The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices

Giovanni Perrone[1], Massimo Vecchio[1], Riccardo Pecori[1] and Raffaele Giaffreda[2]
[1]*SMARTEST Research Centre, eCampus University, Via Isimbardi 10, 22060, Novedrate (CO), Italy*
[2]*OpenIoT Research Area, FBK CREATE-NET, Via alla Cascata 56/D Povo, 38123, Trento (TN), Italy*

Keywords: Internet of Things, Security, Access Control, Publish-subscribe, Open Source MQTT Broker.

Abstract: Recent news of massive Distributed Denial of Service (DDoS) attacks being carried out using thousands of Internet of Things (IoT) devices transformed into attack bots are nothing else than a wake-up call for all the actors having a role on the IoT stage. The need to define and establish, as quickly as possible, viable security standards able to cope with the heterogeneous requirements arising from the IoT world is urgent, now more than ever. Maybe even before that, the dissemination of basic knowledge connected with the culture of IT security seems to play a major role in the overall security balance for IoT. Since it is more likely that systems using lightweight devices can be more vulnerable to security attacks, in this paper we start with analyzing MQTT, a message-based communication protocol explicitly designed having low-end devices in mind. After that, we move on to describe some of the security solutions and improvements typically suggested and implemented in real-life deployments of MQTT. Finally, we conclude this paper with a concise, though not exhaustive, survey on some of the most promising research topics in the IoT security area.

1 INTRODUCTION

Analysts around the world all agree on the fact that the next few years will see a massive increase in the Internet of Things (IoT) penetration and, consequently, in the number of devices active and connected to the Internet. The Ericsson Mobility Report 2016, in its IoT chapter, presents data stating that, by 2021, there will be a total of 28 billion IoT devices active and interconnected (Ericsson, 2016). Even if different analysts and reports provide many forecasts, all of them agree on large growth rates for the IoT by the next 5 years.

At the same time, the month of October 2016 has seen the largest Distributed Denial of Service (DDoS) attack taking place, with unprecedented volumes of data used to knock-down various Internet services in the United States and in Europe (US-CERT, 2016). At the time of writing this paper, it has been ascertained that the attack has been carried out using a malware named *Mirai*, being specifically designed to attack and hijack IoT devices, transforming them into bots that can be later used to carry out coordinated attacks. The source code of Mirai has been publicly released on the open source community at the end of September 2016; an analysis of its structure reveals that Mirai is designed to carry out a scan of the Internet, looking for devices responding on standard ports. Once a device is found, the malware tries 66 hard-coded combinations of user IDs and passwords to gain access as administrator. If one of these combinations works, the device is infected, normal administration services (*e.g.*, telnet and http-based consoles) are disabled and the unit waits for further instructions from the command and control (C&C) center.

The dangers related to what happened are not only connected with the immediate effects of the attack, which have been large and lasted for several hours, but mainly to its very nature: a large number of IoT devices has been transformed into an army of malevolent agents. The root causes that allowed this situation can be summarized in the following:

1. the lack of clear and widely adopted security standards for IoT devices, which are rarely released with out-of-the-box security features;
2. the fact that the vast majority of IoT platforms are deployed and configured (sometimes even designed) without bearing security in mind.

The last statement is also supported by a recent analysis carried out and presented as a session during the 24^{th} DEFCON hackers forum held in Las Vegas

Perrone, G., Vecchio, M., Pecori, R. and Giaffreda, R.
The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices.
DOI: 10.5220/0006287302460253
In *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017)*, pages 246-253
ISBN: 978-989-758-245-5

in August 2016, (DC24, 2016). While the whole forum was dedicated to security in the IoT/M2M area, one of the sessions explicitly targeted the security of MQTT-based systems. The analysis used Internet-wide scans made with both Shodan[1] and Masscan[2] (two scanning tools designed to analyze all Internet addresses and to look for specific ports/services left open). The ultimate goal of the analysis was to discover MQTT-based systems exposed to the Internet and to evaluate the protection status of the systems and of the data stored into them. Unfortunately the results are quite discomforting: thousands of MQTT brokers have been found with little or no authentication or access control mechanisms in place. Even worse, in several cases it was possible to obtain full access to the data transmitted.

