



(19) 대한민국특허청(KR)
(12) 공개특허공보(A)

(11) 공개번호 10-2019-0102432
(43) 공개일자 2019년09월04일

(51) 국제특허분류(Int. Cl.)
H04L 29/06 (2006.01) H04L 29/08 (2006.01)
(52) CPC특허분류
H04L 63/0807 (2013.01)
H04L 63/0869 (2013.01)
(21) 출원번호 10-2018-0022797
(22) 출원일자 2018년02월26일
심사청구일자 2018년02월26일

(71) 출원인
세종대학교산학협력단
서울특별시 광진구 능동로 209 (군자동, 세종대학교)
(72) 발명자
김영갑
서울특별시 광진구 능동로 209, 세종대학교 행복
기숙사 220호(군자동)
오세라
서울특별시 성북구 장위로16길 64, B01호(장위동,
청구빌라)
(74) 대리인
양성보

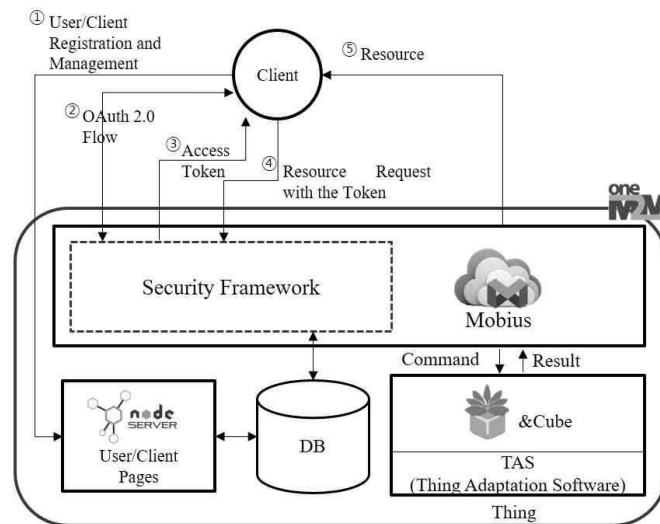
전체 청구항 수 : 총 12 항

(54) 발명의 명칭 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크 및 그 장치

(57) 요약

이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크 및 그 장치가 개시된다. 본 발명의 일 실시예에 따른 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크는 클라이언트 정보가 등록된 이중 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 보안 컴포넌트를 이용하여 상기 클라이언트의 인증을 수행하는 단계; 상기 클라이언트가 인증되면 상기 보안 컴포넌트에서 상기 클라이언트로 토큰을 발급하는 단계; 및 상기 클라이언트에서 상기 발급받은 토큰을 통해 자원이 요청되면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공하는 단계를 포함한다.

대표도 - 도14



(52) CPC특허분류

H04L 67/16 (2013.01)

H04L 67/28 (2013.01)

이 발명을 지원한 국가연구개발사업

과제고유번호 1711055293

부처명 과학기술정보통신부

연구관리전문기관 정보통신기술진흥센터

연구사업명 사물인터넷융합기술개발

연구과제명 IoT 이종 식별체계 상호연동 및 관리체계 기술 개발

기 여 율 1/1

주관기관 부산대학교산학협력단

연구기간 2017.04.01 ~ 2018.12.31

명세서

청구범위

청구항 1

클라이언트 정보가 등록된 이중 사물 인터넷(IoT) 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 보안 컴포넌트를 이용하여 상기 클라이언트의 인증을 수행하는 단계;

상기 클라이언트가 인증되면 상기 보안 컴포넌트에서 상기 클라이언트로 토큰을 발급하는 단계; 및

상기 클라이언트에서 상기 발급받은 토큰을 통해 자원이 요청되면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공하는 단계

를 포함하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크.

청구항 2

제1항에 있어서,

상기 토큰을 발급하는 단계는

OAuth 표준을 통해 상기 토큰을 발급하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크.

청구항 3

제1항에 있어서,

상기 인증을 수행하는 단계는

상기 클라이언트에서 Resource Owner Password Credentials 방식을 이용하여 상기 토큰 발급을 요청하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크.

청구항 4

제1항에 있어서,

상기 요청된 자원을 상기 클라이언트로 제공하는 단계는

상기 발급받은 토큰을 통해 자원이 요청되면 상기 자원을 요청한 토큰을 검증하고, 상기 토큰의 검증이 성공하면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크.

청구항 5

제1항에 있어서,

상기 보안 컴포넌트는

oneM2M 보안 컴포넌트를 포함하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크.

청구항 6

제1항에 있어서,

상기 이중 클라이언트의 인증을 수행하는 단계는

상기 이중 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 미리 설정된 적어도 하나 이상의 REST API에 대해서만 상기 보안 컴포넌트를 이용하여 상기 이중 클라이언트의 인증을 수행하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크.

청구항 7

클라이언트 정보가 등록된 이중 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급 요청을 수신하는 수신부;

상기 토큰 발급을 요청한 상기 클라이언트의 인증을 수행하고, 상기 클라이언트가 인증되면 상기 클라이언트로 토큰을 발급하는 승인부; 및

상기 클라이언트에서 상기 발급받은 토큰을 통해 자원이 요청되면 상기 요청된 자원을 상기 클라이언트로 제공하도록 제어하는 제어부

를 포함하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치.

청구항 8

제7항에 있어서,

상기 승인부는

OAuth 표준을 통해 상기 토큰을 발급하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치.

청구항 9

제7항에 있어서,

상기 수신부는

상기 클라이언트에서 Resource Owner Password Credentials 방식을 이용하여 요청한 상기 토큰 발급 요청을 수신하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치.

청구항 10

제7항에 있어서,

상기 제어부는

상기 발급받은 토큰을 통해 자원이 요청되면 상기 자원을 요청한 토큰을 검증하고, 상기 토큰의 검증이 성공하면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공하는 것을 특징으로 하는 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치.

청구항 11

제7항에 있어서,

상기 보안 컴포넌트는

oneM2M 보안 컴포넌트를 포함하는 것을 특징으로 하는 이종 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치.

청구항 12

제7항에 있어서,

상기 승인부는

상기 이종 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 미리 설정된 적어도 하나 이상의 REST API에 대해서만 상기 보안 컴포넌트를 이용하여 상기 이종 클라이언트의 인증을 수행하는 것을 특징으로 하는 이종 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치.

발명의 설명

기술 분야

[0001] 본 발명은 이종 사물 인터넷(IoT) 서비스 플랫폼 간의 보안 상호운용성 프레임워크 기술에 관한 것으로서, 보다 상세하게는 이종의 IoT 플랫폼 간에 인터워킹(interworking)을 수행할 때 IoT 플랫폼 간에 보안 상호운용성을 확보할 수 있는 보안 상호운용성 프레임워크 및 그 장치에 관한 것이다.

배경 기술

[0003] IoT (Internet of Things)는 일반적으로 상황인지, 지능화, 자동화 기능을 가진 디바이스가 서비스 제공 등의 특정 목적을 위해 네트워크에 연결되는 협업 환경을 나타낸다. 가트너, 시스코, IDC 등의 기업들은 IoT를 유망한 미래의 기술로 꼽았으며, IDC는 IoT 시장 규모가 2015년부터 2020년까지 연평균 15.6%만큼 성장할 것으로 전망했다. 이러한 추세에 맞춰서 구글, 아마존, 삼성, 시스코 등의 세계적 기업들도 IoT 시장을 선점하기 위해서 IoT 관련 기술, 표준, 디바이스 등을 만들고 있는데, IoT에서는 IoT 디바이스가 필요로 하는 기본적인 기능들을 제공해주는 IoT 플랫폼이 중요하다. 현재 FIWARE, oneM2M, AllJoyn, ARTIK, IoTivity와 같은 IoT 플랫폼들이 다수 만들어졌지만 IoT 플랫폼의 기반 기술, 통신 프로토콜, 보안 정책 등이 모두 상이하여, 이종의 플랫폼 간에 원하는 정보나 서비스가 있어도 간단히 교환하거나 이용할 수가 없는 상황이다. 그러므로 과거에 정보의 공유를 위해서 네트워크를 만들고, 네트워크를 엮어서 인터넷을 만든 것처럼, IoT 플랫폼들 또한 유용한 정보나 서비스를 서로 공유하기 위한 인터워킹 연구가 반드시 필요하다. 대다수의 IoT 플랫폼이 인터워킹 연구에 적극적인 것은 아니지만 oneM2M에서는 현재 여러 IoT 플랫폼들과 인터워킹 연구를 수행하였고, 상당한 성과를 보여주고 있다. 그러나 지금 진행 중인 이종 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 상황에서는 IoT 플랫폼 간의 인터워킹이 수행되어도 IoT 플랫폼 사이에서 전달되는 개인 정보나 기업 정보들의 유출 등 크고 작은 보안 문제들을 야기할 수 있다.

[0004] 종래 IoT 보안과 관련된 일 실시예 기술은 세 가지 IoT 기본 특성(이종성, 자원 제약성, 동적 환경)에 기반한 기초 보안 요구사항과 여섯 가지 IoT 요소 (플랫폼, IoT 네트워크, 서비스, 공격자, 사용자, 클라우드)를 선정하여 IoT의 전체적인 보안 요구사항을 분석하였다. 종래 IoT 보안과 관련된 다른 일 실시예 기술은 ITU-T, ISO/IEC JTC1, ETSI, IETF 등의 표준 기구 별 IoT 보안 표준을 조사하여 여섯 가지 카테고리(기밀성, 무결성, 가용성, 부인방지, 인증, 인가) 중 하나로 분류하였다.

