Introducing Usage Control in MQTT

Antonio La Marra[1], Fabio Martinelli[1], Paolo Mori[1], Athanasios Rizos[1,2], and Andrea Saracino[1(✉)]

[1] Istituto di Informatica e Telematica, Consiglio Nazionale delle Ricerche, Pisa, Italy
{antonio.lamarra,fabio.martinelli,paolo.mori,athanasios.rizos,
andrea.saracino}@iit.cnr.it
[2] Department of Computer Science, University of Pisa, Pisa, Italy

Abstract. MQTT is a widely-used general purpose IoT application layer protocol, usable in both constrained and powerful devices, which coordinates data exchanges through a publish/subscribe approach. In this paper we propose a methodology to increase the security of the MQTT protocol, by including Usage Control in its operative workflow. The inclusion of Usage Control enables a fine-grained dynamic control of the rights of subscribers to access data and data-streams over time, by monitoring mutable attributes related to the subscriber, the environment or data itself. We will present the architecture and workflow of MQTT enhanced through Usage Control, also presenting a real implementation on Raspberry Pi 3 for performance evaluation.

1 Introduction

Over the last years, Internet of Things (IoT) devices have become more and more pervasive to our daily life. IoT devices could be very different, since they typically have different types of hardware, depending on the provided functionalities, and software applications to manage them. Hence, in order to have a unique application which eases the control of all the smart devices owned by the same user, a necessity has arisen to be able to easily communicate with a set of distinct IoT devices. To this aim, several protocols have been proposed in the scientific literature, and among them, MQTT, which is recently standardized by OASIS, is one of the most widely used [1]. Since MQTT is based on HTTP functionalities, most of the MQTT security solutions seem to be either application specific, or just leveraging TLS/SSL protocols [1]. Although the effort concerning security of MQTT protocol is rising, two main obstacles occur. The first one is that, although the protocol has the ability to deal with various Publishers and Subscribers, since they use different platforms, it is difficult to create and enforce generic security policies. The second problem is that the current efforts are mainly directed to standard message communication security. Still no efforts

This work has been partially funded by EU Funded projects H2020 C3ISP, GA #700294, H2020 NeCS, GA #675320 and EIT Digital HII on Trusted Cloud Management.

S. K. Katsikas et al. (Eds.): CyberICPS 2017/SECPRE 2017, LNCS 10683, pp. 35–43, 2018.
https://doi.org/10.1007/978-3-319-72817-9_3

have been done in the direction of supporting policy enforcement at Broker level, nor it has been considered the possibility of dynamically revoking subscriptions.

In this paper, we propose the enhancement of the security of the MQTT protocol by adding Usage Control (UCON) in the MQTT architecture and workflow. UCON is an extension of traditional access control which enforces continuity of access decision, by evaluating policies based on mutable attributes, i.e. attributes changing over time [7]. By adding Usage Control in MQTT, we aim at enforcing dynamically fine grained policies, which do not only consider the identity of the Subscriber as a parameter for granting access to data, but also dynamic attributes such as Subscriber reputation, data reliability, or environmental conditions of a specific application. After surveying the main IoT application protocols, and motivating the choice of focusing on MQTT, this work will discuss the architecture and the workflow of the MQTT - UCON integration. The proposed framework is designed to be general, easy to integrate in the Broker component, remaining oblivious to both Publishers and Subscribers. The addition of UCON in fact, does not modify the MQTT protocol, enforcing the policies independently from the implementation of Publisher and Subscriber, which allows the proposed solution to be compatible with any off-the-shelf MQTT Publisher/Subscriber application. Furthermore, we demonstrate the viability of the approach by presenting a real implementation of the framework on both general purpose and performance constrained devices, measuring the performance on two Rasperry Pi 3 model B[1]that are used respectively as Broker and Subscriber.

The rest of the paper is organized as follows: In Sect. 2, a comparison between the main IoT application protocols is reported, detailing afterward the MQTT protocol and motivating our choice to focus on it. Furthermore, some background information about usage control are reported. Section 3 describes the integration of UCON and MQTT detailing the architecture and the operative workflow. Section 4 details the results of the performance analysis. In Sect. 5 are reported a set of related works about security in MQTT and application of UCON in IoT. Finally, Sect. 6 concludes by proposing future directions which stem from this preliminary work.

