Evaluating MQTT Security
with Owntracks and Home Assistant

Vibha Iyengar, Alin Nagraj, Sathwik Nunna

Abstract— **Internet of Things has become a part of people's everyday life, and many people tend to provide sensitive information like health and location details through them, due to unawareness of its risks. This poses a challenge in meeting the expected security and privacy requirements for IoT devices, opening the scope for malicious behavior. Message Queue Telemetry Transport (MQTT) is one of the widely used communication protocols in the IoT domain, and is prone to become vulnerable when the required security configurations are neglected. This leads to exposure of their personal data out in the network.**

This paper describes five attack scenarios used to exploit the MQTT vulnerabilities, and evaluate the security features under each of it. All the experiments used the Mosquitto implementation of MQTT, Owntracks as location publisher, and Home Assistant as a subscriber to location information. A consolidated view of MQTT security features, and a proof-of-concept with suggestions for future work are the key contributions of the paper.

I. INTRODUCTION

Innovations in the area of digital things and Information Communication Technology are driving rapid deployment of Internet of Things (IoT) around the globe. According to IHS, the IoT market will grow from an installed base of 15.4 billion devices in 2015 to 30.7 billion devices in 2020 and 75.4 billion in 2025 [13]. Device to Device communications (D2D) in IoT are envisaged through various protocols such as Constrained Access Protocol (CoAP), Message Queue Telemetry Transport (MQTT) and MQTT-SN (for sensor networks). This work focuses on the IOT network which uses the MQTT protocol.

MQTT is an ISO standard (ISO/IEC 20922:2016), and works on top of TCP/IP [6]. MQTT works on the publisher/subscriber model, and consists of asynchronous message communication between a broker and a client, transmitted by verifying against a list of topics. An MQTT broker is a server that implements the MQTT protocol, and is also exposed by search engines like Shodan and Censys. A topic is a UTF-8 string, which is used by the broker to filter messages for each connected client. A topic consists of one or more topic levels, each topic level is separated by a forward slash. A client can either be a publisher or a subscriber or both. A publisher posts content to a topic and a subscriber receives content about a topic to which it subscribes to. MQTT acknowledges connections using connection codes/return codes [17].

MQTT is used in devices with limited resources for processing information so it was not designed with security in mind to retain the property of a lightweight protocol. The target for MQTT protocol was low powered devices but it is used in different mobile applications sometimes to leverage the resource requirement for this protocol, an example of this is the Owntrack location tracking application. It is also used in many cases for the application to act as a hub for the IoT devices to provide for monitoring, like the HomeAssistant application. This work uses Mosquitto, an open source implementation of MQTT to explore the vulnerabilities in some of the use cases of MQTT.

The rest of this paper is organized as follows. Section II describes the motivation behind this work. Section III discusses about the related work. Sections IV, V presents the approach, experiments and outcomes of this work. Section VI lists out the challenges faced. Finally, sections VII, VIII presents the conclusion and future work.

II. MOTIVATION

The previous studies [18] have indicated that MQTT protocol architecture has multiple vulnerabilities that can be exploited, some of them are:

1) Clients can subscribe without authentication
2) Vendors have no hard enforcement to implement authentication or encryption
3) A client can subscribe to unlimited topics, leading to revealing sensitive published messages
4) There exists no topic based authorization for subscribers or publishers, hence data forgery and data leakage are very likely, and
5) The use of wild-cards, like # and + in the topic subscription, enables an unauthorized client to access all the topic lists served by the broker. These evidently show the lack of authentication, authorization, integrity, confidentiality, and accountability in MQTT protocol.

Lucas Lundgren, the Senior Security Consultant at Fort-Consult presented many alarming exploits using MQTT protocol in the Defcon Hacker Conference in 2016 [15] and 2017. The masscan performed in the demo was able to extract all the sensitive information in the communication between the MQTT brokers and clients, including names, email IDs, phone numbers, physical addresses, and session tokens. The experiments also comprised of Cross Site Scripting (XSS), and SQL Injection attacks on web applications and databases using MQTT. Zack Whittaker [20] in 2017 revealed how exposed IoT brokers could let the hackers unlock prison cells, or modify heart pacemakers, both of which use MQTT protocol.

Since MQTT is widely used in the IoT domain, and is also responsible to transmit sensitive data between applications or devices, securing it is the need of the hour. Implementations of MQTT broker like mosquitto do provide authentication and TLS features, but not all users enable it. This creates inconsistency and loopholes in the entire IoT infrastructure.

Hence it is significant and critical to evaluate the strength of MQTT security features like Access Control List (ACL), SSL/TLS over MQTT, and other security extensions available today.

Although a lot of research is being done in this area, to the best of our knowledge, there is no work which provides a consolidated view of MQTT security. Thus we believe that the evaluation results presented in this work are novel and valuable.

III. RELATED WORK

Since MQTT is meant to be a lightweight protocol the official MQTT specification includes no mandatory requirements for any of the typical security related aspects such as authentication, authorization, data integrity, confidentiality and the like. [9], [5], [3], [6].

A consistent amount of research activity is focusing on analyzing and proposing extensions to a typical MQTT platform to implement additional security functionalities not present in the protocol standard. Neisse [8] implements a solution for the enforcement of security at MQTT layer which is a Model-based Security Toolkit called SecKit. It addresses the privacy and data protection requirement. M.Singh et al [7] proposed a secure version of MQTT in 2015 that they call SMQTT, which was based on Key/ Ciphertext Policy-Attribute Based Encryption(KP/CP- ABE) using lightweight Elliptic Curve Cryptography. This addresses the authentication problem in MQTT. Some studies also focused on the Transport Layer Security (TLS) client authentication, by imposing hardware security elements, while dealing with IoT MQTT communication [4].

