S-Broker 개념을 적용한 NoSec 기반의 Secure-CoAP Protocol

허 웅, 김영세, 김기천*
건국대학교 일반대학원 IT융합정보보호학과, *건국대학교 컴퓨터공학과
e-mail: a201572280@konkuk.ac.kr/ rladudtp87@konkuk.ac.kr /
*kckim@konkuk.ac.kr

Secure-CoAP Protocol over NoSec applying the concept S-Broker

ung Heo, youngse Kim, keecheon Kim*
Dept. of IT Convergence Information Security, Konkuk University
*Dept. of Computer Science & Engineering, Konkuk University
Convergence

요 약

최근 IoT 프로토콜 가운데 가장 활발히 논의되는 프로토콜로 CoAP(Constrained Application Protocol)이 있다. CoAP은 4가지 보안모드로 운영된다. 그 중 3가지 모드인 PresharedKey, RawPublicKey, Certificate 모드의 경우 DTLS(Datagram Transport Layer Security)가 적용된 방식이다. 반면 NoSec 모드는 DTLS가 적용되지 않은 기본방식이다. 본 논문에서는 DTLS의 복잡한 Handshake 방식으로 인한 전력소모 및 Performance의 저하를 고려한 새로운 방식을 제안한다. NoSec 환경의 CoAP 프로토콜에 S-Broker(Secure-Broker)를 적용한 security 및 performance 향상 방안이다. 제안한 방식으로 경량화 통신을 구현하여 무결성과 보안 강도를 높였다. 추가적으로 Proxy의 forwarding 기능과 caching 기능을 구현하여 성능의 향상을 도모한다.

1. 서론

최근 사물인터넷 기술의 급격한 성장에 따라 그 중요성이 더욱 대두되고 있다. 인터넷 국제 표준 단체인 IETF(Internet Engineering TaskForce)에서는 6LoWPAN, CoRE WG 등 저 전력 소형장치의 표준 기술 개발을 진행해오고 있다. 그 가운데 CoRE WG의 메모리, 에너지, 성능 등에 제약이 있는 저전력 환경의 M2M 환경을 위한 웹 기반 프로토콜인 CoAP이 주목받고 있다.

CoAP은 기존의 HTTP나 MQTT와 같은 프로토콜보다 제한적인 환경에서 사용할 수 있으며 호환성이 우수하고 UDP 기반의 신속한 데이터 전송을 지원한다. 또한, DTLS의 지원을 통한 보안적인 부분까지 고려하고 있다. 그러나 DTLS의 적용으로 인한 복잡한 HandShake 과정과 보안키 공유과정, 이로 인한 전력소모의 상승, 속도저하의 문제가 발생한다.[1]

이에 본 논문에서는 DTLS를 적용하지 않은 NoSec 환경의 통신방식을 제안한다. 기존 CoAP의 동작방식은 유지하되 DTLS의 보안적인 요소를 S-Broker로 보완하는 방식이다.

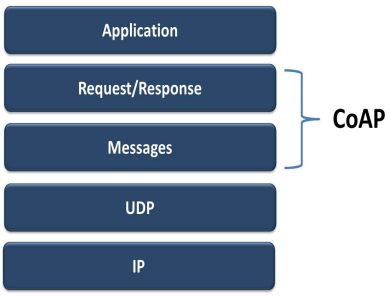
2장에서 CoAP 동작과 CoAP-Security의 관련 기술에 대한 연구를 진행하며, 3장에서는 S-Broker를 활용한 Nosec 환경의 CoAP 프로토콜의 개선된 통신방식에 대해 제안한다. 4장에서 시뮬레이션을 수행 후, 5장에서 결론 및 연구방향을 다룬다.

* 교신저자

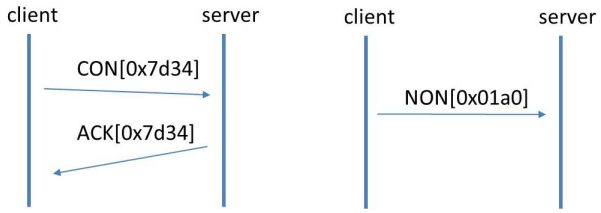
2. 관련 연구

2.1 CoAP 프로토콜 [2]

CoAP은 저전력, 저용량, 소형 노드 등과 같이 제한적인 환경에서 사용될 수 있는 웹 전송 프로토콜이다. RESTful 환경을 적용하기 때문에 HTTP 등의 프로토콜과도 쉽게 호환가능하다. 기본적으로 아래 그림 1과 같이 UDP 환경위에서 Request/Response 방식과 Messages 전송의 계층으로 구성된다. 보안적 요소를 위해 UDP와 CoAP 계층 사이에 DTLS의 계층을 사용할 수 있다.