The scenarios depicted by these analysis are corroborated by the easiness Mirai has been able to spread with and are definitely concerning, especially in the light of the above mentioned IoT growth rates; the combination of these two factors should raise the importance of security and protection in the IoT world to a top-priority position. Yet, in many proposed architectures, security is not taken into account at all, focusing only on functionalities, ease of use and low costs. The IoT dream foresees a world where people are connected in a *seamless* way to every-day objects; this also implies easiness of setup, installation and maintenance in order to allow everyone to be able to, for instance, setup and operate an array of devices in a smart house scenario. In our opinion, this means that the burden of securing the systems should be moved away from the end-user, who will look more and more for a plug-and-play device, to the system itself and it should be implemented *by design*.

In this scenario, an IoT ecosystem should come with a set of security functionalities that have to be the result of a careful balance and trade-off among:

- guaranteeing a "good enough" level of security, possibly upgradable with future releases;
- maintaining "simplicity of use and installation" for end users;
- having all of this (plus the product features to be provided to the users) deployed in small, constrained, battery-powered devices.

To approach the IoT security chapter as a whole would be a titanic job, while recent papers surveying several works lying under this research and development umbrella are already available in the literature (Sicari et al., 2015). Therefore, we narrow our focus on MQTT, a lightweight publish-subscribe messaging protocol that is rapidly becoming a *de facto* standard in the IoT world. Thus, we will first provide an overview of the MQTT protocol in Section 2, where we will highlight its main security weaknesses as well. Then we will present a summary of the standard security measures typically found in real-life applications in Section 3, while in Sec. 4 we will discuss the security functionalities provided by some of the cutting-edge open-source MQTT implementations available, together with an overview on some of the most promising solutions described in other researches. Finally, we will draw some useful conclusions in Section 5.

[1] http://www.shodan.io/

[2] http://github.com/robertdavidgraham/masscan

2 MQTT

MQTT (Message Queue Telemetry Transport) is a *lightweight* message-based communication protocol based on a publish-subscribe paradigm. This section focuses on the basic concepts of the protocol and its specific aspects relevant to security.

2.1 General Features

MQTT has been designed to use as little bandwidth as possible: specifically, the primary requirement for this protocol was to use less bandwidth than that required to carry out the same actions using HTTP or similar protocols. It has been successfully used in several areas where the communication links provided low throughput, *e.g.*, automation, SCADA monitoring, etc., or suffered from low availability, *e.g.*, satellite links. Starting from version 3.1, MQTT has become an OASIS standard and the latest release of the specifications refers to version 3.1.1 of the protocol (Oasis, 2014). From a protocol viewpoint, MQTT stands on top of TCP, therefore delegating flow and error control for the single packets to the lower layers of the protocol stack.

As mentioned above, MQTT is based on a publish-subscribe paradigm; this means that:

- information is classified in *topics*: there is no rule nor recommendation for topic structures, but hierarchies are allowed through separators (like a path in a file system);
- nodes (or clients) can obtain access (reading action) to the information published in a specific topic by *subscribing* to that topic: the subscribe action is carried out sending a *SUBSCRIBE* message to the MQTT broker (or server) that will add the requesting node (if allowed by the possibly enacted access control rules) to the distribution list of those nodes having access to that topic;

- nodes can create contents (writing action) by *publishing* information to a specific topic: the publish action is carried out by sending a *PUBLISH* message to the broker; the broker does not carry out any processing on the message transmitted and relays the payload to all nodes that previously subscribed to the topic.

This model allows to decouple clients (nodes) and server (broker), both from a spatial and a temporal standpoint. In fact, the only entity in an MQTT network who has full knowledge of the composition of the network itself is the broker (no client-to-client connection is possible). Moreover, the reception and dispatching of messages take place in two separate moments; dispatching can also be delayed in order to wait, for instance, for a node to become available.

2.2 Security Aspects in MQTT Standard

The official MQTT specifications include no mandatory requirements for any of the typical security related aspects such as authentication, authorization, data integrity, confidentiality and the like. The lack of security-related functionalities in the standard are related to:

- the fact that MQTT focuses only on message dispatching;
- the willingness to keep the protocol as light as possible, therefore reducing the overhead related to security features;
- the fact that the historical implementations of MQTT for telemetry were based, at least initially, on private networks;
- the fact that MQTT is used in a very heterogeneous range of scenarios, from IoT devices to Facebook messenger mobile application, that of course require significantly different security functionalities to be rendered secure.

