Práctica 6

Objetivo
El alumno se familiarizará con el protocolo MQTT con TLS para el envío de datos contextuales del entorno IoT a diferentes dispositivos, usando el sistema embebido ESP32 DevKit v1, con seguridad por medio de autenticar el servidor y los clientes MQTT. Con el fin de desarrollar aplicaciones IoT basadas en microcontrolador para aplicarlas en la resolución de problemas de cómputo, de una manera eficaz y responsable.

Equipo
Computadora personal con conexión a Internet.
Visual Studio Code con ESP-IDF Development Framework.

Teoría
- Describa a detalle el formato PEM.
- ¿Qué versiones del protocolo TLS soporta el ESP32?

Desarrollo

Parte 1

A) Crear la llave privada y certificado para el cliente

1. Abra una terminal o Git Bash y direccione a la carpeta *C:\certificados*.
2. Cree una llave de 2,048 bits para un dispositivo IoT (denominado *device01*) por medio de ingresar el siguiente comando:

   ```
   openssl genrsa -out device01.key 2048
   ```

3. Ejecute el siguiente comando para generar una solicitud de firma de certificado CSR. El comando utiliza la llave privada de 2,048 bits creada previamente y genera un archivo *device01.csr*:

   ```
   openssl req -new -key device01.key -out device01.csr
   ```

   OpenSSL va a solicitar información para incorporar en el certificado. Ingrese un . en las preguntas que no quiera contestar. En **Common Name** ingrese el nombre del dispositivo, por ejemplo *MQTT CLIENT 01*.

5. Ejecute el siguiente comando para firmar la solicitud de firma de certificado creada anteriormente, es decir, el archivo *device01.csr*. El siguiente comando también utiliza el certificado digital X.509 auto-firmado de la autoridad certificadora y su llave privada: los archivos *ca.crt* y *ca.key*. El comando genera un archivo *device01.crt* con el certificado digital X.509 firmado para el cliente MQTT. La opción *-addTrust clientAuth* indica que queremos usar el certificado para autenticar a un cliente:

```
openssl x509 -req -in device01.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out device01.crt -days 3650 -sha256 -addtrust
clientAuth
```

Después de ejecutar los comandos, se generan los siguientes archivos:

device01.key: Llave del cliente.

device01.csr: Petición de firmado del certificado del cliente.

device01.crt: Certificado del cliente.

Tendremos que proporcionar los siguientes tres archivos a cualquier dispositivo que queramos conectar al servidor Mosquitto:

device01.key: Llave del cliente.

device01.crt: Certificado del cliente.

ca.crt: Certificado de la autoridad certificadora.

Nunca proporcione archivos adicionales a los dispositivos. No desea que los dispositivos puedan generar certificados. Sólo desea que se autentiquen con un certificado válido.

B) Configure el uso de TLS en Mosquitto para autenticar a los clientes

6. Cree una carpeta *certificado_cliente* en el directorio de su preferencia.
7. Copie a esta carpeta los siguientes tres archivo:

   device01.key

   device01.crt

   ca.crt

8. Detenga el servicio Mosquitto en caso de que esté ejecutándose.
9. Haga una copia de respaldo del archivo de configuración *mosquitto.conf* antes de realizar las siguientes modificaciones.
10. Modifique la siguiente línea del archivo *mosquitto.conf* para que Mosquitto requiera un certificado de cliente válido para cualquier cliente que solicite una conexión:

    ```
    require_certificate true
    ```

11. Inicie el servicio Mosquitto.

12. Ingrese el siguiente comando para suscribirse al tema *sensors/+/altitude*, en el cual se especifica el certificado del cliente. Tiene que indicar en `ca.crt`, `device01.crt` y `device01.key` la ruta completa de los archivos de acuerdo al paso 7.

    ```
    mosquitto_sub -p 8883 --cafile ca.crt --cert device01.crt --key
    device01.key -t sensors/+/altitude -d
    ```

13. Ingrese el siguiente comando para generar un cliente MQTT que publica un mensaje en el tema *sensors/drone12/altitud*e, por medio de una conexión cifrada. Recuerde indicar la ruta completa de los archivos `ca.crt`, `device01.crt` y `device01.key` de acuerdo al paso 7.

    ```
    mosquitto_pub -p 8883 --cafile ca.crt --cert device01.crt --key
    device01.key -t sensors/drone12/altitude -m "1350 m" -d
    ```

    Incluya en su reporte capturas de pantalla del envío y recepción de los mensajes publicados.

Uso de MQTT-spy

14. Ejecute MQTT-spy.
15. Realice las configuraciones necesarias en MQTT-spy para que sea posible publicar mensajes usando TLS con autenticación de los clientes. Incluya capturas de pantalla que muestren cada una de las configuraciones.
16. Cree una conexión en MQTT-spy y publique un mensaje que contenga *"170 m*" a los clientes suscritos al tema *sensors/+/altitude*. Incluya una captura de pantalla de la publicación del mensaje en MQTT-spy y una captura de pantalla de la recepción del mensaje en la terminal por parte del suscriptor que se generó en el paso 12.

Parte 2

1. Cree en Visual Studio Code el proyecto P6_pt2 basándose en el código de la Práctica 5. Realice las modificaciones necesarias para que el ESP32 se conecte al broker Mosquitto de su PC usando TLS con autenticación del cliente y publique cada 2 segundos el estado de un sensor. Cree un cliente MQTT en su PC que se suscriba a la información.

   Agregue el código para que el ESP32 se suscriba al tema *sensors/drone05/ReqStatistics* y cuando reciba el mensaje "ReqData" publique la media, mediana y varianza de las últimas 5 mediciones del sensor en formato CSV: *<media>,<mediana>,*<varianza > en el tema *sensors/drone05/Statistics*.

   Cree un cliente MQTT en su PC que publique en *sensors/drone05/ReqStatistics* y se suscriba a *sensors/drone05/Statistics*.

2. Adjunte a su reporte capturas de pantalla de la ejecución de los pasos anteriores.

Conclusiones y comentarios
Dificultades en el desarrollo
Referencias