Using REST based protocol to enable ABAC within IoT systems

Marwah Hemdi
Department of Computer Science
University of Saskatchewan
Saskatoon, SK, Canada
Mah411@mail.usask.ca

Ralph Deters
Department of Computer Science
University of Saskatchewan
Saskatoon, SK, Canada
deters@cs.usask.ca

***Abstract*— with the development of the Internet of Things (IoT) and the usage of low-powered devices (sensors and effectors), a large number of people are using IoT systems in their homes and businesses to have more control over their technology. However, a key challenge of IoT systems is data protection in case the IoT device is lost, stolen, or used by one of the owner's friends or family members. The problem studied here is how to protect the access to data of an IoT system. To solve the problem, an attribute-based access control (ABAC) mechanism is applied to give the system the ability to apply policies to detect any unauthorized entry. Finally, a prototype was built to test the proposed solution. The evaluation plan was applied on the proposed solution to test the performance of the system.**

Keywords—IoT; ABAC; COAP

I. INTRODUCTION

The Internet of Things (IoT) has the ability to connect different devices such as sensors (small devices that can be programmed to complete a specific task) or mobile devices together, which each will complete a task. IoT is like a network of networks. Additionally, the term IoT, which defined in [16] as "a global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies." The IoT allows objects (connected devices) to detect and control remotely across existing network structure, producing chances for more direct incorporation between the physical world and computer-based systems. Furthermore, the sensors became important tools to complete simple tasks in the real world. Sensors were defined as "Sensors are used to measure physical quantities such as temperature, light, pressure, sound, and humidity." [1] In general, developers can program the sensors to predict a particular change in an environment or recover physical information.

[1] http://www.bbc.co.uk/schools/gcsebitesize/ict/measurecontrol/0computercontrolrev2.shtml.

II. PROBLEM DEFENITION

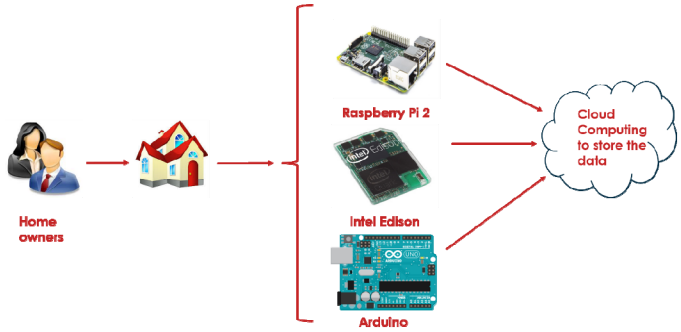


Figure II—Problem definition

Fig II demonstrating the problem description of this research, people building IoT systems in their houses by installing sensors in them. Three different sensors were put as examples:

1. Raspberry Pi 2 (the one is used in this project)
2. Intel Edison
3. Arduino

Nowadays, sensors and effectors gaining popularity; especially, after using them in IoT systems. People now have sensors in their homes; they want to use the technology for their own benefits such as or reduce their energy costs. For instance, a homeowner has an IoT system, which uses sensors and effectors to achieve the next:

1. Recover the temperature in the house.
2. Turn off or turn on the cooling or the heating system.
3. Modify the system settings.

In the previous situation, the system will allow the users three operations:

1. READ to retrieve the temperature.
2. WRITE to switch the system on or off.
3. UPDATE to alter the settings.

978-1-5090-0996-1/16/$31.00 ©2016 IEEE

Therefore, such a scenario could be applied also to a commercial environment like a hotel offering the guests IoT application to the guests to change the room temperature while they are out of the room, so the hotel saves more energy.

Either IoT system was in a house or a business, it means that the connected device takes some data as an input, process it, and store the output in a database (locally or in the cloud). Hence, offering an IoT services and provide a mobile application to the users and give them the permission to use their own personal devices to access and alter the data anytime is risky in case the wrong person accessed the database.