[0005] 표준들은 주로 기밀성, 무결성, 인증, 인가를 다루고 있었으며 가용성과 부인방지에 대한 연구는 비교적 적은 상태이다. 또한, IoT 인터워킹과 관련된 종래 기술로는 OIC(Open Interconnect Consortium), AllJoyn, LWM2M(Lightweight M2M)의 대표적인 IoT 표준기술들을 간단히 기술하고 oneM2M과의 연동 기술과 실제 구현된 사례를 소개하였으며, 다른 일 실시예 기술로는 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 제안한 바도

있다. 그리고, 다른 종래 기술로 IPE(Interworking Proxy Entity)를 통해 oneM2M 기반의 IoT 시스템과 기존의 IoT 상품이 인터워킹할 수 있는 방법을 연구하고, 보안을 고려하여 OAuth 2.0 인증, 인가 프레임워크를 적용하였다. 하지만 토큰을 발급받는 과정에서 로그인을 해야 하거나 인가 코드를 입력해야 하기 때문에 사람이 개입할 수 있는 일부 시나리오에서만 활용할 수 있는 한계가 존재한다.

[0007] 따라서, IoT 환경의 디바이스들은 자동화된 경우가 많아서 사람의 개입이 필요하게 되면 디바이스의 활용성도 제한되므로 보안 과정 또한 최대한 자동화될 수 있는 이중 IoT 서비스 플랫폼들 간의 보안 상호운용성 프레임워크의 필요성이 대두된다.

발명의 내용

해결하려는 과제

[0009] 본 발명의 실시예들은, 이중의 IoT 플랫폼 간에 인터워킹(interworking)을 수행할 때 IoT 플랫폼 간에 보안 상호운용성을 확보할 수 있는 보안 상호운용성 프레임워크 및 그 장치를 제공한다.

[0010] 구체적으로, 본 발명의 실시예들은 IoT 플랫폼인 FIWARE와 oneM2M 간에 인터워킹을 수행할 때 보안 인터워킹의 핵심인 인증과 인가를 제공할 수 있는 OAuth 2.0 기반 보안 컴포넌트를 제공함으로써, 이중 IoT 플랫폼 간에 보안 상호운용성을 확보할 수 있는 보안 상호운용성 프레임워크 및 그 장치를 제공한다.

과제의 해결 수단

[0012] 본 발명의 일 실시예에 따른 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 프레임워크는 클라이언트 정보가 등록된 이중 사물 인터넷(IoT) 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 보안 컴포넌트를 이용하여 상기 클라이언트의 인증을 수행하는 단계; 상기 클라이언트가 인증되면 상기 보안 컴포넌트에서 상기 클라이언트로 토큰을 발급하는 단계; 및 상기 클라이언트에서 상기 발급받은 토큰을 통해 자원이 요청되면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공하는 단계를 포함한다.

[0013] 상기 토큰을 발급하는 단계는 OAuth 표준을 통해 상기 토큰을 발급할 수 있다.

[0014] 상기 인증을 수행하는 단계는 상기 클라이언트에서 Resource Owner Password Credentials 방식을 이용하여 상기 토큰 발급을 요청할 수 있다.

[0015] 상기 요청된 자원을 상기 클라이언트로 제공하는 단계는 상기 발급받은 토큰을 통해 자원이 요청되면 상기 자원을 요청한 토큰을 검증하고, 상기 토큰의 검증이 성공하면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공할 수 있다.

[0016] 상기 보안 컴포넌트는 oneM2M 보안 컴포넌트를 포함할 수 있다.

[0017] 상기 이중 클라이언트의 인증을 수행하는 단계는 상기 이중 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 미리 설정된 적어도 하나 이상의 REST API에 대해서만 상기 보안 컴포넌트를 이용하여 상기 이중 클라이언트의 인증을 수행할 수 있다.

[0019] 본 발명의 일 실시예에 따른 이중 사물 인터넷 서비스 플랫폼 간의 보안 상호운용성 장치는 클라이언트 정보가 등록된 이중 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급 요청을 수신하는 수신부; 상기 토큰 발급을 요청한 상기 클라이언트의 인증을 수행하고, 상기 클라이언트가 인증되면 상기 클라이언트로 토큰을 발급하는 승인부; 및 상기 클라이언트에서 상기 발급받은 토큰을 통해 자원이 요청되면 상기 요청된 자원을 상기 클라이언트로 제공하도록 제어하는 제어부를 포함한다.

[0020] 상기 승인부는 OAuth 표준을 통해 상기 토큰을 발급할 수 있다.

[0021] 상기 수신부는 상기 클라이언트에서 Resource Owner Password Credentials 방식을 이용하여 요청한 상기 토큰 발

급 요청을 수신할 수 있다.

- [0022] 상기 제어부는 상기 발급받은 토큰을 통해 자원이 요청되면 상기 자원을 요청한 토큰을 검증하고, 상기 토큰의 검증이 성공하면 상기 보안 컴포넌트의 제어를 통해 상기 요청된 자원을 상기 클라이언트로 제공할 수 있다.
- [0023] 상기 보안 컴포넌트는 oneM2M 보안 컴포넌트를 포함할 수 있다.
- [0024] 상기 승인부는 상기 이중 IoT 서비스 플랫폼의 클라이언트로부터 토큰 발급이 요청되면 미리 설정된 적어도 하나 이상의 REST API에 대해서만 상기 보안 컴포넌트를 이용하여 상기 이중 클라이언트의 인증을 수행할 수 있다.

발명의 효과

- [0026] 본 발명의 실시예들에 따르면, 이중의 IoT 플랫폼 간에 인터워킹(interworking)을 수행할 때 IoT 플랫폼 간에 보안 상호운용성을 확보할 수 있다.
- [0027] 구체적으로, 본 발명의 실시예들은 IoT 플랫폼인 FIWARE와 oneM2M 간에 인터워킹을 수행할 때 보안 인터워킹의 핵심인 인증과 인가를 제공할 수 있는 OAuth 2.0 기반 보안 컴포넌트를 제공함으로써, 이중 IoT 플랫폼 간에 보안 상호운용성을 확보할 수 있다.
- [0028] 또한, 본 발명의 실시예들에 따르면 보안 상호운용성 프레임워크를 이용하여 적절한 권한 없이 디바이스를 제어하는 사례들을 방지할 수 있으며 이를 활용하여 타 플랫폼과의 보안 인터워킹을 수행할 수 있다.

도면의 간단한 설명

- [0030] 도 1은 OAuth의 동작을 설명하기 위한 일 예시도를 나타낸 것이다.
- 도 2는 Resource owner password credentials 방식의 동작 개요를 나타낸 것이다.
- 도 3은 FIWARE의 보안 아키텍처의 동작 개요를 나타낸 것이다.
- 도 4는 보안 레벨 1의 과정을 설명하기 위한 일 예시도를 나타낸 것이다.
- 도 5는 보안 레벨 2의 과정을 설명하기 위한 일 예시도를 나타낸 것이다.
- 도 6은 보안 컴포넌트를 사용하는 oneM2M 구조를 나타낸 것이다.
- 도 7은 본 발명의 일 실시예에 따른 이중 IoT 플랫폼 간의 보안 상호운용성 프레임워크에 대한 동작 시나리오를 나타낸 것이다.
- 도 8은 FIWARE와 oneM2M 플랫폼 간의 보안 인터워킹에 대한 동작 흐름을 나타낸 것이다.
- 도 9는 보안 GE를 모두 실행한 뒤, KeyRock에게 보낸 토큰 발급 요청과 이후 결과를 나타낸 것이다.
- 도 10은 토큰을 사용하여 FIWARE에서 보호되고 있는 자원인 'My_Room' 정보에 접근한 결과를 나타낸 것이다.
- 도 11은 정상적인 토큰이 아닌 임의의 토큰을 통해 자원을 요청했을 때의 결과 화면을 나타낸 것이다.
- 도 12는 요청자에게 권한이 없는 자원에 대해 접근했을 때의 결과 화면을 나타낸 것이다.
- 도 13은 액세스(access) 동안 Wilma의 로드예 대한 일 예시도를 나타낸 것이다.
- 도 14는 oneM2M 보안 컴포넌트를 이용하여 oneM2M 자원을 이용할 때의 동작 개요를 나타낸 것이다.
- 도 15는 oneM2M 보안 컴포넌트로부터 토큰 응답에 대한 결과 화면의 일 예시도를 나타낸 것이다.
- 도 16은 보안 컴포넌트로부터 예러가 발생한 일 예시도를 나타낸 것이다.
- 도 17은 oneM2M으로부터 성공적인 응답에 대한 결과 화면의 일 예시도를 나타낸 것이다.
- 도 18은 본 발명을 적용한 LED 구조에 대한 일 예시도를 나타낸 것이다.

