Scientific Paper / Artículo Científico

UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR
INGENIUS
Revista de Ciencia y Tecnología
https://doi.org/10.17163/ings.n25.2021.09
pISSN: 1390-650X / eISSN: 1390-860X

Reducing the IoT security breach with a microservice architecture based on TLS and OAuth2

Reduciendo la brecha de seguridad del IoT con una arquitectura de microservicios basada en TLS y OAuth2

Diego Ordóñez-Camacho[1,*]

Received: 15-09-2020, Reviewed: 19-10-2020, Accepted after review: 14-11-2020

Abstract

The Internet of Things has emerged as one of the most promising trends today. The speed of its adoption, however, has caused certain gaps. Amongst the most critical there is the one related with the security of the systems involved. This project addressed the security problem in a broad way but focusing on smart-home environments, where the use of devices with widely heterogeneous technologies and multiple services, generates problems with authentication and with the confidentiality of the data, if the network is compromised. To tackle these problems, state-of-the-art technologies such as OAuth2 and TLS, among others, were put together, along with an architectural methodology of lightly coupled microservices. As a result, a secure and broad range IoT architecture was built, backed up and validated by a reference implementation. The division into functional layers enables both fixed and mobile devices and sensors, to get connected into the system transparently and fluently. The security scheme structured in three incremental levels enables a better device integration, at the level that best adapts to its computing resources and the type of information it shares. The results show the flexibility of the solution and the robustness and novelty of the security scheme presented.

Keywords: IoT, microservices, *software* architecture, systems security, TLS, OAuth.

Resumen

El Internet de las cosas es una de las tendencias más prometedoras en la actualidad. La rapidez de su adopción, sin embargo, ha provocado ciertas brechas críticas en la seguridad de los sistemas involucrados. Este proyecto analizó el problema de seguridad de una manera amplia, pero enfocándose en entornos de tipo hogar inteligente, donde el uso de dispositivos con tecnologías ampliamente heterogéneas genera problemas en la autenticación con múltiples servicios, y en la confidencialidad de los datos, si la red llegara a verse comprometida. Para atacar estos problemas, se juntaron tecnologías de última generación como OAuth2 y TLS, entre otras, junto a una metodología arquitectural de microservicios de acoplamiento ligero, para generar una arquitectura IoT segura y de amplio alcance, respaldada y validada por una implementación de referencia. La división en capas funcionales permite que tanto los dispositivos y sensores fijos como aquellos móviles, puedan acoplarse al sistema de manera transparente y fluida. El esquema de seguridad estructurado en tres niveles incrementales permite que cada equipo pueda integrarse al que mejor se adapte tanto a sus recursos computacionales como al tipo de información que debe entregar o consumir. Los resultados muestran la flexibilidad de la solución y la solidez del esquema de seguridad presentado.

Palabras clave: IoT, microservicios, arquitectura de *software*, seguridad de sistemas, TLS, OAuth

[1,*]Grupo de Investigación en Informática (GrIInf), Universidad UTE – Ecuador.
Corresponding author ✉: dordonez@ute.edu.ec. http://orcid.org/0000-0001-8390-634X

Suggested citation: Ordóñez-Camacho, D. (2021). «Reducing the IoT security breach with a microservice architecture based on TLS and OAuth2». Ingenius. N.° 25, (january-june). pp. 94-103. doi: https://doi.org/10.17163/ings.n25.2021.09.

1. Introduction

The Internet of Things (IoT) is a technology that is strongly getting in real life of people. All environments are involved, namely urban, industrial, office or home. The interest generated and the speed of adoption of the technology have produced certain disorder and informality in the process. Consequently, important elements were put aside, with the security being one of the most relevant [1].

In principle, the security of IoT does not have to be very distant from the security of a typical computer network. However, in practice there are specific difficulties associated to the environment that further complicate the security problem. Many devices for IoT are computationally limited, which hinders using various known robust security mechanisms. The great number of devices that may be involved in an IoT network, as well as the exponential increment in the number of the interactions, aggravates the problem. The diversity of equipment that are utilized, both in hardware and software, complicates the possibility of generalizing the proposed solutions [2]. There is a great variety of methods and tools that may be utilized to do the job [3]. In this work, the intention was to utilize those techniques that, as IoT, set trends and are successfully utilized in other related fields. Among these techniques, the most promising ones were the microservices [4] and OAuth2 [5], which, together with techniques and technologies perhaps a little more traditional but equally successful such as TLS [6] and MQTT [7], provided an appropriate structured environment.

The general objective was to provide the world of IoT with an alternative architecture that is secure and adapted to the new technological trends, which may be utilized in a generic way in multiple situations. More specifically, special relevance was first given to generating an alternative for smart homes, and for this reason the methodology promotes the clear division of functionalities but caring for the fluid integration and the availability of tools for creating new utilities. Second, among the multiple existing security problems, those related with the authentication of clients when calling multiple services and with the confidentiality during the transmission of information were tackled, especially when the network is compromised. At last, aware of the hardware restrictions of many devices, especially sensors, work was carried out to propose an architecture that considers a hierarchy of various levels of security, enabling an interconnection adjusted to the capabilities of various types of equipment, particularly in environments with heterogeneous technologies.

