A Comparative Analysis on Security of MQTT Brokers

Jaidip Kotak, Anal Shah, Ayushi Shah, Palak Rajdev

jaidipkotak@gmail.com, analshah1705@gmail.com, ayushi71195@gmail.com, palakhrajdev@gmail.com
Gujarat Forensic Sciences University, India

Abstract

In the era of rapid revolution of Internet of Things (IoT), the security of IoT devices as well as its application protocols has become significant. MQTT (MQ Telemetry Transport) is a light weight protocol used for communication between IoT devices. It is being extensively used in a low bandwidth environment (e.g. smart cities, home automation, etc.). There are multiple vendors for clients and brokers which are used for publishing/subscribing to topics and which act as an intermediate server respectively. As different brokers are developed by different developers, it may have different implementation flaws which impact the security of the communication between IoT devices in various ways. In this paper, we analyze various brokers available on internet from security point of view by performing DoS attack and information gathering techniques on the broker, compare the outcomes and try to find out the least vulnerable broker that can be used for secure communication between IoT devices.

Keywords

IoT, Internet of Things, Security, Smart Cities, Critical infrastructure, Home Automation, MQTT.

1 Introduction

The term, Internet of Things, a system of interconnected devices, was first proposed by Kevin Ashton in 1999. It has revolutionized the current computing networks with all the physical objects surrounding us to be uniquely identifiable and connected to each other [1].

It is expected that the use of IoT in home and business applications will grow significantly, to contribute to the quality of life and to grow the world's economy. For example, smart-homes will allow their residents to automatically open their garage when reaching home, control the temperature of systems, fridge and other appliances. In order to realize this potential growth, emerging technologies and inventions need to grow proportionally to match market demands and customer needs. Also, devices need to be developed to fit customer requirements so that they are available anywhere and anytime. Apart from it, new protocols are required for communication compatibility between diverse things (living things, vehicles, phones, appliances, goods, etc.) [2]. There are many application level protocols used in IoT devices. Some 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]. We discuss here the most widely used protocol MQTT.

IoT Systems have limited resources and that is why Message Queuing Telemetry Transport (MQTT) is widely used as it has: less bandwidth requirement, is lightweight, open and straightforward to be implemented [4]. In this protocol, sensors publish their data at regular intervals to their related topics. All the devices that have been registered to a particular topic will receive message from the broker each time that topic receives new message [5].

The broker is the main entity of any publish/subscribe protocol. Depending on the implementation, a broker can handle up to thousands of parallelly connected MQTT clients. The broker is responsible for sending the message to the subscribed clients, filtering the messages and classifying the subscribers based on the topics that they have subscribed. It also holds the sessions of all persisted clients, including subscriptions and missed messages. The other duty of the broker is the authentication and authorization of clients. That is why it is important that the broker should be easy to monitor, highly scalable, integrable into back-end systems, and failure-resistant.

As it is expected of the broker to perform various tasks, it is difficult for a single broker to implement all the features necessary to securely communicate between IoT devices. That is why in MQTT there are many brokers available on the Internet that have different levels of security and react differently when various types of publish/subscribe messages are sent.

In this paper, we discuss various brokers available on the Internet, try to find out the vulnerabilities of each broker and compare the results. We have analyzed the behaviour of the MQTT brokers with the help of the same publishers/subscriber. We have used mosquito_sub and mosquito_pub as the clients for our analysis. MQTT protocol version used in our experiment was 3.1 and 3.1.1 and QoS 0.

2 Related Work

2.1 MQTT protocol:

MQTT is a consistent publish/subscribe protocol released by IBM in 1999 and as it’s widely used in major IoT devices that are constrained. It has features as well as flaws. Generally, MQTT protocol is used in respective areas where communication links supply low throughput like home automation, SCADA, smart cities, monitoring, etc. [6].

