OAuth 2.0 기반 CoAP 인증 프레임워크 연구 동향 및 제안

김경한, 임현교, 허주성, 한연희*
한국기술교육대학교

요 약

최근 사물인터넷에 대한 관심과 투자가 크게 증가하고 있으나 보안 측면에서 문제가 지속적으로 제기되고 있다. 그 해결책으로서 IETF ACE 워킹 그룹이 기존의 OAuth 2.0 기반으로 여러 제약적인 사물인터넷 환경에 적합한 새로운 보안 프레임워크인 ACE 프레임워크 표준을 제정 중에 있다. 본 고에서는 우선 OAuth 2.0 기반 사물인터넷 인증 프레임워크 연구 동향에 대해 소개하고, 새로운 OAuth 2.0 기반 CoAP 인증 프레임워크를 제안한다.

I. 서 론

최근 ICT 분야에서 가장 주목 받는 기술 키워드인 사물인터넷[1]은 사물이 인터넷을 통해 서로 연결되어, 다양한 서비스를 만들어 내는 기술을 일컫는다. 사물인터넷에 대한 관심과 투자는 지속적으로 증가하고 있지만 사물인터넷의 확산에 더욱 박차를 가하기 위해서 해결해야 할 가장 큰 문제 중 하나는 보안(Security) 관련 문제이다. 사물인터넷의 대표적인 분야에 스마트 홈, 스마트 카, 스마트 빌딩, 웨어러블 기기 등이 있기 때문에 기기로부터 전송되는 정보들은 민감한 개인정보를 포함하는 경우가 많다.

이러한 사물인터넷의 보안 문제 한계성 때문에 IoT 서비스를 제공하고 있는 사업체들은 자체적으로 개발한 웹 기반 응용이나 스마트폰 응용만을 그들이 설치한 IoT자원에 접근 가능하게 하고 있으며, 서드파티 (3rd Party) 응용에게는 접근을 허용하지 않는 추세이다. IoT 서비스 업체 측에서 운영하고 있는 자사의 IoT자원에 접근 가능한 새로운 응용들을 서드파티 업체에서 다양하게 개발될 수 있다면 결과적으로 IoT 서비스 업체에서 많은 투자를 통하여 운영하는 IoT 서비스에 대한 수요와 이윤도 증가할 것으로 예상된다.

이미 일반적인 웹서비스 분야에서는 이와 같은 서드파티 응용에 대한 자사의 자원 접근을 인가하기 위하여 OAuth 2.0 인증 프레임워크[2]가 활발히 이용되고 있다. 특히, 구글, 페이스북, 네이버, 다음카카오와 같이 많은 자원을 지닌 서비스 업체에서는 자신들의 자원에 접근하기를 원하는 서드파티 응용들에게 OAuth 2.0인증을 요구하고 있으며, 이를 기반으로 해당 자원들의 소유자 허락을 득하여 그 자원들을 개방하고 있다.

한편, IETF (Internet Engineering Task Force)에서는 2014년 1월에 ACE (Authentication and Authorization for Constrained Environments) 워킹 그룹[3]을 결성하여, 상대적으로 열악하고 각종 제약을 가진 소형기기의 통신에 OAuth 2.0을 적용한 보안 프레임워크 표준을 개발하기 시작하였다.

그러나 OAuth 2.0 프레임워크가 TLS에 의존적인 것과는 달리 ACE 프레임워크에서는 자체적으로 각 참여 주체가 보내는 메시지에 대한 서명을 포함한 인증 및 검증 방법, 메시지 암호화 방안 등을 모두 담고 있다. 즉, ACE 프레임워크는 OAuth 2.0의 기존 개발 동기를 고수하며 개발되기 보다는, 다른 계층의 도움없이 오로지 응용 계층으로만 해결이 가능한 새로운 경량 보안 체계를 제안하고 있다. 현재 각종 제약을 지닌 소형기기에 적용가능한 TLS에 대한 연구[4, 5]도 활발하게 진행되고 있기 때문에, OAuth 2.0의 기존 개발 동기에 맞춘 사물인터넷 인증 프레임워크 설계는 계속하여 연구될 필요가 있다. 이에 본 고에서는 TLS가 소형기기에 적용될 수 있다는 가정하에서 기존 OAuth 2.0 프레임워크의 개발 동기를 준수하는 OAuth 2.0 기반 CoAP 보안 프레임워크 설계를 제안한다.

