Desenvolvimento de Software para Implementar Algoritmos de Segurança nas Comunicações de Aplicações (MQTT e/ou COAP) em equipamentos IoT

Filipe Henriques
2180066@my.ipleiria.pt
Tiago Martins
2182716@my.ipleiria.pt

Mestrado em Cibersegurança e Informática Forense
Instituto Politécnico de Leiria
Leiria, PT

Abstract

With the growing usage and dependency of IoT in many industries, having proper security is a must in order to protect your business.

MQTT (Message Queuing Telemetry Transport) is a publish-subscribe messaging protocol that was designed for low bandwidth networks. CoAP (Constrained Application Protocol) is a client-server protocol based on the REST model and designed to be as simple as it can be. Both these protocols have been design with constraints in mind in which makes them interesting protocols to be used in conjunction with low spec IoT devices.

In this paper we explore some implementations of these protocols in popular IoT devices, compare some of its aspects security wise and explain in which scenario one is better suited than the other.

1 Introdução

Nos últimos anos a utilização de dispositivos IoT (Internet of Things) tem vindo a crescer drasticamente. Existem cada vez mais dispositivos conectados às redes WI-FI, como as Smart TVs, Home Assistants, Drones, Frigoríficos, Máquinas de lavar, Impressoras, entre outros. No entanto, muitos dos fabricantes destes dispositivos não implementam boas ou nenhumas medidas de segurança de modo a diminuir o preço de produção, pelo que a informação pode ser facilmente visualizada com um software de network sniffing. Estes dispositivos também têm tido uma grande empregabilidade no ambiente empresarial, pelo que é necessário garantir a segurança da informação que passa pelos mesmos.

Grande parte dos dispositivos IoT são compostos por componentes de baixo poder de processamento de modo a reduzir custos e consumo de energia. Deste modo é recomendável a utilização de protocolos de comunicação que tenham em conta tais limitações. Dois protocolos geralmente utilizados são o MQTT (Message Queuing Telemetry Transport) e o CoAP (Constrained Application Protocol).

Este paper explora algumas das implementações destes protocolos nos dispositivos IoT mais populares tais como o Arduino MKR1000 e o Raspberry Pi 3B. É também feita uma análise comparativa em termos de segurança e cenário em que melhor se adequa cada um dos protocolos.

2 Conceitos

2.1 Internet of Things

Geralmente quando se fala de Internet of things (IoT), ou Internet das coisas, referimo-nos a todos os dispositivos ligados à Internet para além dos habituais smartphones, tablets, computadores portáteis/fixos, entre outros. No entanto, IoT também é um termo utilizado para se referir à infraestrutura global que oferece à sociedade da informação serviços de qualidade, tais como a possibilidade de interconectar, fisicamente ou virtualmente, os mais variados dispositivos que normalmente operavam fora da rede. É através da exploração, identificação, captura de dados, capacidade de processamento e comunicação que o IoT faz utilização a total destes dispositivos. São disponibilizados serviços a qualquer tipo de aplicação, onde é garantida a execução de requisitos de segurança e privacidade.[4]

2.2 MQTT

O MQTT é um protocolo de mensagens extremamente simples e leve com uma arquitetura publish/subscribe. Este foi desenhado para ser utilizado em dispositivos com poder computacional limitado e para redes de baixa largura de banda, com alta latência ou mau desempenho. O MQTT foi inventado pelo Dr. Andy Stanford-Clark da IBM, e por Arlen Nipper da Arcom em 1999. Este protocolo segue os princípios de design que consistem em minimizar a largura de banda de rede assim como os recursos necessários num dipositivo, tentando ainda assim, garantir a confiabilidade e algum grau de garantia de entrega. É com estes princípios que o MQTT se torna num protocolo ideal para as comunicações entre máquinas de normal poder computacional e o mundo dos IoT, onde a largura de banda e a poupança de energia são essenciais para o seu funcionamento.[5]

