Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System

Syaiful Andy[1, a], Budi Rahardjo[2, b], Bagus Hanindhito[3, c]
[1 2 3] *Department of Electrical Engineering, School of Electrical Engineering and Informatics*
Institut Teknologi Bandung, Bandung, Indonesia
[a] *Email: syaifulandy@gmail.com,* [b] *Email: rahard@gmail.com,* [c] *Email: hanindhito@bagus.my.id*

***Abstract*—Various communication protocols are currently used in the Internet of Things (IoT) devices. One of the protocols that are already standardized by ISO is MQTT protocol (ISO / IEC 20922: 2016). Many IoT developers use this protocol because of its minimal bandwidth requirement and low memory consumption. Sometimes, IoT device sends confidential data that should only be accessed by authorized people or devices. Unfortunately, the MQTT protocol only provides authentication for the security mechanism which, by default, does not encrypt the data in transit thus data privacy, authentication, and data integrity become problems in MQTT implementation. This paper discusses several reasons on why there are many IoT system that does not implement adequate security mechanism. Next, it also demonstrates and analyzes how we can attack this protocol easily using several attack scenarios. Finally, after the vulnerabilities of this protocol have been examined, we can improve our security awareness especially in MQTT protocol and then implement security mechanism in our MQTT system to prevent such attack.**

Keywords—attack; MQTT; protocol; scenario

I. INTRODUCTION

Internet of Things (IoT) or inter-machine communication (M2M) over the internet is a concept that allows communication between devices over the Internet. The number of IoT devices is growing rapidly where Cisco IBSG predicts the number of IoT devices will reach 50 billion by 2020 [1]. Moreover, Gartner predicts, by 2020, the internet of things devices will be made up of 20.4 billion units [2]. IoT plays a major role in smart city implementation like smart home, smart transportation, and smart parking.

Nowadays, many protocols are used as a communication protocol in the IoT devices. Five of the most prominent protocols used for IoT is Hypertext Transfer Protocol (HTTP), Constrained Application Protocol (CoAP), Extensible Messaging and Presence Protocol (XMPP), Advanced Message Queuing Protocol (AMQP), and MQ Telemetry Protocol (MQTT) [3]. Some considerations that must be taken into account when we choose the protocol are energy efficiency (total consumed energy for the given execution time), performance (total transmission time it takes to send messages and receive their acknowledgments), resource usage (CPU, RAM, and ROM usage), and reliability (ability to avoid packet loss, i.e. QoS) [4]. Moreover, when advanced functionalities (e.g. message persistence, wills, and exactly once delivery), reliability, and ability to secure multicast message are highly considered, MQTT protocol is one of the best options [5].

A. MQTT Protocol

MQ Telemetry Transport (MQTT) is a messaging protocol using a publish/subscribe mechanism which is originally designed by Andy Stanford-Clark and Arlen Nipper. It is currently in the OASIS (Organization for the Advancement of Structured Information Standards) standard.

Currently, the MQTT protocol also has standard defined in ISO/IEC 20922: 2016 (Information technology - Message Queuing Telemetry Transport (MQTT) v3.1.1). This protocol is used widely for IoT system that has limited resources because of several reasons: lightweight, small bandwidth requirement, open and straightforward to be implemented [6].

Figure 1 shows the example of the usage of MQTT protocol. Publish and subscribe operations can be analogized like client and server models. The central server in MQTT is named broker that acts as the recipient of the message from the client which is, essentially, the entire node involved in the communication process [7]. The message itself can be in the form of publish or subscribe topic. Furthermore, all the devices connected using this protocol can become publishers and subscribers. Usually, in MQTT architecture, several sensors periodically publish the results of their measurements (i.e. payload data) to a topic address. Every device that has been registered as a subscriber to a specific topic will receive a message from the broker each time the topic is updated.

