

# CASO PRÁCTICO: SEGURIDAD EN IOT MONTAJE Y SECURIZACIÓN DE UN BROKER MQTT

Alumno: Gengis Rovi

## 1. PLANTEAMIENTO DEL ÁRBOL JERÁRQUICO

Para organizar las comunicaciones del hogar conectado, se ha diseñado una estructura de tópicos MQTT basada en una jerarquía lógica de ubicación y funcionalidad. Esta estructura facilita la escalabilidad del sistema y la aplicación de políticas de seguridad granulares.

Se ha definido la estructura base como: casa/{habitacion}/{dispositivo}/{funcion}.

Se distinguen dos tipos de flujos de información:

1. **Estado:** Información enviada por los sensores hacia el panel (publicación del dispositivo).
2. **Comando:** Órdenes enviadas por el panel hacia los actuadores (suscripción del dispositivo).

A continuación, se presenta la representación gráfica de la jerarquía propuesta:

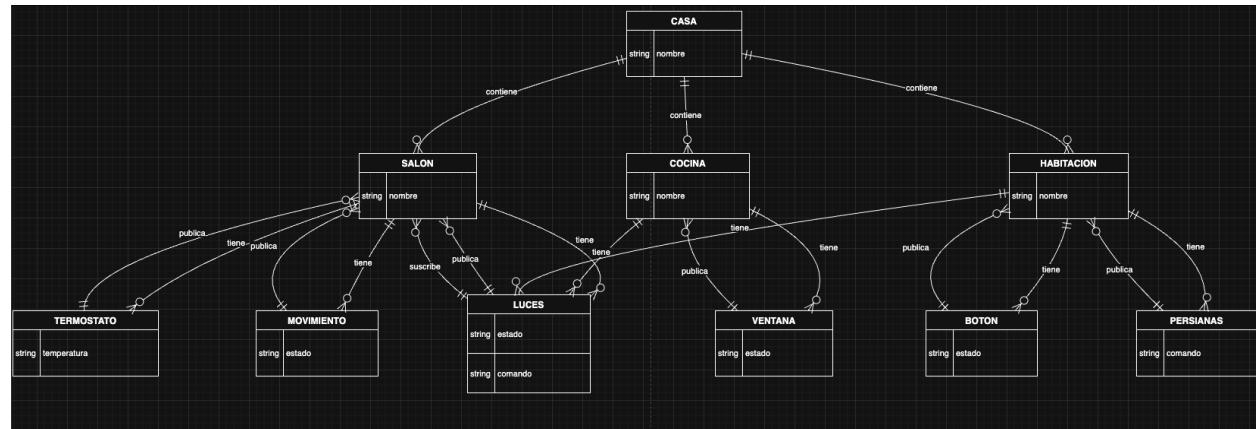


Figura 1. Árbol jerárquico de tópicos MQTT para el hogar conectado.

El detalle de los tópicos implementados es el siguiente:

- **Salón:**
  - casa/salon/termostato/temperatura (Lectura del sensor)
  - casa/salon/termostato/comando (Actuador: encender/apagar)
  - casa/salon/movimiento/estado (Lectura de presencia)
  - casa/salon/luces/comando (Actuador: luces)
- **Cocina:**

- casa/cocina/ventana/estado (Lectura de seguridad)
  - casa/cocina/luces/comando (Actuador: luces)
  - **Habitación:**
    - casa/habitacion/boton/estado (Accionador manual)
    - casa/habitacion/persianas/comando (Actuador: motor)
    - casa/habitacion/luces/comando (Actuador: luces)
- 