본 고에서는 2장에서 OAuth 2.0, 3장에서 IETF ACE 프레임워크에 대해 소개한 뒤, 4장에서 OAuth 2.0 기반 CoAP인증 프레임워크를 제안한다.

* 교신저자: 한연희
본 논문은 2016년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 행된 연구임 (No.B0717-16-0033, 서비스/단말/네트워크 다양성 지원을 위한 미래형 다차원 네트워크 시스템 아키텍쳐).

II. OAuth 2.0

OAuth 2.0은 인터넷에 존재하는 자원에 대해 임의의 제3자 웹, 모바일 앱, 또는 어플리케이션이 접근하기 위한 권한 인증 절차를 표준화한 인증 프레임워크이다[2].

OAuth의 시작은 2006년에 Twitter OpenID 개발자인 Blaine Cook과 소셜 북마크 서비스인 Ma.gnolia의 개발자가 OpenID를 지닌 사용자들로 하여금 권한을 인증해 주고자 하는 방식을 논의하던 때로 거슬러 올라간다. Twitter와 Ma.gnolia 두 회사의 개발자들은 API 접근 위임에 대한 표준이 필요하다는 사실을 알게되고, 2007년 4월 비공식적인 OAuth 논의 그룹을 만든 뒤 OAuth 드래프트 제안서를 만들어 공유했다. 이후, 2008년 73차 미네소타 IETF 회의에서 OAuth BoF (Bird of Feather)가 만들어졌고, 2010년에 OAuth 1.0 공식 표준안이 RFC 5849로 발표되었다[6].

OAuth 1.0이 Twitter를 주축으로 비공식적으로 논의되던 인증 및 권한 부여 프로토콜이 IETF라는 국제 표준으로 인정되었다면, 이후 IETF OAuth WG이 만들어진 이후에는 다양한 회사들이 적극적으로 참여하여 IETF에서 약 2년간의 공식적인 미팅을 통하여 좀더 성숙한 형태의 OAuth 2.0 표준인 RFC 6749 [2]를 발표하였다. OAuth 1.0은 그 자체로 Protocol을 정의하고 있지만 OAuth 2.0은 OAuth 1.0과의 호환성을 고려하지 않고 좀 더 인증 절차를 간단히 만든 프레임워크 (Framework) 형태로 발표가 되었다. 즉, OAuth 2.0은 프레임워크이기 때문에, OAuth 2.0을 도입하는 회사에서 자체적으로 자신들에게 좀 더 맞는 형태로 구현을 하여 사용할 수 있는 융통성이 존재한다. 그러한 융통성때문에 OAuth 2.0은 정식 RFC가 되기 전부터 Facebook, Google, Microsoft 등 주요 인터넷 서비스 업체에서 OAuth 2.0을 사용하기 시작하였다.

OAuth 2.0의 권한 인증 절차에 참여하는 주체들의 역할 구성 형태는 "3-legged OAuth"라고 불리우지만, 자원 소유자 (Resource owner), 클라이언트(Client), 자원 서버(Resource server), 권한 서버(Authorization server), 이렇게 4가지 주체로 이루어져 있다. 자원 소유자는 보호 자원에 대한 접근 권한을 부여할 수 있는 개체로 일반적으로 이용자를 나타낸다. 클라이언트는 자원 서버에 보호 자원을 요청하고 관련 서비스를 제공하는 (서드-파티) 어플리케이션이고, 자원 서버는 보호된 자원에 대한 서비스 API를 제공하는 서버이며, 권한 서버는 클라이언트가 보호된 자원에 대한 제한된 접근을 할 수 있도록 자원 접근 권한을 관리하는 서버이다.

OAuth 2.0를 이용하면 클라이언트는 자원 서버가 가지고 있는 특정 자원 소유자의 자원에 제한적인 접근 권한을 얻을 수 있다. 자원 소유자의 권한 승인 하에 권한 서버를 통해서 클라이언트에게 접근 토큰(Access Token)이 발급된다. 클라이언트는 접근 토큰을 사용해서 보호되고 있는 자원에 접근하는 대신, 특정 범위와 수명 등 접근에 제한이 있다.

OAuth 2.0에는 클라이언트 유형, 자원에 대한 장기적 접근 여부 및 제공 서비스의 특성 등에 따라 다음의 4가지 권한 승인 방법/절차가 존재한다.

