MQTT Protocol Security Using Payload Encryption & Digital Signature Technique

Umesh V. Nikam **Ms. Jaya N. Tatte**

Abstract- The Internet of things (IoT) is the system of interconnection of computing objects with the Internet network using existing technologies and communication protocols without requiring human to human or human to computer interaction. Use of IoT is growing rapidly which raises many new serious issues related to security. Many recent reports on cyber security have highlighted IoT vulnerability and the risk with deployment of intelligent networks. [1] The objects connected to the network can create a lot of attacks and thus can present a danger for integrity of data. So as to avoid these possible attacks, security approaches must take place to ensure a set of criteria, like resistance to attack, data authenticity, user privacy and access control [2]. As IoT is having very limited capacity of memory, bandwidth and energy a new type of protocols and ideas are developed in order to improve quality of service for this type of network. One of the protocols is MQTT (Message queue telemetry transport protocol). This protocol is created by IBM which use publish and subscribe pattern. This protocol requires very small bandwidth. This work proposes a new approach for secured communication which is based on MQTT protocol. In this , we use standard techniques of cryptography such as digital signature & payload encryption to secure communications, in an IoT network.

Keywords- broker, digital signature, payload encryption, publisher, Security, subscriber

I. INTRODUCTION

MQTT is used in IoT as a client server publish/subscribe messaging transport protocol MQTT is light weight, simple, open & can be implemented easily. It is a binary protocol. MQTT has a minimal packet overhead. So in comparison to HTTP protocol it excels more while transferring data over wire [1].

One more important thing is MQTT can be easily implemented on client side. This best suits for constrained devices like IoT which are having limited resources.

II. History

Andy Stanford-Clark & Arlen nipper invented MQTT in 1999. It was their use case to create a protocol for minimum battery loss & minimum bandwidth connecting oil pipeline over to the connection of satellite. Some of the expected goals in future protocol should have:

- Simple for implementation
- Quality of Service Data Delivery
- Bandwidth efficient & lightweight
- Data agnostic
- Continuous awareness for session

The publish and subscribe pattern

Publish/subscribe is an alternative to client server, where a client directly communicates with server. But it decouples a message from another client, who is receiving a message called a subscriber. Publisher & subscriber have no idea about each other but broker knows to both of them. A broker filters all the messages & distributes them accordingly. [4]

Fig: 3.1 MQTT Publish and subscribe

Client

A client may be a publisher or subscriber. A MQTT client can be any device that has a MQTT library running and is connecting to an MQTT broker over a network.

Broker

A broker is a heart of publish subscribe protocol. It can handle up to thousands of concurrently connected clients. Primary responsibility of broker is to receive all messages, filter them and decide who is interested in it and then sending the messages to all the subscribed clients. Broker also holds the session for all the persisted clients including subscription and missed message.

Along with this authentication & authorization is also the responsibility of a broker. Many times it is extensible, which makes it possible to easily integrate custom authentication, authorization into the backend system. In all broker is a central hub through which every message needs to be passed.

III. MQTT Connection

It is based on top of TCP/IP. So it is necessary both client and broker need to have a TCP/IP stack.

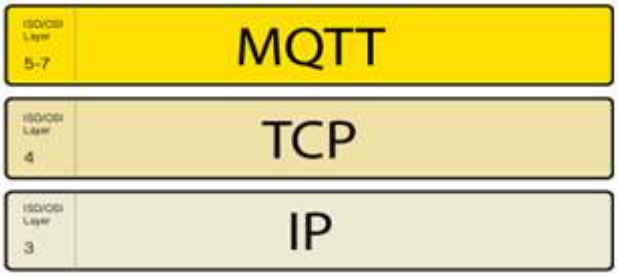

Fig: 4.1 TCP/IP stack

MQTT connection is always between client & broker. A client is never directly connected to another client.
A client initiates a connection by sending CONNECT message to broker.

