TRAINING

Adding Identity Management and Access Control to your app

Alvaro Alonso, Cyril Dangerville Security Chapter

FIWARE

Identity Manager

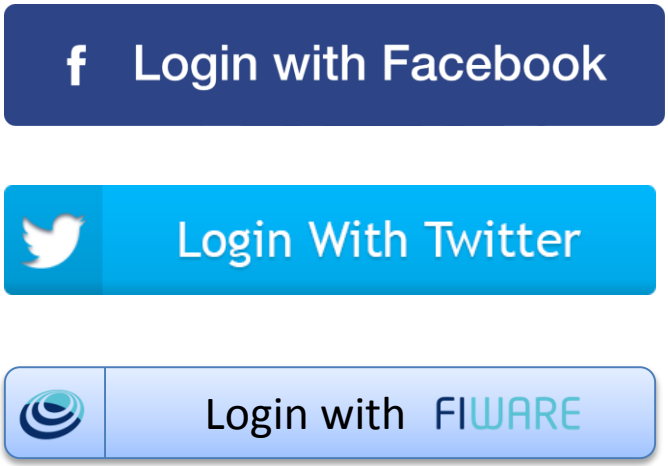
FIWARE

Identity Manager

OAuth 2.0

FIWARE

OAuth 2.0



OAuth 2.0

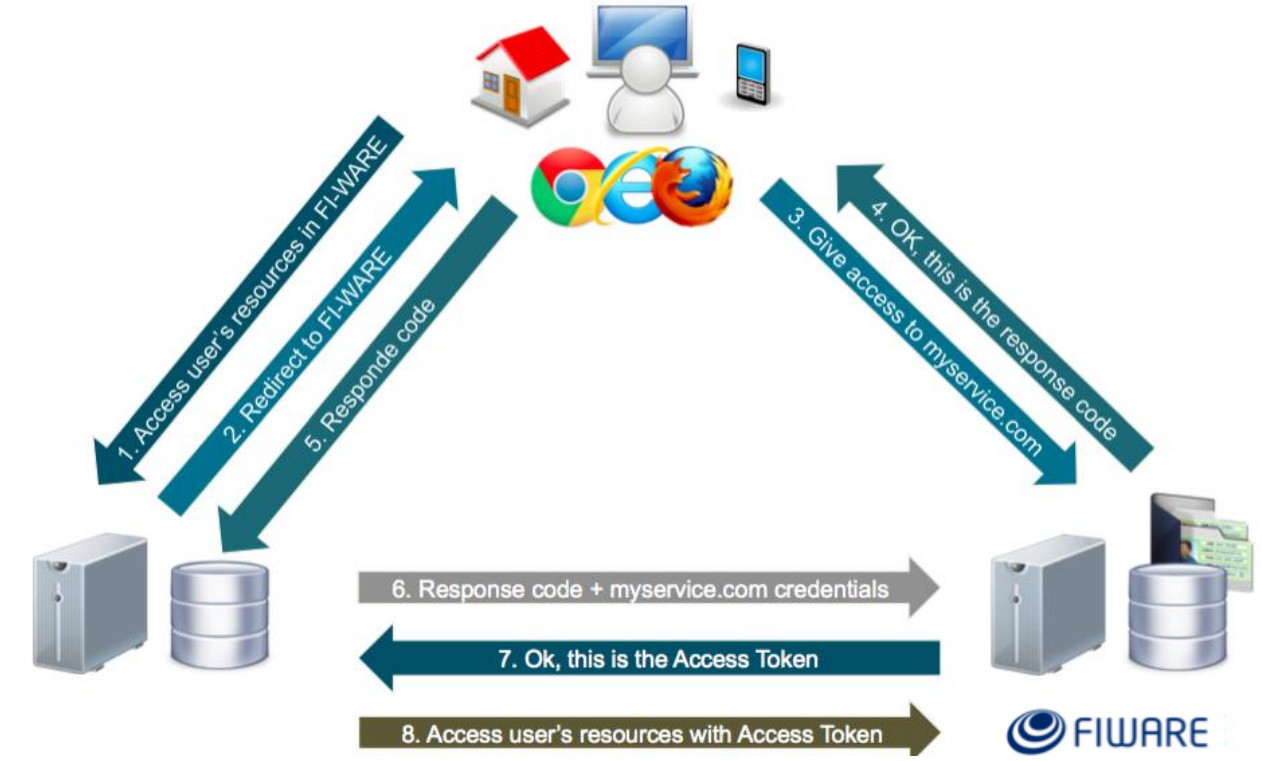
- Mechanism to provide applications access to restricted resources **without sharing credentials**.
  - Applications use **access tokens**, issued by OAuth providers (e.g. FIWARE), to access resources.
  - OAuth 2.0 specification is designed for use with HTTP.

- Roles:
  - **Resource Owner**: Entity capable of granting access to a protected resource (e.g. end-user)
  - **Resource Server**: Server hosting protected resources.
  - **Client**: Application making protected resource requests on behalf of the resource owner.
  - **Authorization Server**: The server issuing access tokens to the client.

FIWARE

OAuth 2.0 Architecture

Authorization Code Grant



OAuth consumer
myservice.com

OAuth Provider
account.lab.fiware.org

Using OAuth2
Authorization Code Grant

Authorization Request

```
GET /oauth2/authorize?response_type=code&client_id=1&state=xyz
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcallback_url HTTP/1.1
Host: account.lab.fiware.org
```

FIWARE

Using OAuth2
Authorization Code Grant

Authorization Response

```
HTTP/1.1 302 Found
Location: https://client.example.com/callback_url?code=SplxlOBeZQQYbYS6WxSbIA&state=xyz
```

Using OAuth2
Authorization Code Grant

Access Token Request

```
POST /oauth2/token HTTP/1.1
Host: account.lab.fiware.org
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcallback_url
```

FIWARE

Using OAuth2
Authorization Code Grant

Access Token Response

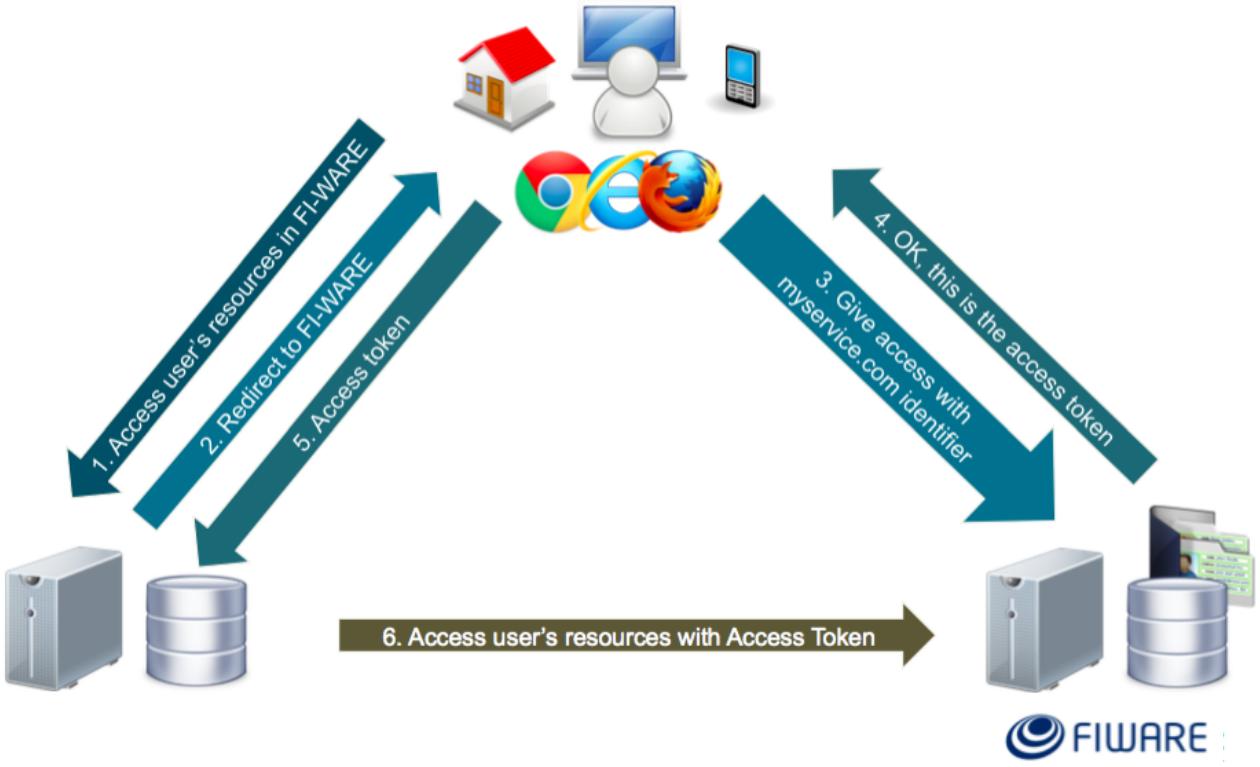
```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
    "access_token":"2YotnFZFEjr1zCsicMWpAA",
    "token_type":"bearer",
    "expires_in":3600,
    "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
}
```