## 2. INSTALACIÓN DEL BROKER MQTT

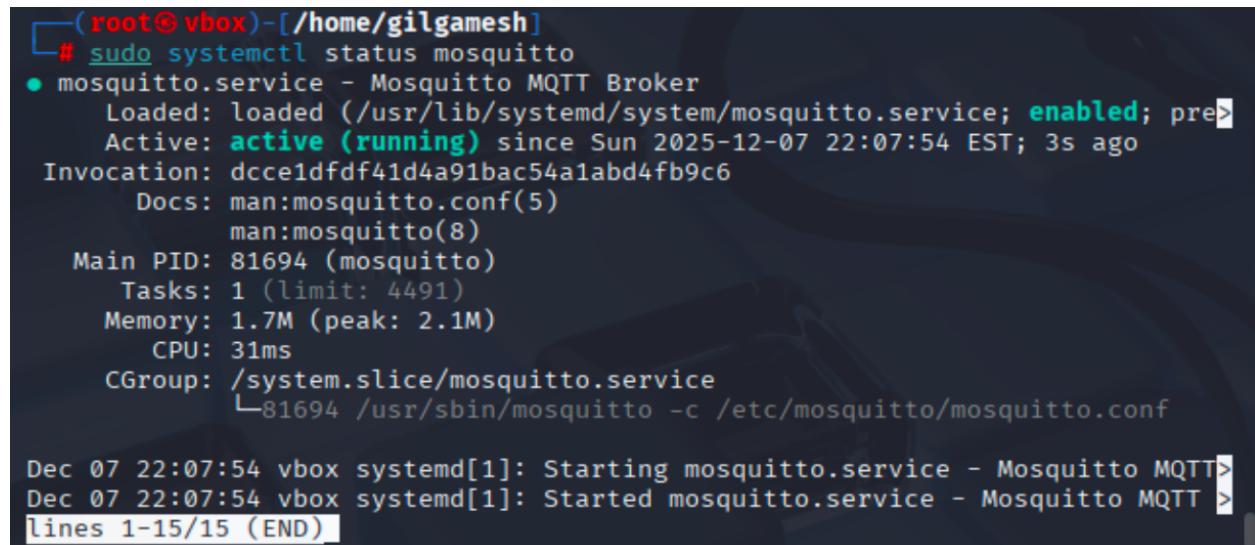
Como núcleo central de las comunicaciones (nodo central), se ha seleccionado **Eclipse Mosquitto** debido a que es un broker de código abierto, ligero y cumple con el estándar MQTT versiones 3.1 y 3.1.1.

El procedimiento se ha realizado sobre un sistema operativo Linux. Se han utilizado los siguientes comandos para la instalación del servicio y las herramientas de cliente necesarias para la simulación posterior:

Bash

```
sudo apt-get update
sudo apt-get install mosquitto mosquitto-clients
```

Tras la instalación, se ha verificado que el servicio se encuentra activo y escuchando en el puerto por defecto (1883).



```
(root@vbox)-[~/home/gilgamesh]
# sudo systemctl status mosquitto
● mosquitto.service - Mosquitto MQTT Broker
   Loaded: loaded (/usr/lib/systemd/system/mosquitto.service; enabled; pre>
   Active: active (running) since Sun 2025-12-07 22:07:54 EST; 3s ago
     Invocation: dcce1dfdf41d4a91bac54a1abd4fb9c6
       Docs: man:mosquitto.conf(5)
              man:mosquitto(8)
     Main PID: 81694 (mosquitto)
        Tasks: 1 (limit: 4491)
      Memory: 1.7M (peak: 2.1M)
        CPU: 31ms
      CGroup: /system.slice/mosquitto.service
              └─81694 /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf

Dec 07 22:07:54 vbox systemd[1]: Starting mosquitto.service - Mosquitto MQTT>
Dec 07 22:07:54 vbox systemd[1]: Started mosquitto.service - Mosquitto MQTT >
[lines 1-15/15 (END)]
```

Figura 2. Verificación del servicio Mosquitto activo.

---

## 3. CREACIÓN DE USUARIOS Y CONTROL DE ACCESOS

Por defecto, Mosquitto permite conexiones anónimas, lo cual representa un riesgo de seguridad crítico en un entorno IoT. Para mitigar esto, se ha procedido a:

1. Deshabilitar el acceso anónimo en el fichero de configuración `mosquitto.conf`.
2. Crear un fichero de contraseñas cifradas para la autenticación.