발명을 실시하기 위한 구체적인 내용

- [0031] 이하, 본 발명에 따른 실시예들을 첨부된 도면을 참조하여 상세하게 설명한다. 그러나 본 발명이 실시예들에 의해 제한되거나 한정되는 것은 아니다. 또한, 각 도면에 제시된 동일한 참조 부호는 동일한 부재를 나타낸다.
- [0033] 본 발명의 실시예들은, 실시예들은 IoT 플랫폼인 FIWARE와 oneM2M 간에 인터워킹을 수행할 때 보안 인터워킹의 핵심인 인증과 인가를 제공할 수 있는 OAuth 2.0 기반 보안 컴포넌트를 제공함으로써, 이중 IoT 플랫폼 간에 보안 상호운용성을 확보하는 것을 그 요지로 한다.
- [0034] 여기서, 본 발명은 토큰을 발급받을 때 'Resource Owner Password Credentials' 방식을 사용할 수 있는데, 이 방식은 추가적인 인증 과정 없이, 자원 소유자의 정보만 확실하면 바로 토큰을 발급받을 수 있기 때문에 토큰 발급이 간단하고 발급 과정의 자동화 또한 간편한 장점이 있다.
- [0035] FIWARE와 oneM2M 간에 보안 인터워킹이 수행되기 위해서는 두 개의 플랫폼에 각각 보안 컴포넌트가 필요하지만 FIWARE의 경우에만 보안 컴포넌트가 존재하고 oneM2M은 그렇지 않다. 따라서, 본 발명에서는 보안 인터워킹에 사용할 수 있는 oneM2M 보안 컴포넌트를 제공하는 것으로 설명한다. 하지만, 본 발명에 따른 이중 IoT 플랫폼 간 보안 상호운용성 프레임워크는 oneM2M 보안 컴포넌트에 한정되지 않으며 이중 IoT 플랫폼 간 인터워킹을 수행할 때 보안 상호운용성을 제공할 수 있는 모든 표준 또는 기기 또는 시스템 등에 적용할 수 있다.
- [0037] 인터워킹은 서로 다른 플랫폼이 상호간에 통신하는 것을 의미하고, 보안 인터워킹은 인터워킹 중 보안을 확보하는 것이다. 보안 인터워킹에서는 상대방이 누구인지 확인하는 과정인 인증과 확인된 상대방에게 권한을 부여하는 인가 과정이 중요한데, OAuth 2.0은 이런 경우에 사용할 수 있는 인증, 인가 프레임워크 표준이다. OAuth 2.0은 FIWARE와 oneM2M의 보안 컴포넌트에 모두 사용되고, OAuth 2.0은 도 1에 도시된 바와 같이 동작하며, 클라이언트, 자원 소유자, 인가 서버, 자원 서버 네 가지 역할이 존재한다.
- [0038] 클라이언트(client)는 자원(REST API 등)을 이용하기 위하여 자원 소유자(resource owner)에게 자원을 사용하기 위한 허가(Grant)를 받는다.
- [0039] 여기서, 자원 소유자에게 인가를 요청하고 승인하는 과정은 인가 서버(authorization server)를 통해 간접적으로 이루어질 수도 있다.
- [0040] 클라이언트는 인가 서버에서 인증을 받고 자원 소유자의 허가를 검증 받은 뒤, 토큰을 발급 받고 이를 통해 자원 서버(resource server)로 자원을 요청한다.
- [0041] OAuth 2.0 표준에서는 도 1의 인가를 허가하는 과정(authorization grant)이 authorization code, implicit, resource owner password credentials, client credentials 네 가지 타입으로 수행될 수 있다. 표준에서 클라이언트는 클라이언트의 신뢰도가 높은 것을 의미하는 confidentials 타입과 그렇지 않은 public 타입으로 나누어지는데, Authorizationcode와 implicit 방식은 클라이언트가 public 타입일때 일반적으로 사용하며, 나머지는 confidentials 타입일 때 사용한다.
- [0042] 본 발명에서는 resource owner password credentials 방식을 사용할 수 있다. 이 방식은 자원 소유자의 정보(보통 username,password)만 알면 다른 추가적인 과정 없이 곧바로 토큰을 발급받을 수 있지만 자원 소유자의 정보는 기본적으로 노출되어서는 안되므로 클라이언트와의 신뢰도가 높은 일부 상황에서만 활용해야 하고, 실제 환경에서는 TLS 등의 암호화 기술과 사용될 수 있다. 하지만 토큰을 발급받는 과정 자체를 자동화하기 간단하기 때문에 사람의 개입 없이 동작해야 하는 confidentials 타입의 IoT 디바이스에서 사용하기에 적합하다.
- [0043] 도 2는 Resource owner password credentials 방식의 동작 개요를 나타낸 것으로, 도 2에 도시된 바와 같이 클라이언트는 기존에 자원 소유자의 비밀 정보(username, password)를 알고 있거나 토큰을 요청하기 전에 받아야 하며, 클라이언트가 자원 소유자의 비밀 정보를 알고 있다면, 인가 서버에서 토큰을 발급해주는 토큰 엔드포인트에 resource owner password credentials 방식에서 필요한 정보들을 보내면 인가 서버로부터 토큰을 발급받을 수 있다. 토큰 발급을 위해 필요한 정보는 grant_type, username, password, scope이며, scope를 제외한 정보는 모두 필수 정보이므로 반드시 HTTP body에 application/x-www-form-urlencoded 방식으로 인코딩해서 첨부해야 한다. grant_type은 resource owner password credentials를 나타내는 'password'로 설정하고, username과 password는 자원 소유자의 것으로 입력한다. Scope는 원하는 권한(읽기, 쓰기 등)이나 접근하고 싶은 범위를 정

하는데 사용되며, 필수는 아니므로 필요한 경우에만 입력한다.

[0044] 예를 들어, 토큰 발급 요청을 만든다면 아래와 같은 형태가 될 수 있다.

[0046] POST /token HTTP/1.1

[0047] Host: authorizationServer.com

[0048] Authorization:Basic pPpDsBBg30G4AAcvbdippM2JL02P

[0049] Content-Type: application/x-www-form-urlencoded

[0050] grant_type=password&username=ABC&password=owner

[0052] 상기 기재된 토큰 요청은 토큰 발급 요청을 위한 body의 필수 정보 세 가지(grant_type, username, password)와 body의 인코딩 방식을 알려주는 Content-Type 헤더, 클라이언트 정보를 HTTP Basic Authentication 방식으로 인코딩한 Authorization 헤더를 authorizationServer.com의 토큰 엔드 포인트인 /token으로 POST 형식으로 보낸다는 것을 의미한다. 상술한 바와 같이 resource owner password credentials 방식을 사용하기 위해서 grant_type은 password로 입력해야 하며, username과 password는 자원 소유자의 것으로 입력한다. 인가 서버에서 위의 요청을 성공적으로 검증하면, 아래와 같은 결과(토큰 응답)를 보낼 수 있다.

[0054] POST /token HTTP/1.1

[0055] Host: authorizationServer.com

[0056] Authorization:Basic pPpDsBBg30G4AAcvbdippM2JL02P

[0057] Content-Type: application/x-www-form-urlencoded

[0058] grant_type=password&username=ABC&password=owner

[0060] 상기 기재된 인가 서버의 응답을 살펴보면, 헤더에서 body 데이터는 json임을 명시했고 캐시를 저장하지 않도록 설정한 것을 알 수 있다. 캐시를 하지 않는 것은 서버의 응답을 네트워크 상에 저장하면 토큰까지 기록되기 때문에 토큰이 노출될 위험이 크기 때문이다. Body에는 토큰(access_token)과 토큰의 타입(token_type), 만료 기간(expires_in)이 존재한다. 표준 상에서 토큰에 대한 사양은 정확히 기재되어 있지 않지만 토큰은 일반적으로 무작위의 문자열이고 Bearer 타입은 가장 기본적인 타입으로, Bearer 토큰만 소유하고 있으면 토큰에 할당된 권한을 이용할 수 있다. 만료 기간은 보통 초 단위로 주어지므로 위의 토큰은 한 시간 동안 이용할 수 있다. 만약, 토큰과 함께 갱신 토큰(refresh token)이 주어지면, 갱신 토큰을 이용해서 토큰을 발급받는 과정이 간단해지지만 갱신 토큰의 발급은 선택 사항이므로 인가 서버에서 주어지지 않을 수도 있다.

[0062] 본 발명의 보안 상호운용성 프레임워크 또는 보안 인터워킹 구조에서 사용되는 FIWARE의 보안 컴포넌트와 oneM2M의 보안 컴포넌트 그리고 보안 인터워킹 구조에 대해 설명하면 다음과 같다.

[0064] 1. FIWARE의 보안 컴포넌트

[0065] FIWARE의 보안 아키텍처는 KeyRock, AuthZForce, Wilma라고 불리는 세 가지 GE(Generic Enabler)로 이루어져 있으며 FIWARE의 보안 컴포넌트로서 서로 협업한다.

