Lab 3.3: API Protection

Purpose
• The purpose of this module is to protect an API using OAuth 2.0 instead of just API keys (the old way).

Requirements
• SSO which will act as authorisation server • An API • A client (single page application)

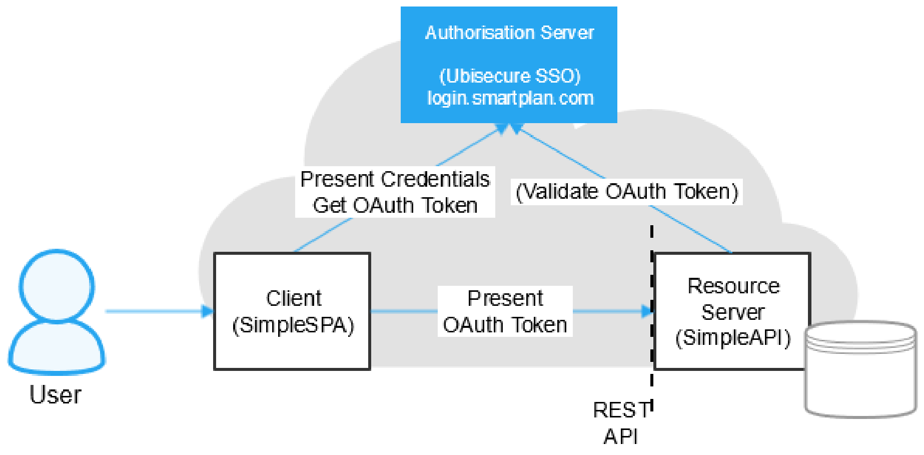
Overview

This lab configures API Protection using OAuth 2.0. The API to be protected is SmartPlan energy meter, which gives real time data on buildings and charging stations. Electric Green Cabs has developed a mobile app (Green cabs Mobile) that will fetch data from SmartPlan API. Users will log in with their existing SmartPlan credentials (CustomerID users' repository).

As a refresher on OAuth2, let us review the three main elements and its equivalent in our SmartPlan use case:

Term in OAuth 2.0	Description	In this lab
Client	For example a web browser that a person uses to access a web application	SimpleSPA (single page application)
Authorisation Server	Server that owns the user identities and credentials	Ubisecure SSO (login.smartplan.com:8443 /ubilogin)
Resource Server	The protected application	SimpleAPI (SmartPlan API)

The diagram below also shows the interactions among these three components.



All three elements will run on your local environment. Both SimpleSPA and SimpleAPI require Apache HTTP Server running.

SimpleSPA will be installed on port 5000.

SimpleAPI will be installed on port 5001.

In a real deployment, each of these services can and would be in different environments: Electric Green Cabs would host the client (mobile app), SmartPlan would run the API. For the lab, all services will be run from the same host.

Business benefits

SmartPlan is now able to securely open their APIs to third-parties like Electric Green Cabs company and ensure that data access is done with the permission of the data owner.

Access to the API can be controlled through permissions and revoked at any time.

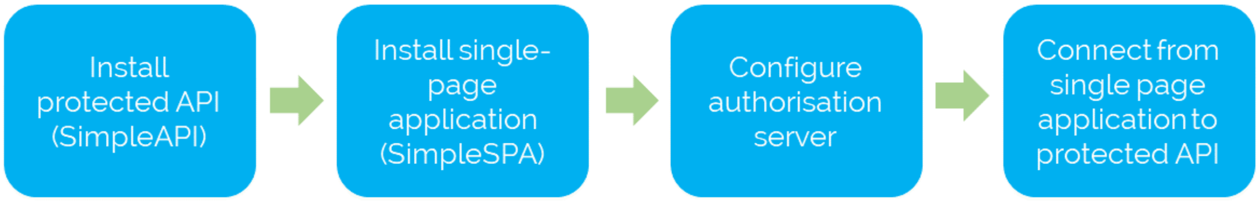
Partners and customers are able to develop innovative new services based on these APIs.

SmartPlan gets competitive advantage and possible additional revenue streams from this service.

Customers are happy that they authorise access to the API using their familiar, existing account.

Instructions

The diagram below shows the main steps on this lab.


Install protected API (SimpleAPI)
Install single-page application (SimpleSPA)
Configure authorisation server
Connect from single page application to protected API

Part 1: Install Apache HTTP server

First you need to install Apache HTTP server on your own instance.

1. Download Apache from Apache Lounge website.
2. First, install C++ Redistributable Visual Studio 2015 from https://www.apachelounge.com/download/VC14/ (SimpleAPI only works with this version). The file to download is vc_redist_x64.exe
3. Restart your computer if asked by the installer
4. To complete Apache installation, copy the file httpd-2.4.39-win64-VC14.zip (at the bottom of this page) to Downloads folder.
5. Unzip the compressed file so it will create a subdirectory \Downloads\Apache24

Part 2: Install SimpleAPI

Next you have to install SimpleAPI, a simple API example built for testing purposes that requires an OAuth2 token to use.

Step 1: Install SimpleAPI

1. Unzip the package SimpleAPI (at the bottom of this page) to Downloads\Apache24 folder. It will create the folder "C:\Users\Administrator\Downloads\Apache24\SimpleAPI-smartplan"
2. Modify file C:\Users\Administrator\Downloads\Apache24\SimpleAPI-smartplan\conf\httpd.conf. Append following lines to the end of file:

Downloads\Apache24\SimpleAPI-smartplan\conf\httpd.conf

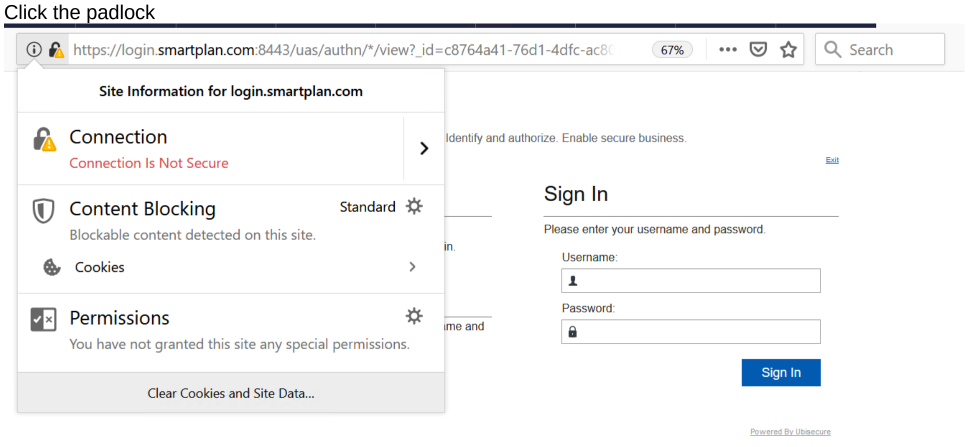
```
LoadModule ssl_module ${ModulesDir}/mod_ssl.so
<VirtualHost _default_:5001>
ServerName greencabs.com:5001
SSLEngine on
SSLCertificateFile "${ServerRoot}/conf/server.crt"
SSLCertificateKeyFile "${ServerRoot}/conf/server.key"
</VirtualHost>
```

Step 2: Install mod_auth_openidc

mod_auth_openidc is an authentication/authorisation module for the Apache 2.x HTTP server that authenticates users against an OpenID Connect Provider. It can also function as an OAuth 2.0 Resource Server, validating access tokens presented by OAuth 2.0 clients against an OAuth 2.0 Authorization Server. Source: https://www.mod-auth-openidc.org/

1. Install mod_auth_openidc from the given file mod_auth_openidc-2.3.8-apache-2.4.x-win64.zip. You will find it at the bottom of this page too. Simply extract the zip file to Downloads folder so it will create the folder "C:\Users\Administrator\Downloads\mod_auth_openidc-2.3.8-apache-2.4.x-win64".
2. The next step is adding login.smartplan.com certificate to mod_auth_openidc's trusted root store. This is required because login.smartplan.com has a self-signed certificate and by default is not trusted. First you must export login.smartplan.com certificate as a *.crt file.
3. In Firefox, go to your browser tab and open https://login.smartplan.com:8443/ubilogin

4. Click the padlock



https://login.smartplan.com:8443/uas/authn/*/view?_id=c8764a41-76d1-4dfc-ac8

Site Information for login.smartplan.com

Connection
Connection Is Not Secure

Content Blocking — Standard
Blockable content detected on this site.
Cookies

Permissions
You have not granted this site any special permissions.

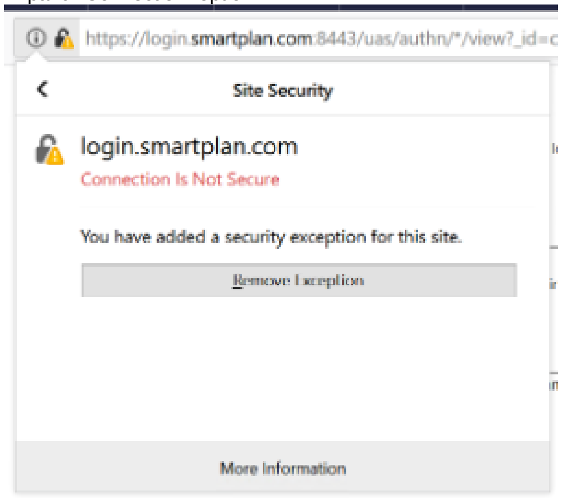
Clear Cookies and Site Data...

Identify and authorize. Enable secure business.

Sign In
Please enter your username and password.
Username:
Password:
Sign In

Powered By Ubisecure

5. Expand "Connection" option



https://login.smartplan.com:8443/uas/authn/*/view?_id=c

Site Security

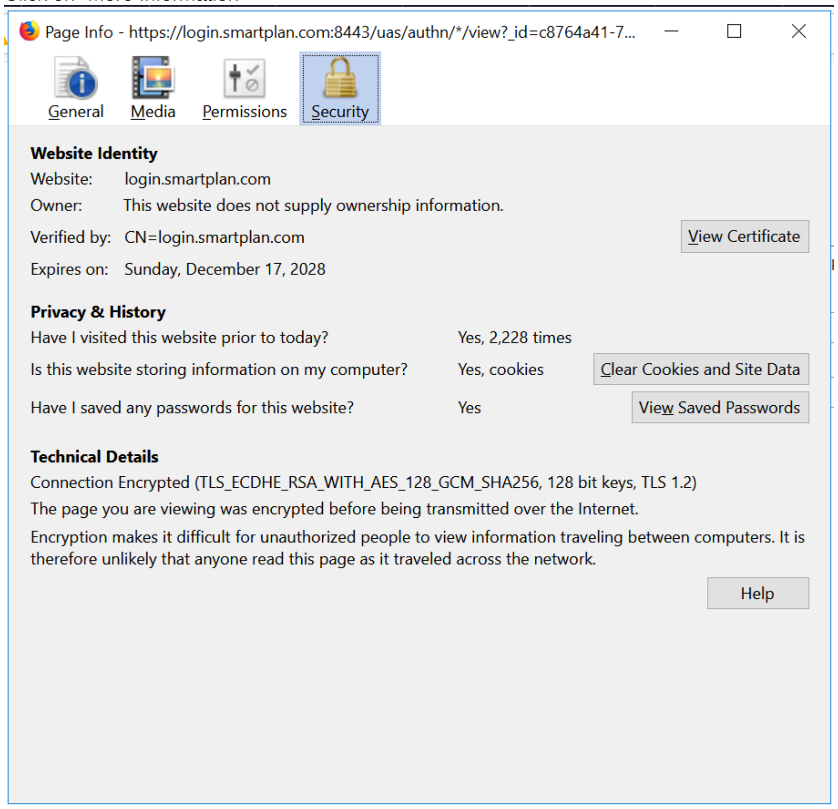
login.smartplan.com
Connection Is Not Secure

You have added a security exception for this site.