2.2.1 Mosquitto

Mosquitto, ou Eclipse Mosquitto, é um intermediário (broker) de mensagens de código-fonte aberto, que implementa as versões 3.1 e 3.1.1 do protocolo MQTT. Sendo o Mosquitto baseado no protocolo MQTT, é leve e adequado para ser utilizado em qualquer tipo de dispositivo desde um computador de baixa potencia até a servidores de grande poder computacional. O Mosquitto também fornece uma biblioteca em linguagem C para implementar os seus clientes MQTT assim como as aplicações da linha de comandos mosquitto_pub e mosquitto_sub. Em termos de segurança, o Mosquitto suporta user authentication e certificados digitais para encriptação ponto a ponto utilizando TLS (Transport Layer Security). [2]

2.3 CoAP

O CoAP é um protocolo de comunicação web desenhado especificamente para uso em dispositivos com baixos recursos computacionais e/ou uma fonte de energia limitada e redes constrangidas. O CoAP utiliza o UDP na camada de transporte e assenta numa arquitetura de request/response entre um cliente e um servidor com vários endpoints aplicacionais. Este protocolo é baseado no princípio dos serviços RESTful, pelo que inclui conceitos chave da Web tais como URIs e Internet Media Types. O CoAP é também desenhado para interagir facilmente com HTTP de modo a facilitar a integração com a Web, atendendo aos requisitos necessários como suporte de multicast, baixa overhead, e design simplificado para ambientes constrangidos. [6]

2.3.1 Libcoap

Neste projeto foi usado uma implementação do CoAP escrita em C, a libcoap, no qual é implementada uma aplicação protocolar leve para dispositivos constrangidos pelos seus recursos, como o poder computacional, tamanho da memoria ou dos pacotes de rede. O libcoap é desenhado para ser multiplataforma, de modo a poder correr tanto em dispositivos embutidos, como também em sistemas de topo a correr sistemas operativos POSIX. O Libcoap também integra medidas de segurança em termos de encriptação ponto a ponto através do DTLS (Datagram Transport Layer Security), uma implementação semelhante ao TLS mas em UDP. [1]

3 Implementação e cenário de testes

De forma a comparar as implementações de MQTT e de CoAP, foram construídos três cenários, um para o MQTT e dois para o CoAP. No cenário de teste do MQTT é utilizado um Arduino a reportar leituras de um sensor utilizando a biblioteca PubSubClient (Uma pequena implementação de MQTT para o Arduino), e um Raspberry Pi a desempenhar o papel de broker utilizando o Mosquitto. Para o cenário de testes no CoAP, neste é usado um Raspberry Pi como servidor e um Arduino a desempenhar o papel de cliente com a implementação do libcoap e CoAP simple library respetivamente. Semelhante ao MQTT, o CoAP suporta encriptação e autenticação através do protocolo DTLS, no entanto de momento não existem bibliotecas para o Arduino que suportem este protocolo. De

modo a testar o protocolo DTLS no CoAP, utilizou-se um computador a correr o cliente do libcoap. Para a captura dos dados em circulação entre os dispositivos dos cenários, é utilizado o tcpdump no Raspberry Pi sendo os ficheiros do output da recolha transferidos para um portátil onde são analisados os ficheiros com a aplicação Wireshark.

3.1 Bateria de testes

Como bateria de testes, foram realizados testes para analisar o desempenho das comunicação consoante os protocolos usados e as implementações de segurança adicionais implementadas pelo MQTT e CoAP. Deste modo realizaram-se os seguintes testes:

MQTT:

- Sem mecanismos de segurança;
- Autenticação de utilizador;
- TLS;
- TLS + Autenticação de utilizador.

CoAP:

- Sem mecanismos de segurança;
- DTLS.

3.1.1 Descrição técnica dos testes

No MQTT, para a utilização de TLS procedeu-se à criação de um certificado digital X.509 auto assinado, a utilização deste certificado foi especificada no ficheiro de configuração do Mosquitto e foi também instalado no firmware do Arduino. De modo a utilizar a funcionalidade de user authentication do Mosquitto, foi criado um password file recorrendo à ferramenta mosquitto_passwd e este ficheiro foi também especificado no ficheiro de configuração do Mosquitto.

