SAPIENZA
UNIVERSITÀ DI ROMA

Analisi delle problematiche di sicurezza del protocollo MQTT

Facoltà di Ingegneria dell'informazione, informatica e statistica

Corso di Laurea in Informatica

Candidato

Edoardo Di Paolo
Matricola 1728334

Relatore

Prof. Angelo Spognardi

Anno Accademico 2019/2020

Analisi delle problematiche di sicurezza del protocollo MQTT
Tesi di Laurea. Sapienza – Università di Roma

Questa tesi è stata composta con LaTeX e la classe Sapthesis.

Email dell'autore: dipaolo.1728334@studenti.uniroma1.it

Dediche.

Indice

Capitolo 1

Introduzione

Negli ultimi decenni il numero di dispositivi collegati ad Internet è cresciuto esponenzialmente. Ormai, nel 2020, non si hanno più solamente computer o cellulari in rete ma anche elettrodomestici, macchine industriali e strumenti medici, tutti dispositivi che qualche anno fa erano offline. Tutto ciò fa riferimento all'IoT, l'*Internet of Things*.
Parallelamente allo sviluppo di queste nuove tecnologie, sono stati studiati nuovi protocolli affinché i dispositivi potessero essere utilizzati in maniera efficiente. Ad esempio ci sono alcuni sensori i quali permettono di misurare temperatura ed umidità di una stanza e funzionano attraverso l'uso di una semplice pila; dunque è necessario andare a ridurre il costo energetico della connessione così da aumentare la durata di utilizzo del dispositivo.
Alcuni esempi di protocolli possono essere: MQTT, CoAP, AMQP e WebSocket. Ovviamente tutti i protocolli hanno la possibilità di essere integrati con TLS (*Transport Layer Security*) così da poter garantire una maggiore sicurezza nello scambio dei dati; questo, però, potrebbe andare a gravare sui consumi del dispositivo poiché dovrebbero essere effettuati più calcoli affinché avvenga il trasporto dei dati. Inoltre, con l'aumento di questi nuovi dispositivi sono aumentate anche le possibili minacce relative all'IoT. Un esempio è il malware Mirai che, nel 2016, ha infettato milioni di dispositivi rendendoli parte di una botnet la quale, successivamente, ha attaccato attraverso un DDoS il fornitore di servizi DNS Dyn così da rendere inaccessibili milioni di siti web. A causa, anche, di questa tipologia d'attacco, sempre più comune, i protocolli sviluppati devono presentarsi sicuri e robusti.

In questa relazione viene descritto lo studio effettuato su uno dei maggiori protocolli nell'IoT: MQTT. Nello specifico durante l'attività di tirocinio siamo

andati alla ricerca di possibili anomalie e/o violazioni del protocollo da parte dei broker server i quali dovrebbero rispettare ogni standard definito per MQTT.

Capitolo 2

MQTT

2.1 Descrizione del protocollo

MQTT, acronimo di *Message Queuing Telemetry Transport*, è un protocollo di tipo *publish-subscribe* il quale permette il trasporto di messaggi tra diversi dispositivi tramite TCP/IP. Il protocollo è molto utilizzato in ambito IoT per la sua semplicità e anche per la banda la cui richiesta è davvero bassa.

MQTT presenta un modello di architettura differente, ad esempio, dal tipico client/server del protocollo HTTP; infatti adotta il meccanisco *publish-subscribe* (pub/sub) attraverso l'utilizzo di un *broker*. Il funzionamento del modello pub/sub avviene attraverso la pubblicazione e la sottoscrizione da parte del client a diversi topic, che possono essere paragonati a dei canali di comunicazione. In MQTT il publisher e il subscriber non comunicano mai direttamente e non sono neppure consapevoli della presenza dell'uno e dell'altro: il collegamento è gestito dal broker il cui compito è quello di filtrare i messaggi che riceve e distribuirli ai vari subscribers. In questo capitolo sono analizzate le principali caratteristiche che il protocollo MQTT mette a disposizione.

2.1.1 Topic

Nel protocollo MQTT un topic non è altro che una stringa codificata in UTF-8 che il broker utilizza per filtrare i messaggi da inviare successivamente ad ogni client sottoscritto a quel topic. I topic sono case-sensitive, quindi bisogna prestare attenzione alle lettere maiuscole o minuscole, e la lunghezza minima dev'essere di un carattere. Inoltre, MQTT mette a disposizione del client dei *wildcard*, i quali permettono di connettersi simultaneamente a più topic; uno è rappresentato dal simbolo +, mentre l'altro dal simbolo #. Il primo è chiamato

single-level wildcard, mentre il secondo *multi-level* wildcard. Ci sono dei topic, a cui non si può pubblicare alcun messaggio, riservati allo stato del sistema e sono quei topic che cominciano per $SYS/.
Alcuni esempi di topic validi possono essere i seguenti:

1. *casa/luci/sala* - topic specifico per le luci della sala;
2. *casa/+/sala* - include tutti i topic dei dispositivi che fanno riferimento alla sala (luci comprese);
3. *casa/#* - include tutti i topic che fanno riferimento alla casa.

2.1.2 Connessione al broker

Come scritto nell'introduzione di questo capitolo, la connessione avviene solamente fra client e broker. Per client intendiamo un qualsiasi dispositivo il quale permette di gestire una connessione ad un broker, che si comporta come un server.

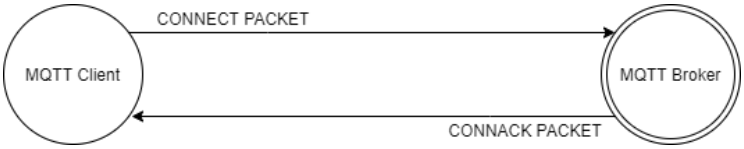

Figura 2.1. Flusso connessione al broker MQTT.

Come si può vedere dalla Figura 2.1, la connessione avviene tramite lo scambio di due pacchetti: *connect*, inviato dal client al broker, e *connack* inviato dal broker al client. Il pacchetto connect è così strutturato:

Tabella 2.1. Struttura pacchetto connect.

Parametro	**Descrizione**
clientId	rappresenta l'identificativo del client che chiede di connettersi
cleanSession	valore booleano il quale specifica se la connessione è persistente o meno
username	rappresenta l'username necessario affinché avvenga la connessione
password	la password associata all'username
lastWillTopic	permette di notificare gli altri client in caso di crash del client
lastWillQos	permette di notificare gli altri client in caso di crash del client
lastWillMessage	permette di notificare gli altri client in caso di crash del client
lastWillRetain	permette di notificare gli altri client in caso di crash del client
keepAlive	rappresenta in secondi l'intervallo massimo in cui broker e client possono non inviarsi messaggi

Il pacchetto connack, invece, è così strutturato:

Tabella 2.2. Struttura pacchetto connack.

Parametro	Descrizione
sessionPresent	flag riferita al cleanSession del pacchetto connect
returnCode	stato della connessione

Per quanto riguarda il returnCode descritto nella Tabella 2.2, questo è un valore che va da 0 a 5:

- 0: connessione accettata;
- 1: connessione rifiutata, versione del protocollo non valida;
- 2: connessione rifiutata, clientId non valido;
- 3: connessione rifiutata, server non disponibile;
- 4: connessione rifiutata, username o password errati;
- 5: connessione rifiutata, non autorizzati.