Remove Exception

More Information

6. Click on "More Information"


Page Info - https://login.smartplan.com:8443/uas/authn/*/view?_id=c8764a41-7...
General
Media
Permissions
Security

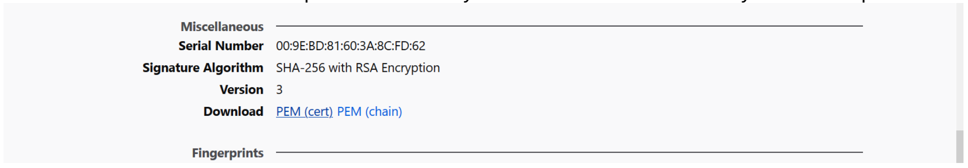
Website Identity
Website: login.smartplan.com
Owner: This website does not supply ownership information.
Verified by: CN=login.smartplan.com
Expires on: Sunday, December 17, 2028

Privacy & History

Have I visited this website prior to today?	Yes, 2,228 times	
Is this website storing information on my computer?	Yes, cookies	Clear Cookies and Site Data
Have I saved any passwords for this website?	Yes	View Saved Passwords

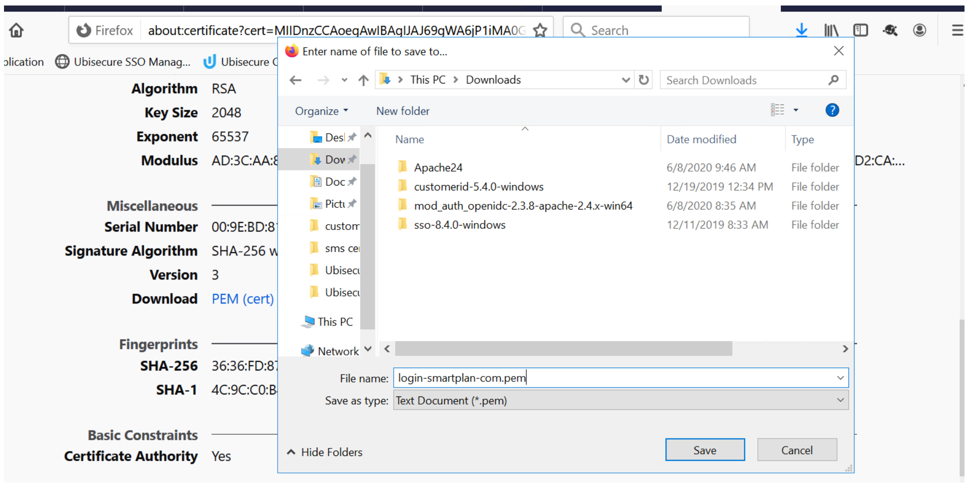
Technical Details
Connection Encrypted (TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, 128 bit keys, TLS 1.2)
The page you are viewing was encrypted before being transmitted over the Internet.
Encryption makes it difficult for unauthorized people to view information traveling between computers. It is therefore unlikely that anyone read this page as it traveled across the network.

7. Click "View Certificate." This will open a new tab on your browser. Scroll down until you find the option "Download."



Miscellaneous	
Serial Number	00:9E:BD:81:60:3A:8C:FD:62
Signature Algorithm	SHA-256 with RSA Encryption
Version	3
Download	PEM (cert) PEM (chain)
Fingerprints	

8. Click on "PEM (cert)" link and save the file to Download folder.



Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	AD:3C:AA:8 ... D2:CA:...
Miscellaneous	
Serial Number	00:9E:BD:8
Signature Algorithm	SHA-256 w
Version	3
Download	PEM (cert)
Fingerprints	
SHA-256	36:36:FD:8
SHA-1	4C:9C:C0:B
Basic Constraints	
Certificate Authority	Yes

Name	Date modified	Type
Apache24	6/8/2020 9:46 AM	File folder
customerid-5.4.0-windows	12/19/2019 12:34 PM	File folder
mod_auth_openidc-2.3.8-apache-2.4.x-win64	6/8/2020 8:35 AM	File folder
sso-8.4.0-windows	12/11/2019 8:33 AM	File folder

File name: login-smartplan-com.pem
Save as type: Text Document (*.pem)

9. Save it as "login_smartplan_com.pem" and open it with Notepad++
10. Now let's find mod_auth_openidc's trusted root store, a file titled "curl-ca-bundle.crt"
11. Open it with Notepad++ (its location is C:\Users\Administrator\Downloads\mod_auth_openidc-2.3.8-apache-2.4.x-win64\Apache24\bin).
12. Copy the full contents of "login_smartplan_com.pem" and paste it at the end of "curl-ca-bundle.crt"
13. Save "curl-ca-bundle.crt"
14. Generate a self-signed certificate. Open a command prompt and execute these two commands:

