Analisi delle problematiche di sicurezza relative al protocollo MQTT

Edoardo Di Paolo

Corso di Laurea in Informatica

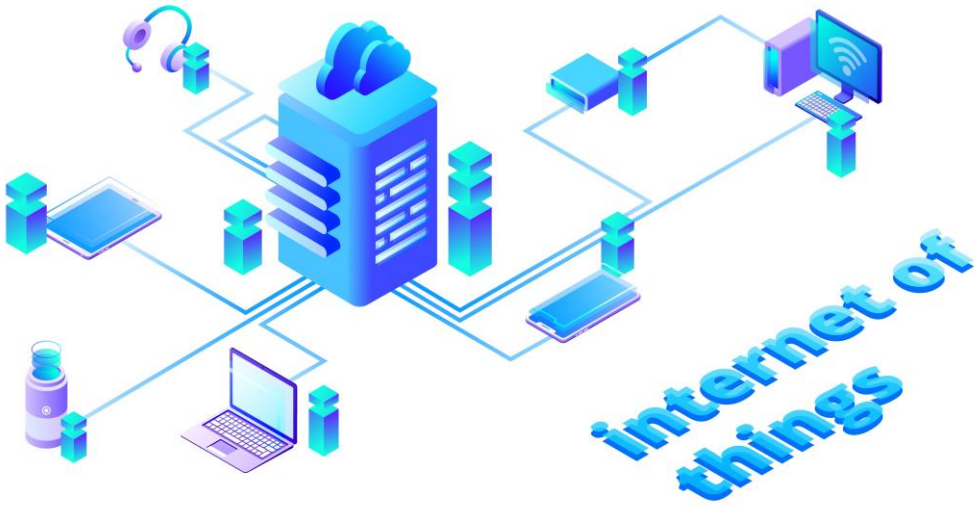
A.A. 2019/2020

SAPIENZA
UNIVERSITÀ DI ROMA

Presentazione dello scenario

Presentazione dello scenario

- L’**Internet of Things (IoT)** è in continua evoluzione e sempre più dispositivi sono connessi simultaneamente.
- L’IoT coinvolge continuamente nuovi campi.
- Sviluppo di nuovi protocolli come MQTT, CoAP e AMQP.
- Aumento degli attacchi nella rete Internet.



Presentazione dello scenario

- Il lavoro di tirocinio ha riguardato il protocollo **MQTT** nel dettaglio.
- Ricerca di **anomalie** nella gestione di alcuni esperimenti da parte delle diverse librerie studiate.
- Ricerca di **standard** del protocollo **non** rispettati dalle librerie.
- Implementazione del protocollo per eseguire gli esperimenti.
- Esperimenti su dispositivo fisico.

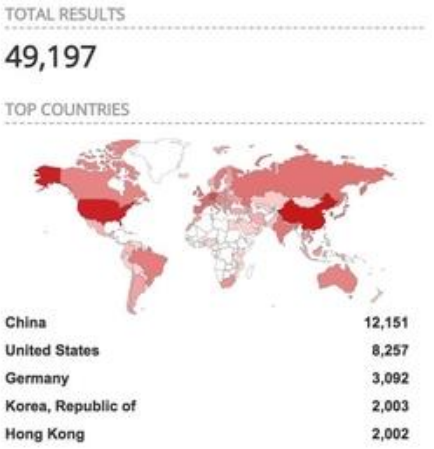
Il protocollo MQTT

Il protocollo MQTT

- **MQTT** (*Message Queue Telemetry Transport*) è un protocollo di tipo **publish-subscribe**.

- Il suo uso è **aumentato** di molto negli ultimi anni grazie alla crescita del numero di dispositivi IoT connessi.

- Molti dispositivi collegati in rete senza alcuna protezione per quanto riguarda l'accesso.



MQTT nel 2018

MQTT nel 2020

Il protocollo MQTT

- Il protocollo è semplice da utilizzare ed è adattabile sia a sistemi semplici che a sistemi molto complessi.

- Per poter funzionare richiede pochissima banda e un numero minimo di risorse da parte del dispositivo, perciò è un protocollo definito *leggero*.

