KIPS Tr. Comp. and Comm. Sys.
Vol.7, No.3 pp.81~90 pISSN: 2287-5891

https://doi.org/10.3745/KTCCS.2018.7.3.81

Interoperable Security Framework for Heterogeneous IoT Platforms

Se-Ra Oh† · Young-Gab Kim††

ABSTRACT

Due to the dramatic advancement of IoT (Internet of Things), it is expected that tens of billions of IoT devices will be connected by the year 2024. Furthermore, as IoT technologies evolves, the security management in IoT platforms has become a critical issue. For example, there are interworking problems between heterogeneous IoT platforms caused by differences from communication protocols, security policies, etc. in each platform. In addition, unsecured interworking can cause major security issues including the information leakage. In this paper, in order to solve these problems, a security interworking architecture is proposed and implemented in both FIWARE and oneM2M, which are representative IoT platforms. First, the security architecture of FIWARE is analyzed and implemented, and then the security framework based on OAuth 2.0 is developed on Mobius platform. Finally, in order to validate the proposed security interworking architecture, a LED (Light-Emitting Diode) example, where the LED is controlled by only authorized users, is developed. The proposed architecture can be extended to the diverse IoT platforms and devices.

Keywords : IoT Security, Security Framework, oneM2M, FIWARE, OAuth, Authentication, Authorization

이종 사물인터넷 플랫폼 간 보안 상호운용을 위한 프레임워크

오 세 라† · 김 영 갑††

요 약

IoT(Internet of Things)의 급격한 발달로 인하여 2024년까지 수백억 개의 IoT 디바이스가 만들어질 것으로 예측되고 있으며, 그러한 IoT 디바이스들에 영향을 미칠 수 있는 IoT 플랫폼의 중요성이 부각되고 있다. 현재 FIWARE, oneM2M, AllJoyn 등의 많은 IoT 플랫폼이 개발되고 있지만 이런 환경에서는 각 IoT 플랫폼의 통신 프로토콜, 보안 정책 등이 상이한 이종성(Heterogeneity)으로 인해 데이터를 연동하거나 보안 인터워킹을 수행하기가 어렵다. 보안이 고려되지 않은 인터워킹은 각종 개인, 기업 정보의 유출 등 심각한 문제를 야기할 수 있다. 이러한 문제를 해결하기 위해, 본 논문에서는 IoT 플랫폼 중에서도 대표적인 IoT 플랫폼인 FIWARE와 oneM2M을 대상으로 보안 인터워킹 구조를 제안하고 구현하였다. 본 논문에서는 해당 보안 인터워킹 구조에서 사용하는 FIWARE의 보안 아키텍처를 분석하고 구현하여 시사점을 도출하고, 현재 공식적인 보안 컴포넌트가 존재하지 않는 oneM2M 플랫폼에 OAuth 2.0 기반의 보안 프레임워크를 개발하였다. 또한, 본 논문에서 제안한 방법을 LED(Light-Emitting Diode) 예제로 개발하여 oneM2M 플랫폼과 FIWARE 플랫폼 간의 인증 및 인가 인터워킹을 수행하였다. 구현된 LED 예제는 인가 받은 사용자에게만 제어될 수 있도록 만들어졌으며, 향후에는 LED 이외의 스마트 홈의 CCTV, 도어 락(Door Lock)과 같이 다양한 디바이스 및 다양한 IoT 플랫폼(예를 들어, Watson IoT, IoTivity, AllJoyn 등)에 적용이 필요하다.

키워드 : 사물인터넷 보안, 보안 프레임워크, oneM2M, FIWARE, OAuth, 인증, 인가

1. 서 론

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(International Data Corporation) 등의 기업들은 IoT를 미래의 유망한 기술로 꼽았으며[1], IDC는 IoT 시장 규모가 2015년부터 2020년까지 연평균 15.6%만큼 성장할 것으로 전망했다[2]. 이에 구글과 아마존, 삼성, 시스코 등의 세계적 기업들이 IoT 시장을 선점하기 위해서 IoT와 관련된 기술이나 표준, 디바이스 등을 개발하고 있으며, 그 중에서도 IoT 플랫폼은 특히 중요하다. 그 이유는 IoT 플랫폼이 IoT 디바이스들의 기본적인 기능들을 제공해줌으로써 향후 만들어질 것으로 예측되는 500억개 이상의 IoT 디바이스들에 직·간접적으로 영향을 미칠 수 있기 때문이다[3]. 지금까지 FIWARE (Future Internet ware), oneM2M, AllJoyn, IoTivity와 같은

※ 이 연구는 2017년도 정부(과학기술정보통신부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.2017-0-00756, IoT 이종 식별체계 상호연동 및 관리체계 기술 개발).
※ 이 논문은 2017년도 한국정보처리학회 추계학술발표대회에서 "사물인터넷 보안 인터워킹에 관한 연구"의 제목으로 발표된 논문을 확장한 것임.
† 준 회 원 : 세종대학교 정보보호학과 석·박사통합과정
†† 정 회 원 : 세종대학교 정보보호학과 부교수
Manuscript Received : December 20, 2017
First Revision : February 5, 2018
Accepted : February 9, 2018
* Corresponding Author : Young-Gab Kim(alwaysgabi@sejong.ac.kr)

IoT 플랫폼들이 다수 만들어졌지만 각 IoT 플랫폼의 기반 기술, 통신 프로토콜, 보안 정책 등이 상이하여 이종 플랫폼 간에 원하는 정보나 서비스를 교환하거나 이용하기 어려운 상황이다. 따라서 각 IoT 플랫폼의 유용한 정보나 서비스를 서로 공유하기 위해 인터워킹 연구가 진행 중이다. 대다수의 IoT 개발 기관이 인터워킹 연구에 적극적인 것은 아니지만, oneM2M은 지금까지 여러 IoT 플랫폼들을 대상으로 인터워킹 연구를 수행하였고, 상당한 성과를 보여주고 있다[4]. 그러나 진행 중인 이종 IoT 플랫폼 간의 인터워킹 작업은 초기 단계로, 기초적인 데이터 교환부터 개발 중이기 때문에 인터워킹 연구에 적극적인 oneM2M의 경우에도 아직까지 보안에 관한 논의가 부족하다. 이러한 상황에서는 IoT 플랫폼 간의 인터워킹이 수행되어도 IoT 플랫폼 사이에서 전달되는 개인정보나 기업 정보들의 유출 등, 크고 작은 보안 문제들을 야기할 수 있다.

