2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)

An Overview of Security in CoAP: Attack and Analysis

S Arvind
TIFAC-CORE in Cyber Security,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
arvind.sundar16@gmail.com

V Anantha Narayanan
Dept. of Computer Science Engineering,
Amrita School of Engineering,
Coimbatore,
Amrita Vishwa Vidyapeetham, India
v_ananthanarayanan@cb.amrita.edu

***Abstract*— Over the last decade, a technology called Internet of Things (IoT) has been evolving at a rapid pace. It enables the development of endless applications in view of availability of affordable components which provide smart ecosystems. The IoT devices are constrained devices which are connected to the internet and perform sensing tasks. Each device is identified by their unique address and also makes use of the Constrained Application Protocol (CoAP) as one of the main web transfer protocols. It is an application layer protocol which does not maintain secure channels to transfer information. For authentication and end-to-end security, Datagram Transport Layer Security (DTLS) is one of the possible approaches to boost the security aspect of CoAP, in addition to which there are many suggested ways to protect the transmission of sensitive information. CoAP uses DTLS as a secure protocol and UDP as a transfer protocol. Therefore, the attacks on UDP or DTLS could be assigned as a CoAP attack. An attack on DTLS could possibly be launched in a single session and a strong authentication mechanism is needed. Man-In-The-Middle attack is one the peak security issues in CoAP as cited by Request For Comments(RFC) 7252, which encompasses attacks like Sniffing, Spoofing, Denial of Service (DoS), Hijacking, Cross-Protocol attacks and other attacks including Replay attacks and Relay attacks.**

In this work, a client-server architecture is setup, whose end devices communicate using CoAP. Also, a proxy system was installed across the client side to launch an active interception between the client and the server. The work will further be enhanced to provide solutions to mitigate these attacks.

***Index Terms*—Internet of Things (IoT), Constrained Application Protocol (CoAP), Datagram Transport Layer Security (DTLS), User Datagram Protocol (UDP), Denial of Service (DoS) attacks, Man-In-The-Middle attack, proxy**

I. INTRODUCTION TO SAFE AND SECURE INTERNET OF THINGS

Internet of Things (IoT) is pigeonholed by various technologies, which is in accordance with the provisioning of state-of- the-art services in various application domains. The Internet of Things (IoT) has immensely changed the way we view, use and interact with smart devices, particularly in the business world. The way of utilizing the existing and emerging technologies for sensing, networking and automating has publicized the fact that it brings major changes in the delivery of products, goods and services, and thereby enhancing the socio-economic impact of these changes. It is anticipated that at least 50 billion 'things' will get linked to the Internet by 2020 [4, 6].

For all the amenities that IoT has afford us, there is a point of concern that every user must contemplate, which is the security aspect. Anything which is connected to the internet has the probability of getting hacked or exploited. It has been reported by security researchers that many solutions deployed are insecure and have many open security vulnerabilities. Also, the complex architecture design of the IoT Environment is being quoted as one of the foremost reasons for the everyday increase in attack vectors [8, 7].

IoT mainly makes use of the standard protocols and networking topologies. But the deployment and the type of the protocol being used largely determines the normal functioning of any IoT system. Some of the popular data protocols used are Constrained Application Protocol (CoAP), Message Queuing Telemetry Transport (MQTT), Simple Network Management Protocol (SNMP), Advanced Message Queuing Protocol (AMQP), OPC Unified Architecture (UA) [8].

Some of the smart devices may be using obsolete protocols, which may not have been updated often. This leads to the identification of backdoors and thereby the device getting hacked easily. One of the recent security hacks was that of the MQTT and the CoAP Protocols, where the sensitive information such as user credentials, the messages transmitted between devices and the device details were leaked. Such attacks may be attributable to the following reasons:-

- Connecting to unsecured networks
- Having a weak web interface
- Insubstantial encryption mechanisms used
- Dated protocols and malicious Software updates

Though many security measures such as the DTLS and TLS, and other end-to-end authentication and encryption mechanisms have been implemented in CoAP and the architecture operates on the HTTP/REST interactions, the protocol is still susceptible to risk engaging attacks. Security risks in CoAP data protocol necessitate attention as they are being widely used in Device-to-Device communications.