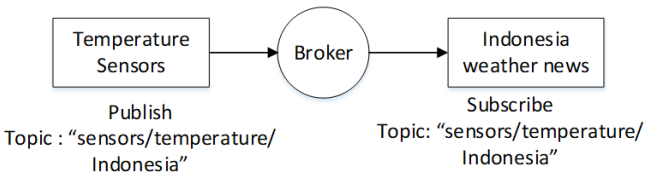

Fig. 1. An example of MQTT protocol use case.

B. Security Requirement and Attack Surface

Information security is also an important thing to consider during making the decision of the protocols because some of the communication protocols in the IoT devices do not have a comprehensive information security mechanism. According to a book published by ISACA [8], the object of information security consists of three components: data confidentiality, data integrity, and data availability. There is also access levels

978-1-5386-0549-3/17/$31.00 ©2017 IEEE

security requirements such as authentication, authorization, and access control which are explained in [9]. In fact, MQTT protocol is one of the protocols that do not yet have overall security mechanism because it only has authentication mechanism without encryption capabilities.

There are various considerations for IoT developer who wants to design security solutions in the IoT communication protocol. Firstly, the limitation of the IoT device itself (e.g. compute performance and low power consumption) that require a lightweight security protocol with small code footprint. Secondly, the heterogeneous environment where each of connected device may use different protocol and different security mechanism. Lastly, the reliability of network which may forces as to use security mechanism with minimum overhead [10].

By understanding the security requirement for IoT devices, we can now discuss the attack surface in IoT. Attack surface is a vulnerability that can be accessed and exploited in a system [11]. In [9], attack surface in IoT is divided into local network and public network. The local network is analog to internal attack where the attacker is on the same network as the IoT devices while the public network is analog to external attack where the attacker might reside anywhere in the public network to attack the IoT system [9].

Last year, a major incident related to IoT system was reported by RSA where the hackers had hijacked many IoT devices and provided access to compromised IoT devices and cameras in criminal forum [12]. Moreover, there was a distributed denial of service (DDoS) attack to krebsonsecurity.com site performed by botnets embedded in the IoT devices. Finally, taken from data owned by Threat Research Akamai team [13], there were reportedly millions of IoT devices used as proxies to route victims traffic to malicious sites.

II. Background

This section explains several reasons for why IoT implementation in the world does not use security mechanism.

A. Resource Constrained Device

There are many devices categorized as a constrained device which, according to RFC 7228 [14], is further divided into three classes based on their RAM and ROM as follows.

TABLE I. Class in Constrained Device (RFC 7228)

Class	RAM (Data Size)	Flash (Code Size)
Class 0	<< 10 KB	<< 100KB
Class 1	~ 10 KB	~ 100KB
Class 2	~ 50 KB	~250KB

Because of the very limited computing performance, most of the resource constrained devices, especially class 0 device, cannot handle most of the security approaches [15], notably the mechanism which has heavy computation such as running TLS for transport security.

B. Vast number of devices

The significant number of connected devices appears to create more vulnerabilities [16]. For IT department, it is cumbersome to manage many different types of devices [17] especially when the security mechanism is applied to IoT system. For example, by using username and password to authenticate, the IT department will have to put much effort to maintain the security credentials (e.g. change the password periodically).

C. Lack of security awareness

The lack of security awareness makes a developer may prefer to choose functionality over security when trade-offs must be done [18]. On the other hand, according to the Bitdefender survey study [19] at US, Romania, Germany, Australia, France, and UK, only less than 50% of people from each country that aware of almost all security awareness parameters (e.g. privacy concerns, losing control of smart device, frequency of a software update). Another study from HP Fortify states that 70% of devices use unencrypted network service [20].

III. Attack Scenarios on MQTT Protocol

In this section, we will discuss how an attack can be carried out on the MQTT protocol.

First, we assume that we do not know anything about the victim system that we want to attack (i.e. no prior knowledge of the infrastructure, defense mechanisms, and communication channels). This type of assumption is called black box penetration testing [21]. The attack is begun by performing information gathering that can be accomplished by using Shodan, Masscan, or NMAP [22]. For this paper, Shodan search engine will be utilized.

