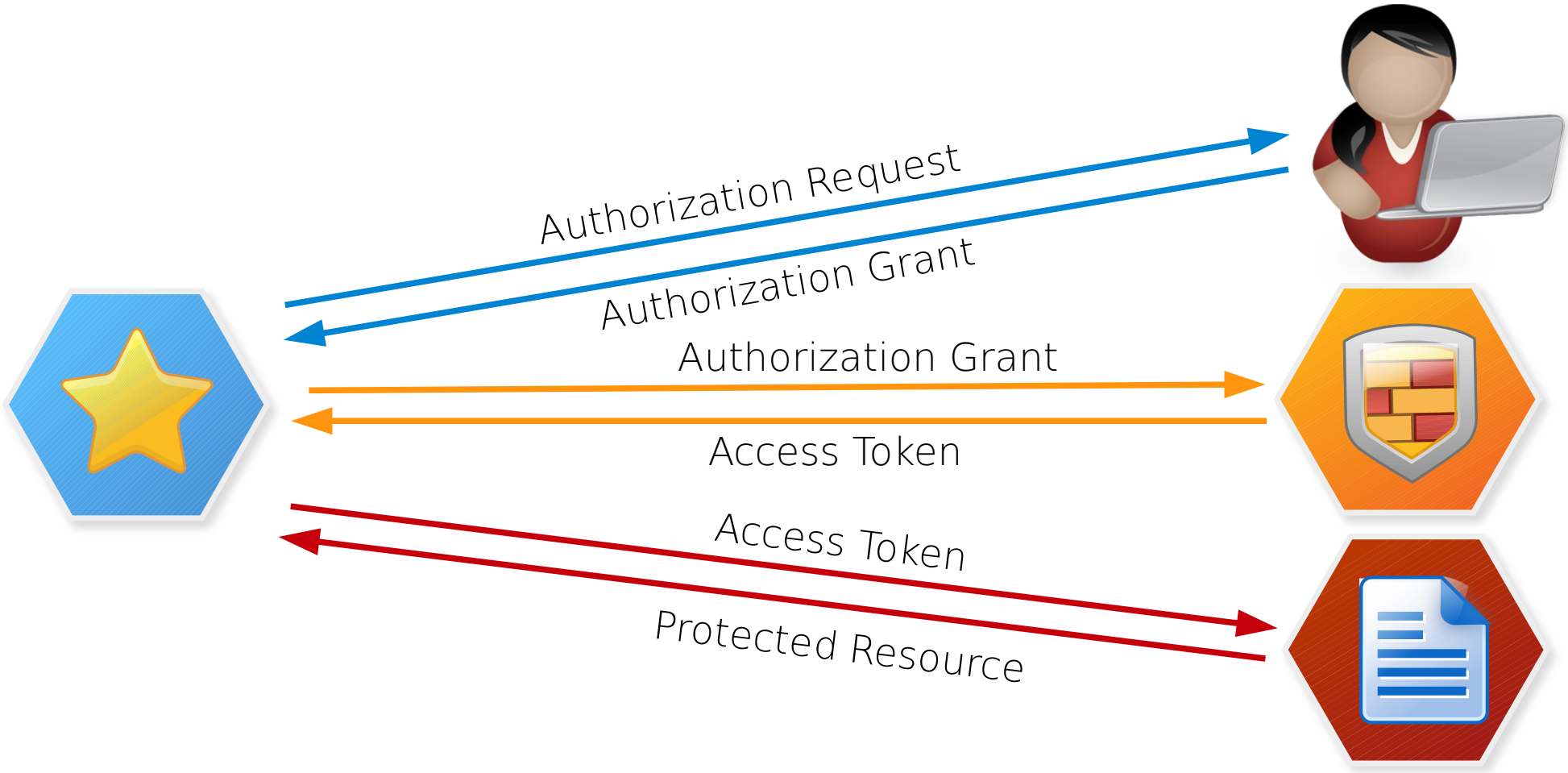
Roles

Resource owner
(end-user)

Client
(third-party)

Authorization
Server

Resource
Server


Authorization Request
Authorization Grant
Authorization Grant
Access Token
Access Token
Protected Resource