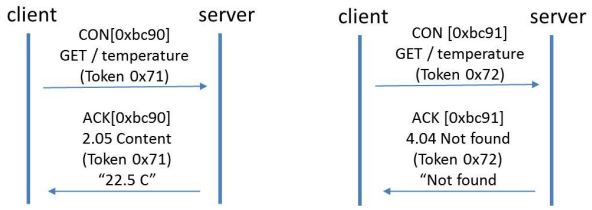
(그림 1) CoAP의 추상 계층



(그림 2) 신뢰 메시지 전송과 비 신뢰 메시지 전송

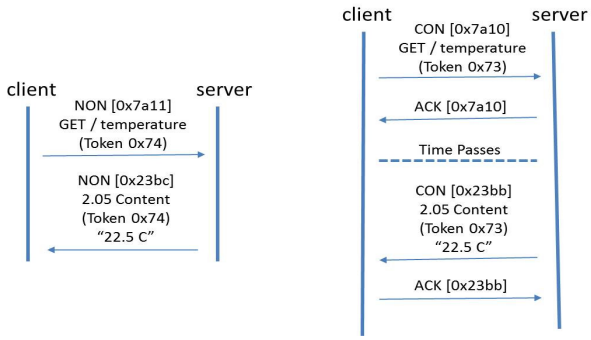
Request/Response 계층의 동작은 위 그림 2와 동일하다. 신뢰성 있는 전달을 위해서는 CON(확인형) 메시지를 전송한다. 여기서 Message ID(0x7d34)는 응답메시지에도 동일하게 사용되며 중복메시지 검출에도 사용된다. 만약 수신자가 응답메시지인 ACK를 처리할 수 없다면 ReSet 메시지를 보내게 된다.

지속적인 신뢰성있는 전송이 필요하지 않다면 비 신뢰성 메시지(NON)을 전송할 수 있다. NON 메시지에 대한 응답은 동일하며 처리할 수 없다면 ReSet 메시지를 통해 응답할 수 있다.



(그림 3) 2 GET Requests와 Piggybacked Responses

그림 3은 GET Request에 대한 Response(온도 값 or 온도 값을 찾을 수 없음)이다. 여기서 Message ID(0xbc91)은 하나의 트랜젝션을 구분하며 Token(0x71)은 각 요청에 대한 응답을 매칭할 때 사용된다.



(그림 4) 좌 : NoN-Con 메시지의 요청과 응답,
우 : GET Requests와 분산 Responses

그림 4의 왼쪽 그림은 비 신뢰 메시지 전송에 대한 요청과 응답을 보여주며, 오른쪽 그림은 GET Request에 대한 분산 Response를 보여주고 있다. 서버가 즉각적인 응답이 어려울 경우 ACK 메시지를 전송 후 일정 시간 경과 후에 CON 메시지를 전송하는 방식이다. 이때 Message ID는 변경되지만 초기 요청 Token은 유지되는 것을 알 수 있다.

2.2 CoAP Security [2]

CoAP의 보안모드에는 4가지 모드가 있다. 기본 모드인 NoSec 모드와 나머지 DTLS를 적용한 3가지의 모드이다. 각 모드별 특징은 아래 표 1과 같다.

<표 1> CoAP의 보안 모드

모드	동작원리
NoSec	보안 프로토콜이 적용되지 않은 기본모드, IPsec이나 기타 하위 계층의 보안 방식이 적용가능하다.
PreSharedKey	DTLS를 적용, 사전 공유키를 활용하여 Node/Key간의 1:1 매칭을 지원한다.
RawPublicKey	DTLS를 적용, 비대칭 비밀키 쌍을 지원한다.
Certificate	DTLS를 적용, 비대칭 비밀키 쌍과 X.509 인증을 지원한다.

2.3 센서 노드의 취약점 분석[3]

국제 표준 문서인 RFC(Request for Comments)의 CoRE WG의 문서 중 하나인 "Requirements for CoAP End-To-End Security"[2]에서는 CoAP의 각 동작 계층에 대한 취약점 부분을 다루고 있다. 이 가운데 센서 노드에 대한 취약점 부분을 요약 분석하면 다음과 같다.

