Miquel Àngel Román Colom
Àrees d'Aplicació d'IoT

Detector de presència amb l'Arduino Nano 33 IoT

Breu descripció del projecte

Implementar un detector de presència amb el node Nano 33 IoT reportar a un servidor remot, en el meu cas: una Raspberry Pi 4.

La pila tecnològica és la següent:

- L'Arduino Nano 33 IoT corrent l'stack del **Arduino SDK** amb les següents dependències:
  - **ArduinoBLE**.h (desenvolupada per Arduino), per accedir fàcilment al xip de BLE.
  - **WiFiNINA**.h (desenvolupada per Arduino), per accedir fàcilment a la xarxa WiFi.
  - **PubSubClient**.h (desenvolupada per Nick O'Leary), un client MQTT que suporta connexions TLS
- A la Rasberry Pi 4 corrent un sistema operatiu basat en Debian (Raspbian, per ARM)
  - **Mosquitto**, un broker MQTT que suporta TLS
  - **InfluxDB**, un sistema gestor de bases de dades enfocat a series temporals de dades
  - **Telegraf**, pot fer molt més, però en aquest projecte actua com un client MQTT subscrivint-se a un tòpic i inserint les publicacions al tòpic a l'InfluxDB.
  - **Grafana**, una eina que de visualització de dades a partir de diverses fonts d'informació. En aquest cas ens mostrarà la informació provinent de l'InfluxDB.

Descripció del Sketch de l'Arduino

L'sketch de l'Arduino s'encarrega de detectar dispositius que fan "*advertising*" de paquets *Bluetooth Low Energy*, així com polseres o telèfons intel·ligents, i recol·lecta informació variada sobre aquests dispositius.

Fem intervals de 10 segons on ens quedem a l'escolta de paquets BLE perquè s'ha de gestionar l'accés a la xarxa ja que no podem emprar simultàniament WiFi i BLE degut a que comparteixen la mateixa antena. Un cop finalitzada la fase d'escanejat, desactivem el Bluetooth i ens connectem a la xarxa WiFi per a enviar quants de dispositius diferents hem detectat durant la fase prèvia; i repetim indefinidament.

Cada vegada que detectem un dispositiu nou l'emmagatzemem dins una estructura de dades de tipus **std::set<BLEDevice>** de C++ que utilitza com a clau l'adreça MAC del dispositiu detectat. D'aquesta manera no reportem dues vegades el mateix dispositiu: l'estructura de dades ja s'encarrega de no admetre entrades amb la mateixa clau (direcció MAC en aquest cas).

L'interval d'escaneig és de 10 segons encara que el podem configurar fàcilment a través del fitxer de capçalera de C++ **config.h** on a més fet podrem configurar:

1. SSID i contrasenya WiFi
2. Credencials d'Autorització del nostre usuari MQTT
3. La IP del nostre broker MQTT
4. Desactivar les directives de logging al port sèrie. (interessant per qüestions de seguretat)

En cas de que es produeixin errors de connexió el node es re-intentarà connectar a la xarxa WiFi indefinidament ja que si no existeix una xarxa on reportar els resultats, de poc serveix seguir captant-los. En qualsevol cas, accés al BLE o WiFi, el terminal fa espera activa a que el xip estigui llest per ser utilitzat per evitar possibles errors en cascada del fet que el hardware no estigui inicialitzat.

Miquel Àngel Román Colom
Àrees d'Aplicació d'IoT

Protecció de la aplicació

Utilització de TLS

Per garantir un alt nivell de seguretat de la aplicació tota comunicació establerta està xifrada amb TLS v 1.2. Això vol dir que els clients saben qui és el servidor i poden contrastar que el servidor és qui diu ser.

No ha estat possible habilitar la verificació del client per part del servidor ja que el node Arduino Nano 33 IoT està un poc fitat en aquest sentit: només permet instal·lar certificats arrel de confiança dins el xip de comunicació i per tant no ha estat possible *flashejar* un certificat de client que proveir al broker IoT.

D'aquesta manera el tràfic MQTT circula xifrat a través de la xarxa i som capaços de verificar la identitat del nostre broker; protegint-nos de atacs de **sniffing** i **man in the middle**.

Desactivació de missatges de log

L'aplicació per facilitar diagnòstics durant la fase de desenvolupament fa ús de missatges de *log* a través del port sèrie integrat. Aquests missatges contenen informació sobre l'estat de l'execució del sketch així com: a quin SSID s'està intentant connectar, si hi ha hagut un error i informació sobre aquest o si s'està re-intentant una operació fallida i amb quins paràmetres.

Aquests missatges són útils per nosaltres però també poden ser útils per algun possible atacant que ens prengués un dispositiu que estès desplegat al món exterior. Per accedir a ells simplement bastaria connectar el node IoT a qualsevol ordenador amb un software lector de ports sèrie tal com el que inclou el mateix Arduino IDE.

Per prevenir-nos d'això s'ha establert una opció al fitxer **config.h** que permet activar o desactivar els missatges de *logging* quan es compila la aplicació. Una alternativa també vàlida però no implementada podria ser definir un pin secret que en cas de que estigués *pontejat* amb VCC i que quan es llegís un valor *HIGH* en aquest pin signifiqués que els missatges de *logging* estan habilitats. L'avantatge d'aquest darrer mètode es que ens permetria solucionar i diagnosticar problemes de nodes que ja han estat desplegats al mon exterior.

Autorització d'usuaris MQTT

Com bé s'ha esmentat en els primers apartats d'aquest document podem configurar les credencials amb les que el nostre es connecta al broker MQTT. Això és perquè feim ús de les *access control lists* del nostre mosquitto.

Això ens permet limitar l'accés dels clients MQTT. Això és útil en cas de que les nostres credencials del nostre node s'haguessin vist compromeses i un tercer intentés fer-ne us maliciós. En el nostre cas, aquest tercer maliciós només podria publicar en el tòpic **presence** sense poder-se subscriure a cap altre tòpic i en cas de que detectéssim alguna cosa estranya en els *logs* del nostre servidor, sempre podem revocar les credencials enviant al atacant un altre cop al punt de partida.

Tot el codi d'aquesta pràctica, a part d'adjunt, es pot trobar al meu repositori públic de github sota la següent URL:

https://github.com/NanoSpicer/PresenceDetector