Some of the main challenges that related to IoT systems are:

1. The security of the information: allowing the users to access and alter the data anytime and anywhere using any of their personal devices is risky. Because anyone of the user's partner, kids, or friends (who knows the signing in credentials of the application) can access, read, write, or update imperative information.

2. The irresponsibility of the users: giving the users the permission to access and use the data could lead to excessive complications if the user was careless. For instance, in the example of allowing the user to change the heating or cooling system, what if it was in the classroom and the students have an application to edit the temperature as they longing, the problem that would happen here is one student is cold and another thinks it is warm. In this case, they will keep changing the temperature, and this type of behavior could cause a massive damage. In another way, the users' behavers could minimize the security of the device that is used to access IoT services; like if they want to download forbidden software, they will change the security setting of the device, and that could cause prodigious problems.

This research is focusing on applying Attribute Based Access Control mechanism on a prototype of IoT system in different ways:

1. Having the client and server both in Raspberry Pi.

2. Using Raspberry Pi 2 as a proxy while the client is a web page and the server created on a cloud platform.

This research is trying to solve the problem of preventing an illegal entry to an IoT system by applying the proposed solution mainly in a low power device, which is Raspberry Pi2. By evaluating some of the users' attributes; specifically, accessed time, device's MAC address, username, and password to detect if the user who is trying to access the services is a permitted or not.

III. Literature Review

A. Attributes Based Access Control (ABAC)

Access Control (AC) definition stated by [1] describing AC as the process that controls every entree to a system and its resources, and allowing only the authorized requests to access the data. Additionally, [2] explained that AC approaches to guarantee the protection of the data from unauthorized expose or alteration. Besides, access control methods control how users work together with data and other network properties.

Many developers defined ABAC based on their researching and understanding of it such as [3] stated that ABAC is a model "based on subject, object, and environment attributes and supports both mandatory and discretionary access control needs." Also, [4] had their own definition of ABAC, which is "An access control method where subject requests to perform operations on objects are granted or denied based on assigned attributes of the subject, assigned attributes of the object, environment conditions, and a set of policies that are specified in terms of those attributes and conditions". To summarize ABAC previous definitions and more, when a user requesting the access to a specific data, the system must apply policies and conditions on the users and/or the object and/or the environment attributes. If the attributes meet those conditions, the system will allow the user the access to the data.

Attributes in ABAC could be considered as characteristics of anything (in the subject or object) that might be defined and to which a value might be assigned. ABAC depends on the evaluation of attributes of the subject, attributes of the object, environment conditions, and predefined policies (by the system originator) to allow actions for the subject. All ABAC applications include the abilities to evaluate attributes (subject and object) and environment. Furthermore, the subject should not be able to change his or her own authorization attribute value because only security authorities (system creator) should be able to provide and declare the attributes, and attribute values based on authoritative permissions of the object [4].

Next, some researchers suggested ABAC models for an IoT system. For example, Kaiwen and Lihua introduced in [5] a model that combines two AC methods, RBAC and ABAC because they wanted to assign roles for each user. They created the term Attribute-Role-Based Hybrid Access Control to describe their system; the idea for IoT system is to evaluate a number of users' attributes every time the user sign in, then depending on the evaluation result the system will apply a specific role to each user with particular permissions.

Other developers used ABAC more as an encryption technique, and it is known as Attributes-Based Encryption (ABE). For instance, Wang et al research work in [6], Their system implementing the basic ABE process from generating keys to encryption to lastly a decryption. In details, "An ABE system usually consists of a key authority, publishers [senders] and subscribers [recipients]. The key authority authenticates publishers and subscribers (verifies they are who they say they are, as well as their attributes), generates public/private keys, and issues the keys to publishers and subscribers" [6].