FIWARE

OAuth 2.0 Architecture

Implicit Grant

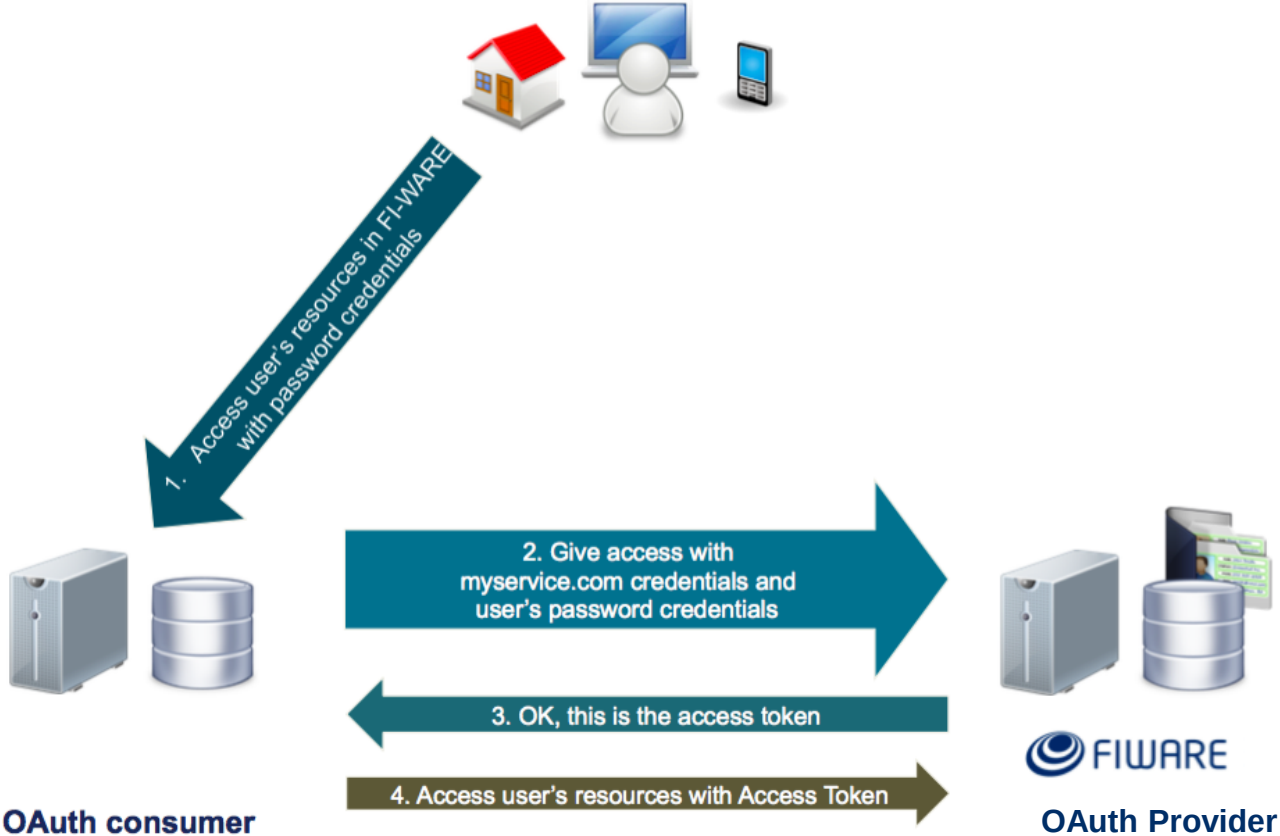


OAuth consumer
myservice.com

OAuth Provider
account.lab.fiware.org

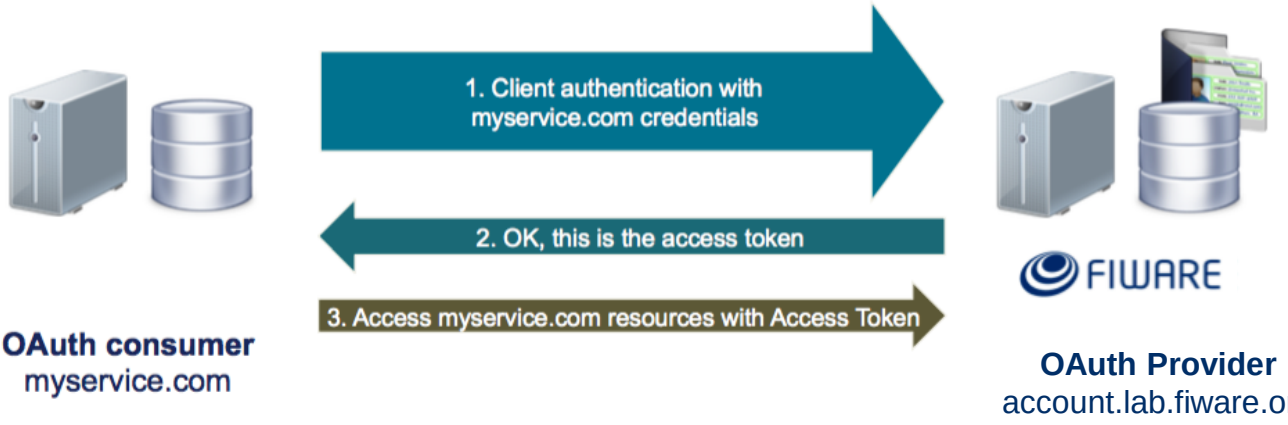
OAuth 2.0 Architecture

Resource Owner Password Credentials Grant



OAuth 2.0 Architecture

Resource Owner Password Credentials Grant



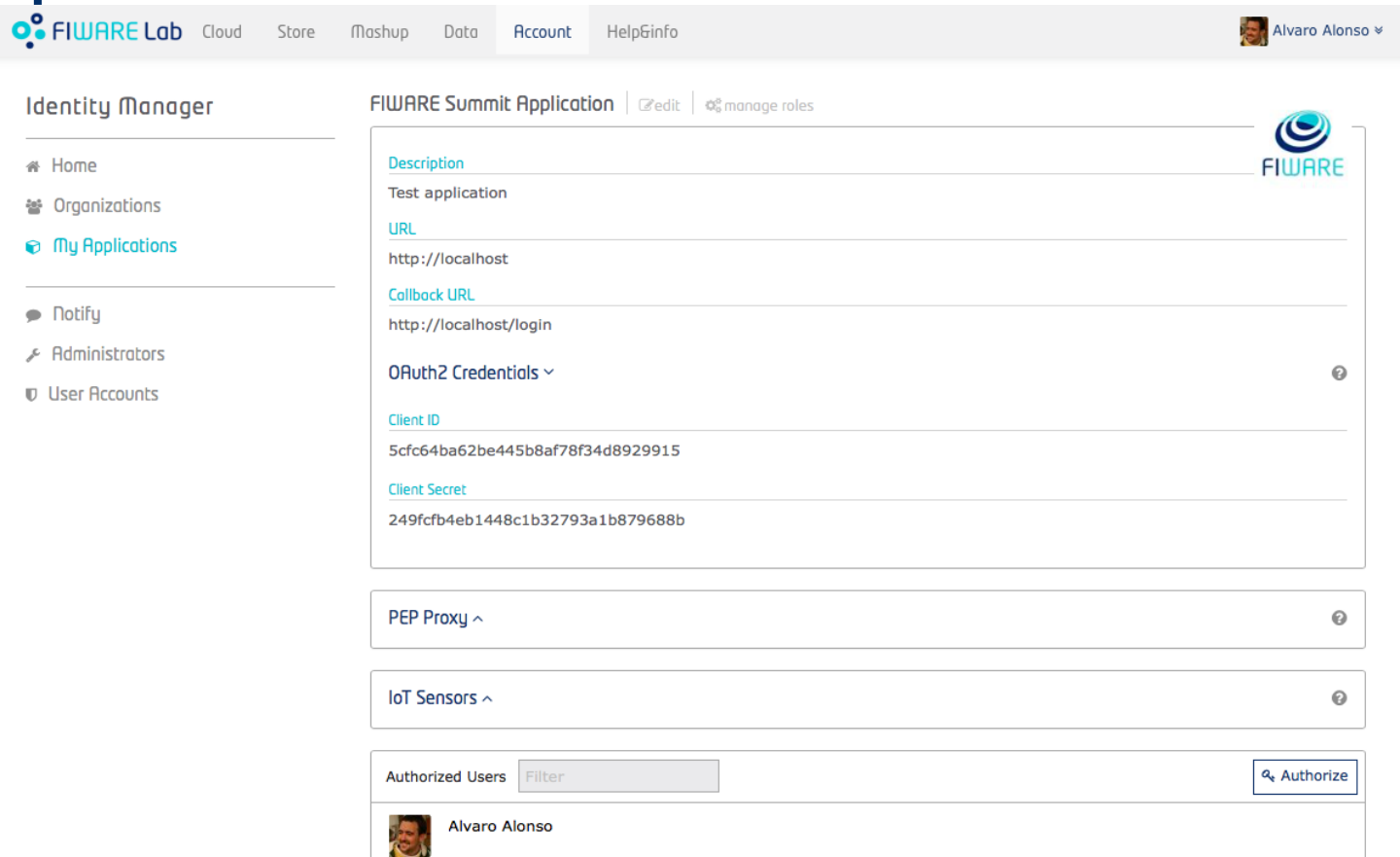
FIWARE

OAuth 2.0 Libraries

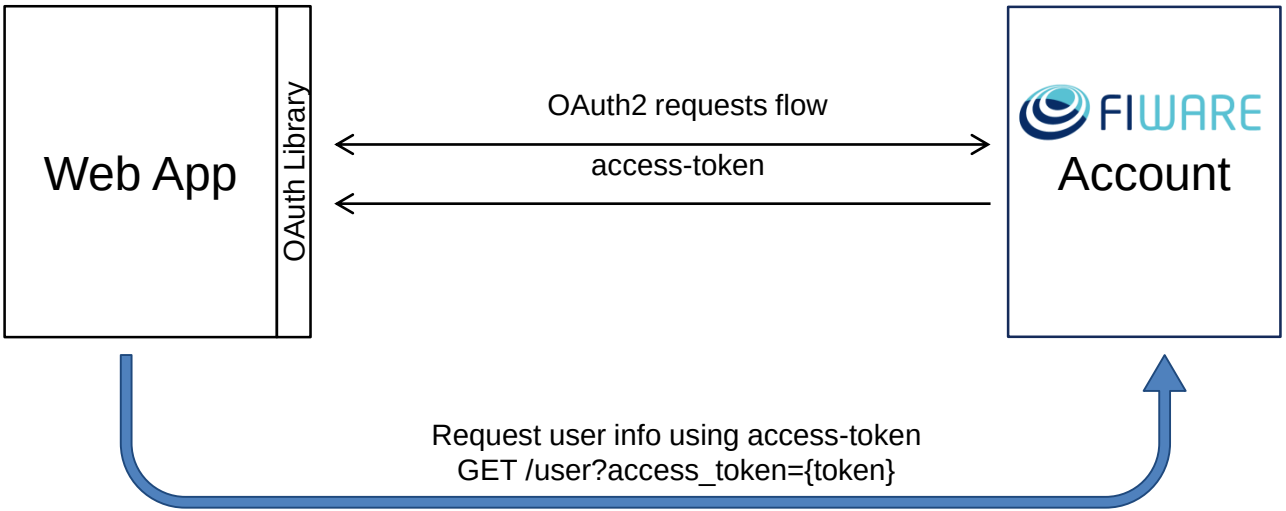
- http://oauth.net/2/
  - PHP, Cocoa, iOS, Java, Ruby, Javascript, Python.