따라서 본 논문에서는 이종 IoT 플랫폼 간 보안 상호운용을 위한 방법을 제안하고자, 대표적인 IoT 플랫폼인 FIWARE와 oneM2M을 중심으로 IoT 플랫폼 간의 보안 인터워킹에 대해 연구하고 구체적인 FIWARE-oneM2M 보안 인터워킹 구조를 제안한다. FIWARE와 oneM2M은 대표적인 IoT 플랫폼이며 이미 Wise-IoT(Wolrdwide Interobability for Semantics IoT)의 일환으로 인터워킹 연구를 수행한 사례가 존재하기도 하지만 그 과정에서 보안이 고려되지는 않았다. FIWARE와 oneM2M 간에 보안 인터워킹이 수행되기 위해서는 각 플랫폼에 보안 컴포넌트들이 필요하지만 FIWARE의 경우에만 보안 컴포넌트가 존재하고 oneM2M은 제공되지 않고 있다. 본 연구에서는 FIWARE-oneM2M 보안 인터워킹 구조에서 핵심 보안 기능인 인증과 인가를 수행할 수 있는 OAuth 2.0 기반의 보안 프레임워크를 제안 및 구현하며, 이는 향후 oneM2M을 제외한 다른 IoT 플랫폼을 대상으로도 적용될 수 있을 것이다. 본 연구에서는 더 나아가, 보안 인터워킹 구조와 보안 프레임워크의 동작을 검증하기 위해서 LED (Light- Emitting Diode) 예제를 개발한다. LED 예제는 정당한 권한을 가진 유저만 oneM2M의 LED를 제어할 수 있도록 만들어졌으며, 향후에는 LED 외에 CCTV나 도어 락(Door Lock) 같이, 다양한 사물들을 대상으로 적용할 수 있을 것이다. 또한 본 연구에서 제안하는 보안 인터워킹 구조와 보안 프레임워크는 향후 타 IoT 플랫폼에 적용 가능하다.

본 논문의 구성은 다음과 같다. 2장에서는 OAuth 2.0과 FIWARE 보안 아키텍처, IoT 보안 및 인터워킹과 관련된 연구들에 대해 분석하며, 3장에서는 oneM2M의 보안 프레임워크 및 FIWARE와 oneM2M 간의 보안 인터워킹 구조를 제안한다. 4장에서는 2장에서 서술한 FIWARE 보안 아키텍처와 3장에서 제안한 보안 프레임워크를 기반으로 oneM2M 보안 컴포넌트를 구현하며, 이를 검증하기 위한 LED 예제에 대해 기술한다. 5장에서는 보안 인터워킹 정책과 향후 연구에 대해서 논의하고, 마지막 6장에서 결론을 서술한다.

2. 배경지식 및 관련 연구

본 장의 2.1절에서는 FIWARE의 보안 아키텍처와 본 논문에서 제안하는 보안 프레임워크를 이해하기 위해 중요한 OAuth 2.0의 핵심 내용을 기술하고, 2.2절에서는 FIWARE의 보안 아키텍처를 분석한다. 마지막으로 2.3절에서는 관련 연구에 대해 서술한다.

2.1 OAuth 2.0 프레임워크

보안 인터워킹에서는 상대방이 누구인지 확인하는 과정인 인증과 확인된 상대에게 권한을 부여하는 인가 과정이 중요하며, OAuth 2.0은 이런 경우에 사용할 수 있는 인증, 인가 프레임워크 표준[5]이다. FIWARE의 보안 아키텍처나 본 연구에서 제안하는 보안 프레임워크 모두 OAuth 2.0을 사용하기 때문에 본 절에서 OAuth 2.0의 핵심적인 내용들을 기술한다.

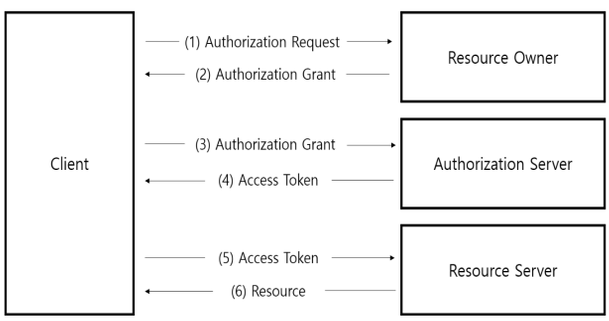

Fig. 1. General Flow of OAuth [5]

일반적으로 OAuth 2.0은 Fig. 1과 같이 동작하며, 클라이언트, 자원 소유자, 인가 서버, 자원 서버 네 가지 역할이 존재한다. 클라이언트는 REST API(Representational State Transfer Application Programming Interface) 등의 자원을 이용하기 위해서 자원 소유자에게 자원을 사용하기 위한 허가(grant)를 받은 뒤, 인가 서버에서 자원 소유자의 허가를 검증 받고 토큰을 발급 받아서 자원을 요청하게 된다. Fig. 1의 (2)과정은 authorization code, implicit, resource owner password credentials, client credentials 네 가지 타입으로 수행될 수 있다. 표준 문서에서는 클라이언트를 클라이언트의 신뢰도에 따라서 높은 신뢰도의 confidentials 타입과 그렇지 않은 public 타입으로 나누는데, authorization code와 implicit 방식은 클라이언트가 public 타입일 때 일반적으로 사용하며, 나머지는 confidentials 타입일 때 사용한다. 본 연구에서 사용한 방식은 resource owner password credentials 방식이다. 이 방식은 자원 소유자의 정보인 username과 password만 알면 다른 추가적인 과정 없이 곧바로 토큰을 발급받을 수 있지만, 자원 소유자의 정보는 기본적으로 노출되어서는 안 되므로 클라이언트와의 신뢰도가 높은 일부 상황에서만 활용해야 하고, 실제 환경에서는 TLS(Transport Layer Security) 등의 암호화 기술과 사용되어야 한다. 하지만 resource owner password credentials 방식은 토큰을 발급받는 과정 자체를

간단히 자동화할 수 있기 때문에 사람의 개입 없이 동작해야 하는 confidentials 타입의 IoT 디바이스에서 사용하기 적합하다.

2.2 FIWARE 보안 아키텍처

FIWARE의 보안 아키텍처[6]는 KeyRock, AuthZForce, Wilma라고 불리는 세 가지 GE(Generic Enabler)로 이루어져 있으며 FIWARE의 보안 컴포넌트로써 서로 협업한다. 동작 개요는 Fig. 2와 같다.

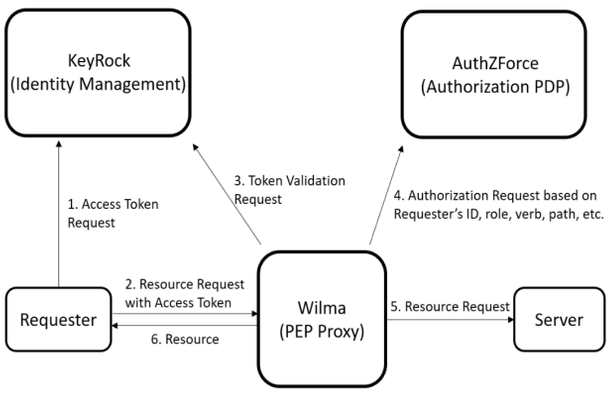

Fig. 2. Security Architecture of FIWARE

가장 먼저 요청자(Requester)는 KeyRock에 인증 정보를 전송하여 인증을 받은 뒤, 액세스 토큰을 발급 받아 Wilma에게 원하는 자원을 요청한다. 이후 Wilma는 해당 토큰을 검증하여 요청자 정보를 KeyRock에게 전달 받아 AuthZForce에 전달하여 권한 부여를 요청하고, AuthZForce의 응답이 “permit”인 경우, Wilma가 요청자 대신 서버에 자원을 요청하고 결과를 요청자에게 전달하게 된다.