2 IoT Protocols and MQTT

The most known application layer protocols in IoT are CoAP, MQTT, XMPP, HTTP, AMQP and WebSocket. CoAP is more resource-friendly than MQTT [5] but in terms of Message Oriented Approach (MOA), MQTT stands out. All the protocols mentioned above use TCP as transport layer. Only CoAP uses UDP. The same happens as for the security layer. All protocols use TLS/SSL except from CoAP that uses DTLS. In fact, CoAP targets to very constrained environments. Furthermore, according to [6], MQTT provides the smallest header size of two bytes, although it is based on TCP. Moreover, it provides three levels of QoS which puts this protocol in the first place in terms of QoS, even though it needs extra load in the network for message retransmission. On the other hand,

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

XMPP requires processing and storing XML data, which necessitates memory space that is too large for most IoT devices. In addition, HTTP performs better in non constrained environments when PC, Laptop and Servers are used. It is generally not applicable in IoT devices due to its high overhead. AMQP [9], is more suitable for server-to-server communication than device-to-device communication. Websocket is neither a request/response nor a Publish/Subscribe protocol. In Websocket, a client initializes a handshake with a server to establish a Websocket session. The handshake process is intended to be compatible with HTTP-based server-side software so that a single port can be used by both HTTP and Websocket clients [4]. According to [12], MQTT messages experience lower delays than CoAP for lower packet loss and vice versa. When the message size is small the loss rate is equal. AllJoyn [13], is a full stack of protocols intended for IoT. Though quite popular, the main disadvantage of AllJoyn is that the application protocol cannot be separated from the rest of the protocol stack. As a synopsis to the basis, reader can also consult Table 1. This comparison gives the details about the existence of Quality of Service (QoS). Moreover, it refers to the communication pattern, which in the case of MQTT is the Publish/Subscribe. The most significant column is the third. In this column, we identified that MQTT is more general purpose.

Table 1. Application layer protocol comparison

Protocol	QoS	Communication pattern	Target devices
CoAP	Yes	Req/Resp	Very constrained
MQTT	Yes	Pub/Sub	Generic, small header
XMPP	No	Req/Resp Pub/Sub	High memory consumption
HTTP	No	Req/Resp	High performance
AMQP	Yes	Pub/Sub	Ser-2-Ser communication
Web Socket	No	Client/Server Pub/Sub	Needs less power than HTTP still needs high power
AllJoyn	No	Client/Server Pub/Sub	High computational power

MQTT is an open pub/sub protocol where the system complexities reside on the Brokers side. MQTT does not specify any routing or networking techniques; it assumes that the underlying network provides a point-to-point, session-oriented, auto-segmenting data transport service with in-order delivery (such as TCP/IP) and employs this service for the exchange of messages. MQTT is a topic-based Publish/Subscribe protocol that uses character strings to provide

support of hierarchical topics. This also gives also the opportunity to the subscription to multiple topics. MQTT supports basic end-to-end Quality of Service (QoS) [3]. Depending on how reliably messages should be delivered to their receivers, MQTT provides three QoS levels. QoS level 0 is the simplest one: it offers a best -effort delivery service, in which messages are delivered either once or not at all to their destination. No retransmission or acknowledgment is defined. QoS level 1 provides a more reliable transport: messages with QoS level 1 are retransmitted until they are acknowledged by the receivers. This means that QoS level 1 messages may arrive multiple times at the destination because of the retransmissions. The highest QoS level, QoS level 2, ensures not only the reception of the messages, but also that they are delivered only once.

3 Introducing UCON in MQTT

In this section we present the proposed architecture, presenting first the model, then the operative workflow and the performed implementation. As previously mentioned, MQTT protocol is based on the Publish/Subscribe model, thus the entities participating to the protocol can act either as *Publishers* or as *Subscribers*. Publishers could be sensors or other devices which collect and provide specific data, when available, periodically or even as a stream. Subscribers are instead entities that register to the Broker to receive, when available, specific data or set of information grouped under a *Topic*. The *Broker* acts as middleware and coordinator, managing the subscription requests and dispatching data to Subscribers, when available by prosumers.