- Example using Node.js
  - https://github.com/ging/oauth2-example-client

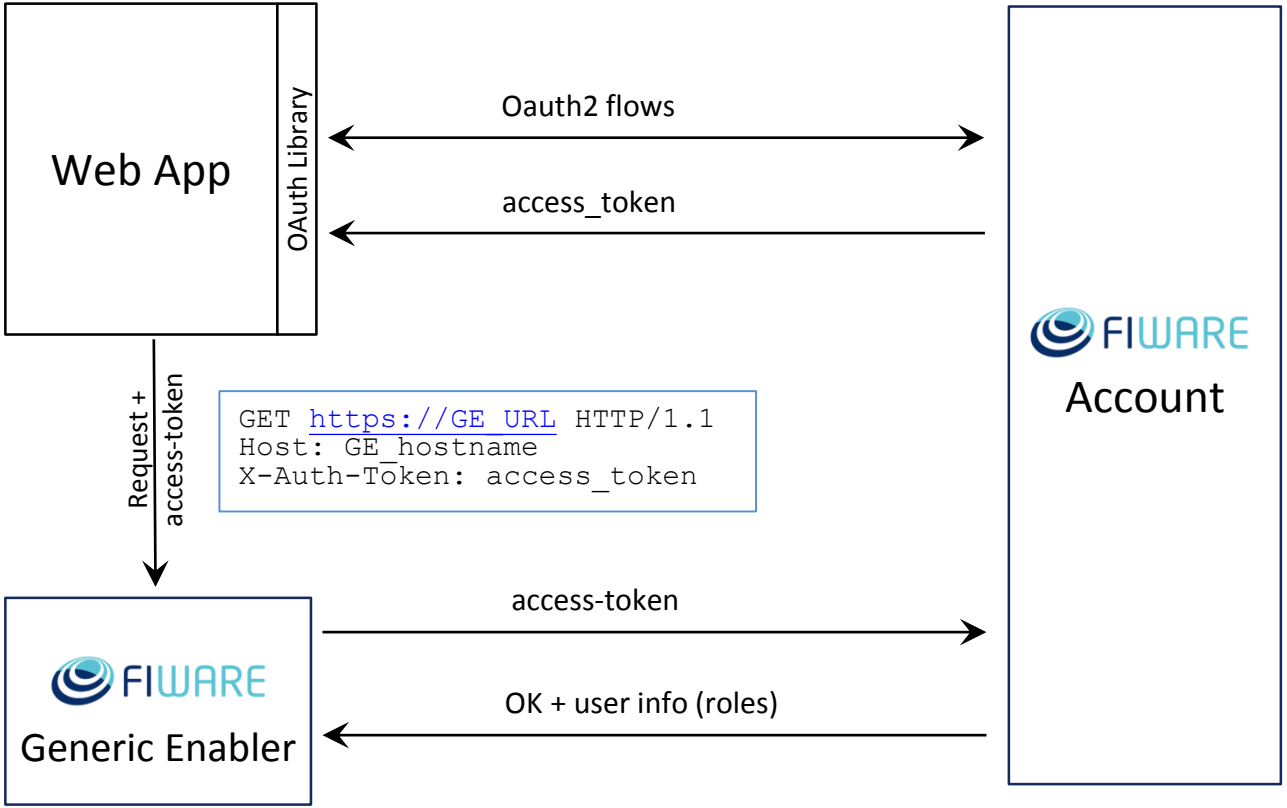
OAuth2 credentials in FIWARE Account



Getting protected user info

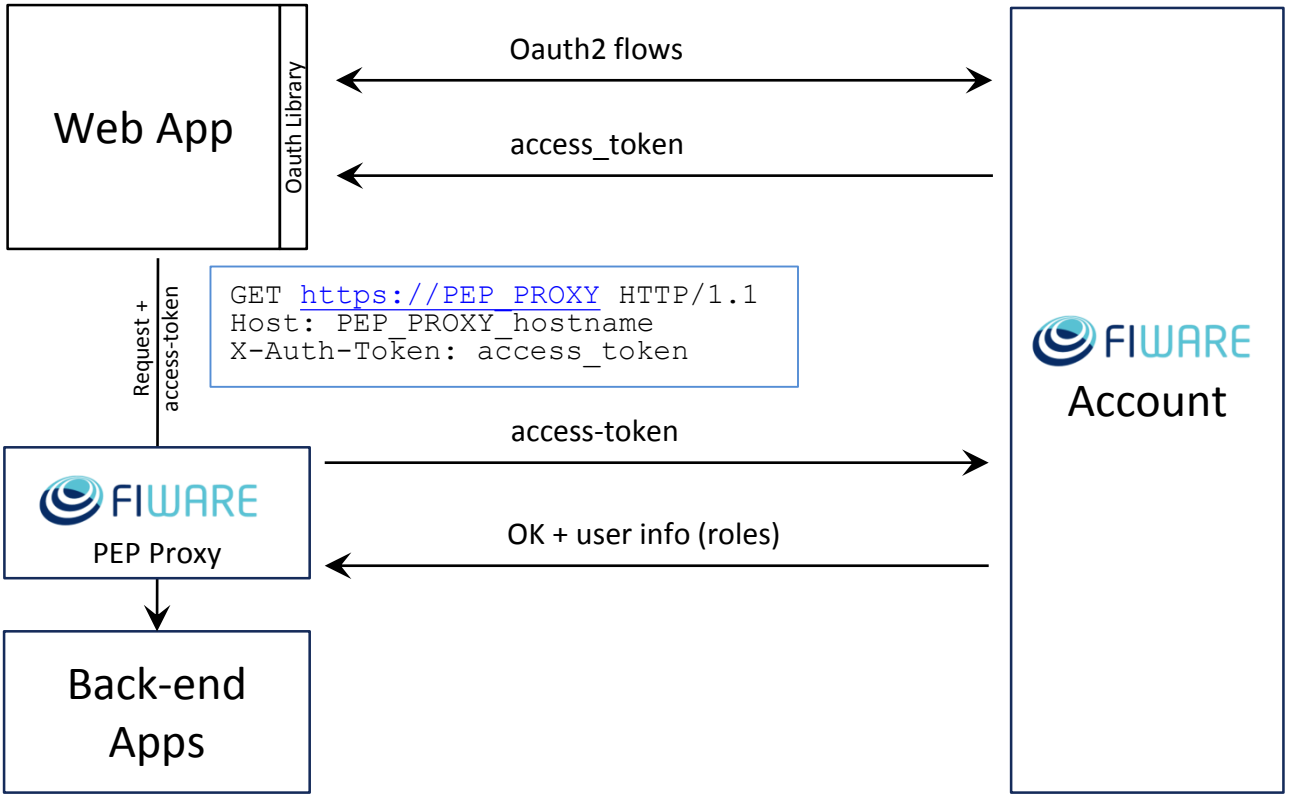


Web Applications and GEs



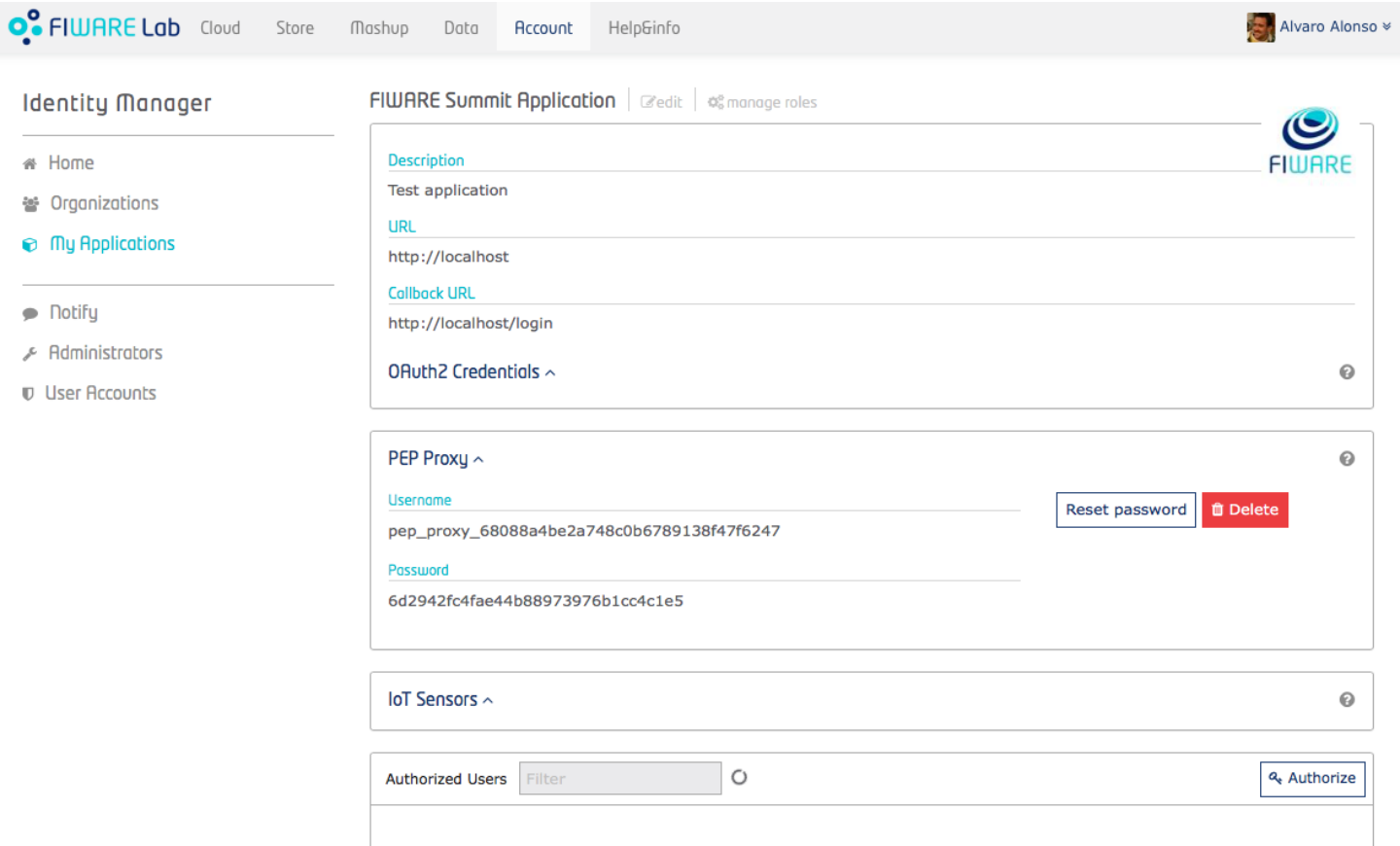
FIWARE

Securing your back-end



FIWARE