[0066] 도 3은 FIWARE의 보안 아키텍처의 동작 개요를 나타낸 것으로, 도 3에 도시된 바와 같이, 가장 먼저 접근자(Requester)는 KeyRock에 인증 정보를 전송하여 인증을 받은 뒤(access token request), 액세스 토큰을 발급받아 Wilma에게 원하는 자원을 요청한다(resource request with access token). 이후 Wilma는 해당 토큰을 검증(token validation request)하고 요청자에 관한 정보를 KeyRock에게 전달 받아 AuthZForce에 전달하여 권한

부여를 요청한다(authorization request). AuthZForce의 응답이 Permit인 경우, Wilma가 요청자 대신 서버에 자원을 요청(resource request)하고 결과를 요청자에게 전달하게 된다. 각 GE에 대해 더 상세히 기술하면 다음과 같다.

- [0067] KeyRock은 백 엔드의 Keystone과 프론트 엔트의 Horizon으로 구성된다. OpenStack 프로젝트의 일부인 Keystone은 인증과 인가, 서비스 discovery 기능을 제공하고 Horizon은 회원, 클라이언트 등 KeyRock의 정보들을 관리할 수 있는 웹 페이지를 제공한다. KeyRock은 OAuth 2.0, SAML 2.0을 통해서 간단한 인증과 인가를 수행할 수 있으며 유저와 어플리케이션 정보를 관리한다. 또한 KeyRock은 SSO(Single Sign-On), Identity Federation을 지원한다.
- [0068] AuthZForce는 인가 정책을 관리하고 인가를 결정하여 AuthorizationPDP(Policy Decision Point)라고도 불린다. AuthZForce는 PAP(Policy Administration Point) API와 PDP API를 제공하는데, 정책 관리에는 PAP API를 이용하고, 정책 결정에는 PDP API를 이용한다. 인가 결정은 요청자의 정보, 자원 경로(Path), HTTP Method 등을 참조하여 XACML(eXtensible Access Control Markup Language)로 만들어진 정책을 기반으로 수행된다.
- [0069] Wilma는 서버의 앞단에서 자원을 보호하는 컴포넌트로 유저의 요청을 인터셉트하여 서버를 보호한다. 유저를 인증하기 위해서 KeyRock에 의존하며, 권한을 부여하기 위해서 AuthZForce와 협업한다. 서버에 접근하는 모든 요청은 Wilma를 거치게 되고 Wilma의 보안은 크게 세 가지 레벨로 나뉘어지는데, 레벨 1(Authentication) 보안은 가장 간단하지만 취약하며, 유일하게 AuthZForce가 사용되지 않는다. 레벨 2(Basic Authorization)부터는 AuthZForce를 통해 요청에 대해서 인가하는 작업이 추가되고 레벨 3(Advanced Authorization)은 XACML Request를 이용하여 정교한 인가 요청을 보낼 수 있다.
- [0070] 도 4에 도시된 레벨 1 과정은 도 3에 도시된 동작과 전체적으로 유사하지만, 대부분의 시나리오에서 AuthZForce가 필요 없기 때문에 AuthZForce가 생략되어 있고 유저의 토큰을 검증하는 작업만 수행된다. 반면 도 5에 도시된 레벨 2 과정에서는 관리자가 AuthZForce의PAP API를 이용하여 XACML 정책을 만들 수 있으며 이를 기반으로 레벨 1보다 정교한 인증 및 인가를 할 수 있다.
- [0071] 레벨 3은 보안 레벨 2의 인가 요청 단계에서 역할과 HTTP Method (Verb), 경로(Path) 외 더 다양한 파라미터들로 XACML Request를 보낼 수 있다는 점만 다르다. 예를 들어, 레벨 3은 요청자의 역할이나 경로 외, 사는 지역이나 접속 시간, 사용하는 메일의 도메인 등을 검증할 수 있다. 레벨 3 보안을 구현하려면 특정 도메인에 필요한 XACML 정책이 만들어져야 하고 Wilma의 코드가 적절히 수정되어야 한다.
- [0073] 2. oneM2M 보안 컴포넌트
- [0074] FIWARE와 달리, oneM2M은 보안 아키텍처에 대한 논의가 이루어지고 있지만 이에 대해서 구현된 컴포넌트가 없다. 본 발명은 보안 인터위킹 제공하기 위한 것으로, IoT 플랫폼간의 인터위킹 시나리오에서 인증과 인가를 제공하는 것이 핵심이기 때문에 다양한 보안적 기능을 요구하는 oneM2M의 보안 아키텍처를 분석하거나 구현하지 않고 인증과 인가를 제공하는 OAuth 2.0 기반 보안 컴포넌트를 제공하는 것이다.
- [0075] 여기서, 본 발명에서 제공되는 OAuth 2.0 기반 보안 컴포넌트는 KETI(Korea Electronics Technology Institute)에서 oneM2M의 표준을 토대로 개발한 Mobius에 적용할 수 있다.
- [0076] oneM2M 보안 컴포넌트가 정상적으로 동작하기 위해서는 다음과 같은 기본적인 요구사항들이 존재한다.
- [0077] (1) 토큰을 발급받을 때, 클라이언트 정보와 회원 정보가 필요하므로 회원 등록 페이지와 클라이언트 등록 페이지가 필요하다.
- [0078] (2) 클라이언트 정보와 회원 정보를 저장하기 위해 데이터베이스가 필요하다.
- [0079] (3) 클라이언트 비밀 정보와 회원의 비밀번호 정보는 암호화가 되어야 한다. 만약 회원 정보나 클라이언트 정보가 유출되면 공격자는 해당 정보를 이용하여 특정 자원에 접근 가능한 토큰을 생성할 수 있게 된다.
- [0080] (4) Mobius에 대한 모든 요청은 반드시 보안 컴포넌트를 거쳐야 한다. 특히, oneM2M의 시스템을 제어하는 명령이나 중요 데이터, 개인 정보 등을 요청하는 REST API같은 경우 반드시 보안 컴포넌트를 통해 토큰을 검증 받아야 한다.
- [0081] Mobius는 디바이스(Application Entity)의 등록이나 관리, 자원의 CRUD(Create, Read, Update, Delete) 동작

등을 수행하는데 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.

[0082] oneM2M 보안 컴포넌트는 Mobius에 접근하는 모든 요청을 막고, 정당한 토큰이 있어야만 REST API를 사용할 수 있도록 설정되었다. 이러한 환경에서는 Mobius의 실행 파일 중 일부가 Mobius가 실행될 때 REST API를 사용하기 때문에, 기존의 Mobius 코드가 정상적으로 실행되지 않는다. 해당 파일들이 Mobius의 REST API를 사용할 때 토큰을 발급받고, 첨부하도록 만들어져 있지 않기 때문이다. 따라서 실행될 때 Mobius에 REST API 요청을 하는 pxy_coap.js, pxy_mqtt.js, ts_agent.js같은 파일의 코드를 수정하여 Mobius의 OAuth 2.0 서버에서 토큰을 발급받고, 해당 토큰으로 REST API를 사용하도록 수정되어야 한다.

[0083] 도 6은 보안 컴포넌트를 사용하는 oneM2M 구조를 나타낸 것으로, 도 6에 도시된 바와 같이 보안 컴포넌트는 IoT 어플리케이션이나 디바이스 플랫폼에서 REST API를 이용하여 Mobius에 자원을 요청하면 차단하고 토큰의 검증이 끝나면 요청을 이어서 처리한다.

[0085] 3. FIWARE-oneM2M 보안 인터워킹

[0086] 도 7은 본 발명의 일 실시예에 따른 이중 IoT 플랫폼 간의 보안 상호운용성 프레임워크에 대한 동작 시나리오를 나타낸 것으로, FIWARE와 oneM2M 플랫폼 간의 보안 인터워킹 동작 시나리오를 나타낸 것이다.

[0087] 도 7에 도시된 바와 같이, FIWARE와 oneM2M 플랫폼 간의 보안 인터워킹 구조에서는 FIWARE나 oneM2M의 자원에 접근하려면 보안 컴포넌트들을 거쳐서 인증과 인가를 받아야 한다. 예를 들어, FIWARE 플랫폼을 이용하는 스마트 폰이 oneM2M 도어 락을 열기 위해서 oneM2M의 보안 컴포넌트에서 인증과 인가를 받아야 하고, oneM2M도 마찬가지로 FIWARE의 램프를 켜기 위해서는 FIWARE의 보안 컴포넌트들(GEs)에 의해 인증과 인가를 받아야 한다. 이 과정을 도 8을 참조하여 조금 더 설명하면 다음과 같다.

[0088] 도 8은 FIWARE와 oneM2M 플랫폼 간의 보안 인터워킹에 대한 동작 흐름을 나타낸 것으로, 도 8에 도시된 바와 같이 oneM2M 디바이스가 FIWARE의 자원을 요청하려면 먼저 KeyRock에 가입하고 클라이언트를 등록한 뒤, OAuth 2.0을 통해 토큰을 발급 받아야 한다. 그리고 Wilma에 자원을 요청하면 KeyRock과 AuthZForce를 통해서 토큰의 검증 받아야 한다. 모든 절차가 정상적으로 처리되면 Wilma는 요청한 결과를 oneM2M에 반환한다.

[0089] 이와 같이, FIWARE 측에서도 oneM2M의 자원을 제어하기 위해서는 oneM2M의 유저/클라이언트 페이지를 이용하여 가입하고 클라이언트를 등록한 뒤 보안 컴포넌트에서 토큰을 발급 받아야 한다. FIWARE 디바이스가 토큰을 발급 받으면, 주어진 권한에 따라서 oneM2M을 기반으로 동작하는 스마트 카를 제어하거나, 특정 데이터를 조작하거나 요청하는 등 oneM2M의 자원을 이용할 수 있게 된다.

