Comentario técnico: CTC-102
Componente: **MQTT (sobre TLS) con SIMCOM**
Autor: Sergio R. Caprile, Senior R&D Engineer

Revisiones	Fecha	Comentarios
0	14/07/20	

Algunos módulos SIMCOM como el SIM7600 incorporan un cliente MQTT con soporte TLS. Esto nos facilita el conectarnos a proveedores de servicios IoT y mantener una cierta seguridad en el contenido de la información y la autenticidad de quien supuestamente la publica; dado que TLS nos permite por un lado autenticar al servidor, y por otro que la infraestructura del servidor pueda autenticar la identidad de esas conexiones, es decir, que realmente sean usuarios autorizados. Para más información sobre MQTT, consultar el CTC-087. El funcionamiento de TLS lo describimos brevemente en el CTC-092; mientras que en el CTC-093 realizamos un desarrollo similar para el ESP32.

Configuración

El cliente MQTT de SIMCOM, al menos en el SIM7600, puede tener hasta dos instancias; es decir, podemos conectarnos a dos brokers diferentes. Cada instancia puede utilizar o no TLS, y de hacerlo dispone de varios contextos para elegir. Cada contexto almacena las claves y se configura de manera independiente. Quien decide qué tipo de autenticación TLS (simple o doble) utilizamos es el broker, aunque nosotros debemos proveer el certificado cuando es requerido.
Certificados y claves se envían al módulo mediante comandos AT, y permanecen en éste hasta que se apague el módulo.
Cada cliente puede eventualmente ser desconectado y liberado.

Operación

La operación es muy simple, una vez que estamos conectados a la red (`AT+NETOPEN`), crearemos el cliente y nos conectaremos al broker mediante una secuencia de comandos. En el listado hemos puesto en bastardilla los parámetros que dependen de lo que estamos enviando; además utilizamos el cliente número 0 (para utilizar el cliente 1 reemplazamos el '0' por '1'):

`AT+CMQTTSTART`
`AT+CMQTTACCQ=0,"nombre del cliente",`*`servertype`*

servertype: 0 = TCP, 1 = TLS , por lo que utilizaremos 0 en este caso

En caso que tengamos que configurar una “última voluntad”[1], lo hacemos enviando:

`AT+CMQTTWILLTOPIC=0,`*`len`*

len: longitud del tópico, a continuación recibiremos un prompt y enviaremos el tópico

`AT+CMQTTWILLMSG=0,`*`len`*`,`*`qos`*

len: longitud del mensaje, a continuación recibiremos un prompt y enviaremos el mensaje

qos: calidad de servicio a utilizar cuando se envíe el mensaje (cuando nos desconectemos)

Finalmente nos conectamos:

`AT+CMQTTCONNECT=0,"tcp://`*`host`*`:`*`port`*`",`*`keepalive`*`,`*`cleansession`*`[,"user"[,"pass"]]`

1 Se refiere al mensaje *last will* que es enviado por el broker en nuestro nombre cuando éste detecta una desconexión (cierre de la conexión TCP o expiración del tiempo entre mensajes *keepalive*)

cleansession: 1; 0 significa que no queremos una sesión que arranque en limpio

user y pass son opcionales (por eso los indicamos entre corchetes) y tienen una longitud máxima de 256 bytes cada uno

port puede omitirse, en cuyo caso es 1883 por defecto

A partir de aquí podemos publicar y/o suscribirnos.