PEP Proxy in FIWARE Lab Account

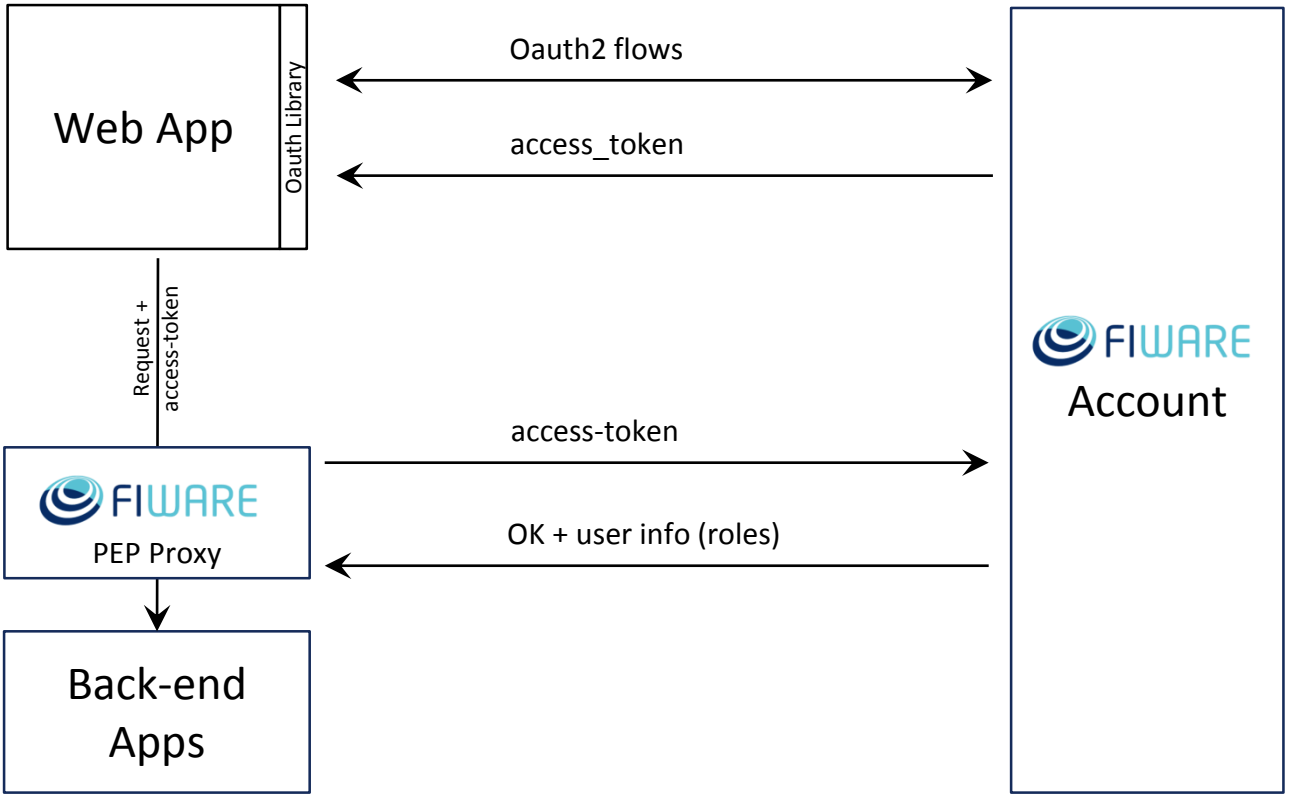


Securing your back-end

- Level 1: Authentication
  - Check if a user has a FIWARE account
- Level 2: Basic Authorization
  - Checks if a user has permissions to access a resource
  - HTTP verb + resource path
- Level 3: Advanced Authorization
  - Custom XACML policies