[0090] 이 때, 몇 가지 쟁점이 있는데 예를 들어, FIWARE에서 발급 받은 토큰이 있고 이것을 oneM2M 쪽에서 사용할 수 있을지, 혹은 반대로 oneM2M에서 발급 받은 토큰을 FIWARE에서 사용할 수 있을지의 문제들이 있을 수 있다.

[0092] FIWARE-oneM2M의 보안 컴포넌트, 보안 인터워킹 구현

[0094] 1. FIWARE 보안 컴포넌트 구현

[0095] FIWARE 보안 컴포넌트들은 상술한 도 3과 같이 동작하며, 이에 대해서는 도 3에서 설명하였기에, 이에 대한 상세한 설명은 생략한다. FIWARE 보안 GE들은 보안 아키텍처를 구현하기 위해 다른 버전들보다 KeyRock 5.4.1, AuthZForce5.4.1, Wilma 5.4.0을 이용하는 것이 안정적일 수 있다. 물론, 이러한 버전들에 한정되지 않으며 본 발명을 구현하기 위한 보안 컴포넌트들을 모두 사용할 수 있다.

[0096] 도 9는 보안 GE를 모두 실행한 뒤, KeyRock에게 보낸 토큰 발급 요청과 이후 결과를 나타낸 것으로, FIWARE에서의 토큰 발급 요청은 KeyRock의 토큰 엔드 포인트(<http://localhost:8000/oauth2/token>)에 POST 방식으로 보내야 한다. 이 때 필요한 정보는 OAuth 2.0 표준에 따라 클라이언트 ID와 Secret, Content-Type(application/x-www-form-urlencoded), grant_type (password), username, password이며, 이는 사용하는 인가 방식(grant type)에 따라서 달라지는데, 본 발명에서는 간단한 Resource Owner Password Grant 방식을 사용할 수 있다. 기

본적인 인증을 위해 클라이언트 ID와 Secret은 HTTP Basic Authentication에 따라 base-64로 인코딩되며 실제 상용 환경에서는 Body의 password 정보와 함께 TLS 같은 암호 기술과 혼용할 수 있다.

- [0097] 다음으로 Body의 username과 password 정보는 KeyRock에 등록되어 있는 자원 소유자의 정보이다. Root 계정과 다른 없는 'idm' 외 다른 계정을 사용하려면 KeyRock의 웹 사이트에 접속하여 가입 절차를 수행하면 된다. 보통은 'http://localhost:8000'이 KeyRock의 주소이며 가입에는 메일 인증이 필요하기 때문에 환경에 따라서 메일 서버를 구성할 수도 있다. KeyRock이 사용하는 메일 서버를 지정하기 위해서는 Horizon의 local_settings.py 파일에 위치한 'EMAIL_HOST', 'EMAIL_PORT' 같은 옵션들을 적절하게 수정할 수 있다.
- [0098] 도 9에서 알 수 있듯이, 토큰 발급 요청에 필요한 정보들이 모두 제대로 갖춰지면 토큰과 관련 정보들이 KeyRock으로부터 전송된다. FIWARE의 토큰은 일반적인 'Bearer' 방식이며 만료 시간은 한 시간으로 설정되어 있다. 토큰을 발급 받으면, 서버의 보호되고 있는 자원을 요청할 수 있다.
- [0099] 도 10은 토큰을 사용하여 FIWARE에서 보호되고 있는 자원인 'My_Room' 정보에 접근한 결과를 나타낸 것으로, 토큰은 헤더의 'X-Auth-Token'에 넣고, Wilma(localhost:1706)에게 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 발명에서는 Context 정보를 관리해주는 Orion Context Broker의 주소(localhost:1026)로 설정할 수 있다. 따라서 Wilma는 토큰을 성공적으로 검증하면 Orion Context Broker에 요청을 Redirect하여 'My_Room'의 정보를 검색하고, 이에 대한 결과를 요청자에게 다시 보내준다. 만약 토큰을 발급받은 유저에게 'My_Room'에 대한 권한이 없거나, 토큰이 만료되었거나, 정상적인 토큰이 아니라면 Wilma는 접근을 도 11이나 도 12와 같이 불허한다.
- [0100] 예를 들어, 도 11에 도시된 바와 같이 정상적인 토큰이 아닌 임의의 토큰(123456789)을 통해 자원을 요청했을 때 Wilma는 접근을 불허하고, 도 12에 도시된 바와 같이 요청자에게 권한이 없는 자원에 대해 접근했을 때 Wilma는 접근을 불허한다.
- [0101] 이 때, FIWARE의 모든 보안 GE 테스트는 Wilma의 보안 레벨 2로 진행될 수 있으며, 상기 도 3에서의 과정들은 도 13에 도시된 Wilma 로그를 통해 정확히 확인할 수 있다.
- [0102] Wilma는 가장 먼저 KeyRock와 협업하여 토큰을 검증하고 AuthZForce를 통해 유저의 역할, HTTP Verb, 자원 경로를 확인하며, 토큰을 발급받은 'user0' 계정은 'Other_Room'에 대해서 권한이 없기 때문에 요청은 정상적으로 실패한다.
- [0104] 2. oneM2M 보안 컴포넌트 구현
- [0105] 본 발명에서는 oneM2M 보안 컴포넌트의 OAuth 2.0 서버 구현을 위해 Node.js 기반의 'oauth2-server' 모듈을 사용하고 토큰의 발급을 위해 필요한 클라이언트, 회원 정보들은 Node.js를 이용하여 임시로 만든 웹 페이지를 통해 생성한다.
- [0106] 도 14는 oneM2M 보안 컴포넌트를 이용하여 oneM2M 자원을 이용할 때의 동작 개요를 나타낸 것으로, oneM2M 기반의 보안 상호운용성 프레임워크가 동작하는 흐름을 나타낸 것이다.
- [0107] 도 14를 참조하여 oneM2M 기반의 보안 상호운용성 프레임워크에 대해 설명하면, 외부 IoT 서비스 플랫폼의 클라이언트는 보안 상호운용성 프레임워크를 통해 토큰을 발급 받아야 하며, 이를 위해 회원/클라이언트 등록이 선행되어야 한다.
- [0108] 여기서, 회원/클라이언트 등록은 회원/클라이언트 페이지를 통해 이루어질 수 있다.
- [0109] 회원/클라이언트 등록 후 OAuth 2.0 과정을 마치면 토큰이 발급되고 클라이언트는 해당 토큰으로 자원(IoT 서비스 등)을 이용할 수 있게 된다.
- [0110] 본 발명에서는 토큰을 발급 받기 위해서 Resource Owner Password Credentials 방식을 사용할 수 있으며, 도 15는 Resource Owner Password Credentials 방식을 이용하여 oneM2M 보안 컴포넌트에 토큰 발급 요청을 보낸 결과를 나타낸 것이다.
- [0111] oneM2M 보안 컴포넌트에서 토큰을 발급받기 위해서는 토큰 엔드포인트인 'localhost:7579/oauth/token'에 필요한 정보(grant_type, 클라이언트 ID와 Secret, 자원 소유자의 username과 password)를 POST 방식으로 요청을

보내야 하고, 토큰 발급이 성공하면 도 15와 같이 토큰 타입과 토큰, 만료 시간이 전송된다. 토큰 타입과 만료 시간은 FIWARE와 마찬가지로 각각 Bearer와 1시간이며 토큰은 무작위한 문자열일 수 있다.

[0112] 만약 보안 컴포넌트가 동작하는 상태에서 토큰없이 Mobius의 자원에 접근하거나 비정상 토큰을 사용하면 도 16에 도시된 일 예와 같은 결과가 반환된다. 여기서, 코드 400은 요청자가 자원을 요청할 때 토큰이 없으면 나타나는 에러를 의미하고, 코드 401은 요청자가 제시한 토큰의 검증이 실패했을 때 주어질 수 있다.

[0113] 반면, 정상 토큰으로 Mobius에 요청하는 경우 도 17에 도시된 바와 같이 요청자가 원하는 결과를 반환해준다.

[0115] 3. LED를 활용한 보안 인터워킹

[0116] 본 발명에서의 LED 예제는 OCEAN(Open alliance for iotstANDARD)의 LED 예제와 도 8의 보안 인터워킹 구조를 기반으로 개발될 수 있으며, 도 18과 같은 구조를 가질 수 있다.

[0117] 여기서, 도 18의 구조는 LED만이 아니라 스마트 홈의 온도 조절기, 도어락 등의 다른 사물들이나 &Cube와의 여러 디바이스 플랫폼을 대상으로 적용될 수 있다.

[0118] 도 18에 도시된 바와 같이, FIWARE나 기타 플랫폼이 oneM2M의 LED를 제어하기 위해서는 Mobius에 요청을 보내야 하는데, 본 발명에서는 반드시 그보다 먼저 Mobius의 보안 컴포넌트에서 REST API에 접근하기 위한 토큰을 발급 받아야 한다.