The MQTT protocol supports ID and password-based authentication for both Publishers and Subscribers. The enforcement is performed on Broker's side, which keeps track of the ID and authentication password of authorized Publishers and Subscribers. However, we argue that this authentication model is too simplistic and coarse grained, making impossible to check the right to access information over time. In fact, once a Subscriber has been authorized, the subscription remains valid until the Subscriber explicitly invokes an `unsubscribe` for the topic(s) it was registered for. The same goes for Publishers which keeps the right to publish continuously or on demand, till they have valid credentials. In real applications, several features might imply a condition for a subscription to decay, or for a publication to be denied. Detected Publisher malfunction or corruption, conditions on time spans in which a subscription should be allowed, and Subscriber reputation, are just few examples of aspects on which a more complex policy should be enforced. To be able to enforce policies with similar conditions to the aforementioned ones, and to have the possibility of revoking a subscription, usage control has been added to the MQTT logical architecture. In Fig. 1, we depict the logical architecture of the proposed framework.

As shown, the UCS is physically integrated in the Broker Device, i.e. the physical machine that is hosting the Broker software, which enables the MQTT protocol. It is worth noting that we consider in this example three abstract PIPs, which are conceptually grouping the PIPs reading attributes related to the subject (PIP_S), to the resource (PIP_R) and to the environment (PIP_E). The PEP

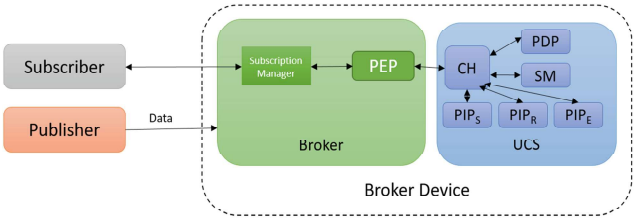

Fig. 1. UCON implementation in MQTT.

is (partially) embedded in the broker, to dynamically control the subscription events. In particular, the PEP will intercept the subscription events and interact directly with the Broker subscription manager, deleting and inserting the entries for Subscribers from the list of authorized ones, according on the UCS decision. In such a way, the PEP ensures that no Subscribers can register by avoiding the enforcement of the usage control policy. Since the PEP is embedded in the Broker, the proposed architecture remains compatible with any implementation of MQTT Subscribers. The only requirement is that the Subscriber is configured to access with username and password, otherwise the connection will be refused by the broker. In Fig. 2, we report the envisioned workflow. For the sake of simplicity, we will consider a simple system made out of a Broker and a single Publisher and Subscriber. The workflow is initiated by a subscription request from the Subscriber to the Broker. This request is intercepted by the PEP, which interprets it, so as to take the credentials of the Subscriber that are needed in order to create and send the request to the UCS for evaluation. Hence the PEP invokes the `TryAccess` sending to the CH request and policy. The request is eventually filled by attributes retrieved through the PIP, then is sent, together with the policy, to the PDP for evaluation, which should return a Permit or Deny decision. In case of Deny, the subscription request is dropped and the Subscriber will be notified, as if a wrong username/password has been inserted. In case of Permit, the Session Manager (SM) creates the session and sends its ID to the PEP (via the CH) which is informed about this decision and performs the `StartAccess`. Supposing a permit decision has been received, the Broker informs the Subscriber about the successful subscription and starts to send data related to the topic when available, eventually stimulating Publishers in an idle state. To illustrate the `revoke` workflow, we suppose that one of the attributes relevant for the Subscriber policy change its value `(OnAttributeUpdate)`. This causes the PIP to send this new attribute to the CH that forwards it to the PDP for reevaluation. Supposing that the value of this attribute leads to a conclusion that this session must be revoked (Deny decision), the CH invokes the `RevokeAccess` on the PEP, also informing the Subscriber that the access is no longer granted `(RevokeAccess)`. The termination of the access could happen also if the Subscriber is no longer interested to the data, invoking the `Unsubscribe`. The unsubscribe triggers the PEP to send an `EndAccess` to the CH. The latter informs the PIP to take the last value of the attribute `(PostAttributeUpdate)`.