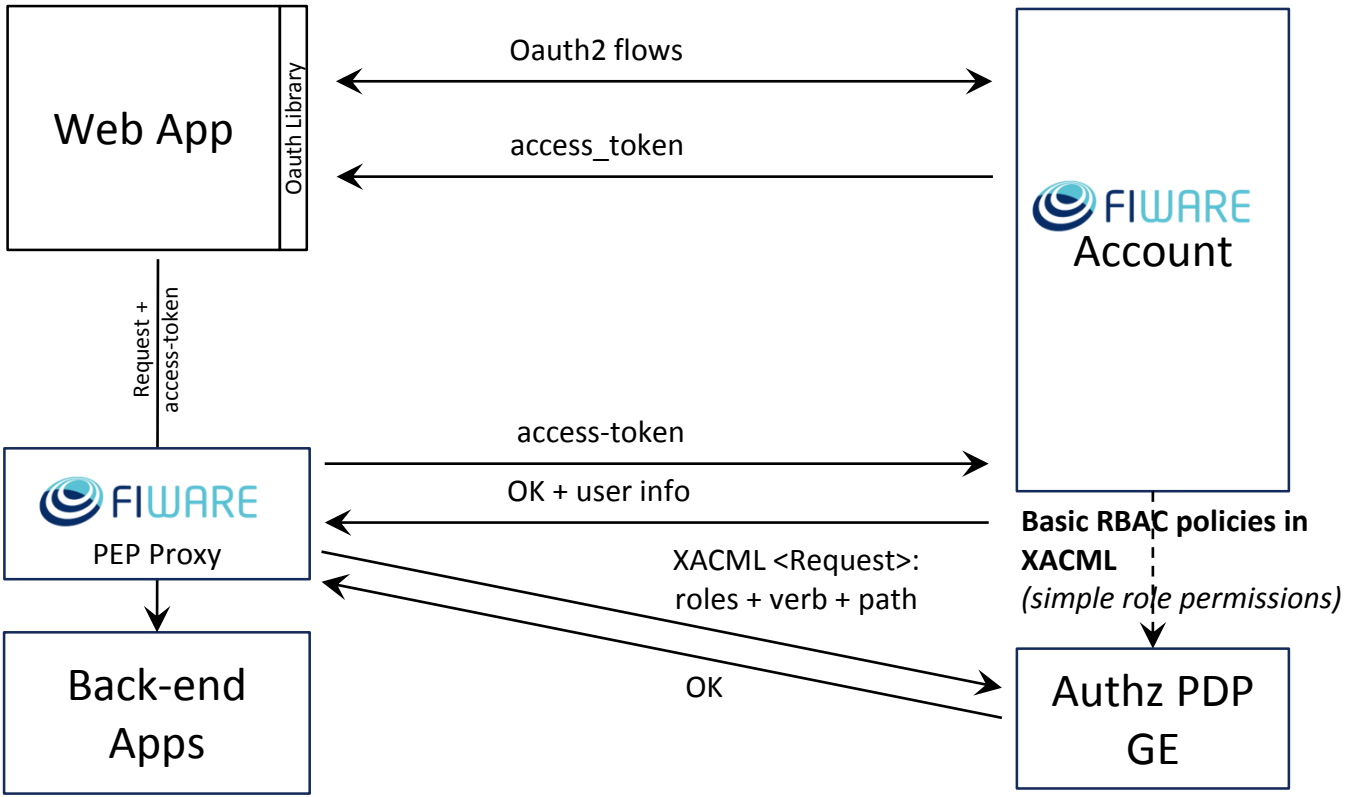
FIWARE

Level 1: Authentication



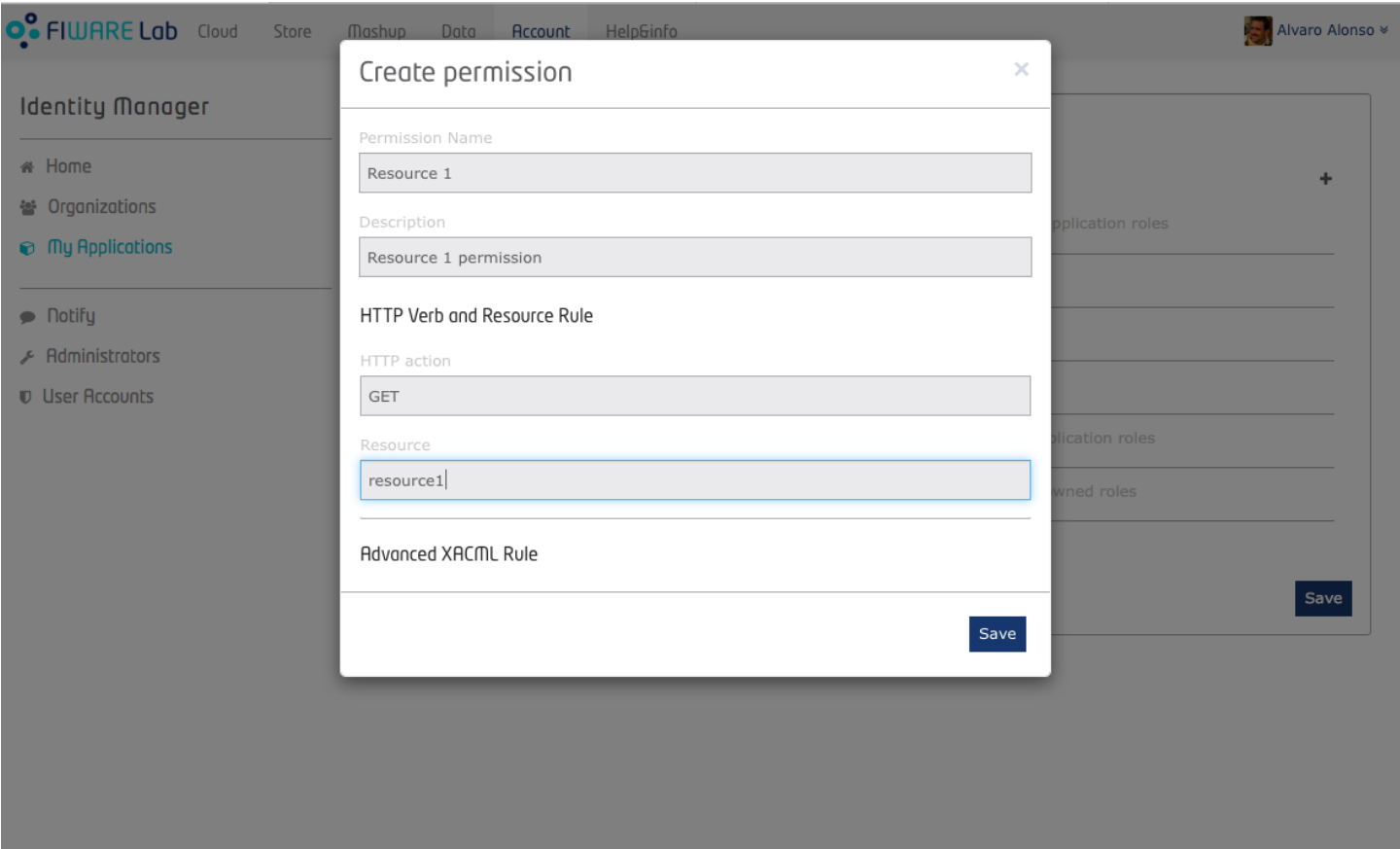
FIWARE

Level 2: Basic Authorization

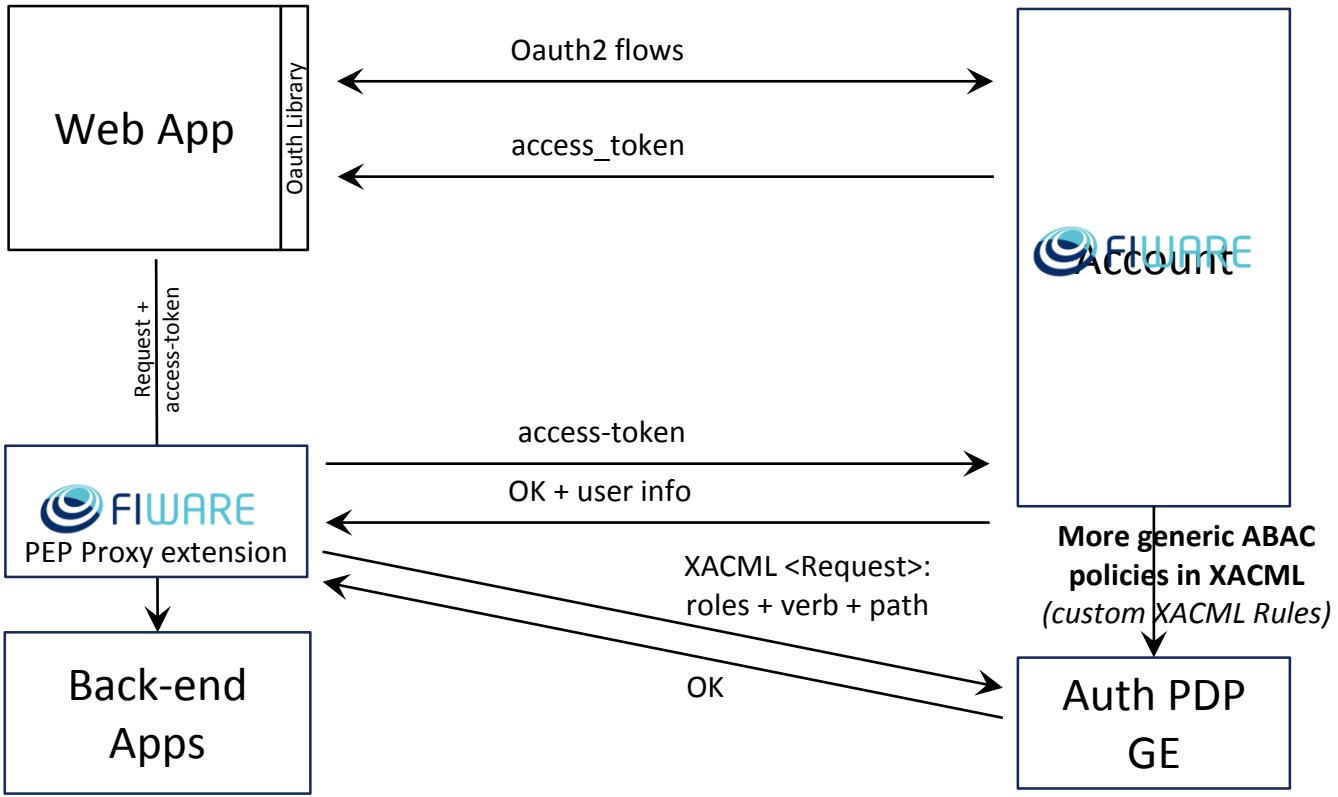


FIWARE

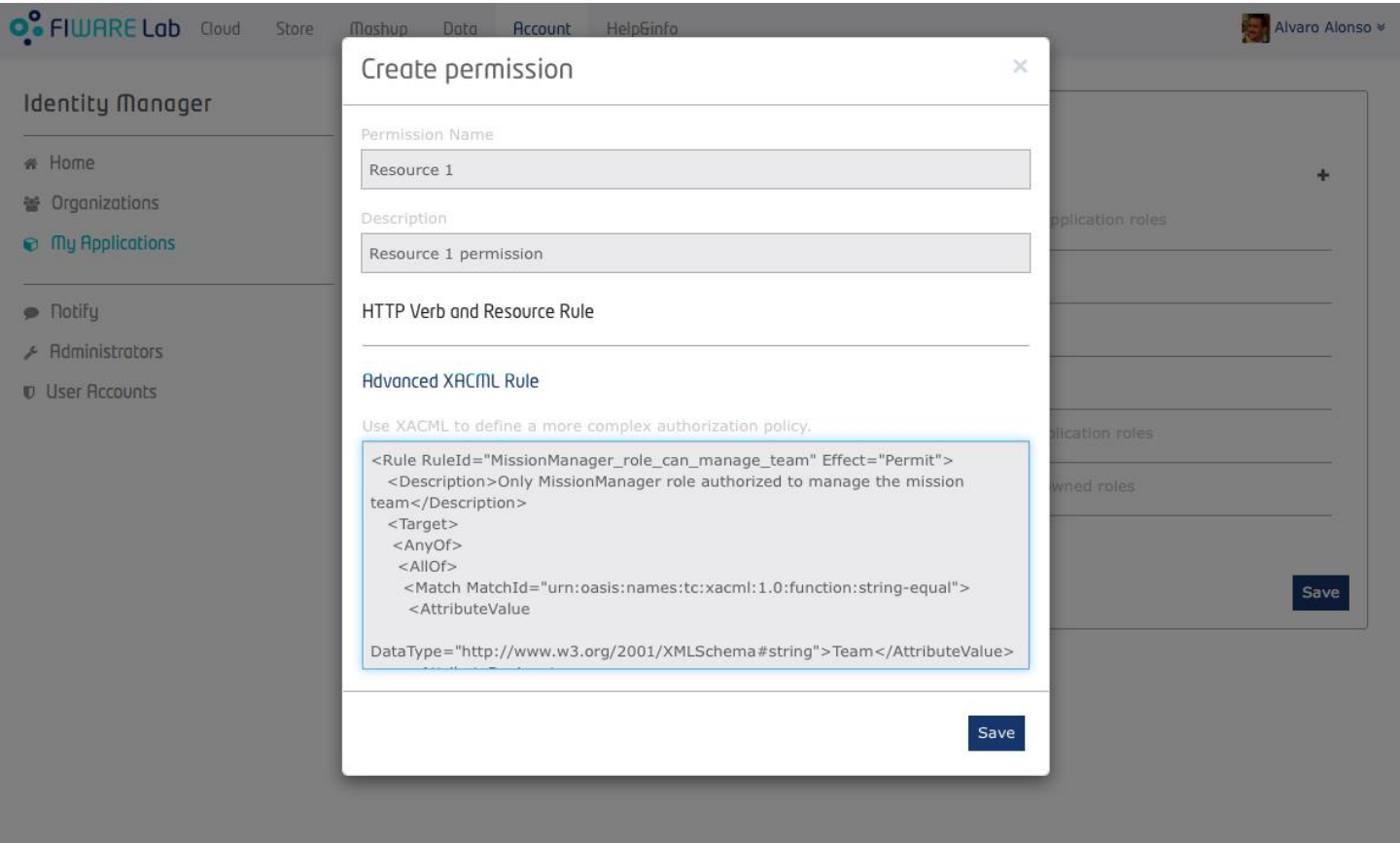
Level 2: Basic Authorization



Level 3: Advanced Authorization



Level 3: Advanced Authorization

