Cybersecurity in Internet of Things

Mestrado em Engenharia Informática – Computação Móvel

Roberto Emanuel Fernandes Leal

Leiria, novembro de 2020

Cybersecurity in Internet of Things

Mestrado em Engenharia Informática – Computação Móvel

Roberto Emanuel Fernandes Leal

Trabalho de projeto de Mestrado realizado sob a orientação do Professor Doutor Carlos Rabadão, professor Coordenador da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e coorientação do Professor Doutor Leonel Santos, Professor Adjunto da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria

Leiria, novembro de 2020

Originalidade e Direitos de Autor

O presente relatório de projeto é original, elaborado unicamente para este fim, tendo sido devidamente citados todos os autores cujos estudos e publicações contribuíram para o elaborar.

Reproduções parciais deste documento serão autorizadas na condição de que seja mencionado o Autor e feita referência ao ciclo de estudos no âmbito do qual o mesmo foi realizado, a saber, Curso em Mestrado em Engenharia Informática – Computação Móvel, no ano letivo 2019/2020, da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria, Portugal, e, bem assim, à data das provas públicas que visaram a avaliação destes trabalhos.

Dedicatória

À minha mulher.

Agradecimentos

Agradeço a todos os intervenientes que me ajudaram, direta e indiretamente, a realizar a presente trabalho de projeto, e a terminar este ciclo de estudos.

Em particular,

Ao Professor Doutor Leonel Santos e ao Professor Doutor Carlos Rabadão, que foram incansáveis no apoio prestado, na orientação e na passagem de conhecimentos.

Ao meu colega de trabalho e estudante do mestrado de Cibersegurança e Informática Forense, o Luís Canuto, pelo apoio, pela ajuda e pela partilha de conhecimentos.

À Vânia Ferreira, pela força, pelo apoio e pela confiança que depositou em mim.

Ao meu pai, à minha mãe e à minha irmã pelo apoio, bem como, aos meus amigos, em particular ao Diogo Falcão que em vários momentos me ajudou.

Muito obrigado a todos.

Nota Prévia

Este trabalho de projeto tem o propósito de demonstrar uma arquitetura que pretende fazer a deteção de intrusões em sistemas *Internet of Things* (IoT), para detetar, em tempo real, intrusões com origem interna ou externa, com o foco nas intrusões dirigidas às camadas de rede e de aplicação de uma arquitetura IoT.

O desenvolvimento deste trabalho de projeto teve origem num grupo de trabalho constituído por um estudante de doutoramento e dois estudantes de mestrado, integrados no Centro de Investigação em Informática e Comunicações do Politécnico de Leiria. A base deste trabalho de projeto, enquadra-se no trabalho de doutoramento do Professor Doutor Leonel Santos, designado de "Sistema de deteção de intrusões para a Internet das Coisas (2020)".

Em específico, este trabalho de projeto procede à análise, caracterização e avaliação das especificações para o tráfego gerado pelos aplicativos *Message Queue Telemetry Transport* (MQTT) e *Message Queuing Telemetry Transport over* TLS (MQTTS), tendo em consideração que ambos os protocolos aplicacionais utilizam como protocolo de transporte o *Transmission Control Protocol* (TCP).

No decorrer da elaboração do presente trabalho de projeto, foi também publicado um artigo com o título "*MQTT Flow Signatures for the Internet of Things*", Leal, et al. (2019).

Resumo

Um sistema *Internet of Things* (IoT) tem um largo leque de aplicações e de ambientes de utilização, isto é, desde uma rede doméstica a uma rede industrial, ou desde uma rede com alguns dispositivos IoT a uma rede com milhares de dispositivos IoT. Os dispositivos que estão presentes nos sistemas IoT podem ter características diversas, por exemplo, diferentes versões de sistemas operativos instalados e diferentes capacidades e recursos disponíveis. Em suma, um sistema IoT pode apresentar um elevando nível de heterogeneidade.

De forma a facilitar e a normalizar a operação dos sistemas IoT, existem vários tipos de *standards* e de protocolos que podem ser utilizados, tais como, o *Message Queuing Telemetry Transport* (MQTT), que não é específico para um sistema IoT mas que pela sua simplicidade é bastante utilizado, e a sua variante segura o *Message Queuing Telemetry Transport over* TLS (MQTTS). As utilizações do MQTT e do MQTTS facilitam a uniformização e a comunicação entre os vários dispositivos IoT.

Estes sistemas IoT, são compostos por vários dispositivos IoT. Estes caracterizam-se por terem fracos recursos de *hardware*, nomeadamente ao nível da capacidade de memória *Random-Access Memory* (RAM) ou *Read-Only Memory* (ROM), de processamento e de armazenamento em disco. Estes dispositivos estão frequentemente expostos a ambientes exteriores e, por isso, estão mais vulneráveis a ataques à sua integridade e disponibilidade, bem como, à informação por si recolhida e transmitida.

A monitorização do tráfego e a deteção das anomalias podem dar um forte contributo para a mitigação destas ameaças, podendo ser efetuadas através de soluções do tipo *Intrusion Detection System* (IDS). Estas soluções são caracterizadas por causar um baixo impacto no desempenho dos sistemas IoT e, por isso, representam uma solução interessante de integração neste tipo de sistemas. No entanto, é essencial que os sistemas de IDS tenham na sua base de dados interna as especificações, as regras e os comportamentos necessários para analisar o tráfego que flui num sistema IoT.

É neste âmbito que se enquadra este trabalho, nomeadamente na análise, na caracterização e na avaliação das especificações do tráfego gerado pelos aplicativos MQTT e MQTTS, tendo-se determinado os *Information Elements* (IE) mais relevantes a considerado e criado um

cenário de testes capaz de capturar esses IE. Posteriormente, esses IE foram analisados com o intuito de caracterizar de forma generalista o tráfego MQTT e o tráfego MQTTS.