- 권한 코드 승인 (Authorization Code Grant)
- 암묵적 승인 (Implicit Grant)
- 자원 소유자 비밀번호 자격증명 승인 (Resource Owner Password Credentials Grant)
- 클라이언트 자격증명 승인 (Client Credentials Grant)

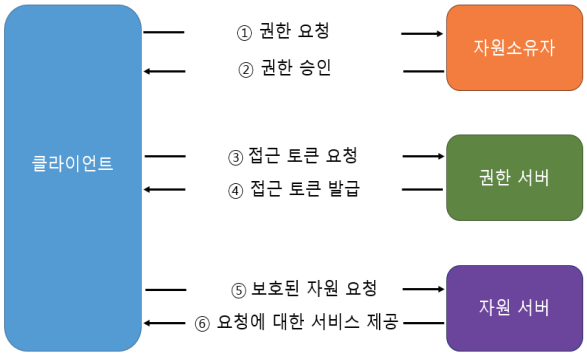

그림 1. OAuth 2.0 동작 흐름

표 1. OAuth 2.0 동작 흐름 설명

단계	세부 내용
①단계	클라이언트가 자원 소유자에게 자원에 대한 접근 권한을 요청
②단계	클라이언트에게 자원에 대한 접근 권한을 승인 (4가지 권한 승인 방법 정의)
③단계	클라이언트가 권한 서버에게 접근 토큰을 요청
④단계	권한 서버는 클라이언트를 인증 및 부여 받은 권한을 검증하고, 검증된 경우 클라이언트에게 접근 토큰을 발급
⑤단계	클라이언트가 자원 서버에게 접근 토큰으로 인증 및 자원을 요청
⑥단계	자원 서버는 접근 토큰을 검증하고, 검증된 경우 서비스 제공

한편, OAuth 2.0에서는 메시지의 암호화 및 각 주체들의 서명(Signature)을 포함한 인증 및 검증 등의 과정은 TLS (Transport Layer Security)쪽으로 모두 위임하고 있다. 즉, HTTP 기반 통신 환경에서는 OAuth 2.0 기술과 HTTPS를 함께 사용할 것을 권고하고 있으며, UDP 기반의 통신에서는

DTLS (Datagram Transport Layer Transport) 기술과 함께 사용할 것을 권고하고 있다.

III. IETF ACE 프레임워크

ACE 프레임워크는 한정된 자원을 지닌 기기가 지닌 자원에 대한 접근에 대해 인증 및 권한 부여 보안 프레임워크이다[7]. OAuth 2.0 프레임워크가 TLS에 의존적인 것과는 달리 ACE 프레임워크에서는 자체적으로 각 참여 주체가 보내는 메시지에 대한 서명을 포함한 인증 및 검증 방법, 메시지 암호화 방안 등을 모두 담고 있다. 즉, 하부 계층과는 독립적인 응용 계층의 보안 절차 및 방안을 자체적으로 제정하고 있다. 그러므로 TLS의 도움없이도 ACE 프레임워크 만으로 암호화, 인증 및 권한 부여까지 가능한 모든 보안 이슈가 해결되도록 설계하고 있다.

ACE 프레임워크를 이루는 주요한 4가지 구성요소는 다음과 같다.

- OAuth 2.0
- CoAP (Constrained Application Protocol): 전송 지연이 높고, 패킷 손실률이 높은 제약이 있는 네트워크 환경 및 노드 환경을 고려한 HTTP 스타일의 응용 계층 데이터 전달 프로토콜[8]
- CBOR (Concise Binary Object Representation): 최대한 작은 코드 사이즈, 작은 메시지 크기 그리고 버전 협상이 필요 없는 확장성이 가능하도록 설계된 데이터 포맷[9]
- COSE (CBOR Object Signing and Encryption): CBOR 메시지를 기반으로 한 서명 및 암호화 포맷[10]

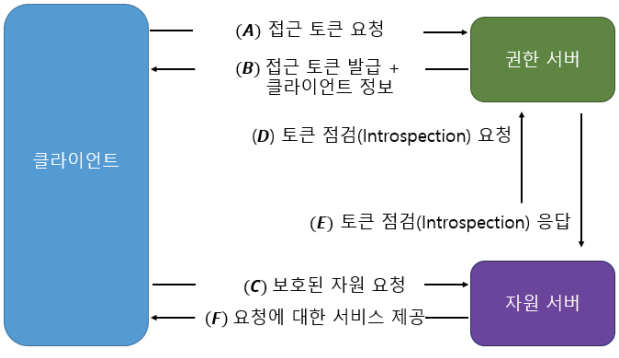

그림 2. ACE 동작 과정

위에서 언급한 4가지 구성요소를 기반으로 한 ACE의 전반적인 동작 과정은 〈그림 2〉에서 확인할 수 있으며 클라이언트와 권한 서버, 자원 서버와 권한 서버간의 통신은 미리 설정된 안전이 보장된 프로토콜을 사용하고, 클라이언트와 자원 서버간의 통신에 대해 두 노드가 공통으로 사용할 수 있는 프로토콜을 권한 서버가 각 노드에게 알려준다. 〈표 2〉는 〈그림 2〉에 표시된 절차 (A)~(F)의 세부내용을 나타낸다.

