

amazonPrime

Shop. Listen. Watch.
Read. Photos. > [Try Prime free](#)



[Home](#)

[MQTT](#)

[Networking](#)

[My MQTT Python Scripts](#)

Updated: May 15, 2017

Mosquitto SSL Configuration -MQTT TLS Security

Configuring TLS on Mosquitto

In this tutorial we will configure the mosquitto MQTT broker to use **TLS** security.

We will be using **openssl** to create our own Certificate authority (**CA**), Server keys and certificates.

We will also test the broker by using the Paho Python client to connect to the broker using a **SSL connection**.

You should have a basic understanding of **PKI**, certificates and keys before proceeding. See

Polls

What Would you
Like to Learn
More About?

- ☐ Internet Protocols
- ☐ Python
- ☐ Networking
- ☐ MQTT
- ☐ Mosquitto

Vote

[View Results](#)

Categories

Encryption and Digital Signatures For Beginners

The steps covered here will create an **encrypted connection** between the MQTT broker and the MQTT client just like the one between a web browser client and a Web Server.

In this case we only need a **trusted server certificate** on the Broker.

We **do not need** to create client certificates and keys.

Client Requirements

- A CA (certificate authority) certificate of the CA that has signed the server certificate on the Mosquitto Broker.

Broker Requirements

- CA certificate of the CA that has signed the server certificate on the Mosquitto Broker.
- CA certificated server certificate.
- Server Private key for decryption

Creating and Installing Broker Certificates and keys

To create these certificates and keys we use the **openssl** software.

To get the software on Windows you need to install **openssl** which is included as part of the **mosquitto**

- [home](#)
- [networking](#)
- [Internet](#)
- [IOT](#)
- [Mosquitto](#)
- [MQTT](#)
- [Networking](#)



Subscribe to Newsletter

Email *

Subscribe!

broker install instructions.

On Linux you can install openssl using :

sudo apt-get install openssl

Although the commands to create the various certificates and keys are given in this [Mosquitto manual page](#). Here is a quick snapshot:

It is important to use different certificate subject parameters for your CA, server and clients. If th separately, the broker/client will not be able to distinguish between them and you will experience

Certificate Authority

Generate a certificate authority certificate and key.

o openssl req -new -x509 -days <duration> -extensions v3_ca -keyout ca.key -out ca.crt

Server

Generate a server key.

~~o openssl genrsa -des3 -out server.key 2048~~

Generate a server key without encryption.

o openssl genrsa -out server.key 2048

Generate a certificate signing request to send to the CA.

**Can't use until you have
create your own ca.crt
and ca.key files**

**creates password
protected key**

Use this option

There is a problem with the page because **openssl** no longer comes with a **CA certificate**, and so you will need to create your own **self signed CA certificate**.

You should also note that when you generate keys you **shouldn't use** encryption (the **-ds3** switch) for the server certificate as this creates a **password protected** key which the broker can't decode.

Note the certificates and keys created can be used on the Mosquitto broker/server, and also on a web



server, which is why you see the term server used in the Mosquitto manual and not broker.

Overview of Steps

1. Create a **CA key pair**
2. Create **CA certificate** and use the **CA key** from step 1 to sign it.
3. Create a **broker key pair** don't password protect.
4. Create a **broker certificate** request using key from step 3
5. Use the **CA certificate** to sign the **broker certificate** request from step 4.
6. Now we should have a **CA key file**, a **CA certificate file**, a **broker key file**, and a **broker certificate file**.
7. Place all files in a directory on the broker e.g. certs
8. Copy the **CA certificate file** to the client.
9. Edit the **Mosquitto conf** file to use the files -details below
10. Edit the client script to use TLS and the CA certificate. -details below

Detailed Steps Windows

Note this as done on a windows XP machine. I did not install any extra software other than that what I needed to install as part of the original broker setup.

The same commands and procedures apply to linux but the folder locations will be different and you may need to change permissions, as well as using the SU command.

Step 1:

First create a key pair for the **CA**

Command is: **openssl genrsa -des3 -out ca.key 2048**

```
C:\steve>openssl genrsa -des3 -out ca.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001)
Enter pass phrase for ca.key:
Verifying - Enter pass phrase for ca.key:
```

Note: it is OK to create a password protected key for the CA.

Step 2:

Now Create a certificate for the CA using the **CA key** that we create in step 1

Command is: **openssl req -new -x509 -days 1826 -key ca.key -out ca.crt**

```
C:\steve>openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
Enter pass phrase for ca.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name <2 letter code> [AU]:UK
State or Province Name <full name> [Some-State]:Shropshire
Locality Name <eg, city> []:Ironbridge
Organization Name <eg, company> [Internet Widgits Pty Ltd]:CAmaster
Organizational Unit Name <eg, section> []:TEST
Common Name <e.g. server FQDN or YOUR name> []:ws4
Email Address []:steve@testemail.com
```

Step 3:

Now we create a server key pair that will be used by the broker

Command is: **openssl genrsa -out server.key 2048**

```
C:\steve>
C:\steve>openssl genrsa -out server.key 2048
Generating RSA private key, 2048 bit long modulus
.....+++
e is 65537 (0x10001) .....
```

Step 4:

Now we create a certificate request **.csr**. When filling out the form the **common name** is important and is usually the **domain name** of the server.