The only authentication-related requirement is the (optional) possibility to specify a user-name and password during the initial connection phase between a node and the broker. As said, the presence of these fields is considered optional and has to be notified to the broker by setting the relevant option flags in the *CONNECT* packet header. Moreover, both the user-ID and password fields are transmitted in plain text, making possible the eavesdropping of the credentials by using a simple protocol sniffer. Some implementations favour even more simplicity against security and use the client-ID field to authenticate the device. Since all nodes must possess a unique client-id that is communicated to the broker in the *CONNECT* message, it is possible to use this field alone for authentication of the node. In some implementations the client-id is the MAC address of the network card used to connect the node, therefore providing uniqueness of the field. While this can ease the authentication process in some simple applications, since the client-id (just like the user-ID and password) is transmitted as plain text, once again there is no protection against eavesdropping, sniffing, man-in-the-middle attacks and the like.

At the same time, the protocol standard highlights that "MQTT solutions are often deployed in hostile communication environments", thus, authentication, authorization, packet integrity and privacy are, if not required, at least strongly recommended. A dedicated OASIS sub-committee is currently working on recommendations for security specifications to be added to the basic MQTT protocol, but approaching and solving the different security issues related to the utilization of MQTT remains a project (or implementation)-specific matter, with no real standard solution.

On top of what a "normal" implementation of MQTT entails, another interesting aspect, which may affect security and is completely not covered by MQTT standard, is brokers' interconnection, wherein two or more brokers are connected to each other through a "bridge" or by forming a cluster. Typically, a bridge is a connection based on topic exchange between two brokers, where one of them behaves as a user node, subscribing and publishing to a given set of topics. Clusters typically offer more advanced functionalities like data propagation management and workload balancing. These forms of broker aggregation could be useful, from a security viewpoint, to enact single sign-on and fail-over procedures, anyway they are implementation-specific features as the most of the security capabilities currently put in place. In the next section we will cover in more detail some security features offered by some of the open-source MQTT implementations currently available.

3 SECURITY REQUIREMENTS AND SOLUTIONS FOR MQTT IMPLEMENTATIONS

The security requirements being specific for MQTT can be categorized in the following areas:

- **authentication:** warranting the identity of the nodes belonging to the MQTT network, in order to

prevent unauthorized accesses (both as subscribers or publishers);

- **access control:** guaranteeing access to information only to the nodes allowed to get to that information;
- **data integrity:** guaranteeing that the data being received are actually what has been transmitted by the source (*e.g.*, no tampering during transmission);
- **confidentiality:** prevention of data sniffing and, more in general, protection of data secrecy and privacy.

The aforementioned categories can surely make up the basis for security-related assessments of any IT environment but, when applied to MQTT implementations, additional considerations have to be made.

First of all, it has to be considered that the main goal of MQTT is to minimize bandwidth occupation in order to reduce communication channel utilization and, most importantly, power consumption. Even though nothing prevents using MQTT in scenarios where power and bandwidth consumptions do not represent a major concern, we will consider a typical MQTT deployment as constituted of a number of small, battery-powered devices using wireless communication links. In such a situation, computational resources and overall power consumptions have to be considered limited. Moreover, active devices will be often deployed in a dispersed (*i.e.*, not concentrated) geographical region, meaning that reaching all the units may be difficult to realize, if not unfeasible. Similar considerations can be drawn for remote administration of the devices: not only for security reasons, but also to limit bandwidth and power, it may be not practical to rely on frequent massive update operations of the firmware.

In this section we list some of the solutions that can be typically found in MQTT systems to implement different levels of protection from various security threats. More in detail, we introduce protection mechanisms based on standard or widely spread technologies and/or solutions that can be found in real-life applications.

3.1 Network Layer Security

At network layer IPSec is surely the best solution to provide authentication and confidentiality. It could be employed in end-to-end communications and in device-to-broker transmissions and implemented directly in the stack of tiny devices even if giving up key exchange protocols. However, some problems may regard the overhead due to extra headers, issue partly solved by employing 6LowPAN compression, and the usage of computationally intensive cryptographic algorithms, theme discussed in the following.

Another viable solution at network level could be Host Identity Protocol (HIP) where locators of nodes and broker could be decoupled from their own identifiers. In this case as well, the problem could be in the cryptographic algorithms to be employed as well as in the pubic key distribution, but the advantages may relay into multihoming and host mobility.