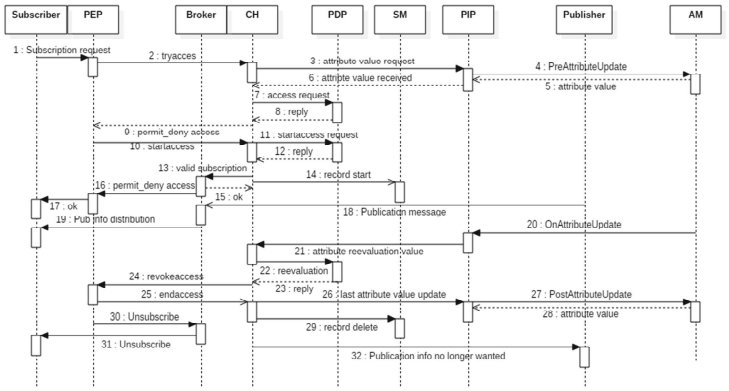

Fig. 2. Full workflow sequence diagram.

Also the UCS informs the Broker that the Subscriber is no longer subscribed and forces the unsubscription. Moreover, the SM is also informed that this session is over so that the record should be archived or deleted. Finally, if this Subscriber is assumed to be the only one that was interested to the Publisher, the Broker informs him to stop data publication due to fact that there is no more any interest from any Subscriber.

4 Results

To demonstrate the viability of the proposed approach, the overhead introduced by usage control has been measured in a simulated and in a real environment. The framework has been tested in two different environments: the first one is a virtual machine installing Ubuntu 16.04 64-bit, equipped with an Intel i7-6700HQ with 8 cores enabled, 8 GB DDR4 RAM running in 2133 MHz, the second one is a Raspberry Pi 3. In Fig. 3, there is reported the performance variation at the increase of the number of attributes used in the policy. As shown, the timing behavior is almost linear to the amount of attributes, which is expected, due to the longer time needed to collect a larger number of attributes and for the evaluation performed by the PDP. However, in the real case, even considering 40 attributes, the timings are still acceptable for most of applications. Moreover, it is worth noting that policies with a large number of attributes such as 40 are quite unusual [10]. As expected, the low computational power of the Raspberry alongside the existence of a real network among the MQTT components, justify the longer timings than in the simulated environment. Considering the subscription time, we see that there is some overhead caused be UCS. This is not be considered as a constraint because, since the Broker provides a buffer, we can still send all the published messages between the time of the request and the actual acceptance of the Subscriber. This causes no packet loss to the Subscriber and high QoS. Furthermore, the most significant time is the one of

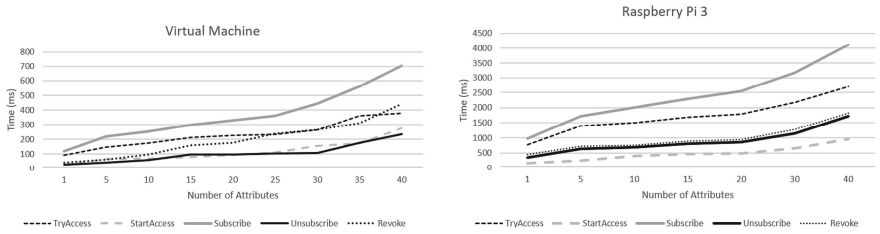

Fig. 3. Timings on the simulated and real testbed.

the revocation. This time is in fact the actual time in which the policy is violated and should be minimized. As shown, this time is equivalent to 216 ms in the real use case and 27 ms in the Virtual Machine, considering a policy with a single attribute. For several applications, this time can be considered as negligible. As shown, the time between a non valid value is taken and revocation of the access is very small. Finally, it is worth mentioning that in the ongoing phase, i.e. after a successful `StartAccess`, no delay is introduced by UCON while delivering messages to the Subscribers independently also of the number of attributes.