Palavras-chave: MQTT, Internet das Coisas, IDS, Segurança, Ataques, Fluxos de tráfego

Abstract

An Internet of Things (IoT) system has a broad spectrum of applications and deployable scenarios. From a domestic scenario with some IoT devices to an Industrial environment with thousands of IoT devices. The devices present in an IoT system can have diverse characteristics and configurations, such as different Operating Systems, resources, or components. Therefore an IoT system could represent a high degree of heterogeneity.

In a bid to simplify, facilitate and normalize the operation of IoT systems, there are various standards and protocols the can be used, such as the Message Queuing Telemetry Transport protocol (MQTT) though not specific to an IoT system, its simplicity influenced its wide use along with its secure variant, Message Queuing Telemetry Transport over TLS (MQTTS). The use of both MQTT and MQTTS facilitates and improves the uniformity and communication between the potential diverse IoT devices.

These IoT systems are composed of various devices that can be characterized by being weak in hardware resources, namely the amount of Random Access Memory (RAM), Read-Only Memory (ROM), processing power, or storage. These devices are often exposed to the elements or exterior environments where they become more vulnerable to attacks on their integrity, availability, or the communication transmitted or received.

Network traffic monitoring and anomaly detection can be a great contribution to the mitigation of these threats. Among others, an Intrusion Detection System (IDS) based solution is a prime example for its low tax on an IoT system's performance. However, these IDS systems must integrate within their databases all specifications, rules, and behaviors necessary to analyze the traffic within an IoT system.

The objective of this research centers on the analysis, categorization, and evaluation of traffic generated by the MQTT and MQTTS protocols, determining the most relevant Information Elements (IE) to create test scenarios using said IEs. The traffic captured in these scenarios is then used to create proper generalized characterizations of MQTT and MQTTS traffic for an IoT system.

Keywords: MQTT, Internet of Things, IDS, Security, Attacks, Traffic Flow

Índice

Lista de Figuras

Lista de tabelas

Lista de siglas e acrónimos

6LoWPAN	IPv6 over Low power Wireless Personal Area Network
AMQP	Advanced Message Queuing Protocol
AVAC	Aquecimento, Ventilação e Ar Condicionado
BLE	Bluetooth Low Energy
CA	Certificate Authority
CERT	Carnegie Mellon University
CIA	Confidentiality, Integrity and Availability
CoAP	Constrained Application Protocol
CORE	Constrained Restful Environments
CPE	Complex Event-Processing
DDoS	Distributed Denial of Service
DDS	Data Distribution Service
DHCP	Dynamic Host Configuration Protocol
DIY	Do-It-Yourself
DNS-SD	DNS-Based Service Discovery
DoS	Denial of Service
DPI	Deep Packet Inspection
DUP	Duplicate and Damaged Packets
EMS	Event Management System
EPC	Electronic Product Code
EPCglobal	Electronic Product Code global
ESTG	Escola Superior de Tecnologia e Gestão
EXI	Efficient XML Interchange
FN	Falsos Negativos

FP	Falsos Positivos
HIDS	Host Intrusion Detection System
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Number Authority
IBM	International Business Machines Corporation
IDS	Intrusion Detection System
IE	Information Element
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IM	Instant Messaging
IoT	Internet of Things
IP	Internet Protocol
IPFIX	Internet Protocol Flow Information Export
IPL	Instituto Politécnico de Leira
IPSec	IP Security Protocol
IPSO	IP of Smart Objects
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
JSON	JavaScript Object Notation
Kalis	Knowledge-driven Adaptable Lightweight Intrusion Detection System
LTE-A	Long Term Evolution Advanced
M2M	Machine to Machine
MAC	Media Access Control
mDNS	multicast Domain Name Server
MIT	Massachusetts Institute of Technology

MQTT	Message Queue Telemetry Transport
MQTTS	Message Queuing Telemetry Transport over TLS
NFC	Near Field Communication
NIDS	Network IDS
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
OSI	Open System Interconnection
P&G	Procter & Gamble
PCAP	Packet CAPture
QoF	Quality of Flow
QoS	Quality of Service
RAM	Random-Access Memory
RDF	Re-source Description Framework
REST	REpresentational State Transfer
RFA	Registos de Fluxos de tráfego Anormal
RFC	Request For Comment
RFID	Radio-Frequency IDentification
RFN	Registos de Fluxos de tráfego Normais
ROM	Read-Only Memory
RPL	Low-Power and Lossy Networks
RTT	Round-Trip Time
SBC	Single Board Computer
SIEM	Security Information and Event Management
SLA	Service Level Agreement

SO	Sistema Operativo
SSL	Secure Sockets Layer
TAP	Test Access Port
TCP	Transmission Control Protocol
TD	Taxa de deteção
TLS	Transport Layer Security
ToS	Type of Service
uCODE	Universal Computer Code
UDP	User Datagram Protocol
uID	ubiquitous IDdentifer
UTC	Universal Time Coordinate
UWB	Ultra-Wide-Band
VLAN	Virtual Local Area Network
VP	Verdadeiros Positivos
W3C	World Wide Web Consortium
WAN	Wide Area Network
WOL	Web Ontology Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol
YAF	Yet Another Flowmeter

1. Introdução

Internet of Things (IoT) é um modelo relativamente recente de interligação entre dispositivos simples, que permite executar tarefas simples, com controlo, monitorização e segurança. A utilização de dispositivos IoT e de sistemas IoT é cada vez maior, conforme se pode verificar através do gráfico ilustrativo da Figura 1 (iot-analytics, 2020), onde é visível o aumento de conexões ativas nos últimos anos e do que é expetável num futuro próximo.