Because I'm using Windows on a local network I used the Windows name for the computer that is running the Mosquitto broker which is **ws4**.

You could use the IP address or Full domain name. You must use the same name when configuring the client connection.

Command is: **openssl req -new -out server.csr -key server.key**

```
C:\steve>openssl req -new -out server.csr -key server.key
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:UK
State or Province Name (full name) [Some-State]:Shropshire
Locality Name (eg, city) [Ironbridge]
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Server-cert
Organizational Unit Name (eg, section) [test]
Common Name (e.g. server FQDN or YOUR name) [ws4]
Email Address [steve@testemail.com]
```

Note: We don't send this to the CA as we are the CA

Step 5:

Now we use the **CA key** to verify and sign the server certificate. This create the **server.crt** file

Command is: **openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days 360**

```
C:\steve>openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial  
out server.csr -days 360  
Signature ok  
subject=C=UK/ST=Shropshire/L=Ironbridge/O=MYCOMPANY/OU=MTDEPT\x1BID\x1BID\x1BID  
\x1BID\x1BID\x1BID\x1B3~\x1B13~\x1BIC\x1BIC\x1BIC\x1BID\x1BVDEPT/CN=w44/emailAddress=steve@  
testemail.com  
Getting CA Private Key  
Enter pass phrase for ca.key:
```

Step 6:

The above steps created various files. This is what the directory looks like now:

```
C:\steve>dir
Volume in drive C has no label.
Volume Serial Number is E091-351C

Directory of C:\steve

06/11/2016  17:40    <DIR>          .
06/11/2016  17:40    <DIR>          ..
06/11/2016  17:35             1,419 ca.crt
06/11/2016  17:28             1,743 ca.key
06/11/2016  17:40                17 ca.srl
06/11/2016  17:40             1,359 server.crt
06/11/2016  17:39             1,143 server.csr
06/11/2016  17:37             1,675 server.key
               6 File(s)              7,356 bytes
               2 Dir(s)  19,440,615,424 bytes free
```

Note: We don't need to copy the CA.key file. This file is used when creating new server or client certificates.

Step 7:

Copy the files **ca.crt**, **serever.crt** and **server.key** to a folder under the mosquitto folder. I have used a folder called **certs**.

on Linux you should already have a **ca_certificates** folder under **/etc/mosquitto/** and also a **certs** folder.

Use the **ca_certificates** folder for the **CA certificate** and the **certs** folder for the **server certificate and key**.

Step 8:

Copy the **CA certificate file ca.crt** to the client.

Step 9:

Edit the mosquitto.conf file as shown:

```
# Default listener
# =====
# IP address/hostname to bind the default listener to. If not
# given, the default listener will not be bound to a specific
# address and so will be accessible on all network interfaces.
# bind_address ip-address/host name
#bind_address

# Port to use for the default listener.
port 8883

# =====
# Certificate based SSL/TLS support
# =====
# The following options can be used to enable SSL/TLS support for
# this listener. Note that the recommended port for MQTT over TLS
# is 8883, but this must be set manually.
# See also the mosquitto-tls man page.
# At least one of cafile or capath must be defined. They both
# define methods of accessing the PEM encoded Certificate
# Authority certificates that have signed your server certificate
# and that you wish to trust.
# cafile defines the path to a file containing the CA certificates.
# capath defines a directory that will be searched for files
# containing the CA certificates. For capath to work correctly, the
# certificate files must have ".cer" as the file ending. You must run
# "c_rehash <path to capath>" each time you add/remove a certificate.
#capath
cafile c:\mosquitto\certs\ca.crt
keyfile c:\mosquitto\certs\server.key
certfile c:\mosquitto\certs\server.crt
tls_version tlsv1
```

Edit default listener to use port 8883

Don't need capath So no need to do this c_rehash

Basic TLS Support on Mosquitto Broker

Notes:

1. I've used the **default listener** but you could also add an **extra listener**.
2. The **ca path** is not used as I told it the file location instead.

3. On my Linux install the entire **TLS section** of the **mosquitto.conf file** was missing I had to copy it from my windows install and then edit it. Here is the **mosquitto.conf file documentation**

Step 10 -Client Configuration:

Edit the client to tell it to use **TLS** and give it the path of the **CA certificate** file that you copied over.

I'm using the **python client** and the client method is **tls_set()**. Although there are several parameters that you can pass the only one you must give is the **CA file** as shown below.

```
client1.tls_set('c:/python34/steve/MQTT-  
demos/certs/ca.crt').
```

The python client will default to TLSv1.

You should need to change it as the mosquitto broker also defaults to TLSv1.

However to change it to TLSv1.2 use:

```
client1.tls_set('c:/python34/steve/MQTT-  
demos/certs/ca.crt',tls_version=2)
```

The **pub and subscribe scripts** that come with the mosquitto broker default to TLSv1.2.