3.2 Transport Layer Security

As regards security at transport layer, OASIS explicitly recommends the utilization of TLS and certificates, whenever possible, in order to have a viable solution for authentication, data integrity and confidentiality. However, due to the computational and power consumption limitations typically present in MQTT-operated devices, the introduction of TLS can be problematic, due to the additional workload required to setup the secured connections and to cipher all traffic. TLS Session Resumption capability can be used in all these situations, significantly reducing the frequency of the complete TLS Handshake procedure (and of its associated computational load). TLS implementations in hardware are also analyzed in the literature, in order to alleviate the problems related to the computational load due to the ciphering mechanisms implemented (Lesjak et al., 2015). Using an hardware-based TLS implementation could represent a solution to move away this additional workload from the processing device, but certainly requires an additional component that has to be deployed and that will increase the overall device cost and power consumption.

Notwithstanding its widespread use, some scientists, like Singh et al. (2015), raise concerns in the very first place about the utilization of TLS at all for IoT applications. Those concerns are mainly related to:

- **Configuration**: improper configuration of the TLS layer or even just the utilization of weaker or obsolete cipher suites can significantly lower the security of the protocol exposing it to attacks;
- **TLS vulnerabilities**: being the most used security protocol employed for payments over the Internet, TLS is extensively tested for its security and, not surprisingly, there are several vulnerabilities known for it. Sheffer et al. (2015) summarize known attacks that have been exploited against TLS. Many of these attacks are related to weaker cipher suites supported in previous releases of

TLS or SSL or supported back-compatibility with older releases of the protocol;

- **Certificates management**: several scientists raise concerns related to the certificates management, once the initial deployment is completed: updating, revoking or managing the certificates can be a hard task to achieve with dispersed devices.

The problems related to the management of certificates installed on remote devices is also common to the first two points. In fact, executing periodic configuration and cipher updates on a large number of devices over not completely reliable links can be a problematic task to be carried out in an controlled way, and the additional traffic has to be considered when analyzing throughput and battery consumption factors. All of the above notwithstanding, virtually all MQTT broker implementations, both open-source and commercial solutions, fully support TLS utilization, both for ensuring data confidentiality and authentication of the nodes through the certificates presented at the moment of the TLS handshake.

3.3 Application Layer Security

Within the application layer, systems that have to deal with multiple items connecting, and therefore subject to the authentication process, may decide to use a centralized authentication system, external to the broker itself. For these applications, the MQTT standard explicitly mentions the possibility of using LDAP or OAuth authentication systems. In both cases, an external system is responsible for authentication and granting of access tokens based on the credentials provided. The implementation of such systems introduces necessarily an additional layer of complexity, both in terms of configuration and management, and the pros and cons have to be accurately weighted during the design phase. Moreover, it has to be considered that both LDAP and OAuth are normally implemented on top of a TLS connection, bringing up the potential issues highlighted in the previous section. Notwithstanding all these considerations, as we will see in Section 4, LDAP is often directly supported by MQTT implementations, while the support for OAuth is still somehow less common.

3.4 Ciphering

It is possible to use symmetric ciphering to protect the credentials and/or the payloads. While symmetric ciphering algorithms use less computational resources than public key ciphering, it is clear that the key has to be known to all involved parties and once again we are faced with the same challenges mentioned above and related to perform system-wide updates of keys. Moreover, a compromised key would automatically render the whole system non-secure. Nevertheless various implementations allow different security scenarios based on symmetric encryption.

Some deployments implement an end-to-end ciphering mechanism: in these cases, the nodes cipher transmitted payloads using a chosen cipher suite unknown to the broker, but known only to the nodes that are supposed to receive the information. In this situation the broker, which could represent a security single point of failure, becomes less critical, as an attacker that gains access to its message queues cannot achieve access to the information shared on the network.

Conversely, other scenarios provide the broker with ciphering and deciphering capabilities and therefore with the ability to decide whether to deliver encrypted received packets as a plain text or again as a cipher text. In such environments, the broker could also have the chance to change the cipher suite, thus differentiating both sides of the communication. In such cases, the broker represents a critical single point of failure and effective security mechanisms for its storage system should be enacted.

Some important lightweight symmetric ciphering solutions, usable within MQTT, may entail Tiny Encryption Algorithm (TEA) family (TEA, XTEA, TinyXTEA, etc.), based on XOR and shift operations, Scalable Encryption Algorithm (SEA), suitable for different platforms and implementations, PRESENT, HIGHT and the like.

