Introduction

IOT (internet of things) is all about network of interconnecting devices, collecting and exchanging data. MQTT (message queuing telemetry transport) is one of the ways to do it in a secure manner.

MQTT is an open OASIS and ISO standard (ISO/IEC PRF 20922)[3] lightweight, simple machine to machine TCP/IP based protocol which can be used for communication between the devices (wired and wireless both).

It is a Publish-Subscribe architecture based protocol, it does not require direct connection between devices as it relays all the massages via Broker (central server), which makes it very appropriate for the IOT technology as new devices can be easily added to the existing setup and also does not require compatibility with other devices.

Working mechanism

In MQTT protocol data flows in both direction from publisher to subscriber and from subscriber to publisher but in both cases the data is relayed through a central server called Broker. The Broker is just a software running on an operating system, it works as a post office regulating all the data traffic from one device to another and decides what data flows to which device but instead of using address of the device it uses topic for unique identification. The subscriber has to subscribe to a topic relevant to its requirement, It allows both one-to-many and many-to-one model, i.e. a subscriber can subscribe to any number of topics or any number of publisher can publish data to one or any number of topics. It is bi-directional so a device can send the message and at the same time can receive configuration data or control commands simultaneously.
QoS (quality of service)

- QoS 0 (at most once) - in this service level the message is sent only once and does not require any acknowledgment of whether the message is received/delivered or not.
- QoS 1 (at least once) - in the service level the message is multiple times till it receives the acknowledgment of delivery. There is a chance of duplicity in case of acknowledgement message is lost.
- QoS 3 (exactly once) - sender and receiver engage in 2 level handshake to ensure only one copy of the message is delivered.

MQTT packet structure

<table>
<thead>
<tr>
<th>Fixed header</th>
<th>Variable header</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>(always present)</td>
<td>(optional)</td>
<td>(optional)</td>
</tr>
<tr>
<td>Size: 2 byte</td>
<td>Size: variable</td>
<td>Size: variable</td>
</tr>
</tbody>
</table>
Wild card

when a client want to subscribe to a topic it can subscribe to a particular topic or it can subscribe to multiple topics simultaneously, it can be done by using wildcards. It can be used only for the subscription activities not publishing.

- Single level wildcard (+): it can replace only one topic level
  Eg. Myhome/1stfloor/room1/+/temperature
- Multilevel wildcards (#): it can replace multilevel topics
  Eg. Myhome/#/

Miscellaneous

We can send the payload/data of maximum of 268435455 bytes (~268Mb). The MQTT protocol has no limits on the number of subscriptions. However, you will be restrained by available memory.

POC

We are taking Raspberry pie OS as our broker and Kali Linux as subscriber. We are using mosquitto tool in both the OS to interact using MQTT protocol.

Open the command Terminal and run:

Step1. sudo apt-get update

Step2. sudo apt-get install mosquitto

Step3. sudo apt-get install mosquitto-clients
pi@osboxes:~ $ apt-get update
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done
Reading package lists... Done

Finding new packages...

Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
apparmor
The following NEW packages will be installed:
mqtt
0 upgraded, 1 newly installed, 0 to remove and 267 not upgraded.
Need to get 0 B/141 kB of archives.
After this operation, 327 kB of additional disk space will be used.
Selecting previously unselected package mqtt.
(Reading database ... 124818 files and directories currently installed.)
Preparing to unpack .../mqtt_1.4.10-3+deb9u4_i386.deb ...
Unpacking mqtt (1.4.10-3+deb9u4) ...
Processing triggers for systemd (232-2+deb9u1) ...
Processing triggers for man-db (2.76.1-2) ...
pi@osboxes:~ $
# Above commands are to be run on both the OS before moving forward.

Now in kali terminal (subscriber) we are subscribing to a topic called test.

**Step 4**  
`mosquitto_sub -u pie -h 192.168.180.147 -t test`
In Pi OS terminal (Broker) we will try to publish some message in the topic test.

**Step 5:**
```
mosquitto_pub -t test -h 192.168.180.174 -m "hello devices"
```

Here in the Kali terminal (subscriber) we can see the delivered message.
References