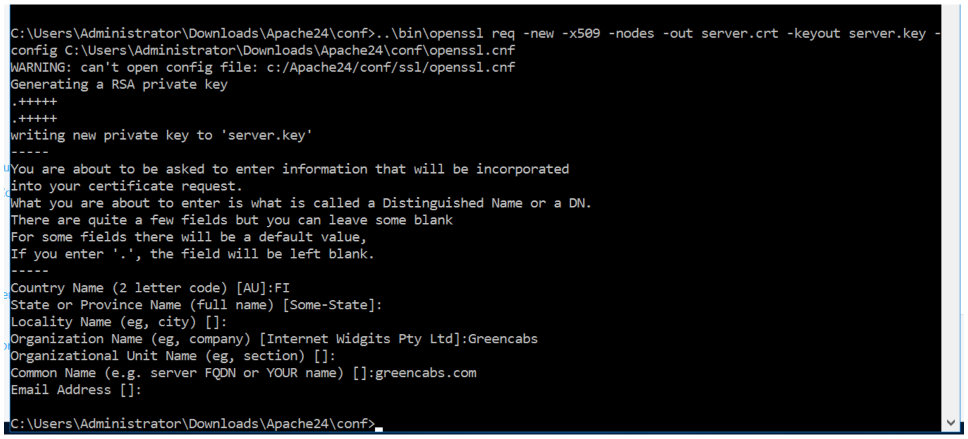
```
cd \Users\Administrator\Downloads\Apache24\conf

..\bin\openssl req -new -x509 -nodes -out server.crt -keyout server.key -config C:
\Users\Administrator\Downloads\Apache24\conf\openssl.cnf
```

15. You wil be prompted to enter values several times and press "Enter" to continue. The values you must fill are:

Country Name (2 letter code) [AU]:	FI
Organization Name (eg, company) [Internet Widgits Pty Ltd]:	Greencabs
Common Name (e.g. server FQDN or YOUR name) []:	greencabs.com

All the other fields you can leave empty.
After successful completion will look like this:



```
C:\Users\Administrator\Downloads\Apache24\conf>..\bin\openssl req -new -x509 -nodes -out server.crt -keyout server.key -
config C:\Users\Administrator\Downloads\Apache24\conf\openssl.cnf
WARNING: can't open config file: c:/Apache24/conf/ssl/openssl.cnf
Generating a RSA private key
.+++++
.+++++
writing new private key to 'server.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FI
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Greencabs
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:greencabs.com
Email Address []:

C:\Users\Administrator\Downloads\Apache24\conf>_
```

Step 3: Invoke SimpleAPI

1. Open the command prompt
2. Run **cd %USERPROFILE%\Downloads\Apache24\SimpleAPI-smartplan**
3. Then execute **`run-apache.cmd`** to start Apache HTTP server
4. The API is now running in https://greencabs.com:5001/simple. Open the URL on a browser and accept the security warnings. You will get a 401 Unauthorized error. That is fine because you still need a client to invoke the API. In next section (Part 3) you will install a single page application (the client) to invoke the API and get a response from it.
5. Keep the command prompt window running. Don't close it until asked.

localhost

Windows' hosts file has been already edited so localhost redirects to greencabs.com

127.0.0.1   greencabs.com

Part 3: Customise SimpleAPI to your authorisation server

Your local SSO installation will become the authorisation server. Now you must customise SimpleAPI to connect it with SSO (authorisation server) via OAuth2.

On SSO Management Console

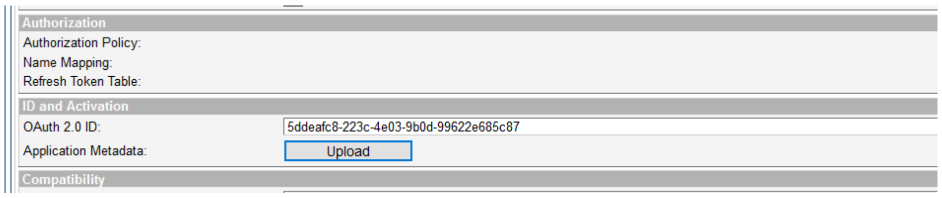
1. In SmartPlan site, create new application "SimpleAPI." Select type "OAuth 2.0," tick Enabled box and press OK button.
2. Under "ID and Activation" press "Activate" button. This will generate a client_id and client_secret.
3. Save generated file as simpleapi.json in a temporary folder (e.g. Downloads).
4. Press "Update" at the bottom

5. You must also upload metadata. First, create a JSON file with the minimum metadata needed using the example below. As this is an API, it's important that redirect_uris and grant_types are empty.

metadata-api-client.json

```
{"redirect_uris":[] ,
"grant_types":[]
}
```

6. Save this JSON file as metadata-api-client-json and then upload the file as seen below.



Authorization
Authorization Policy:
Name Mapping:
Refresh Token Table:
ID and Activation
OAuth 2.0 ID: 5ddeafc8-223c-4e03-9b0d-99622e685c87
Application Metadata: Upload
Compatibility

7. Press "Update" at the bottom
8. Finally, add "CustomerID Password" in Allowed Methods, and "eIDMUser" in Allowed To tab.

On the Simple API

We'll now configure client_id and client_secret to the SimpleAPI configuration, to integrate SSO and the SimpleAPI.

1. Edit httpd.conf file with the values of your server. Open C:\Users\Administrator\Downloads\Apache24\SimpleAPI-smartplan\conf\httpd.conf in Notepad++.
2. Below you can see the three lines to be edited. Replace YOUR-CLIENT-ID and YOUR-CLIENT-SECRET with the actual values from simpleapi.json

C:\Users\Administrator\Downloads\Apache24\SimpleAPI-smartplan\conf\httpd.conf

```
# OAuth 2.0 resource server

OIDCOAuthIntrospectionEndpoint https://login.smartplan.com:8443/uas/oauth2/introspection

OIDCOAuthClientID YOUR-CLIENT-ID
OIDCOAuthClientSecret YOUR-CLIENT-SECRET
```

3. Save the file. Now the simple API is configured to connect to your SSO installation.

When an API request comes with an access token, the mod_auth_openidc module in front of the SimpleAPI app will call the OAuth2 Token Instrospection Endpoint (https://login.smartplan.com:8443/uas/oauth2/introspection), using the client_id and client_secret and credentials. In response, information about the token will be returned in JSON format - if the token is valid, access will be granted and the API will return results.

Part 4: Install and start SimpleSPA

Now you need a client that connects to the API. For this purpose, Ubisecure has designed a single page application called SimpleSPA, which runs on Apache HTTP Server too.