1.1. Related work

In the ecosystem of IoT devices, these are largely insecure, since they are equipment of small size and with low energy consumption; thus, they also have limited computational resources. This latter condition greatly affects when attempting to incorporate in them complex security systems [8]. At the industrial level, there are various entrepreneurships applying IoT, for example, in intelligent transportation [9, 10] and in agriculture [11]; however, one of the main current objectives related to home or office automation is the connected living. This objective requires important advances in the field of IoT, where it is necessary to provide answers to problems related with the enormous increment of devices that should interact [12]. A particular case of IoT is the one corresponding to smart homes, since the implemented solutions are frequently ad-hoc, by the users themselves, who often attempt to minimize costs and efforts, which in general results in a minimum and probably inexistent security scheme [13].

The devices involved require interconnecting in a many-to-many scheme. In order to ensure the information exchange, it is necessary to implement a system for administration of identities that scales appropriately. In this sense, Ayed, Boujezza and Riabi [14] propose a home system which combines EAP, OAuth and DTLS. Also concerned about the administration of identities and the access control, Fernández, Alonso, Marco and Salvachúa [15] confirm the possibilities of OAuth and propose an architecture compatible with services.

Bugeja, Jacobsson and Davidsson [16] analyze the IoT security problem for home systems and remark the importance of avoiding that sensors capture and distribute, indiscriminately, home data. As an example, they present the case of private conversations, which should not be published. Among the possible approaches, they mention a service-oriented approach as a viable alternative to balance between centralization and distribution of the control; furthermore, the trend in distributed software and applications seems generally directed to the use of microservices [17–19]. Delving deeper in this line, Díaz-Sánchez, Marín-López, Almenarez, Arias and Sherrat [20] highlight the benefits that an architecture based on microservices, additionally based on TLS/PKI, may have on IoT, by making lighter the development and maintenance tasks, which is beneficial for both providers and distributors and for users, while simultaneously reinforcing the interconnection security. Case studies such as the one by Urien [21] confirm in a practical manner the possibilities of these techniques.

Although SSL/TLS is the preferred security and encryption mechanism in most of the related works, it is very interesting the proposal of Hoz *et al.* [22] when analyzing the complications that there may be in IoT with TLS at a practical level. Then, such work analyzes the possibilities of using SSH and highlights the advantages provided by the data compression included in such protocol, which shows to be especially

beneficial when working on HTTP.

Khan, Anwar, Azam, Samea and Shinwari [23] contribute to the IoT environment with an approach administered by models, and precisely propose an OAuth oriented model with a strong UML inclination. Through transformations, this proposal might be adapted to a specific architecture, offering the possibility of being customized to the required environment. Another interesting proposal of architecture and security is the one presented by Kim, Wasicek, Mehne and Lee [24], where rather than a traditional mechanism such as the one presented by SSL certification authorities, it would be used an approach to local certification authorities who more frequently, but also in a lighter process, would authenticate the IoT equipment. The proposal by Pahl and Donini [25] may be considered a middle point between the two proposals just reviewed, which uses traditional certificates, but with authentication nearby, rather at the nodes level; they highlight that this mechanism might be complemented with any one of authorization, such as OAuth or similar.

Sciancalepore, Piro, Caldarola, Boggia and Bianchi [26] emphasize on OAuth, but especially with the feature of concentrating their security architecture in the gateway equipment, which houses the base station or sink node, which is in charge of the heavy procedure of authenticating, authorizing and establishing links between clients and resources. This approach is very relevant when it is considered the susceptibility of the edge computing equipment [27], through which a whole IoT system may be compromised. One of the security high points in edge IoT systems is often the one related to the use of MQTT (or similar protocols), and consequently various works such as the one by Singh, Rajan, Shivraj and Balamuralidhar [28] propose improvements on such protocol.

2. Materials and methods

For this work it was mainly considered that within the IoT ecosystem it is necessary to segment the location, range and access of the equipment involved in two layers: local or edge and centralized.

In the local layer, schematically represented in Figure 1, there are those elements that will be invariably installed in the smart household or office, as it is the case of sensors, actuators, edge processors such as gateways and brokers, and user mobile devices.

