사물인터넷 보안 인터워킹에 관한 연구

오세라*, 김영갑*,†
*세종대학교 정보보호학과 보안공학연구실
e-mail: terious551@sju.ac.kr, alwaysgabi@sejong.ac.kr

A Study on Secure Interworking in Internet of Things

Se-Ra Oh*, Young-Gab Kim*,†
*Security Engineering Lab., Dept. of Computer and Information Security, Sejong Univ.

요 약

사물인터넷(Internet of Things; IoT)이 발달하면서 FIWARE, oneM2M, AllJoyn 등의 많은 IoT 플랫폼이 개발되고 관련 표준들도 제정되고 있다. 이런 환경에서는 각 IoT 플랫폼의 통신 프로토콜, 보안 정책 등이 상이하기 때문에 데이터가 연동되기 어렵고, 보안 인터워킹을 수행하는 것은 더욱 어려운 문제다. oneM2M과 FIWARE의 경우, Fi-Proxy 인터워킹 어댑터를 이용하여 상호간 인터워킹이 시연된 사례가 있지만 그 과정에서 보안이 고려되지는 않았다. oneM2M은 아직 보안 컴포넌트가 별도로 존재하지 않기 때문에, 본 논문에서는 oneM2M-FIWARE 보안 인터워킹 시나리오에 적용할 수 있는 OAuth 2.0 기반 oneM2M 보안 컴포넌트를 구현하고, FIWARE 보안 아키텍처를 분석 및 구현하여 시사점을 도출한다. 또한 본 논문은 oneM2M과 FIWARE 간의 보안 인터워킹 아키텍처를 제안하고, 이를 기반으로 향후 여러 도메인에 활용될 수 있는 구조를 가진 LED 예제를 개발한다.

1. 서론

IoT(Internet of Things)가 각광 받으면서 현재 FIWARE, oneM2M, AllJoyn, ARTIK, IoTivity와 같은 IoT 플랫폼들이 다수 만들어졌지만 각각의 플랫폼들은 기반 기술, 통신 프로토콜, 보안 정책 등이 모두 다르기 때문에 서로 원하는 정보가 있어도 쉽게 교환할 수가 없다. 과거에 정보의 공유를 위해서 네트워크를 만들고, 네트워크를 엮어서 인터넷을 만든 것처럼, IoT 플랫폼들 또한 유용한 정보를 서로 공유하기 위해서 인터워킹 작업이 필요하다. 대다수의 IoT 플랫폼이 인터워킹 연구에 적극적인 것은 아니지만 oneM2M 측은 이미 다른 IoT 플랫폼들과 인터워킹 연구를 수행하였고, 상당한 성과를 보여주고 있다[1]. 그러나 현재 진행 중인 이기종 IoT 플랫폼 간의 인터워킹 작업은 초기 단계로, 가장 중요한 데이터 교환부터 진행 중이기 때문에 인터워킹 연구에 적극적인 oneM2M의 경우에도 아직까지 보안에 관한 논의가 부족하다. 이러한 상황은 미래에 수집될 다양한 개인 정보의 유출 같은 크고 작은 문제들을 야기할 수 있다. 보안은 개발 생명주기의 처음부터 마지막까지 논의가 되어야 하는 중요한 부분이므로 지금이 보안 요구사항 분석과 보안 컴포넌트의 개발 및 테스트 등을 수행할 시점이다.

본 논문에서는 IoT를 포함한 미래의 인터넷 어플리케이션 개발을 위해 EU가 주도로 제정한 표준인 FIWARE(Future Internet ware)와 ETSI(European Telecommunications Standards Institute), TTA(Telecommunications Technology Association) 등의 단체들이 협력하여 제정하고 있는 표준인 oneM2M의 인터워킹 연구 현황과 보안 컴포넌트를 분석하고 이를 적용한 보안 인터워킹 아키텍처를 제안한다. oneM2M의 보안 컴포넌트는 아직 개발되지 않았기 때문에, 본 연구에서 인증과 인가를 제공하는 OAuth 2.0 기반 oneM2M 보안 컴포넌트를 구현한다. 그리고 본 논문에서 제안하는 보안 인터워킹 아키텍처를 기반으로, 향후 다양한 도메인에 적용될 수 있는 LED 예제를 개발한다. 본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 살펴보고 3장에서 oneM2M과 FIWARE의 보안 컴포넌트에 대해서 기술한다. 4장에서는 보안을 고려한 oneM2M과 FIWARE 간의 인터워킹 아키텍처를 제안하고 oneM2M 보안 컴포넌트를 활용한 LED 예제를 소개한다. 마지막 5장에서는 본 연구의 결론을 서술한다.