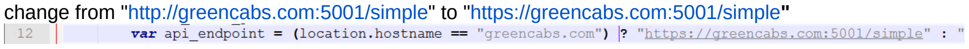
As you have Apache already installed, you can copy the single page application to your computer

1. Unzip the package SimpleSPA-greencabs.zip (at the bottom of this page) to Downloads\Apache24 folder. It will create the folder "C:\Users\Administrator\Downloads\Apache24\SimpleSPA-greencabs"
2. Modify file C:\Users\Administrator\Downloads\Apache24\SimpleSPA-greencabs\conf\httpd.conf. Append following lines to the end of the file:

Downloads\Apache24\SimpleSPA-greencabs\conf\httpd.conf

```
LoadModule ssl_module ${ModulesDir}/mod_ssl.so
<VirtualHost _default_:5000>
ServerName greencabs.com:5000
SSLEngine on
SSLCertificateFile "${ServerRoot}/conf/server.crt"
SSLCertificateKeyFile "${ServerRoot}/conf/server.key"
</VirtualHost>
```

3. Modify file C:\Users\Administrator\Downloads\Apache24\SimpleSPA-greencabs\docs\spa.html. Open the file with Notepad++ and in line 12 change from "http://greencabs.com:5001/simple" to "https://greencabs.com:5001/simple**"**



```
12          var api_endpoint = (location.hostname == "greencabs.com") ? "https://greencabs.com:5001/simple" : "
```

4. Open another command prompt without closing the one used for SimpleAPI.
5. Run **cd %USERPROFILE%\Downloads\Apache24\SimpleSPA-greencabs**
6. Then execute **`run-apache.cmd`** to start this other Apache HTTP server instance
7. Now the single page application is running. Open https://greencabs.com:5000/spa.html in a browser
8. You can load the application, but can't login yet. The next step is configuring the authorisation server.

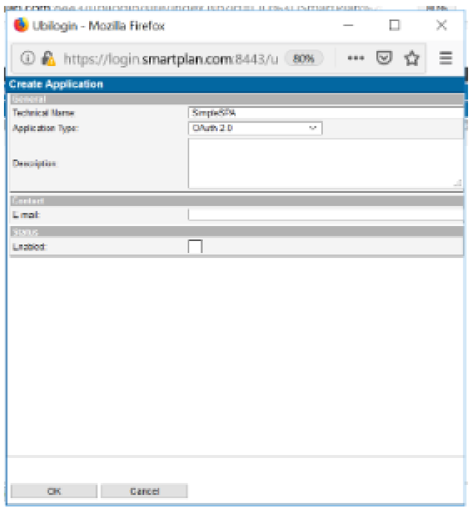
Part 5: Register SimpleSPA to the Authorisation Server

Now that the single page application is running, we will configure it to connect to our local SSO installation (login.smartplan.com).

On SSO Management Console

The first part of the configuration is done in SSO Management Console (https://login.smartplan.com:8443/ubilogin).

1. In SmartPlan site, create new application "SimpleSPA." Select type "OAuth 2.0," tick "Enabled" box and press OK button.



Ubilogin - Mozilla Firefox

https://login.smartplan.com:8443/u

Create Application

General

Technical Name: SimpleSPA

Application Type: OAuth 2.0

Description:

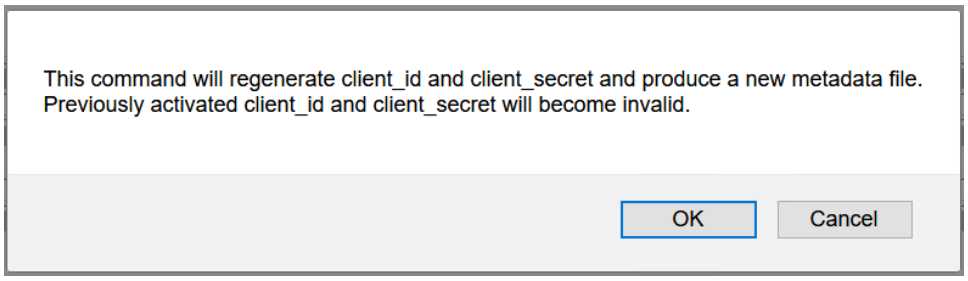
Contact

E-mail:

Status

Enabled: ☐

2. Under "ID and Activation" press "Activate" button. This will generate a client_id and client_secret.



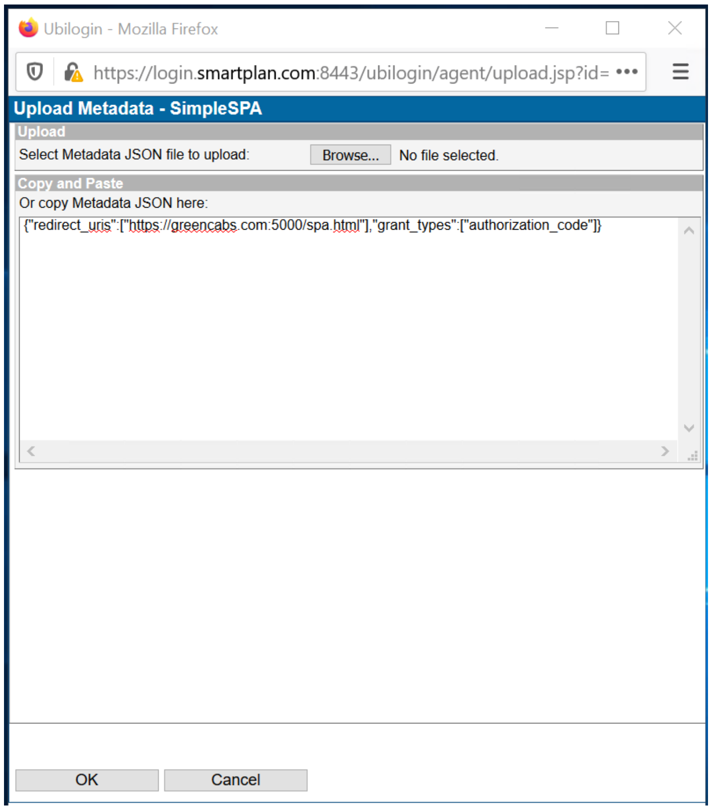
This command will regenerate client_id and client_secret and produce a new metadata file.
Previously activated client_id and client_secret will become invalid.