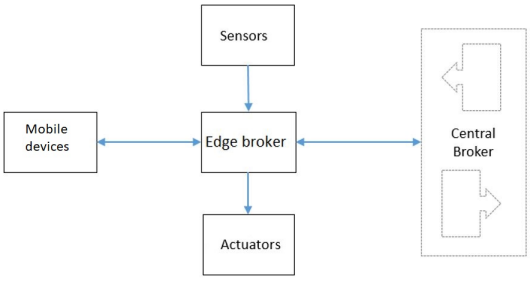

Figure 1. Components in the local or edge layer of the IoT system

The sensors are the equipment capable of capturing physical phenomena, virtual events or periodic signals. Those sensors with the necessary capability may communicate directly with the central broker; otherwise they will interact with equipment in the edge processing subsystem. The sensors will be static, when emitting a constant signal that in general would be used by mobile equipment moving around the environment as Bluetooth beacons for positioning. The dynamic sensors will capture measurements from the environment, which will vary depending on the environmental conditions, such as luminescence, temperature, humidity, among others.

The actuators will enable the interaction with hardware or software, generating events or actions. They will receive instructions either directly from the broker, or from a preprocessor when necessary. They may be local, located in the smart environment, such as light or temperature controllers; they may also be remote such as those capable of sending instructions, probably through the network, for a distant equipment, but controlled from the smart household or office, such as when it is necessary to send an SMS, mail or tweet.

At last, this layer considers the preprocessing equipment, which may be also called processors or edge brokers. They capture raw information coming from the sensors, to resend it to the central broker or to the actuator when the sensor is not capable. The information could be sent as received by the sensor or could be preprocessed and send the result. This equipment may be Raspberry Pi or small computers such as tablets.

In the centralized layer, shown in Figure 2, it is considered the equipment in charge of the general coordination of all the components, and it is considered that three subsystems are necessary: administration, processing and persistence. These subsystems are linked with each other and also with the edge layer through a centralized broker.

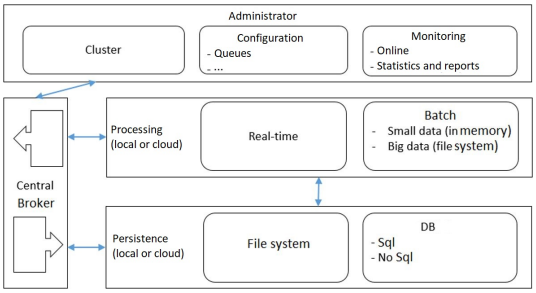

Figure 2. Components in the centralized layer of the IoT system

The administration subsystem defines the parameters, and the configuration of the system presents the web interfaces so that administrative users interact with the whole system. In the case that the central processing is performed with various equipment, it also manages the resulting cluster. Then, a main part in this subsystem is the monitoring, which will enable all type of users to review relevant information, preferably by means of dashboards and statistical charts.

For all heavy information processing, the corresponding subsystem takes the data collected from the broker and process them as defined by the applications or specific requirements of the IoT system. In general, the processing will be divided depending, mainly, on the urgency of the processing, in: real-time, which processes the data in a continuous flow, as they arrive from the sensors; in memory, which collects the information in the cluster memory, depending on the needs, and process it in small batches; and batch, which generally interacts with the storage system, for those processings where the amount of data is larger than the capacity of the living memory of the system.

The information generated by the IoT system is directed to the persistence module, which safeguards the data for further use either in model development or in the generation of reports. Various alternatives should be considered, depending on the size of the information and the way it will be accessed. At least it is necessary to consider an HDFS support, one of database both SQL and NoSQL, and in case that the system evolution requires it, a support in the cloud.

At last, the whole system, and more specifically the edge and centralized layers, should connect to each other and exchange information, which is achieved by means of a central broker, in charge of handling all the message queues thus reducing the complexity of the interactions.

3. Results

The resulting architecture design mainly considered the need of ensuring the exchange of information of all system components, always seeking to guarantee the speed of calculation. These elements require to reconcile characteristics that are often incompatible. For example, more robust cryptography systems may require more computational power than the one provided by many light devices, such as sensors.

The final architecture designed, implemented and tested, is the one sketched in Figure 3, which will be described in detail in the following. First, the components involved are pointed out, and then the security functionality is presented in a general way.

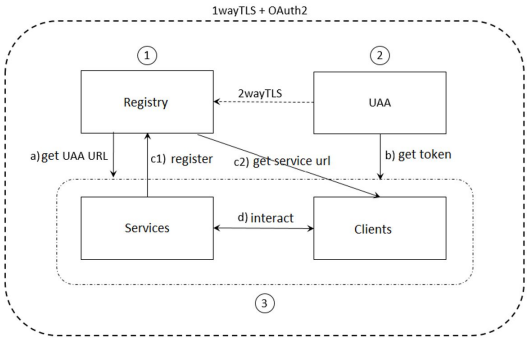

Figure 3. Secure architecture

3.1. Components

This section will attempt to define in a general way the types of components or equipment involved in the architecture proposed in Figure 3, which basically consists of the elements in charge of the registry services (Registry), the equipment providing authentication services for clients and users (UAA) and then, in a broad manner, all equipment providing general services, as well as all client equipment (Services and Clients). Although for simplicity reference will only be made to the components in singular, the architecture considers that, for each category or type of component, especially the services, these can work in clusters.