2. 관련 연구

IoT 인터워킹과 관련된 연구로 김재호 외[2]는 OIC(Open Interconnect Consortium), AllJoyn,

† 교신저자

이 연구는 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임 (No.2017-0-00756, IoT 이종 식별체계 상호연동 및 관리체계 기술 개발)

LWM2M(Lightweight M2M)의 대표적인 IoT 표준기술들을 간단히 기술하고 oneM2M과의 연동 기술과 실제 구현된 사례를 소개하였다. 또한 이동규 외[3]는 CoAP(Constrained Application Protocol), MQTT(Message Queuing Telemetry Transport), HTTP(Hyper Text Transfer Protocol) 같은 통신 프로토콜을 DDS(Data Distribution Service)로 바인딩하여 이기종의 IoT 플랫폼에서 인터워킹을 수행할 수 있는 방안을 제안했으며, 윤재석 외[4]는 IPE(Interworking Proxy Entity)를 통해 oneM2M 기반의 IoT 시스템과 기존의 IoT 디바이스가 인터워킹할 수 있는 방법을 연구했다.

3. oneM2M과 FIWARE의 보안 컴포넌트

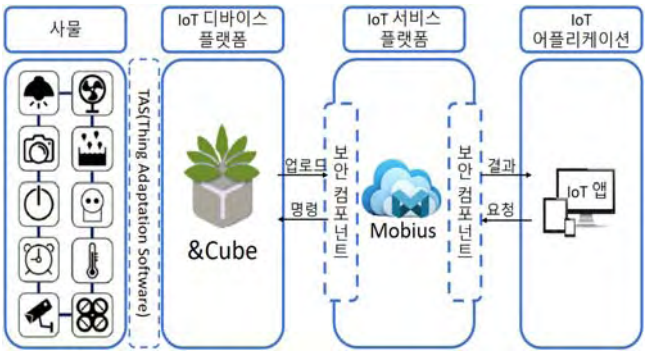
oneM2M의 경우, 구현된 보안 컴포넌트는 아직 없고 보안 아키텍처가 계속 논의되고 있다. 따라서 본 연구에서는 논의 중인 oneM2M의 보안 아키텍처를 분석하거나 구현하는 것 대신, 보안 인터워킹의 핵심인 인증과 인가를 제공하는 OAuth 2.0 기반 보안 컴포넌트를 개발하고 Mobius에 적용하였다. OAuth 2.0 서버 구현을 위해 Node.js 기반의 'oauth2-server' 모듈을 사용하였으며, 토큰의 발급을 위해 필요한 클라이언트, 회원 정보들은 임시로 만든 웹 페이지를 통해 임의로 생성하였다. 그림 1은 'Resource Owner Password Credentials' 방식을 이용하여 oneM2M 보안 컴포넌트에 토큰 발급 요청을 보낸 결과이다.


```
{
    "token_type": "bearer",
    "access_token": "81d7004b73d87bc9a034386daff0c61569d0d2ed",
    "expires_in": 3600
}
```


(그림 1) oneM2M 보안 컴포넌트의 토큰 발급 결과

토큰을 발급받기 위해서는 Mobius의 토큰 엔드포인트인 'localhost:7579/oauth/token'에 필요한 정보(grant_type, 클라이언트ID와 Secret, 자원 소유자의 ID와 비밀번호)를 POST 방식으로 요청해야 한다. 토큰 발급이 성공하면 그림 1처럼 토큰 타입과 토큰, 만료 시간이 전송된다. 토큰 타입과 만료 시간은 각각 bearer와 1시간이며 토큰은 랜덤한 문자열이다.