By inputting string " port:1883 "MQTT" " in search box inside Shodan, we perform searching on MQTT protocol on port 1883, the default MQTT broker port that doesn't use TLS mechanism for security purpose, to find available broker server. The search result provided in figure 2 shows at that moment (April 27, 2017), there were 24998 brokers with default port successfully indexed by Shodan.

Fig. 2. Result of MQTT broker on port 1883 in Shodan

Fig. 3. MQTT connection code in Shodan Page search result

Besides the result shown in Figure 2, there is also MQTT connection code on the right of each broker that is provided in Figure 3. All the brokers that have connection code of "0" are easier to be attacked because this kind of broker does not use any client authentication mechanism thus anonymous publisher or subscriber can connect to this broker freely.

For the first scenario, we can start to subscribe to all topics in that broker (subscribe to #) which may give us confidential data to be analyzed later. This attack scenario is illustrated in Figure 4.

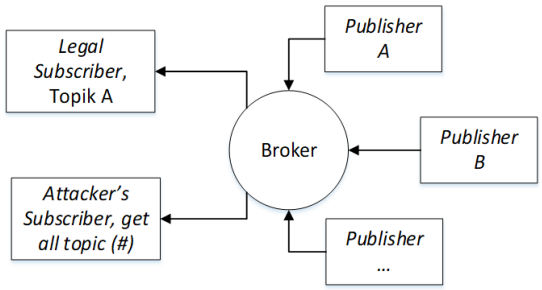

Fig. 4. Attacker can subscribe to all topic message

Another scenario can be initiated by publishing data to the broker who does not have authentication mechanism which is illustrated in Figure 5. Street lamps act as subscriber where the legal publisher can publish a message to control the street lamps. On the other hand, since the broker does not have an authentication mechanism, an attacker can subscribe to the broker to get any message that is used to control the street lamps. By analyzing the control message, the attacker can publish his message to take over the street lights. This kind of scenario can also be used by an attacker to publish spam data so that both broker and subscriber get flooded and may result in denial of service.

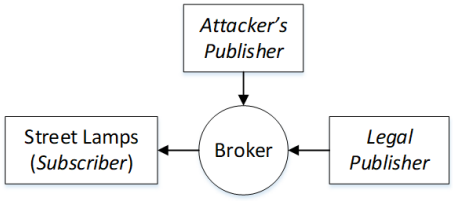

Fig. 5. Attack scenario from attacker's publisher

The first and second scenarios are a generic scenario that can be applied both in the local network and public network. The next scenario that will be discussed has the assumption that the attacker is connected to the same network with IoT system (e.g. at publisher network or broker network).

Using this assumption, the attacker can perform traffic analysis on that network to extract valuable information from data in-transit of MQTT protocol in the form of plain text, such as:

a. IP broker (usually public IP address)
b. Name of topic
c. Data payload
d. Port number of MQTT that IoT system use

To demonstrate this scenario, an Espectro board (based on ESP 8266 board) will act as a publisher and is on the same wireless network as the attacker computer which runs Kali Linux operating system. Meanwhile, subscriber and broker are on the another network. Publisher device publishes to topic "outTopic", with message payload "hello world", while, for this demonstration, the subscriber will subscribe to all topic (#).

The attacker will use Wireshark and Ettercap to perform the attack. An attacker that is in the same network with a publisher can sniff and modify the data in transit thus he can exploit the data privacy, authenticity, and integrity of MQTT packet.

A. Data privacy

Data privacy in MQTT message is absolutely an issue since, by default, MQTT does not provide any data encryption. Whether the broker system uses authentication mechanism or not, the attacker can still sniff the data in transit easily. Figure 6 gives a screenshot of attacker's Wireshark packet capture that shows the MQTT topic and message of the data in-transit from the publisher device earlier.