Eventualmente nos desconectaremos y liberaremos el cliente:

```
AT+CMQTTDISC=0,timeout
```

timeout: tiempo máximo para esperar respuesta del broker antes de informar que falló la operación

```
AT+CMQTTREL=0
AT+CMQTTSTOP
```

Publicación

Para publicar un mensaje en un tópico, disponemos de la siguiente secuencia de comandos:

```
AT+CMQTTTOPIC=0,len
```

len: longitud del tópico, a continuación recibiremos un prompt y enviaremos el tópico (<= 1024)

```
AT+CMQTTPAYLOAD=0,len
```

len: longitud del mensaje, a continuación recibiremos un prompt y enviaremos el mensaje (<= 10240)

```
AT+CMQTTPUB=0,qos,timeout
```

qos: calidad de servicio a utilizar

timeout: tiempo máximo para esperar confirmación del broker antes de informar que falló la operación

Suscripción

Para suscribirnos a un tópico, disponemos de la siguiente secuencia de comandos:

```
AT+CMQTTSUBTOPIC=0,len,qos
```

len: longitud del tópico, a continuación recibiremos un prompt y enviaremos el tópico (<= 1024)

qos: calidad de servicio a utilizar cuando se nos entreguen los mensajes

```
AT+CMQTTSUB=0
```

Cuando alguien publique en ese tópico, recibiremos indicaciones no solicitadas conteniendo el nombre del tópico (máximo 1024 bytes) y el contenido del mensaje (máximo 10240 bytes):

```
+CMQTTRXSTART: 0,topiclen,msglen
+CMQTTRXTOPIC: 0,len
topictext
+CMQTTRXPAYLOAD: 0,len
msgtext
+CMQTTRXEND: 0
```

La aparente redundancia se debe a que de acuerdo a la longitud del tópico y/o el mensaje, pueden partirse en varias indicaciones, cada una con su longitud, de modo que la longitud total del campo corresponde a la suma de éstas, indicada en el mensaje inicial.

Operación con TLS

Manteniendo la nomenclatura y recomendaciones, la siguiente es la secuencia de operación con TLS.

Como primer paso deberemos configurar el contexto SSL[2]. Como mencionamos, existen varios contextos disponibles, en este caso utilizaremos el 0.

Disponemos de comandos para enviar un certificado o clave:

2 Algunos proveedores utilizan el nombre SSL, recordemos que hablamos de lo mismo y que TLS deriva de SSL.

```
AT+CCERTDOWN="filename",len
```

len: longitud del certificado, a continuación recibiremos un prompt y lo enviaremos
para listarlos

```
AT+CCERTLIST
```

para eventualmente borrarlos

```
AT+CCERTDELE="filename"
```

Con un comando adicional indicamos el tipo de autenticación:

```
AT+CSSLCFG="authmode",0,mode
```

mode: 1 = simple, autentica al server, 2 = dual
con otro asociamos el nombre de archivo que elegimos con la función que realiza

```
AT+CSSLCFG=function,0,"filename"
```

Más adelante lo describimos en detalle para cada opción de operación.
El procedimiento continúa de forma muy similar que para la operación sin TLS:

```
AT+CMQTTSTART
AT+CMQTTACCQ=0,"nombre del cliente",servertype
```

servertype: 0 = TCP, 1 = TLS , por lo que utilizaremos 1 en este caso

```
AT+CMQTTSSLCFG=0,ctxidx
```

ctxidx : el número de contexto SSL que usamos; por ejemplo, como indicamos al inicio, hemos configurado el 0 así que en este caso usamos el 0
En caso que tengamos que configurar un mensaje “de última voluntad”, lo hacemos como indicamos para la operación sin TLS.
El comando de conexión es el mismo, sólo que debemos indicar el port, que usualmente es 8883:

```
AT+CMQTTCONNECT=0,"tcp://host:port",keepalive,cleansession[,"user"[,"pass"]]
```

Ya estamos conectados, la operación de publicar y suscribirse no cambia.