Mobius는 디바이스의 등록이나 관리, 자원의 CRUD(Create, Read, Update, Delete) 동작 등을 수행하기 위해 REST API를 사용한다. 일부 보안적으로 유해하지 않은 REST API도 있지만, 시스템을 제어하거나, 개인 데이터나 보안적으로 중요한 데이터들을 요청, 조작하는 REST API들은 심각한 피해를 야기할 수도 있다. 그러므로 Mobius의 디바이스나 데이터 같은 자원들을 보호하려면, REST API의 사용을 제한할 필요가 있다. 현재 oneM2M 보안 컴포넌트는 Mobius(http://localhost:7579/*)에 접근하는 모든 요청을 막고, 정당한 토큰이 있어야만 REST API를 사용할 수 있도록 설정해놓았다. 그림 2는 oneM2M 보안 컴포넌트를 보여준다.



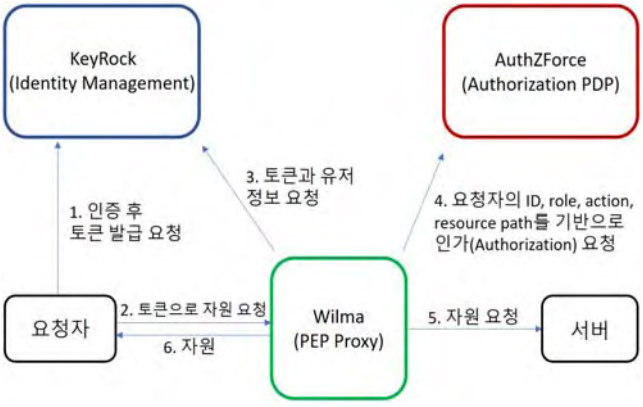
(그림 2) oneM2M 보안 컴포넌트

보안 컴포넌트는 IoT 어플리케이션이나 디바이스 플랫폼에서 REST API를 이용하여 Mobius에 자원을 요청하면, 일단 요청을 막고 토큰의 검증이 끝난 후 처리한다. 만약 보안 컴포넌트가 동작하는 상태에서 토큰 없이 Mobius의 자원(데이터, REST API 등)에 접근하거나 비정상 토큰을 사용하면 그림 3같은 결과가 반환되고, 문제가 없다면 정상적으로 요청이 처리된다.


```
{
    "code": 400,
    "error": "invalid_request",
    "error_description": "The access token was not found"
}
{
    "code": 401,
    "error": "invalid_token",
    "error_description": "The access token provided is invalid."
}
```


(그림 3) Mobius 접근 실패 결과

oneM2M과 달리, FIWARE는 이미 구체적인 보안 아키텍처가 존재하고 보안 컴포넌트들을 구현했다. FIWARE의 보안 아키텍처[5]는 KeyRock, AuthZForce, Wilma라고 불리는 세 가지 GE(Generic Enabler)로 이루어져 있으며, oneM2M 보안 컴포넌트와 마찬가지로 OAuth 2.0을 지원한다. FIWARE 보안 아키텍처의 동작을 간단히 도식화하면 그림 4와 같다.



(그림 4) FIWARE 보안 아키텍처 동작 개요

가장 먼저 요청자(Requester)는 KeyRock에 인증 정보를 전송하여 인증을 받은 뒤, 액세스 토큰을 발급 받고 토큰으로 Wilma에게 원하는 자원을 요청한다. Wilma는 해당 토큰과 요청자에 관한 정보를 KeyRock에게 전달 받고

AuthZForce에 전달하여 요청자의 요청이 적절한지 검증한다. 마지막으로 AuthZForce의 응답이 Permit인 경우, Wilma가 요청자 대신 서버에 자원을 요청하고 결과를 요청자에게 전달하게 된다.

서버에 접근하는 모든 요청은 Wilma를 거치게 되는데, 여기서 Wilma의 보안은 크게 세 가지 레벨로 나뉜다[6]. 1레벨(Authentication) 보안은 토큰의 검증만 수행하여 가장 간단하지만 취약하며, 유일하게 AuthZForce가 사용되지 않는다. 2레벨(Basic Authorization)부터는 AuthZForce를 통해 요청에 대해서 인가하는 작업이 추가되고 3레벨(Advanced Authorization)은 XACML Request를 이용하여 정교한 인가 요청을 보낼 수 있다.