Eclipse Foundation has MQTT server and client implementations called Mosquitto [16]. Mosquitto is used in several research activities, which includes evaluating MQTT for use in Smart City Services [2], and development of an environmental monitoring system [1]. Commercial products like Owntracks and the openHAB home automation project also have Mosquitto integrations. The developers have been persistent at monitoring security flaws in the implementation, and enhancing their security using ACLs and TLS features. Since Mosquitto is best suited for academia research, this project uses its MQTT implementation in all the experiments.

MQTT implementations provide a lot of options to make it completely secure, but still there is no silver bullet and the security requirements are often dependent on the use case. The large number of possible solutions and the lack of a de facto standard represents a security threat itself.

IV. APPROACH

The aim of the project was to evaluate vulnerabilities in the Message Queue Telemetry Transport (MQTT) protocol using Mosquitto, to provide a consolidated reflection of its security, and propose possible security patches. For an effective evaluation of MQTT security the following aspects were identified to be the most relevant:

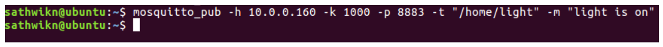
- Exploit vulnerabilities in MQTT using mosquitto implementation of MQTT
- Analyze the security properties achieved using in-built security features provided by Mosquitto
- Build a consolidated reflection of MQTT security
- Propose possible security patches based on the gaps observed

A. Project Description

The first step was background study on MQTT to understand the protocol and also to identify the gaps to work on. After a thorough research the next step was to hunt for applications that use MQTT and several applications were found out of which Owntracks and Home Assistant were picked. The test bed included Mosquitto, Owntracks and Homeassistant and the security features available in these components were explored. Home Assistant [12] is an open source home automation platform. It tracks and automates control within all devices at home, while the OwnTracks [10] allows users to track their location on Android and iOS phones and publish it to an MQTT broker. The focus was on evaluating the security of MQTT using Mosquitto therefore vulnerabilities were identified in the communication happening between various parties by implementing different attacks, and six security properties were explored in depth - confidentiality, integrity, authentication, authorization, accountability and timeliness. The final step was to look for mitigations for the identified vulnerabilities to fill in the gaps.

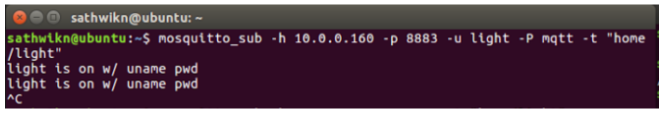
B. Experiment Setup

The project had 3 major components, the MQTT mosquitto broker and two client applications. First client was the publisher - an android application called Owntracks and second was the subscriber i.e. the Home Assistant application. These clients used MQTT protocol to communicate with the mosquitto broker. Home Assistant was configured to receive location updates from Owntracks via Mosquitto broker. Owntracks app was setup on a mobile device, while the home assistant and the mosquitto broker were on ubuntu virtual machines. For some of the experiments a malicious broker was run on a different virtual machine and a malicious publisher on an additional mobile device. Few experiments required installation of openssl certificates [19] in these devices.



```
sathwikn@ubuntu:~$ mosquitto_pub -h 10.0.0.160 -k 1000 -p 8883 -t "/home/light" -m "light is on"
sathwikn@ubuntu:~$ 
```

Fig. 1. Command used to publish a message to a topic



```
sathwikn@ubuntu: ~
sathwikn@ubuntu:~$ mosquitto_sub -h 10.0.0.160 -p 8883 -u light -P mqtt -t "home
/light"
light is on w/ uname pwd
light is on w/ uname pwd
^C
```

Fig. 2. Command used to subscribe to a topic

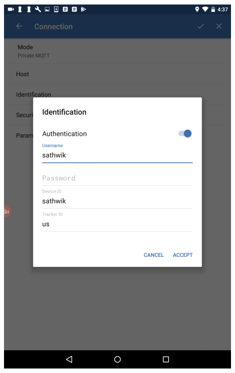

```
Connection
Mode
Private MQTT
Host
Identification
Securi
Param

Identification
Authentication
Username
sathwik
Password
Device ID
sathwik
Tracker ID
us
CANCEL  ACCEPT
```

Fig. 3. Owntracks configuration - identification

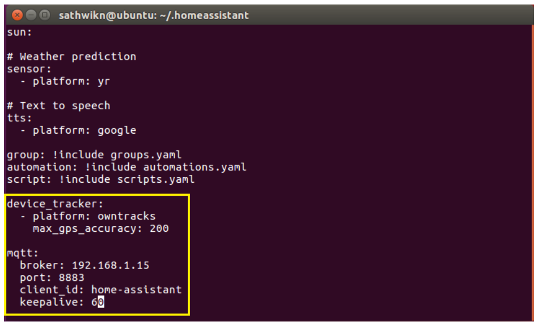

```
sathwikn@ubuntu: ~/.homeassistant
sun:

# Weather prediction
sensor:
  - platform: yr

# Text to speech
tts:
  - platform: google

group: !include groups.yaml
automation: !include automations.yaml
script: !include scripts.yaml

device_tracker:
  - platform: owntracks
    max_gps_accuracy: 200

mqtt:
  broker: 192.168.1.15
  port: 8883
  client_id: home-assistant
  keepalive: 60
```