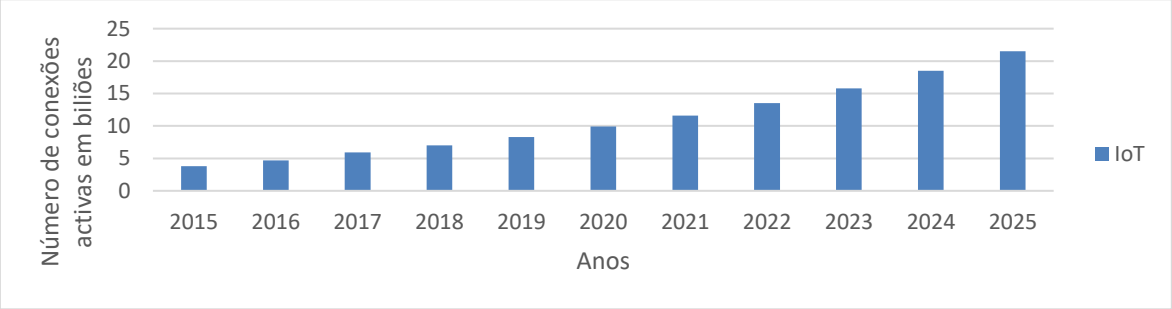

Figura 1 - Tendência do número de conexões ativas de dispositivos IoT

Estes tipos de dispositivos simples apresentam baixas capacidades de processamento, de memória e de armazenamento, contudo, a sua aplicabilidade no quotidiano da sociedade é relativamente importante. A título de exemplo, na área da domótica residencial, um termómetro é considerado um dispositivo simples, o qual executa uma tarefa simples como a verificação da temperatura numa habitação.

Visto desta forma, o conceito IoT não aparenta ter grande interesse, contudo, um dispositivo IoT integrado numa rede de vários dispositivos IoT, que recolhem vários tipos de informações, podem ser bastantes funcionais e interessantes. O mesmo dispositivo, mencionado anteriormente, conectado com outros dispositivos simples (por exemplo, dispositivos que verificam o grau de humidade e de luminosidade), podem interliga-se, criando assim um sistema IoT, agilizando a recolha de informações e automatizando na execução de ações que permitem ajustar a temperatura da referida habitação, com o comando para baixar as persianas, entre outros exemplos.

Portanto, num sistema IoT, as funções básicas dos dispositivos IoT passam pelo envio de informações para um dispositivo central IoT, que tem a capacidade de processamento e/ou

de tomada de decisão, e enviar, se for necessário, as informações recolhidas para um operador, fazendo deste ponto de vista o conceito IoT bastante interessante.

Os *standards* e os protocolos utilizados nos sistemas IoT devem ser simples e ir de encontro à simplicidade dos dispositivos IoT. Os *standards* e os protocolos devem ser simples para que não existam entropias entre estes e os dispositivos IoT, uma vez que, por exemplo, não é adequado interligar dispositivos simples a *standards* ou protocolos complexos, já que, esta interligação tem como consequência o aumento do consumo de recursos por parte dos dispositivos IoT, causando, assim, problemas de recolha de informação por parte dos dispositivos ou mesmo a sua rápida falência energética (Al-Fuqaha, et al. 2015).

Um exemplo de um protocolo utilizado em IoT é o *Message Queuing Telemetry Transport* (MQTT) (mqtt.org, 2020). Este protocolo foi desenvolvido para as comunicações simples entre máquinas, *Machine to Machine* (M2M), no entanto, é bastante utilizado em sistemas IoT (Sain, et al. 2017), já que funciona com mensagens simples de pedido e de resposta, as quais exigem pouco processamento aos vários dispositivos interligados.

A crescente utilização de sistemas IoT origina um aumento de problemas associados à falta de segurança por parte destes (bankinfosecurity, 2020). A falta de segurança pode levar a que um atacante substitua um dispositivo IoT credível por um dispositivo IoT adulterado. Este dispositivo adulterado pode autenticar-se num sistema IoT e, consequentemente, gerar ataques ao sistema, isto é, e a título de exemplos: indisponibilidade do sistema IoT, envio de informações confidenciais para outro dispositivo não autorizado e envio de informações erradas para o sistema IoT (Valério, 2015). Outro problema de falta de segurança, relacionado com os exemplos mencionados, é a possibilidade de serem tomadas decisões ou ações erradas por parte do dispositivo central IoT ou dos utilizadores, devido, uma vez mais, à adulteração de informação (Atzori, et al. 2010).

Para amenizar a falta de segurança da informação foi desenvolvida uma variante de segurança do MQTT denominada de *Message Queuing Telemetry Transport over Transport Layer Security* (MQTTS). Esta variante de segurança consiste na adição do *Transport Layer Security* (TLS) ao tráfego gerado pelo MQTT e permite garantir a este tráfego um sistema de autenticação, de autorização e de auditoria (Al-Fuqaha, et al. 2015). O MQTTS permite, ainda, resolver outros problemas, como o envio de informações confidenciais para outro dispositivo não autorizado, com a encriptação das informações trocadas.

Uma outra forma de atenuar os problemas de segurança é através da monitorização e deteção de anomalias do tráfego trocado entre os dispositivos IoT com recurso a uma ferramenta do tipo *Intrusion Detection System* (IDS) (Zarpelão & Miani, 2017). Este tipo de ferramentas necessitam de possuir um conjunto de parâmetros pré-carregados para efetuar a referida monitorização e deteção, sendo que, também analisam e fornecem informações sobre o estado de todo o tráfego e emitem alertas para situações de anomalia.

1.1.Motivação e objetivos

Como referido, no quotidiano é cada vez mais usual o recurso a ambientes IoT, sendo que os problemas de segurança associados são cada vez mais objeto de preocupação e de estudo.

Para dar respostas a questões como o consumo de recursos (processamento, memória, armazenamento e consumo de energia) e de segurança, já foram desenvolvidos alguns trabalhos de forma a definir adequados *standards* ou protocolos para dispositivos e sistemas IoT.

