

IoT Security and Privacy

Lightweight IoT communication protocol - Message Queuing Telemetry Transport (MQTT)

YIER JIN

UNIVERSITY OF FLORIDA

EMAIL: YIER.JIN@ECE.UFL.EDU

SLIDES ARE ADAPTED FROM PROF. XINWEN FU @ UCF/UMASS

Learning Outcomes

Upon completion of this unit:

- Students will be able to restate the IoT communication protocol Message Queuing Telemetry Transport (MQTT)
- Students will be able to practice installation, configuration and use the MQTT implementation – Mosquitto
- Students will be able to practice the use of SSL with Mosquitto to secure communication
- Students will be able to practice the use of SSL with Mosquitto for authentication

Prerequisites and Module Time

Prerequisites

- Students should have taken classes on operating system and computer architecture.
- Students must have taken crypto and know how public key crypto and symmetric key crypto work.
- Students should have mastered programming Raspberry Pi.
- Students should know basic concepts of networking.

Module time

- Two-hour lecture
- One-hour homework

Outline

Message Queuing Telemetry Transport (MQTT)

MQTT implementation: mosquitto

MQTT Mosquitto transport security

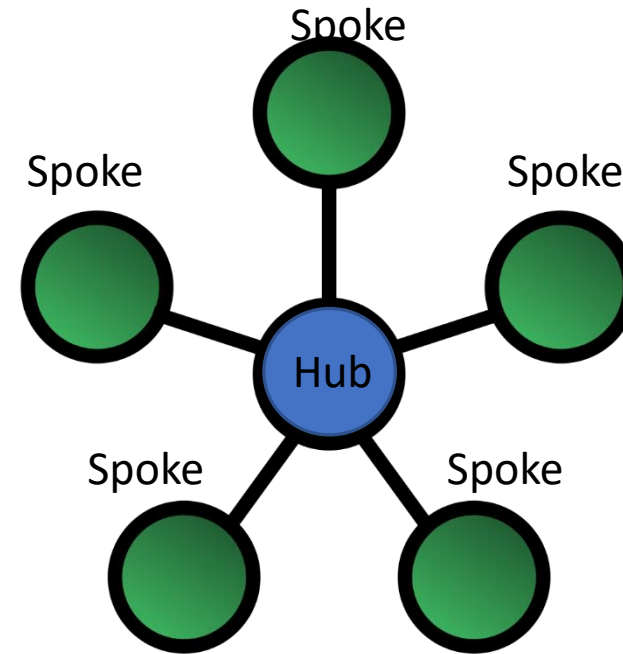
MQTT authentication

Messaging Broker System

A messaging broker system uses a publish/subscribe protocol based on a “hub and spoke” model

- Hub: server/broker
- Spokes: clients

Clients communicate with each other through the hub



Message Brokering Basic Terms

Broker, called “servers” too

- Accepts messages from clients
- Delivers the messages to any interested clients

Client

- Publishes a message to a topic, or
- Subscribes to a topic
- or both.

Topic: A namespace for messages on the broker

- A forward slash / is used to separate the topic hierarchy
- Clients do not need to initialize a topic before subscribing and publishing, and the broker will process the request automatically
- e.g., myhome/groundfloor/familyroom/humidity

Message Brokering Basic Terms (Cont'd)

Publish: a client sends a message to the broker, using a topic name.

Subscribe: a client notifies the broker the topics of interest

- The broker sends messages published to that topic to subscribers
- A client can subscribe to multiple topics.

Unsubscribe:

- Tell the broker not to send the client the messages to a particular topic any more

MQTT Introduction

MQTT is a messaging broker system

Clients can publish (Pub) messages and subscribe (Sub) to topics.

- Clients can both publish and subscribe.

A broker communicates with clients.

Topics can have subtopics.

- Topics starting with \$ are reserved for special topics
- Refer to AWS IoT topics

Why not just use HTTP?

HTTP is heavy

- A lot of fields in the headers.
- Needs multiple POST operations to distribute one message to multiple clients while a MQTT broker needs one publish

A message **brokering** system is light

- An MQTT packet can be only 2 bytes.

Representational state transfer (REST) vs HTTP

REST-compliant Web services define operations for clients to work on web resources

- **Roy Fielding** defines REST in 2000 in his Ph.D dissertation

Roy Fielding used REST to design HTTP 1.1 and Uniform Resource Identifiers (URI).