3.1.1. Registry services (REG)

The main task of the REG is to enable the services to register by means of IP and alias (service name), and make them available to the clients, which will connect to the REG to request information with which they will finally connect to the services of interest. The REG also provides a load balancing service, when detecting that a service is registered in cluster (various equipment with the same service).

The first contact point for all remaining components of the system, whether they are services or clients, is the REG, which requires a static IP; however, all the remaining components of the system may work with dynamic IPs, by means of a DNS.

3.1.2. Authentication services (UAA)

The initials UAA stand for User Authentication and Authorization. Within the proposed architecture, it basically provides the authentication service, which works under OAuth2. The UAA stores the data of all clients in the system, including their roles; with this information the UAA user services may also authorize or not the use of certain elements. Any component may connect with the UAA for requesting an access token, by means of client credentials (user and password). In a similar manner, any component of the system may request to the UAA the validation of the token received by a third party.

3.1.3. Generic services (SRV) and client equipment (CLI)

The last category of components gathers all other services and all clients. In general, these components will interact with each other after being registered/authenticated in the system with the help of the REG and the UAA. The services may be very diverse, and it is in the hands of the system administrator to decide which will be required. However, in the proposed IoT architecture, some are fundamental, and for this reason they have been implemented in the test system and will be mentioned in the following.

To enable interconnectivity and, at the same time, reduce its complexity, it was implemented a messaging service, which in the methodology is represented by the central broker (Figures 1 and 2). The broker is capable of receiving and distributing all messages circulating in the system, and it basically enables that in general all services and clients may establish a unique connection with the broker, to deposit messages and retrieve them from one or more queues. This broker may work with any communication protocol, or a combination of them. However, since in the IoT world, at least at present, the most widely used protocol is the Message Queuing Telemetry Transport (MQTT), it is the one that is used in the implementation presented here.

Other service implemented for testing the concept of the architecture is the one related to the persistence, as a necessary support to further implement batch processing. For this it was implemented a transit service that takes the information from the broker and transfers it to a Hadoop cluster [29], where different types of tools from such ecosystem may be used for information treatment. One of the cases that was worked here, given the nature of the IoT information, especially the one coming from the sensors, was the one of time series. For this purpose, two data series services were built, thus providing graphing and trend analysis, among others.

With respect to the clients, all sensors are considered here, which deliver information to the system, the actuators, which react with the environment thanks to the information of the system, and all those devices, mobile or desktop, that enable the user to enter to configure the system, collect processed information, or even also acting as sensors and actuators. For example, a mobile device may deliver information about the activity of the user or automatically send tweets.

3.2. Security schemes

The system security architecture consists of three fundamental scenarios: basic scheme, to which all elements should stick to in their transactions, unless it is specified otherwise; light scheme, generally used only when starting a work process in the system; strengthened scheme, for trusting relationships between services.

3.2.1. Basic scheme

This is the scheme that the components of the system will use in their transactions by default. This scheme is represented in Figure 3 by the dotted line that encompasses the system and uses a combination of one-way TLS plus OAuth2. All services should provide its public key infrastructure (PKI) certificate to the clients, who could thus validate it with the certification authority (CA). Similarly, all clients should provide an OAuth access token to the services so that they can validate it with the authentication service.

In the proposed scheme, the use of the TLS is especially necessary to be able to encrypt the content of the information being transmitted. It is only utilized on the side of the services to limit as much as possible the overload that may occur, especially, at the administration level (but also at the resources and processing levels), use it in all components. The security gap that appears is compensated with the use of OAuth2, through which the clients are, in turn, validated by the services.

3.2.2. Light scheme

This is a scheme that may be considered insecure, and for this reason it is only provided for those cases where an access token may not be yet obtained, or when it is considered redundant to request it.

Now, two cases exist in the working environment that implement this scheme. When the components enter the system to be able to start their transactions, it is in general necessary to have the OAuth token, but since the IP address of the UAA server may have changed, the first step is to contact the REG to request the updated IP address. For this connection the client does not yet have the token, and for this reason it is not possible to work with the basic scheme. The second implementation of this scheme was applied to avoid unnecessary redundant connections and occurs when the service receives the token and should validate it

with the UAA. Since the service was already initially validated with the UAA, is it prevented to duplicate this step.

It is for this type of cases that the light scheme comes into play. The service provides its PKI with which the communication is encrypted, but the client is not obliged to validate it (although it is recommended to do so), the service provides an access point «insecure» for the client, who does not require the token. This is the scheme provided by REG exclusively to be able to deliver the data of the UAA.

3.2.3. Strengthened scheme

Similar to the problem worked with in the light scheme, sometimes two services require to interconnect, but at least one of them (who acts as client) is not capable of obtaining its access token. Since this is about services, it is not convenient to open an insecure channel as it is done in the light scheme. Then, to maintain the security standard it was decided to implement a two-way TLS scheme, which is possible without incurring in much overload, since as they are services, they anyway have their PKI. In addition, in general the services will be run in equipment with larger processing capacity.