Ainda em relação às questões de segurança e respetivas medidas de mitigação, atualmente ainda não existe uma solução consensual de IDS adaptado especificamente para os sistemas IoT. As soluções existentes de IDS apresentam problemas de adaptação para estes tipos de sistemas, devido ao constante surgimento de novas ameaças, ataques e vulnerabilidades, quer internas ou externas, assim como, devido a uma ineficiente gestão dos recursos dos dispositivos IoT (Al-Fuqaha, et al. 2015).

Deste modo e de forma a mitigar os problemas de segurança a que estes tipos de sistemas estão sujeitos, são definidos como principais objetivos do presente projeto, a elaboração de especificações e a caracterização do tráfego MQTT e MQTTS para uma arquitetura proposta. Esta arquitetura proposta tem como função a deteção, em tempo útil, de intrusões para um sistema IoT com origem interna ou externa. Sendo que a deteção é efetuada através da análise dos registos de fluxos do tráfego *Internet Protocol Flow Information Export* (IPFIX), gerados num cenário de testes, na qual os mesmos fluxos também são utilizados para validar a arquitetura proposta.

Assim, estes principais objetivos são uma tentativa de mitigar os problemas existentes na deteção de intrusões em sistemas IoT, de modo a contribuir para o aumento da segurança

nestes sistemas. De maneira a alcançar os objetivos principais, é necessário definir objetivos complementares como:

- analisar publicações científicas, teses, dissertações e os *standards* e protocolos relacionados com os ambientes IoT, na qual estão incluídos a análise das vulnerabilidades de seguranças dos sistemas IoT e as soluções existentes para a deteção de intrusões para estes sistemas;
- analisar os registos de fluxos de tráfego IP gerados num ambiente IoT controlado;
- criar especificações do tráfego, de forma a classificar os registos de fluxos de tráfego IP como normais ou anormais;
- elaborar testes de validação, de funcionamento e de desempenho da arquitetura proposta.

1.2.Metodologia de trabalho

Com o intuito de alcançar as motivações e os objetivos descritos em 1.1, a metodologia de trabalho consistiu em dividir o trabalho de projeto em duas partes. Numa primeira parte, teórica, com a finalidade de desenvolver uma revisão de literatura sobre o contexto IoT e as questões de segurança associadas, e numa segunda parte, mais prática, com a finalidade de desenvolver as especificações para a arquitetura proposta.

Deste modo, na parte teórica fez-se um estudo através de publicações científicas, de teses, de dissertações e de *standards* e protocolos sobre o tema do presente trabalho. O estudo consistiu em analisar os tópicos existentes sobre os dispositivos e sistemas IoT. Um dos tópicos abordados foi o da segurança IoT, os problemas de segurança associados, bem como, as contramedidas existentes, para mitigar os problemas de segurança. Nesta parte foram também estudados os IDS existentes, com a finalidade de verificar a necessidade de contributos nesta área.

Posteriormente, procedeu-se à parte prática, na qual se definiu um cenário de testes com o objetivo de definir e validar as especificações definidas para a arquitetura proposta. Numa primeira fase, foram criados registos de fluxos de tráfego IP normal, de forma a obter registos de tráfego IP sem anomalias, para se conseguir fazer uma análise do tráfego MQTT e MQTTS, e desta forma criar especificações para este tipo de tráfego. Numa segunda fase, foram criados registos de fluxos de tráfego IP com a introdução de tráfego anormal, com o

objetivo de validar as especificações criadas na primeira fase. No final, foram criados e efetuados testes, de forma a obter resultados dos testes de deteção baseados nas especificações criadas na primeira fase.

1.3.Estrutura do documento

Este trabalho de projeto está dividida em sete capítulos, com o intuito de facilitar o melhor manuseamento e entendimento deste trabalho de projeto.

No presente capítulo são descritas as motivações, os desafios e a bordagem deste trabalho de projeto.

No capítulo 2 é apresentado o estado atual dos sistemas IoT, isto é, os principais conceitos a eles associados, os diferentes protocolos aplicacionais a utilizar neste trabalho de projeto, e os problemas de segurança associados, bem como as medidas que podem ser tomadas para resolver estes mesmos problemas.

No capítulo **Erro! A origem da referência não foi encontrada.** é apresentada a análise dos diferentes IDS existentes e, posteriormente, é efetuada uma análise dos vários estudos e dos trabalhos realizados na área dos sistemas IDS para um sistema IoT.

No capítulo 4 encontra-se descrita a nossa proposta de um IDS para sistemas IoT. Também são apresentados os registos de fluxos e os IE que serão exportados, para posterior execução e análise do tráfego MQTT e MQTTS.

No capítulo 5 demostra-se o funcionamento do protótipo proposto, seguindo-se a descrição das especificações para os registos de fluxos de tráfego normal MQTT e MQTTS, terminando com a apresentação de um plano de testes.

No capítulo 6 podem-se encontrar os resultados dos testes previamente planeados no capítulo anterior, seguindo-se a análise e discussão dos resultados.

Por fim, no Capítulo 7, encontram-se as conclusões, as principais contribuições e propostas de trabalhos futuros neste âmbito.

2. Internet das Coisas

O conceito IoT é relativamente recente, a primeira referência ao conceito foi utilizado por Kevin Ashton (Valério, 2015), quando trabalhava na *Procter & Gamble* (P&G), onde criou uma rede de objetos capazes de ligarem-se entre si usando a tecnologia de *Radio-Frequency IDentification* (RFID) (Atzori, et al. 2010).

Um sistema IoT tem muitas aplicações nas diversas áreas do quotidiano, como por exemplo, na área da domótica de uma habitação, na área da saúde e bem-estar e na área da indústria 4.0 (CNCS, 2020), isto é, na automatização da temperatura de uma casa, na medicação de insulina num paciente e na automatização de tarefas numa fábrica, respetivamente.

Os sistemas IoT têm a capacidade de permitir que um dispositivo simples se transforme num dispositivo que tenha a capacidade de comunicar com outros dispositivos simples através de uma rede, criando assim a possibilidade de comunicarem entre si e tomarem decisões de forma autónoma (Al-Fuqaha et al. 2015).