2.3 관련 연구

IoT 보안과 관련된 연구로 오세라 외[1]는 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. 황인태 외[7]는 ITU-T (International Telecommunication Union Telecommunication Standardization Sector), ETSI(European Telecommunications Standards Institute) 등의 표준 기구 별 IoT 보안 표준을 조사하여 해당 표준을 여섯 가지 범주(기밀성, 무결성, 가용성, 부인방지, 인증, 인가) 중 하나로 분류하였다. 표준들은 주로 기밀성, 무결성, 인증, 인가를 다루고 있었으며 가용성과 부인방지에 대한 연구는 비교적 적은 것으로 분석되었다.

IoT 인터워킹과 관련된 연구로는 김재호 외[8]가 OIC(Open Interconnect Consortium), AllJoyn, LWM2M(Lightweight M2M)의 대표적인 IoT 표준기술들을 간단히 기술하고, oneM2M과의 연동 기술과 실제 구현된 사례를 소개하였다. 또한 이동규 외[9]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 제안했다.

IoT에 OAuth를 적용한 연구들도 존재한다. 김경한 외[10]는 ACE(Authentication and Authorization for Constrained Environments) 워킹 그룹에서 진행 중인 OAuth 2.0 기반의 ACE 보안 프레임워크의 한계점을 극복하기 위해서 CoAP 및 DTLS(Datagram Transport Layer Security) 기반의 경량 인증 프레임워크를 개발하였다. 또한 Sciancalepore 외[11]는 IoT의 제한적(constrained) 환경과 부족한 상호운용성을 고려하여 가볍고 유연한 OAuth 기반의 인증, 인가 프레임워크인 OAuth-IoT를 제안하였으며 Jung 외[12]는 IoT 환경에서 OAuth를 적용하기 위한 방안과 개인 OAuth 인가 서버 개발에 관련한 연구를 수행하였다. 그리고 Solapurkar[13]와 Fernandez 외[14]도 각각 IoT 클라우드 환경에서의 안전한 보건 서비스와 IoT 환경의 접근 통제를 위해 OAuth 기반 보안 아키텍처를 제안하였다.

하지만 앞서 언급된 IoT 및 OAuth 관련 연구들은 IoT 환경에서 활용할 수 있는 OAuth 기반의 보안 아키텍처나 프레임워크를 개발하는 것만을 목적으로 하여, 실제 IoT 플랫폼을 대상으로 보안 프레임워크를 적용하거나 IoT 플랫폼 간의 보안 인터워킹을 수행하는 것에 대해서는 고려하지 않았다. 이에 반해 본 연구는 국제적인 IoT 플랫폼인 FIWARE와 oneM2M을 대상으로 인증과 인가에 중점을 맞춘 구체적인 보안 인터워킹 구조와 보안 프레임워크를 제안한다. 마지막으로, 윤재석 외[15]은 IPE(Interworking Proxy Entity)를 통해 oneM2M 기반의 IoT 시스템과 기존의 IoT 상품이 인터워킹할 수 있는 방법을 연구했으며 보안을 고려하여 OAuth 2.0을 적용했다. 하지만 해당 연구의 방식은 토큰을 발급받는 과정에서 로그인을 해야 하거나 인가 코드를 입력해야 하기 때문에 사람이 개입할 수 있는 일부 시나리오에서만 활용할 수 있는 한계가 존재하였다. IoT 환경의 디바이스들은 자동화된 경우가 많아서 사람의 개입이 필요하게 되면 활용성이 제한되므로 보안 과정의 자동화가 필요하다. 본 연구에서는 이러한 한계점을 극복하기 위해서 토큰을 발급받을 때 resource owner password credentials 방식을 사용하였다. 이 방식은 추가적인 인증 과정 없이, 자원 소유자의 정보만 확실하면 바로 토큰을 발급받을 수 있기 때문에 토큰 발급이 간단하고 발급 과정의 자동화 또한 간편하지만 자원 소유자의 정보가 유출되면 타인도 해당 정보로 토큰을 발급받을 수 있게 된다. 따라서 신뢰할 수 있는 디바이스, 클라이언트에서만 해당 방식을 사용해야 한다.

IoT 플랫폼 간 인터워킹 연구는 oneM2M에서 활발히 진행 중에 있다. oneM2M은 oneM2M 디바이스와 oneM2M이 아닌 디바이스 간의 시맨틱 인터워킹을 위해 기반 온톨로지(Ontology)를 기술한 TS-0012 “Base Ontology”[16]부터 TS-0014 “LWM2M Interworking”[17], TS-0021 “oneM2M and AllJoyn Interworking”[18], TS-0024 “OIC Interworking”[19], TS-0026 “3GPP Interworking”[20], TS-0030 “Generic

Interworking"[21], TS-0033 "Proximal IoT Interworking" [22], TR-0027 "DDS usage in oneM2M system"[23], TR-0042 "WoT Interworking"[24], TR-0043 "Modbus Interworking" [25] 등의 인터워킹 관련 표준들을 제정하고 있다.

3. oneM2M 보안 프레임워크와 FIWARE-oneM2M 보안 인터워킹 구조

본 장의 3.1절에서는 보안 인터워킹의 핵심인 인증과 인가를 제공할 수 있는 oneM2M 보안 프레임워크를 제안하고, 3.2절에서는 FIWARE와 oneM2M 간의 보안 인터워킹 구조를 제안한다.

3.1 oneM2M 보안 프레임워크

FIWARE와 달리, oneM2M은 보안 아키텍처에 대한 논의만 이루어지고 있고, 아직 공식적으로 구현되어 제공되는 보안 컴포넌트가 존재하지 않는다. 본 논문의 주목적은 보안 인터워킹의 연구로, IoT 플랫폼 간의 인터워킹 시나리오에서 인증과 인가를 수행하는 것이 핵심이다. 따라서 본 논문에서는 다양한 보안적 사양을 요구하는 oneM2M의 보안 아키텍처를 분석하거나 구현하지 않고 인증과 인가를 제공할 수 있는 OAuth 2.0 기반 보안 프레임워크를 설계하여, KETI(Korea Electronics Technology Institute)에서 개발한 Mobius 플랫폼에 oneM2M 보안 프레임워크를 구현하고자 한다.

oneM2M 보안 프레임워크가 정상적으로 동작하기 위해서는 다음과 같은 기본적인 요구사항들을 만족해야 한다.

- 토큰을 발급받을 때, 클라이언트 정보와 회원 정보가 필요하므로 회원 등록 페이지와 클라이언트 등록 페이지가 필요하다.
- 클라이언트 정보와 회원 정보를 저장하기 위한 데이터베이스가 필요하다.
- 민감한 정보들(클라이언트나 회원 정보 등)은 암호화가 되어야 한다.
- 서버(Mobius)에 대한 요청들은 보안 컴포넌트를 거쳐야 한다. 특히, oneM2M의 시스템을 제어하는 명령이나 중요 데이터, 개인 정보 등을 요청하는 REST API들은 반드시 보안 컴포넌트를 통해 요청을 인증, 인가 받은 뒤 사용되어야 한다.