Some of ABAC advantages, which make it one of most important AC models and an appropriate model for this research [5]:

- In ABAC, the entities such as the user's name, password, or location are called attributes, and there are policies must be applied to those entities. The attributes are changeable, yet the policies are static; thus, ABAC is different from other models because it controls access to

objects by evaluating rules against the subject attributes.

- ABAC is best used in situations where users are dynamically changing.
- ABAC is a flexible model since it does not need to define a prior relationship between the subject and the object. In case the object owner wants to change the access decision, simply he or she needs to change the access policies or the attributes' values.

B. *Security challenges*

In [7] some of the security risks that could happen in each layer of the IoT structure:

1) Perception Layer: With all the available technology, hackers could spy on a communication using software like malware. The attackers aim at this layer because they know it is the fundamental part of the IoT (this layer's basic function is to perceive and collect information). Therefore, attackers can easily obtain and explore the data from this layer to capture the users' information, and that might cause excessive damage.
2) Network Layer (transport layer): One of the most known features of the IoT is its vast amount of data. Thus, this layer is primarily responsible for transferring data between the Internet and the mobile telecommunication network and so on. Because a huge amount of data is transmitted between the perception layer and the application layer, the network layer will unavoidably generate enormous numbers of redundant data. Consequently, the developers must add the filtration devices between the network layer and the application layer to avoid service attacks and leave the network unblocked.
3) Application Layer: This part achieves the main goal of developing the IoT system, which makes the users' lives smarter and decreases the workload. The biggest security challenge in this part is protecting the users' data, which includes their confidential information.

C. *RESTful services*

The Web is a universal ecosystem to apply all kind of operations on applications and services; it makes us able to search, transfer, cache, replicate, and much more. One of biggest factors that made the Web essential is people; "human users are the direct consumers of the services offered by the majority of today's web applications [8]." Applications including the ones in IoT are a type of distributed systems because they have the ability to connect numerous devices in a network. Whatever a domain that the distributed system is a part of, the system's developer needs to send and receive data between two sides the client such as a web page or a mobile application and the server whether it was locally or in a cloud, and that could be accomplished that by using Web services.

According to [9], "Web services are a solution for the integration of distributed information systems, autonomous, heterogeneous and self-adaptable to the context." The REST architectural is grounded on four principles [10]:

- Resource identification: Uniform Resource Identifier (URI) is the resource identifier. Each RESTful Web service represents a set of resources that classify the requests between the servers and the clients in the interaction; the transaction is based on the global address (URI).
- Uniform interface: the systems operate any resources are using a static set of CRUD operations (Create, Read, Update, and Delete).
- Self-descriptive messages: resources are separated from their representation so that their content can be retrieved in a diversity of formats (e.g. XML, plain text, PDF, and so on).
- Stateful interactions: every interaction with an item is stateless; in other word, requests are self-contained. Stateful interactions are founded on the impression of obvious state transfer, which hyperlinks could achieve.

Christensen [11] concluded that RESTful services improve the created application by surpassing the abilities of old-style smart devices. Over time, developers created protocols based on RESTful services, and they have great potential to build scalable connections due to the support for caching, clustering, and load balancing that are built into REST. All of this leads to a new generation of developing IoT applications with remarkable potential.

IV. Solution Architecture

With the growth and increasing usage of technology, the risks and threats are also growing. Today's systems including IoT systems store an enormous amount of private information about the users. Moreover, the process of storing data has changed; we now have cloud computing to do the job without needing paper or servers to maintain records. Moreover, IoT is now used everywhere, thanks to all the capabilities it offers such as sensors or Machine-to-Machine (M2M) services. Hence, the private information in organizations has become a target for malicious people. As a result, the number of security incidents and malware is rapidly increasing.

Nielsen believed that "Threats today are designed to be largely invisible, blending in with background noise and traffic, requiring a form of contextual-based security analytics to detect and identify them" [12]. Because the threats that target enterprises can be dangerous and subtle, developers need to think of new ways to protect and secure private information; one of these new ways is analyzing user context.