Se han creado usuarios específicos para segmentar la seguridad: un usuario administrador para el **Panel Principal** y usuarios individuales para cada zona de la casa, limitando el impacto en caso de que un dispositivo sea comprometido.

### Comandos utilizados:

Para crear el archivo y el primer usuario (Panel):

Bash

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd panel_admin
```

(El parámetro `-c` crea un nuevo archivo, sobrescribiendo si existe).

Para añadir los usuarios de los dispositivos:

Bash

```
sudo mosquitto_passwd -b /etc/mosquitto/passwd disp_salon [contraseña]
sudo mosquitto_passwd -b /etc/mosquitto/passwd disp_cocina [contraseña]
sudo mosquitto_passwd -b /etc/mosquitto/passwd disp_habitacion [contraseña]
```

(El parámetro `-b` permite añadir usuarios en modo batch a un archivo existente).

The screenshot shows a terminal session on a Linux system (Ubuntu) with root privileges. The user is creating a new password file for the Mosquitto broker. They first run `sudo mosquitto_passwd -c /etc/mosquitto/passwd panel_admin` to create a file for the main panel user. Then, they add three new users ('disp\_salon', 'disp\_cocina', and 'disp\_habitacion') to the same password file using the `-b` option. Finally, they log out of the root session.

```
[root@vbox]# sudo mosquitto_passwd -c /etc/mosquitto/passwd panel_admin
Password:
Reenter password:

[root@vbox]# sudo mosquitto_passwd -b /etc/mosquitto/passwd disp_salon secreto_salon
[root@vbox]# sudo mosquitto_passwd -b /etc/mosquitto/passwd disp_cocina secreto_cocina
[root@vbox]# sudo mosquitto_passwd -b /etc/mosquitto/passwd disp_habitacion secreto_habitacion
[root@vbox]#
```

Figura 3. Creación de usuarios y fichero de contraseñas.

**Configuración en `mosquitto.conf`:** Se ha editado el fichero de configuración para obligar el uso de este archivo de contraseñas:

Fragmento de código

```
allow_anonymous false
password_file /etc/mosquitto/passwd
```

---

## 4. SECURIZACIÓN MEDIANTE LISTAS DE CONTROL DE ACCESO (ACL)

Para cumplir con el requisito de que **solo el Panel Principal** tenga visibilidad completa y control sobre toda la casa, y que los dispositivos estén restringidos únicamente a sus funciones, se ha implementado un archivo de Listas de Control de Acceso (ACL).

Se ha creado el archivo `/etc/mosquitto/aclfile` con las siguientes reglas:

Fragmento de código

```
# --- PANEL PRINCIPAL ---
# Permiso total de lectura y escritura en toda la jerarquía
user panel_admin
topic readwrite casa/#

# --- SALÓN ---
user disp_salon
# Solo puede publicar sus estados
topic write casa/salon/+/estado
topic write casa/salon/termostato/temperatura
# Solo puede leer comandos dirigidos a él
topic read casa/salon/+/comando

# --- COCINA ---
user disp_cocina
topic write casa/cocina/ventana/estado
topic read casa/cocina/luces/comando

# --- HABITACIÓN ---
user disp_habitacion
topic write casa/habitacion/+/estado
topic read casa/habitacion/+/comando
```

### Justificación de las reglas:

- Se utiliza el comodín # (multinivel) para que el Panel tenga acceso a todo el árbol casa/.
- Se utiliza el comodín + (un nivel) para simplificar las reglas de los dispositivos que tienen múltiples funciones (ej. luces y persianas) bajo la misma habitación.
- Se separa `read` (leer órdenes) de `write` (reportar datos) para evitar que un sensor envíe comandos falsos a otros actuadores.