Authorization Grant

Authorization Code

- More secure
- Server side applications
- Tokens hidden to end user

Implicit

- Access token directly sent
- Designed for JS client application

Resource Owner Password Credentials

- Requires high trust between end-user and client

Client credentials

- Client is often the resource owner

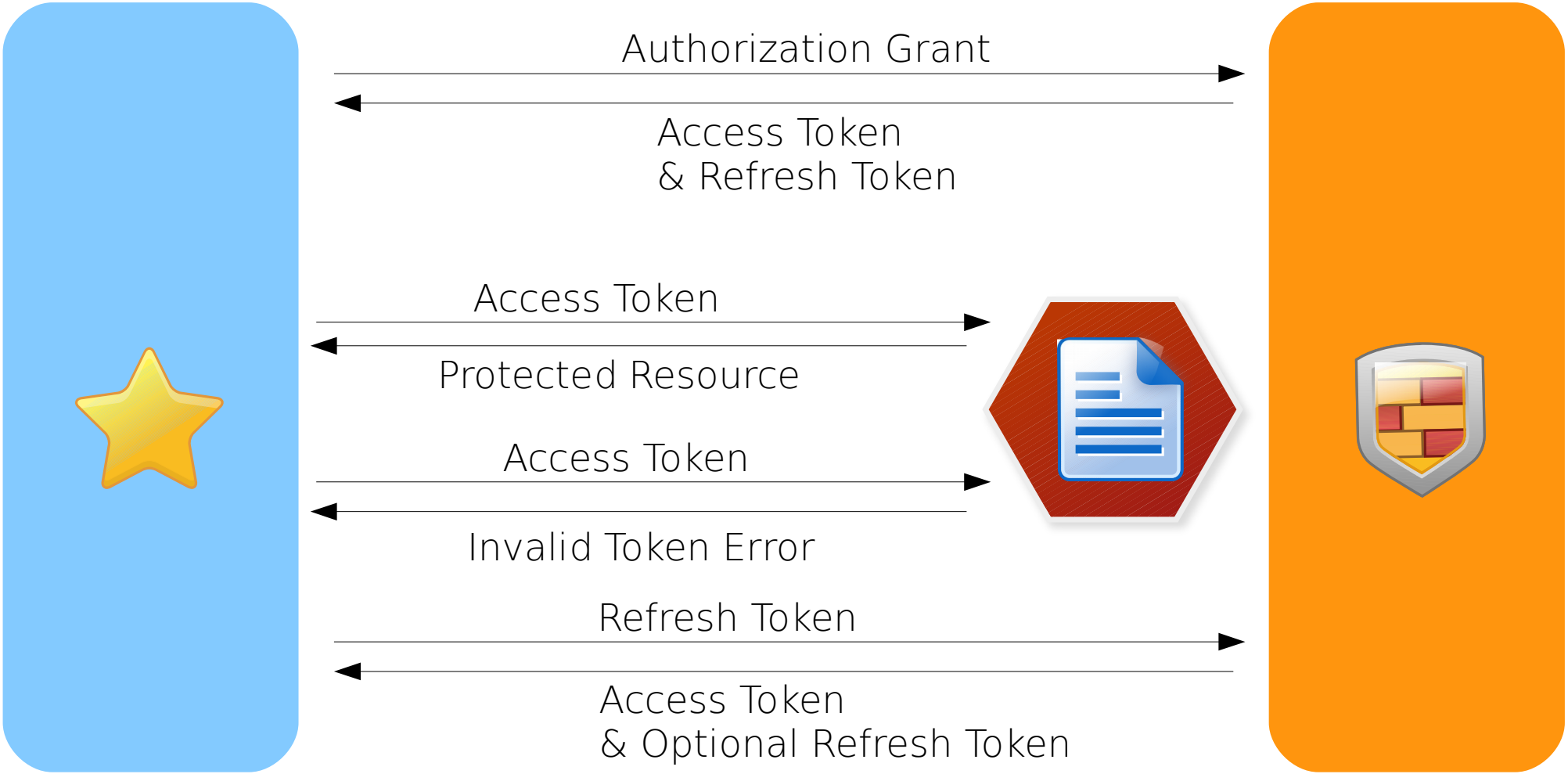
Tokens

Access Token :

- Opaque
- Limited duration
- Scope
- Give access to the resource server

Refresh Token :

- Allow to get a new access token
- Optional
- Can not be used as an access token


Authorization Grant
Access Token
& Refresh Token
Access Token
Protected Resource
Access Token
Invalid Token Error
Refresh Token
Access Token
& Optional Refresh Token

Client Registration

- Client has to be registered with the authorization server
- OAuth 2.0 do not specify how this registration is done
- Information that should be registered:
  - Client type
  - Redirection URIs
  - Other: application name, logo, etc.
- The client then received a client_id and a client_password