3. Save generated file simplespa.json on a temporary folder such as Downloads. You will use it for next step.
4. Press "Update" at the bottom

5. Also under "ID and Activation" you will see "Application Metadata." Click "Upload" button and in the pop up windows that opens enter the following text:

```
{"redirect_uris":["https://greencabs.com:5000/spa.html"],"grant_types":["authorization_code"]}
```

6. Click "OK"


Ubilogin - Mozilla Firefox
https://login.smartplan.com:8443/ubilogin/agent/upload.jsp?id=
Upload Metadata - SimpleSPA
Upload
Browse...
Copy and Paste
OK
Cancel

Select Metadata JSON file to upload: No file selected.

Or copy Metadata JSON here:

```
{"redirect_uris":["https://greencabs.com:5000/spa.html"],"grant_types":["authorization_code"]}
```

7. Press "Update" at the bottom.
8. This has specified the redirect URI of the application and that Authorization Code is the OAuth2 grant type that will be used.
9. Finally, add "CustomerID Password" in Allowed Methods, and "eIDMUser" in Allowed To tab.

On the Single Page Application

We will now use the client_id and client_secret generated in the previous step. This will complete the integration between SSO and the single page application, so the single page application will get an access token for API calls.

The single page application's logic is on file C:\Users\Administrator\Downloads\Apache24\SimpleSPA-greencabs\docs\spa.html. You need to edit a few lines on it to match the authorisation server.

1. First, edit line 9:

spa.html

```
var issuer = "https://login.smartplan.com:8443/uas";
```

2. The next step is to add the client_id and client_secret values, which are inside the file simplespa.json you saved earlier.
3. Edit line 229 as explained in the table below

```
229                     .then(config => invokeTokenRequest(config, "SimpleSPA", "public", code)
```

Parameter	Replace it with
SimpleSPA	value of client_id on simplespa.json
public	value of client_secret on simplespa.json

4. When the single page application request the access token, it must specify the target application of that access token. In this case is the SimpleAPI. Finally, edit line 273 as explained in the table below

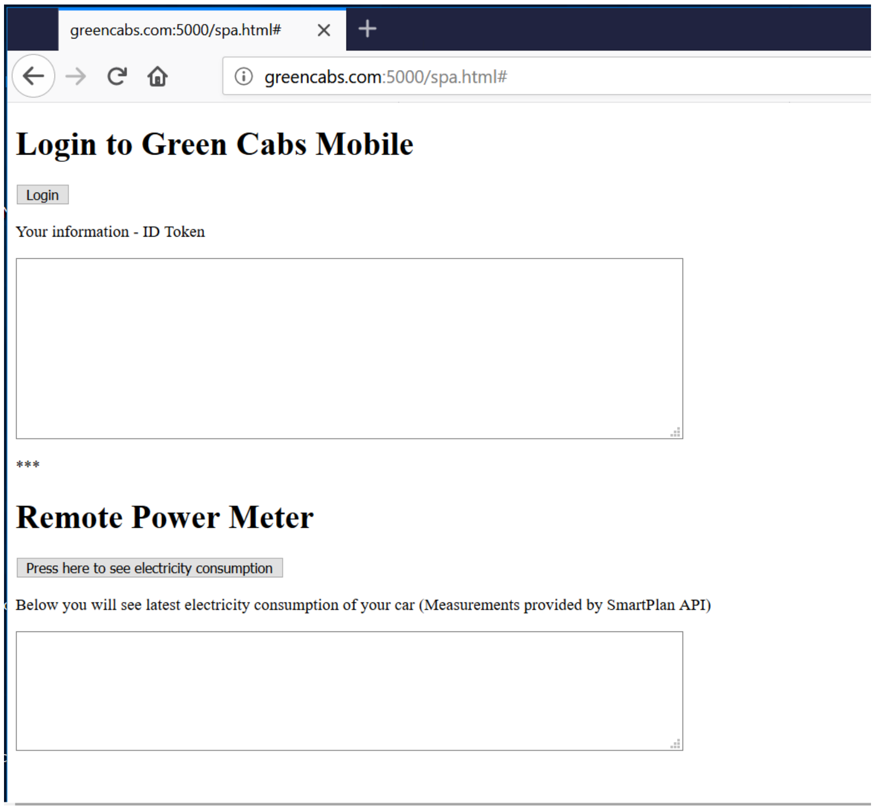
```
273                 .then(config => sendAuthenticationRequest(config, "SimpleSPA", "openid api"))
```

Parameter	Replace it with
SimpleSPA	value of client_id on simplespa.json
api	value of client_id on simpleapi.json