최신 버전의 FIWARE 보안 GE들은 테스트가 충분하지 않을 수 있기 때문에, 보안 아키텍처를 구현한다면 KeyRock 5.4.1, AuthZForce 5.4.1, Wilma 5.4.0을 이용하는 것이 좋다. 본 연구에서는 AuthZForce 5.4.1이 Ubuntu 16.04를 지원하지 않으므로 7.0.0 버전을 사용하였다. 그림 5는 보안 GE를 모두 실행한 뒤, KeyRock에게 보낸 토큰 발급 요청과 이후 결과를 보여준다.

```
weepi@ubuntu:~/fiware/fiware-pep-proxy$ curl -X POST -u b16e8edcb2c046a
daab5a9c5022e728c:37a1204628724093b62eab866ee00a04 -H 'Content-Type: ap
plication/x-www-form-urlencoded' -d 'grant_type=password&username=user0
@test.com&password=test' http://localhost:8000/oauth2/token | python -m
json.tool
  % Total    % Received % Xferd  Average Speed   Time    Time     Time
 Current
                                 Dload  Upload   Total   Spent    Left
 Speed
100   202    0   145  100    57   2257    887 --:--:-- --:--:-- --:--:-
-  2265
{
    "access_token": "TBmt3X42kTg6meQ84WBF9EX576COcY",
    "expires_in": 3600,
    "refresh_token": "RFibMetRdJnayB4vhlBILVA2fZpvKg",
    "token_type": "Bearer"
}
```

(그림 5) 토큰 발급 요청과 결과

토큰 발급 요청은 KeyRock의 토큰 엔드포인트(http://localhost:8000/oauth2/token)에 POST 방식으로 보낸다. 이때 필요한 정보는 그림 5처럼 클라이언트 ID(b16e8…e728c)와 Secret(37a12…00a04), Content-Type(application/x-www-form-urlencoded), grant_type(password), username(user0@test.com), password(test)이며, 이는 사용하는 인가 방식(grant type)에 따라 차이가 있다.

그림 5에서 볼 수 있듯이 토큰 발급 요청에 필요한 정보들이 모두 갖춰지면 토큰과 관련 정보들이 KeyRock으로부터 전송된다. FIWARE의 토큰은 일반적인 'Bearer' 방식이며 만료 시간은 기본 한 시간으로 설정되어 있다. 앞서 설명하였듯이, 발급받은 토큰은 서버의 보호되고 있는 자원을 요청하는데 사용된다. 그림 6은 FIWARE에서 토큰을 사용하여 'My_Room' 정보에 접근한 결과이다.

```
weepi@ubuntu:~/fiware/fiware-pep-proxy$ curl -H "X-Auth-Token: TBmt3X42
kTg6meQ84WBF9EX576COcY" http://localhost:1706/v2/entities/My_Room
{"id":"My_Room","type":"Room","pressure":{"type":"Number","value":720,"
metadata":{}},"temperature":{"type":"Number","value":23,"metadata":{}}}
```