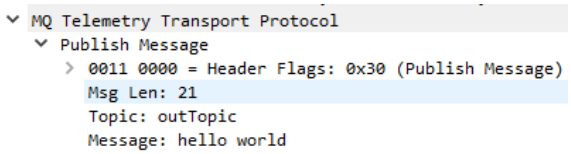

Fig. 6. Published message that captured in Wireshark

B. Authentication

If the broker uses client authentication mechanism by using username and password, the attacker could not act as publisher or subscriber as long as the attacker does not know the username and password (i.e. MQTT connection code will be 5 if we don't provide username-password, or 4 if bad username or password is supplied). In the case of our scenario, the attacker is in the same network with the publisher. Thus the attacker can sniff the traffic on the network while waiting for a "Connect" packet from the publisher is in transit so that the username and password that are used to connect to the broker can be revealed.

During the authentication process, there is a header in the packet known as KeepAlive which indicates how long the IoT device (publisher/subscriber) remains connected to the broker. Therefore, when the KeepAlive time is expired, the device (publisher/subscriber) will resend the "Connect" packet to restart the connection. Figure 7 shows the "Connect" packet from the publisher that has been sniffed by the attacker.

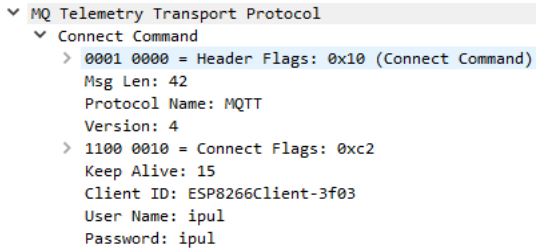

Fig. 7. Result of sniffing the MQTT Connect command packet

C. Data Integrity

Another possible attack is targeting the integrity of data in transit. The attacker who has already known the data packets by sniffing the traffic can modify the data in transit. In this scenario, the attacker wants to change the topic name from "outTopic" to "outTopuc". To do so, the attacker makes a filter file (named owned.filter) which will filter the packet data in transit that has TCP port 1883 and destination address to broker IP. After the packet that matched the filter criteria is identified, it will also search the string "outTopic" and replace it with "outTopuc" as seen in figure 8. Next, Etterfilter application is used to compile "owned.filter" file which will give an output file named "owned.ef".

```
#owned.filter
if (ip.proto == TCP && tcp.dst == 1883 && ip.dst == 'IP Broker' &&
search(DATA.data, "outTopic")) {
  replace("outTopic", "outTopuc");
  msg("payload replaced\n");
}
```

Fig. 8. Filter file to filter MQTT packet

Finally, by using Ettercap application running at the specific interface in which the attacker used to connect to the internet, the attacker uses the compiled filter to modify the packet after successfully performed ARP poisoning to make another network connection going through the attacker computer. This step is given in figure 9.

```
etterfilter owned.filter -o owned.ef
ettercap -T -q -i eth0 -F owned.ef -M ARP /// ///
```

Fig. 9. Command to run ettercap with specific parameter

Figure 10 shows published message topic that has been successfully altered and has been received in subscriber device. Because the subscriber subscribes to all topic, it still receives the message. Furthermore, the attacker can change the message to execute another interesting attack in this protocol. One of the interesting scenarios happens when attacker identifies someone who sends a link to download a firmware update for some devices over MQTT. The attacker can change the link in such way that the victim devices install malicious firmware that transforms them into botnets.

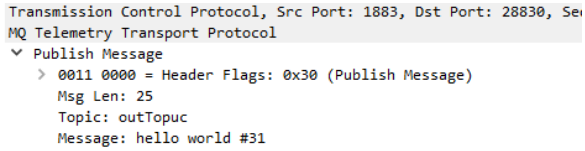

Fig. 10. Result of change in topic name

D. Port Obscurity

The official IANA port number used by MQTT is 1883 for the regular MQTT and 8883 for MQTT using SSL / TLS. However, a broker administrator can configure to use the non-standard port on the system. Unfortunately, if the security mechanism only depends on the MQTT protocol itself, the attacker can still easily observe packets that pass through the network.

For example, the attacker can use Wireshark to sniff the packet and apply data filtering by selecting Edit menu ➔ Find packet … ➔ Type MQTT ➔ String and Packet Byte. This filtering can be done because, in the MQTT, there is a variable header containing the MQTT protocol name that is sent along with the "Connect" packet by the client (publisher or subscriber) to the server (broker). Figure 11 shows MQTT data packet in port 1884 from Wireshark application.