This implementation also requires a dedicated channel to be able to execute this type of validation, and the example is given by the communication between the REG and the UAA. The UAA is the one that provides the access tokens, and therefore it should validate itself which would generate a security hole. Then, the REG opens a dedicated channel so that at any moment a UAA service may register that way. As the UAA connects with the REG, they exchange their corresponding PKIs, validating each other by TLS, without reducing the security standard of the system.

3.3. Functionality

Referring again to Figure 3, the dotted line represents the scope of the basic security scheme, which encompasses the whole system. Numbers 1, 2 and 3 may be seen inside, enclosed in circles, which indicate the recommended starting sequence to guarantee the fluency of the service. In practice, at least the services have in their base library the functionality to retry the connection when this starting sequence is not followed. However, this may be susceptible to unnecessary delays.

First the registry server REG is started, which will provide a central access point for acquiring the contact information for the other services: all services will register in the REG their corresponding IP and alias (name of service) and all clients will look for here, by alias, the IP of the required service to be able to connect to it. REG offers three access points, each of which should handle a different security mode: the first, light, enables any client to obtain the UAA IP without additional security; the second mode, strengthened, enables the connection of the UAA by means of a two-way TLS; the last, basic, which requires OAuth2, enables clients to request information about services, and services to register their contact information.

Second, an authentication server (UAA) is started, which will provide OAuth2 credentials to the clients. A reinforced security mechanism is used between the UAA and the REG, with two-way TLS validation. The UAA connects with the REG, as well as with any other service, to deliver its IP and alias and thus being available for the whole system. Once these two services, REG and UAA, are online, all the remaining components, services and clients may start their work.

Then, at last, as point 3, any other component, either service or client, will proceed as follows: first, using the light security scheme, they will connect with the REG to request the IP address of the UAA; they establish the connection with the UAA and request the access token, using their client credentials. With the access token at hand the basic security scheme may be already utilized and, in the case of services, they will register with the REG, delivering IP and alias, and be ready to wait for the requests of the clients, or act as a client of another service, as necessary. In the case of a client, the next step is using a service, where the basic security scheme will be applied; it is connected to the REG and by means of an alias it requests the IP of the service of interest, to further connect with such service. The last action becomes that of the services that receive a request from a client, since in this case they should, by means of the light scheme, connect to the UAA and request the validation of the access token, with which they can serve the request of the client.

3.4. Implementation and tests

The implementation decision was mainly that each of the components may be run in a variety of equipment with the lowest interdependency, for which it was proceeded to work in a microservices architecture that enables their deployment either as independent processes, or within a container structure, such as Docker [30]. For desktop services and clients, it was used Java with Spring Boot in general. For the central messaging service, it was decided to work using the MQTT protocol, and for the development of the broker the Moquette library [31] was taken as base, which was modified mainly for adding support for OAuth2. For the persistence services an HDFS cluster [32] was used as base in the local network, and time series services were built on it which, according to the current interests of the project, were the more appropriate to process data coming from the sensors. It was interacted with OpenTSDB [33] and Prometheus [34].

Regarding clients, it was decided to construct generic libraries for different types of systems, which ease the process of developing specific applications. A Java client library was developed, one for Android mobile devices, one for Arduino MKR [35] and another for ESP32 [36], the last three based on Eclipse Paho [37]. The Arduino libraries were exclusively oriented for their use in controllers of sensors and actuators.

The tests of the system were carried out in a controlled office environment. The main equipment was an RPi 3B+ which acted as a Wi-Fi Gateway, providing the DNS and NTP services, among others. Controllers MKR 1010 and ESP32 were simultaneously used, which permanently received information from temperature, humidity and noise sensors, such as the DHT11 and the KY038 [38]. The RPi3B+ also housed the REG, and UAA services and the MQTT messaging broker. The persistence services on HDFS, as well as those of the TSDB were installed in Linux on an i5-4210U with 8 GB of RAM memory. In this last equipment generic services were also installed for sending and receiving messages, with which it was verified the fluency of the interaction. A N9005 was used as mobile client, which had two functionalities: bobo sensor, sending a great quantity of random numbers to the system, and light actuator, notifying every time that the measurements of the real sensors exceeded particular levels parameterized in the app. Figure 4 presents one of the setups of the test system.

In the experimentation relative to the security tests, it was decided to suppose that a possible attacker was already connected to the local network and was potentially in the capacity of making any request and capturing all the traffic. In the first test, Zap was used to perform a scanning both active and passive, and it was verified that the security was maintained (encrypted traffic), except in the case (documented in the architecture) of the light scheme.