```

File Actions Edit View Help
GNU nano 8.7          /etc/mosquitto/aclfile *
# — PANEL PRINCIPAL (Admin) —
# Tiene permiso total de lectura y escritura en toda la casa
user panel_admin
topic readwrite casa/#

# — SALON —
user disp_salon
# Solo puede publicar (escribir) sus estados
topic write casa/salon/+/estado
topic write casa/salon/termostato/temperatura
# Solo puede leer (suscribirse) comandos dirigidos a él
topic read casa/salon/+/comando

# — COCINA —
user disp_salon
# Solo puede publicar (escribir) sus estados
topic write casa/cocina/ventana/estado
topic read casa/cocina/luces/comando

# — HABITACION —
user disp_habitacion
topic write casa/habitacion/+/estado
topic read casa/habitacion/+/comando

^G Help      ^O Write Out   ^F Where Is   ^K Cut       ^T Execute
^X Exit      ^R Read File   ^\ Replace    ^U Paste     ^J Justify

```

Figura 4. Configuración de las Listas de Control de Acceso (ACL).

---

## 5. CAPA DE TRANSPORTE SEGURO SSL/TLS

Para proteger la confidencialidad de los datos y las credenciales de autenticación frente a escuchas en la red, se ha configurado el cifrado SSL/TLS en el puerto 8883.

### Procedimiento de generación de certificados (OpenSSL):

- Creación de la Autoridad Certificadora (CA):** openssl req -new -x509 -days 365 -extensions v3\_ca -keyout ca.key -out ca.crt Esto genera la raíz de confianza para firmar los certificados.
- Creación de la clave y solicitud del servidor:** openssl genrsa -out server.key 2048

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

### 3. Firma del certificado del servidor:

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out server.crt -days 365.
```

```

(root@vbox)-[/etc/mosquitto/certs]
# ls -l /etc/mosquitto/certs/
total 28
-rw-r--r-- 1 root root 1298 Dec  7 23:02 ca.crt
-rw----- 1 root root 1886 Dec  7 23:01 ca.key
-rw-r--r-- 1 root root  41 Dec  7 23:21 ca.srl
-rw-r--r-- 1 root root 130 Mar 21 2025 README
-rw-r--r-- 1 root root 1281 Dec  7 23:21 server.crt
-rw-r--r-- 1 root root 1025 Dec  7 23:05 server.csr
-rw----- 1 root root 1704 Dec  7 23:03 server.key

```

Figura 5. Certificados generados para el cifrado SSL.

Finalmente, se ha actualizado el archivo `mosquitto.conf` para habilitar el listener seguro:

#### Fragmento de código

```
listener 8883
cafile /etc/mosquitto/ca_certificates/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
require_certificate false
```

## 6. SIMULACIÓN Y VERIFICACIÓN

Para verificar el correcto funcionamiento de la arquitectura y la seguridad implementada, se han utilizado las herramientas de línea de comandos `mosquitto_pub` y `mosquitto_sub`.

### Prueba 1: El Panel Principal recibe datos

Simulamos que el **Panel** se suscribe a toda la casa (#) y el **Termostato del Salón** envía la temperatura.

- **Panel (Suscripción):** `mosquitto_sub -h localhost -p 8883 --cafile ca.crt -u panel_admin -P [pass] -t "casa/#" -v`
- **Termostato (Publicación):** `mosquitto_pub -h localhost -p 8883 --cafile ca.crt -u disp_salon -P [pass] -t "casa/salon/termostato/temperatura" -m "24.5"`