Simple autenticación

De esta forma autenticamos al servidor, es decir, podemos confiar en que es el que esperamos que sea, pero el servidor no tiene idea de quién podemos llegar a ser nosotros.

```
AT+CSSLCFG="authmode",0,1
AT+CSSLCFG="cacert",0,"filename"
```

Doble autenticación

De esta forma autenticamos tanto al servidor como al cliente, es decir, ahora el servidor sabe que deberíamos ser quien nuestro certificado dice que somos.

```
AT+CSSLCFG="authmode",0,2
AT+CSSLCFG="cacert",0,"filename"
AT+CSSLCFG="clientcert",0,"filename2"
AT+CSSLCFG="clientkey",0,"filename3"
```

Ejemplos

En los archivos provistos incluimos un par de scripts para Docklight Scripting[3], junto con un archivo de comandos AT que también puede utilizarse en la versión sin scripting. Proveemos además certificados y claves para facilitar las pruebas.[4]

- ssl1 contiene la secuencia necesaria para cargar el certificado de la CA
- ssl2 contiene la secuencia necesaria para cargar certificado y clave del cliente

Conexión sin TLS

- Enviamos las secuencias desde “MQTT start” hasta la anterior a “Publish ” (ésta no)
  - como no vamos a utilizar TLS, enviamos las que dicen “sin TLS”
- Continuamos con el proceso de operación, publicando o suscribiéndonos a algún tópico
- Eventualmente, cuando no utilizamos más el cliente, nos desconectamos y liberamos el cliente enviando las secuencias “MQTT Disconnect”, “MQTT release”, y “MQTT stop”

En las capturas siguientes, resaltamos lo que enviamos:

```
AT+CMQTTSTART
+CMQTTSTART: 0
OK
AT+CMQTTACCQ=0,"myMQTTclient",0
OK
AT+CMQTTCONNECT=0,"tcp://mibrokermqtt.midominio",60,1
OK
+CMQTTCONNECT: 0,0
```

operamos, y eventualmente nos desconectamos:

```
AT+CMQTTDISC=0,60
OK
+CMQTTDISC: 0,0
AT+CMQTTREL=0
OK
AT+CMQTTSTOP
+CMQTTSTOP: 0
OK
```

Operación: publicar un mensaje

- Conectamos algún cliente (mosquitto_sub) al broker y nos suscribimos al tópico en el que enviaremos (/test/1)[5]
- Enviamos la serie de secuencias de publicación completa, es necesario enviar el tópico y el payload

```
AT+CMQTTTOPIC=0,7
>
/test/1
OK
AT+CMQTTPAYLOAD=0,4
>
HoLa
OK
AT+CMQTTPUB=0,1,60
OK
+CMQTTPUB: 0,0
```

Observaremos en la ventana del cliente el texto del mensaje (HoLa)

3 https://docklight.de/information/#docklight-scripting
4 Los scripts cargan los archivos desde el mismo directorio desde donde se los invoca, de modo que no es necesario hacer malabares con operaciones de transferencia ni copiar/pegar. Los certificados son idénticos a los entregados junto con el CTC-093.
5 `$ mosquitto_sub -h mibrokermqtt.midominio -t /test/1`

Operación: suscribirse a un tópico

- Enviamos la serie de secuencias de suscripción
- Con algún cliente (mosquitto_pub) publicamos en el broker un mensaje al tópico en que nos suscribimos (/SIM/7600)
- Observamos la indicación de recepción del mensaje quenos entrega el módulo

```
AT+CMQTTSUBTOPIC=0,9,1
>
/SIM/7600
OK
AT+CMQTTSUB=0
OK
+CMQTTSUB: 0,0
```

desde un cliente (mosquitto_pub[6]) publicamos en ese tópico:

```
+CMQTTRXSTART: 0,9,22
+CMQTTRXTOPIC: 0,9
/SIM/7600
+CMQTTRXPAYLOAD: 0,22
Sigue al conejo blanco
+CMQTTRXEND: 0
```

Conexión con TLS

- Como primer paso deberemos configurar el contexto SSL. Esto lo haremos mediante los scripts y lo describimos más adelante para autenticación simple y doble, respectivamente.
- Comenzamos enviando las secuencias desde “MQTT start” hasta la anterior a “Publish ” (ésta no) igual que como se describe para la operación sin TLS
  - como utilizamos TLS, enviaremos las que dicen “con TLS”
- Continuamos con el proceso igual que como se describe para la operación sin TLS