The objective in the proposed solution is to prevent any unauthorized access to an IoT system. This research is working

on the ability to secure the data by applying one of the access control types—ABAC. By evaluating some of the users' attributes; specifically, access time, MAC address, username, and password, the system will apply some of ABAC policies to make sure that the user who is trying to access the data is authorized to do so.

Figure IV—Proposed solution

A. Client Side

First, to build a client prototype for Raspberry Pi 2 implementation, a Web page was created using HyperText Markup Language (HTML), it is a markup language is a set of markup tags, and each HTML tag implements different functions; Web browsers can read and execute HTML files. However, because a low-power device (Raspberry Pi 2) is used, the solution was built as compactly as it could be; thus, CoAP was used as a web service. "One of the main goals of CoAP is to design a generic web protocol for the special requirements of this constrained environment, especially considering energy, building automation, and other machine-to-machine (M2M) applications [13]." In order to use CoAP, the system had to change the client (the Web page) to Copper (it is a CoAP client) because HTML does not have a CoAP library.

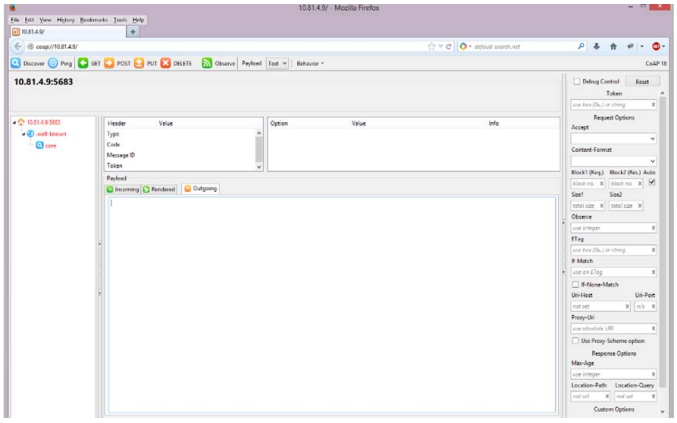


Figure IV.A— Copper (Cu) CoAP user-agent for Firefox web browser

B. Server Side

The server side in the Raspberry Pi 2 divided into two sections:

1. The proxy that handles sending the attributes to the database

The proxy was written using Golang because it has a good cross-platform support, concurrent and modern, and it has faster compiler than other languages like Java; also, it connects easily to MySQL.

Initially, this proxy is created to get a specific user attributes (username, password, and MAC address) from the client Cu) and send it to the database. Furthermore, a request handler function was created which apply CoAP protocol to get the required attributes from the users to send it to the database. To achieve that the entered attributes had to be parsed to the JSON (JavaScript Object Notation), then obtained the entered context in variables; subsequently, those attributes were sent to another function called POST. In POST, a connection with MySQL database was created. Afterward, an authentication step was written where the system applied a query to compare the username, password, and MAC address with the ones stored in the database.

2. MySQL database that stored in the Raspberry Pi 2

The database in this research contains number of tables:

- Users table: This is the main table because it is the one used for authentication, and all other tables are connected to this table using a foreign key. It includes all the usernames and passwords of the clients of an enterprise that wants to provide an IoT application.
- Mac_address table: This table includes all MAC addresses of the users' devices that they are using to sign in. Worth declaring here is that a client can use more than one device. Each mobile phone or tablet has a MAC address depending on the network connection with the device. In this table, all the MAC addresses of each user's devices were stored, and they have the user_id from the users table as a foreign key.
- Logs table: This table evaluates the context of each user who is trying to sign in. The login process starts in this implementation from the CoAP client. The system will send the username, password, and MAC address to the Golang proxy, which sends the attributes to the MySQL database to apply the ABAC. It starts with an authentication process, which compares these credentials to the usernames and passwords in the users table. If they exist, then the matching ID in the users table will be sent to the logs table. Correspondingly, when a client signs in, the new MAC address will be compared to the current ones in the mac_address table, and its ID will be sent and stored in the logs table to be evaluated. Moreover, this table includes a column called time, which is a timestamp to return the server time when the username, password, and MAC address entered the database. The server time was used because it does not change; even if the mobile device changed its time or location, the server time is unalterable.