5. Now the single page application is configured.

Part 6: Connect from your single page application to the protected API

1. First, restart both the API and the single page application. Do Ctrl-Z on the two command prompts, and then start them again as when were invoked for the first time.
2. Open https://greencabs.com:5000/spa.html on your browser:


greencabs.com:5000/spa.html#
greencabs.com:5000/spa.html#

Login to Green Cabs Mobile

Your information - ID Token

***

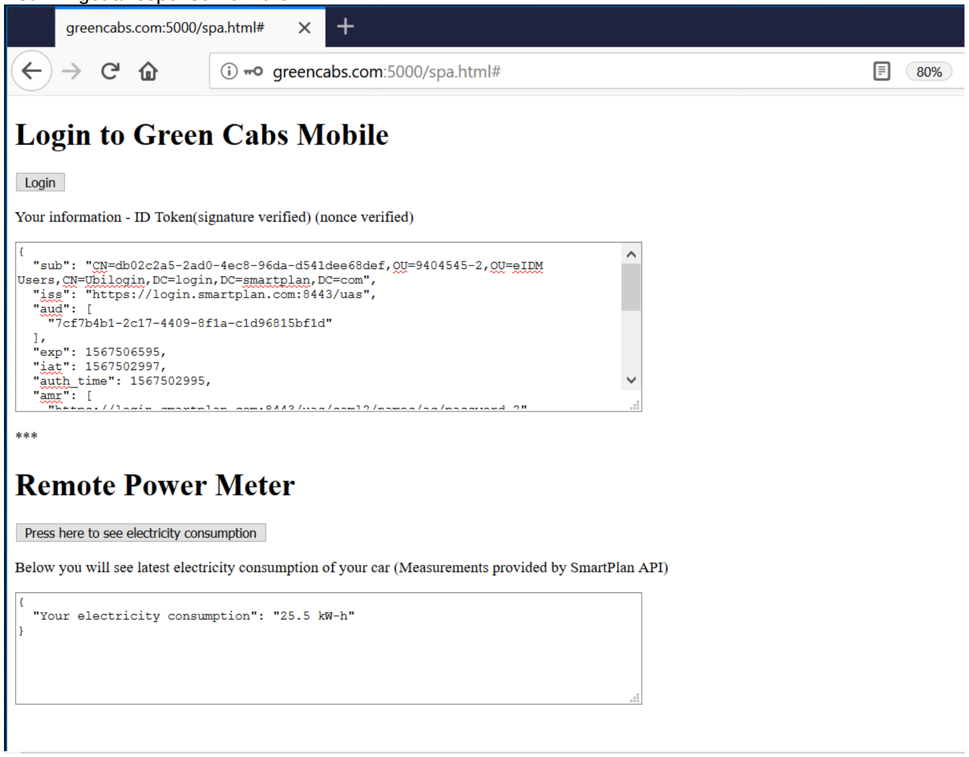
Remote Power Meter

Press here to see electricity consumption

Below you will see latest electricity consumption of your car (Measurements provided by SmartPlan API)

3. Click "Login" button and authenticate with your CustomerID user account. Use Karl Kearnes (karl@kokomedia.local) account which was created in Lab 2.2: CustomerID Workflow Configuration. Password: Qwerty1234
4. You will see your ID Token on the upper box.
5. Next, invoke the API

6. You will get a response from the API



Login to Green Cabs Mobile

Login

Your information - ID Token(signature verified) (nonce verified)

```
{
  "sub": "CN=db02c2a5-2ad0-4ec8-96da-d541dee68def,OU=9404545-2,OU=eIDM
Users,CN=Ubilogin,DC=login,DC=smartplan,DC=com",
  "iss": "https://login.smartplan.com:8443/uas",
  "aud": [
    "7cf7b4b1-2c17-4409-8f1a-c1d96815bf1d"
  ],
  "exp": 1567506595,
  "iat": 1567502997,
  "auth_time": 1567502995,
  "amr": [
```

***

Remote Power Meter

Press here to see electricity consumption

Below you will see latest electricity consumption of your car (Measurements provided by SmartPlan API)

```
{
  "Your electricity consumption": "25.5 kW-h"
}
```

Part 7: Create an authorisation policy

At this point, all CustomerID users can log in to Green Cabs Mobile and from there can call SmartPlan API. The next step is to make that only a group of authorised users can access the application and API.

Another inconvenience you might have noticed is that the "sub" field in the ID Token reveals too much information about the data structure of the authorisation server .We'll use Persistent ID Mapping to solve this challenge.

Persistent ID Mapping

SSO Management Console allows user mappings. Mappings are used to map a Ubisecure user's user id to a user id required by an application.

Persistent ID (UUID format) mapping sends a Persistent ID to the target application. This ID will not change after logout so every time the user logs in, the same ID will be used.

For more information, visit the page Management UI Mappings - SSO

Let's do the following two improvements:

1. Mask user distinguished name by creating and assigning a Persistent ID Mapping. Thus the authorisation server will send that persistent ID instead of the distinguished name, and will appear in ID token's "sub" field.
2. Create an authorisation policy that:
   a. Allows access to the application and API only to a new group called "SmartPlan API users." Include only Karl Kearnes in this group.
   b. Has two attributes so you can visualize name and surname along with the ID token fields.

Step 1: Create a Persistent ID Mapping

Open SSO Management Console.

Go to SmartPlan site

Mappings

New Mapping ...

Name: Green Cabs persistent ID

NameID Format: Persistent ID (UUID format)