All in all, as said, all solutions relying on public cryptography need opportune asymmetric algorithms that are not so computationally intensive as the most traditional ones. For MQTT implementations a promising solution could surely be Elliptic Curve Cryptography (ECC) with its own shorter keys, together with a web-of-trust or group-based key distribution strategy for certificate or ID management.

3.5 Physical Layer Security

In the first industrial applications of the predecessors of MQTT, an inherent level of security was introduced by operating the systems on private networks. The same concepts can be applied nowadays to a normal MQTT deployment, therefore delegating the protection confidence from the components of the MQTT system to the network connecting them. Some scientists suggest the use of an onion or peer-to-peer structured network on top of other solutions (Weber, 2010), while others propose architectures based on devices

creating a point-to-point network, thus limiting those risks connected to shared communication channels (Espinosa-Aranda et al., 2015). In all these cases, security is achieved by *physical segregation* of the units, something which may be impossible in some applications (*e.g.*, sensors installed in wide open areas) or simply too risky, if the installation environment is not protected enough.

3.6 Authorization and Access Control

Access control (AC) and authorization refer to privileges and allowed actions over a certain resource, and may be subject-based or object-based. Generally, while in the former the focus is on active entities (*i.e.*, the subjects) that own certain access capabilities over specific passive entities, the latter concentrates on the resources (*i.e.*, the objects) to be accessed. In this case, AC is usually described by means of Access Control Lists (ACLs). A well-known security structure able to fulfill both subject and object-based AC approaches is the Access Control Matrix. Other possible implementation strategies are based on mapping permissions into *attributes* and on the the definition of *roles* and *groups* both for subjects and/or objects. In the case of MQTT, the subjects are the users and the objects are the topics the users may subscribe to, while the permissions usually encompass either the ability to *read* a certain topic (*SUBSCRIBE*) or to *write* into a certain topic (*PUBLISH*).

AC, even though strongly related to a prior authentication phase, is not mentioned at all in the OASIS MQTT standard and the implementation of any AC mechanism (being it subject or object-based) is strongly dependent on the specific broker used. The next section summarizes the core features of the most commonly employed open-source MQTT brokers, with an emphasis on the security functionalities they can provide.

4 CURRENT OPEN-SOURCE MQTT IMPLEMENTATIONS

Mosquitto is one of the most commonly used implementations, mainly for its configuration simplicity and its light footprint (Mosquitto, 2016). TLS is supported as well as basic user authentication capabilities based on user-password (with a credential file stored on the broker). This implementation allows for single key ciphering but no single sign on in the bridge interconnections. Regarding access control, both generic (*i.e.*, valid for all users), as well as user-specific rules are allowed; in both cases it is possible to specify read or write access at topic level, using wildcards for the inclusion of multiple topics as specified in MQTT standards. Mosquitto also supports two specific types of wildcards useful in certain implementations to allow controlled access to topics' trees. Mosquitto has a plug-in architecture that can be used to expand its basic features, allowing therefore to add functionalities such as support for LDAP or OAuth platforms, which are however not supported directly by the distribution.

eMQTT is another full open-source MQTT broker (EMQTT, 2016). TLS is fully supported and a module for LDAP integration is available, however preshared key ciphering, single sign on and centralized access control are not implemented. Compared to Mosquitto, available access control rules follow a different syntax, but provide similar granularity, allowing therefore to create regulations that are user and/or topic-based. It is also possible to use one of the existing database plug-ins to connect to MySQL, PostgreSQL or Redis data structures to store and retrieve users-related information. LDAP is supported natively through one of the plug-ins available 'out of the box' in the distribution, while there is no direct support for OAuth.

Apollo is a broker implementation based on ActiveMQ, a larger project on message based protocols from the Apache foundation (Apollo, 2016). It inherits many of the functionalities available for ActiveMQ, introducing at the same time a number of features designed for MQTT and for the other supported protocols. Since ActiveMQ is equipped with JAAS modules for authentication, this functionality is present in Apollo as well, furnishing therefore the possibility of using the different login modules present in the JAAS framework (including native support for connection to an LDAP server). Apollo does also support the usage of TLS and of preshared key ciphering, while no clustering or bridging are allowed and therefore no single sign on or centralized access control policies may be enacted. The access control engine allows to create rules with several actions, including administrative ones, such as the creation or destruction of queues, and consumption of data in the queues without removing the data read. Rules are activated on the basis of the identity of the user accessing the system, upon authentication. Plug-ins to add support for OAuth are available but are not included in the official distribution.