Para o CoAP, foi utilizada uma versão do libcoap com o tinyDTLS em que o processo de autenticação consistia na utilização de um utilizador e chave partilhada entre o cliente e o servidor. Para aproximar o máximo possível aos testes do MQTT, recorreu-se ao método PUT do CoAP e foi enviada a mesma mensagem por todos os cenários de teste.

3.2 Valores a recolher nas experiências e método de recolha

Durante as experiências foram tomados em importância os valores referentes ao tamanho do pacote, o tamanho do corpo do pacote, e o número de pacotes até ao primeiro pacote referente aos dados que a aplicação enviava ao broker. São também observados os conteúdos do body dos pacotes enviados de modo confirmar a confidencialidade da mensagem e dados de autenticação. Para recolher estes valores é usado o tcpdump no Raspberry Pi, sendo depois analisados os resultados desta recolha na aplicação Wireshark.

4 Análise dos valores capturados

Os valores obtidos presentes na tabela 1, foram escolhidos e estabelecidos segunda a perspetiva de que não existe nenhum tipo de erro ou outra ocorrência inesperada durante o processamento das comunicações nos cenários de teste. Desta forma não serão contabilizadas os pacotes retransmitidos derivado à não receção do pacote do protocolo MQTT ou CoAP por qualquer um dos dispositivos, e também os pacotes referentes a protocolos não essenciais ao funcionamento e avaliação do desempenho dos protocolos de comunicação. Para efeitos de esclarecimento, na tabela 1, os valores recolhidos pertencem ao numero de pacotes até ao primeiro pacote contendo dados aplicacionais, onde o próprio pacote é contabilizado. O tamanho da mensagem e dos pacotes encontra-se em bytes. A palavra Auth nos cabeçalhos da tabela, refere-se ao mecanismo de autenticação oferecido pelo Mosquitto para aceder um tópico no broker e refere-se a autenticação de utilizador e não de mensagem.

Como se pode observar na tabela 1, os valores relativos aos tamanhos capturados para o protocolo MQTT divergem onde o TLS é utilizado com os que não o utilizam, no entanto, quanto ao número de pacotes enviados até ao primeiro pacote que contém dados aplicacionais, não há grande variação. Curiosamente o teste de TLS é o que apresenta menor número de pacotes, com 13. Já com o CoAP já não se verifica a mesma dispersão nos valores dos tamanhos entre os dois diferentes testes. A diferença está no número de pacotes, no teste de DTLS pode-se verificar que existe uma grande quantidade de pacotes para elaborar o seu processo. Por sua vez, o funcionamento do CoAP sem nenhum tipo de mecanismo de segurança, é enviada a mensagem no primeiro pacote enviado ao servidor. Isto demonstra os benefícios do CoAP face ao MQTT em redes constrangidas. O facto de existir um maior número de pacotes quando segurança do tipo TLS ou DTLS é utilizada, está inerente ao processo de handshake que era observável durante a troca de pacotes. Para além dos valores recolhidos na tabela anterior, foi visualizado o conteúdo das mensagem dos pacotes protocolares, em especial o conteúdo do pacote do protocolo MQTT que inicia a autenticação no teste do Auth. Aqui por não existir encriptação da mensagem, as credenciais para autenticação passam pela rede em pleno texto. Apenas nos testes com TLS se verifica a confidencialidade da mensagem devido à encriptação do conteúdo aplicacional dos pacotes ponto a ponto.