- Supporta la comunicazione attraverso TLS/SSL per *cifrare* lo scambio di dati.

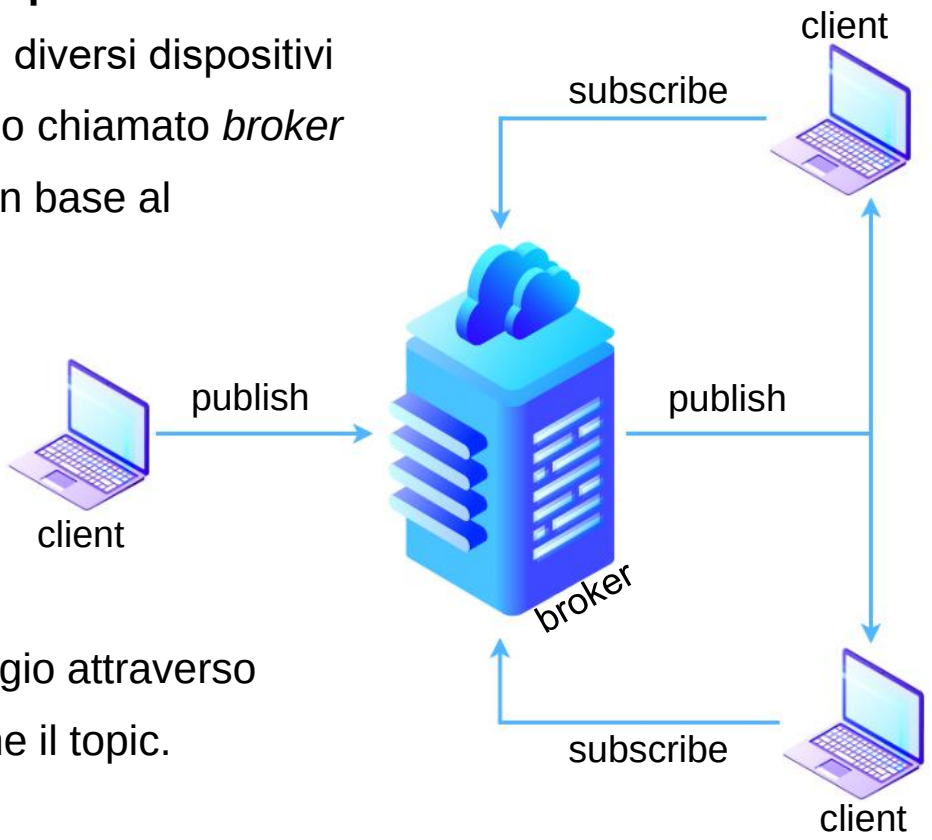
- I dispositivi IoT, a causa delle poche risorse che offrono, non possono garantire l'eventuale ricezione di un messaggio. Proprio per questo, MQTT mette a disposizione del client il *Quality of Service*.

Il protocollo MQTT – Esempi di utilizzo

- Moltissimi dispositivi che fanno parte dell'IoT permettono di essere gestiti attraverso MQTT.
- Facebook, in passato, ha utilizzato MQTT in Messenger.
- Viene utilizzato nel settore automobilistico ad esempio nei servizi di *car sharing* o anche per quanto riguarda la gestione dell'automobile stessa come lo sblocco di una portiera.
- È utilizzato anche nel settore ferroviaro per inviare informazioni come posizione e velocità di un treno.

Il protocollo MQTT - publish-suscribe

- L'architettura del protocollo è del tipo **publish-subscribe**. In un'architettura di questo genere, i diversi dispositivi dialogano attraverso un intermediario chiamato *broker* che distribuisce i messaggi ricevuti in base al topic del pacchetto ricevuto.