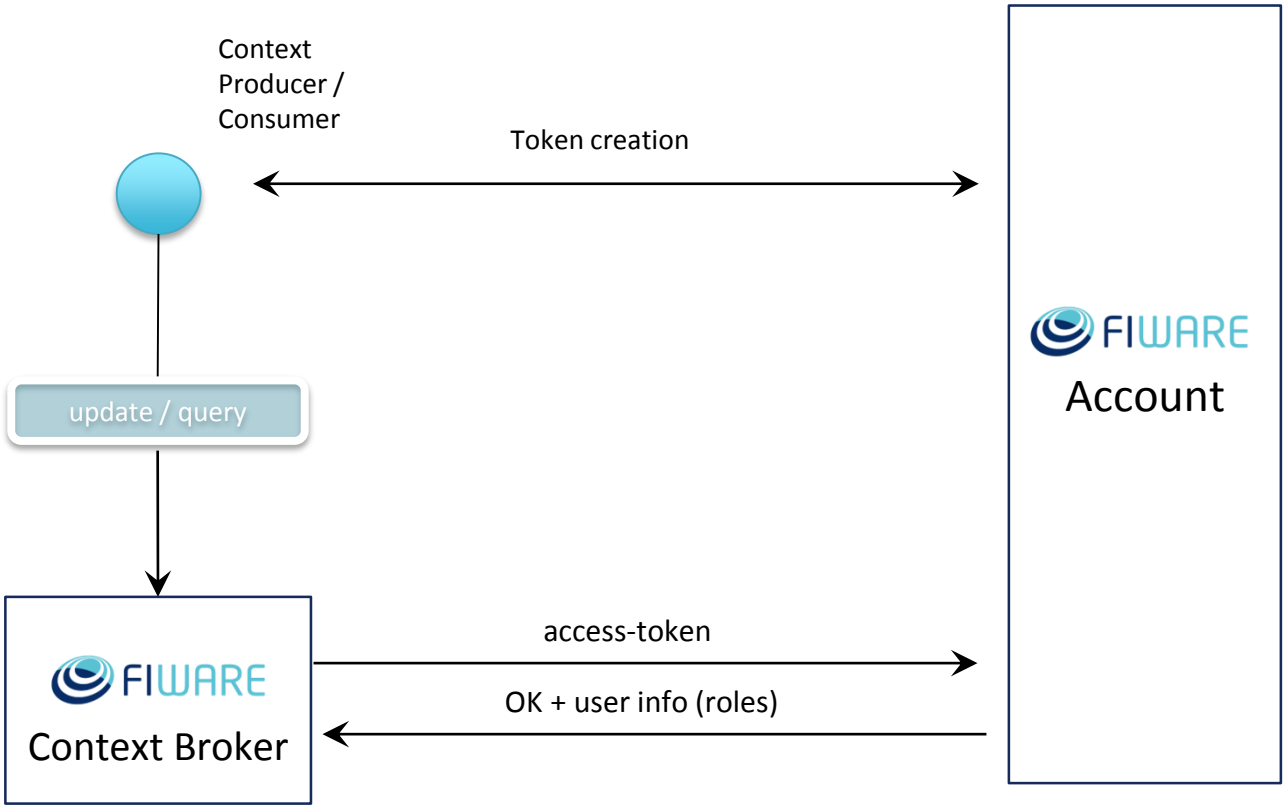


IoT Authentication

- Context Broker
  - IoT Management
  - Publish / subscribe model
    - Context producers
    - Context consumers
- Sensors Authentication
  - Sensor registration in IdM applications
  - Each sensor has its own account
    - Token creation and validation