Client types

- **Confidential**: Clients **capable** of maintaining the confidentiality of their credentials :
  - Application on a secure server
- **Public**: Clients **incapable** of maintaining the confidentiality of their credentials :
  - Native mobile application
  - Web browser based application

Endpoints

- Authorization Server:
  - Authorization: where the resource owner gives authorization
  - Token: where the client get tokens
- Client:
  - Redirection: where the resource owner is redirected after authorization

Authorization

GET /authorize?
response_type=code&client_id=s6BhdRkqt3&state=xyz&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb

https://client.example.com/cb?
code=SplxlOBeZQQYbYS6WxSbIA
&state=xyz

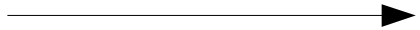
Token

```
POST /token HTTP/1.1
Host: server.example.com
Authorization: Basic
czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-
urlencoded

grant_type=authorization_code&code=SplxlOBe
ZQQYbYS6WxSbIA&redirect_uri=https%3A%2F
%2Fclient%2Eexample%2Ecom%2Fcb
```

Token

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
"access_token":"2YotnFZFEjr1zCsicMWpAA",
"token_type":"example",
"expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
"example_parameter":"example_value"
}
```

Resource



```
GET /resource/1 HTTP/1.1
Host: example.com
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```