- I diversi client non comunicano *mai* direttamente tra di loro.
- Un client può pubblicare un messaggio attraverso il pacchetto *«publish»* specificandone il topic.
- Un client può sottoscriversi ad un topic attraverso il pacchetto *«subscribe».*

Il protocollo MQTT - Quality of Service

Il **Quality of Service** è un *«contratto»* stipulato tra mittente e broker che definisce la garanzia di consegna di un messaggio. In MQTT ci sono **3** livelli di QoS:

- Il *livello 0* è utile quando*:*
  - la rete è affidabile;
  - una piccola perdita di pacchetti non è importante;
  - i pacchetti devono essere consegnati il più velocemente possibili.
- Il *livello 1* è utile quando:
  - i pacchetti devono essere **necessariamente** consegnati e si ha una gestione dei possibili pacchetti duplicati;
  - non si può sopportare il possibile overload causato dal *livello 2.*
- Il *livello 2* è utile quando:
  - i messaggi possono essere consegnati lentamente;
  - i possibili messaggi duplicati possono causare problemi.

MQTT Broker ed implementazione del protocollo

MQTT Broker

- **MOSQUITTO:** broker molto utilizzato, open source e leggero. Supporta tutte le versioni del protocollo;
- **EMQ X:** broker molto utilizzato, open source scritto in *Erlang.* Permette di gestire milioni di connessioni simultanee anche con un unico server;
- **HiveMQ Community Edition:** scritto in *Java* ed open source, supporta tutte le versioni disponibili di MQTT;
- **Moquette:** broker meno conosciuto scritto in *Java* ed open source, supporta tutte le versioni disponibili di MQTT;
- **Aedes:** broker meno conosciuto scritto in *NodeJS* e *non* supporta MQTT 5. Ha molte librerie con le quali può essere integrato.

Implementazione del protocollo

- Il protocollo è stato implementato al fine di poter eseguire degli **esperimenti nel dettaglio**.
- Sono stati implementati tutti i pacchetti più importanti offerti da MQTT.
- Per il **trasporto** dei pacchetti è stata utilizzata la libreria *twisted*.
- **Implementazione** del pacchetto *publish*.

```
def publish(self, topic, message, dup=False, qos=0, retain=False, messageId=None):
    print(self.sentPacketColor + " PACKET SENT => PUBLISH [QoS: "+ str(qos) +", id: "+ str(messageId) +", payload: "+ str(message) +"]" + self.endColor)
    header = bytearray() # fix header
    varHeader = bytearray() # variable header
    payload = bytearray() # payload

    header.append(0x03 << 4 | dup << 3 | qos << 1 | retain) # campi del fix header (tipo pacchetto, duplicate flag, QoS, retain)

    varHeader.extend(_encodeString(topic.encode('utf-8'))) # topic nel var header


    if qos > 0:
        if messageId is None:
            varHeader.extend(_encodeValue(random.randint(1, 65535)))
        else:
            varHeader.extend(_encodeValue(messageId))

    payload.extend(_encodeString(message.encode("utf-8")))  # messaggio del  pacchetto
    header.extend(_encodeLength(len(varHeader) + len(payload))) # variable header + payload

    # trasporto del pacchetto
    self.transport.write(header)
    self.transport.write(varHeader)
    self.transport.write(payload)
```

Esperimenti

Implementazione del protocollo – Esperimenti

Una lista di alcuni degli esperimenti effettuati:

- Topic di pubblicazione (o sottoscrizione) molto lungo.
- Topic e payload non codificati in utf-8.
- Topic di sottoscrizione non valido.
- Client id non codificato in utf-8.
- Esperimenti con pacchetti malformati.
- Esperimenti con payload pesanti.
- Esperimenti con flood di pacchetti (subscribe o publish).
- Esperimenti sul Quality of Service.
- Esperimenti su librerie client.

Implementazione del protocollo – Esperimenti

- Possibilità di **gestire manualmente** il flusso dei diversi esperimenti attraverso quest'implementazione.
- Diverse tipologie: esperimenti sul *QoS,* esperimenti sulle *codifiche,* esperimenti con flood di pacchetti ed esperimenti con pacchetti malformati.
- Test scritti in *json* in cui vengono specificati i differenti pacchetti insieme ai differenti parametri.

```
[
    {
        "type": "subscribe",
        "params": {
            "topic": "test/topic"
        }
    },
    {
        "type": "publish",
        "params": {
            "topic": "test/topic",
            "message": "pacchetto #1",
            "qos": 2,
            "dup": false,
            "retain": false,
            "packetId": 1
        }
    },
    {
        "type": "disconnect"
    }
]
```

Implementazione del protocollo – Esperimenti

- I broker si sono comportati in maniera **differente** in diversi esperimenti.