MQTT keeps bandwidth at an absolute minimum and deals with unreliable networks. MQTT stands on top of TCP, thereby empowering error and flow control for a single packet to the lower layers of the protocol pyramid. As MQTT works on publisher/subscriber paradigm, information is classified through "topic" hierarchies with path and nodes followed by subscribers. MQTT also provides a QoS level (quality of services) that deals with two clients (publisher/subscriber) for assurance of message delivery [7]. While working with protocol, security is an essential aspect with the focus on authentication, access control, data integrity and confidentiality. So in architecture where MQTT protocol is being used, security is significantly dependent on the development of broker's mechanism. Various brokers have their own pros and cons related to supporting of TLS with client certificate-based authentication, authorization using a database, horizontal and vertical scalability (clustering, multithreading and other) etc. While comparing various brokers we have measured possible attack vectors, reconnaissance for information leaks/banner grabbing, performance and authentication/authorization complexity.

2.2 DOS (Denial of Service) and DDOS (Distributed Denial of Service) attack:

DoS attack is caused by a single attacker which makes genuine users or organizations not able to achieve the services of resources they would normally presume to gain. Whereas DDoS is caused by multiple attackers restricting the legitimate users or organizations to gain access to services. There are already twice as many devices connected to the internet than there are people on our planet. The concept of IoT security is no newer than hardware security, machine to machine security, cloud security, connected devices security and embedded devices security, as the possible security risk resides in firmware of devices or Back-end mechanism of it. Application-level DDoS attacks caused using compromised IoT devices are emerging as a critical problem. DOS can take place via sniffing subscriber's unique ID towards "topic" of brokers, authentication/authorization failure, Unicode support (non UTF8). It can possibly cause the server to fall down or create an anonymous connection to the attacker. Also, DDoS campaigns are carried out by botnets which exploits an army of infected computers/devices to dazzle a target web service or Internet infrastructure element with malicious traffic [8]. In this coming trend, large number of devices raises important issues for vendors regarding the quality of service (QOS), which can only be mitigated by detecting and defending the systems from the Denial of Service attacks. DOS can be carried out by compromising a device and exploiting via weakness on Over the air management (OAM) and traffic [9]. In the MQTT context, brokers are the heart of any IoT infrastructure as well as prime focus of attackers for DDOS especially in the use of open source brokers. There are some basic counter measures which can defeat DDOS attacks like client authentication, connection capacity, throttling (level of QoS used), Topic space partitioning.

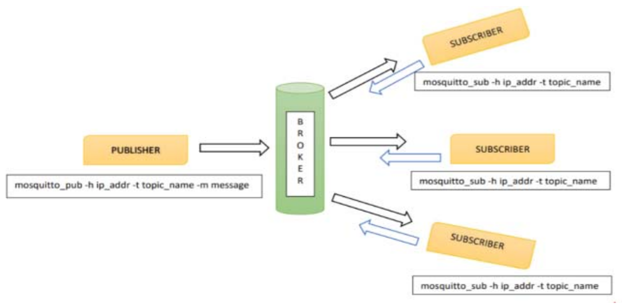
2.3 Reconnaissance:

As we know that anything connected to internet is open for threats. Since IoT devices are used and operated by humans, any crawler may want to gain gratuitous access of it. As we know MQTT uses brokers and clients for communication, end nodes should be secure and reliable. For example, MQTT-PWN is considered as a one-stop-shop for IoT Broker penetration-testing and security assessment operations, as it combines enumeration and supportive functions. Also, monitoring of systems is extremely important for production environments in smart cities or for organizations. When deploying a MQTT broker or a cluster of MQTT brokers, the operation team needs to know the health of the MQTT servers and uses monitoring data to circumvent foreseeable future problems. From the past few years developers use so called SYS-Topics for not only monitoring but also for debugging purpose. These are the default special meta-data topics used for acquiring information about broker itself and its connected client's sessions. All the SYS topic start with $SYS and readable by clients but publishing to this topic is prohibited. This is strictly intended for debugging purpose for developers as they can quickly monitor the current state of the broker, calculate and verify metrics - like message amplification rate or network metrics. It's still important to hide such deployment information. SYS-Topics expose information like broker software used and its version number to any subscriber. These may be valuable information for attackers but often don't provide too much value to actual legitimate subscribers. So, it is recommended that SYS topic should be disabled by default in any brokers.