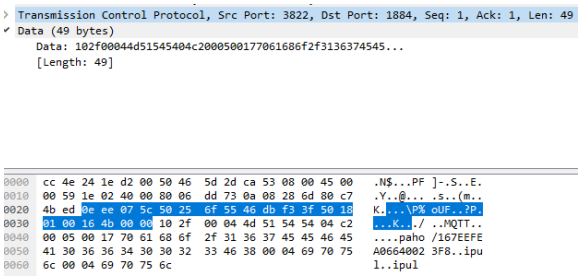

Fig. 11. MQTT packet in port 1884

E. Botnet over MQTT

Botnet over MQTT had been presented during Defcon 24 event, which demonstrated BotMaster sent a command to bots over MQTT protocol [23]. A botnet is a network consisting of many bots--a new type of malware installed on a compromised computer--which then can be controlled by BotMaster [24]. We can obtain a broker using Shodan search engine as we have done before and transform it to become free broker server that connects attacker to victim's device. By using this scenario, the attacker can hide from any investigation because he uses the unsecured broker as an arbiter to communicate with the botnet.

As we can see in figure 12, BotMaster acts as commander to a botnet and uses a certain broker to control many IoT devices (botnet) at once with only one published message in a specific topic. BotMaster can also receive victim status and subscribe to the status of every IoT device (botnet). This scenario is very efficient especially if BotMaster wants to give one command to all botnet at once (e.g. launch a DDoS attack, send a large amount of spam or phishing emails [24]).

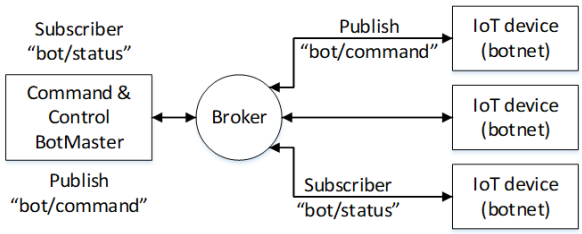

Fig. 12. Botnet command and control scenario using MQTT

IV. Conclusion

MQTT is one of the protocols used in IoT system where several scenarios to attack this protocol has been discussed in this paper. The first scenario takes places in the public network where we can scan the network by using Shodan search engine to search MQTT public server to make denial of service attack to devices (clients) connected to that broker or get/send incorrect data to its clients. This public broker can become a good candidate to control the botnet because of the nature of MQTT publish and subscribe. Then, from the local network, an attacker can sniff and modify packet data from the network to attack data privacy, data integrity, and MQTT authentication mechanism. Moreover, using nonstandard port (port obscurity) does not improve the security of MQTT at all.

For mitigation purpose, a security mechanism for MQTT protocol must be implemented such as TLS, which is a good choice if the IoT devices that are used is an unconstrained device. Besides using TLS, Singh et.al. [25] have proposed another security mechanism based on ECC which focuses on data confidentiality with less resource requirement compared to TLS. Furthermore, Mektoubi et.al [26] have performed a study comparing RSA and ECC to protect the data confidentiality and provide good non-repudiation. In the case of constrained devices, Niruntasukrat et.al. [3] have tried to make a security mechanism that focused on authentication and authorization of the devices to broker while Katsikeas [27] uses AES encryption that focuses on confidentiality and message authenticity. Security mechanism of MQTT protocol, especially for resource constrained device still need development because each research that has been done still have certain focus which not yet integrated.

References

[1] D. Evans, " The Internet of things: how the next evolution of the Internet is changing everything," Cisco Internet Business Solution Group White Paper, April 2011.

[2] Gartner. (2017, February 7). Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. Available: http://www.gartner.com/newsroom/id/3598917.

[3] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul and A. Panya, "Authorization mechanism for MQTT-based Internet of Things," 2016 IEEE International Conference on Communications Workshops (ICC), pp. 290-295, 2016.