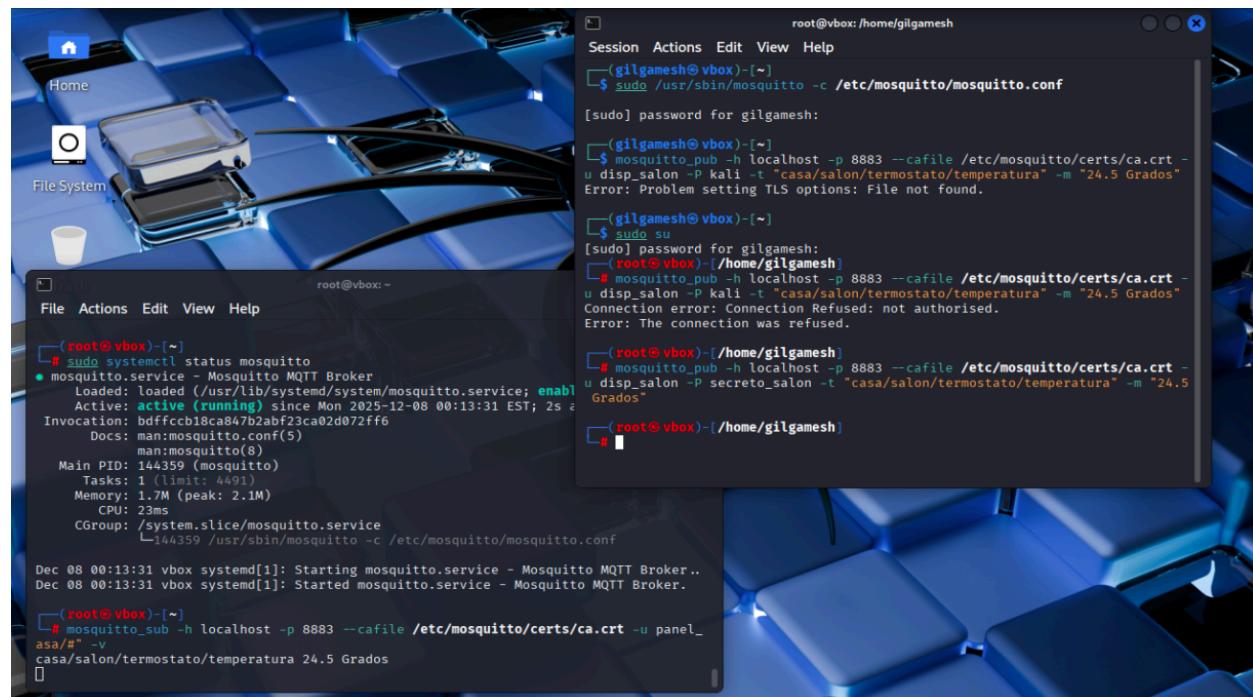


Figura 6. Verificación de recepción de datos por el Panel Principal.

## Prueba 2: Control de dispositivos (Panel a Cocina)

Simulamos que el **Panel** envía la orden de encender la luz a la **Cocina**.

- **Cocina (Suscripción a comandos):** mosquitto\_sub -h localhost -p 8883 --cafile ca.crt -u disp\_cocina -P [pass] -t "casa/cocina/luces/comando"
- **Panel (Envío de orden):** mosquitto\_pub -h localhost -p 8883 --cafile ca.crt -u panel\_admin -P [pass] -t "casa/cocina/luces/comando" -m "ON"

The screenshot shows a Kali Linux desktop environment with several open terminal windows. The desktop background features a blue metallic keyboard theme. One terminal window shows the root shell on the host system, while others show the root shell on a virtual machine named 'vbox'. The terminals are used to run MQTT commands using the mosquitto\_pub and mosquitto\_sub tools over port 8883, secured by certificates located in /etc/mosquitto/certs/ca.crt. The messages being exchanged are related to controlling lights ('luces') and a temperature sensor ('termostato/temperatura'). The 'panel\_admin' user on the host system is publishing 'ON' messages to the 'casa/cocina/luces/comando' topic, which is then subscribed to by the 'disp\_cocina' user on the host system, resulting in the message 'casa/#' being received.

```
root@vbox:~# mosquitto_pub -h localhost -p 8883 --cafile /etc/mosquitto/certs/ca.crt -u disp_cocina -P secreto_cocina -t "casa/cocina/luces/comando" -m "ON"
root@vbox:~# mosquitto_sub -h localhost -p 8883 --cafile /etc/mosquitto/certs/ca.crt -u disp_cocina -P secreto_cocina -t "casa/cocina/luces/comando" -t "casa/#"
casa/#
```

Figura 7. Verificación de control de actuadores desde el Panel.

## Prueba 3: Verificación de Seguridad (ACL)

Simulamos un intento de violación de permisos: El dispositivo de la **Habitación** intenta publicar una orden falsa para apagar las luces del salón.