3 Need for Security

For years, consumers and businesses alike have been obsessed with securing computers and smart phones. But in reality, those devices are less at risk than more simplistic connected items [10]. It is because these smart devices have been around for a long time and so various security measures are already taken. Consumers and businesses are aware of the importance of security in PCs and smart phones. But there is a lack of awareness when it comes to securing simplistic things like security cameras placed inside smart home or even a simple thermostat. If attackers are able to get inside these IoT devices, they can get personal information of the people living in home and thus invade privacy.

Most IoT devices can serve as entry points to home or corporate network and thus provide a risk of data breach. 80% of the world's data is in private servers and if the attackers are able to gain entry to those servers, the damage can be monumental. While the attacks cannot be stopped entirely, proactive measures can be taken to mitigate threats to network security and IT systems. MQTT protocol should include security aspects such as confidentiality, availability, integrity and privacy of data as IoT security will become increasingly important in preventing business and personal catastrophes.[11]



```
mosquitto_sub -h ip_addr -t topic_name
mosquitto_sub -h ip_addr -t topic_name
mosquitto_pub -h ip_addr -t topic_name -m message
mosquitto_sub -h ip_addr -t topic_name
```

Figure 1: MQTT publish/subscribe framework

4 Methodology

In MQTT protocol, publisher publishing messages and communicating with users or clients subscribing to that topics is ordinarily considered as a publish/subscribe model or methodology. In this model the published message to particular topic can be considered as messaged subject and subscriber can then subscribe to these topics for acquiring messages. Here we have used seven (7) brokers and clients in different virtual machines so as to note the behaviour of the brokers with same clients. Broker controls the distribution of substance and is mainly responsible for receiving all messages from publisher, separate out them; decide who is concerned in it and then sending the messages to all subscribed clients [12]. And client could be a publisher or subscriber, who forever establishes the network connection to the server (broker). As shown in figure 1 we first created topic_name using server's (broker's) IP address and published messages on it. Now client will subscribe to same topic_name to server (broker) and after that broker publishes messages to clients who have subscribed to relevant topics. Here for describing the topic name we used -t, for mentioning broker's IP address we used -h and -m for sending the message. We have used the default port 1883 (unencrypted) for our experiments.

Some of the distinguishing features of the brokers we analysed are described in Table 1.

Table 1: Features of MQTT

No.	Protocol	Details	Features
1	Mosca	-Support for MQTT 3.1 and 3.1.1 -Node.js MQTT broker used as standalone and embedded in other node.js application	-Support QoS0 and QoS1 -Usable inside any other Node.js app. TCP support and Web-socket support
2	Hbmqtt	-Support for MQTT 3.1.1 -Built on top of TCP asyncio framework -Python 3.5.0 fully supported -Used for running autonomous MQTT broker.	-Support all QoS -Client auto reconnect on network lost -Authentication through passwd file -SSL support over TCP and web socket
3	VerneMQ	-Support for MQTT 3.1 and 3.1.1 -Written in erlang -Distributed message broker -Uses master-less clustering technology -Low entry and exit risk	-Support all QoS. File-based Authentication and Authorization -HTTP Administration API -Web socket support and TLS encryption available
4	Apache ActiveMQ	-Supports JMS 1.1 and J2EE 1.4 -It also supports cross languages clients and protocols from C, C++, Java, C#, Ruby, Perl, Python, PHP	-Support all QoS -Web socket -GUI support -TLS(SSL) encryption
5	HiveMQ	-Support MQTT 3.1,3.1.1,3.5 versions -Java based SDK -Extension Hot Reload. -Pre-built Extensions	-Support all QoS -Automatic elastic and linear scalability at runtime -Web socket and TLS encryption. -Proxy protocol for advance load balancer integration
6	RabbitMQ	-Support for MQTT 3.1.1 -Can be deployed with bosh, chef, docker and Puppet -Develop cross language messaging with programming languages such as Java, .NET, Ruby,	-Support all QoS -Deploy as cluster for high availability and throughput -TLS encryption -Support all messaging protocols
7	Eclipse Mosquitto	-Support MQTT protocol versions 3.1 and 3.1.1	-Support all QoS, TLS encryption port 8883 -Dynamic topics