Fig. 4. Home Assistant Configuration to integrate it with owntracks and MQTT broker

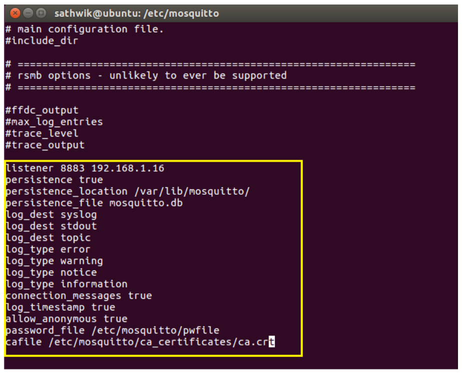

```
sathwik@ubuntu: /etc/mosquitto
# main configuration file.
#include_dir

# =================================================================
# rsmb options - unlikely to ever be supported
# =================================================================

#ffdc_output
#max_log_entries
#trace_level
#trace_output

listener 8883 192.168.1.16
persistence true
persistence_location /var/lib/mosquitto/
persistence_file mosquitto.db
log_dest syslog
log_dest stdout
log_dest topic
log_type error
log_type warning
log_type notice
log_type information
connection_messages true
log_timestamp true
allow_anonymous true
password_file /etc/mosquitto/pwfile
cafile /etc/mosquitto/ca_certificates/ca.crt
```

Fig. 5. Mosquitto Configuration relevant to our project

V. EXPERIMENTS AND OUTCOMES

A. *Experiment 1: Passive - Listen to the traffic on wireshark*

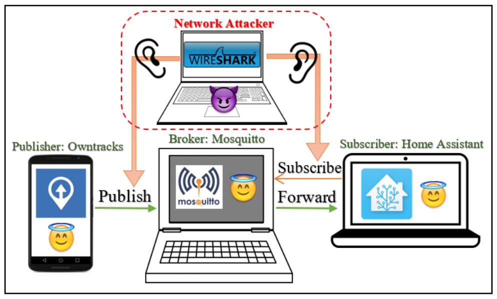


Fig. 6. Experiment 1 Setup

Testbed components: 1 Mosquitto broker, 1 publisher (Owntracks), 1 subscriber (Home Assistant), Wireshark

Procedure: The MQTT broker was on the virtual machine and the Owntracks was configured to connect to the broker. The details required for the connection were the identification information such as the username, password, device id and tracker id (Owntracks provides an option to check the status of the connection). The Homeassistant received updates from Owntracks, when it subscribed to the broker. This communication traffic was captured on Wireshark, which was examined to derive the outcomes in table I.

TABLE I
EXPERIMENT 1 OUTCOMES

ACL Enabled	TLS Enabled	Vulnerable	Observations
No	No	Yes	Topics and contents visible in clear text (Figure 7)
Yes	No	Yes	Topics, contents, username and passwords visible in clear text
Yes / No	Yes	No	TLS handshake happens before sub-pub communication Topics, contents, username and passwords are encrypted

Screen Captures:

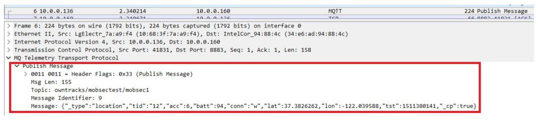

```
  6 10.0.0.136        2.340214        10.0.0.160           MQTT                224 Publish Message
> Frame 6: 224 bytes on wire (1792 bits), 224 bytes captured (1792 bits) on interface 0
> Ethernet II, Src: LgElectr_7a:a9:f4 (10:68:3f:7a:a9:f4), Dst: IntelCor_94:88:4c (34:e6:ad:94:88:4c)
> Internet Protocol Version 4, Src: 10.0.0.136, Dst: 10.0.0.160
> Transmission Control Protocol, Src Port: 41831, Dst Port: 8883, Seq: 1, Ack: 1, Len: 158
v MQ Telemetry Transport Protocol
  v Publish Message
    > 0011 0011 = Header Flags: 0x33 (Publish Message)
      Msg Len: 155
      Topic: owntracks/mobsectest/mobsec1
      Message Identifier: 9
      Message: {"_type":"location","tid":"12","acc":6,"batt":94,"conn":"w","lat":37.3826262,"lon":-122.039588,"tst":1511380141,"_cp":true}
```

Fig. 7. Clear text MQTT publish message

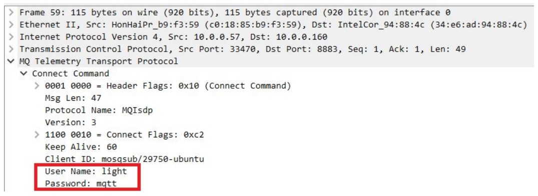

```
> Frame 59: 115 bytes on wire (920 bits), 115 bytes captured (920 bits) on interface 0
> Ethernet II, Src: HonHaiPr_b9:f3:59 (c0:18:85:b9:f3:59), Dst: IntelCor_94:88:4c (34:e6:ad:94:88:4c)
> Internet Protocol Version 4, Src: 10.0.0.57, Dst: 10.0.0.160
> Transmission Control Protocol, Src Port: 33470, Dst Port: 8883, Seq: 1, Ack: 1, Len: 49
v MQ Telemetry Transport Protocol
  v Connect Command
    > 0001 0000 = Header Flags: 0x10 (Connect Command)
      Msg Len: 47
      Protocol Name: MQIsdp
      Version: 3
    > 1100 0010 = Connect Flags: 0xc2
      Keep Alive: 60
      Client ID: mosqsub/29750-ubuntu
      User Name: light
      Password: mqtt
```