[0119] 발급받은 토큰은 Mobius에 LED를 켜기 위한 요청을 보낼 때 헤더의 'Authorization'에 'Bearer 토큰' 형태로 조합하여 첨부되어 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 동작하게 된다. 만약 토큰이 제대로 검증이 안 된다면, LED 제어 명령은 보안 컴포넌트에서 막히게 된다. 보안 컴포넌트를 통해서 FIWARE나 기타 플랫폼에서 들어오는 토큰이 없는 요청들은 Mobius의 자원을 이용할 수 없게 된다.

[0120] 여기서, 본 발명은 모든 요청에 대해서 토큰을 검증하도록 할 수도 있지만, 보안 컴포넌트의 설정과 보안 정책에 따라서 특정 REST API들의 이용만 토큰이 필요하도록 만들 수도 있다. 본 발명에서의 중요한 점은 보안을 위협할 수 있는 REST API는 반드시 사용할 때 토큰의 발급과 검증을 받도록 하는 것이다.

[0121] 또한, LED 제어를 위한 요청은 REST API이므로 Authorization 헤더를 통해서 토큰을 넣어야만 정상적으로 LED를 제어할 수 있다. 여기서 토큰은 자체 문자열만 Authorization에 입력되지 않고, 토큰의 타입을 나타내는 Bearer와 같이 입력되어야 한다.

[0122] 이렇게 도 18에서 도시한 구조는 LED로 한정되지 않으며 다른 사물들을 대상으로도 사용할 수 있다.

[0124] 이와 같이, 본 발명에 따른 보안 상호운용성 프레임워크는 적절한 권한 없이 디바이스를 제어하는 사례들을 방지할 수 있으며 이를 활용하여 타 플랫폼과의 보안 인터워킹을 수행할 수 있다.

[0125] 보안 인터워킹 시나리오는 크게 관리자 시나리오와 일반 시나리오 두 가지로 나눌 수 있으며, 관리자 시나리오는 본 발명에서 설명한 방식으로 이는 자원 소유자나 클라이언트에 관한 정보를 모두 알고 있는 상황에서 Resource Owner Password Credentials나 Client Credentials 방식을 이용하여 토큰을 발급 받아 자원을 이용하는 시나리오이다. 토큰을 발급받기 위한 정보에 접근할 수 있는 대상 자체가 제한적이므로 토큰을 발급받기가 쉽고 자동화 작업이 간단한 장점이 있지만 자원 소유자나 클라이언트 정보를 안전하게 관리하여야 한다. 일반 시나리오는 관리자가 아닌 일반 유저들이 사용하는 시나리오로 관리자 시나리오와는 달리 Authorization Code, Implicit 방식을 사용하여 토큰을 발급 받는다. 일반 시나리오에서 중요한 것은 유연한 토큰 발급의 가능성이다. 기존의 PC 환경과 달리 IoT의 상황인지(context-aware) 특성을 이용하면 더 유연한 토큰 발급이 가능할 수 있다. 예를 들어, 특정 데이터를 업데이트 하려고 할 때, A 기업에서 만든 온도계 중에서도 20도 이하의 지역에 위치한 온도계만 토큰을 발급받을 수 있게 하거나 주변에 사람이 존재해야만 토큰 발급이 가능하도록 보안 정책을 만들 수 있다.

[0126] 더 나아가, 보안 정책에 따라 토큰을 사용할 때 FIWARE 토큰을 oneM2M에서 사용하거나 oneM2M 토큰을 FIWARE에서 사용할 수도 있다. 다만, oneM2M 측에서 FIWARE 토큰을 받았을 때 해당 토큰이 FIWARE의 토큰임을 알아야 하

고, FIWARE 측에 토큰의 검증을 요청할 수 있어야 하며 해당 토큰의 권한으로 특정 oneM2M 자원에 접근할 수 있는지 확실한 보안 정책이 수립되는 것이 바람직하다.

[0128] 이러한 본 발명에 따른 보안 상호운용성 프레임워크는 기능적인 구성 수단으로서의 장치 또는 시스템으로 구성될 수 있다. 예를 들어, oneM2M 보안 컴포넌트는 이종 IoT 플랫폼의 클라이언트로부터 토큰 발급을 요청받는 수신부와 토큰 발급을 요청한 클라이언트에 대한 승인과 인가를 수행하는 승인부 나아가 발급받은 토큰을 이용한 자원 요청이 수신되는 경우 자원을 제공하도록 제어하는 제어부를 포함할 수 있다. 물론, 이러한 장치 또는 시스템은 도 1 내지 도 18에서 설명한 모든 내용을 포함할 수 있으며, 이러한 기능적 구성 수단을 통해 이종 IoT 플랫폼 서비스 간 보안 상호운용성 장치를 제공할 수 있다.

[0130] 이상에서 설명된 시스템 또는 장치는 하드웨어 구성요소, 소프트웨어 구성요소, 및/또는 하드웨어 구성요소 및 소프트웨어 구성요소의 조합으로 구현될 수 있다. 예를 들어, 실시예들에서 설명된 시스템, 장치 및 구성요소는, 예를 들어, 프로세서, 콘트롤러, ALU(arithmetic logic unit), 디지털 신호 프로세서(digital signal processor), 마이크로컴퓨터, FPA(field programmable array), PLU(programmable logic unit), 마이크로프로세서, 또는 명령(instruction)을 실행하고 응답할 수 있는 다른 어떠한 장치와 같이, 하나 이상의 범용 컴퓨터 또는 특수 목적 컴퓨터를 이용하여 구현될 수 있다. 처리 장치는 운영 체제(OS) 및 상기 운영 체제 상에서 수행되는 하나 이상의 소프트웨어 애플리케이션을 수행할 수 있다. 또한, 처리 장치는 소프트웨어의 실행에 응답하여, 데이터를 접근, 저장, 조작, 처리 및 생성할 수도 있다. 이해의 편의를 위하여, 처리 장치는 하나가 사용되는 것으로 설명된 경우도 있지만, 해당 기술분야에서 통상의 지식을 가진 자는, 처리 장치가 복수 개의 처리 요소(processing element) 및/또는 복수 유형의 처리 요소를 포함할 수 있음을 알 수 있다. 예를 들어, 처리 장치는 복수 개의 프로세서 또는 하나의 프로세서 및 하나의 콘트롤러를 포함할 수 있다. 또한, 병렬 프로세서(parallel processor)와 같은, 다른 처리 구성(processing configuration)도 가능하다.

[0131] 소프트웨어는 컴퓨터 프로그램(computer program), 코드(code), 명령(instruction), 또는 이들 중 하나 이상의 조합을 포함할 수 있으며, 원하는 대로 동작하도록 처리 장치를 구성하거나 독립적으로 또는 결합적으로(collectively) 처리 장치를 명령할 수 있다. 소프트웨어 및/또는 데이터는, 처리 장치에 의하여 해석되거나 처리 장치에 명령 또는 데이터를 제공하기 위하여, 어떤 유형의 기계, 구성요소(component), 물리적 장치, 가상 장치(virtual equipment), 컴퓨터 저장 매체 또는 장치, 또는 전송되는 신호 파(signal wave)에 영구적으로, 또는 일시적으로 구체화(embody)될 수 있다. 소프트웨어는 네트워크로 연결된 컴퓨터 시스템 상에 분산되어서, 분산된 방법으로 저장되거나 실행될 수도 있다. 소프트웨어 및 데이터는 하나 이상의 컴퓨터 판독 가능 기록 매체에 저장될 수 있다.

[0132] 실시예들에 따른 방법은 다양한 컴퓨터 수단을 통하여 수행될 수 있는 프로그램 명령 형태로 구현되어 컴퓨터 판독 가능 매체에 기록될 수 있다. 상기 컴퓨터 판독 가능 매체는 프로그램 명령, 데이터 파일, 데이터 구조 등을 단독으로 또는 조합하여 포함할 수 있다. 상기 매체에 기록되는 프로그램 명령은 실시예를 위하여 특별히 설계되고 구성된 것들이거나 컴퓨터 소프트웨어 당업자에게 공지되어 사용 가능한 것일 수도 있다. 컴퓨터 판독 가능 기록 매체의 예에는 하드 디스크, 플로피 디스크 및 자기 테이프와 같은 자기 매체(magnetic media), CD-ROM, DVD와 같은 광기록 매체(optical media), 플롭티컬 디스크(floptical disk)와 같은 자기-광 매체(magneto-optical media), 및 롬(ROM), 램(RAM), 플래시 메모리 등과 같은 프로그램 명령을 저장하고 수행하도록 특별히 구성된 하드웨어 장치가 포함된다. 프로그램 명령의 예에는 컴파일러에 의해 만들어지는 것과 같은 기계어 코드뿐만 아니라 인터프리터 등을 사용해서 컴퓨터에 의해서 실행될 수 있는 고급 언어 코드를 포함한다. 상기된 하드웨어 장치는 실시예의 동작을 수행하기 위해 하나 이상의 소프트웨어 모듈로서 작동하도록 구성될 수 있으며, 그 역도 마찬가지이다.