This paper tries to project a new dimension by triggering an amendment in the temperature and humidity data, using

978-1-5386-9533-3/19/$31.00 ©2019 IEEE

the CoAP proxy model that could deliver a skewed result altogether in applications like healthcare monitoring.

The paper is structured as follows: Section II provides a general outline of the Constrained Application Protocol; Section III analyzes the other research works relating to this paper ; Section IV reveals the results of performing an attack on the CoAP Protocol along with the simulation setup; Section V concludes the paper with the future work and its implementation thought.

II. AN OUTLINE OF COAP

A. Introduction

Designed as a lightweight machine-to-machine (M2M) protocol, the CoAP can run on internet-connected devices where memory and computing resources are scarce.

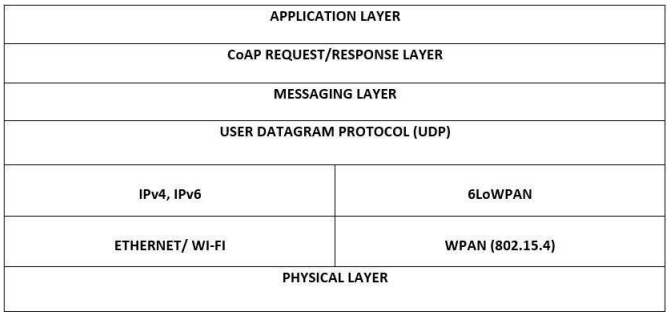

Fig. 1. General CoAP Architecture

This application layer protocol is mainly used in constrained environments such as the wireless sensor networks. The main goal of CoAP is to provide a common web interface for explicit requirements of the constrained devices. Some of the features of the protocol include:-

- User Datagram Protocol (UDP)
- Integration of IPv6 with 6LoWPAN
- Support for both unicast and multicast communications
- Client/Server Architecture
- Achievement of less overhead
- Proxying and caching capabilities
- Adoption of DTLS for Security
- Usage of Uniform Resource Indicator (URI) Methods and HTTP Mapping mechanisms [11, 14]

B. The CoAP Interaction Model

The protocol operates with a Client/Server model in which a Request is made by the client and a Response is generated from the Server. This Request/Response functionality is aided by method codes or response codes and Token options which are used to match the response to the requests. Each Response matches with the original message using a unique message ID, and there are three message types:-

- Confirmable message, which require an acknowledgement to be sent back to the requesting client
- Non-Confirmable messages, for messages that does not require an acknowledgement to be sent
- Acknowledgement messages indicate that an explicit Confirmable Message has been received
- Reset messages indicate that a Confirmable message has been received, signifying a missing context to process the incoming message [12]

C. Message Format

The messages which are being communicated between the end devices are encoded in a binary format. Each message consists of a Header with fixed length, trailed by an options field and the payload field whose length is derived from the datagram length.

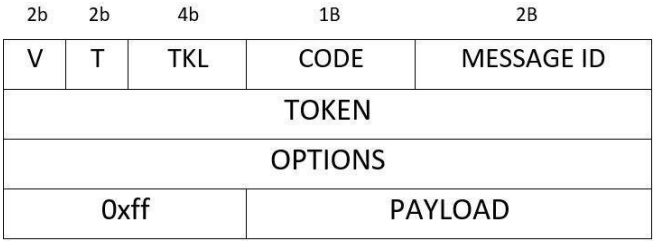

Fig. 2. CoAP Message Format

V - Version Number
T - Type of the message
TKL - Token Length
Code - CoAP Request/Response Code

D. Proxying and Caching

One of the aims of an IoT system is to shrink the response time and thereby reduce the network bandwidth consumption. For this purpose, a suitable caching mechanism is implemented, reusing an earlier response message to satisfy a current request with a prerequisite that a CoAP end-point must not use a stored response. To determine the freshness of a message in the cache, a message expiration time is also set to play a fundamental role in enhancing the security of the messages being transmitted. A distinctive way of sending the requests is through a proxy, where messages are transmitted from the proxy on behalf of the end-points. A Proxy uses a store and forward approach to accumulate the requests and send it to the end-point on an ad-hoc basis [11, 13].