Fig. 8. Username and password revealed in the connect command without TLS

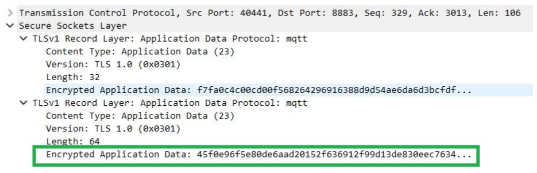

```
> Transmission Control Protocol, Src Port: 40441, Dst Port: 8883, Seq: 329, Ack: 3013, Len: 106
v Secure Sockets Layer
  v TLSv1 Record Layer: Application Data Protocol: mqtt
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 32
      Encrypted Application Data: f7fa0c4c00cd00f568264296916388d9d54ae6da6d3bcfdf...
  v TLSv1 Record Layer: Application Data Protocol: mqtt
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 64
      Encrypted Application Data: 45f0e96f5e80de6aad20152f636912f99d13de830eec7634...
```

Fig. 9. Encrypted MQTT text when TLS is used

B. *Experiment 2: Passive - Subscriptions to topics from malicious end*

Testbed components: 1 Mosquitto broker, 1 publisher (Owntracks), 1 subscriber (Home Assistant), 1 malicious subscriber (Mosquitto)

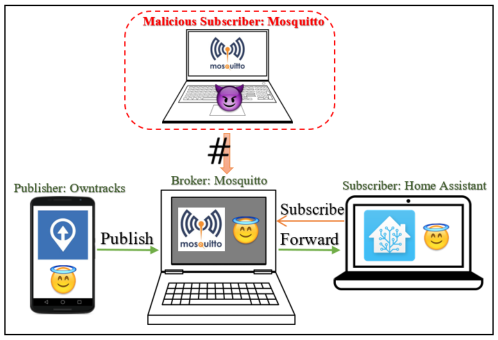


Fig. 10. Experiment 2 Setup

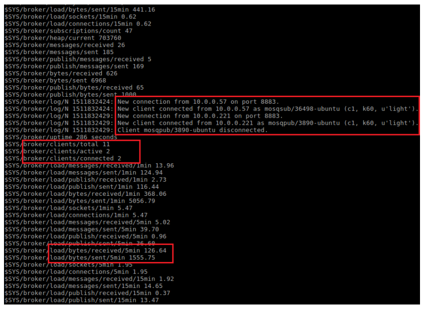
Procedure: The Experiment 1 configurations were retained, and a malicious subscriber was added to the environment. This new client subscribed to # wild-card topic, and other irrelevant topics on the Mosquitto broker. The parameters of this attack are listed in table II.

TABLE II
EXPERIMENT 2 OUTCOMES

ACL Enabled	**TLS Enabled**	**Vulnerable**	**Observations**
No	No	Yes	Easy subscription, and receipt of all content published
Yes	No	Yes	Cannot subscribe to specific topic without username and password, but user credentials can be obtained by passive listening
No	Yes	Yes	Subscription to # topic fetched content published on all topics to the broker TLS provides authentication based on certificate, not topic based authorization
Yes	Yes	No	Cannot subscribe to specific topic without username and password, and the credentials are not available due to encryption

Subscription to the $SYS/# topic fetched the log details from broker from time to time. This can be protected by blocking topic read $SYS/# in the acl_file in the mosquitto broker

Screen Capture:



```
$SYS/broker/load/bytes/sent/15min 441.16
$SYS/broker/load/sockets/15min 0.62
$SYS/broker/load/connections/15min 0.62
$SYS/broker/subscriptions/count 47
$SYS/broker/heap/current 703760
$SYS/broker/messages/received 26
$SYS/broker/messages/sent 185
$SYS/broker/publish/messages/received 5
$SYS/broker/publish/messages/sent 169
$SYS/broker/bytes/received 626
$SYS/broker/bytes/sent 6968
$SYS/broker/publish/bytes/received 65
$SYS/broker/publish/bytes/sent 1000
$SYS/broker/log/N 1511832424: New connection from 10.0.0.57 on port 8883.
$SYS/broker/log/N 1511832424: New client connected from 10.0.0.57 as mosqsub/36498-ubuntu (c1, k60, u'light').
$SYS/broker/log/N 1511832429: New connection from 10.0.0.221 on port 8883.
$SYS/broker/log/N 1511832429: New client connected from 10.0.0.221 as mosqpub/3890-ubuntu (c1, k60, u'light').
$SYS/broker/log/N 1511832429: Client mosqpub/3890-ubuntu disconnected.
$SYS/broker/uptime 286 seconds
$SYS/broker/clients/total 11
$SYS/broker/clients/active 2
$SYS/broker/clients/connected 2
$SYS/broker/load/messages/received/1min 13.96
$SYS/broker/load/messages/sent/1min 124.94
$SYS/broker/load/publish/received/1min 2.73
$SYS/broker/load/publish/sent/1min 116.44
$SYS/broker/load/bytes/received/1min 368.06
$SYS/broker/load/bytes/sent/1min 5056.79
$SYS/broker/load/sockets/1min 5.47
$SYS/broker/load/connections/1min 5.47
$SYS/broker/load/messages/received/5min 5.02
$SYS/broker/load/messages/sent/5min 39.70
$SYS/broker/load/publish/received/5min 0.96
$SYS/broker/load/publish/sent/5min 36.60
$SYS/broker/load/bytes/received/5min 126.64
$SYS/broker/load/bytes/sent/5min 1555.75
$SYS/broker/load/sockets/5min 1.95
$SYS/broker/load/connections/5min 1.95
$SYS/broker/load/messages/received/15min 1.92
$SYS/broker/load/messages/sent/15min 14.65
$SYS/broker/load/publish/received/15min 0.37
$SYS/broker/load/publish/sent/15min 13.47
```