After the required attributes' ID is entered and the time stored in the logs table, the logs contains t_result, u_result, p_result, and mac_result columns that return the evaluation of the accessed time, username, password, and MAC address. The

username results column has a trigger to set the word "YES" if the username ID was entered in the logs table from the users table; this also applied to the password results and mac results column. The evaluation of the accessed time is defined in an SQL trigger, which takes the time when the username, password, and MAC address entered the server and compares it to the assigned ABAC policies to test the prototype:

- If the accessed time is between 8 a.m. and 5 p.m., the result is "Yes" because these are work hours.
- If the accessed time is between 12 p.m. and 1 p.m., the result is "Maybe" because this is a lunch hour.
- If the accessed time is after 5 p.m. and before 8 a.m., the result is "No" because these are after work hours.

A function procedural was constructed using SQL commands to return the evaluation and access control results of each user's signing in operation to the application.

All the ABAC policies were implemented on the database side because having them on the application level would make the IoT system vulnerable to hackers. Additionally, the IoT application could be suitable on different platforms.

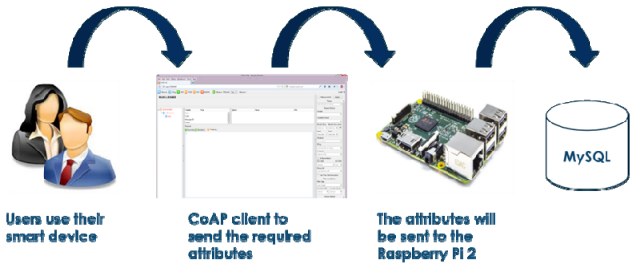


Figure IV.B— Implementation of the system

Finally, the POST function returns the word "Hello to you" in case the system sent the attributes to the database and the ABAC policies were applied on the entered context without any errors.

C. *Web services*

In this implementation, CoAP was applied as web service; CoAP is RESTful architecture based protocol; therefore, its resources act as server-controlled constructs require an identified Universal Resource Identifiers (URIs). It is used in low-powered electronics devices in another word (nodes), to make them capable of connecting interactively together; also, it allows them the ability to connect to the Internet. The Internet Engineering Task Force (IETF) CoRE Working Group has started the regularization activity on CoAP in March 2010. "Like HTTP, CoAP is a document transfer protocol. Unlike HTTP, CoAP is designed for the needs of constrained devices" Said [14]. Moreover, [14] Added "CoAP runs over UDP, not TCP. Clients and servers communicate through connectionless datagrams. Retries and reordering are implemented in the application stack. Removing the need for TCP may allow full IP networking in small microcontrollers."

Next fig presents the interactive structure model for CoAP; like an HTTP client/server model, it is a transfer protocol. However, CoAP has a bottom layer, which is messages layer that has been created to handle UDP and asynchronous switching. The upper layer is request/response layer is mainly for the communication method and handles request/response messages.

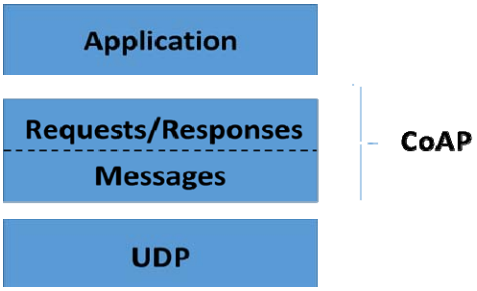


Figure IV.C— Abstract Layering of CoAP

V. Evaluation

To evaluate the implemented solution of this research, a sensor was used, which is Raspberry Pi 2 (RPi2) Model B Quad-Core 900 MHz 1 GB RAM. The kit contains CanaKit 2.5A Micro USB Power Supply with Noise Filter (UL Listed) specially designed for the Raspberry Pi 2. Additionally, 8 GB MicroSD Card and CanaKit Wi-Fi Adapter are included.