표 2. ACE 동작 흐름 설명

단계	세부 내용
(A) 접근 토큰 요청	클라이언트는 권한 서버에 있는 토큰 종점(endpoint)에 토큰 요청 메시지를 전달한다. 이 메시지에는 접근 토큰 발급을 위해 필요한 정보가 포함되어있는데, 특히 ACE 프레임워크는 접근 토큰 종류로서 주로 PoP (Proof-of-Possession) 토큰[11]을 사용하며, 클라이언트가 사용하길 원하는 키의 종류를 알려줘야 한다. 또한 권한 서버는 접근 토큰과 그에 해당되는 키를 자체적으로 저장 및 관리한다.
(B) 접근 토큰 응답	권한 서버가 클라이언트의 요청이 유효하다고 판단할 경우, 응답 메시지에 클라이언트 정보(Client Information)라고 하는 다양한 파라미터와 접근 토큰 담아 클라이언트에게 보낸다.
(C) 자원 요청	클라이언트는 자원 서버의 보호된 자원에 접근하기 위해 자원 서버에게 접근 토큰을 제공할 뿐만 아니라 접근 토큰의 소유를 증명해야만 한다. 접근 토큰의 소유를 증명하기 위해, 클라이언트는 접근 토큰에 해당되는 키를 이용해 메시지 다이제스트(Message Digest)나 메시지 서명(signature)을 통해 요청 메시지를 암호화한다. 권한 서버에게 요청, 데이터 베이스 접근, 토큰 안에 해당되는 키를 포함시키는 방법 등으로 키를 알고 있는 자원 서버는 요청 메시지를 받았을 때, 이 요청 메시지를 보낸 클라이언트가 보호된 자원에 접근 권한이 승인이 되었는지 아닌지 확인할 수 있다.
(D) 토큰 점검 요청	자원 서버는 점검 종점(Introspection endpoint) [12]에 토큰 점검 요청 메시지를 보낸다. 토큰 점검 요청은 선택적인 단계로서 자원 서버가 토큰의 유효성을 스스로 판단할 수 있다면 생략 가능하다.
(E) 토큰 점검 응답	권한 서버는 토큰의 유효성을 확인하고 접근 범위, 유효성 등의 파라미터들과 함께 응답한다. 이후, 자원 서버는 권한 서버에게 받은 파라미터들을 분석하여 클라이언트의 자원 요청을 수락할지 거부할지 선택한다.
(F) 보호된 자원 제공	자원 서버는 적절한 응답 코드와 함께 클라이언트에게 응답 메시지를 보내며 이 응답 메시지는 키를 이용해 암호화시킨다.

접근 토큰 요청과 응답에서 모든 OAuth 파라미터들은 CBOR 형식에 따라 문자열이 아닌 정숫값으로 매핑된다. 〈표 3, 4, 5〉는 그러한 CBOR 매핑을 보인다.

표 3. 권한 승인 방법에 대한 매핑

grant_type	CBOR Key	Major Type
Password	0	0 (uint)
authorization_code	1	0
client_credentials	2	0
refresh_token	3	0

표 4. profile에 대한 매핑

parameter name	CBOR Key	Major Type
http_tls	0	0 (uint)
coap_dtls	1	0
coap_oscoap	2	0

표 5. 토큰 요청 메시지에 대한 매핑

parameter name	CBOR Key	Major Type
client_id	1	3 (test string)
client_secret	2	2 (byte string)
response_type	3	3
redirect_uri	4	3
scope	5	3
state	6	3
code	7	2
error_description	8	3
error_uri	9	3
grant_type	10	0 (unit)
access_token	11	3
token_type	12	0
expires_in	13	0
username	14	3
password	15	3
refresh_token	16	3
alg	17	3
cnf	18	5 (map)
aud	19	3
profile	20	0

IV. OAuth 2.0 기반 CoAP 인증 프레임워크 설계 제안

앞서 언급한 대로, ACE 프레임워크는 응용 계층만으로 모든 보안 이슈를 해결하기 위한 새로운 경량 보안 체계이다. 하지만, 제약사항이 있는 사물인터넷 노드에 적용 가능한 TLS에 대한 연구도 활발하게 진행되고 있기 때문에, 이러한 경량 TLS를 기반으로 보안 체계를 제안하는 기존OAuth 2.0에 맞춘 사물인터넷 인증 프레임워크 설계도 계속하여 연구될 필요 있다. 따라서 본 고에서는 OAuth 2.0 기반 CoAP 인증 프레임워크를 제안한다.