Fig. 11. Details about publishers, subscribers, and internal log information revealed when subscribed to $SYS/#

C. *Experiment 3: Active - Impersonate Publisher*

Testbed components: 1 Mosquitto broker, 1 publisher (Owntracks), 1 subscriber (Home Assistant), 1 malicious publisher (Owntracks)

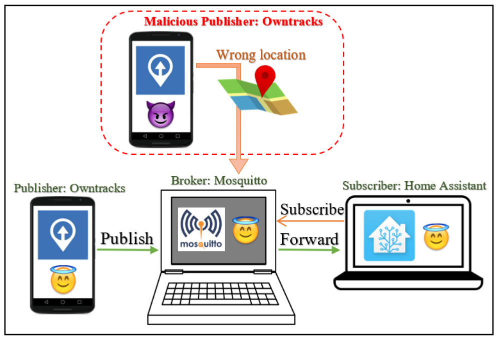


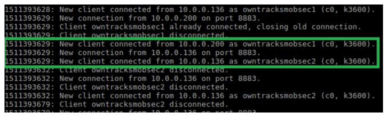
Fig. 12. Experiment 3 Setup

Procedure: Retaining Experiment 1 configurations, an additional attacker device was used with Owntracks app running along with FakeLocation app to manipulate the location seen by the subscriber. It was assumed that device id and username of genuine publisher are known, first the attack was carried out keeping one of the parameters (device id or username) different and then making them all identical. The results of these attacks are listed in table III.

TABLE III
EXPERIMENT 3 OUTCOMES

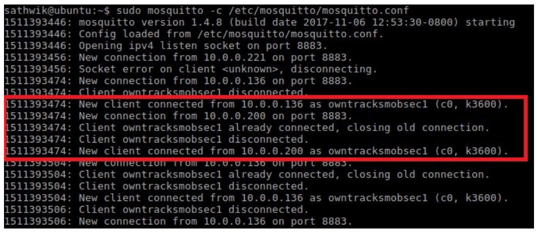
ACL Enabled	**TLS Enabled**	**Configuration of publisher**	**Vulnerable**	**Observations**
Yes	Yes	All kept same except Device ID or username	No	Broker accepts both publishers and considers them as different. Owntracks uses combination of username and device ID to distinguish a client
Yes	Yes	All kept same including Device ID or username	Yes	Broker connected to only one publisher at a time. The other one is considered as old connection, and is disconnected.

Screen Captures:



```
1511393628: New client connected from 10.0.0.136 as owntracksmobsec1 (c0, k3600).
1511393629: New connection from 10.0.0.200 on port 8883.
1511393629: Client owntracksmobsec1 already connected, closing old connection.
1511393629: Client owntracksmobsec1 disconnected.
1511393629: New client connected from 10.0.0.200 as owntracksmobsec1 (c0, k3600).
1511393629: New connection from 10.0.0.136 on port 8883.
1511393629: New client connected from 10.0.0.136 as owntracksmobsec2 (c0, k3600).
1511393632: Client owntracksmobsec2 disconnected.
1511393632: New connection from 10.0.0.136 on port 8883.
1511393632: Client owntracksmobsec2 disconnected.
1511393632: New client connected from 10.0.0.136 as owntracksmobsec2 (c0, k3600).
1511393679: Client owntracksmobsec2 disconnected.
1511393679: New connection from 10.0.0.136 on port 8883.
```

Fig. 13. Different connections established when different Device Ids are used by two publishers



```
sathwik@ubuntu:~$ sudo mosquitto -c /etc/mosquitto/mosquitto.conf
1511393446: mosquitto version 1.4.8 (build date 2017-11-06 12:53:30-0800) starting
1511393446: Config loaded from /etc/mosquitto/mosquitto.conf.
1511393446: Opening ipv4 listen socket on port 8883.
1511393456: New connection from 10.0.0.221 on port 8883.
1511393456: Socket error on client <unknown>, disconnecting.
1511393474: New connection from 10.0.0.136 on port 8883.
1511393474: Client owntracksmobsec1 disconnected.
1511393474: New client connected from 10.0.0.136 as owntracksmobsec1 (c0, k3600).
1511393474: New connection from 10.0.0.200 on port 8883.
1511393474: Client owntracksmobsec1 already connected, closing old connection.
1511393474: Client owntracksmobsec1 disconnected.
1511393474: New client connected from 10.0.0.200 as owntracksmobsec1 (c0, k3600).
1511393504: New connection from 10.0.0.136 on port 8883.
1511393504: Client owntracksmobsec1 already connected, closing old connection.
1511393504: Client owntracksmobsec1 disconnected.
1511393504: New client connected from 10.0.0.136 as owntracksmobsec1 (c0, k3600).
1511393506: Client owntracksmobsec1 disconnected.
1511393506: New connection from 10.0.0.136 on port 8883.
```

Fig. 14. Old connection disconnected when same Device Id and username used by two publishers

D. Experiment 4: Active - Replay Attack

Testbed components: 1 Mosquitto broker, 1 publisher (Owntracks), 1 subscriber (Home Assistant), Wireshark, Colasoft (packet replay tool) [21]