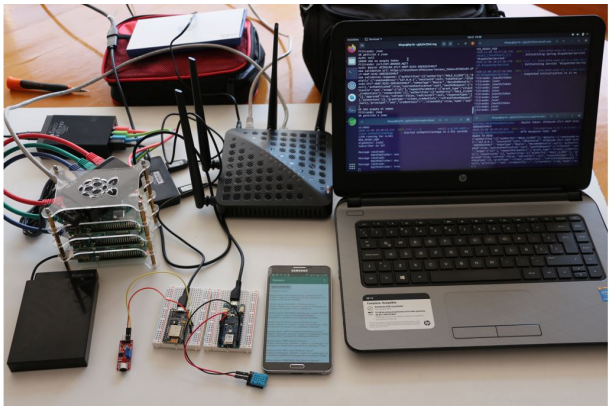

Figure 4. Setup for tests: from left to right it is observed a cluster of 3 Raspberry Pi running all services on Docker; an ESP8266 monitoring the level of noise with an KY038; an MKR1010 monitoring ambient temperature with an DHT11; an N9005 injecting random messages, and a laptop monitoring all services.

Then, Wireshark was used for a manual verification, analyzing the packets by type of connection or scheme, and it was again validated, as expected, that, except for the light scheme, all traffic was kept encrypted, as shown in Figure 5.

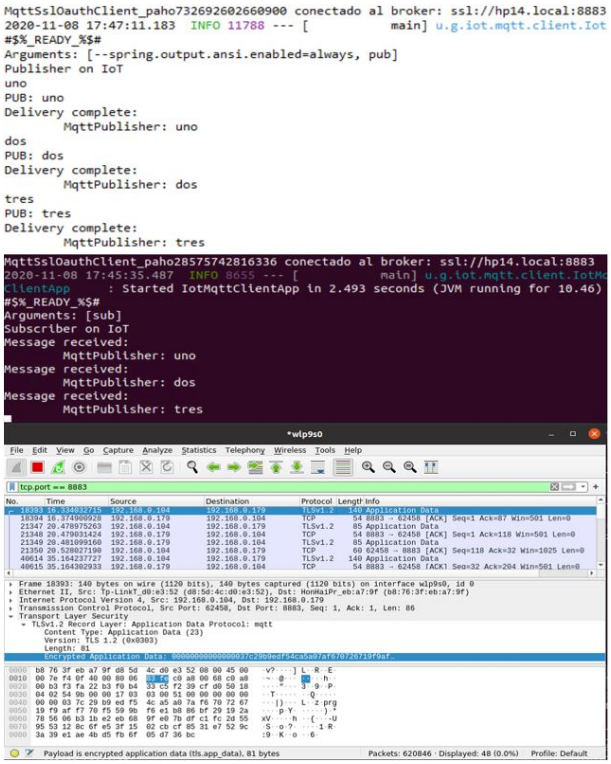

Figure 5. Run and traces: from top to bottom, first, a client sends three simple text messages; then the receiver obtains the messages and, at last, the Wireshark trace captures the packages and verifies that, during transit, they are encrypted.

4. Discussion

The process of including complex security schemes in light IoT devices, is not trivial as stated by Khan and Salah [8], and this work coincides with this statement. Two remarkable cases were that managing the TLS with self-generated certificates for MKR 1010 require regenerating all the firmware to include the CA, and it was not possible to carry this out in ESP8266 controllers due to the unavailability, in practice, of open-source libraries sufficiently complete to guarantee the expected security level.

In general, the adopted approach retakes the warnings made by Lin and Bergmann [13], and attempts to provide a sufficiently complete system so that with minimum technical support it could be implemented at home, and then directly administered by the user.

Clearly, this exhibits limitations given by the great variety of types of users that may want to be included in the IoT. However, the tests carried out indicate that the heart of the prototype would enable providing this, if it is possible to include some features in the user interface, equipment and installers.

This work confirms what was presented by Ayed *et al.*, [14] and by Fernández *et al.*, [15] regarding the problem of scaling identities and the benefit that may be obtained both with OAuth and with a services approach. Delegating the authentication to a single point, further trusting in temporary credentials, as enabled by OAuth2, maintains a light infrastructure of identities, while the distribution in services provides a great easiness when replicating and redounding services in a cluster when the work load of the system requires it. It is just this services/microservices approach which enables that working with TLS is not too demanding at the level of maintenance, as highlighted by Díaz-Sánchez *et al.* [20].

This proposal acknowledges the importance that should be given to the edge equipment at the security level, and following the line of Shapsough *et al.* [27] and of Sciancalepore *et al.* [26], it especially reinforces the gateway equipment to further enable implementing improvements in the MQTT protocol, similar to what was made by Singh *et al.* [28] but mainly including the use of OAuth2 and TLS.

5. Conclusions

This work was motivated by a current urgent need: to secure the IoT systems, especially those linked to a home environment, and making it without undermining the freedom of access of the user to collect information and to modify the configuration of the system. As it was seen, this need comes from the relative informality of the IoT, among other things, especially in the environment of a smart home.