Mobius는 디바이스의 등록이나 관리, 자원의 CRUD(Create, Read, Update, Delete) 등을 수행하기 위해서 REST API를 사용한다. 이때, 일부 보안적으로 유해하지 않은 REST API도 존재하지만 시스템이나 디바이스를 제어하거나 보안적으로 중요한 데이터들을 요청하고 조작하는 REST API들은 심각한 보안 피해를 야기할 수 있다. 그러므로 Mobius 자체를 포함하여 디바이스나 중요 데이터들을 안전하게 보호하려면 REST API의 사용을 제한할 필요가 있으며, 이를 위해 본 연구에서는 OAuth 2.0에 기반한 oneM2M 보안 프레임워크를 제안한다.

oneM2M 보안 프레임워크는 설정에 따라서 특정 REST API만 사용을 제한할 수 있지만, 본 연구에서는 기본적으로 어떤 REST API든지 Mobius의 자원(http://localhost:7579/*)에 접근하면 토큰을 검증받도록 설정하였다. 이때 중요한 것은 Mobius의 실행 파일 중 일부가 처음 실행될 때 REST API를 사용한다는 사실이다. 기존에 구현되어 있는 파일들(pxy_coap.js, pxy_mqtt.js 등)은 토큰을 발급받은 뒤 REST API를 사용하도록 만들어져 있지 않기 때문에 Fig. 3처럼 토큰 발급과 사용에 대한 적절한 수정이 필요하다.

```
function coap_message_handler(request, response) {
    var headers = {};
    headers['X-M2M-TY'] = '';
```

```
function coap_message_handler(request, response) {
    reqAT.requestAccessToken(function(accessToken) {
      if (accessToken == -1) {
        console.log('Access Token is not issued!');
      }
      else {
        var headers = {};
        var authToken = 'Bearer ' + accessToken;
        console.log(authToken);
        headers['X-M2M-TY'] = '';
        headers['Authorization'] = authToken;
```

Fig. 3. Modification of coap_message_handler to Request and Use an Access Token

Fig. 3은 기존 pxy_coap.js에 존재하는 coap_message_handler 함수를 재작성한 그림이다. 그림에서 볼 수 있듯이 기존의 코드와는 달리 수정된 코드에서는 Mobius에 REST API를 요청하기 전에 oneM2M 보안 프레임워크에서 토큰을 발급받고, 이를 헤더("Authorization")에 추가하는 과정이 추가되었다.

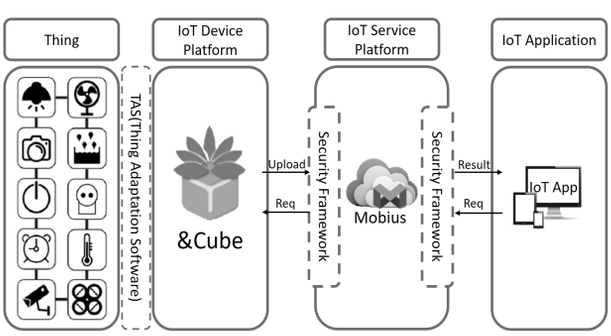

Fig. 4. Mobius with Security Framework

Fig. 4는 본 연구에서 구현하고자 하는 oneM2M 보안 프레임워크가 적용된 Mobius 서비스를 도식화한 것이다. Fig. 4에서 oneM2M 보안 프레임워크는 IoT 어플리케이션이나 디

바이스 플랫폼의 요청을 검증하고, 성공적으로 토큰을 검증한 경우에만 요청을 Mobius에 리다이렉트(redirect)하여 처리될 수 있도록 한다. 이에 대한 자세한 동작은 Fig. 5와 같다.

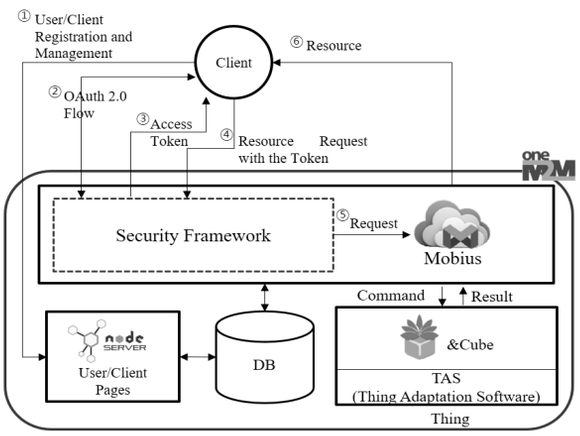

Fig. 5. Detailed Flow of oneM2M Security Framework

Fig. 5에서 볼 수 있듯이 외부 클라이언트가 oneM2M의 자원에 접근하기 위해서는 먼저 회원과 클라이언트를 등록하여 OAuth 2.0 Flow를 위한 정보를 얻어야 한다. 이를 위해 본 연구에서는 Fig. 6과 같이 node.js 기반의 로그인, 회원 가입, 클라이언트 등록 페이지를 만들었다.

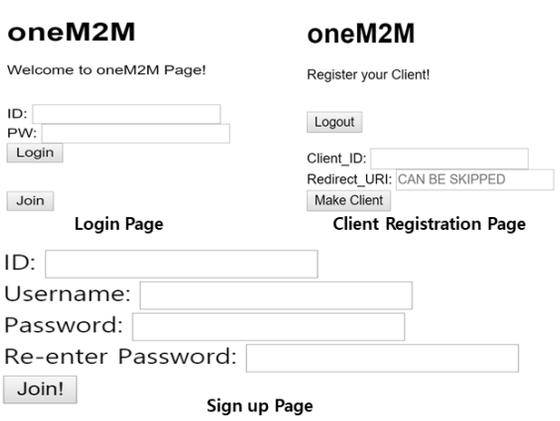

Fig. 6. Login, Client Registration, and Sign Up Pages

외부 클라이언트는 회원 등록 후 로그인하여 클라이언트를 등록하면 username, password, Client ID, Client Secret정보를 얻을 수 있으며, 이를 이용하여 oneM2M 보안 프레임워크와 OAuth 2.0 Flow를 수행한 뒤 토큰을 발급받을 수 있다. 그리고 클라이언트는 해당 토큰으로 oneM2M의 자원들을 이용할 수 있게 된다.

3.2 FIWARE-oneM2M 보안 인터워킹

2015년 NEC(Nippon Electric Company), 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. 문제는 인터워킹 시나리오에서 보안이 고려되지 않은 점이다. 적절한 보안 인터워킹이 수행되지 않으면 보안 공격의 영향이 인터워킹과 연관된 모든 플랫폼들에 미칠 수 있다. Fig. 7은 본 논문에서 제안하는 FIWARE와 oneM2M 플랫폼 간의 보안 인터워킹 시나리오를 나타낸다.