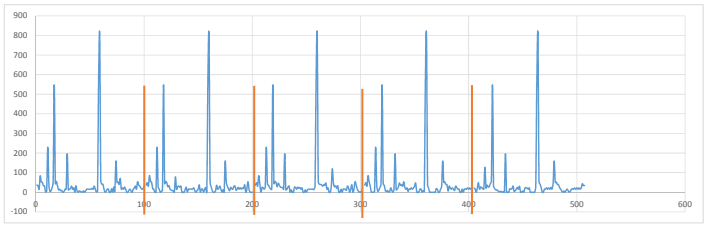
The evaluation obtains the response time (is the amount of time from the moment that a client sends a request to server until the time that the client receives the response and the entire request has completed) of the proposed solution by sending different numbers of requests from diverse numbers of users in different delays (table 6-1).

Table V—Evaluation

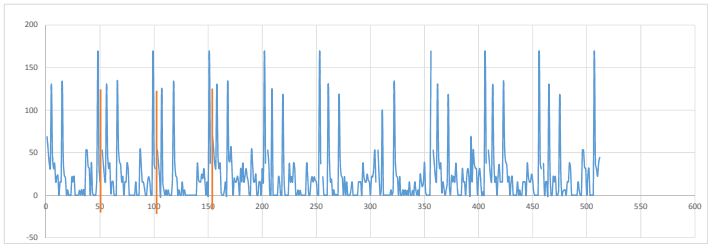
Delays	125 MS	250 MS	500 MS
Requests(r) per users(u)	100r per 5u	100r per 5u	100r per 5u
	50r per 10u	50r per 10u	50r per 10u
	25r per 15u	25r per 15u	25r per 15u

The results of the response times from applying the evaluation plan are presented below:

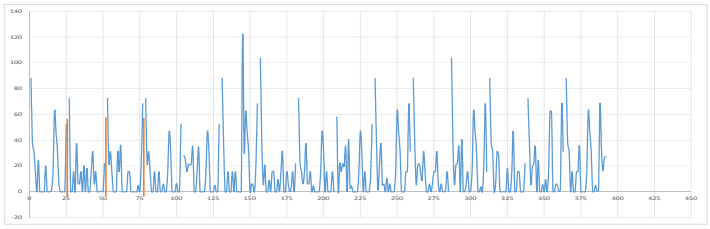
A. When the delay = 125 MS



a) The response times when 100 requests sent per 5 users



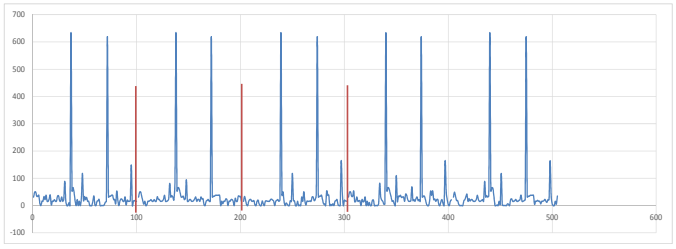
b) The response times when 50 requests sent per 10 users



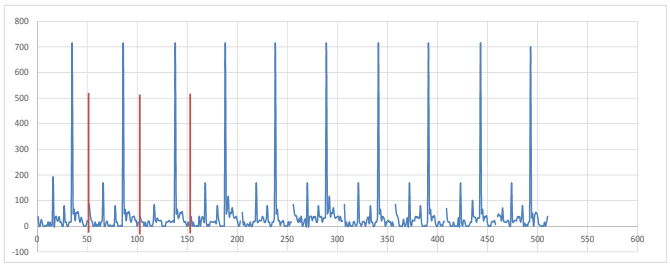
c) The Response times when 25 requests sent per 15 users