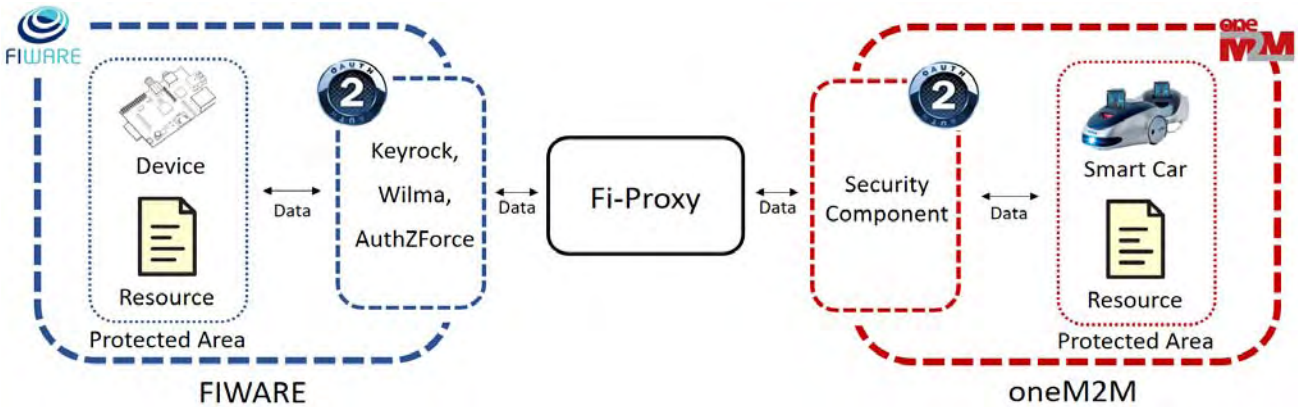
(그림 6) 보호된 자원 요청과 결과

토큰은 헤더의 'X-Auth-Token'에 첨부하고, Wilma(localhost:1706)에 원하는 자원의 URL(v2/entities/My_Room)을 보낸다. Wilma의 설정 파일(config.js)에는 토큰의 검증을 성공적으로 마치면 요청을 Redirect할 주소를 입력하도록 되어 있는데, 본 연구의 Wilma는 Context 정보를 관리해주는 Orion Context Broker의 주소(localhost:1026)로 설정하였다. 따라서 Wilma는 토큰을 성공적으로 검증하면 Orion Context Broker에 요청을 Redirect하여 'My_Room'의 정보를 검색하고, 결과는 요청자에게 다시 보내준다.

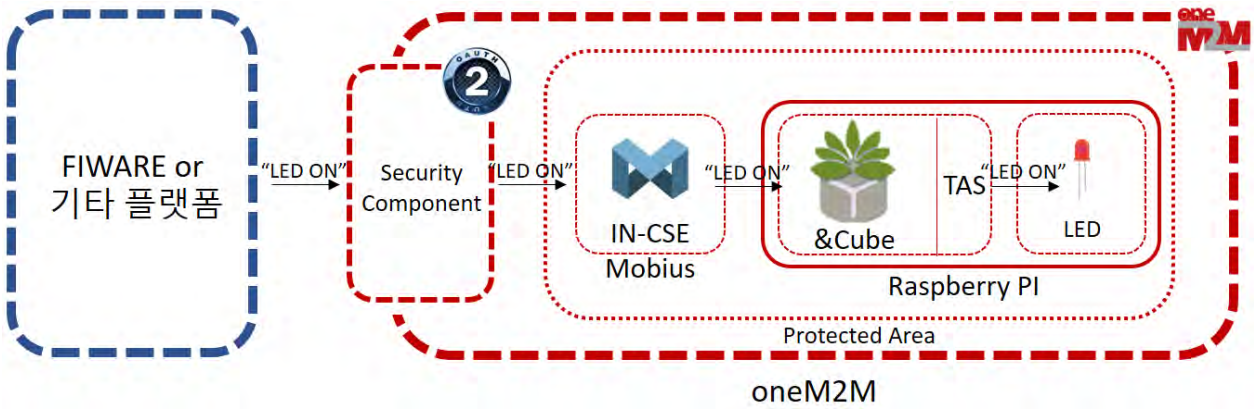
FIWARE는 이미 몇 차례 보안 관련 GE를 만들었고, KeyRock, Wilma, AuthZForce는 아직까지 폐기되지 않고 업데이트와 테스트가 진행되고 있는 GE들이다. 그러나 본 연구에서 실제 보안 GE들을 설치하고 테스트해본 결과, 아직까지 GE들의 설정 과정이 까다롭고 특정 버전끼리의 의존성이 강한 문제가 있음을 확인하였다.

4. oneM2M-FIWARE 보안 인터워킹 구조

FIWARE 측에서 인터워킹을 위해 타 플랫폼과 협력한 것은 oneM2M이 대표적이다. 2015년 NEC, KETI, 세종대학교 등은 FIESTA (Federated Interoperable Semantic IoT/cloud Testbeds and Applications)의 일환인 oneM2M 쇼 케이스에서 Fi-Proxy 인터워킹 어댑터를 이용하여 Mobius와 FIWARE를 연동하고 스마트 시티 시연을 보였다. 그러나 인터워킹 시나리오에서 보안이 고려되지 않았



(그림 7) FIWARE-oneM2M 보안 인터워킹 구조



(그림 8) LED 예제 구조

다. 만약 기존의 인터워킹에 oneM2M과 FIWARE의 보안 컴포넌트가 적용된다면 그림 7같은 구조가 될 것이다. 그림 7의 구조는 Fi-Proxy를 거쳐 데이터가 교환되는 부분은 변함이 없으나 FIWARE나 oneM2M의 자원에 접근하려면 보안 컴포넌트들을 거쳐서 인증과 인가를 받아야 한다. oneM2M 측에서 FIWARE의 자원을 요청하려면 3장에서 설명한 것처럼, 먼저 KeyRock에서 인증 후 토큰을 발급 받은 뒤, Wilma에 자원을 요청하고 KeyRock과 AuthZForce를 통해서 토큰의 검증과 인가를 받아야 한다. 이와 같이, FIWARE 측에서도 oneM2M의 자원을 제어하기 위해서는 보안 컴포넌트에서 인증과 인가를 받을 필요가 있다.