<표 2> 센서 노드의 보안 취약점

Vulnerability of Sensor Node(Sensor-n)
Threat 1 : Spoofing - Client to Sensor-n **Threat 2** : Delaying - Client, Sensor-n **Threat 3** : GET Flooding(DDoS) - Client, Sensor-n **Threat 4** : Eavesdropping - Client to Sensor-n

위협 1은 Client, 센서로 위장하는 Spoofing 공격의 취약점이다. Client로 위장시 전송된 데이터를 가로채거나 Rest 명령을 통해 센서의 동작을 중단시킬 수 있다. 센서로 위장하는 경우 잘못된 정보를 전송하여 문제를 발생시킨다.

위협 2는 Client의 응답처리 및 네트워크 과부하에 따른 지연의 취약점이다. Sub-Client를 활용하거나 Proxy, Broker의 적용을 통해 보완 가능하다.

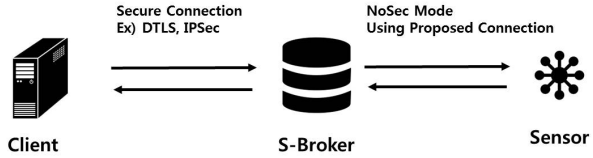
위협 3은 Client, 센서에 대한 DDoS 공격 취약점이다. Client의 경우 DDoS 대응 장비 및 블랙홀 라우팅 등의 다양한 방어 기술이 적용가능하다. 반면 제한된 센서-N 의 경우 위와 같은 기술이 적용 불가하다.

위협 4는 Node와 S-Broker 중간의 Sniffing 공격이다. DTLS 적용으로 보안 강도를 높일 수 있다. 그러나 복잡한 연산과 제한된 환경 내에서 DTLS를 적용하는 방법은 효율성이 떨어진다.

위 4가지 위협가운데 본 논문에서 제안하는 방안을 적용할 경우 해당 위협에 대한 보안 강도를 높일 수 있다.

3. 제안 방안

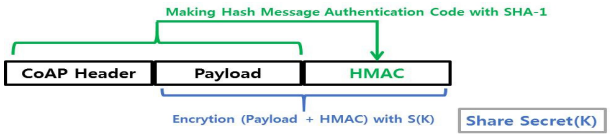
3.1 제안 네트워크



(그림 5) 제안 방안의 네트워크 구성도

본 논문에서 제안하는 방식의 네트워크 구성도는 그림 5와 같으며 Client와 S-Broker는 기존 방식과 동일하게 Secure한 연결로 구성한다. 다만, S-Broker와 센서단의 방식은 CoAP에 정의된 NoSec방식으로 구현하되 제안하는 메커니즘을 활용하여 보안강도를 보완한다.

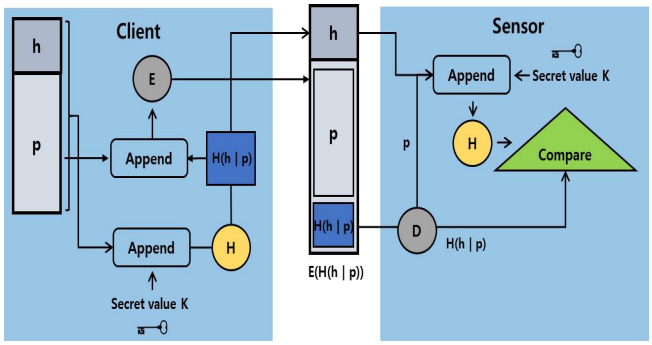
3.2 향상된 암복호화 검증과정



(그림 6) 제안하는 S-Broker를 활용한 방식의 메시지 구조

제안하는 S-Broker를 활용한 CoAP 통신방식의 메시지 구조는 그림 6과 같으며 동작방식은 다음과 같다.

1) 사전에 공유된 Share Secret Key(S(K))을 메시지에 첨부하여 해시한 HMAC을 생성한다.
2) 생성된 HMAC을 메시지의 Tail부에 부착한다.
3) Payload와 HMAC에 대해 S(K)를 활용하여 암호화를 수행한 후, 패킷을 전송한다.
4) 복호화 및 HMAC 검증의 과정은 위 과정의 역순으로 진행되며 해시값 비교를 통해 무결성 검증을 수행한다.