Figure V.A—Evaluation results when delay = 125 MS

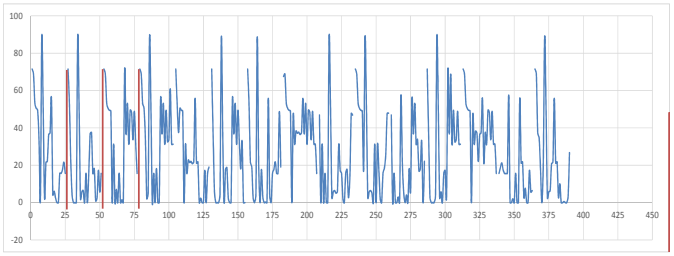
B. When the delay = 250 MS



a) The response times when 100 requests sent per 5 users



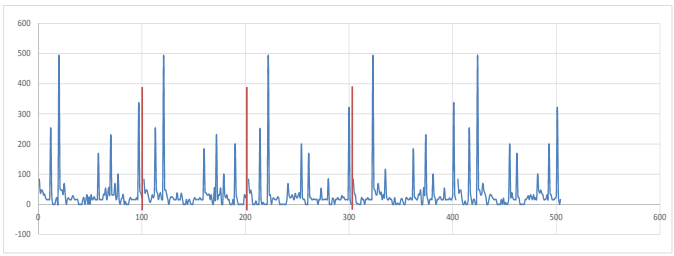
b) The response times when 50 requests sent per 10 users



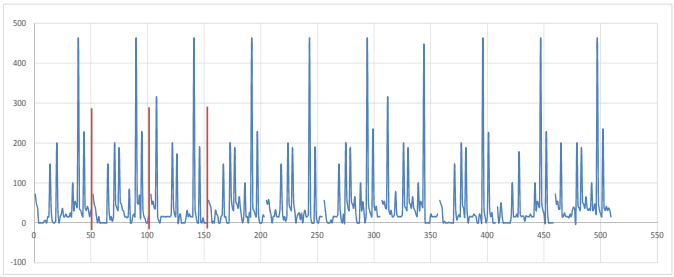
c) The Response times when 25 requests sent per 15 users

Figure V.B—Evaluation results when delay = 250 MS

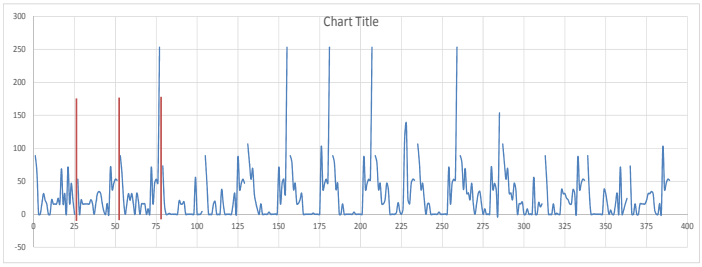
C. When the delay = 500 MS



a) The response times when 100 requests sent per 5 users



b) The response times when 50 requests sent per 10 users



c) The response times when 25 requests sent per 15 users

Figure V.C—Evaluation results when delay = 500 MS

In all (a) and (b) charts in each step of the previous evaluation results; the total number of requests is 500 requests because in (a) we have 5 users each sends 100 requests, and in (b) 10 users each sending 50 requests. In all (c) charts, there are 15 users each sends 25 requests, so the total of requests is 375. Hence, all charts of (c) have the fastest round-trip times (response times). On the other hand, (a) and (b) have slower response times because they have more number of requests. In general, sending the requests in all the delays was relatively fast; however, we can see in the charts that the response times have an increase suddenly. In my opinion, I think the reason for that is because I have the database and the ABAC policies

in the Raspberry Pi 2 along with the client and the server. As a result, the users sending the requests and the systems applies ABAC rules simultaneously, and that takes some time and power from the CPU of the sensor.