E. Method Definitions

Unlike a network request/response communication, where a HTTP session is initiated, the CoAP works on the Representational State Transfer Model (REST). In this model, resources can be accessed from an URL and clients make use of these resources using the standard HTTP methods such as GET, PUT, POST and DELETE [11].

F. Mapping

The table 1 describes some of the most frequently generated HTTP Response codes with which each CoAP response gets mapped to the request [11].

TABLE I
RESPONSE CODES

HTTP Response Codes	CoAP Response Codes
200 – OK	2.00 - OK
201 – Created	2.01 - Created
400 - Bad Request	4.00 - Bad Request
403 – Forbidden	4.03 - Forbidden
404 - Not Found	4.04 - Not Found
501 - Not Implemented	5.01 - Not Implemented
504 - Gateway Timeout	5.04 - Gateway Timeout

G. Resource Discovery

Finding the correct end-points and establishing a communication with these devices have always been a challenge in the IoT Systems due to the conceivable intervention by malicious intruders. The discovery of resources in CoAP supports the CoRE Link Format as described by IETF Group. Ultimately, the server decides which resource should be made available to and discoverable in the network. All the resources are located in the core directory '/.well-known/core ', which is the URI scheme used in CoAP [11].

H. Binding DTLS with CoAP

With the HTTPS transactions are safeguarded by Secure Sockets Layer (SSL) and the Transport Layer Security (TLS) as an encryption layer, the CoAP is also protected by Datagram Transport Layer Security (DTLS). The RFC 6347 describes the design of TLS over datagram transport. It is being preferred for implementation in CoAP and it differs from TLS by the following parameters:-

- Encryption state is not created, due to the modification in the records
- The messages are being transmitted in fragments
- Data alterations can be endured, which employs a threshold mechanism to withstand the extent of errors
- Addition of sequence numbers
- Mapping of DTLS records with a datagram

I. Security in CoAP

Although security mechanisms have been instigated in CoAP, it does suffer from common attacks including:-

- Parsing attacks, in which a remote node could be crashed by executing an arbitrary code on the node
- Caching attacks, wherein a proxy having the ability to cache can gain control. This may serve as a threat for clients who are exchanging data with the proxy, unknowing a possible intruder in the network
- The amplification attacks, in which an attacker can use the end devices to convert a small packet into a larger packet. A CoAP Server can in fact reduce the amplification attacks by using Blocking/Slicing modes. But amplification attacks are still on the rise, and according to [5], it has been estimated that the amplification factor of CoAP can go up to 32, which means that an attacker who has access to a 1 Mbps Network connectivity can target another link which is equipped with a capacity of 32 Mbps
- Spoofing attacks
- Cross-Protocol attacks, where the translation from TCP to UDP is liable to attacks [11]

J. CoAP Implementations

There are various methods of executions of CoAP, based on the project requirements. The most popular implementations are depicted in table 2:-

TABLE II
CoAP Tools

Constrained Devices	Erbium, libcoap, tinyDTLS, wakaama, SMCP
Server Side	Californium, CoAPSharp, Erlang, aiocoap, etc
Browser Based	Copper
Android	Aneska