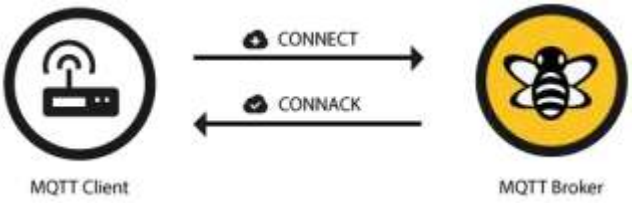

Fig: 4.2 MQTT Client & Broker

A broker can close connection if CONNECT message is damaged or it takes too long time from opening a network socket to sending it. A legitimate client sends a connect message with following parameters.

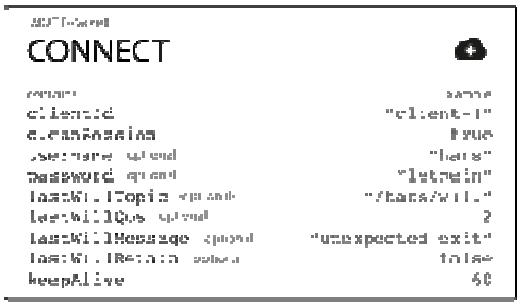

Fig: 4.3 MQTT Packet

Id

It is a unique identity number for every MQTT client connecting to broker. It is always unique per broker. Broker use client id for identification of client and its current state.

Clean Session

Clean session is used by broker to indicate whether a client wants to form a persistent session or not.

A persistent session means, the broker will store all subscriptions for the client and also all missed messages, when subscribing with quality of services. If clean session flag is set to true, the broker will not store any information for the client and will also remove all information for a previous persistent session.

Username and Password

For authenticating & authorization of the client MQTT allows sending a username and password [2].

Will Message

It is about the last will and testament feature of MQTT. When a client is disconnected ungracefully it notify to other clients. A connecting client then provide his will in form of an MQTT message & topic in the CONNECT message. If the client is disconnected ungracefully, on behalf of the client a broker sends this message.

Keep Alive

A Client sends regular PING Request messages to the broker and as a response broker also sends PING Response message. This mechanism allows both sides to check if the other one is still alive and reachable.

Session Present flag

This flag indicates whether a broker already has a persistent session of client from previous interaction. If a client connects & it sets clean session to true, this flag is always false, because there is no session available. If it sets to false, the flag is depending on, if session information is available for the client id. If session information exist, the flag is true otherwise false.

Connect acknowledge flag

It informs the client, whether the connection attempt was successful or not and otherwise what the issue is.

Following table shows all the return codes at a glance.

Return Code	Code Response
0	Connection is accepted
1	Connection is refused, unacceptable protocol version
2	Connection is refused, Identifier is rejected
3	Connection is refused, server is unavailable
4	Connection is refused, bad user name or password
5	Connection is refused, not authorized

IV. Proposed MQTT Payload encryption Technique:

In this technique application specific data is encrypted on application level. It can encrypt application specific as well as untrusted data also. It encrypts payload of the message only whereas metadata like MQTT topic stays intact. [5]

As MQTT payload is always binary so no need to encode an encrypted message to textual representation like base 64. This saves the additional bandwidth.

Fig: 5.1 MQTT publish packet

Whole MQTT publish metadata stays intact, only a payload is encrypted. Because of this no other technique is needed on the broker side to decrypt the data to route MQTT messages to the subscriber. Generally MQTT payload encryption is used only for MQTT publish packets.

Encryption scenario (End to end (E2E) Encryption)

It is ensured that an encrypted data stays encrypted all the time. MQTT broker use unencrypted packet metadata for routing and quality of service handling. Only the trusted clients have a key to decrypt the data. Broker has no way to look into the encrypted data.

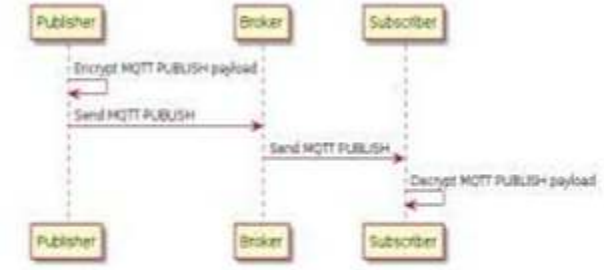

Fig: 6.1 Encryption Scenario client to broker

An attacker can get access to MQTT packet but he can't decrypt it without a key. E2E can be applied to any topic by any MQTT client & this encryption is broker

implementation independent. If authentication and authorization mechanisms are not possible or a public broker is used, you can protect your application data from attacker and MQTT clients.