Problems I Encountered

While creating and working through these procedures i encountered the following problems

1. Error when connecting due to the **common name** on the server certificate not matching.
2. I password protected the server key and the broker couldn't read it. I found this command which will remove the passphrase from the key – **openssl rsa -in server.key -out server-nopass.key.**
3. Not using the correct name for the broker. I used the IP address and not the name that I entered into the certificate. You can use the **tls_insecure_set(True)** option to override name checking as a temporary measure.
4. Authentication errors as I had previously configured my broker to require passwords. Therefore try to start with a clean conf file and beware that the errors you are getting may not be SSL related.

Testing

If all goes well you should be able to publish and subscribe to topics as normal, but now the connection between client and broker is encrypted.

Unfortunately there is no easy way of seeing this.

This is the Python script I used:

```

import paho.mqtt.client as paho
import time
broker="ws4" ← Must match common name
port=8883 on server certificate
conn_flag=False
def on_connect(client, userdata, flags, rc):
    global conn_flag
    conn_flag=True
    print("connected", conn_flag)
    conn_flag=True
def on_log(client, userdata, level, buf):
    print("buffer ",buf)
def on_disconnect(client, userdata, rc):
    print("client disconnected ok")
client1= paho.Client("control1") #create client object
client1.on_log=on_log
client1.tls_set('c:/python34/steve/MQTT-demos/certs/ca.crt')|
client1.on_connect = on_connect
client1.on_disconnect = on_disconnect
client1.connect(broker,port) #establish connection
while not conn_flag:
    time.sleep(1)
    print("waiting", conn_flag)
    client1.loop()
time.sleep(3)
print("publishing")
client1.publish("house/bulb1","The Quick brown fox jumps over the lazy dogs tail")
time.sleep(2)
client1.loop()
time.sleep(2)
client1.disconnect()


```

To test using the mosquitto_pub client use:

```

C:\Python34\steve\mos>mosquitto_pub -h ws4 -t house/bulb1 --cafile
certs/ca.crt -m "test message" -p 8883

```



Mosquitto Configuration Tutorials

- [Installing The Mosquitto broker on Windows and Linux](#)
- [Configuring and Testing MQTT Topic Restrictions](#)
- [Mosquitto username and Password Authentication Configuration Guide](#)
- [Configuring Logging on Mosquitto](#)
- [Mosquitto MQTT Bridge -Usage and Configuration](#)

Other Related Articles and Resources:

- [MQTT for Beginners](#)
- [Beginners guide to PKI](#)

- [Hive MQTT security essentials TLS](#)

Was This Article Helpful? Please Rate..

[Total: 6 Average: 5/5]

6 comments



Kirk Bailey says:

April 3, 2017 at 7:57 pm

I tried using your certificate-generation technique because it's simpler than the one that comes with the Mosquitto test code. If I run the broker using these certs, but when I try to connect a client, I get the error (at the broker) "OpenSSL Error: error:14094416:SSL routines:ssl3_read_bytes:sslv3 alert certificate unknown". The client I used was the "mosquitto_sub" that comes with the package. Everything works if I instead use the certs generated by the cert-generation script that comes with the test code. Any idea why this would be? Do your certs work with your version of mosquitto_sub?

[Reply](#)



steve says:

April 4, 2017 at 3:49 pm

It could be the TLS version that you have configured on the broker try changing it and let me know.

[Reply](#)



Max Kay says:

April 20, 2017 at 9:08 pm

Running into the same issue like 'Kirk Bailey'.
Getin this strange error even though i did everything like
you.
I changed the TLS version to 1.1 and 1.2 but without
success.
If anybody solved this, i would appreacate your help.

Cheers, Max

Reply



steve says:

April 21, 2017 at 7:25 pm

Use the contact form and send me a screenshot of the
broker console. If you send me your certificates and
keys I'll try them on my broker.
I've been trying to reproduce your error. I get a similar
one when using different CA files on client and broker.
Have you copied over the broker CA to the client.
Additionally the mosquiito_sub and pub programs
default to version 1.2. If you use the python client it
defaults to version1.
If you are using the wrong version the consoles log
tells you quite clearly the problem

Reply

cameron says:



May 9, 2017 at 1:52 am

In case this helps, I was running into those same errors as Kirk and Max (I am using mosquitto_pub/mosquitto_sub as the client) I was able to get it working by using the parameter ‘–cafile’ rather than “–capath” and specifying the complete path to the cafile. Below is an example:

```
mosquitto_pub -h host.name -u username -P password -t test/topic -p 8883 --cafile ~/keys/ca.crt -m message
```

Great tutorial Steve, many thanks!

[Reply](#)



steve says:

May 9, 2017 at 10:08 am

Cameron

Glad you found it useful. Tks for the mosquitto_pub I'll add it to the tutorial

Rgds

Steve

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

Post Comment

[Sitemap](#) | [About & Contact](#) | [Disclosure](#)

Copyright © 2011-2017 Steve's internet Guide

By Steve Cope