(그림 7) 제안 방식의 검증 메커니즘

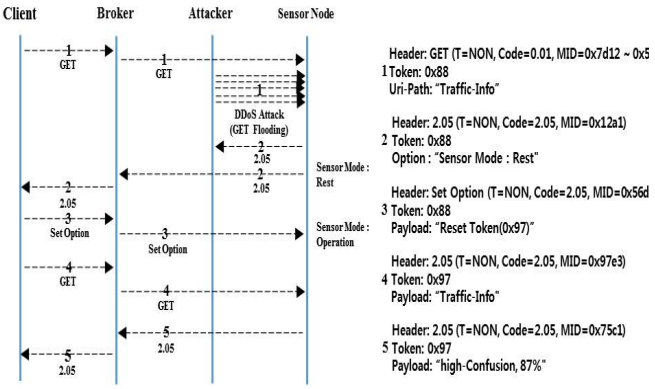
제안방식의 암호화 검증 메커니즘은 그림 7과 같으며 본 방식을 통해 메시지 검증과 더불어 사용자 검증이 가능하다. 해더와 해싱된 페이로드에 대한 해싱값과 HMAC값의 비교를 통해 소규모 연산으로 메시지의 무결성 검증이 가능하다. 또한, 사전 공유된 S(K)값을 통해 메시지 복호화가 성공하면 인증된 사용자임이 검증가능하다. 추가로 메시지의 암호화를 통해 Payload와 HMAC 대한 보안강도를 높일 수 있었다.

3.3 DoS 대응을 위한 동작과정

제안하는 방식은 end-to-end단인 센서 노드의 DDoS 공격에 대한 대응방안이다.

먼저 Token이 탈취된 상황에 대해 센서에서 accept/sec 단위의 GET Flooding 공격 식별을 수행한다. 식별이후 S-Broker에게 센서의 상태 Option을 전달하고 센서는 Rest Mode로 동작한다. 이후 서버로부터 Token 재 할당이 이루어지면 다시 Operation Mode로 동작을 수행한다.

S-Broker의 동작은 Proxy와 유사하다. 동일한 응답에 대한 반응성을 높인 cache 기능과 신속한 Forward 기능을 통해 성능 저하를 감소시켰다. 기존의 Proxy와 다르게 NoSec 환경에서 DTLS를 적용하지 않고 Broker에서 설정된 공격을 식별하여 센서 노드에 대한 통제를 수행한다.



(그림 8) DDoS(GET Flooding) 공격에 대한 센서 노드 대응방안

Secure-CoAP over Nosec 프로토콜의 메시지 흐름은 위 그림 5과 동일하며 동작방식은 다음과 같다.

1) 일반적인 Client의 GET 요청에 대하여 S-Broker(Proxy)를 통해 Forwarding된 GET 요청을 Attacker가 스니핑한다. 이때 획득한 센서 노드 정보와 Token 값을 바탕으로 GET 요청을 위조하여 GET Flooding 공격을 수행한다.
2) DDoS 공격을 식별한 센서 노드는 Mode를 Rest로 전환하고 NON 메시지를 통해 자신의 상태가 Rest 상태임을 Client로 전송한다.
3) 응답을 받은 Client는 새로운 Token값을 설정하여 Payload를 통해 센서 노드로 전송한다. 응답의 신뢰성을 위해 RFC에 정의된 Retransmission 파라미터를 활용하여 재전송 과정을 거친다.
4), 5) 변경된 주소의 센서 노드로 GET 요청을 수행하면 해당 시간에 대한 Traffic값이 Non 메시지로 응답된다.

4. 시뮬레이션

본 논문에서 제안하는 방식에 사용된 시뮬레이션의 환경은 다음과 같이 구성된다.

- CPU : Intel(R) Core(TM) i5-4590, 3.30GHz
- RAM : 8G
- HDD : 1000G
- OS : Window 7 Ultimate k 64bit version

시뮬레이션 과정은 다음과 같다.

1) CoAP 방식의 Client와 센서 노드(Server)를 구축한다.
2) Client와 Server 중간에 S-Broker를 구현하여 Caching 및 Forward 기능을 적용한다.
3) S-Broker 함수에서 라이브러리 함수 호출을 위한 포트번호는 '8080'으로 임시 지정한다.
4) Server는 Client의 요청 정보를 검증한 후, 검증이 완료되었음을 알린다.
5) Client에서 Server까지의 S-Broker를 통해 전송되는 패킷의 유효성 검증에 필요한 시간을 측정한다.

```
9월 16, 2016 12:46:53 오후 org.eclipse.californium.core.network.EndpointManager createDefaultEndpoint
정보: Created implicit default endpoint 0.0.0.0/0.0.0.0:49728
2.05
{"Content-Format":"text/plain"}
Hello World!

ADVANCED

current time : 2016-09-16, 12:46:53
==[ CoAP Response ]=============================================
MID   : 39993
Token  : d5faf4
Type   : ACK
Status : 2.05
Options: {"Content-Format":"text/plain"}
Payload: 12 Bytes
---------------------------------------------------------------
Hello World!
===============================================================
Execution time : 0.173(sec)
```