제안 프레임워크를 설명하기에 앞서, OAuth 2.0 기반 CoAP 인증 프레임워크에는 몇 가지 기술적인 고려사항이 있다.

- 자원 서버와 권한 서버의 분리
- CoAP 추가 옵션 설정
- 자원 서버의 토큰 점검 메시지 캐시
- 토큰 점검 메시지 캐시의 일관성 유지

현재의 일반적인 OAuth 2.0 구성요소는 자원 소유자, 클라이언트, 자원 서버, 권한 서버이고, 자원 서버와 권한 서버는 구축 및 운영 방식에 따라 동일 또는 별도 서버로 존재 가능하다. 그러나 사물인터넷의 제약 환경에서는 기존의 OAuth 2.0과는 달리 권한 서버와 자원 서버가 동일 서버로 존재 하는 것은 불가능하다. 따라서 본 고에서 제안하는 프레임워크에서 권한 서버와 자원 서버의 분리는 필수 사항이다.

또한, 제안 프레임워크에서는 자원 서버의 한정된 네트워크 및 계산 자원을 고려하여, 클라이언트와 자원 서버, 권한 서버와 자원 서버의 통신은 CoAP을 통해 이루어져야 하며, 그 통신 과정의 보안을 위해 DTLS를 사용한다. 그 외의 User-agent와 클라이언트, User-agent와 권한 서버, 권한 서버와 클라이언트의 통신은 기존의 OAuth 2.0과 동일하게 CoAP이 아닌 HTTPS를 통해 이루어질 수 있다. 한편, 제안 프레임워크에서는 기존과 유사하게 토큰 전송 방식으로 "bearer"토큰을 사용한다[13].

한편, CoAP을 통해 클라이언트가 보호된 자원에 접근하기 위해, 혹은 기존 OAuth 2.0의 토큰 점검을 CoAP에 적용하기 위해서는 CoAP의 기존 옵션과는 별도로 〈표 6,7〉과 같이 새로운 옵션을 추가해야만 한다. 토큰 점검 요청 메시지와 오류 메시지를 위한 추가 옵션은 [14]에서 이미 정의된 옵션을 활용한다.

표 6. CoAP 토큰 점검 요청메시지 및 오류 메시지를 위한 추가 옵션

Name	Format	Length	Default
Bearer	opaque	var	(none)
Token-type-hint	string	1-255	(none)
Error	uint	0-2	(none)

표 7. CoAP 토큰 점검 응답 메시지 위한 추가 옵션

Name	Format	Length	Default
Active	string	1-255	(none)
Scope	string	1-255	(none)
Token-Type	string	1-255	(none)
Exp	uint	0-4	(none)

〈표 6〉은 토큰 점검 요청 메시지와 오류 메시지를 위해 CoAP에 추가해야 하는 옵션으로, 클라이언트는"Bearer"옵션을 통해 토큰을 전송하고 "Token-type-hint"옵션을 통해 해당 토큰의 타입을 전송한다."Error"옵션으로 사용되는 오류 코드는 [10]을 참고하여 invalid_request (0), invalid_token (1), insufficient_scope (2)와 같이 총 3가지로 구성된다. invalid_request (0)은 요청 메시지가 토큰 점검에 꼭 필요한 파라미터를 포함하고 있지 않거나 지원하고 있지 않는 파라미터를 포함 발생한다. 이 경우 자원 서버는 응답 코드로 4.00 (Bad Request)을 전송한다. invalid_token (1)은 만료, 훼손, 폐지 등 여러 이유로 접근 토큰이 유효하지 않을 경우 발생한다. 이 경우 자원 서버는 응답 코드로 4.01 (Unauthorized)을 전송한다. insufficient_scope (2)은 요청 메시지가 보호된 자원에 대해 접근 토큰에 의해 제공되는 권한 보다 더 많은 권한을 요구할 때 발생한다. 이 경우 자원 서버는 응답 코드로 4.03 (Forbidden)을 전송한다.