Table 2: Comparison of Brokers based on DOS attack

Type of DOS	Mosca	Hbmqtt	VerneMQ	Apache ActiveMQ	HiveMQ	RabbitMQ	Eclipse Mosquitto
Type 1	✓		✓	✓	✓	✓	✓
Type 2		✓					

Table 3: Comparison of Brokers based on $SYS Vulnerability

	Mosca	Hbmqtt	VerneMQ	Apache ActiveMQ	HiveMQ	RabbitMQ	Eclipse Mosquitto
%SYS(metadata found)	✓	✓					✓

5 Analysis

Based on the setup shown in figure 1 we tested sending different messages to the brokers in order to analyse the behaviour. We observed the different outcomes and have provided the result in the tabular format.

5.1 Denial of Service Attack

When a subscriber is already connected to the broker, by sniffing the packets with the help of wireshark, an attacker can find out the id of the subscriber used to connect to the broker. If the attacker sends subscriber message with the same id to the broker it leads to denial of service attack for victim subscriber.

All the brokers we have explored are prone to this attack but the behaviour is different. We have noted two kinds of behaviour when the above attack is performed.

A subscriber connects to the broker, then the attacker connects to the broker with the same id and one of the below mentioned behaviour is noted:

Type 1: The victim subscriber gets disconnected and the published messages with that topic are received by the attacker.

Type 2: The broker server hangs and no further action can be taken.

Table 2 shows observations in which we show different brokers behaving differently according to the points stated above. Note: for Hbmqtt broker we have used its own clients as mosquito clients were not compatible with it.

5.2 Reconnaissance Vulnerability

The metadata of the broker can be seen by subscribing to the topics within the $SYS hierarchy. It is usually used for debugging purposes but many brokers provide this information to all the clients who subscribe to the topic of '$SYS/#'.

This can be considered as vulnerability as it can expose all types of information about the broker to the attacker which can increase the chances of any attack in future. The brokers which are prone to such vulnerability (by default) are listed in Table 3.

6 Conclusion and Future Works

One of the protocols used in IoT devices is MQTT and few attack scenarios on the brokers of this protocol are discussed in this paper. We can try connecting to the broker using the same id (sniffed from wireshark) as one of the clients and try to find out the messages that are being sent. Also, the meta-data about the broker can be collected if the reconnaissance vulnerability is present in the broker. After performing such attacks, we provided a comparative analysis of the various brokers found for the MQTT protocol in order to observe the different characteristics they possess when such special messages are sent.

As part of our future work, we continue to perform other forms of attacks on the brokers to find out the least vulnerable broker that can be used for secure communication between IoT devices in the smart cities.

References

[1] A. Jain, B. Sharma, & P. Gupta (2016). Internet of things: Architecture, security goals, and challenges — a survey. International Journal of Innovative Research in Science and Engineering, vol. 2, no. 4, pp. 154– 163

[2] Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari & M. Ayyash. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376

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

[4] Banks, A. and Gupta, R, “MQTT version 3.1.1,” OASIS Standard, 2014

[5] Syaiful Andy, Budi Rahardjo, Bagus Hanindhito. Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System

[6] Perrone, Giovanni & Vecchio, Massimo & Pecori, Riccardo & Giaffreda, Raffaele. (2017). The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices. 246-253. 10.5220/0006287302460253.

[7] Soni, Dipa & Makwana, Ashwin. (2017). A Survey on MQTT: A Protocol of Internet of Things (IOT).

[8] Ketan Bhardwaj, Joaquin Chung Miranda, Ada Gavrilovska. Towards IoT-DDoS Prevention Using Edge Computing

[9] Sunita Godara, Mr. Narendra Kumar. A Survey of Security Attacks in M2M Communication.

[10] TechTarget “The importance of securing the internet of things”

[11] Anthraper, Joseph Jose and Kotak, Jaidip, Security, Privacy and Forensic Concern of Mqtt Protocol (March 19, 2019). In Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM-2019), February 26 - 28, 2019

[12] OASIS MQTT version 3.1.1 Committee Specification Draft.