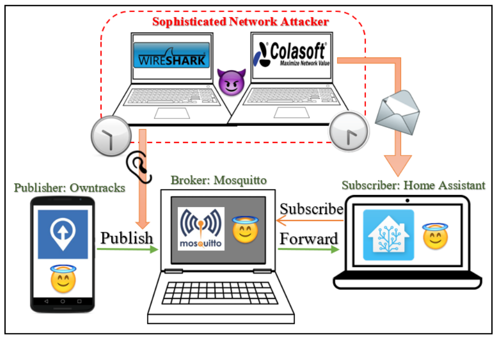


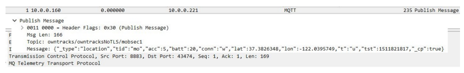
Fig. 15. Experiment 4 Setup

Procedure: Retaining Experiment 1 configurations and packet captures, one publish packet was replayed using a packet replay tool, Colasoft [21]. This experiment failed because a TCP Reset was sent by the subscriber as the segment was already marked as received at the TCP layer. Manipulating segment sequence numbers required delving too much into the TCP layer which was out of scope for the project.

TABLE IV
EXPERIMENT 4 OUTCOMES

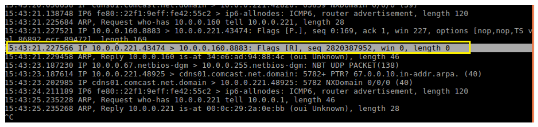
ACL Enabled	TLS Enabled	Vulnerable	Observations
No / Yes	No / Yes	Depends on sophistication of attack on Transport layer	Replay of publish message using a packet replay tool, was rejected at the Transport layer at the subscriber because of sequence number mismatch Thus, successful replay attacks require manipulations at the Transport layer and lower which is out of scope for our project

Screen Captures:



```
1 10.0.0.160     0.000000     10.0.0.221     MQTT     235 Publish Message
  Publish Message
    > 0011 0000 = Header Flags: 0x30 (Publish Message)
      Msg Len: 166
      Topic: owntracks/owntracksNoTLS/mobsec1
      Message: {"_type":"location","tid":"mo","acc":5,"batt":20,"conn":"w","lat":37.3826348,"lon":-122.0395749,"t":"u","tst":1511821817,"_cp":true}
Transmission Control Protocol, Src Port: 8883, Dst Port: 43474, Seq: 1, Ack: 1, Len: 169
MQ Telemetry Transport Protocol
```

Fig. 16. Packet captured for replay



```
15:43:20.036858 IP cdns01.comcast.net.domain > 10.0.0.221.42826: 65859 NXDomain 0/0/0 (39)
15:43:21.138748 IP6 fe80::22f1:9eff:fe42:55c2 > ip6-allnodes: ICMP6, router advertisement, length 120
15:43:21.225684 ARP, Request who-has 10.0.0.160 tell 10.0.0.221, length 28
15:43:21.227521 IP 10.0.0.160.8883 > 10.0.0.221.43474: Flags [P.], seq 0:169, ack 1, win 227, options [nop,nop,TS v
al 86892 ecr 89472], length 169
15:43:21.227566 IP 10.0.0.221.43474 > 10.0.0.160.8883: Flags [R], seq 2820387952, win 0, length 0
15:43:21.229458 ARP, Reply 10.0.0.160 is-at 34:e6:ad:94:88:4c (oui Unknown), length 46
15:43:23.187230 IP 10.0.0.67.netbios-dgm > 10.0.0.255.netbios-dgm: NBT UDP PACKET(138)
15:43:23.187614 IP 10.0.0.221.48925 > cdns01.comcast.net.domain: 5782+ PTR? 67.0.0.10.in-addr.arpa. (40)
15:43:23.202985 IP cdns01.comcast.net.domain > 10.0.0.221.48925: 5782 NXDomain 0/0/0 (40)
15:43:24.211189 IP6 fe80::22f1:9eff:fe42:55c2 > ip6-allnodes: ICMP6, router advertisement, length 120
15:43:25.235228 ARP, Request who-has 10.0.0.221 tell 10.0.0.1, length 46
15:43:25.235268 ARP, Reply 10.0.0.221 is-at 00:0c:29:2a:0e:bb (oui Unknown), length 28
^C
```

Fig. 17. Tcp reset sent from subscriber TCP stack during the replay attack

E. Experiment 5: Active - Communication Hijack- Malicious Broker, Malicious Publisher

Testbed components: 1 Mosquitto broker, 1 publisher (Owntracks), 1 subscriber (Home Assistant), 1 malicious broker (Mosquitto), 1 malicious publisher (Owntracks), Nmap and Arping command line tools

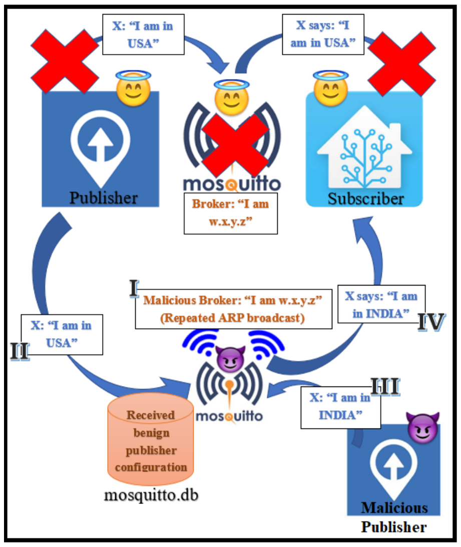


Fig. 18. Experiment 5 Setup

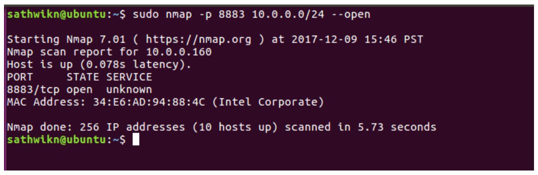
Procedure: Retaining Experiment configurations, the malicious broker was configured with the benign brokers IP address which can be easily obtained by an nmap scan for the port. To make sure that the publishers and subscribers connected to malicious broker only, a repeated ARP broadcast was maintained so that all ARP table entries had the malicious MAC. Once the genuine publisher was