	MQTT (Mosquitto)				CoAP (libcoap)	
	TLS + Auth	TLS	Auth	No Security	No Security	DTLS
Size of the message	69	69	14	14	28	44
Size of the packet	123	123	68	68	70	86
Packets until 1st message	14	13	14	14	1	11

Table 1: Valores recolhidos nos testes

5 Discussão dos resultados

5.1 Dispersão dos valores obtidos

Os valores obtidos nos testes apresentam uma dispersão significativa entre os testes com TLS/DTLS e os testes sem. Apesar destes valores não parecerem grandes em tamanho ou em quantidade, no mundo dos IoT uma diferença minima de bytes pode ter um grande impacto em termos de overhead. Os valores relativos aos testes que apresentam menor ou nenhuma segurança acabam por ser mais baixos devido aos algoritmos e mecanismos serem simples e exigirem menos passos no estabelecimento da comunicação, e daí necessitarem de menos recursos. Por outro lado, os valores nos testes de TLS/DTLS são mais altos devido à forma como o processo de estabelecimento e troca de chaves entre os dispositivos é realizada, o handshake envolvido para trocar a chave simétrica no qual é usado para estabelecer o canal seguro é um processo longo que requer uma maior utilização dos recursos computacionais dos dispositivos.

5.2 Cenários adequados ao MQTT e CoAP

Os dois protocolos usados para os testes são protocolos para a transmissão de dados aplicacionais entre dispositivos com poucos recursos, mas apesar disto eles apresentam diferenças no modo de funcionamento no qual se tornam mais adequados para diferentes cenários. O MQTT é um protocolo de publish/subscribe o que significa que o broker tem comunicação entre vários clientes (one-to-many). Desta forma o broker irá necessitar de suficientes recursos computacionais para processar o funcionamento das comunicações e também para os processar os mecanismos de segurança necessários para o cenário no qual se encontra. Isto pode-se verificar pela amostra de valores recolhidos nos testes, o MQTT quando usa TLS, é aquele que mostra obter os valores mais elevados tanto no número de pacotes até à mensagem como no tamanho do pacote. No CoAP as comunicações são de request/response, que é uma comunicação mais orientada de um para um (one-to-one), e como é desenhado exatamente para dispositivos com recursos constrangidos utiliza pacotes UDP de modo a diminuir

tanto a latência como o poder de processamento necessário à custa da entrega ser menos fiável. Para efeitos de segurança o CoAP usa o DTLS o que mesmo assim, como é possível verificar pelos valores obtidos, não causa grande discrepância nos tamanhos dos pacotes com dados aplicacionais. Mas por outro lado aumenta o número de passos necessários para estabelecer a segurança nas comunicações, o que em alguns casos pode não ser o mais desejável.

5.3 Impacto da segurança no desempenho

Apesar não terem sido captados os diferentes tempos de processamento dos diferentes protocolos nos testes, podemos dizer que persentimos ligeiras diferenças de desempenho no funcionamento em cerca de 1 segundo ou menos. Isto com os dispositivos que foram utilizados para desempenhar os testes, mas se fossem utilizados equipamentos mais pequenos e constrangidos nos recursos, acreditamos que o desempenho iria ser afetado negativamente, especialmente ao usar TLS ou DTLS. Isto mostra que a aplicação de segurança nestes tipos de equipamentos é uma decisão extremamente difícil de tomar e talvez impossível nos tempos atuais dependendo quase sempre do cenário de utilização. O processo de handshake é o que impacta mais a performance tanto do dispositivo como da ligação, visto que são necessários mais pacotes para estabelecer a ligação entre o cliente e o servidor/brooker. O TLS e o DTLS são parecidos no processo de handshake, em ambos existe uma primeira conexão de modo a que o servidor crie um canal de comunicação com o cliente. No caso do TLS é um connection request seguido de um connection acknowledged, enquanto no DTLS devido à natureza stateless do UDP, o servidor cria e envia um cookie para o cliente de modo a reconhecê-lo nas próximas comunicações. Após esta primeira conexão, dá-se o processo de autenticação onde, no caso do TLS, existe um certificado digital específico para o servidor que poderá ser assinado por uma entidade certificadora. Também pode existir um certificado para o cliente mas não é obrigatório. O servidor cria uma chave de sessão para a ligação, assina e encripta essa chave com a chave privada do certificado. O cliente vai decifrar a chave de sessão com a chave publica do certificado e o processo de autenticação dá-se por concluído, passando assim à troca de mensagens cifradas pelo canal seguro estabelecido.[3] No caso do DTLS, também pode funcionar via certificados da mesma maneira que o TLS, mas no caso do TinyDTLS (implementação de DTLS utilizada na libcoap), é utilizada uma identidade e uma chave partilhada pré-definidas no servidor CoAP.