TLS Simple autenticación (Autenticación del broker)

- Ejecutamos el script ssl1, enviamos la secuencia “SSL single”
- Continuamos con el proceso de conexión y operación

El texto en bastardilla es un indicador que hemos incluido en el script

```
 SSL: CA CERTIFICATE LOADING
AT+CCERTDOWN="ca.pem",1631
>
-----BEGIN CERTIFICATE-----
[...]
-----END CERTIFICATE-----

OK
AT+CCERTLIST
+CCERTLIST: "ca.pem"
OK
AT+CSSLCFG="cacert",0,"ca.pem"
OK
 SSL: CA CERTIFICATE LOADED

AT+CSSLCFG="authmode",0,1
OK

AT+CMQTTSTART
```

6 `$ mosquitto_pub -h mibrokermqtt.midominio -t /SIM/7600 -m "Sigue al conejo blanco"`

```
+CMQTTSTART: 0
OK
AT+CMQTTACCQ=0,"myMQTTTLSclient",1
OK
AT+CMQTTSSLCFG=0,0
OK
AT+CMQTTCONNECT=0,"tcp://mibrokermqtt.midominio:8883",60,1
OK
+CMQTTCONNECT: 0,0
```

TLS Doble autenticación (Autenticación de broker y clientes)

- Si no ejecutamos el script ssl1 aún, debemos hacerlo, caso contrario el certificado de la CA ya está almacenado.
- Ejecutamos el script ssl2, enviamos la secuencia “SSL dual”
- Continuamos con el proceso de conexión y operación

El texto en bastardilla es un indicador que hemos incluido en el script

```
 SSL: CA CERTIFICATE LOADING
AT+CCERTDOWN="ca.pem",1631
>
-----BEGIN CERTIFICATE-----
[...]
-----END CERTIFICATE-----

OK
AT+CCERTLIST
+CCERTLIST: "ca.pem"
OK
AT+CSSLCFG="cacert",0,"ca.pem"
OK
 SSL: CA CERTIFICATE LOADED

 SSL: CLIENT CREDENTIALS LOADING
AT+CCERTDOWN="ccert.pem",1310
>
-----BEGIN CERTIFICATE-----
[...]
-----END CERTIFICATE-----
OK
AT+CCERTDOWN="ckey.pem",1675
>
-----BEGIN RSA PRIVATE KEY-----
[...]
-----END RSA PRIVATE KEY-----
OK
AT+CCERTLIST
+CCERTLIST: "ca.pem"
+CCERTLIST: "ccert.pem"
+CCERTLIST: "ckey.pem"
OK
AT+CSSLCFG="clientcert",0,"ccert.pem"
OK
AT+CSSLCFG="clientkey",0,"ckey.pem"
OK
 SSL: CLIENT CREDENTIALS LOADED

AT+CSSLCFG="authmode",0,2
OK
```

Brokers

Repetimos aquí el contenido publicado en el CTC-093 a este respecto, para comodidad del lector.[7]
Para las pruebas hemos usado Mosquitto, el cual configuramos manualmente para cada esquema. Entre algunas de las opciones disponibles en la nube que mencionamos en el CTC-092, comprobamos que tanto Eclipse como CloudMQTT soportan autenticación simple (del broker) por TLS[8].

Mosquitto

Detallamos, a modo instructivo y como ayuda rápida, las configuraciones mínimas necesarias para operar en Mosquitto. Los paths están en formato GNU/Linux y se espera que pongamos nuestros certificados allí.

Simple autenticación (Autenticación del broker)

```
listener 8883 192.168.5.1
log_dest syslog
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
```

Doble autenticación (Autenticación de broker y clientes)

```
listener 8883 192.168.5.1
log_dest syslog
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
require_certificate true
```

7 Ante posibles diferencias, ese CTC tiene precedencia.
8 Algunos proveedores utilizan el nombre SSL, recordemos que hablamos de lo mismo y que TLS deriva de SSL.