III. Related Works

- Paper [1] discusses the various ongoing security issues with distinct reference to CoAP. An outline of certain IoT protocols like the 802.15.4, 6LoWPAN and RPL were discussed along with their security issues. However, the paper highlighted some of the key security solutions that CoAP currently uses, including the heavyweight DTLS Protocol and the exploitation of 6LoWPAN in header compression.
- The work in [2] forms the basis for establishing a CoAP environment. The paper mainly focuses on analyzing the security of IoT systems in the healthcare applications, whose sole purpose is to derive the status of patient's health condition through a web browser. For the purpose, a CoAP Architecture was setup using the Cooja simulator to simulate the client-server communication, which produces a resource containing information about the patient's health status. Here, the Copper browser which is an add-on service provided by Firefox for RESTful services, was used.
- The work of an intermediate entity called proxy was discussed in [3], highlighting the importance of caching and translation of communication between differentprotocols. The paper also discusses the enactment of proxies as an independent entity, its mapping from HTTP to CoAP and vice-versa and the use of certain CoAP libraries like Californium.

IV. Security Testing and Results

A. Initial Setup

We used the DHT11 Sensor and the ESP8266 to sense the temperature in the surroundings and generate a series of outputs on the Arduino serial monitor. The DHT11 sensor is chosen to produce digital output. It can be integrated with any microcontroller like Arduino, Raspberry Pi, etc. Keeping in mind the cost implications, the DHT11 sensor provides the much needed reliability and stability. On the other hand, the NodeMCU (ESP8266) is a low-cost Wi-Fi chip that supports serial communication protocols like Universal Asynchronous Receiver/Transmitter (UART), Inter-Integrated Circuit(I2C),

etc. The ESP8266 was connected to a guest Wi-Fi, to which the Client was also connected.

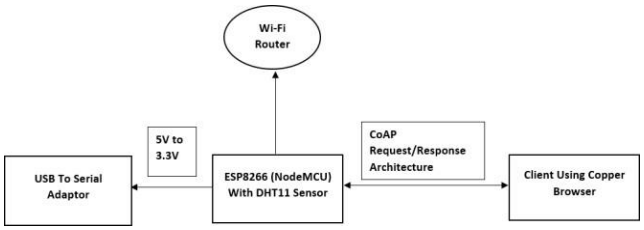

Fig. 3. Initial setup of a CoAP Architecture

The CoAP Code is executed in the Arduino console. The port is set to COM3. The initial temperature and humidity values are displayed with a Baud rate set to 115200.

B. *Copper framework Output*

Copper was designed as a browser add-on for the Internet of Things. The concept of Browser for the Internet of Things (BIT) has been very helpful in providing a platform for communicating with the constrained devices. In this work, the Firefox version is downgraded to version 50 and the Copper add-on is downloaded from Firefox. The Copper browser is started by typing 'coap:// 'and in the address bar, an IP address of the server is generated. When the Resource Discovery option is clicked, we get an output as shown in Fig. 4.

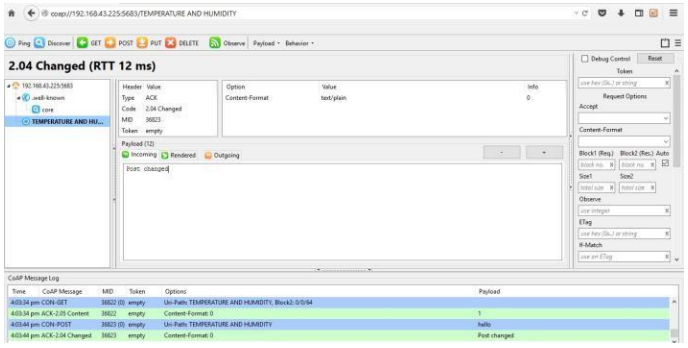

Fig. 4. Copper Browser Output showing Resource Discovery

From the above result, it could be observed that the re- source is discovered as "TEMPERATURE AND HUMIDITY", whose values are received from the Arduino console. The Copper browser is also equipped with an option to change the request commands to the server. When the request for change is posted in Copper, a response code "2.04 Changed" is received with a Round Trip Time (RTT) of 12ms.