6 Conclusão

As implementações dos protocolos MQTT e CoAP utilizadas Mosquitto e libcoap respetivamente, apresentam bons mecanismos de segurança para dados aplicacionais. Com os valores obtidos nos testes, comprovou-se que é possível assegurar um bom nível de segurança nas comunicações aplicacionais dos protocolos sem comprometer demasiado o desempenho do software nos dispositivos utilizados, que são dispositivos ligados ao mundo do IoT. No entanto a utilização destes mecanismos de segurança apresentam sempre algum impacto na performance. No caso dos testes efetuados este impacto não foi muito elevado visto que apenas era simulado o envio de valores referentes à leitura de um sensor, mas para outros cenários onde muitas vezes os recursos disponibilizados por estes dispositivos são escassos, a escolha do mecanismo de segurança poderá ser muito importante. Existem cenários onde a confidencialidade não é a maior preocupação, no caso de apenas se querer criar confidencialidade nos dados das mensagens para impedir os mais curiosos de visualizar, basta aplicar um algoritmo simétrico à mensagem e utilizar uma chave partilhada. Deste modo existe um maior overhead pelo facto de não haver o processo de handshake. Este cenário partiria do princípio que a informação que passa nestes dispositivos não é muito sensível, e que o proprietário aceita o risco de que esta seja comprometida no caso de uma pessoa mais experiente em criptografia a decifre. Na nossa opinião, este cenário é um dos mais comuns para utilizadores que utilizam plataformas como o Raspberry Pi e Arduino para projetos pessoais, onde se quer ter alguma confidencialidade e poderá não justificar sacrificar poder de processamento para proteger o tipo de informação transportada. Deste modo achamos que para um trabalho futuro seria interessante desenvolver uma biblioteca que disponibilize ditas funções para estas plataformas.

Independentemente da finalidade para o qual é utilizada, tanto o MQTT como o CoAP, conseguem garantir formas semelhantes de segurança. No entanto, comparando as várias implementações existentes para cada protocolo, as que que oferecem mais formas de segurança são as implementações do MQTT, como o caso do Mosquitto. No decorrer do trabalho para aplicar o protocolo de segurança DTLS no CoAP, tornou-se difícil em encontrar bibliotecas que suportassem o mesmo. O CoAP é um protocolo relativamente recente, pelo que o seu suporte total é escasso em plataformas como o Arduino, onde não existe uma implementação que suporte DTLS, daí se dar preferência ao MQTT aquando se necessitar de garantir confidencialidade e integridade da informação. Tendo trabalhado com o CoAP, conseguimos identificar o potencial deste protocolo e a sua importância no mundo dos IoT, pelo que um possível trabalho futuro seria desenvolver uma implementação de DTLS nesta plataforma, contribuindo assim para a sua popularização.

References

[1] O. Bergmann. libcoap: C-implementation of coap. 2018. https://libcoap.net/.

[2] Mosquitto Eclipse. Eclipse mosquitto™ an open source mqtt broker. 2018. https://mosquitto.org/.

[3] Christine Hennebert and Jessye Dos Santos. Security protocols and privacy issues into 6lowpan stack: A synthesis. *IEEE Internet of Things Journal*, 1: 384–398, 10 2014. doi: 10.1109/JIOT.2014.2359538.

[4] ITU-T. Itu-t y.4000/y.2060. 2012. https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=y.2060.

[5] MQTT.org. Frequently asked questions. 2018. http://mqtt.org/faq.

[6] Z. Shelby. The constrained application protocol (coap). 2018. https://www.ietf.org/rfc/rfc7252.txt.