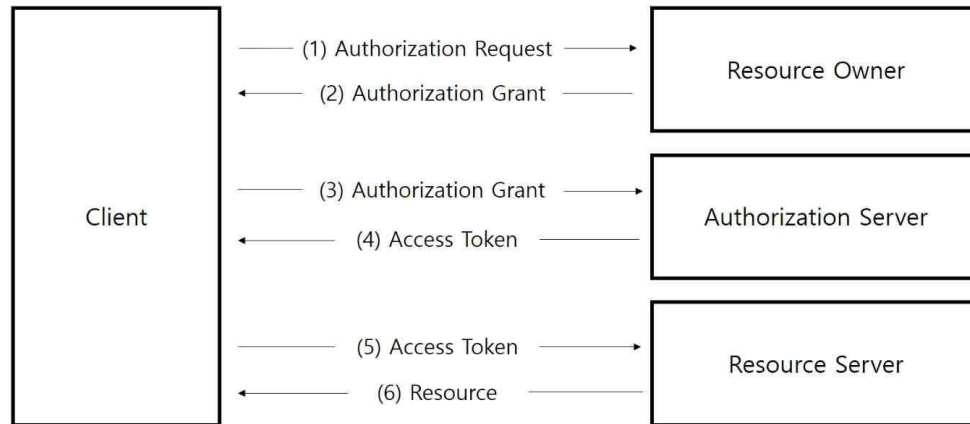
[0133] 이상과 같이 실시예들이 비록 한정된 실시예와 도면에 의해 설명되었으나, 해당 기술분야에서 통상의 지식을 가진 자라면 상기의 기재로부터 다양한 수정 및 변형이 가능하다. 예를 들어, 설명된 기술들이 설명된 방법과 다른 순서로 수행되거나, 및/또는 설명된 시스템, 구조, 장치, 회로 등의 구성요소들이 설명된 방법과 다른 형태로 결합 또는 조합되거나, 다른 구성요소 또는 균등물에 의하여 대치되거나 치환되더라도 적절한 결과가 달성될 수 있다.

[0134] 그러므로, 다른 구현들, 다른 실시예들 및 특허청구범위와 균등한 것들도 후술하는 특허청구범위의 범위에 속한

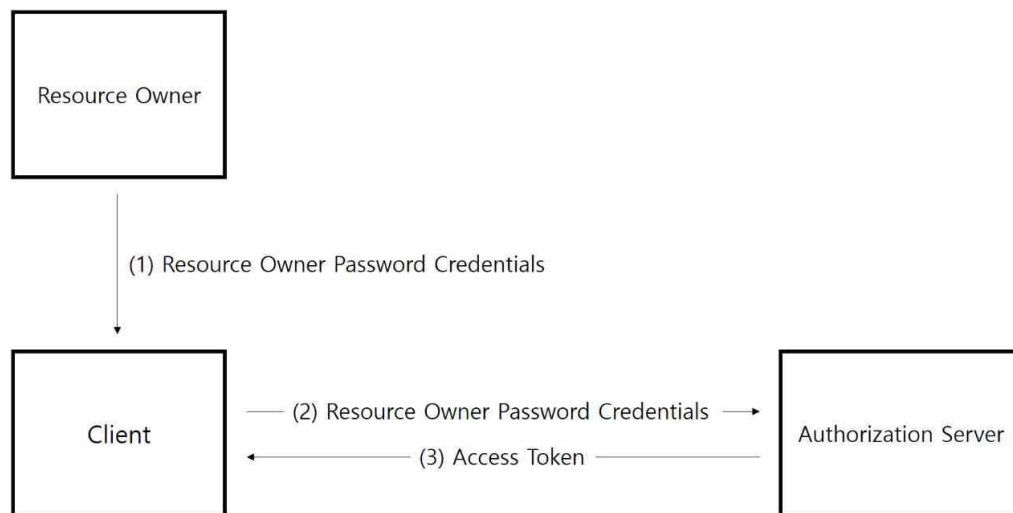
다.