Artemis is another implementation arising from ActiveMQ (Artemis, 2016), from which it inherits the use of JAAS for authentication, support for TLS

and single key ciphering. The creation of role-based access control rules is allowed, pretty much like what happens with Apollo. One of the main differences between Apollo and Artemis is that the latter provides advanced features for bridging and clustering, enabling the creation of high-availability entities based on the auto-discovery of the nodes, as well as single sign on and load balancing capabilities. OAuth support is not directly included but can be added through additional modules.

4.1 Other Solutions and Ongoing Researches

Several EU-funded projects are related to IoT security, such as: FP7 IoT-A, COMPOSE, iCORE, IoT.EST, Ebbits, uTRUSTit, Butler (CORDIS, 2016). Within the framework provided by these research projects and in parallel to them, a large number of research initiatives are currently running to identify and present innovative solutions in the IoT security area. The already mentioned (Sicari et al., 2015) provides an excellent overview on some of the activities going on in that area. In this paper we limit our exploration, deliberately not exhaustive, to the MQTT reference application scenario we have detailed at the beginning. In this domain, a consistent amount of research activity is focusing onto analyzing and proposing extensions to a typical MQTT platform to implement additional security functionalities not present in the standard protocol specification. These extensions can take the form of *plug-ins* (or modules) connecting to existing broker implementations or middleware modules that are placed between nodes and brokers and control resources utilization based on security policies.

Rizzardi et al. (2016) present a middleware-based solution called AUPS based on NOSs (Networked Smart Objects), a concept already presented in previous works by the same authors. AUPS supports keys management and security policies. All the security aspects are delegated to a network of multiple NOSs currently implemented in Java, that act as intermediate layer between different data sources and the MQTT broker. In this scenario, data sources do not use native MQTT pub-sub primitives to publish data, rather they employ HTTP protocol to communicate data to the NOSs. After processing the applicable security policies, data are then posted by the NOS core to the broker, which remains external. This solution is independent from the broker implementation and does not add any computational load on the broker itself, but the introduction of the middleware layer represents a deviation from a standard MQTT deployment and its compatibility should be evaluated.

Neisse et al. (2015) describe and assess the implementation of SecKit, a security solution presented as a model-based security toolkit for IoT. SecKit is a module that enforces access control rules based on security policies with the ultimate goal of providing access control tools, as well as policy protection. Specifically, their work describe the implementation of SecKit as a Mosquitto plug-in, thus preserving the fundamental architecture of a typical MQTT system. At the same time, the additional workload required to process security policies is imposed on the broker, and the possible impacts in terms of scalability for large applications have to be evaluated properly.

Chase (2007) elaborates on a concept originally proposed in (Sahai and Waters, 2005) and related to attributes based encryption (ABE), i.e., an encryption method applicable to scenarios with different recipients (like in MQTT) receiving broadcast messages. In such a scenario, the recipients will be able to decipher the messages based on the possession of a given number of attributes related to their identity. In other words, each recipient has a set of attributes (*e.g.*, type of node, clearance level, etc.) and the messages are encrypted using methods that allow decryption only to recipients that possess a minimum number of those attributes. An implementation of this concept is provided in (Bethencourt et al., 2007), where the performances of the model are discussed as well. However, we have not been able to retrieve information on the results of the application of the model on a real-life MQTT system.

5 CONCLUSIONS

The security related aspects appear still to be an open field of research and discussion for the IoT world. The large number of possible solutions and the lack of (at least) a *de facto* standard represents a security threat itself. The recent release of the special publication from NIST (Ross et al., 2016) on Systems Security Engineering may represent a significant step in the right direction, though still confined to a series of voluntary recommendations, far from being able to fix the fundamental issues at hand.