C. *Contiki*

When designing an Operating System (OS) for the Internet of Things, the most important aspects that should be concentrated are the memory size, power of the constrained devices and their processing capabilities. Contiki is one such operating system which is specifically designed for resource constrained devices such as the sensor nodes. It is built with an event-driven kernel that supports pre-emptive multithreading. This multithreading approach is useful when the execution of thread has to be interrupted by another thread, in event driven situations like handling network packets or disk drives, etc.

D. *Cooja simulator*

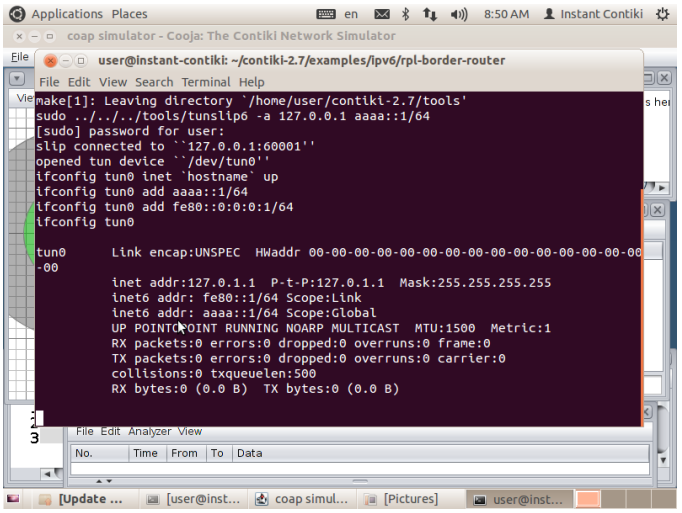

Fig. 5. Cooja Simulator compiling the Server Code

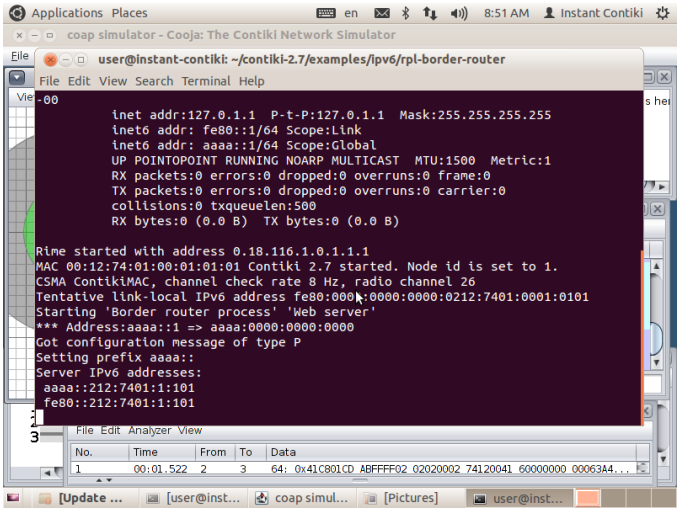

Fig. 6. Cooja Simulator compiling the Border Router

Cooja is a network simulator, used in Contiki for simulation purposes. Cooja has been developed in JAVA and its goal is to provide extendibility by using interfaces and plugins. An interface represents a sensor node and the plugin is used to cooperate with the simulation. Java Native Interface (JNI) is used to link the simulator with Contiki thereby allowing applications to run in Contiki. This approach has laid the foundations for applications to run on a real sensor node intact. For this research work, the simulation results are obtained by starting the Contiki OS and the Cooja Simulator. The CoAP setup mote is connected and the server code is run on