[4] D. H. Mun, M. L. Dinh and Y. W. Kwon, "An Assessment of Internet of Things Protocols for Resource-Constrained Applications," 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), pp. 555-560, 2016.

[5] N. De Caro, W. Colitti, K. Steenhaut, G. Mangino and G. Reali, "Comparison of two lightweight protocols for smartphone-based sensing," 2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT), 2013, pp. 1-6.

[6] Banks, A. and Gupta, R, "MQTT version 3.1.1*,"* OASIS Standard, 2014.

[7] Prada, A., & dkk, "Communication with resource-constrained devices through MQTT for control education," 11th IFAC Symposium on Advances in Control Education ACE, pp 150-155, Bratislava, Slovakia, 2016.

[8] ISACA Volunteer Member, "Cybersecurity Fundamentals Study Guide," ISACA, 2015.

[9] M. M. Hossain, M. Fotouhi and R. Hasan, "Towards an Analysis of Security Issues, Challenges, and Open Problems in the Internet of Things," *2015 IEEE World Congress on Services*, New York City, NY, 2015, pp. 21-28.

[10] M. Iqbal and M. Bayoumi, "Secure End-to-End key establishment protocol for resource-constrained healthcare sensors in the context of IoT," International Conference on High Performance Computing & Simulation (HPCS), pp. 523-530, 2016.

[11] W. Stallings, "Cryptography and Network Security Principles and Practice 7th Edition," Pearson, England, 2017.

[12] S. Alasmari and M. Anwar, "Security & Privacy Challenges in IoT-based Health Cloud," International Conference on Computational Science and Computational Intelligence, pp. 198-201, 2016.

[13] Caltum, E. and Segal, O, "Exploitation of IoT devices for Launching Mass-Scale Attack Campaigns," Akamai Threat Research, October 2016.

[14] C. Bormann, M. Ersue, and A. Keranen, " RFC 7228 Terminology for Constrained-Node Networks," IETF, May 2014.

[15] J. King and A. Ismail,"Distributed Security Mechanism for Resource Constrained IoT Device," Informatica 40, pp 133-143, 2016.

[16] Anonym. (2016, January 15). *Iot Security Awareness*. InfoSec Institute [Online]. Available: http://resources.infosecinstitute.com/iot-security-awareness/

[17] Anonym, "IoT Security: Protecting The Networked Society," Ericsson White Paper, February 2017.

[18] Ernst and Young, "Mobile device security: Understanding vulnerabilities and managing risks," Insight on governance, risk and compliance, January 2012.

[19] Bitdefender, "Security Awareness in the Age of Internet of Things," A Bitdefender Study White Paper, 2016.

[20] Anonym, "Internet of Things research study," Hewlett Packard Enterprise, 2015.

[21] F. Alisherov, and F. Sattarova, " Methodology for Penetration Testing", International journal of Grid and Distributed Computing, Vol 2 No 2, June 2009.

[22] L. Markowsky and G. Markowsky, "Scanning for Vulnerable Devices in the Internet of Things," The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing System: Technology and Applications, September 2015.

[23] L. Lundgren, "Light Weight Protocol Serious Equipment Critical Implications", Defcon 24, 2016.

[24] H. R. Zeidanloo and A. A. Manaf, "Botnet Command and Control Mechanisms," 2009 Second International Conference on Computer and Electrical Engineering, Dubai, 2009, pp. 564-568.

[25] M. Singh, M. A. Rajan, V. L. Shivraj and P. Balamuralidhar, "Secure MQTT for Internet of Things (IoT)," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, 2015, pp. 746-751.

[26] A. Mektoubi, H. L. Hassani, H. Belhadaoui, M. Rifi and A. Zakari, "New approach for securing communication over MQTT protocol A comparison between RSA and Elliptic Curve," 2016 Third International Conference on Systems of Collaboration (SysCo), Casablanca, 2016, pp. 1-6.

[27] S.Katsikeas, "A lightweight and secure MQTT implementation for wireless sensor node", 2016 Technical University of Crete.