5 Related Work

Although there exist applications of UCON in GRID [11] and Cloud [2] systems, there is only one targeting on IoT [10]. In this paper the authors present a version of Usage Control called UCIoT that aims to bring the UCON on IoT architectures. Their work mainly focuses on an implementation of UCIoT in a smart home environment. They present specific policies alongside experiments on testbed. They do not, though, state how this framework (UCIoT) can be applied to the several application protocols. Their implementation in a P2P environment does not state how the UCON framework deals with each protocol. Our solution addresses this lack of addressing these protocols and how UCON can work alongside them. The authors of [8] propose a solution to securing Smart Maintenance Services. Their goal is to proactively predict and optimize the Maintenance, Repair and Operations (MRO) processes carried out by a device maintainer for industrial devices deployed at the customer. They focus on the MQTT routing information asset and they define two elementary security goals regarding the client authentication. Their solution is based on Transport Layer Security (TLS) which is already a basic feature of the protocol. They proposed on how to use it more efficient as a hardware element. Although they claim that the performance impact is not significant, the adoption of an extra hardware component might be critical in the constrained environment of IoT.

6 Conclusion

In this paper we have presented a first preliminary effort to increase the security of the MQTT protocol, by enabling the dynamic enforcement of Usage

Control policies. We have presented a general methodology which allows to integrate UCON in a seamless way, without requiring protocol modifications. A real implementation has been presented, with performance evaluation to demonstrate the viability of this approach. As future work we plan to test the presented framework on a larger testbed with a larger number of Publishers and Subscribers for the definition and enforcement of more complex policies, with a possible evaluation in a real applicative setting. Furthermore, we point out that the applied methodology can be easily extended to other IoT application protocols, where the benefits of integration are worth to be investigated in future works.

References

1. Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., Ayyash, M.: Internet of Things: a survey on enabling technologies, protocols, and applications. IEEE Commun. Surv. Tutorials **17**(4), 2347–2376 (2015)
2. Carniani, E., D'Arenzo, D., Lazouski, A., Martinelli, F., Mori, P.: Usage control on cloud systems. Future Gener. Comput. Syst. **63**(C), 37–55 (2016)
3. Chen, D., Varshney, P.K.: QoS support in wireless sensor networks: a survey. In: International Conference on Wireless Networks, vol. 233, pp. 1–7 (2004)
4. Colitti, W., Steenhaut, K., De Caro, N., Buta, B., Dobrota, V.: Evaluation of constrained application protocol for wireless sensor networks. In: 2011 18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), pp. 1–6, October 2011
5. Fysarakis, K., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., Katos, V.: Which IoT protocol? Comparing standardized approaches over a common M2M application. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2016)
6. Karagiannis, V., Chatzimisios, P., Vzquez-Gallego, F., Alonso-Zrate, J.: A survey on application layer protocols for the Internet of Things. Trans. IoT Cloud Comput. **1**(1), 11–17 (2015)
7. Lazouski, A., Martinelli, F., Mori, P.: Usage control in computer security: a survey. Comput. Sci. Rev. **4**(2), 81–99 (2010)
8. Lesjak, C., Hein, D., Hofmann, M., Maritsch, M., Aldrian, A., Priller, P., Ebner, T., Ruprechter, T., Pregartner, G.: Securing smart maintenance services: hardware-security and TLS for MQTT. In: 2015 IEEE 13th International Conference on Industrial Informatics (INDIN), pp. 1243–1250, July 2015
9. Luzuriaga, J.E., Perez, M., Boronat, P., Cano, J.C., Calafate, C., Manzoni, P.: A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. In: 2015 12th Annual IEEE Consumer Communications and Networking Conference (CCNC), pp. 931–936, January 2015
10. La Marra, A., Martinelli, F., Mori, P., Saracino, A.: Implementing usage control in Internet of Things: a smart home use case. In: 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, Australia, 1–4 August 2017, pp. 1056–1063 (2017)
11. Martinelli, F., Mori, P.: On usage control for grid systems. Future Gener. Comput. Syst. **26**(7), 1032–1042 (2010)

12. Thangavel, D., Ma, X., Valera, A., Tan, H.X., Tan, C.K.Y.: Performance evaluation of MQTT and CoAP via a common middleware. In: 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6. IEEE (2014)
13. Villari, M., Celesti, A., Fazio, M., Puliafito, A.: Alljoyn lambda: an architecture for the management of smart environments in IoT. In: 2014 International Conference on Smart Computing Workshops, pp. 9–14, November 2014