**COMP.6610.201 Advanced Topics in Network Security**

**Assignment 4 – Secure IoT**

**By: David Rocca, Isaac Pittman, Brian Carlson, and Jeremie Nieto**

**10 points**

**Requirements:**

1. Mosquitto MQTT offers *username/password authentication* and *pre-shared-key based encryption/authentication*. Please search the Internet and read anything that can be found to set up one of these two authentication strategies for mosquitto. Note: link encryption must be provided between the clients and Document the setup procedure and test results, including all the commands. (4 points)

We decided that we were going to do research on the Username/Password authentication because we are writing a mobile application as one of our clients. Username and password will allow multiple mobile applications to act as our “Client 2”. For Client 1 and the Broker they will be on the raspberry pi. Client 1 will be a Python program.

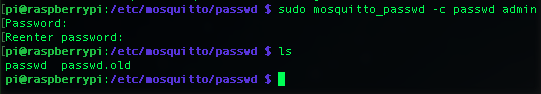
At this point we have used the same process that we mentioned in our last assignment (assignment 3) to generate the certificates and keys for the Server, CA, and Client (However, we will not be using the client in this assignment)

Using the official MQTT documentation at <https://mosquitto.org/man/mosquitto_passwd-1.html>we did the following to set up our MQTT server.

First we created our password file directory:



Next, using the official documentation, we created the password file:

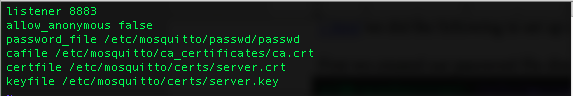


Username: admin

Password: fuiscool

Next, we set up our mosquitto configuration file:

Located at: /etc/mosquitto/conf.d/ssl.conf



In this file we turn anonymous connections off, set the directory of the password file, and the CA certificate and server Certificates and keyfiles.

At this point we can start the server:



My PEM password that I used when I created my certificates for assignment 3: david

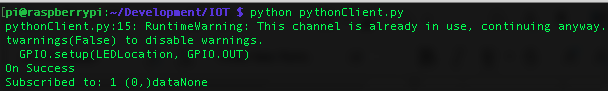
When we try to connect with one of our clients that is set up with SSL but NOT with the username and password:

E/AndroidRuntime: FATAL EXCEPTION: MQTT Rec: paho105425058324312

Process: roccasoftware.iot, PID: 27722

SIDE NOTE: At this point the android program crashes. HOWEVER, it is not required that we handle that crash for this assignment. For our final project we will recover from that exception.

However when we run our Python Client with the username and password everything is okay, and the client subscribes to the “SSLTOPIC” :



1. Write programs in Python or C/C++/Java for the two clients. The program on Client 2 sends commands via the broker to the program on Client 1 in order to control a sensor such as a LED on Client 1, which is a Raspberry Pi. Post a video of the sensor control (e.g., turning on or off) to youtube and provide the link in the report. Zip all programs and submit the zip file as a standalone document to Blackboard. (6 points)

For our Clients we have a Python Client on the same Pi as our broker. This client will be responsible for turning on the light on our breadboard.

For our Python Client we used Professor Fu’s Python example as a guide, however we wrote ours in 2.7.

To handle the two important parts of this assignment, the SSL and the Username/Password, we used the Paho MQTT python library’s two commands take care of it:



For the Android part of our program we had to take one extra step. Android does not handle CRT files. We had to use an external library called BouncyCastles to fix the format of our CRT file into something that Android can handle:

Sorry for bad formatting Could not take a good Screenshot:

Davids-MBP:temp davidrocca$ keytool -import -alias DO\_NOT\_TRUST\_testca -file ca.crt -keypass david -keystore IOT.bks -storetype BKS -storepass david1 -providerClass org.bouncycastle.jce.provider.BouncyCastleProvider -providerpath bcprov-ext-jdk14-1.53.jar

Owner: EMAILADDRESS=ca@example.com, CN=DO\_NOT\_TRUST\_testca, O=DO\_NOT\_TRUST\_mqttpi, ST=Massachusetts, C=US

Issuer: EMAILADDRESS=ca@example.com, CN=DO\_NOT\_TRUST\_testca, O=DO\_NOT\_TRUST\_mqttpi, ST=Massachusetts, C=US

Serial number: f90f79d7cad8278e

Valid from: Thu Nov 17 23:12:48 EST 2016 until: Wed May 11 00:12:48 EDT 2022

Certificate fingerprints:

MD5: 61:CB:BB:89:36:FC:CA:E8:14:55:98:E1:30:3A:80:6D

SHA1: 49:BD:B4:AC:B7:62:CA:61:B8:52:E7:AD:50:6C:EA:A1:01:AD:A8:FE

SHA256: 34:11:34:2E:0B:80:1F:94:EA:6F:C5:52:76:0B:A5:E9:CF:1B:FE:35:07:BA:20:49:C6:E6:CC:CC:38:13:77:33

Signature algorithm name: SHA256withRSA

Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false

AuthorityKeyIdentifier [

KeyIdentifier [

0000: 03 96 8A 84 7B 7E 09 E2 13 47 C4 21 0F 95 4C A4 .........G.!..L.

0010: A7 DB 4A A1 ..J.

]

]

#2: ObjectId: 2.5.29.19 Criticality=false

BasicConstraints:[

CA:true

PathLen:2147483647

]

#3: ObjectId: 2.5.29.14 Criticality=false

SubjectKeyIdentifier [

KeyIdentifier [

0000: 03 96 8A 84 7B 7E 09 E2 13 47 C4 21 0F 95 4C A4 .........G.!..L.

0010: A7 DB 4A A1 ..J.

]

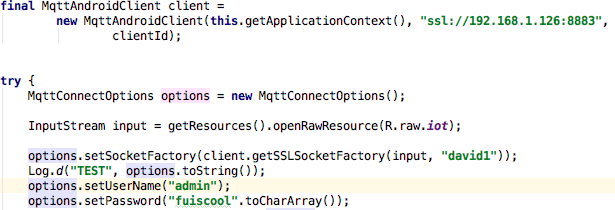
]

Trust this certificate? [no]: yes

Certificate was added to keystore

The file that was generated was then added to the Android Project.

The android Client also used a MQTT Paho library. This next snippet shows the adding of the SSL cert, username and password to the connection attempt in Java. Note the use of the “david1” password when accessing the BKS file. “David1” was set in the command used above to generate the BKS file.



The video is located at:

https://youtu.be/E7CcXEur1TM