A necessidade da existência de protocolos e *standards*, que permitem a comunicação e a compatibilidade de diferentes dispositivos IoT, é cada vez mais essencial, já que se verifica um aumento na utilização destes tipos de dispositivos no dia-a-dia.

Em paralelo, as questões de privacidade e de segurança estão cada vez mais em foco, tendo uma maior importância e relevo, sendo que, as mesmas questões deixaram de ter respostas e soluções opcionais e facultativas, passado a ser imperativo fornecer respostas e soluções concretas e obrigatórias.

2.1. Paradigmas IoT

Existem vários paradigmas de como é explicado o conceito de IoT, sendo que, por isso, o mesmo é um conceito não concreto e, como tal existe dificuldade em definir o que é o IoT.

O artigo de Atzori, et al. (2010) aborda os desafios dos sistemas IoT e a consequente dificuldade em definir o que é o IoT. No artigo, é definido o conceito IoT através de três paradigmas, isto é: direcionado à Internet, direcionado às Coisas e direcionado à Semântica.

A Figura 2, adaptada de Atzori, et al. (2010), ilustra os três paradigmas e de como os mesmos convergem formando o conceito de IoT.

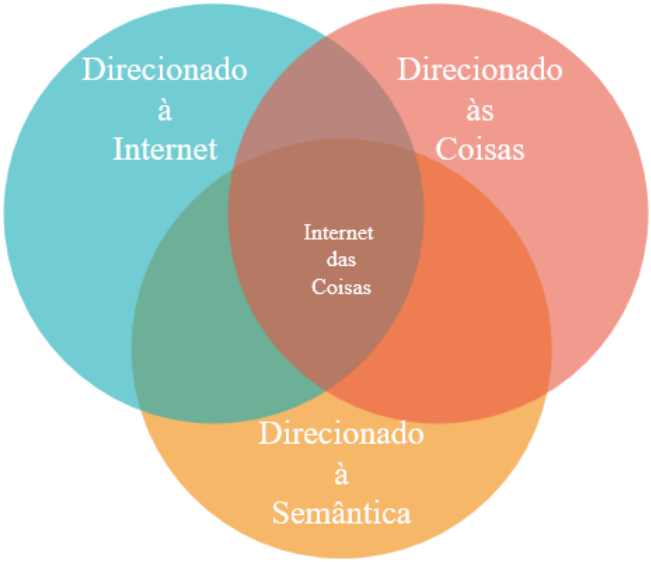

Figura 2 - IoT convergência de paradigmas

De seguida, de acordo com o referido artigo (Atzori, et al. 2010), são explicados sucintamente os três paradigmas.

2.1.1. Direcionado à Internet

Do ponto de vista da aliança IP *of Smart Objects* (IPSO), organização sem fins lucrativos formada em setembro de 2008, esta fundamenta-se de que a existência da pilha *Internet Protocol* (IP), que é um protocolo leve, pode ser utilizada para fazer a ligação entre todos os dispositivos em sistemas embebidos, já que são dispositivos de baixo consumo energético, de processamento e de memória *Random-Access Memory* (RAM) e de *Read-Only Memory* (ROM).

Outra abordagem é a do protocolo Internet Ø, desenvolvido pelo *Massachusetts Institute of Technology* (MIT) *Center for Bits and Atoms*, que refere que a mesma pilha IP deve ser também utilizada, no entanto, com a diferença que a sua complexidade deve ser reduzida, através de um protocolo chamado IP *Over Anything*.

Com base nas referências de abordagem descritas acima (IPSO e Internet Ø), pode-se dizer que um sistema IoT deve ser simplificado a uma pilha IP, para que todos os dispositivos sejam endereçáveis e acessíveis em qualquer localização.

2.1.2. Direcionado às Coisas

O referido artigo considera que os dispositivos, "coisas", são objetos simples e, por isso, podem conter etiquetas RFID ou *Near Field Communication* (NFC). Os mesmos dipositivos também têm de se basear numa arquitetura *ubiquitous IDdentifer* (uID), para que esta arquitetura ajude a identificar cada dispositivo de forma concreta e única.

2.1.3. Direcionado à Semântica

No que se refere à semântica, o artigo de Atzori, et al. (2010) refere que a organização dos dados, as ligações e as pesquisas geradas pelos dispositivos IoT são um problema, já que, os dispositivos IoT geram muita informação desorganizada. No entanto, o mesmo artigo refere que estes problemas podem ser resolvidos através de tecnologias semânticas, como: o *Resource Description Framework* (RDF), o *Web Ontology Language* (WOL) e o *Efficient XML Interchange* (EXI), sendo que, em 2011, o *World Wide Web Consortium* (W3C) adaptou o EXI como a tecnologia semântica recomendada.

2.2. Arquitetura IoT

Um sistema IoT contém muitos e variados dispositivos IoT (com diferentes processadores, diferentes capacidades de memória e de armazenamento e diferentes finalidades) ligados entre si, sendo que por isso, é necessária a existência de uma arquitetura que abranja esta heterogeneidade de dispositivos.

De acordo com o artigo Lin, et al. (2017), um sistema IoT é composto por três camadas, nomeadamente: a camada de aplicação, a camada de rede e a camada de perceção.

Contudo, o número de camadas exatas não é consensual entre vários estudos já que, de acordo com outro artigo Al-Fuqaha, et al. (2015), um sistema IoT é composto por um modelo de arquitetura de cinco camadas, isto é: a camada de negócio, a camada de aplicação, a camada de gestão de serviços, a camada de abstração do objeto e a camada do objeto.

Apesar desta discordância relativa ao número de camadas, pode-se dizer que estes dois artigos estão interligados e em acordo já que, o modelo de arquitetura de três camadas está englobado no modelo de arquitetura de cinco camadas, como é demostrado na Figura 3.