마지막으로 소개할 본 연구의 LED 예제는 OCEAN(Open allianCE for iot stANdard)의 LED 예제[7]와 그림 7의 구조를 기반으로 개발되었으며, 그림 8의 구조는 LED만이 아니라 다른 사물이나 디바이스 플랫폼을 대상으로 적용될 수 있는 장점이 있다. 그림 8처럼 FIWARE나 기타 플랫폼이 oneM2M의 LED를 제어하기 위해서는 Mobius에 요청을 보내야 하는데, 본 연구에서 제안한 구조에서는 반드시 그보다 먼저 Mobius의 보안 컴포넌트에서 REST API에 접근하기 위한 토큰을 발급받아야 한다.

발급받은 토큰은 Mobius에 LED를 켜기 위한 요청을 보낼 때 헤더의 'Authorization'에 'Bearer 토큰' 형태로 조합하여 첨부되어 oneM2M 보안 컴포넌트에 의해 검증된다. 토큰이 정상적이라면 보안 컴포넌트는 실행 흐름을 Mobius에게 돌려주고, 이후에는 &Cube(Thyme), TAS(Thing Adaptation Software)를 통해 LED 제어 명령이 전달되어 LED가 동작한다. 만약 토큰이 제대로 검증이 안 된다면, LED 제어 명령은 보안 컴포넌트에서 차단된다. 여기서 핵심은 보안 컴포넌트에 의해 수행되는 토큰의 발급, 토큰의 검증 여부와 그림 8 구조의 확장성이다. 해당 구조를 통해서 토큰이 없는 비인가자는 Mobius의 자원을 제어할 수 없게 되고, LED외에도 다양한 사물을 그림 8에 적용할 수 있다.

5. 결론

본 연구에서는 oneM2M(Mobius)에서 사용할 수 있는 OAuth 2.0 기반의 보안 컴포넌트를 구현하고 테스트하였다. 또한 FIWARE의 보안 아키텍처를 구성하는 보안 GE(KeyRock, AuthZForce, Wilma)를 분석했고, 실제 구현을 통해 아직까지 보안 GE들의 설정 과정이 간단하지 않고 버전 간의 호환성 문제가 존재하는 것을 발견하였다. 마지막으로, 보안이 고려되지 않은 oneM2M-FIWARE 인터워킹 구조에 oneM2M과 FIWARE의 보안 컴포넌트를 적용한 보안 인터워킹 구조를 제안하였으며 이를 기반으로 향후 여러 도메인에 적용할 수 있는 LED 예제를 개발하였다. LED 예제의 테스트를 통해서 oneM2M 보안 컴포넌트는 Mobius의 자원(LED)에 접근하는 요청들을 제한하는데 효과가 있음을 확인하였다.

참고문헌

[1] oneM2M "Latest Draft Specifications" http://www.onem2m.org/technical/latest-drafts

[2] 김재호, 최성찬, 성낙명, 윤재석 "사물인터넷 표준 인터워킹 기술" 한국통신학회지 33(5) pp.55-64 2016

[3] 이동규, 김동희, 정태명 "사물인터넷 플랫폼에 DDS 인터워킹 지원 방안 제안" 한국통신학회 2016(6) pp.385-386 2016

[4] Jaeseok Yun, Ramnath Chekka Teja, Nan Chen, Nak-Myoung Sung, Jaeho Kim "Interworking of oneM2M-based IoT Systems and Legacy Systems for Consumer Products" Information and Communication Technology Convergence (ICTC) pp.423-428 2016

[5] FIWARE "Security Architecture" https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Security_Architecture

[6] FIWARE "PEP Proxy User Guide" http://fiware-pep-proxy.readthedocs.io/en/latest/user_guide/

[7] OCEAN(Open alliance for iot standard) "LED Sample" http://www.iotocean.org/download/?tab1=2