OpenID Connect

(2) AuthN & AuthZ

(1) AuthN Request

RP

(3) AuthN Response

(4) UserInfo Request

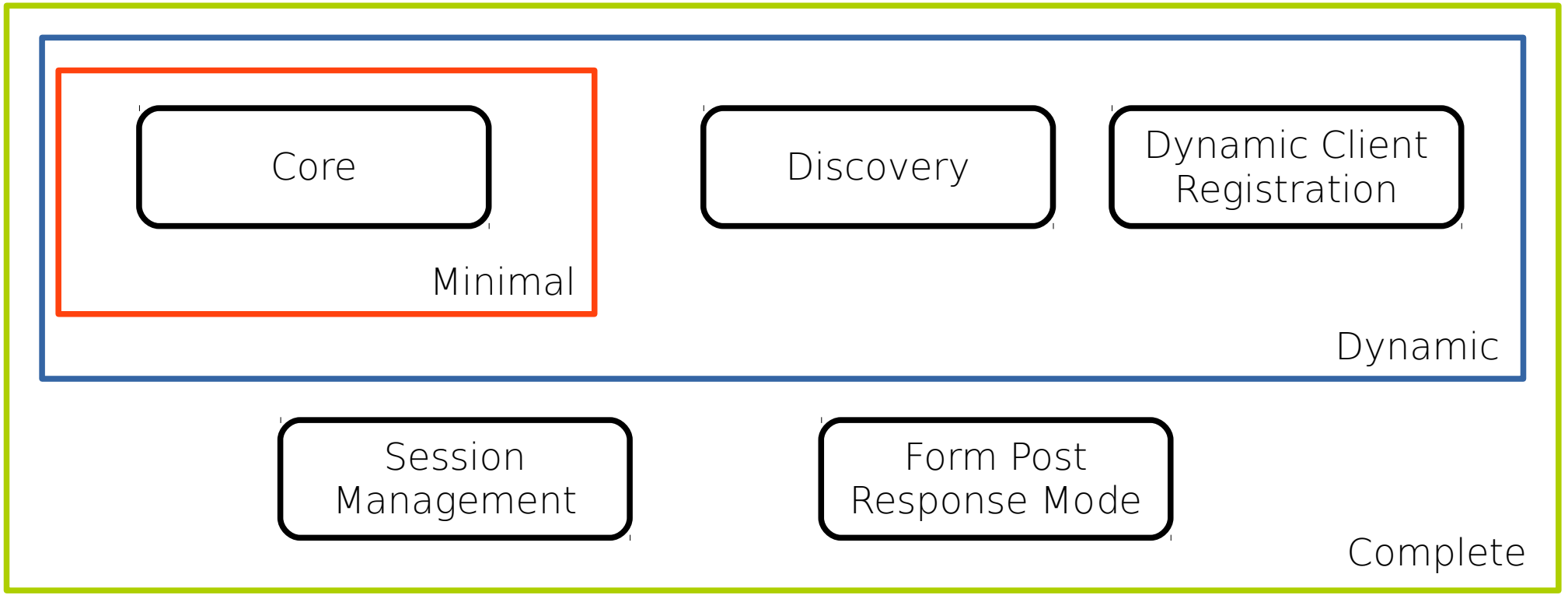
OP

(5) UserInfo Response

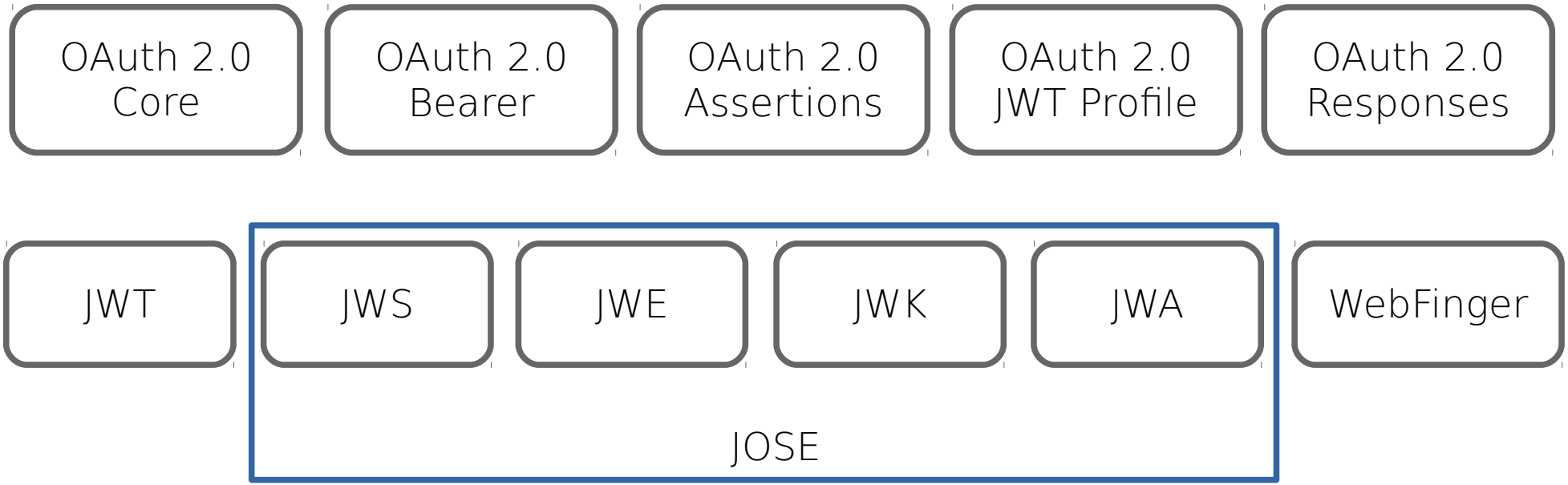
Built on top of OAuth 2.0

- Flows:
  - Based on OAuth 2.0 Authorization grants:
    - Authorization Code
    - Implicit
  - New flow: Hybrid
- Scope:
  - New scope: “openid”
- Endpoints:
  - Use Authorize, Token and Redirection endpoints
  - New endpoint: UserInfo
- Tokens:
  - Use access and refresh tokens
  - New token: ID token (JWT)

OpenID Connect Protocol Suite



Underpinnings



Security over Javascript?

JOSE

Javascript

Object

Signing

and

Encryption

JWT

JSON

Web

Token

JWT

- Concatenation with dots of:
  - base64(Header)
  - base64(Payload)
  - base64(Signature)


```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "admin": true
}
```