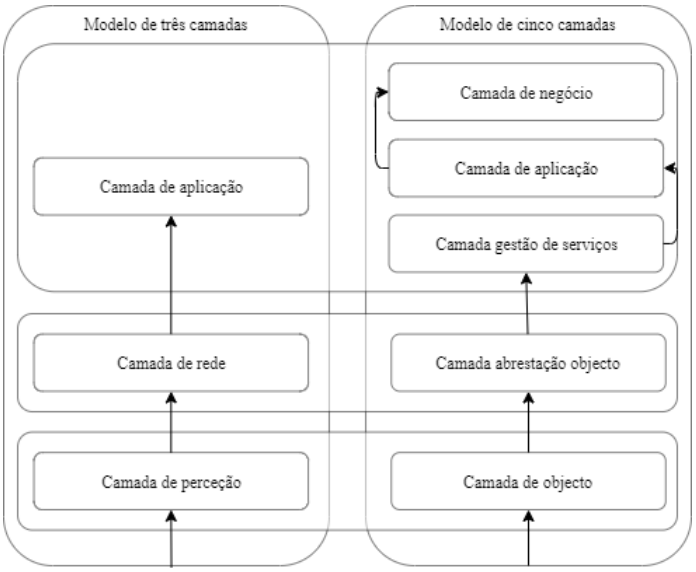

Figura 3 - Comparativo entre o modelo três e cinco camadas

De seguida, descreve-se o modelo de arquitetura de cinco camadas, na qual ao longo da descrição se faz referência ao modelo de arquitetura de três camadas, nomeadamente onde este está enquadrado no modelo de cinco camadas, sendo que a Figura 3 também auxilia na compreensão da referida descrição.

2.2.1. Camada de perceção

A camada de objeto (modelo de cinco camadas) ou a camada de perceção (modelo de três camadas) representa fisicamente o sensor e o atuador, isto é, os dispositivos que detetam informações na qual estão expostos, a título de exemplo, a temperatura ambiente. Para configurar estes dispositivos poderá ser necessário recorrer a mecanismos de *plug-and-play*. A camada de objeto ou camada de perceção está ligada à camada abstração objeto (modelo de cinco camadas) ou à camada de rede (modelo de três camadas) através de canais seguros.

2.2.2. Camada abstração objeto

A camada abstração objeto (modelo de cinco camadas) ou a camada de rede (modelo de três camadas) tem a finalidade de transferir informações, recolhidas pelos dispositivos, da camada objeto para a camada de gestão de serviços, no modelo de cinco camadas. Esta transferência é efetuada através de tecnologias de transmissão, tais como: RFID, *Wi-Fi*, *Bluetooth Low Energy* (BLE), entre outras. Relativamente ao modelo de três camadas, a transmissão pela camada de rede é efetuada da camada de perceção para a camada de

aplicação, sendo que esta última está inserida em três camadas (gestão de serviços, aplicação, negócio) no modelo de cinco camadas.

2.2.3. Camada de gestão de serviços

Esta camada apenas descrita no modelo de cinco camadas, tem a responsabilidade de fornecer serviços necessários ao cliente, a título de exemplo prático: um dispositivo que recolhe a temperatura do local onde está inserido e envia essa informação (temperatura) para um cliente.

2.2.4. Camada de aplicação

Esta camada está descrita tanto no modelo de três camadas como no modelo de cinco camadas e permite programar os dispositivos independentemente da plataforma. Esta camada também recebe dados, toma decisões e envia os serviços requeridos, sobre protocolos de rede. Como referido em 2.2.2, no modelo de três camadas, a camada de aplicação engloba três camadas no modelo de cinco camadas, respetivamente: a camada gestão de serviços, a camada de aplicação e a camada de negócio.

2.2.5. Camada de negócio

Esta camada, apenas descrita no modelo de cinco camadas, é a última camada da arquitetura IoT e é a responsável pela gestão dos serviços e das atividades dos sistemas IoT. Esta camada recolhe toda a informação processada pelas camadas anteriores para a execução de gráficos e plataformas de monitorização que, posteriormente, são utilizados no auxílio das tomadas de decisões.

2.3.Elementos de um sistema IoT

Os elementos de um sistema IoT estão divididos em seis partes, nomeadamente: identificação, deteção, comunicação, computação, serviços e semântica (Al-Fuqaha, et al. 2015).

A Figura 4, aptada do artigo Al-Fuqaha et al. (2015), ilustra como diferentes componentes simples interligados formam um dispositivo IoT.

Figura 4 - Elementos de um dispositivo IoT

2.3.1. Identificação

Os dispositivos IoT têm uma identificação única, de forma a que seja possível a correlação entre o nome e o serviço que vai disponibilizar. O endereçamento dos dispositivos IoT é fundamental para que exista diferenciação entre a identificação do objeto (referente ao nome do dispositivo ou um identificador) e o endereço IP, que pode ser *Internet Protocol version 4* (IPv4) ou *Internet Protocol version 6* (IPv6). Este endereçamento estabelece a junção do nome do objeto com o endereço IP, na qual a junção é designada por uID.

2.3.2. Deteção

Este elemento recolhe a informação do ambiente em que está inserido e envia essa informação para um ponto centralizado. Para a recolha de informação os dispositivos IoT utilizam uma *Single Board Computer* (SBC) juntamente com componentes adicionais, como, sensores (exemplo: de temperatura, de humidade, ente outros) e uma pilha IP, que pode ou não conter elementos de segurança.

2.3.3. Comunicação

Este elemento tem a capacidade de ligar diferentes dispositivos entres si, mesmo que sejam dispositivos com diferentes *hardwares* ou com diferentes objetivos. Este elemento envia as informações entre dispositivos, sendo que o meio de comunicação tem como desafio o de consumir poucos recursos.

Existem vários protocolos que podem ser utilizados para a comunicação de dispositivos IoT, tais como: o BLE, o *Ultra-Wide-Band* (UWB), o *Wi-Fi*, o RFID e o NFC. Cada um dos protocolos apresentados tem as suas características próprias, isto é: alcance (em metros), eficiência energética, estabilidade na comunicação, segurança na comunicação e as suas aplicações, como se pode verificar através da Tabela 1 (BlueRange, 2020) (Zou, et al. 2016).