- **Comando atacante (Habitación):** mosquitto\_pub -h localhost -p 8883 --cafile ca.crt -u disp\_habitacion -P [pass] -t "casa/salon/luces/comando" -m "OFF"

**Resultado:** El broker acepta la conexión pero rechaza silenciosamente la publicación o devuelve error (dependiendo de la versión), ya que el usuario `disp_habitacion` no tiene permiso de escritura en `casa/salon`. El Panel o el Salón nunca reciben este mensaje.

```

root@vbox: /home/gilgamesh
Session Actions Edit View Help
( root@vbox )-[ /home/gilgamesh ]
# mosquitto_pub -h localhost -p 8883 -u panel_admin -P kali --cafile /etc/mosquitto/certs/ca.crt -t "casa/cochina/luces/comando" -m "ON"
( root@vbox )-[ /home/gilgamesh ]
# mosquitto_pub -h localhost -p 8883 --cafile /etc/mosquitto/certs/ca.crt -u disp_habitacion -P secreto_habitacion -t "casa/salon/luces/comando" -m "OFF"
( root@vbox )-[ /home/gilgamesh ]
# mosquitto_pub -h localhost -p 8883 --cafile /etc/mosquitto/certs/ca.crt -u disp_habitacion -P secreto_habitacion -t "casa/salon/luces/comando" -m "OFF"
( root@vbox )-[ /home/gilgamesh ]
# mosquitto_pub -h localhost -p 8883 --cafile /etc/mosquitto/certs/ca.crt -u disp_habitacion -P secreto_habitacion -t "casa/salon/luces/comando" -m "OFF"
Error: Connection refused
( root@vbox )-[ /home/gilgamesh ]
# mosquitto_pub -h localhost -p 8883 --cafile /etc/mosquitto/certs/ca.crt -u disp_habitacion -P secreto_habitacion -t "casa/salon/luces/comando" -m "OFF"

File Actions Edit View Help
6.767s wall clock time, 2.5M memory peak.
^C
( root@vbox )-[ ~ ]
# sudo systemctl start mosquitto
( root@vbox )-[ ~ ]
# sudo journalctl -u mosquitto -f
Dec 08 00:08:37 vbox systemd[1]: mosquitto.service: Failed with result 'start-limit-reached'.
Dec 08 00:08:37 vbox systemd[1]: Failed to start mosquitto.service.
Dec 08 00:13:31 vbox systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker...
Dec 08 00:13:31 vbox systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.
Dec 08 01:13:37 vbox systemd[1]: Stopping mosquitto.service - Mosquitto MQTT Broker...
Dec 08 01:13:37 vbox systemd[1]: mosquitto.service: Deactivated successfully.
Dec 08 01:13:37 vbox systemd[1]: Stopped mosquitto.service - Mosquitto MQTT Broker.
Dec 08 01:13:37 vbox systemd[1]: mosquitto.service: Consumed 6.344s CPU time over 1h
6.767s wall clock time, 2.5M memory peak.
Dec 08 01:17:05 vbox systemd[1]: Starting mosquitto.service - Mosquitto MQTT Broker...
Dec 08 01:17:05 vbox systemd[1]: Started mosquitto.service - Mosquitto MQTT Broker.

```

Figura 8. Verificación del bloqueo por ACL ante intento de acceso no autorizado.

## 7. CONCLUSIONES

A través de este caso práctico se ha logrado desplegar una infraestructura IoT segura basada en MQTT. Se han aplicado las siguientes capas de seguridad fundamentales en entornos industriales y domésticos:

- Autenticación:** Eliminación de accesos anónimos, asegurando que solo dispositivos conocidos interactúen con el sistema.
- Autorización (ACLs):** Implementación del principio de mínimo privilegio. Cada sensor solo puede escribir sus datos y leer sus comandos, mientras que la lógica central se reserva exclusivamente al Panel Principal.
- Confidencialidad (SSL/TLS):** Cifrado del canal de comunicación para proteger los datos de telemetría y las credenciales de acceso frente a ataques de intercepción (Man-in-the-Middle).

Esta configuración garantiza la integridad y confidencialidad del hogar conectado planteado.