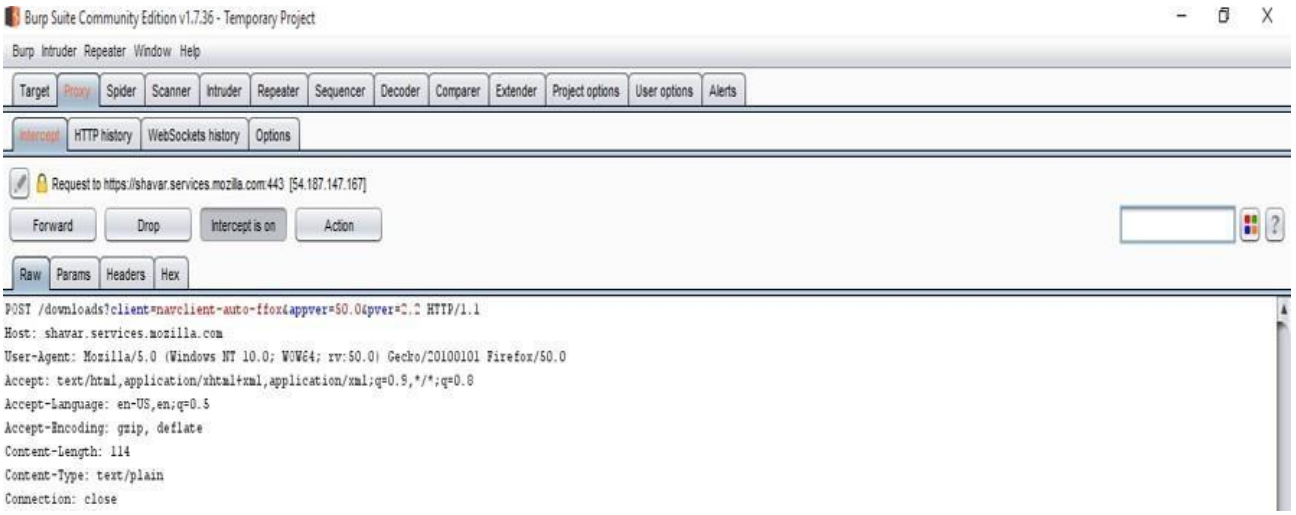

Fig. 7. Burp Suite output showing the interception of communication

one terminal. On another terminal, the Border Router code is compiled using the Routing Protocol for low power and lossy networks (RPL). These RPL border routers are required to be connected from one network to another and they are typically used in Wireless Sensor Networks (WSN). Now both the codes are compiled on their respective terminals and outputs obtained are shown in Fig. 5 and Fig. 6.

An IPv6 Addresses is received from the border router terminal, which when pinged in the Copper browser displays a list of resources the server offers. In our case, the copper web page with the "TEMPERATURE AND HUMIDITY" resource gets displayed.

E. The Attack

The attack was launched by setting up a proxy server. The proxy basically acts as an intermediary between the client and server and it allows clients to make subsidiary connections to the outside network services. The Eclipse Californium is one such example of a java-based proxy implementation, whose central focus is to achieve scalability and usability, apart from resource-efficiency [15].

The Californium proxy was imported in to the Eclipse IDE, which provides APIs for initiating RESTful web services. The CoAP client is run on Mozilla Firefox with support from Copper and Californium. Subsequent to requesting the resources, the client gets access to the "TEMPERATURE AND HUMIDITY" resource from the server.

To test the security of the protocol, the community edition of Burp Suite is installed. This tool helps in performing active web penetration testing. The communication between the client running the proxy and the server is intercepted with the help of this tool. The Burp Suite is configured to localhost and the connection is captured as shown in Fig. 7.

The result shows that the information is transmitted in plain text format, indicating the likelihood of attacks on the protocol.

V. CONCLUSIONS FOR FUTURE APPROACH

The Internet of Things is considered as one of the biggest jumps towards a technically strong future. It is imperative to have a secure IoT system to develop and adopt this technology in our daily life. In this work, one of the security issues pinpointed in the protocol - sniffing attack - is performed on a test network with a CoAP client-server and a proxy model. By this attack, we are able to gain Intel on the type of information being shared between the client and server. The future work will focus on integrating Object Security for Constrained RESTful Environments (OSCORE) with CoAP, which will further enhance the end-to-end security in CoAP. In addition to OSCORE, the work will also add a layer of access control mechanism to prevent the possible intervention of malicious third party in the network. While threats to IoT are on the rise, an efficient security testing has become a paramount importance particularly in CoAP requiring a deep and wide research.