Recent events (and in particular the Mirai attack that took place in October 2016) highlight the necessity and the importance of making sure that at least basic protection measures (such as changing the default admin credentials for the central nodes of the networks) are used in order to limit exposure to massive and global attacks. The implementation of such a basic protection should happen *by design* in all sy-

stems and should be enforced by all the actors involved in production and manufacturing of components that can be used to create an IoT ecosystem. Considering the projected growth rate of these devices and the open nature of the hardware and software components that are typically used, it would appear that this shift in mentality should have highest importance and priority in everyone who is involved in the IoT arena. Based on the *status quo* and on the fact that Moore's law certainly applies also to small-size devices, the authors are inclined to believe that a good mid-term solution to large-scale MQTT security problems could be represented by implementation of TLS. Strength points of this approach would be the utilization of a standard technology, continually tested for weaknesses by the global IT security community; this would allow easier implementation and interconnection of different systems. At the same time, certificates management may remain an obstacle to a wide-spread adoption of TLS, especially in all cases where low-throughput networks (e.g., LPWANs) are used. For these situations, single key ciphering or network segregation could represent a more viable solution.

REFERENCES

Apollo (2016). Apache ActiveMQ Apollo homepage. available: http://activemq.apache.org/apollo/. accessed: March 21, 2017.

Artemis (2016). Apache ActiveMQ Artemis homepage. available: http://activemq.apache.org/artemis/. accessed: March 21, 2017.

Bethencourt, J., Sahai, A., and Waters, B. (2007). Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE.

Chase, M. (2007). Multi-authority Attribute Based Encryption. In *Proceedings of the 4th Conference on Theory of Cryptography*, TCC'07, pages 515–534, Berlin, Heidelberg. Springer-Verlag.

CORDIS (2016). CORDIS: Community Research and Development Information Service. available: http://cordis.europa.eu/. accessed: March 21, 2017.

DC24 (2016). The DEFCON homepage. available: http://www.defcon.org/.

EMQTT (2016). EMQTT homepage. available: http://emqtt.io/. accessed: March 21, 2017.

Ericsson (2016). Ericsson Mobility Report. available: http://www.ericsson.com/res/docs/2016/ericsson-mobility-report-2016.pdf. accessed: March 21, 2017.

Espinosa-Aranda, J. L., Vallez, N., Sanchez-Bueno, C., Aguado-Araujo, D., Bueno, G., and Deniz, O. (2015). Pulga, a tiny open-source MQTT broker for flexible and secure IoT deployments. In *2015 IEEE Conference on Communications and Network Security (CNS)*, pages 690–694.

Lesjak, C., Hein, D., Hofmann, M., Maritsch, M., Aldrian, A., Priller, P., Ebner, T., Ruprechter, T., and Pregartner, G. (2015). Securing smart maintenance services: Hardware-security and TLS for MQTT. In *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pages 1243–1250.

Mosquitto (2016). Mosquitto homepage. available: http://mosquitto.org/. accessed: March 21, 2017.

Neisse, R., Steri, G., Fovino, I. N., and Baldini, G. (2015). SecKit: A Model-based Security Toolkit for the Internet of Things. *Computers & Security*, 54:60–76.

Oasis (2014). MQTT Version 3.1.1 Specifications. available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. accessed: March 21, 2017.

Rizzardi, A., Sicari, S., Miorandi, D., and Coen-Porisini, A. (2016). AUPS: An Open Source AUthenticated Publish/Subscribe system for the Internet of Things. *Information Systems*, 62:29–41.

Ross, R., McEvilley, M., and Carrier Oren, J. (2016). NIST Special Publication 800-160: Systems Security Engineering Considerations for a Multidisciplinary Approach in the Engineering of Trustworthy Secure Systems. available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-160.pdf. accessed: March 21, 2017.

Sahai, A. and Waters, B. (2005). Fuzzy identity-based encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473.

Sheffer, Y., Holz, R., and Saint-Andre, P. (2015). Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS). *Internet Engineering Task Force (IETF)*, Request for Comments: 7457.

Sicari, S., Rizzardi, A., Grieco, L., and Coen-Porisini, A. (2015). Security, privacy and trust in Internet of Things: The road ahead. *Computer Networks*, 76:146–164.

Singh, M., Rajan, M., Shivraj, V., and Balamuralidhar, P. (2015). Secure MQTT for Internet of Things (IoT). In *2015 5th International Conference on Communication Systems and Network Technologies*, pages 746–751.

US-CERT (2016). Alert (TA16-288A): Heightened DDoS Threat Posed by Mirai and Other Botnets. available: http://www.us-cert.gov/ncas/alerts/TA16-288A. accessed: March 21, 2017.

Weber, R. (2010). Internet of Things–New security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30.