It started with a methodological conception that stratifies the environment in layers: the one more closely related with the interaction with the space through the collection of information and execution of actions to modify the microenvironment, and the layer of centralized processing and analysis of information. Both layers were interrelated with a centralized connection that unifies them also maintaining their light coupling.

This methodology was oriented to enable an implementation based on microservices, where, besides avoiding to the extent possible any type of monolithic structure, it was provided a group of services and libraries for clients which ease to a large extent the generation of new utilities and components. The tests carried out with the services, clients and sensors generated under this infrastructure enabled confirming both the solidity and the relative easiness of use of the components.

The main point, the security, although it was not the easiest to implement, after debugging the three schemes defined for the different types of connection, it demonstrated being a robust choice, which withstood the illicit access tests. However, it is worth indicating that a methodical testing scheme specific for the field is still pending to be designed and applied, which will be considered in future works. Nevertheless, it may be stated that combining TLS, OAuth2 and MQTT produced the expected results to a large extent.

As main contributions it could be mentioned the implementation of a solid secure IoT architecture for households; the stable and fluent combination of at least three high level technologies for handling security and an implementation of functional reference that can be made publicly available for free use.

References

[1] Y. Lu and L. D. Xu, "Internet of things (IoT) cybersecurity research: A review of current research topics," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2103–2115, 2019. [Online]. Available: https://doi.org/10.1109/JIOT.2018.2869847

[2] A. Riahi Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A roadmap for security challenges in the internet of things," *Digital Communications and Networks*, vol. 4, no. 2, pp. 118–137, 2018. [Online]. Available: https://doi.org/10.1016/j.dcan.2017.04.003

[3] P. Lea, *Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security.* Packt Publishing Ltd, 2018. [Online]. Available: https://bit.ly/3oJ1XRl

[4] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," *IEEE Software*, vol. 35, no. 3, pp. 24–35, 2018. [Online]. Available: https://doi.org/10.1109/MS.2018.2141039

[5] J. Khan, J. p. Li, I. Ali, S. Parveen, G. a. Khan, M. Khalil, A. Khan, A. U. Haq, and M. Shahid, "An authentication technique based on oauth 2.0 protocol for internet of things (IoT) network," in *2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, 2018, pp. 160–165. [Online]. Available: https://doi.org/10.1109/ICCWAMTIP.2018.8632587

[6] C. Chan, R. Fontugne, K. Cho, and S. Goto, "Monitoring tls adoption using backbone and edge traffic," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 208–213. [Online]. Available: https://doi.org/10.1109/INFCOMW.2018.8406957

[7] F. Izquierdo, M. Ciurana, F. Barcelo, J. Paradells, and E. Zola, "Performance evaluation of a TOA-based trilateration method to locate terminals in WLAN," in *2006 1st International Symposium on Wireless Pervasive Computing*, 2006, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISWPC.2006.1613598

[8] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2017.11.022

[9] J. P. Rojas, J. C. Bustos, and D. Ordóñez Camacho, "Smart public transportation at your fingertips," *Enfoque UTE*, vol. 8, no. 1, pp. 122–134, Feb. 2017. [Online]. Available: https://doi.org/10.29019/enfoqueute.v8n1.143

[10] J. P. Rojas, J. C. Bustos, and D. Ordóñez-Camacho, "Qbus: Movilidad inteligente para el usuario de transporte público," in *Proceedings of the International Conference on Information Systems and Computer Science, INCISCOS 2016*, 2016. [Online]. Available: https://bit.ly/3jZlBpE

[11] E. A. Q. Montoya, S. F. J. Colorado, W. Y. C. Muñoz, and G. E. C. Golondrino, "Propuesta de una arquitectura para agricultura de precisión soportada en IoT," *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, pp. 39–56, 2017. [Online]. Available: http://dx.doi.org/10.17013/risti.24.39-56

[12] M. Agiwal, N. Saxena, and A. Roy, "Towards connected living: 5g enabled internet of things (IoT)," *IETE Technical Review*, vol. 36, no. 2, pp. 190–202, 2019. [Online]. Available: https://doi.org/10.1080/02564602.2018.1444516

[13] H. Lin and N. Bergmann, "IoT privacy and security challenges for smart home environments," *Information*, vol. 7, no. 3, p. 44, Jul 2016. [Online]. Available: http://dx.doi.org/10.3390/info7030044

[14] H. Kaffel-Ben Ayed, H. Boujezza, and I. Riabi, "An idms approach towards privacy and new requirements in IoT," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017, pp. 429–434. [Online]. Available: https://doi.org/10.1109/IWCMC.2017.7986324

[15] F. Fernández, A. Alonso, L. Marco, and J. Salvachúa, "A model to enable application-scoped access control as a service for IoT using OAuth 2.0," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, 2017, pp. 322–324. [Online]. Available: https://doi.org/10.1109/ICIN.2017.7899433

[16] J. Bugeja, A. Jacobsson, and P. Davidsson, "On privacy and security challenges in smart connected homes," in *2016 European Intelligence and Security Informatics Conference (EISIC)*, 2016, pp. 172–175. [Online]. Available: https://doi.org/10.1109/EISIC.2016.044