도면

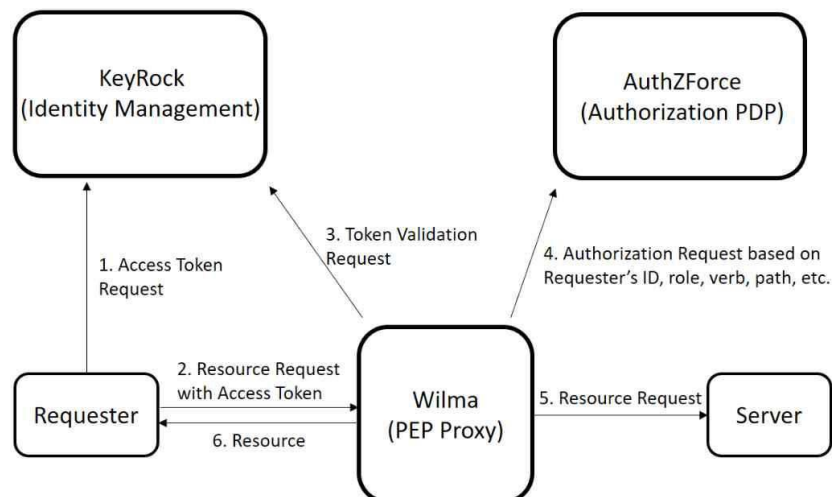
도면1



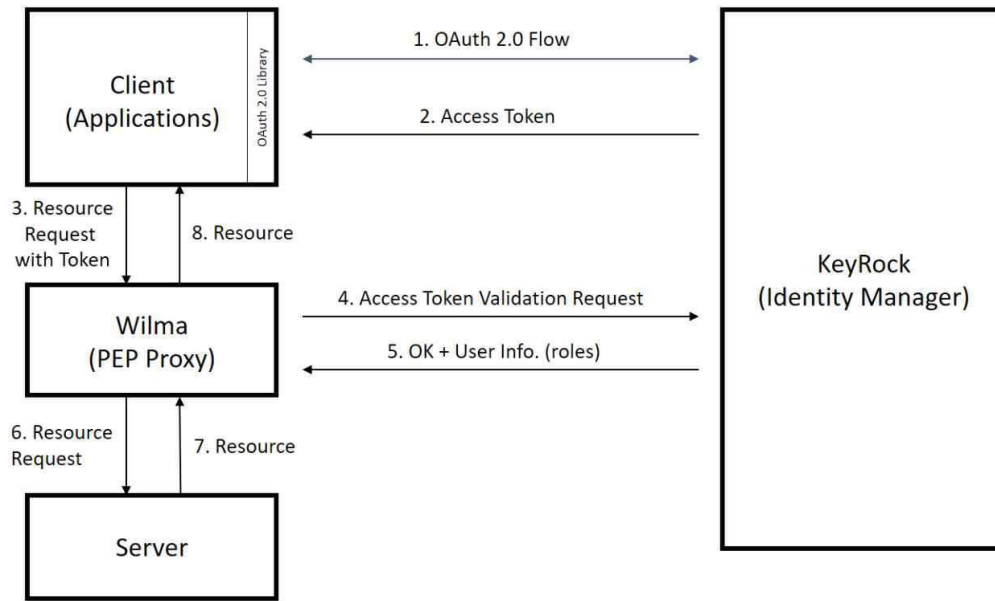
도면2



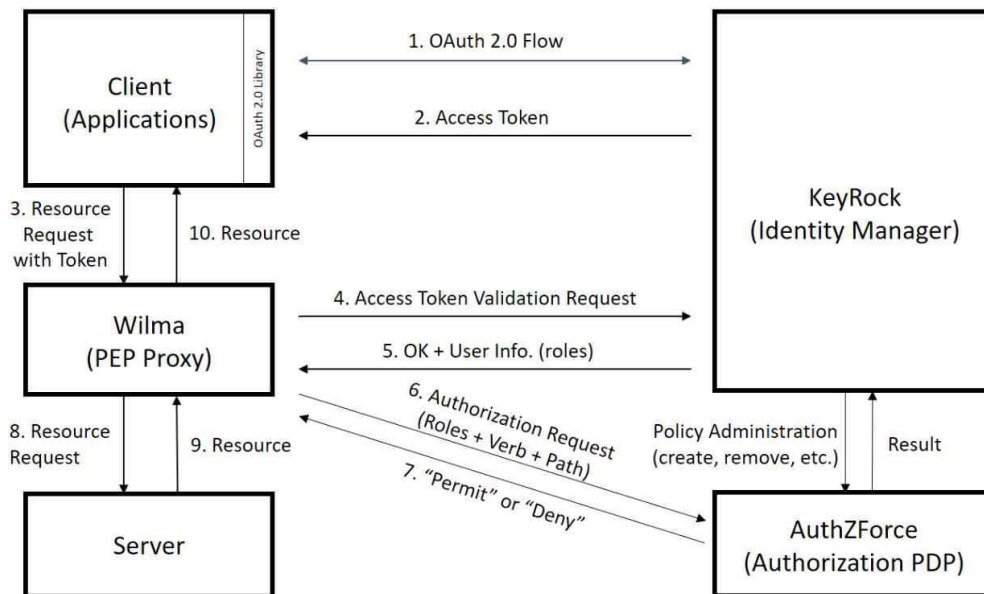
도면3



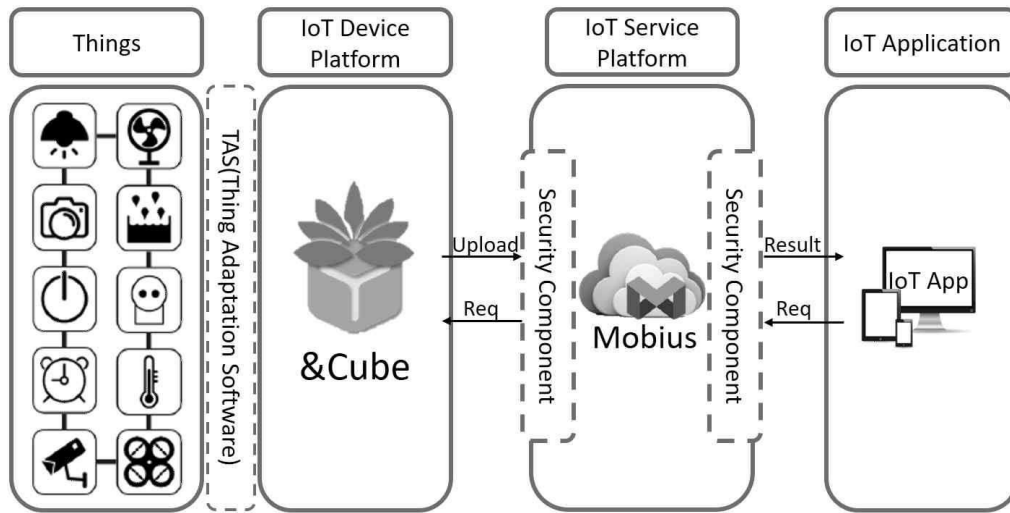
도면4



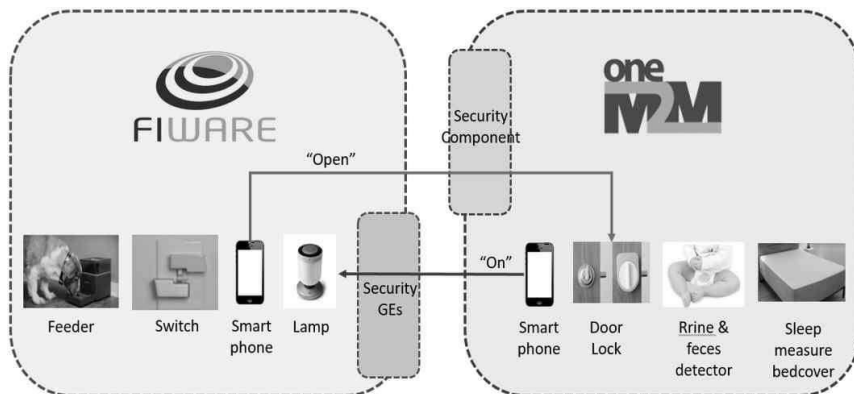
도면5



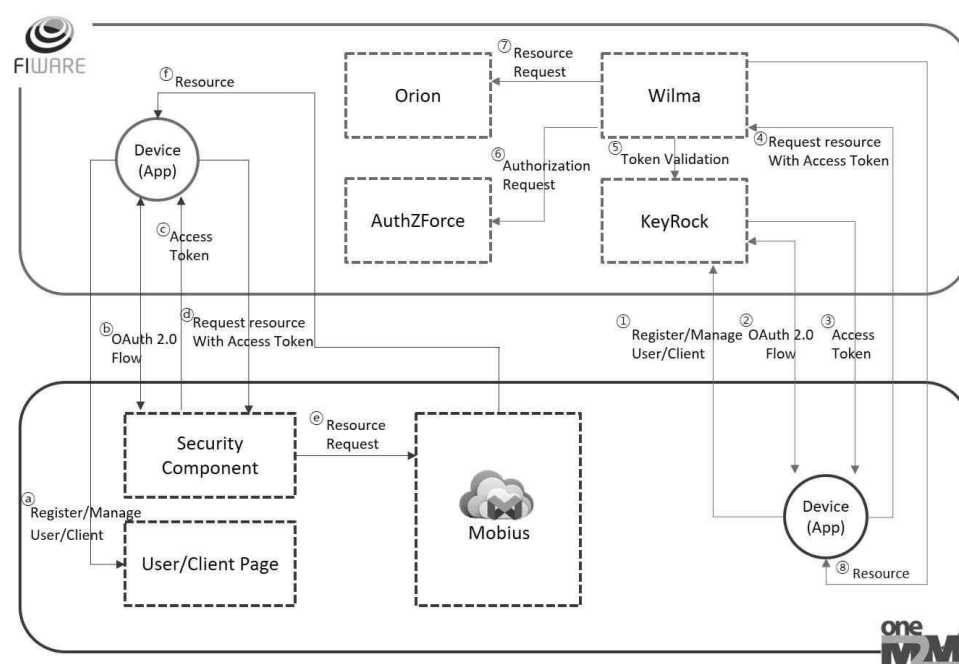
도면6



도면7



도면8



도면9

```
weepl@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": "TBmt3X42kTg6meQ84WBF9EX576C0cY",
  "expires_in": 3600,
  "refresh_token": "RFibMetRdJnayB4vhl8ILVA2fZpvKg",
  "token_type": "Bearer"
}
```

도면10

```
weepl@ubuntu:~/fiware/fiware-pep-proxy$ curl -H "X-Auth-Token: TBmt3X42
kTg6meQ84WBF9EX576C0cY" http://localhost:1706/v2/entities/My_Room
{"id": "My_Room", "type": "Room", "pressure": {"type": "Number", "value": 720, "
metadata": {}}, "temperature": {"type": "Number", "value": 23, "metadata": {}}}
```

도면11

```
weepl@ubuntu:~/fiware/fiware-pep-proxy$ curl -H "X-Auth-Token: 123456789"
http://localhost:1706/v2/entities/My_Room
User token not authorizedweepl@ubuntu:~/fiware/fiware-pep-proxy$
```

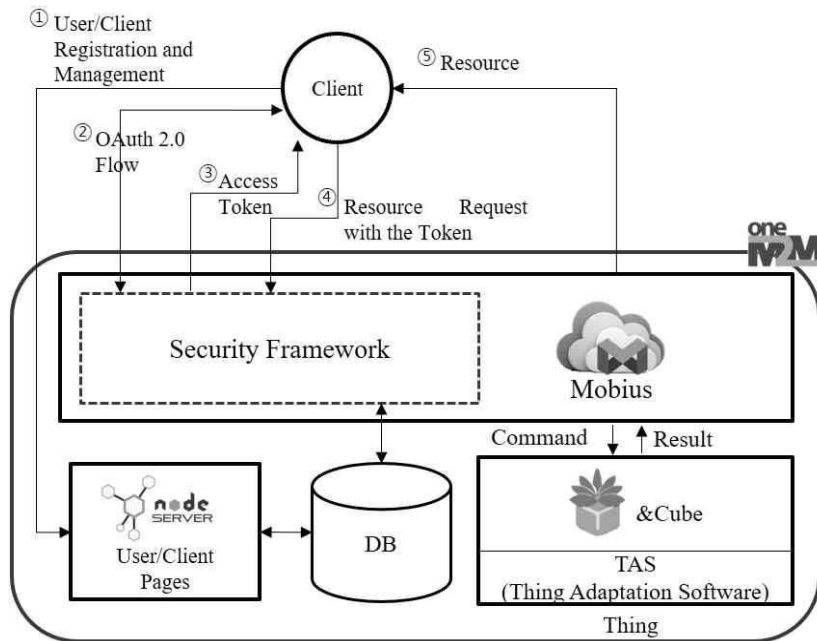
도면12

```
weepl@ubuntu:~/fiware/fiware-pep-proxy$ curl -H "X-Auth-Token: TBmt3X42kTg
6meQ84WBF9EX576C0cY" http://localhost:1706/v2/entities/Other_Room
User token not authorizedweepl@ubuntu:~/fiware/fiware-pep-proxy$
```

도면13

```
2017-09-14 15:44:50.207 - INFO: ION-Client - Token in cache, checking timestamp...
2017-09-14 15:44:50.207 - INFO: AZF-Client - Checking auth with AZF...
2017-09-14 15:44:50.207 - INFO: AZF-Client - Checking authorization to roles [ 'provide
r',
  '133a7012193049050f96c8a74824a894',
  'c915244239274d988406fdaaed3aaad' ] to do GET on v2/entities/Other_Room and app b
16e8edcb2c046adaab5a9c5022e728c
2017-09-14 15:44:50.208 - INFO: AZF-Client - Checking auth with AZF...
2017-09-14 15:44:50.220 - ERROR: Root - User access-token not authorized: User not aut
horized in AZF for the given action and resource
```

도면14



도면15

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

도면16

```
{
  "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."
}
```

도면17

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<m2m:cb xmlns:m2m="http://www.onem2m.org/xml/protocols"
  <ty>5</ty>
  <ct>20170408T124426</ct>
  <ri>/mobius-yt</ri>
  <lt>20170408T124426</lt>
  <lbl>mobius-yt</lbl>
  <dst>1</dst>
  <csi>/mobius-yt</csi>
  <srt>1 2 3 4 10 16 23 24 25 26</srt>
  <poa>http://192.168.0.12:7579</poa>
</m2m:cb>
```

도면18