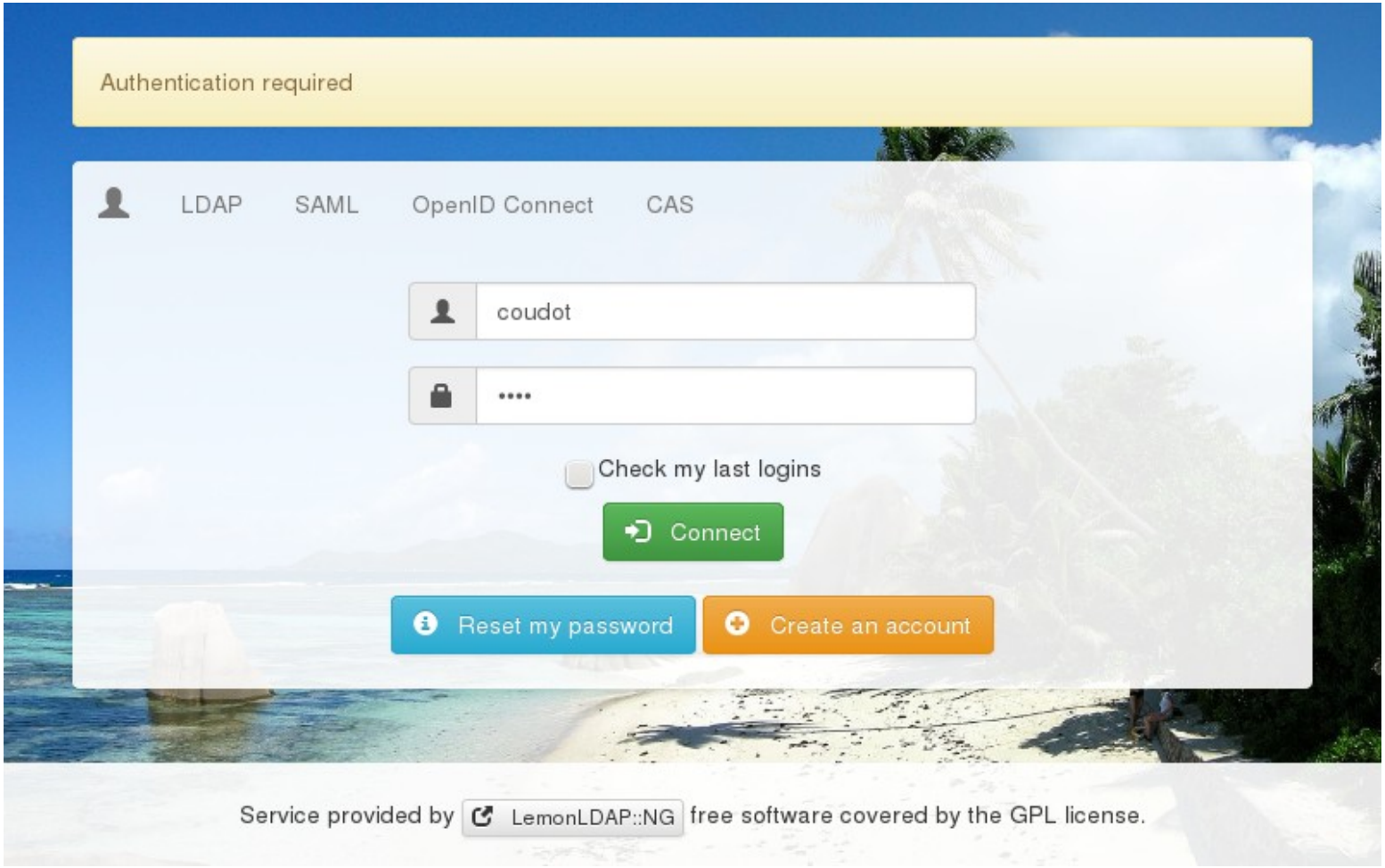
```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) secret base64 encoded
```