In addition, each chart has three oranges lines, these lines to show the first three users. Just to indicate that each user has the same number of requests and the same delay.

VI. Summary and Contribution

In conclusion, technology is growing quickly, especially in IoT systems. The growth is evidenced by the increase in the number of sensors, effectors, and the mobile application that apply the term IoT. Those circumstances led to

- This research problem is how to secure data in IoT systems from an unauthorized access.
- The motivation in solving this problem is to prevent loss of money to the IoT system owner and avoid safety threats to the users.
- This research proposed solution is an access control mechanism applied on the users' attributes to verify the identity of the user, and it was built in a low power device (Raspberry Pi 2).
- For evaluating the proposed solution, we send different number of requests from different number of users in dissimilar delays to calculate the round trip time (response time).

This research's domain is IoT hence it is focusing on low-powered and Internet-connected devices because they are essentials for IoT application. Besides, in Chapter 2, some of the challenges in IoT systems; especially, in security area were presented.

The contribution of this research is trying to prove that an access control method could be built in the sensor which used in IoT system. By applying an Access Control method on IoT system prototype using the low-powered device, which is Raspberry Pi 2, and a CoAP client that is Copper (Cu) CoAP user-agent for Firefox web browser.

VII. References

[1] R. Focardi and R. Gorrieri, Eds., *Foundations of security analysis and design: tutorial lectures*. Berlin ; New York: Springer, 2001.

[2] S. Musa, "Cybersecurity: Access Control," *The EvoLLLution*, 04-Feb-2014. .

[3] E. Yuan and J. Tong, "Attributed based access control (ABAC) for Web services," in *2005 IEEE International Conference on Web Services, 2005. ICWS 2005. Proceedings*, 2005, p. 569.

[4] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," National Institute of Standards and Technology, NIST SP 800-162, Jan. 2014.

[5] S. Kaiwen and Y. Lihua, "Attribute-Role-Based Hybrid Access Control in the Internet of Things," in *Web Technologies and Applications*, Springer, 2014, pp. 333–343.

[6] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the IoT," in *Communications (ICC), 2014 IEEE International Conference on*, 2014, pp. 725–730.

[7] X. Yang, Z. Li, Z. Geng, and H. Zhang, "A Multi-layer Security Model for Internet of Things," *IOT Workshop 2012*, pp. 388–393, 2012.

[8] J. Webber, S. Parastatidis, and I. Robinson, *REST in Practice: Hypermedia and Systems Architecture*. O'Reilly Media, Inc., 2010.

[9] O. DOSPINESCU and M. PERCA, "Web Services in Mobile Applications," *Inform. Econ.*, vol. 17, no. 2, pp. 17–26, 2013.

[10] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. big'web services: making the right architectural decision," in *Proceedings of the 17th international conference on World Wide Web*, 2008, pp. 805–814.

[11] J. H. Christensen, "Using RESTful Web-services and Cloud Computing to Create Next Generation Mobile Applications," in *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, New York, NY, USA, 2009, pp. 627–634.

[12] P. Nielsen, "The importance of context in keeping end users secure," *Netw. Secur.*, vol. 2015, no. 2, pp. 10–13, Feb. 2015.

[13] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," Jun-2014. [Online]. Available: https://tools.ietf.org/html/rfc7252. [Accessed: 21-Mar-2016].

[14] T. Jaffey, "MQTT and CoAP, IoT Protocols," Feb-2014. [Online]. Available: http://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php. [Accessed: 24-Mar-2016].

[15] "Raspberry Pi 2 Ultimate Starter Kit." [Online]. Available: https://www.canakit.com/raspberry-pi-starter-ultimate-kit.html. [Accessed: 27-Jul-2016].

[16] I. T. S. Sector, "ITU-T Recommendation Z. 120," *Message Seq. Charts MSC96*, 1996.