Simon Gomez - Microservices

oAuth2/Openid Microservice

Goal of this TD is to implement a simple oAuth2/OpenID microservice

Step 0 : use session from previous TD

express-session has to be set up

Step 1 : redirect if not connected

In your motus App, if the user is not logged in, redirect him to the authent server http://localhost:5000/ authorize add the correct openId Parameters :

- clientid
- scope
- redirect_uri
- ….

TIPS :

- it s not mandatory but your app could run with or without authent base on an ENV param (using for example process.ENV.AUTHENT_OPENID=http://localhost:5000)

Step 2 : intiate the OpenID provider

update your auth microservice app

ensure that the autorize URL is working :

- it should control the client ID : wrong clientid => error
- it should control the scope value
- it should control the redirect_uri
- if everything is OK, display a login form (login/password)

Step 3 : validate the login password

You can use static data in your code or a flat file or a database to store the login/password. if the password is NOT correct, display an error message. if password is correct :

- generate a random code
- store in your app (local variable / database / file) the code and the client login
- redirect to rederict_uri with a param code=XXXXX (XXXXX beiing the generated code)

Step 4 : implement the redirect_uri in your motus APP

Ensure that the redirect_uri is working in your motus APP :

- it should not redirect to the auth server
- it should call an API /token on your auth server that provide the code in a parameter (it is not yet

implemented and should not work)

Step 5 : implement the /token on the auth server

the API /token should validate the code received with what you have store previously it should return an id_token using the JWT format you can use the following librabry jsonwebtoken https://www.npmjs.com/package/jsonwebtoken

You can use a simple secret as provided in this example

```
npm install jsonwebtoken
```

```
var token = jwt.sign({ foo: 'bar' }, 'shhhhh');
```

Step 6 : handle the /token reponse in your motus APP

finalise the redirect_uri implementation by

- parsing the id_token (jwt format you ve just generated))
- seting the user in the session

Documentation

Update your readme file to reflect these change in your micro applications

BONUS

Keycloack instance

keycload is a great product to have an authentication provider implementing all security good practie use docker image quay.io/keycloak/keycloak:19.0.2 to spawn a keycloak instance

configure a new client, the client is going to be your motus app.

Follow the tutorial to set up your OpenID client https://auth0.com/docs/quickstart/webapp/express

other provider

- implement a login with google of login with facebook

Published with GitHub Pages