Finally, connect the Persistent ID Mapping to the SimpleSPA application

Step 2: Create an authorisation policy

Go to SmartPlan site

Groups

New Group

Create group: SmartPlan API users

Users

Add Karl Kearnes

In Allowed Applications, add both SimpleAPI and SimpleSPA

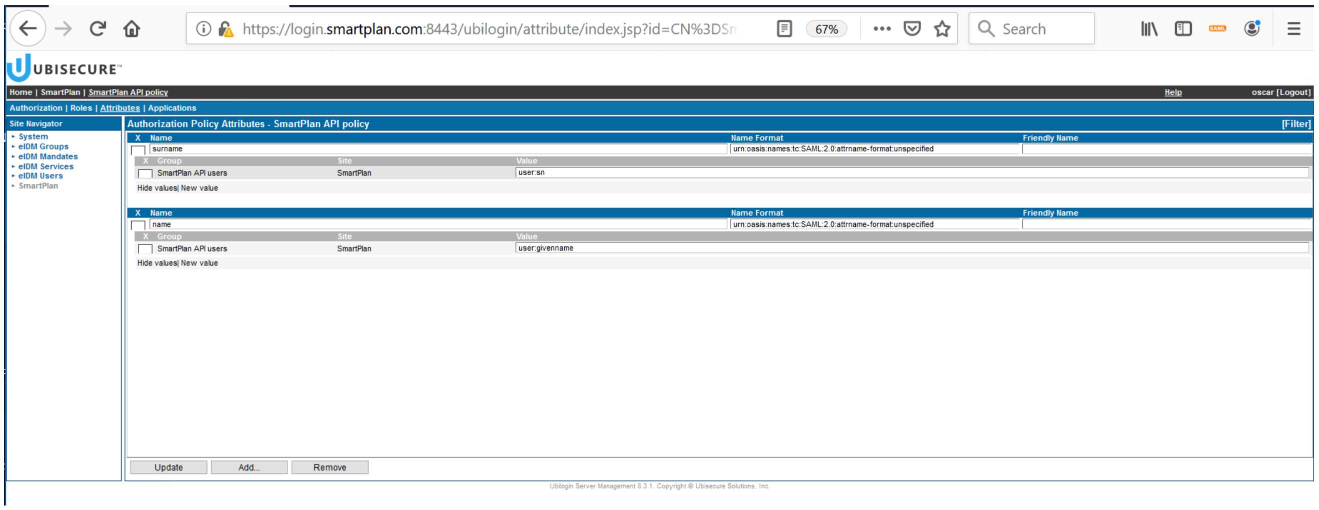
Go to SmartPlan site

Authorization Policies

New policy

Create "SmartPlan API Policy"

In Attributes add name and surname


https://login.smartplan.com:8443/ubilogin/attribute/index.jsp?id=CN%3DSm

Home | SmartPlan | SmartPlan API policy

Authorization | Roles | Attributes | Applications

Authorization Policy Attributes - SmartPlan API policy

X	Name	Name Format	Friendly Name
	surname	urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified	

X	Group	Site	Value
	SmartPlan API users	SmartPlan	user:sn

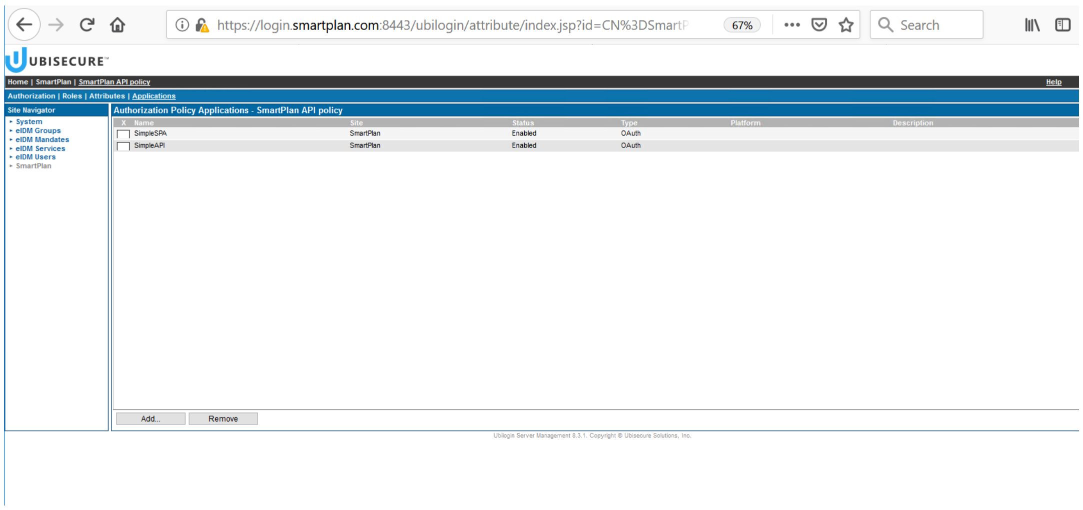
Hide values| New value

X	Name	Name Format	Friendly Name
	name	urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified	

X	Group	Site	Value
	SmartPlan API users	SmartPlan	user:givenname

Hide values| New value

In "Applications" tab add both SimpleAPI and SimpleSPA


https://login.smartplan.com:8443/ubilogin/attribute/index.jsp?id=CN%3DSmartP

Home | SmartPlan | SmartPlan API policy

Authorization | Roles | Attributes | Applications

Authorization Policy Applications - SmartPlan API policy

X	Name	Site	Status	Type	Platform	Description
☐	SimpleSPA	SmartPlan	Enabled	OAuth		
☐	SimpleAPI	SmartPlan	Enabled	OAuth		

You still need to do a final step. At this point all users on CustomerID repository will be able to log in, as earlier in this lab you authorised them all. Now, remove eIDMUser group from both applications SimpleSPA and SimpleAPI. After that, only users from the new group "SmartPlan API users" will be authorised.

Once this is done, verify that Karl Kearnes can access the API but other normal users can't. Use Mary Smith (the user you created on Lab 2.2: CustomerID Workflow Configuration, Part 1)

Additional Exercise (Optional): Use Postman to view the access tokens

SimpleSPA (Green Cabs Mobile) has allowed you to see the contents of ID tokens.

Now use Postman as the client to authenticate yourself and visualize the access tokens.

Use the access token to get response from SmartPlan API.

***** END OF THIS LAB *****

For more information

The code used in this exercise was developed by Petteri Stenius. Full repository of SimpleAPI and SimpleSPA are available at GitHub:

https://github.com/psteniusubi/SimpleAPI

Obs: to install SimpleAPI properly, you need to install Visual Studio C++ 2015 aka VC14, newer versions don't work with mod_auth_openidc.

https://github.com/psteniusubi/SimpleSPA

Files needed for this lab:

Apache HTTP Server installation package

httpd-2.4.39-win64-VC14.zip

SimpleAPI

SimpleAPI-smartplan.zip

mod_auth_openidc

mod_auth_openidc-2.3.8-apache-2.4.x-win64.zip

SimpleSPA

SimpleSPA-greencabs.zip