[17] L. Sun, Y. Li, and R. A. Memon, "An open IoT framework based on microservices architecture," *China Communications*, vol. 14, no. 2, pp. 154–162, 2017. [Online]. Available: https://doi.org/10.1109/CC.2017.7868163

[18] T. Vresk and I. Çavrak, "Architecture of an interoperable IoT platform based on microservices," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2016, pp. 1196–1201. [Online]. Available: https://doi.org/10.1109/MIPRO.2016.7522321

[19] R. Yu, V. T. Kilari, G. Xue, and D. Yang, "Load balancing for interdependent IoT microservices," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 298–306. [Online]. Available: https://doi.org/10.1109/INFOCOM.2019.8737450

[20] D. Díaz-Sánchez, A. Marín-Lopez, F. A. Mendoza, P. A. Cabarcos, and R. S. Sherratt, "TLS/PKI challenges and certificate pinning techniques for IoT and M2M secure communications," *IEEE Communications Surveys Tutorials*, vol. 21,

no. 4, pp. 3502–3531, 2019. [Online]. Available: https://doi.org/10.1109/COMST.2019.2914453

[21] P. Urien, "Securing the IoT with TLS/DTLS server stacks embedded in secure elements: An ePlug usecase," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 569–570. [Online]. Available: https://doi.org/10.1109/CCNC.2017.7983170

[22] J. D. Hoz, J. Saldana, J. Fernández-Navajas, J. Ruiz-Mas, R. G. Rodríguez, and F. d. J. M. Luna, "SSH as an alternative to TLS in IoT environments using HTTP," in *2018 Global Internet of Things Summit (GIoTS)*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/GIOTS.2018.8534545

[23] M. Khan, M. W. Anwar, F. Azam, F. Samea, and M. F. Shinwari, *A Model-Driven Approach for Access Control in Internet of Things (IoT) Applications – An Introduction to UMLOA.* Communications in Computer and Information Science, Springer, 2018, vol. 920. [Online]. Available: https://doi.org/10.1007/978-3-319-99972-2_16

[24] H. Kim, A. Wasicek, B. Mehne, and E. A. Lee, "A secure network architecture for the internet of things based on local authorization entities," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, 2016, pp. 114–122. [Online]. Available: https://doi.org/10.1109/FiCloud.2016.24

[25] M. Pahl and L. Donini, "Securing IoT microservices with certificates," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, 2018, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NOMS.2018.8406189

[26] 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," in *2017 IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 676–681. [Online]. Available: https://doi.org/10.1109/ISCC.2017.8024606

[27] S. Shapsough, F. Aloul, and I. A. Zualkernan, "Securing low-resource edge devices for IoT systems," in *2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI)*, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1109/ISSI.2018.8538135

[28] M. Singh, M. A. Rajan, V. L. Shivraj, and P. Balamuralidhar, "Secure mqtt for internet of things (IoT)," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 746–751. [Online]. Available: https://doi.org/10.1109/CSNT.2015.16

[29] C. Singh and M. Kumar, *Mastering Hadoop 3: Big data processing at scale to unlock unique business insights.* Packt Publishing, 2019. [Online]. Available: https://bit.ly/37Qi2O9

[30] J. Turnbull, *The Docker Book: Containerization is the new virtualization*, 2014. [Online]. Available: https://bit.ly/3m7nqRY

[31] A. Selva. (2014) Java MQTT lightweight broker. moquette. [Online]. Available: https://bit.ly/3gB82Mw

[32] M. Bhushan, *Big Data and Hadoop: Learn by Example.* BPB Publications, 2018. [Online]. Available: https://bit.ly/2W0AmP1

[33] T. Dunning and E. Friedman, *Time Series Databases: New Ways to Store and Access Data, Edition: 1. Sebastopol.* O'Reilly Media, Inc, 2014. [Online]. Available: https://bit.ly/2W1VnsU

[34] B. Brazil, *Prometheus: Up & Running: Infrastructure and Application Performance Monitoring.* O'Reilly Media, 2018. [Online]. Available: https://bit.ly/39V80xX

[35] A. Kurniawan, *Arduino MKR WIFI 1010 Development Workshop.* PE Press, 2018. [Online]. Available: https://bit.ly/37OEnvD

[36] I. Dogan and I. Ahmet, *The Official ESP32 Book.* Elektor International Media, 2017. [Online]. Available: https://bit.ly/2IzEW3G

[37] G. C. Hillar, *Hands-On MQTT Programming with Python: Work with the lightweight IoT protocol in Python.* Packt Publishing, 2018. [Online]. Available: https://bit.ly/33YpdTg

[38] B. Charles, *Beginning Sensor Networks with Arduino and Raspberry Pi.* Apress, 2013. [Online]. Available: https://bit.ly/3m5syGj