Tabela 1 - Comparativo entre meios de comunicação

Tipo	Alcance	Eficiência	Estabilidade	Segurança	Aplicação
BLE	1 a 5 m	Alta	Alta	Alta	Rede de sensores
UWB	10 a 50 m	Baixa	Alta	Alta	Rede de sensores
Wi-Fi	2 a 5 m	Baixo	Médio	Médio	Redes *ad-hoc*
RFID/NFC	1 a 3 m	Alto	Alto	Baixo	Rastreamento de objetos

2.3.4. Computação

A computação é composta por duas partes, isto é, o *hardware* e o *software*. Ao longo da evolução dos dispositivos e dos sistemas IoT foram desenvolvidas várias plataformas de *hardware*, tais como: o *Raspberry Pi*, o *Arduíno*, entre outros. Paralelamente também foram desenvolvidos sistemas operativos (*software*) com a capacidade de funcionarem sobre o *Raspberry Pi* ou o *Arduíno* de forma a fornecerem características para o IoT, por exemplo. Exemplos dos sistemas operativos desenvolvidos em exclusivamente para o sistema IoT são o TinyOS, o LiteOS e o RiotOS, e por isso têm características úteis para os ambientes IoT, tais como: a eficiência energética e o baixo consumo de recursos de memória e de processamento.

2.3.5. Serviços

Os serviços IoT são categorizados em quatro tipos: serviços de identificação, serviços de agregação, serviços de colaboração e serviços onipresentes.

- Serviços de identificação - este é o serviço mais simples e, em simultâneo, o mais importante dos quatro, por ser utilizado noutros tipos de serviços. Este serviço permite a um objeto “transporta-se do mundo real para o mundo virtual”, sendo que para ocorrer esta mudança é necessário que os objetos sejam identificados de forma única;
- Serviços de agregação - este serviço agrega e sumariza toda a informação recebida pelos dispositivos, de forma a que seja processada e reportada às aplicações IoT;
- Serviços de colaboração - este tipo de serviço situa-se acima do serviço de agregação da informação e é utlizado para tomar decisões e reagir de acordo com a informação recebida pelos dispositivos;
- Serviços onipresentes - este serviço funciona juntamente com o serviço de colaboração e o seu intuito é fornecer o serviço de colaboração a qualquer altura, a quem necessita e em qualquer lugar.

2.3.6. Semântica

Por último, a semântica. Este elemento tem a capacidade de extrair informações de diferentes elementos e fornecê-las aos serviços que requerem essas mesmas informações, sendo que também tem a capacidade de enviar pedidos a recursos.

A Tabela 2, adaptada do artigo Al-Fuqaha et al. (2015), representa os diferentes elementos IoT e respetivos exemplos em cada camada.

Tabela 2 - Elementos IoT e respetivos exemplos

Elemento IoT		**Exemplo**
Identificação	Nomeação	*Electronic Product Code* (EPC), *Universal Computer Code* (uCODE)
	Endereçamento	IPv4, IPv6
Deteção		Sensores, etiquetas RFID
Comunicação		RFID, NFC, UWB, BLE, *Wi-Fi*
Computação	*Hardware*	*Raspberry pi, Arduino*
	Software	LiteOS, RiotOS
Serviços		Rasteiro de encomendas, domótica
Semântica		RDF, WOL, EXI

2.4.*Standards* e protocolos IoT

Ao longo dos últimos anos foram desenvolvidos vários *standards* e protocolos de forma a facilitar e a simplificar a tarefa de fornecer serviços e aplicações.

Os *standards* e os protocolos mais conhecidos são: o *Data Distribution Service* (DDS), o *Constrained Application Protocol* (CoAP), o *Advanced Message Queuing Protocol* (AMQP), o *Extensible Messaging and Presence Protocol* (XMPP), o MQTT e o MQTTS. Este *standards* e protocolos utilizam como serviços de descoberta o *multicast Domain Name Server* (mDNS) e o DNS-*Based Service Discovery* (DNS-SD), que são os mais comuns.

Os protocolos de infraestrutura estão divididos em quatro pontos, eles são:

- Encaminhamento - o mais utilizado é o *Routing Protocol for Low-Power and Lossy Networks* (RPL);
- Camada de rede - os mais comuns são o *IPv6 over Low power Wireless Personal Area Network* (6LoWPAN), o IPv4 e o IPv6;
- Camada de ligação - é utilizado o *Institute of Electrical and Electronics Engineers* (IEEE) 802.15.4;

- Camada física - os mais recorrentes são o *Long Term Evolution Advanced* (LTE-A), o *Electronic Product Code* (EPCglobal), o IEEE 802.15.4 e o Z-Wave.

Sendo que, por fim, os protocolos mais comuns são: o IEEE 1888.3, o IP *Security Protocol* (IPSec) e o IEEE 1905.1.

Na Tabela 3, adaptada do artigo Al-Fuqaha, et al. (2015), descrevem-se as várias características dos *standards* e dos protocolos apresentados anteriormente.

Tabela 3 - IoT *standards* e protocolos

***Standards* e Protocolos**		**DDS**	**CoAP**	**AMQP**	**XMPP**	**MQTT**	**MQTTS**
Serviço de descoberta		mDNS			DNS-SD		
Protocolos de infraestrutura	Encaminhamento	RPL					
	Camada de rede	6LoWPAN					IPv4/IPv6
	Camada de ligação	IEEE 802.15.4					
	Camada física	LTE-A	EPCglobal	IEEE 802.15.4			Z-Wave
Protocolos influentes		IEEE 1888.3, IPSec				IEEE 1905.1	

De seguida, sumarizam-se as descrições de alguns *standards* e protocolos, descritos na Tabela 3, utilizados num sistema IoT e destaca-se o MQTT, em 2.4.5, e o MQTTS, em 2.4.6, que são alvo de estudo no presente trabalho de projeto.

