Overview

MQTT protocol is heavily used for IOT operations. A number of supported software packages are available that implemented MQTT protocol. In this research, we conducted an empirical analysis of the MQTT message broker mosquitto which are deployed for IoT operations on the Internet. The research provides insights into the exposed mosquitto message brokers in real world and security issues associated with them.

MQTT Protocol

Message Queuing Telemetry Transport (MQTT) protocol is considered as a lightweight messaging protocol that is based on the mechanism of publish and subscribe operations for exchanging data between client and server. MQTT is heavily used for the “machine-to-machine” or “Internet of Things” operations across the Internet.

MQTT is based on the client server relationship. MQTT server is called as broker and the client is a connected IOT device. The components of MQTT are discussed below:
MQTT basically uses publish/subscribe mechanism in contrast to request/response mechanism used by HTTP. MQTT publish/subscribe is event driven process that communicates with the centralized entity named as broker which is a server that dispatches messages among senders and intended receivers. Client includes a topic into the message before publishing it to the broker. The topic is the routing information for the broker. If the client wants to receive the messages from the broker, then it has to subscribe the topic. Clients do not require to share the identity as all the communication occurs by setting references to the topic listed by the broker.

**Packets Dissection: MQTT Session**

In this section, we discuss how the MQTT protocol communication to an unsecured message broker configured without authentication (sending null value for username and password). An MQTT session is divided into four stages: connection, authentication, communication and termination. A client initiates a TCP/IP connection to the MQTT message broker by using a configured TCP port. TCP port 1883 is used for unencrypted communication where as TCP port 8883 used for encrypted communication over SSL/TLS. Server can reuse the earlier session if the client identity is known. The complete MQTT session communication related to packet processing between client and mosquitto message broker is discussed below:

**Phase 1: MQTT Connect**: The very first request issued by the client is connect as shown below. The connect request can be sent with or without credentials (combination of username and password). It depends on the configuration of the message broker how exactly the request is accepted. The highlighted payloads show that connect request is sent with username and password value set to null.
Phase 2: MQTT Connect ACK: Once the message broker receives the request it verifies the connect request with the configured parameters on its end. The highlighted request below shows that the message broker accepted the connect request without any credentials. It means that message broker is ready to publish the topics without authentication.

Phase 3: MQTT Subscribe: After the client receives a response from the message broker, it sends request to subscribe to the topic “$SYS/#” as shown in highlighted below. More details about $SYS topics can be found here: https://github.com/mqtt/mqtt.github.io/wiki/SYS-Topics
Phase 4: MQTT Subscribe ACK: The message broker acknowledge the subscribe request as shown in highlighted below.

Phase 5: MQTT Publish Message: The message broker publishes the different $SYS topics based on the subscribe request. The highlighted parameters below shows that the different SYS topics such as "$SYS/broker/clients/active", "$SYS/broker/clients/connected" and "$SYS/broker/load/messages/1min" etc. are received. Only few $SYS topics messages have been shown here.
**Phase 6: MQTT Disconnect:** Once the client receives the published messages, it send a disconnect request as shown in the highlighted below.

The six phases basically highlights MQTT protocol communication in which client and server sent subscribe and publish messages respectively.

**Empirical Analysis**

WootCloud conducted a detailed empirical analysis of the deployed MQTT broker named mosquitto in the wild. The analysis aimed at discovering the MQTT brokers that can be accessed in an unauthenticated manner on the Internet and at the same time provides an opportunity to the remote users to retrieve internal details about the deployment including command execution.
The research uncovers the following;

- The security posture of the deployed mosquitto message broker using MQTT protocol
- The deployed versions of the mosquitto message broker
- The geographical distribution of the deployed mosquitto message brokers
- The security issues associated with the discovered mosquitto message brokers

Mosquitto message broker implements a lightweight version of the software that implements MQTT protocol for low powered devices including servers with full processing power. Mosquitto message broker is accessed remotely via service running on TCP port 1883. Initiating full blown TCP and SYN scans on TCP port 1883 are used to fingerprint the state of configured mosquitto message broker. WootCloud authored a script that fingerprints the mosquitto message broker on a large scale on the Internet.

**Figure 1** shows the output of the script after successful access to remote mosquitto message brokers without authentication.
Figure 1: Script in Execution Phase to Fingerprint the Version of Mosquitto message broker

In addition, Figure 2 reflects the type of information that can be received from the remote mosquitto message brokers. The topics list shows how the data can be received and sent to remote service.
As per the documentation [https://mosquitto.org/man/mosquitto-8.html](https://mosquitto.org/man/mosquitto-8.html), clients can find information about the broker by subscribing to topics in the $SYS hierarchy as follows. Topics marked as static are only sent once per client on subscription. All other topics are updated every sys_interval seconds. If sys_interval is 0, then updates are not sent. Figure shows our script subscribed to MQTT $SYS topics listed by the remote mosquitto message brokers running in an unauthenticated manner.

Figure 2: $SYS topics are retrieved from the remote unauthenticated mosquitto message brokers

For more details about the MQTT $SYS topic, refer: [https://github.com/mqtt/mqtt.github.io/wiki/SYS-Topics](https://github.com/mqtt/mqtt.github.io/wiki/SYS-Topics)

Results
WootCloud conducted a controlled mass scanning exercise aimed at detecting exposed mosquitto message brokers on the Internet. We detected close to close 26000+ instances of mosquitto message brokers running in an unauthenticated manner. It means any remote or unauthorized can access these mosquitto message brokers without authentication thereby resulting in compromising the communication of the associated IoT devices.

**Figure 3** shows the top 10 countries that were found to be running the mosquitto message broker without authentication.

![Top 10 Countries - Unauthenticated MQTT (Mosquitto) Access](image)

**Figure 3: Top 10 Countries with Mosquitto Broker - Unauthenticated Access**

**Figure 4** shows the most deployed software versions of mosquitto message brokers in the wild.
Security Implications

A number of security implications are discussed below:

- The analysis highlights that insecure versions of the mosquitto message brokers deployed on the Internet. As per the fingerprinted versions shown above, a number of inherent security vulnerabilities exist in the related software version. More details can be found here: https://mosquitto.org/security/

- A number of mosquitto message brokers are deployed without authentication which makes it susceptible to remote exploitation and compromise. The attackers can fingerprint and conduct stealth tests to validate the authentication posture of these brokers. As a result, the associated IOT device can be compromised by the remote attacker.

- A number of mosquitto message brokers analyzed during research were found to be configured without SSL/TLS (MQTT+SSL/TLS:8883) which makes the network communication vulnerable to Man-in-the-Middle (MitM) attacks as messages exchanged between the clients and the brokers can be hijacked and altered accordingly.
On the same note, mosquitto message brokers can leak information about the present state of the system. This results in information leakage to unauthorized entities and reveal the internal working of the associated devices.

Conclusion

The research highlights the deployment of MQTT message broker -- mosquitto in the real world and the security issues associated with them. It has become importance and even necessity to secure MQTT message brokers deployed for handling operations for IoT devices.