- RESTful Web service operations through HTTP verbs GET, POST, PUT, DELETE
- Note: HTTP was initiated by [Tim Berners-Lee](#)

Outline

Message Queuing Telemetry Transport (MQTT)

MQTT implementation: mosquitto

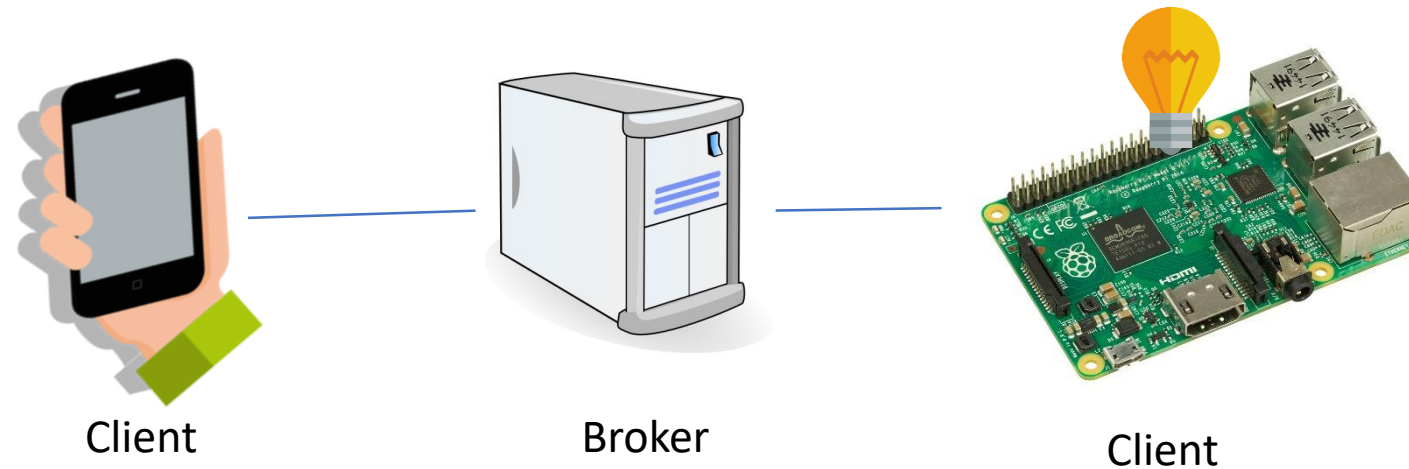
MQTT Mosquitto transport security

MQTT authentication

Example IoT system

Example IoT system components

- Smartphone – controller
- MQTT server – broker
- Client: IoT device



Mosquitto

Open source MQTT Mosquitto is a broker server

- Shipped with publishing and subscribing utilities - use *mosquitto_pub* and *mosquitto_sub*

Windows: binary installers on mosquitto.org

Linux: install “mosquitto” or “mosquitto-mqtt” with a package manager

Mac: use homebrew to install mosquitto.

- Add /usr/local/sbin to PATH by editing /etc/paths if necessary

Running Mosquitto

- Runs on port 1883 with no security by default

Testing Mosquitto

On one computer, we can test the whole MQTT system

1st console (terminal): the server

- `mosquitto -v` # verbose mode

2nd console: subscribing to MQTT Topic with Mosquitto

- `mosquitto_sub -h 127.0.0.1 -i testSub -t debug`
- **Host flag (-h)** indicates the mosquitto server
- **Identity flag (-i)** is client id. `mosquitto_sub` creates one if not provided.
- **Topic flag (-t)** indicates the topic to subscribe

Notice: no topic is pre-created on the server

- The topic is created when the subscriber or publisher connects to the server.

Raspberry Pi as MQTT Client

Choice one:

- Install mosquitto and use mosquitto_pub and mosquitto_sub to communicate with a MQTT server.

Choice two:

- Write a client with an MQTT library for a preferred language like Python

Install MQTT for Python

MQTT Python library: [Paho Python Client](#)

- Open source
- Supports the latest version of MQTT

Installation

- Install “pip”
- `pip install paho-mqtt`
- Or `pip3 install paho-mqtt` for python v3

Example MQTT Python Code for Raspberry Pi

```
1 import paho.mqtt.publish as publish
2 import time
3 print("Sending 0...")
4 publish.single("ledStatus", "0", hostname="macman")
5 time.sleep(1)
6 print("Sending 1...")
7 publish.single("ledStatus", "1", hostname="macman")
```

Single - Publish a single message to a broker, then disconnect cleanly.

Outline

Message Queuing Telemetry Transport (MQTT)

MQTT implementation: mosquitto

MQTT Mosquitto transport security

MQTT authentication

Mosquitto broker with SSL/TLS

Generating the server certificates

- `wget https://github.com/owntracks/tools/raw/master/TLS/generate-CA.sh` .
- This script generates a self signed certificate to be used by Mosquito for providing TLS for the MQTT and WebSocket protocol.
- `openssl` is needed.

The following files are generated:

- **ca.crt** – The CA (Certificate Authority, who published the host certificate) public certificate.
- **hostname.crt** – The hostname, that will run the mosquitto broker, public certificate.
- **hostname.key** – The hostname private key.