- Esperimento con invio di due publish con *QoS* differente e stesso *packet id*.

Mosquitto	*EMQ X*	*HiveMQ*	*Moquette*	*Aedes*
Viene pubblicato solamente il primo pacchetto.	Vengono pubblicati entrambi i pacchetti.	Vengono pubblicati entrambi i pacchetti.	Vengono pubblicati entrambi i pacchetti.	Vengono pubblicati entrambi i pacchetti.


```
[
    {
      "type": "subscribe",
      "params": {
          "topic": "test/topic"
      }
    },
    {
      "type": "publish",
      "params": {
          "topic": "test/topic",
          "message": "pacchetto #1",
          "qos": 2,
          "dup": false,
          "retain": false,
          "packetId": 1
      }
    },
    {
      "type": "publish",
      "params": {
          "topic": "test/topic",
          "message": "pacchetto #2",
          "qos": 1,
          "dup": false,
          "retain": false,
          "packetId": 1
      }
    },
    {
      "type": "pubrel",
      "params": {
          "packetId": 1
      }
    }
]
```

Esperimenti sul dispositivo fisico

Implementazione del protocollo – Dispositivo fisico

- Esperimenti effettuati sui broker e su un dispositivo fisico.
- Il dispositivo fisico supporta il protocollo MQTT e si connette ad un broker che gli viene specificato.
- Mette a disposizione diversi topic con i quali si può interagire per modificare, ad esempio, l'intensità della luce.
- I risultati degli esperimenti effettuati in precedenza son stati confermati anche in questo caso.

Dispositivo fisico testato

Implementazione del protocollo – Dispositivo fisico

- Il **firmware** del dispositivo può essere un problema.
- Possibilità di aggiornare il firmware attraverso comandi MQTT.
- Un attaccante può caricare un firmware malevolo e prendere il controllo del dispositivo.
- Problemi di privacy, partecipazione a botnet e *man in the middle.*

Grazie!

Librerie client MQTT

- Sono disponibili librerie scritte in diversi linguaggi di programmazione, ad esempio ***paho*** e ***MQTT.*****js**

- Supportano tutte le versioni di MQTT.

- Si sono dimostrate sicure e robuste agli esperimenti a cui sono state sottoposte.

- Sono librerie con community molto attive che permettono di limitare i possibili problemi.

Ulteriore esperimento QoS

- I broker si sono comportati in maniera **differente** in diversi esperimenti.

- Esperimento in cui si inviano due pacchetti con lo stesso QoS ma si invia un solo pubrel corretto.

Mosquitto	*EMQ X*	*HiveMQ*	*Moquette*	*Aedes*
Viene pubblicato solamente il primo pacchetto.	Viene pubblicato solamente il primo pacchetto.	Vengono pubblicati entrambi i pacchetti.	Vengono pubblicati entrambi i pacchetti.	Il client si disconnette non producendo alcun log.


```
[
    {
      "type": "subscribe",
      "params": {
         "topic": "test/topic"
      }
    },
    {
      "type": "publish",
      "params": {
          "topic": "test/topic",
          "message": "pacchetto #1",
          "qos": 2,
          "dup": false,
          "retain": false,
          "packetId": 1
      }
    },
    {
      "type": "publish",
      "params": {
          "topic": "test/topic",
          "message": "pacchetto #2",
          "qos": 2,
          "dup": false,
          "retain": false,
          "packetId": 1
      }
    },
    {
      "type": "pubrel",
      "params": {
          "packetId": 2
      }
    },
    {
      "type": "pubrel",
      "params": {
          "packetId": 1
      }
    }
]
```

Retained Messages

- È un normale pacchetto con la flag *«retain»* impostata su ***true***.
- Il broker salva l'**ultimo** retain message e il QoS di quel pacchetto per il topic specifico.
- Tutti i client che si sottoscrivono al topic riceveranno, appena dopo la sottoscrizione, il retained message.
- È molto utile nel caso in cui un client debba ricevere un update sullo stato del sistema non appena si sottoscrive al topic.