Fig. 7의 시나리오에서는 FIWARE나 oneM2M의 자원에 접근하기 위해 각 플랫폼의 보안 컴포넌트를 거쳐서 인증과 인가를 받아야 한다. 예를 들어, FIWARE 스마트 폰이 oneM2M 도어 락을 열기 위해서는 oneM2M의 보안 컴포넌트에서 인증과 인가를 받아야 하고, oneM2M도 마찬가지로 FIWARE의 램프를 켜기 위해서는 FIWARE의 보안 컴포넌트들(GEs)에 의해 인증과 인가를 받아야 한다. 이때, 실제 보안 인터워킹 과정에서는 oneM2M의 공식 보안 컴포넌트가 존재하지 않기 때문에 본 논문에서 제안하는 보안 프레임워크를 구현한 것으로 대체하며, 과정을 더 자세히 나타내면 Fig. 8과 같다. oneM2M 디바이스가 FIWARE의 자원을 요청하려면 앞서 설명한 것처럼, 먼저 KeyRock에 가입하고 클라이언트를 등록한 뒤, OAuth 2.0을 통해 토큰을 발급 받아 Wilma에 자원을 요청해야 한다. KeyRock과 AuthZForce를 통해 토큰이 검증되고 모든 절차가 정상적으로 처리되면 Wilma는 요청한 결과를 디바이스에 반환해준다. 이와 같이, FIWARE 디바이스가 oneM2M의 자원을 이용하기 위해서는 oneM2M의 회원/클라이언트 등록 페이지에서 OAuth 2.0 관련 정보를 얻고 보안 컴포넌트에서 토큰을 발급 받아야 한다.

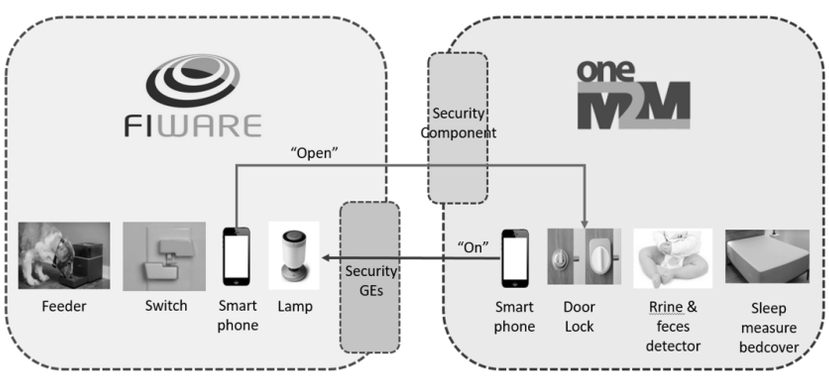

Fig. 7. Secure Interworking Scenario between FIWARE and oneM2M

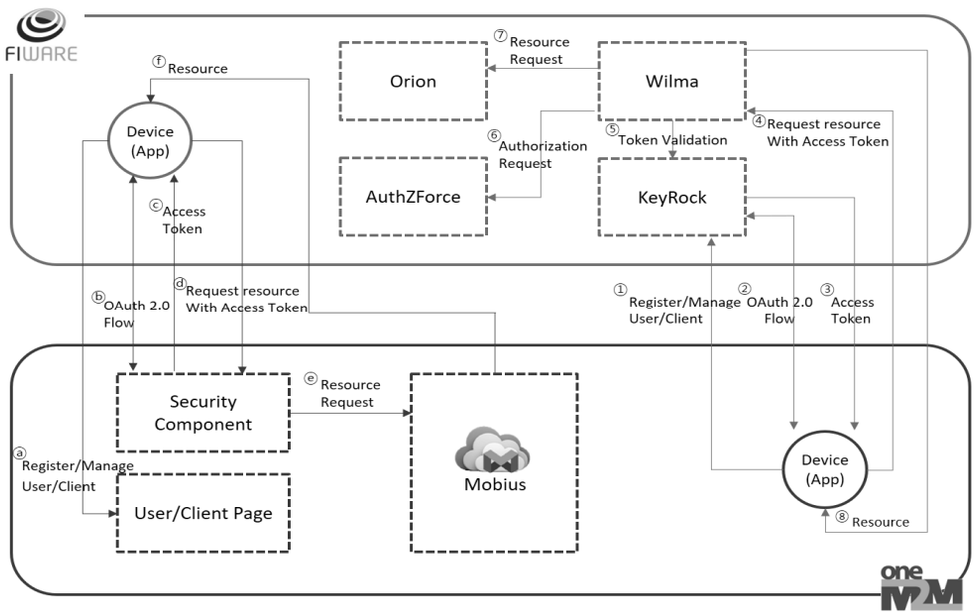

Fig. 8. Detailed Flow of FIWARE-oneM2M Secure Interworking

FIWARE 디바이스가 토큰을 발급 받으면, oneM2M을 기반의 디바이스들을 제어하거나 특정 데이터를 조작하고 요청하는 등 oneM2M의 자원을 이용할 수 있게 된다.

4. FIWARE-oneM2M의 보안 컴포넌트 구현

본 장은 FIWARE와 oneM2M의 보안 컴포넌트에 대한 구현을 중점으로 서술한다. 4.1절에서는 FIWARE 보안 컴포넌트의 구현에 대해 기술하고 시사점을 도출하며, 4.2절에서 oneM2M 보안 프레임워크를 구현한 내용에 관해 서술한다. 마지막 4.3절에서는 향후 다양한 사물을 대상으로 적용될 수 있는 LED 예제의 구현에 대하여 기술한다.

4.1 FIWARE 보안 컴포넌트 구현

본 절의 FIWARE 보안 컴포넌트들은 Fig. 2와 같이 동작한다. 최신 버전의 FIWARE 보안 GE들은 테스트가 충분하지 않을 수 있기 때문에, 보안 아키텍처를 구현한다면 다른 버전들보다 KeyRock 5.4.1, AuthZForce 5.4.1, Wilma 5.4.0을 이용하는 것이 안정적이다. 2017년 5월을 기준으로 KeyRock과 Wilma는 AuthZForce 5.4.1을 대상으로만 공식적으로 테스트가 되었기 때문에 5.4.1이 아닌 다른 버전에서는 의도하지 않은 문제가 발생할 수 있다. 본 연구에서는 AuthZForce 5.4.1이 Ubuntu 16.04를 지원하지 않으므로 7.0.0 버전을 사용하였는데, 별다른 버그나 의도하지 않은 문제가 발생하지는 않는 것을 확인하였다. Fig. 9는 보안 컴포넌트들을 모두 실행한 뒤, KeyRock에 보낸 토큰 발급 요청과 이후 결과를 보여준다. FIWARE에서의 토큰 발급 요청은 KeyRock의 토큰 엔드포인트(http://localhost:8000/oauth2/token)에 post 방식으로 보낸다. 이때 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, 2장에서 언급한대로 본 연구에서는 resource owner password grant 방식을 사용하였다. 또한, 클라이언트 id와 secret은 HTTP basic authentication에 따라 base-64로 인코딩되어야 하고 실제 상용 환경에서는 클라이언트 secret이나 사용자 password는 반드시 TLS 등으로 암호화되어야 한다.

```
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"
}
```