〈표 7〉은 토큰 점검 응답 메시지를 위해 추가해야 하는 옵션이다. 기존 OAuth 2.0의 토큰 점검 응답 메시지에 대해 정의된 파라미터[12]와는 달리 제안 프레임워크에서는 제약 노드인 자원 서버를 고려해 필수적으로 필요한 옵션만 사용한다."Active" 옵션은 접근 토큰의 상태를 알 수 있는 값이다. "Scope"옵션은 클라이언트가 보호된 자원에 접근 가능한 범위를 나타낸다. "Token-type"옵션은 접근 토큰의 타입을 나타내는 옵션으로 토큰을 검색할 때 사용한다. "Exp"옵션은 접근 토큰이 만료되는 시간을 나타내며 만료시간이 지난 토큰은 거절해야 한다.

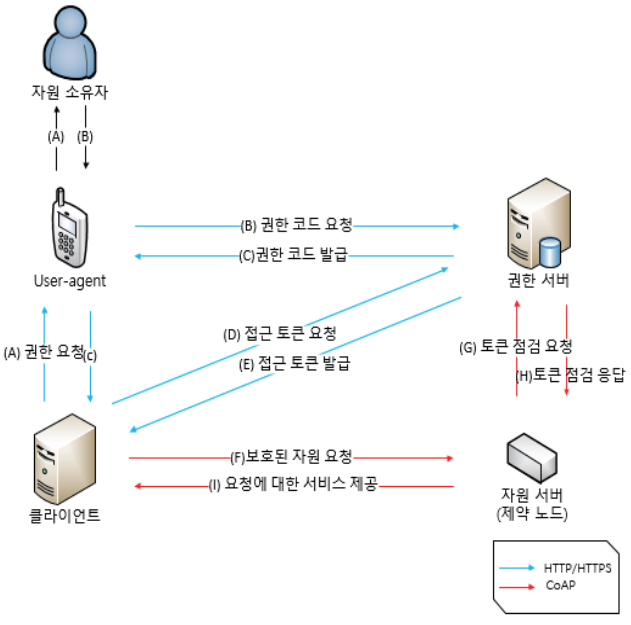

그림 3. 제안 프레임워크 동작 과정

〈그림 3〉은 클라이언트가 자원 서버의 보호된 자원에 접근이 필요해서 자원 소유자에게 처음으로 권한을 요청할 때, 제안 프레임워크 동작 과정을 나타내며 〈표 8〉은 〈그림 3〉에 표시된 절차 (A)~(I)의 세부내용을 나타낸다.

표 8. 제안 프레임워크 동작 흐름 설명

단계	세부 내용
(A) 권한요청	클라이언트가 User-agent를 통해 자원 소유자에게 자원에 대한 접근 권한을 요청한다.
(B) 권한 코드 요청	자원 소유자는 User-agent를 통해 권한 서버에게 권한 코드를 요청한다.
(C) 권한 코드 발급	권한 서버의 자원 소유자에 대한 인증 방식을 통해 유효할 경우, 권한 코드를 생성하여 코드를 User-agent에게 전송한다.
(D) 접근 토큰 요청	User-agent를 통해 권한 코드를 전달 받은 클라이언트는 권한 코드를 기반으로 client_id 및 client_secret을 권한 서버에게 전달함으로써 접근 토큰 요청한다.
(E) 접근 토큰 발급	권한 서버는 클라이언트의 요청이 유효하다고 판단할 경우, 접근 토큰을 생성하고 클라이언트에게 전송한다.
(F) 보호된 자원 요청	클라이언트가 자원 서버에게 접근 토큰으로 인증 및 자원을 요청한다.
(G) 토큰 점검 요청	자원 서버는 권한 서버에게 자원 관찰 기능 옵션을 포함시킨 토큰 점검 요청 메시지를 보낸다.
(H) 토큰 점검 응답	권한 서버는 토큰의 유효성을 확인하고 접근 범위, 유효성 등의 파라미터[12]들과 함께 응답하고 해당 토큰에 대한 자원 관찰 목록에 자원 서버를 추가한다. 이후, 자원 서버는 권한 서버에게 받은 파라미터들을 분석하여 클라이언트의 자원 요청을 수락할지 거부할지 선택한다.
(I) 요청에 대한 서비스 제공	자원 서버는 적절한 응답 코드와 함께 클라이언트에게 응답 메시지를 보낸다.