Client to broker

This approach makes sure the payload is encrypted in the communication between a client and broker. Before a message is distributed a broker can decrypt it. So all subscriber receives original, unencrypted message. It can be used in case a you can't use TLS & want to protect a data from eavesdroppers on publishing side. Trusted subscriber are connected via a secure channel i.e TLS and so they can receive a data in plain text.

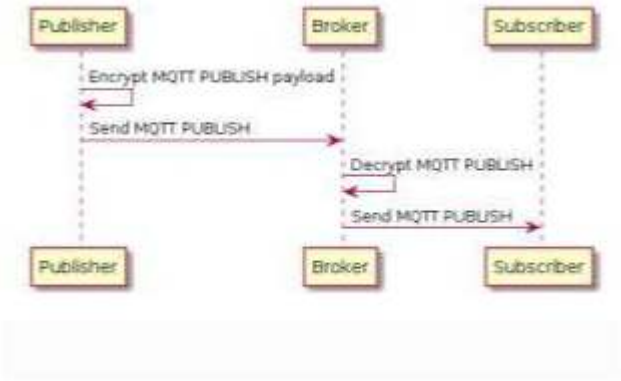

Fig: 6.1 Encryption Scenario broker to client

V.Mechanism of encryption

There are two encryption mechanisms:

Asymmetric encryption (Public or Private Key encryption)

If there are few trusted subscribers and many publishers, asymmetric approach is perfect. In asymmetric encryption, everyone is allowed to encrypt a message with that key but only trusted clients can decrypt the message.

In this technique every MQTT topic gets its own public and private key pair.

Symmetric encryption

This encryption tends to be easier to implement than an asymmetric encryption. It is recommended that every MQTT topic must get its own key, but if a key for a topic is lost, a message on another topic cannot be decrypted by attacker. [5]

VI. Data Integrity by digital signature

In the MQTT context, PUBLISH packets can contain a digital signature. This calculated stamp is typically added to the payload (e.g. at the beginning of the payload). The receiver of the packet can verify the data integrity by recalculating/validating the stamp and so the client has the confidence that the message was not altered by some malicious third party [5].

Typically the following data is used as part of the stamp:

- The MQTT PUBLISH Payload
- The Topic of the MQTT PUBLISH packet

Fig: 6.1 MQTT publish packet

VII. Mechanisms for creating stamps using digital signatures with MQTT

Digital Signatures uses public key cryptography. First a message is signed by a sender with its private key and the receiver validates the stamp with the public key of the sending client. If an attacker did not get a private key, it is not possible to fake them as only private key can create a signature.

Adding a data integrity checks with stamps is a good technique for encrypting messages. Even if the message gets decrypted, the integrity check will still fail if the message was altered. It provides an additional layer of security.

VIII. Conclusion

This paper focuses on the improvement of Communication security in the field of Internet of things (IoT). Based on protocol (MQTT), we proposed a technique for payload encryption and digital signature. Despite the effort, we would like to implement this technique in the simulated environment and obtain the result with encryption algorithm like AES and comparison of AES with other encryption algorithm.

REFERENCES

1. L. Atzori, A. Iera, G. Morabito, The internet of things: a survey, Comput. Netw. 54 (15) (2010) 2787– 2805, doi: 10.1016/j.comnet.2010.05.010

2. D. Miorandi, S. Sicari, F.D. Pellegrini, I. Chlamtac, Internet of things: vision, applications and research challenges, Ad Hoc Netw. 10 (7) (2012) 1497–1516, doi1016/j.adhoc.2012.02.016.

3. R.H. Weber, Internet of things - new security and privacy challenges, Comput. Law Secur. Rev. 26 (1) (2010) 23–30, doi: 10.1016/j.clsr.20 09.11.0

4. HIVEMQ https://www.hivemq.com/mqtt-security

5. R.L. Rivest, A. Shamir, and L. Adleman "A method of obtaining Digital Signatures and Public-Key Cryptosystems"