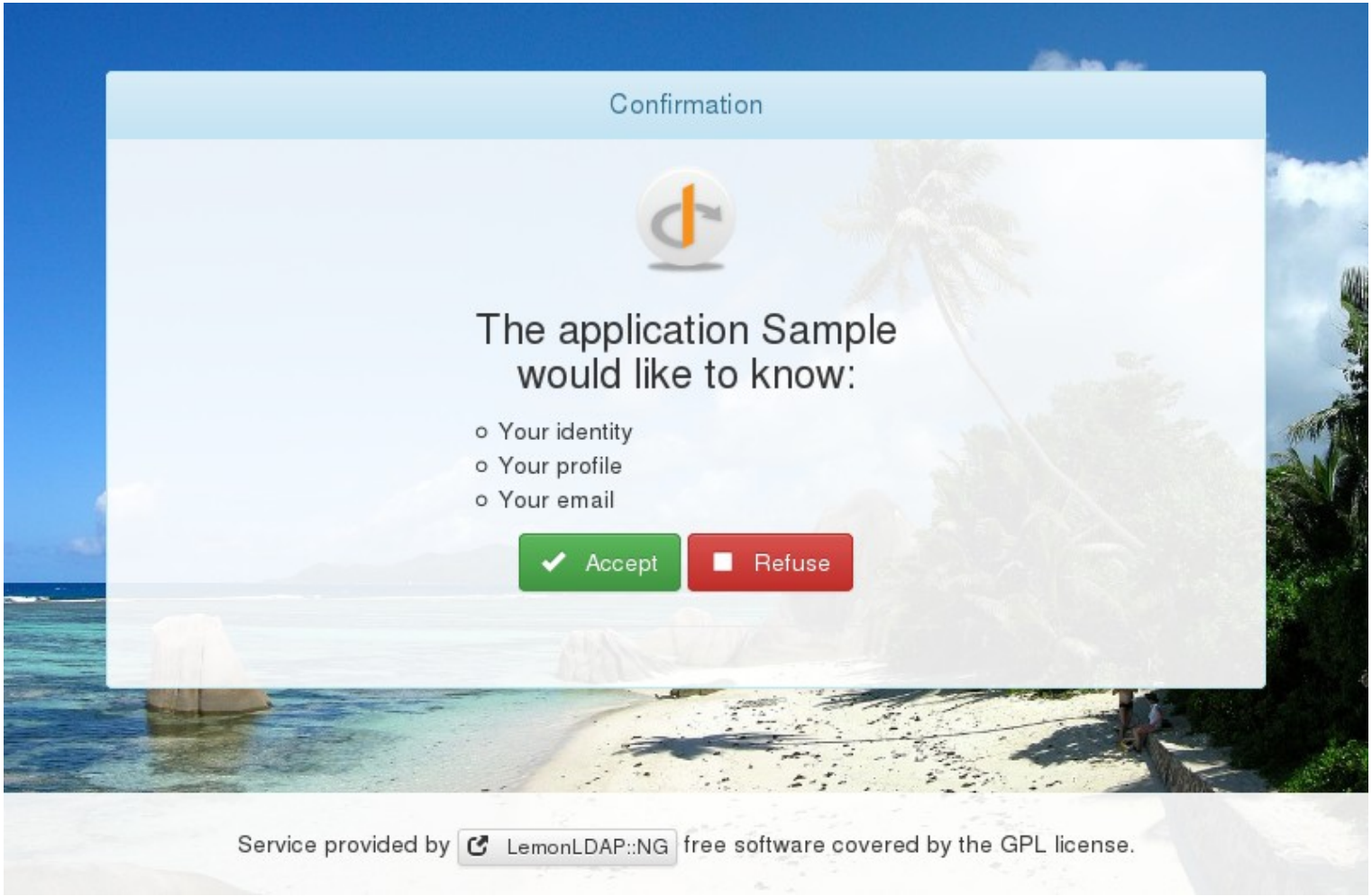
http://jwt.io/

RP

http://auth.example.com/oauth2/authorize?
response_type=code
&client_id=lemonldap
&scope=openid%20profile%20email
&redirect_uri=http%3A%2F
%2Fauth.example.com%2Foauth2.pl
%3Fopenidconnectcallback%3D1
&state=ABCDEFGHIJKLMNOPQRSTUVWXXZ

OP


Authentication required
LDAP
SAML
OpenID Connect
CAS
coudot
Check my last logins
Connect
Reset my password
Create an account
Service provided by
LemonLDAP::NG
free software covered by the GPL license.


Confirmation
The application Sample would like to know:
Your identity
Your profile
Your email
Accept
Refuse
Service provided by LemonLDAP::NG free software covered by the GPL license.

RP

http://auth.example.com/oauth2.pl?
openidconnectcallback=1;
code=f6267efe92d0fc39bf2761c29de44286;
state=ABCDEFGHIJKLMNOPQRSTUVWXXZ