Fig. 9. Token Request and Response of FIWARE

Fig. 9에서 볼 수 있듯이, 토큰 발급 요청이 성공하면 토큰과 만료 기간 같은 관련 정보들이 KeyRock으로부터 전송된다. FIWARE의 토큰은 일반적인 bearer 방식이고 만료 시간은 한 시간으로 설정되며 토큰을 발급받은 이후에는 FIWARE의 자원을 요청할 수 있다. Fig. 10은 토큰을 사용하여 FIWARE에서 보호되고 있는 자원인 My_Room에 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":{}}}
```

Fig. 10. Result of Resource Access

토큰은 헤더의 X-Auth-Token에 넣고, Wilma (localhost:1706)에 원하는 자원의 URL(v2/entities/My_Room)을 보낸다. Wilma의 설정 파일(config.js)에는 토큰의 검증을 성공적으로 마치면 요청을 리다이렉트할 주소를 입력하도록 되어 있는데, 본 연구에서는 컨텍스트 정보를 관리해주는 Orion의 주소(localhost:1026)로 설정하였다. 따라서 Wilma는 토큰을 성공적으로 검증하면 Orion에 요청을 리다이렉트하여 My_Room의 정보를 검색하고, 결과는 요청자에게 다시 보내준다. 만약 토큰을 발급받은 유저에게 My_Room에 대한 권한이 없거나, 토큰이 만료되었거나, 정상적인 토큰이 아니라면 Wilma는 Fig. 11이나 Fig. 12처럼 접근을 불허한다.

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

Fig. 11. Invalid Access using Invalid Token

Fig. 11은 정상적인 토큰이 아닌 임의의 토큰(123456789)을 통해 자원을 요청했을 때의 결과이며 Fig. 12는 요청자가 자신에게 권한이 없는 자원에 접근했을 때의 결과이다.

```
weepi@ubuntu:~/fiware/fiware-pep-proxy$ curl -H "X-Auth-Token: TBmt3X42kTg
6meQ84WBF9EX576COcY" http://localhost:1706/v2/entities/Other Room
User token not authorizedweepi@ubuntu:~/fiware/fiware-pep-proxy$
```

Fig. 12. Invalid Access without Proper Rights

FIWARE는 과거 여러 차례 보안 컴포넌트를 개발했지만 KeyRock, Wilma, AuthZForce는 아직까지 폐기되지 않고 업데이트와 테스트가 진행되고 있다. 그러나 보안 인터워킹의 연구와 구현을 위해 실제 보안 컴포넌트들을 설치하고 테스트해본 결과, 아직 설정이 까다롭고 특정 버전끼리 의존성이 강한 문제가 있음을 확인하였다.

4.2 oneM2M 보안 프레임워크 구현

본 연구에서는 oneM2M 보안 프레임워크의 OAuth 2.0 서버 기능 구현을 위해 node.js 기반의 'oauth2-server' 라이브러리를 사용하였으며, 이를 Mobius Yellow Turtle 2.3.8에 적용하였다.

Fig. 13은 본 연구에서 Mobius의 메인 코드인 app.js에 작성한 OAuth 2.0 관련 코드이다. 앞서 언급한대로 Mobius의 REST API를 사용하기 위한 모든 URL(/mobius-yt/와 /mobius-yt/*)이 oneM2M 보안 프레임워크에 의해 차단되어 있고, 이는 토큰 엔드포인트인 /oauth/token을 통해 적절한 토큰을 발급받아야만 인가를 받아 접근할 수 있다.

```
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

app.oauth = oauthserver({
  model: require('./model.js'),
  grants: ['password'],
  debug: false
});

app.all('/oauth/token', app.oauth.grant());
app.all('/mobius-yt/', app.oauth.authorise());
app.all('/mobius-yt/*', app.oauth.authorise());
app.use(app.oauth.errorHandler());
```

Fig. 13. OAuth-related Code in app.js of Mobius

oneM2M 보안 프레임워크에서 토큰을 발급받기 위해서는 토큰 엔드포인트에 필요한 정보(grant_type, Client ID와 Client Secret, 자원 소유자의 username과 password)를 post 방식으로 요청을 보내야 한다. 토큰 발급이 성공하면 Fig. 14처럼 토큰 타입과 토큰, 만료 시간이 전송된다.

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

Fig. 14. Token Response of oneM2M Security Framework

토큰 타입과 만료 시간은 FIWARE와 마찬가지로 각각 Bearer와 1시간이며 토큰은 무작위한 문자열로 생성된다. 또한, 누군가가 보안 프레임워크가 동작하는 상태에서 토큰 없이 Mobius의 자원에 접근하거나 비정상 토큰을 사용한다면 Fig. 15 같은 결과가 반환된다. 코드 400은 요청자가 자원을 요청할 때 토큰이 요청에서 발견되지 않은 것을 의미하고, 코드 401은 요청자가 제시한 토큰의 검증이 실패한 경우를 나타낸다.

```
{
    "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."
}
```

Fig. 15. Errors occurred from oneM2M Security Framework

Fig. 15와 달리, 정상 토큰으로 Mobius에 요청하면 Fig. 16과 같이 요청자가 원한 결과를 반환해준다.

Fig. 16은 요청자가 oneM2M 보안 프레임워크의 인가를 받은 뒤, Mobius의 CSE(Common Service Entity) 정보를 얻은 결과를 보여주고 있다.

```
<?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>
    <cst>1</cst>
    <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>
```

Fig. 16. Successful Response from Mobius

4.3 LED를 활용한 보안 인터워킹 예제 구현

본 연구의 LED 예제는 OCEAN(Open allianCE for iot stANdard)의 LED 예제[26]와 Fig. 8에서 제안한 보안 인터워킹 구조를 기반으로 개발되었다. Fig. 17은 LED 예제의 구조이다.

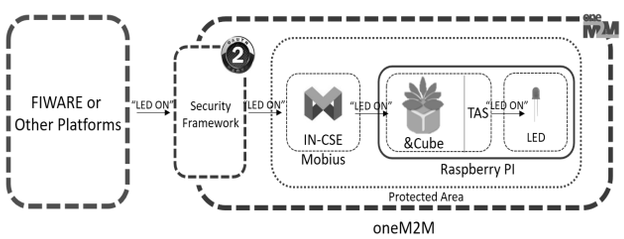

Fig. 17. Architecture of LED Example

Fig. 17의 LED 예제에서는 FIWARE나 기타 플랫폼이 oneM2M의 LED를 제어하기 위해, 반드시 본 연구에서 개발한 보안 프레임워크를 통하여 REST API의 접근을 위한 토큰부터 발급받아야 한다. 그 후, 발급받은 토큰은 Mobius에 LED 제어 요청을 보낼 때 헤더의 “Authorization” 항목에 “Bearer 토큰” 형태로 조합하여 보안 프레임워크에 전달해야 한다. 만약 토큰의 검증을 성공적으로 마치면, 보안 프레임워크는 실행 흐름을 Mobius에게 돌려주고, 이후에는 라즈베리 파이 내부의 &Cube(Thyme), TAS를 통해 LED 제어 명령이 전달되며, 토큰 검증이 실패하면 LED 제어 명령은 보안 프레임워크에 의해 차단된다.

Fig. 17에서 LED 제어를 위해 사용된 라즈베리 파이 관련 환경은 Table 1로 요약할 수 있다.

Table 1. Respberry Pi Environment of LED Example

Environment	Version
Respberry Pi	3 (model B)
Raspbian OS	8 (Jessie)
&Cube(Thyme)	1.7.7
Node.js	0.12.6

LED 예제의 구현을 위해서 한 가지 중요한 점은 라즈베리 파이에서 구동되는 Thyme가 처음 실행될 때 Mobius와 통신을 한다는 사실이다. 따라서 Fig. 3과 마찬가지로, Thyme의 코드도 Fig. 18과 같이 oneM2M 보안 프레임워크에서 토큰을 발급 받고, 이를 통해 REST API를 이용할 수 있도록 수정될 필요가 있다. Thyme의 경우 Mobius와의 통신을 위해 CoAP, MQTT, HTTP 프로토콜을 지원하는데, Fig. 18은 그 중에서 HTTP를 사용하여 Mobius와 통신하기 위해 http_adn.js 파일을 수정한 그림이다.