REFERENCES

[1] R. A. Rahman, B. Shah, "Security analysis of IoT protocols: A focus in CoAP", 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), pp. 1-7, 2016.

[2] Kanchana P. Naik, U. Rakesh Joshi, "Performance analysis of constrained application protocol using Cooja simulator in Contiki OS", 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT), pp. 547 - 550, 2017.

[3] J. Esquiagola, L. Costa, P. Calcina, M. Zuffo, "Enabling CoAP into the swarm: A transparent interception CoAP-HTTP proxy for the Internet of Things", 2017 Global Internet of Things Summit (GIoTS), pp. 1 - 6, 2017.

[4] Praveen Vijai, Bagavathi Sivakumar, "Design of IoT Systems and Analytics in the context of Smart City initiatives in India", 2nd International Conference on

Intelligent Computing, Communication and Convergence (ICCC-2016), Procedia Computer Science 92 (2016), pp. 583 588.

[5] Federico Maggi, Rainer Vosseler,Davide Quarta, In the document ”The Fragility of Industrial IoTs Data Backbone Security and Privacy Issues in MQTT and CoAP Protocols”, Trend Micro-Research, 28 July 2018

[6] S Krco, B. Pokric, F. Carrez, ”Designing IoT architecture(s): A European perspective”, Proc. IEEE WF-IoT, pp. 79-84, 2014.

[7] Satyadevan S., Kalarickal B.S., Jinesh M.K. (2015) Security, Trust and Implementation Limitations of Prominent IoT Platforms. In: Satapathy S., Biswal B., Udgata S., Mandal J. (eds) Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014. Advances in Intelligent Systems and Computing, vol 328. Springer, Cham.

[8] L. Atzori, A. Iera, G. Morabito, ”The Internet of Things: A survey”, Computer. Networks, vol. 54, no. 15, pp. 2787- 2805, Oct. 2010.

[9] Jelena Mii, M. Zulfiker Ali, Vojislav B. Mii, “Architecture for IoT Domain with CoAP Observe Feature”, IEEE Internet of Things Journal, Volume: 5, Issue: 2, pp. 1196 - 1205, 2018.

[10] Shambhavi Mishra, Pawan Singh, Deepak Arora, Krishna Kant Agrawal, ”Analyzing and evaluating the performance of 6L0WPAN and RPL using CONTIKI”, Intelligent Sustainable Systems (ICISS) 2017 International Conference on, pp. 1100-1105, 2017.

[11] Z.Shelby, K.Hartke, C. Bormann ,June 2014. The Constrained Application Protocol (CoAP). Available from: https://tools.ietf.org/html/rfc7252.

[12] A. Rahman, Ed-Inter Digital Communications, LLC; E. Dijk, Ed-Philips Research Group October 2014. Communication for the Constrained Application Protocol (CoAP). Available from: //tools.ietf.org/html/rfc7390.

[13] A. Ludovici, A. Calveras, A. Calveras, ”A proxy design to leverage the interconnection of CoAP wireless sensor networks with web applications”, Sensors (Switzerland), vol. 15, no. 1, pp. 1217-1244, 2015.

[14] Shahid Raza, Hossein Shafagh, Kasun Hewage, Ren Hummen, and Thiemo Voigt, ”.Lithe: Lightweight Secure CoAP for the Internet of Things”, IEEE sensors journal, vol. 13, No.10. October 2013.

[15] Best practices for http-coap mapping implementation, https://tools.ietf.org/id/draft-ietf-core-http-mapping-01.html, last accessed: January 2016.