TABLE V
EXPERIMENT 5 OUTCOMES

ACL Enabled	TLS Enabled	Vulnerable	Observations
No / Yes	No	Yes	One can impersonate a genuine publisher, and get the subscriber believe that it is still receiving information from the genuine publisher. The original communication channel is hijacked.
No / Yes	Yes	No	The original communication is not tampered. The subscriber disconnects when the malicious broker broadcasts itself and tries to connect with the subscriber.

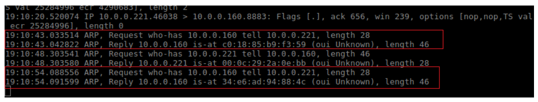
connected to malicious broker, information like username, device id could be obtained using the publish messages. Tracker id could also be obtained from the mosquitto.db. All this information was sufficient to impersonate genuine publisher and send fake information in publish messages to the subscriber.

Screen Captures:



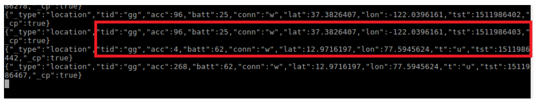
```
sathwikn@ubuntu:~$ sudo nmap -p 8883 10.0.0.0/24 --open

Starting Nmap 7.01 ( https://nmap.org ) at 2017-12-09 15:46 PST
Nmap scan report for 10.0.0.160
Host is up (0.078s latency).
PORT     STATE SERVICE
8883/tcp open  unknown
MAC Address: 34:E6:AD:94:88:4C (Intel Corporate)

Nmap done: 256 IP addresses (10 hosts up) scanned in 5.73 seconds
sathwikn@ubuntu:~$ 
```

Fig. 19. Nmap result to retrieve the IP of the broker



```
S val 25284996 ecr 4290683], length 2
19:10:20.520074 IP 10.0.0.221.46038 > 10.0.0.160.8883: Flags [.], ack 656, win 239, options [nop,nop,TS val
 ecr 25284996], length 0
19:10:43.033514 ARP, Request who-has 10.0.0.160 tell 10.0.0.221, length 28
19:10:43.042822 ARP, Reply 10.0.0.160 is-at c0:18:85:b9:f3:59 (oui Unknown), length 46
19:10:48.303541 ARP, Request who-has 10.0.0.221 tell 10.0.0.160, length 46
19:10:48.303580 ARP, Reply 10.0.0.221 is-at 00:0c:29:2a:0e:bb (oui Unknown), length 28
19:10:54.088556 ARP, Request who-has 10.0.0.160 tell 10.0.0.221, length 28
19:10:54.091599 ARP, Reply 10.0.0.160 is-at 34:e6:ad:94:88:4c (oui Unknown), length 46
```

Fig. 20. Tcpdump: Same IP and different MAC address of two brokers - One Original and other Malicious



```
86278,"_cp":true}
{"_type":"location","tid":"gg","acc":96,"batt":25,"conn":"w","lat":37.3826407,"lon":-122.0396161,"tst":1511986402,"
_cp":true}
{"_type":"location","tid":"gg","acc":96,"batt":25,"conn":"w","lat":37.3826407,"lon":-122.0396161,"tst":1511986403,"
_cp":true}
{"_type":"location","tid":"gg","acc":4,"batt":62,"conn":"w","lat":12.9716197,"lon":77.5945624,"t":"u","tst":1511986
442,"_cp":true}
{"_type":"location","tid":"gg","acc":268,"batt":62,"conn":"w","lat":12.9716197,"lon":77.5945624,"t":"u","tst":15119
86467,"_cp":true}
```

Fig. 21. At the Subscriber end: Different locations with same tid (Attack successful)

Based on the experiments, a consolidated reflection of MQTT security is given in table VI.

TABLE VI
CONSOLIDATED REFLECTION OF MQTT SECURITY

	Only ACL	Only TLS	ACL & TLS
Authentication	✓		✓
Authorization	✓		✓
Confidentiality		✓	✓
Integrity		✓	✓
Accountability			✓

VI. CHALLENGES

There were challenges at every phase of the project because of limited publicly available documentation and lack of interoperability of the MQTT implementation and security solutions. Also, there were few open source MQTT mobile applications which have sufficient documentation and were compatible to Mosquitto broker. Haptik was one such application found, but it is available for use only within India [11]. Hence all the experiments in this paper were restricted to Owntracks and Home Assistant applications. The testbed environment setup required multiple trials and iterations, due to lack of consolidated documentation to guide the process. Open Source implementations of MQTT security extensions Seckit and SMQTT were not readily available. Although the developers of SecKit mentioned that it is an open source implementation, the integration to MQTT was not available on open source platforms. On the other hand, the existing implementation of SMQTT could not be directly integrated to the use case studied in this paper. This lead to re-scoping of project goals at an intermediate stage.