Create a folder for certificates and key files

Copy the certificates and key files to a folder.

- `sudo -s`
- `mkdir -p /etc/mosquitto/certs`
- `cp ca.crt /etc/mosquitto/certs`
- `cp hostname.* /etc/mosquitto/certs`

mosquitto.conf

mosquitto.conf

- Configuration file for mosquitto.
- Can be put anywhere.
- By default, mosquitto does not need a configuration file and will use the default values.
- Refer to the man page `mosquitto(8)` for information on how to load a configuration file.

Format

- Line with # as the very first character are comments.
- Configuration lines: a variable name and its value separated by a single space.

Mosquitto configuration

```
# Plain MQTT protocol
listener 1883

# End of plain MQTT configuration

# MQTT over TLS/SSL
listener 8883
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/hostname.crt
keyfile /etc/mosquitto/certs/hostname.key

# End of MQTT over TLS/SSL configuration
```

```
# Plain WebSockets configuration
listener 9001
protocol websockets

# End of plain Websockets configuration

# WebSockets over TLS/SSL
listener 9883
protocol websockets
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/hostname.crt
keyfile /etc/mosquitto/certs/hostname.key
```

Testing MQTT TLS/SSL configuration

```
mosquitto_pub --cafile /etc/mosquitto/certs/ca.crt -h localhost -t "test" -  
m "message" -p 8883
```

```
mosquitto_sub -t \${SYS}/broker/bytes/\# -v --cafile  
/etc/mosquitto/certs/ca.crt -p 8883
```

Outline

Message Queuing Telemetry Transport (MQTT)

MQTT implementation: mosquitto

MQTT Mosquitto transport security

MQTT authentication

Authentication

By default, no authentication

Unauthenticated encrypted support is provided through the use of the certificate based SSL/TLS based options cafile/capath, certfile and keyfile.

- The broker needs to provide the client a certificate

username/password authentication

Through password_file

- Define usernames and passwords.

If no encryption used, the username and password will be transmitted in plaintext

- SSL/TLS should be used

Authentication via certificate based encryption

Through `require_certificate`, which can be true or false

If `require_certificate` false, no certificate based authentication for clients

- Clients can verify server's certificate
- Authentication of clients will have to rely on username/password if needed

If `require_certificate` true, the client must provide a valid certificate to the server before further communication

- `use_identity_as_username` can affect the authentication.
 - If true, the Common Name (CN) from the client certificate is used as the identity
 - "If false, the client **must** (?) authenticate as normal (if required by `password_file`) through the MQTT options."

pre-shared-key based encryption

pre-shared-key based encryption through the `psk_hint` and `psk_file` options in the configuration file,

- The client must provide a valid identity and key to connect to the broker

```
pi@raspberrypi:~ $ mosquitto -c /etc/mosquitto/conf.d/mosquitto.conf
pi@raspberrypi:~ $ mosquitto_pub -h 129.63.17.134 -p 8883 -t mqtt --psk-identity
client --psk 1234abcd -m "Test: different side"
```

If `use_identity_as_username` is true

- The PSK identity is used for access control purposes.

If `use_identity_as_username` is false

- The client may still authenticate via username/password so that different users have different passwords?

Note: for programming paho does not support psk yet and c language libmosquitto supports it

Notes

Certificate and PSK based encryption are configured for each listener.

Authentication plugins can be created to replace the password authentication (password_file) and psk authentication psk_file

- For example, database lookups.

Multiple authentication schemes can be simultaneously supported

```
# Plain MQTT protocol
listener 1883

# End of plain MQTT configuration

# MQTT over TLS/SSL
listener 8883
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/hostname.crt
keyfile /etc/mosquitto/certs/hostname.key

# End of MQTT over TLS/SSL configuration
```

References

- [1] James Lewis, MQTT Introduction and Tutorial Part One - Message Brokers and why your IoT device should use them, February 17, 2016.
- [2] James Lewis, MQTT Tutorial for Raspberry Pi, Arduino, and ESP8266 - Send MQTT messages between 3 different platforms, February 24, 2016
- [3] Python Software Foundation, paho-mqtt 1.2, 2016
- [4] mosquitto.conf — the configuration file for mosquitto, 2016
- [5] Primal Cortex, MQTT Mosquitto broker with SSL/TLS transport security, March 31, 2016
- [6] J. Dunmire, SSL/TLS Client Certs to Secure MQTT, 2016
- [7] MosquittoAn Open Source MQTT v3.1/v3.1.1 Broker, Documentation, 2016
- [8] HuyITF, Configure SSL/TLS for MQTT broker mosquitto, Jun 2, 2016
- [9] Roger Light, libmosquitto — MQTT version 3.1 c client library, 2016
- [10] mosquitto.h, 2016