```
function http_request(path, method, ty, bodyString, callback) {
    reqAT.requestAccessToken(function(accessToken) {
      if (accessToken == -1) {
        console.log('Access Token is not issued!');
      }
      else {
        authToken = 'Bearer ' + accessToken;
        var options = {
            hostname: conf.cse.host,
            port: conf.cse.port,
            path: path,
            method: method,
            headers: {
                'X-M2M-RI': shortid.generate(),
                'Accept': 'application/' + conf.ae.bodytype,
                'X-M2M-Origin': conf.ae.id,
                  'Authorization': authToken
            }
        };
```

Fig. 18. Modification of http_request to Request and Use an Access Token

다음으로, LED 제어를 위한 요청 헤더의 목록을 자세히 살펴보면 Fig. 19와 같다.

```
Accept: application/json
X-M2M-RI: 12345
X-M2M-Origin: SOrigin
Content-Type: application/vnd.onem2m-res+json; ty=4
Authorization: Bearer
81d7004b73d87bc9a034386daff0c61569d0d2ed
```

Fig. 19. Header for Mobius LED Control

Fig. 19에서 가장 중요한 항목은 앞서 언급한 Authorization 항목이다. 요청자는 Authorization에 토큰을 삽입함으로써 보안 프레임워크의 인가를 받아 Mobius의 REST API를 이용할 수 있게 되고, 최종적으로 LED를 제어한다. Fig. 20은 LED 제어 요청 시 사용되는 바디 데이터를 보여준다.

```
{
  “m2m:cin”: {
    “con”: “on”,
    “rn”: “LED_On_01”
  }
}
```

Fig. 20. Body for Mobius LED Control

LED on 명령어를 나타내는 Fig. 20의 바디 데이터를 토큰이 포함된 Fig. 19의 헤더 정보들과 함께 post 방식으로 Mobius에 보내면 Fig. 21과 같은 결과가 반환되며, Fig. 22처럼 LED가 동작하게 된다.