OP

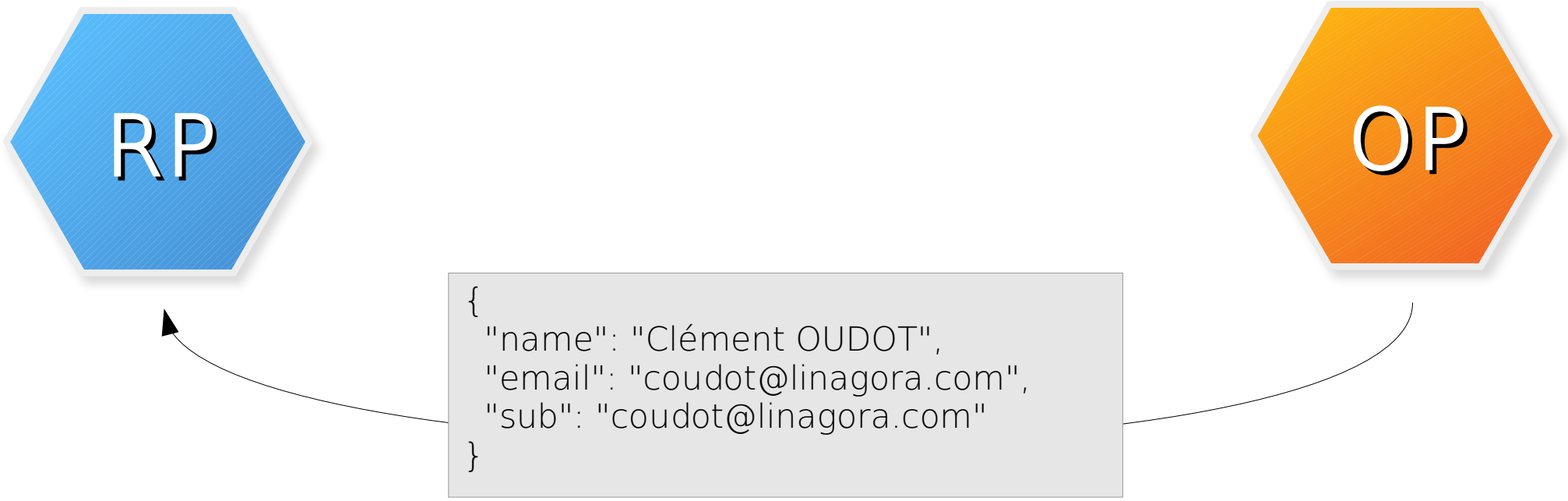
```
POST /oauth2/token HTTP/1.1
Host: auth.example.com
Authorization: Basic xxxx
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=f6267efe92d0fc39bf2761c29de44286
&redirect_uri=http%3A%2F%2Fauth.example.com
%2Foauth2.pl%3Fopenidconnectcallback%3D1
```

{"id_token" :"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ
9.eyJhY3IiOiJsb2EtMiIsImF1dGhfdGltZSI6MTQzMjEx
MzU5MywiaWF0IjoxNDMyMTEzOTY2LCJhdF9oYXNo
IjoiOWF4enNOaTlwTkRrNXpXZWZLc002QSIsImlzcy
I6Imh0dHA6Ly9hdXRoLmV4YW1wbGUuY29tLyIsIm
V4cCI6IjM2MDAiLCJhenAiOiJsZW1vbmxkYXAiLCJub
25jZSI6IjEyMzQ1Njc4OTAiLCJzdWIiOiJjb3Vkb3RAbG
luYWdvcmEuY29tIiwiYXVkIjpbImxlbW9ubGRhcCJdf
Q==.daYGlzIr37dC1R0biIwdvQLM1LlCMsBFFcEufe
MZtXsZvCiiAm-1LFJwJJJDHFOhd-
WQnc9_GvtP3gTabXB8U4gQ2IW-
bPNLUsjT24njmBPYunHy8YTQ5PV-
QnQI5EK5WrrTS04AF86U5Qu6m3b27yWKFXkIuGI7
EUvvByv8L1Anh1gPG3il5cEOnMFHIUzAaC6PkJiy1sj
SBM53nLRAf9NQ6eux4iCVBIRwl26CCgmRTsTRy-
iTxB3bf0LrILohUlAR_-
HPWGseaIAMvqUpGeaovgGDPt4Zip9KERo7368ykg
Qc09VFlLvZIwyMTWQdVBIYdW0oY6el9ZHjofn0mg"
, "expires_in" : "3600","access_token" :
"512cdb7b97e073d0656ac9684cc715fe",
"token_type" : "Bearer"}

ID Token payload


```
{
 "acr": "loa-2",
 "auth_time": 1432113593,
 "iat": 1432113966,
 "at_hash": "9axzsNi9pNDk5zWefKsM6A",
 "iss": "http://auth.example.com/",
 "exp": "3600",
 "azp": "lemonldap",
 "nonce": "1234567890",
 "sub": "coudot@linagora.com",
 "aud": [
  "lemonldap"
 ]
}
```

```
POST /oauth2/userinfo HTTP/1.1
Host: auth.example.com
Authorization: Bearer 512cdb7b97e073d0656ac9684cc715fe
Content-Type: application/x-www-form-urlencoded
```


RP
OP
{
"name": "Clément OUDOT",
"email": "coudot@linagora.com",
"sub": "coudot@linagora.com"
}

OpenID Connect

Vs

SAML

Frameworks

OpenID Connect:

- REST
- JSON
- JWT/JOSE
- HTTP GET/POST
- Offline mode possible

SAML:

- SOAP
- XML
- XMLSec
- HTTP GET/POST
- No offline mode

Network flows

OpenID Connect:

- Direct connection between RP and OP required
- Request can be passed as reference (Request URI)
- Always RP initiated

SAML:

- Can work without link between SP and IDP
- Request and responses can be passed as references (Artefacts)
- IDP initiated possibility

Configuration

OpenID Connect:

- Published as JSON (openid-configuration)
- Client (RP) registration needed
- Keys publication (jwks)

SAML:

- Published as XML (metadata)
- SP and IDP registration needed
- Keys publication (metadata)

Security

OpenID Connect:

- HTTPS
- Signature and encryption of JWT

SAML:

- HTTPS
- Signature and encryption of all messages

User consent

OpenID Connect:

- Consent required to authorize requested scopes
- No account federation

SAML:

- No consent needed to share attributes
- Consent can be asked to federate accounts

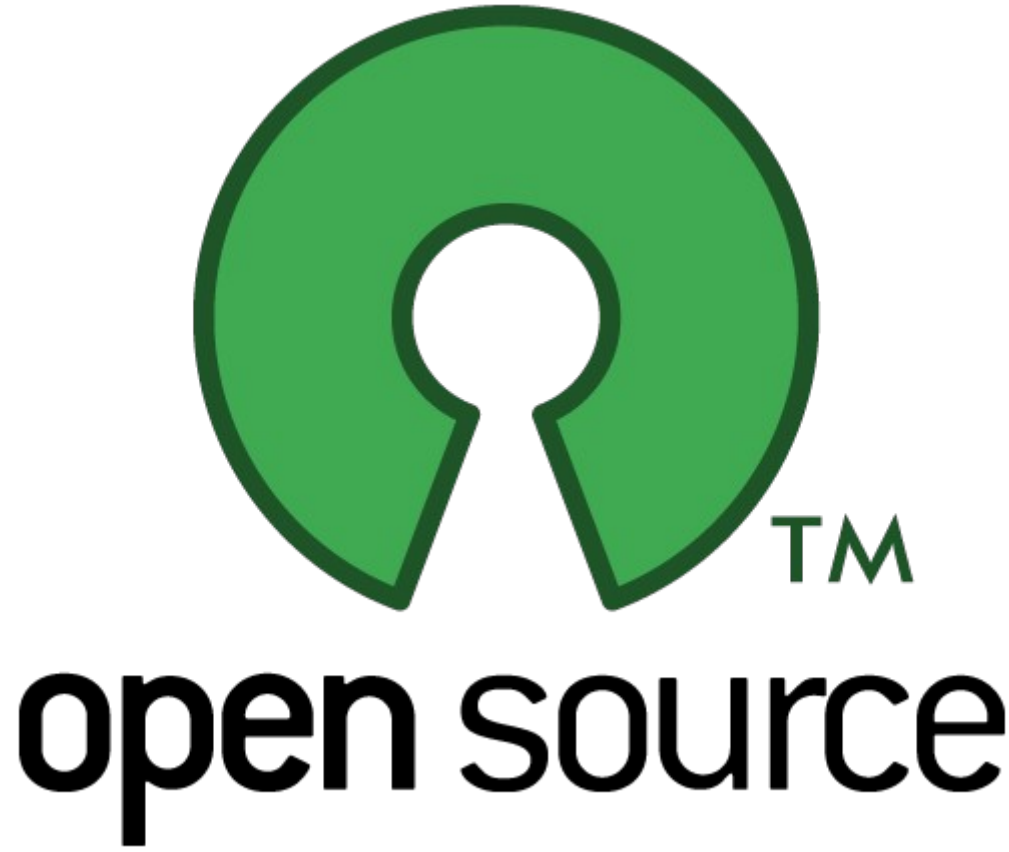
Implementation

OpenID Connect:

- RP: quite easy
- OP: difficult

SAML:

- SP: difficult
- IDP: difficult


TM
open source

Some Open Source implementations

- Free Software (GPLv2+) / OW2 consortium
- Single Sign On, Access Control
- Service Provider / Identity Provider (CAS, SAML, OpenID)
- OpenIDConnect support in version 2.0
- Perl/Apache/CGI/FCGI
- Lost Password and Account Register self services
- http://www.lemonldap-ng.org

Thanks for your attention!

@clementoudot