(그림 9) CoAP 프로토콜의 기본 동작 시간 측정

```
97  public void SecureBroker() throws IOException {
98
99      String Contents = "";
100
101     ProxyHttpServer httpServer = new ProxyHttpServer(8080);
102     httpServer.equals(Contents);
103 }
104
105
106
Problems  Console
ExampleCrossProxy [Java Application] C:₩Program Files₩Java₩jdk1.8.0_77₩bin₩javaw.exe (2016. 9. 17. 오후 2:51:38)
9월 17, 2016 2:51:40 오후 org.eclipse.californium.proxy.HttpStack$HttpServer <init>
정보: HttpStack listening on port 8080
9월 17, 2016 2:51:40 오후 org.eclipse.californium.proxy.HttpStack$HttpServer <init>
정보: HttpStack started
9월 17, 2016 2:51:40 오후 org.eclipse.californium.proxy.HttpStack$HttpServer$1 run
정보: Submitted http listening to thread 'HttpStack listener'
CoAP resource "target" available over HTTP at: http://localhost:8080/proxy/coap://localhost:PORT/target
Execution time : 1.104(sec)
```

(그림 10) S-Broker를 구현한 CoAP 프로토콜의 동작시간 측정

기존의 CoAP 프로토콜의 동작은 센서와 클라이언트가 직접 통신하는 방식이다. 저전력의 제한된 환경에서 신속하게 동작 가능하나 보안적인 부분은 매우 취약하다. 추가적인 보안을 위한 요소가 필요하다.

이에 기존의 CoAP 프로토콜 동작에서 S-Broker를 추가하여 데이터 전송 간에 무결성을 확보하였다. 그림 6과 그림 7은 S-Broker 적용 전, 후의 검증에 대한 응답측정 시간이다. 속도의 감소가 발생하나 동일 요청에 대한 Caching 및 Forwarding 기능을 통해 지연감소 및 성능의 향상을 기대할 수 있다.

5. 결론 및 향후 연구 방향

본 논문에서는 기존의 CoAP 통신방식에 대한 보안 취약점 분석을 수행한 후, NoSec 환경에서 S-Broker를 적용한 저전력 환경에 특화된 제안 방식을 통해 무결성 검증과 보안 강도의 향상에 대해 연구하였다. 기존의 무겁고 복잡한 Handshake 방식을 수행하는 DTLS대신 NoSec 모드에서 S-Broker를 통해 센서 단에서 가장 필요한 무결성 검증과 더불어 저전력 환경에서 효과적인 경량 암호화를 통해 보안강도를 높일 수 있었다.

센서 노드는 매우 제한된 저전력 환경에서 동작하기에 연산속도 등의 많은 제한이 따른다. 제한된 환경의 극복을 위해 추가적인 복잡한 보안 프로토콜을 적용하는 대신 기본 모드(NoSec)에서 S-Broker를 추가하여 저전력의 제한된 환경을 보완하였으며 DTLS의 복잡한 방식을 경량화시켜 필수적인 무결성과 최소한의 보안강도를 보장하기 위한 요소만을 제안하였다.

S-Broker의 적용을 통해 보안 강도의 향상 외에도 Proxy의 Caching과 Forwarding 기능을 적용하여 전력소모 감소, 대역폭 절약, 동작시간 최소화의 성능 향상을 기대할 수 있을 것이다.

마지막으로 센서단에서 DoS 공격에 대한 간략한 대응 방안을 제안하였다. 향후 연구과제로는 제한된 저 전력 환경에서 센서 노드에 대한 토픽기반의 데이터 통신 및 암호화 강도 분류 기술이 연구될 예정이다.

Acknowledgement

이 논문은 2016년도 한국인터넷진흥원의 지원을 받아 고용계약형 정보보호 석사과정 인력양성사업의 일환으로 수행된 연구임

참고문헌

[1] Giulio Peretti, CoAP over DTLS TinyOS Implementationand Performance Analysis, University of Padova, 2013.

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

[3] G.Selander, F. Palombini, K. Hartke, Requirements for CoAP End-To-End Security, Draft-hartke-core-e2e-security-reqs-01, 2016.

[4] M. Koster, A. Keranen, J. Jimenez, Publish-Subscribe S-Broker for the Constrained Application Protocol (CoAP), Draft-koster-core-coap-pubsub-05, 2016.

[5] https://github.com/eclipse/californium