```
{
    "m2m:cin": {
        "rn": "LED_On11255",
        "ty": 4,
        "pi": "ryHsiPj66x",
        "ri": "r1fM45gbjb",
        "ct": "20170921T083657",
        "et": "20190921T083657",
        "lt": "20170921T083657",
        "st": 109,
        "cs": 2,
        "con": "on",
        "cr": "SOrigin"
    }
}
```


Fig. 21. Created Container Instance

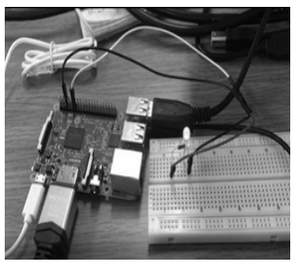

Fig. 22. Operated LED

과거 수차례 스마트 홈을 대상으로 권한이 없는 외부 비인가자가 스마트 홈 내부의 온도 조절기를 마음대로 제어하거나 CCTV 데이터에 접근하는 사례가 있었다. 하지만 본 연구의 보안 프레임워크는 적절한 권한 없이 디바이스를 제어하는 사례들을 방지할 수 있으며 이를 활용하여 타 플랫폼과의 보안 인터워킹을 수행할 수 있다.

5. 고 찰

앞서 언급하였듯이, IoT 플랫폼 간의 인터워킹 연구들이 진행되고 있지만 아직까지 보안에 관한 논의가 부족한 실정이다. 이에 본 논문은 대표적인 IoT 플랫폼인 FIWARE와 oneM2M을 대상으로 IoT 플랫폼 간의 보안 인터워킹에 대해 연구하였으며, 보안 인터워킹 구조를 제안하고 공식 보안 컴포넌트가 존재하지 않는 oneM2M에서 활용할 수 있는 OAuth 2.0 기반 보안 프레임워크를 개발하였다.

하지만 본 연구에서 개발한 보안 프레임워크는 토큰을 통한 기본적인 인가만 수행하였으며 세부적인 인가는 수행하지 않았다. 추후 본 연구의 보안 프레임워크는 단순 토큰의 검증으로만 인가를 수행하는 것이 아니라, 요청자의 나이, 지역, 접속한 시간, 사용하는 이메일 도메인 등을 통해서 세세한 인가를 수행하는 것이 필요하며 인증과 인가 외에도 복잡한 보안 인터워킹 시나리오에서 요구되는 다양한 보안 기능을 제공해야 한다. 또한 본 연구에서 토큰을 발급받기 위해 사용한 resource owner password credentials 방식은 신뢰도가 높은 클라이언트에서만 사용할 수 있는 한계점이 존재하기 때문에 다른 허가 방식으로도 IoT 환경에서 토큰 발급과 사용의 자동화를 단순화하고 경량화할 수 있을지 더 연구할 필요가 있다. 그리고 구체적인 보안 인터워킹 구조의 설계와 구현은 FIWARE와 oneM2M을 대상으로만 수행되었다. 이에 대해서는 향후 더 다양한 IoT 플랫폼을 대상으로 확장하여 일반적으로 사용할 수 있는 보안 인터워킹 구조에 대해서 연구해야 한다.

본 논문에서 제안한 보안 프레임워크의 토큰 발급과 활용에 대해서도 논의가 더 필요하다. PC 환경과는 달리 IoT 환경에서 토큰을 발급받을 때 중요한 차이점은 상황인지(context-aware) 특성의 여부이다. 기존의 PC 환경과 달리 IoT의 상황인지 특성을 이용하면 더 유연한 토큰 발급이 가능해질 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.

더 나아가, 보안 정책에 따라 토큰을 사용할 때 FIWARE 토큰을 oneM2M에서 사용하거나 oneM2M 토큰을 FIWARE에서 사용할 수 있도록 만들 수 있다. 다만, oneM2M 측에서 FIWARE 토큰을 받았을 때 해당 토큰이 FIWARE의 토큰임을 알아야 하고, FIWARE 측에 토큰의 검증을 요청할 수 있어야 하며 해당 토큰의 권한으로 특정 oneM2M 자원의 어느 수준까지 접근할 수 있는지 확실한 보안 정책이 수립되어야 한다. 또한, 이를 위해 FIWARE와 oneM2M은 기존에 신뢰 관계가 형성되어야 하고 전송된 검증 정보 등을 신뢰하기 위한 프로토콜도 마련되어야 한다. 이러한 IoT 플랫폼 간 토큰의 교차 사용에 대해서는 향후에 보다 집중적인 연구가 필요하다.

6. 결론 및 향후 연구

본 연구에서는 IoT 플랫폼 간의 보안 인터워킹에서 가장 중요한 인증과 인가를 제공하는 OAuth 2.0 기반의 보안 프레임워크를 개발하여 oneM2M을 대상으로 적용하여 테스트하였으며 FIWARE의 보안 아키텍처를 분석하고 구현하였다. 또한 oneM2M 보안 프레임워크와 FIWARE의 보안 아키텍처를 기반으로 보안 인터워킹 구조를 제안하고 구현하였으며 이 과정에서 보안 인터워킹 구조와 보안 프레임워크의 검증을 위해 LED 예제를 개발하였다. LED 예제는 oneM2M의 외부에서 접근하는 사용자가 보안 프레임워크에서 인증과 인가를 받지 않으면 LED를 제어할 수 없도록 만들어졌으며 향후 LED만이 아니라 oneM2M 플랫폼이 적용된 CCTV, 도어락 등 실용적인 사물들에 적용될 것이다. 본 논문에서는 LED 예제를 통해 제안한 보안 프레임워크가 권한이 없는 외부 플랫폼의 사용자가 oneM2M의 자원(LED)에 접근하는 요청들을 제한하는데 효과가 있음을 확인하였다.

5장에서 언급하였듯이, 향후 연구로는 지역이나 시간 정보 등을 이용한 상세한 인가를 제공할 수 있는 보안 프레임워크의 개발이 필요하며 이와 관련하여 보안 인터워킹 시나리오들을 자세히 정의해야 한다. 또한 보안 프레임워크는 resource owner password credentials 방식만이 아닌 다양한 OAuth 2.0 grant 유형을 제공해야 하며, 이에 관한 토큰 발급의 자동화나 경량화 등의 연구와 IoT의 상황인지 특성을 십분 활용한 토큰 발급 방안에 대한 연구도 필요하다. 그리고 FIWARE와 oneM2M이 아닌 다른 플랫폼을 대상으로 사용할 수 있는 일반적인 보안 인터워킹 구조를 연구하고, 그 과정에서 각 도메인의 요구사항(보안 정책)에 따라 구체적인 인증과 인가 과정을 구현하고, 인증, 인가 이외의 다른 보안 기능도 보안 프레임워크가 제공할 수 있도록 개발되어야 한다.

References

[1] S.-R Oh and Y.-G Kim, “Security Requirements for Internet of Things,” *IEEE 2017 Platform Technology and Service (PlatCon)*, pp.1-6, February 2017.

[2] IDC (International Data Corporation), “Worldwide Semiannual Internet of Things Spending Guide,” [Internet], https://www.idc.com/ getdoc.jsp?containerId=IDC_P29475.

[3] Cisco, “The Internet of Things,” [Internet], https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[4] oneM2M, “Latest Draft Specifications,” [Internet], http://www.onem2m. org/technical/latest-drafts.

[5] D. Hardt. RFC6749: The OAuth 2.0 Authorization Framework. 2012.

[6] FIWARE, “Security Architecture,” [Internet], https://forge.fiware.org/ plugins/mediawiki/wiki/fiware/index.php/Security_Architecture.

[7] I. T. Hwang and Y.-G. Kim, “Analysis of Security Standardization for the Internet of Things,” *IEEE 2017 Platform Technology and Service (PlatCon)*, pp.1-6, February, 2017.

[8] J. H. Kim, S.-C. Choi, N.-M. Sung, and J. S. Yun, “Standard Interworking Technologies for Internet of Things,” *The Journal of The Korean Institute of Communication Sciences*, Vol.33, pp.55-64, 2016.

[9] D. G. Lee, D.-H. Kim, and T.-M. Chung, “A Proposal for a Method of Interworking with DDS on IoT Platforms,” *Proceedings of Symposium of the Korean Institute of Communications and Information Sciences*, Vol.60, pp.385-386, 2016.

[10] K.-H. Kim, H.-K. Lim, J.-S. Heo, and Y.-H. Han, “Design and Implementation of CoAP Authorization Framework Based on OAuth 2.0,” *Tr. Comp. and Comm. Sys.*, Vol.6, pp.329-342, 2017.

[11] S. Sciancalepore, G. Piro, D. Caldarola, G. Boggia, and G. Bianchi, “OAuth-IoT: an access control framework for the Internet of Things based on open standards,” *Computers and Communications (ISCC)*, pp.676-681, 2017.

[12] S. W. Jung and S. Jung, “Personal OAuth authorization server and push OAuth for Internet of Things,” *International Journal of Distributed Sensor Networks*, Vol.13, pp.1-11, 2017.

[13] P. Solapurkar, “Building Secure Healthcare Services Using OAuth 2.0 and JSON Web Token in IOT Cloud Scenario,” *Contemporary Computing and Informatics (IC3I)*, pp.99-104, 2016.

[14] F. Fernandez, A. Alonso, L. Marco, and J. Salvachua, “A Model to Enable Application-scoped Access Control as a Service for IoT Using OAuth 2.0,” *Innovations in Clouds, Internet and Networks (ICIN)*, pp.322-324, 2017.

[15] J. S. Yun, R. C. Teja, N. Chen, N.-M. Sung, and J. H. Kim, “Interworking of oneM2M-based IoT Systems and Legacy Systems for Consumer Products,” *Information and Communication Technology Convergence (ICTC)*, pp.423-428, October, 2016.

[16] oneM2M, “TS-0012: Base Ontology v3.3.0,” 2017.

[17] oneM2M, “TS-0014: LWM2M Interworking v3.1.0,” 2017.

[18] oneM2M, “TS-0021: oneM2M and AllJoyn Interworking v2.0.0,” 2016.

[19] oneM2M, “TS-0024: OIC Interworking v2.0.1,” 2017.

[20] oneM2M, “TS-0026: 3GPP Interworking v0.2.0,” 2017.

[21] oneM2M, “TS-0030: Generic Interworking v0.2.0,” 2017.

[22] oneM2M, “TS-0033: Proximal IoT Interworking v0.1.0,” 2017.

[23] oneM2M, “TR-0027: DDS usage in oneM2M system v0.2.0,” 2016.

[24] oneM2M, “TR-0042: WoT Interworking v0.0.1,” 2017.

[25] oneM2M, “TR-0043: Modbus Interworking v0.1.0,” 2017.

[26] OCEAN (Open alliance for iot standard), “LED Sample,” [Internet], http://www.iotocean.org/download/?tab1=2.

오 세 라

http://orcid.org/0000-0002-4448-2748
e-mail : terious551@sju.ac.kr
2016년 동양미래대학교 컴퓨터소프트웨어공학과(학사)
2016년~현 재 세종대학교 정보보호학과 석・박사통합과정
관심분야 : 사물인터넷 보안, 접근통제

김 영 갑

http://orcid.org/0000-0001-9585-8808
e-mail : alwaysgabi@sejong.ac.kr
2001년 고려대학교 컴퓨터학과부전공 (학사)
2003년 고려대학교 컴퓨터학과(석사)
2006년 고려대학교 컴퓨터학과(박사)
2006년~2008년 고려대학교 정보보호대학원 연구교수
2008년~2010년 국가평생교육진흥원 선임전문원
2010년~2013년 고려대학교 연구교수
2013년~2015년 대구가톨릭대학교 IT공학부 조교수
2015년~현 재 세종대학교 정보보호학과 부교수
관심분야 : 사물인터넷 보안, 보안공학, 위험분석