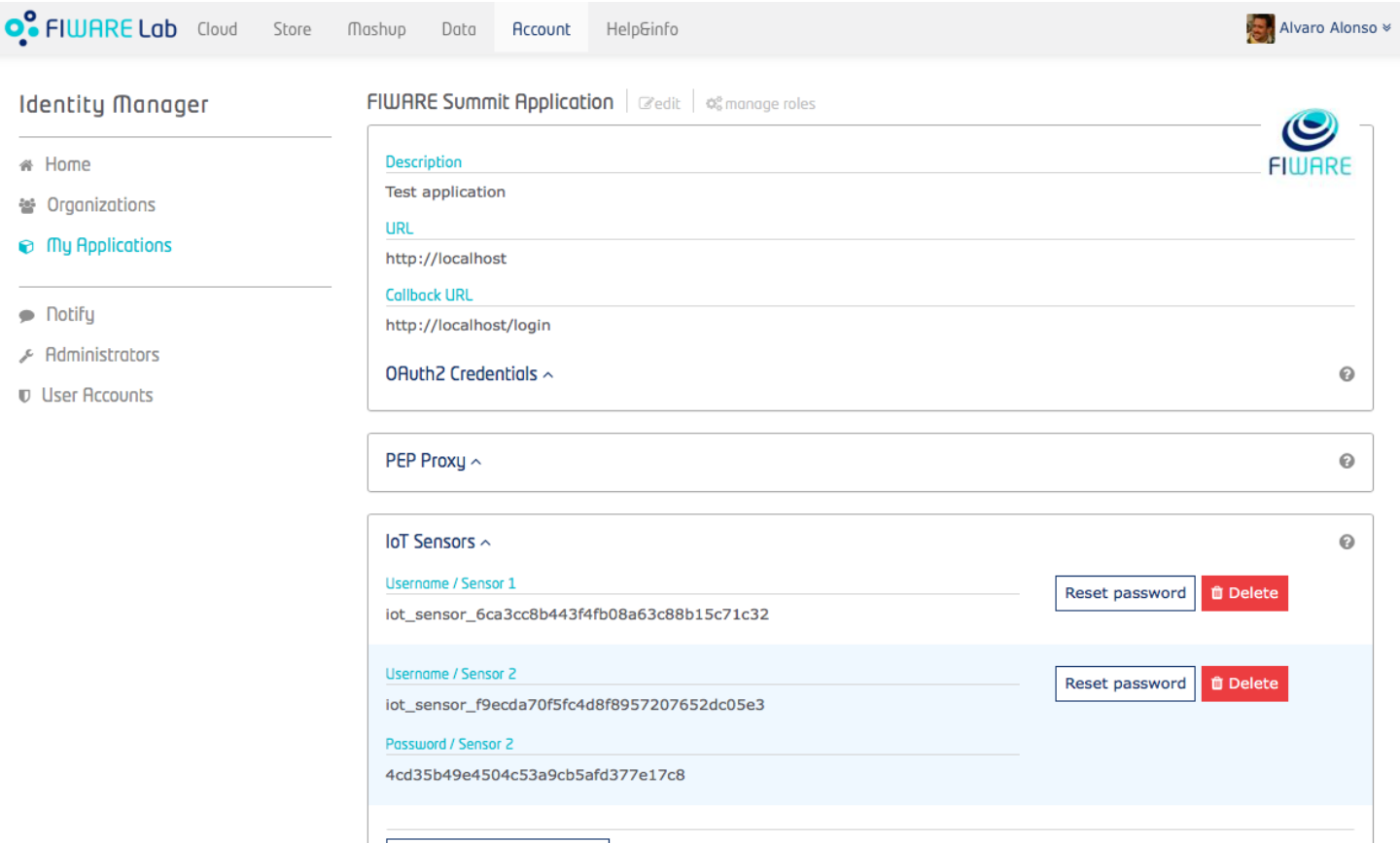
FIWARE

IoT Authentication



FIWARE

IoT Sensors in FIWARE Account



Security GEs

- Identity Management – Keyrock
- Authorization PDP – AuthZForce
- PEP Proxy - Wilma

Security GEs – IdM - KeyRock

- Keystone + Horizon +Extensions
- APIs
  - OAuth2
  - Keystone v3
  - SCIM 2.0
- Source Code
  - https://github.com/ging/fi-ware-idm
- Documentation
  - http://catalogue.fiware.org/enablers/identity-management-keyrock
- FIWARE OAuth2 Demo:
  - https://github.com/ging/oauth2-example-client

FIWARE

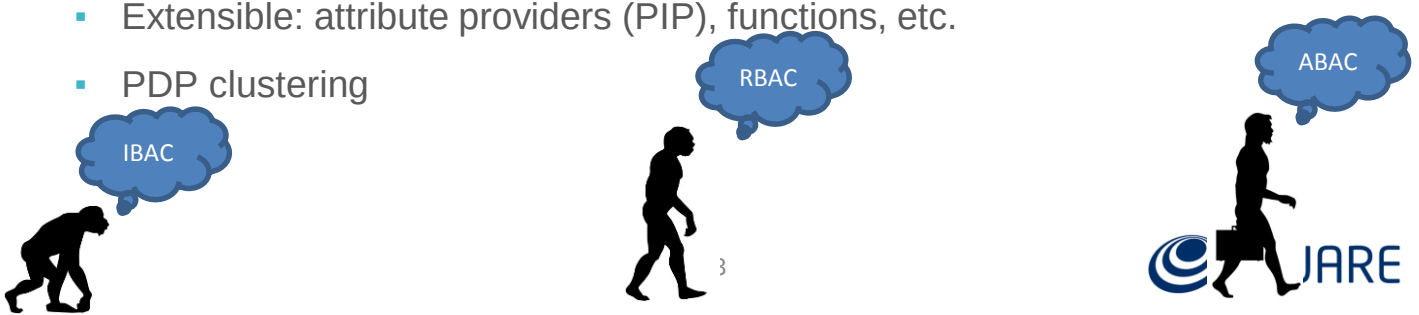
Security GEs – PEP Proxy - Wilma

- Policy Enforcement Point
- Compatible with OAuth2 and Keystone tokens
- Source code:
  - https://github.com/ging/fi-ware-pep-proxy
- Documentation
  - http://catalogue.fiware.org/enablers/pep-proxy-wilma
- Global instance

Security GEs – Authorization PDP – AuthZForce (1/2)

*By **2020**, the majority of enterprises will use ABAC as the dominant mechanism to protect critical assets, up from less than five percent today.* (Gartner, 2013)

- Single Open Spec (**Authorization PDP GE**) & Open Source implementation (GEri **Authzforce**) of 100% **XACML-3.0** standard-compliant and **cloud-ready RESTful ABAC** framework with **XML optimization**
- Multi-tenant REST API for PDP(s)/PAP(s)
- Standards:
  - OASIS: **XACML 3.0 + Profiles** (REST, RBAC, Multiple Decision)
  - ISO: **Fast Infoset**
- Extensible: attribute providers (PIP), functions, etc.
- PDP clustering



Security GEs – Authorization PDP – AuthZForce (2/2)

- FIWARE catalogue: https://catalogue.fiware.org/enablers/authorization-pdp-authzforce
- FIWARE Lab image: authzforce-5.4.1
- Authorization PDP GE's APIary: http://docs.authorizationpdp.apiary.io/#
- AuthzForce (GEri) source code:
  - API spec in WADL: https://github.com/authzforce/rest-api-model
  - Implementation: https://github.com/authzforce/server/
- AuthzForce distribution
  - Ubuntu/Debian-like: .deb / others: .tar.gz on Maven Central: http://central.maven.org/maven2/org/ow2/authzforce/authzforce-ce-server-dist/
  - Docker: https://hub.docker.com/r/fiware/authzforce-ce-server/
- Global instance for testing: https://az.lab.fiware.org/authzforce-ce/
- Documentation: http://catalogue.fi-ware.org/enablers/access-control-tha-implementation/documentation

Thank you!