VII. CONCLUSION AND FUTURE WORK

MQTT is a lightweight protocol, and hence suitable for IoT environment. The need for integration of security features, and its evaluation is vital. The experiments from this paper suggest that enabling ACL and TLS while using MQTT is a good way to achieve important security properties, like authentication, authorization, confidentiality, integrity and accountability in the communication process. The findings also recommend disabling dollar sys hash and certain other default configurations in the broker to avoid exposing the stored logs.

As part of the future work, alternate mitigation techniques in place of SSL/TLS needs to be explored, because of the overhead involved in using SSL/TLS with constrained devices [14]. Another activity would be to identify more attack scenarios involving MQTT vulnerabilities, and evaluate existing security patches against them.

ACKNOWLEDGMENT

We would like to take this opportunity to thank Dr. Patrick Tague for his continued guidance and support all through the project. This project would not have been possible without his expertise in the area of IoT Security. Also, we would like to thank Carnegie Mellon University for providing us the necessary infrastructure required for this project.

REFERENCES

[1] C. G. Bellavista, Paolo and R. Zamagna. The pervasive environment sensing and sharing solution. *Sustainability 9 (4): 585. doi:10.3390/su9040585*, 2017.
[2] A. et.al. Comparison of the cupus middleware and mqtt protocol for smart city services. *13th International Conference on Telecommunications (ConTEL) 18. doi:10.1109/ConTEL.2015.7231225*, 2015.
[3] A. N. et.al. Authorization mechanism for mqtt-based internet of things. *IEEE*, 2016.
[4] A. O. et.al. Access control in iot: Survey & state of the art. *IEEE*, 2016.
[5] C. L. et.al. Securing smart maintenance services: Hardware-security and tls for mqtt. *IEEE*, 2015.
[6] D. M. et.al. Internet of things: Survey on security and privacy. 2016.
[7] G. P. et.al. The day after mirai: A survey on mqtt security solutions after the largest cyber-attack carried out through an army of iot devices. *ResearchGate conference*, 2017.
[8] M. S. et.al. Secure mqtt for internet of things (iot). *IEEE*, 2015.
[9] R. N. et.al. Enforcement of security policy rules for the internet of things. *ResearchGate conference*, 2014.
[10] http://owntracks.org/. Owntracks application website.
[11] https://haptik.ai/about app. Haptik application website.
[12] https://home assistant.io/. Home assistant application website.
[13] https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates 2016/#1ab2da2a292d. Roundup of internet of things forecasts and market estimates, 2016.
[14] https://www.hivemq.com/blog/mqtt-security-fundamentals-tls ssl. Tls adds significant overhead when used in constrained devices.
[15] N. H. Lucas Lundgren. Light weight protocol: Critical implications https://www.youtube.com/watch?v=o7qdvzr0t2c. *DEF CON 24.*

[16] Mosquitto. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software; 2(13) 265; doi:10.21105/joss.00265.*
[17] H. MQ. Mqtt essentials, https://www.hivemq.com/blog/mqtt-essentials-part-3-client-broker -connection-establishment.
[18] S. B.-V. G.-V. I.-A. Nagraj. Discovering and investigating vulnerabilities in smart-home iot devices using shodan. *Unpublished Report.*
[19] G. Valkov. Tls certificate generation scripts for owntracks. *https://github.com/owntracks/tools/blob/master/TLS/generate-CA.sh.*
[20] Z. Whittaker. Exposed iot servers let hackers unlock prison cells, modify pacemakers. *Article on www.zdnet.com, July 31, 2017.*
[21] www.runasia.com.hk. Colasoft application website.

APPENDIX

A. Contributions of individual team members:

TABLE VII
PROJECT TIMELINE

Task	Completion Date	Deliverable	Alin	Sathwik	Vibha
1	Sep 21	Project Pitch Video	✓	✓	✓
2	Sep 25	Background study about MQTT	✓	✓	✓
3	Oct 1	Virtual setup of MQTT environment		✓	
4	Oct 12	SoW Presentation and Report	✓	✓	✓
5	Oct 15	Search for applications/ IoT devices that use MQTT		✓	
6	Oct 20	Study the use cases found in Task 5 and find vulnerabilities	✓	✓	✓
7	Oct 28	Design experiments and exploit techniques for the vulnerabilities found in Task 6	✓		✓
8	Nov 4	Explored the SecKit and SMQTT extensions, and searched for implementations that can be integrated into the environment	✓	✓	✓
9	Nov 9	Progress report	✓	✓	✓
10	Nov 25	Perform the experiments designed in Task 7	✓	✓	✓
11	Nov 27	Consolidated the results found in the experiments	✓	✓	✓
12	Nov 30	Project Final Presentation	✓	✓	✓
13	Dec 3	Explored to find alternatives of TLS	✓	✓	✓
14	Dec 7	Project Report, Poster, Demo Video	✓	✓	✓

The table VII indicates the task each team member was responsible to accomplish. All the members contributed equally in the preparation and presentation of the course deliverables. Sathwik was responsible to setup the experiment testbed, while Alin and Vibha were responsible to design the experiments to be performed. Apart from these tasks, all other ones including background study, and execution of experiments required the entire team to work together. Moreover, due to the many devices involved in the experiments, a collaborative work was inescapable.

B. Other project deliverables:

Link for Video: https://drive.google.com/a/west.cmu.edu/file/d/1 DjR3mMXZ6QdDaYq6FPo5-PCbvGKUpso2/view?usp=sharing

Link for Poster: https://drive.google.com/a/west.cmu.edu/file/d/ 11a7GYnH91FJOieeD8knJnH2rAoiCGNHx/view?usp=sharing