한편, 기존 OAuth2.0에에서는 네트워크의 트래픽 양과 노드의 부하를 줄이기 위해, 선택적으로 자원 서버에서 토큰 점검 메시지를 캐시(Cache)하여 추후 해당 토큰으로 보호된 자원에 접근 시 자원 서버가 자체적으로 토큰의 상태를 점검할 수 있게 한다. 이러한 방식은 사물인터넷의 제약 환경에서는 매우 중요하므로 자원 서버의 토큰 점검 메시지 캐시는 제안 프레임워크에서는 필수사항이다.

그러나, 이러한 토큰 점검 매시지 캐시 방식을 도입하면 실제 토큰 정보를 관리하는 권한 서버에서의 토큰 정보와의 일관성(Consistency)를 유지해야 하는 절차가 필요하다. 예를 들어, 자원 소유자가 토큰을 철회를 요청하는 경우 권한 서버에서는 해당 토큰을 폐기시키고 해당 토큰에 관한 정보를 캐시하고 있는 자원 서버에게 갱신된 정보를 알려야만한다.

제안 프레임워크에서는 이러한 일관성 문제를 해결하기 위해 CoAP의 확장기능인 자원 관찰(Observing Resources) 기능을 활용한다[15]. 자원 관찰 기능은 자원(예를 들어 토큰 자원)을 관리하는 측에서 자원의 상태가 변화할 때마다 해당 자원의 상태 변화에 관심이 있는 측으로 통지를 하는 기능이다.

이를 위해 제안 프레임워크의 자원 서버는 토큰 점검 메시지의 캐시 시간(Cache Time)을 토큰 만료 시간으로 설정한다. 자원 서버는 토큰 점검 메시지를 권한 서버에게 요청함과 동시에 자원 관찰 등록(Registration)을 수행한다. 이후, 토큰 발급 시간 만료, 자원 소유자의 토큰 철회, 클라이언트의 토큰 재발급 요청 등 토큰의 상태가 변화할 경우 권한 서버는 자원 서버에게 해당 토큰에 대한 상태 변화 통지 메시지를 전송한다. 이 메시지를 받은 자원 서버는 해당 통지 메시지를 해석하여 자신이 캐시한 토큰에 대해 적절한 작업을 수행한다. 이러한 방법에 의하여 자원 서버에 캐시된 토큰 정보는 늘 최신성이 유지되기 때문에 잘못된 토큰 정보로 인한 오류는 발생하지 않는다.

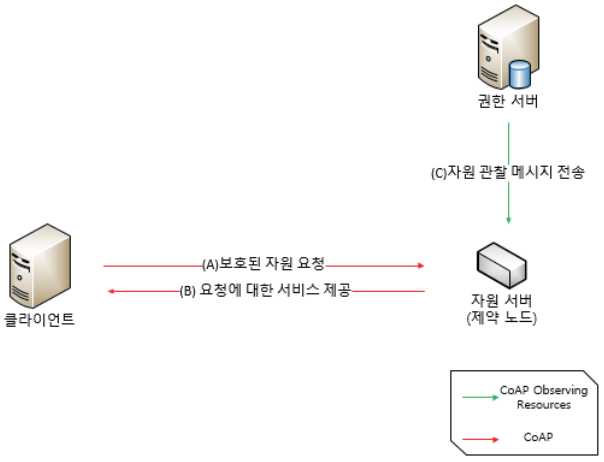

그림 4. 자원 서버의 토큰 캐시 이후 자원 관찰 기능을 통한 토큰 캐시 일관성 관리

〈그림 4〉는 자원 서버의 토큰 점검 메시지 캐시 후 제안 프레임워크 동작 과정을 나타내며 〈표 9〉는 〈그림 4〉에 소개된 (A)~(C) 과정의 세부내용을 나타낸다.

표 9. 자원 서버의 토큰 캐시 이후 제안 프레임워크 동작 흐름 설명

단계	세부 내용
(A)보호된 자원 요청	클라이언트가 자원 서버에게 접근 토큰으로 인증 및 자원을 요청한다.
(B)요청에 대한 서비스 제공	자원 서버는 캐시 된 토큰 정보를 통해 토큰의 유효성을 확인 후 적절한 응답 코드와 함께 클라이언트에게 응답 메시지를 보낸다.
(C)자원 관찰 메시지 전송	토큰에 대한 정보가 갱신 될 경우, 해당 토큰에 자원 관찰 기능 등록이 되어있는 자원 버에게 통지 메시지를 전송한다.