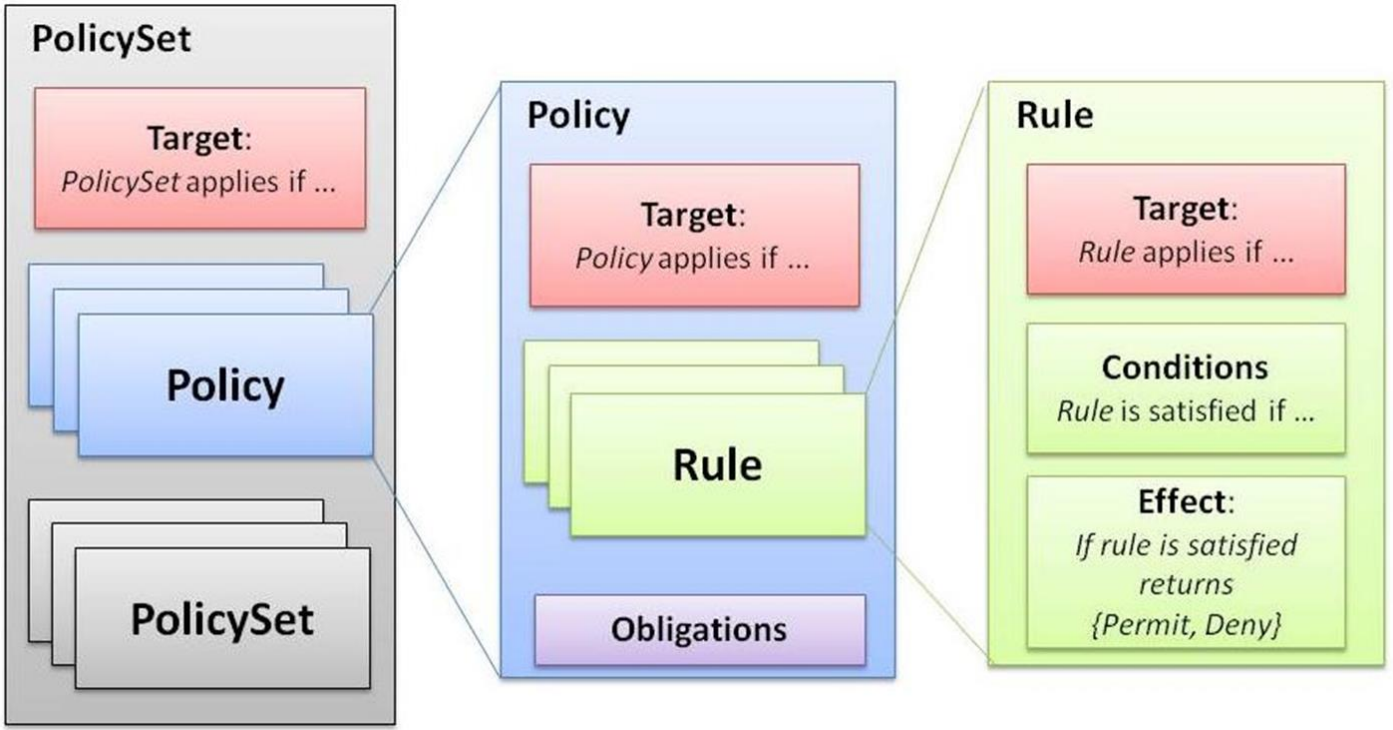
http://fiware.org
Follow @FIWARE on Twitter

RBAC vs. ABAC

- Role explosion example:
  - Roles in a bank: *Teller*, *Supervisor*, *Branch director*
  - Many bank agencies: *Paris*, *London*, *Berlin*
  - What about *Teller in Paris*, *Teller in London, Teller in Berlin, Supervisor in Paris*, *Supervisor in London*…? → **9 roles**!
- RBAC ☹ / ABAC ☺: Doctor-patient and patient-record relationships
  - Doctor may only access medical records of his/her own patients
    If *resource.type* = 'MEDICAL_RECORD'
    AND *action.id* in {'read','write'}
    AND ***user.id = medical_record.doctor_id***, then Permit
  - A patient may only access medical records about him/herself
    If *resource.type* = 'MEDICAL_RECORD'
    AND *action.id* ='read'
    AND ***user.id = medical_record.patient_id***, then Permit
- RBAC ☹ / ABAC ☺: Dynamic separation of duties
  - User may approve purchase order only if not assigned to him/herself (approver ≠ assignee)
  - ABAC-style (deny unless permit):
    If *resource.type* = 'PURCHASE_ORDER'
    AND *action.id* = 'approve'
    AND ***user.id ≠ purchase_order.assignee,*** then Permit

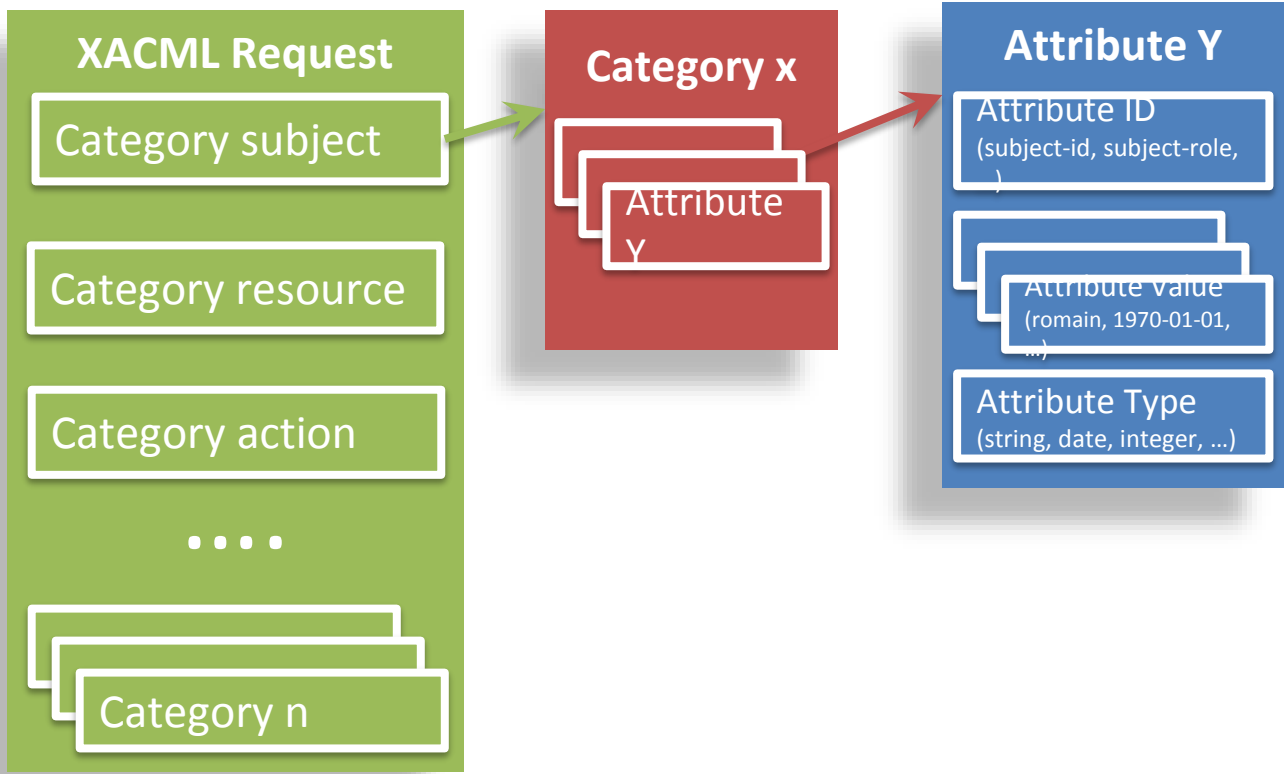
FIWARE

XACML Policy Language



FIWARE

XACML Request



FIWARE

XACML v3.0 vs. v2.0: upgrade!

- **More advanced and flexible Target** matching capabilities
- **Custom attribute categories** (limited to Resource, Action, Environment and a few Subject categories in v2.0)
- **Dynamic Obligations** using variables evaluated at runtime (limited to static values in v2.0) from request context after possible transformations by XACML functions
- **Obligations in Rules** (limited to Policies and PolicySets in v2.0)

FIWARE

FIWARE ABAC Architecture