V. 결론

인터넷 프로토콜 표준을 제정해 왔던 IETF에서는 CORE WG을 중심으로 사물인터넷을 위한 표준인 CoAP 프로토콜을 제정하였으며, 최근에 결성된 ACE 워킹그룹은 사물인터넷 환경에서 기존 OAuth 2.0를 기반으로 응용 계층 기반의 새로운 ACE 보안 프레임워크 표준을 제정하고 있다. 본 고는 우선 OAuth 2.0의 역사 및 기술을 소개하였으며, 현재 한창 표준화가 진행중인 ACE 프레임워크의 주요 내용을 정리하여 소개하였다. 또한 ACE 프레임워크와는 달리 경량 TLS를 활용하는 새로운 OAuth 2.0 기반 CoAP 인증 프레임워크 및 인증 절차를 제안하였다. 특히, 제안 프레임워크에서 네트워크의 트래픽 양과 노드의 부하를 줄이기 위하여 자원 서버의 토큰 캐시 방안 및 자원 관찰 기능을 통한 캐시 일관성 유지 방안을 제안하였다.

참 고 문 헌

[1] 김호원, 김동규, “IoT 기술과 보안”, 정보보호 학회지, 제22권, 제1호, pp. 7-31. 2012. 2.

[2] D. Hart, “The OAuth 2.0 Authorization Framework,” IETF RFC 6749, Oct. 2012.

[3] IETF ACE WG, https://datatracker.ietf.org/wg/ace/

[4] 정진희, 조대호, “무선 환경에서 SSL/TLS를 사용하는 IoT의 에너지 효율성 향상을 위한 기법,” 정보보호학회논문지, 제26권, 제3호, pp. 661-666. 2016.06.

[5] A. Capossele, V. Cervo, G. D. Cicco, C. Petrioli, “Security as a CoAP resource: An optimized DTLS implementation for the IoT,” IEEE International Conference on Communications (ICC), 2015.

[6] E. Hammer-Lahav (Ed.), “The OAuth 1.0 Protocol,” IETF RFC 5849, Apr. 2010.

[7] L. Seitz, G. Selander, and E. Wahlstroem, “Authentication and Authorization for Constrained Environments (ACE),” draft-ietf-ace-oauth-authz-02, June 2016.

[8] Z. Shelby, K. Hartke, C. Bormann, “The Constrained Application Protocol (CoAP),” RFC 7252, June 2014.

[9] C. Bormann, P. Hoffman, “Concise Binary Object Representation (CBOR),” IETF RFC7049, Oct. 2013.

[10] J. Schaad, “CBOR Object Signing and Encryption (COSE),” draft-ietf-cose-msg-17, Aug. 2016.

[11] J. Bradley, P. Hunt, M. Jones, T. Tschofenig, "OAuth 2.0 Proof-of-Possession: Authorization Server to Client Key Distribution," draft-ietf-oauth-pop-key-distribution-02, July 2014.
[12] J. Richer (Ed.), "OAuth 2.0 Token Introspection," RFC 7662, Oct. 2015.
[13] M. Jones, D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage"RFC 6750, Oct, 2012.
[14] H. Tschofenig, "The OAuth 2.0 Bearer Token Usage over the Constrained Application Protocol (CoAP)" draft-tschofenig-ace-oauth-bt-00, July 2014.
[15] K. Hartke, "Observing Resources in the Constrained Application Protocol (CoAP)," IETF RFC 7641, Sep. 2015.

약 력

김 경 한

2015년 한국기술교육대학교 컴퓨터공학부 학사
2015년~현재 한국기술교육대학교 컴퓨터공학과 석사과정
주요 관심분야: Internet of Things, OAuth 2.0

임 현 교

2015년 한국기술교육대학교 컴퓨터공학부 학사
2015년~현재 한국기술교육대학교 컴퓨터공학과 석사과정
주요 관심분야: LTE/EPC Core Networks, Software-defined Networks (SDN), Mobility Management

허 주 성

2016년 한국기술교육대학교 컴퓨터공학부 학사
2016년~현재 한국기술교육대학교 컴퓨터공학과 석사과정
주요 관심분야: Machine Learning, Social Network Analysis

한 연 희

1996년 고려대학교 수학과 학사
1998년 고려대학교 컴퓨터공학과 석사
2002년 고려대학교 컴퓨터공학부 박사
2002년~2006년 삼성종합기술원 전문연구원
2013년~2014년 뉴욕주립대 (SUNY) 방문교수
2006년~현재 한국기술교육대학교 컴퓨터공학부 부교수
주요 관심분야: Mobile Networks, Internet of Things, Machine Learning, Social Network Analysis, OAuth 2.0