2.4.1. DDS

O DDS foi desenvolvido pelo *Object Management Group* (OMG), com base na arquitetura *Publisher* e *Subscriber* para comunicações M2M. Este protocolo não é específico para sistemas IoT, no entanto, é uma boa opção como protocolo, já que não necessita de um *Broker*, ao contrário do MQTT ou do AMQP. Utiliza domínios de *multicast*, o que lhe permite uma boa qualidade de serviço e uma elevada confiança na entrega de mensagens.

2.4.2. COAP

O CoAP foi desenvolvido pelo *Internet Engineering Task Force* (IETF) *Constrained Restful Environments* (CORE) e é específico para sistemas IoT. A arquitetura do CoAP é baseada em *REpresentational State Transfer* (REST) sobre *Hypertext Transfer Protocol* (HTTP) e utiliza o protocolo de transporte *User Datagram Protocol* (UDP). Este protocolo é um protocolo simples de transferência de mensagens que recorre os métodos HTTP (GET, PUT, POST e o DELETE).

2.4.3. AMQP

O AMQP é um protocolo *open standard* para sistemas IoT, na qual é orientado para ambiente de troca de mensagens. Este tipo de protocolo requer que o protocolo de comunicação seja o TCP, com base na arquitetura *Publisher* e *Subscriber*.

2.4.4. XMPP

O XMPP não é um *standard* desenvolvido especificamente para sistemas IoT, no entanto, é largamente utilizado para esse fim. É um *standard* do IEFT baseado em mensagens instantâneas, em inglês *Instant Messaging* (IM), que frequentemente é utilizado em ferramentas de conversação, tanto de texto, como de voz ou de vídeo conferência. É um protocolo de código aberto com o objetivo de fornecer um método de mensagens aberto, seguro, descentralizado e independente do Sistema Operativo (SO) utilizado, sendo que utiliza o *Extensible Markup Language* (XML).

2.4.5. MQTT

Em 1999, Andy Stanford Clak da *International Business Machines Corporation* (IBM) e Arlen Nipper da Arcon, agora designada como EuroTech, começaram a desenvolver o conceito MQTT sendo que, em 2013, o seu *standard* foi definido pela *Organization for the Advancement of Structured Information Standards* (OASIS). O MQTT é um protocolo de mensagens simples que permite ligar dispositivos embebidos a redes com aplicações e serviços. Não sendo criado especificamente para dispositivos e sistemas IoT, ele é bastante utilizado, devido à sua simplicidade e baixo consumo de recursos (Al-Fuqaha, et al. 2015).

As operações do MQTT utilizam mecanismos de encaminhamento (um para um, um para muitos ou muitos para muitos) e permitem que o MQTT estabeleça ligações ideais entre dispositivos e sistemas IoT, com o M2M a funcionar sobre o protocolo *Transmission Control Protocol* (TCP) e com o porto por defeito de 1883.

O MQTT é baseado na arquitetura *Publisher* e *Subscriber*. Fornece uma comunicação simples e flexível, sendo, portanto, útil para dispositivos que tenham baixos recursos ou para ligações com baixas larguras de banda.

O MQTT é composto por três elementos: o *Publisher*, o *Subscriber* e o *Broker*:

- *Publisher* - é um dispositivo, por exemplo um sensor ou um atuador, que tem associado a ele um tópico, na qual transmite informações ao *Broker*;
- *Subscriber* - é um dispositivo que subscreve um tópico, sendo que após a subscrição do tópico recebe informações sobre esse tópico pelo *Broker*;
- *Broker* - é um dispositivo que faz a ligação entre o *Publisher* e o *Subscriber*. Este dispositivo recebe as informações do *Publisher* e envia para o *Subscriber*, sendo que tem a possibilidade de automatizar tarefas, tomar decisões e de guardar numa base de dados as informações provenientes do *Publisher*.

A Figura 5 ilustra o funcionamento do protocolo MQTT entre os três elementos.

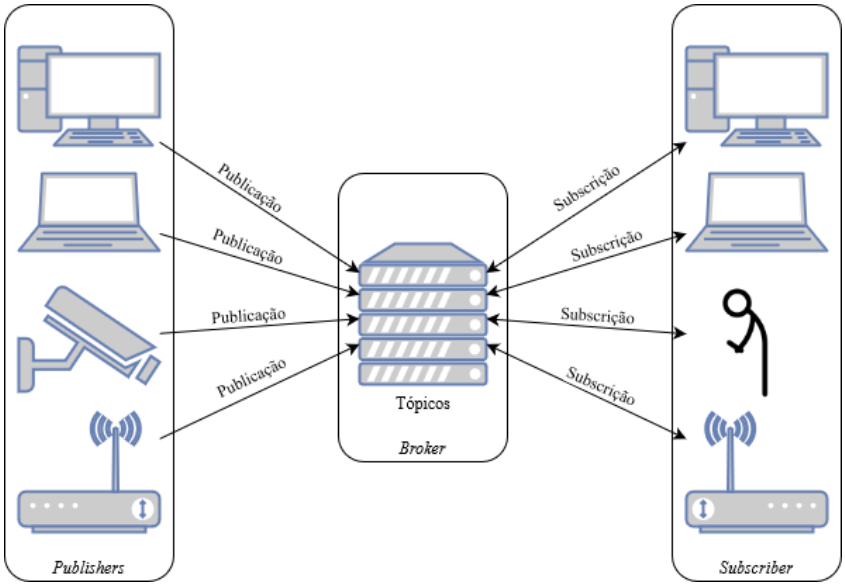

Figura 5 - Protocolo MQTT

É de notar que a comunicação é unidirecional entre o *Publisher* e o *Broker*, no entanto, a comunicação entre o *Broker* e o *Subscriber* (ou *Subscribers*) é bidirecional, já que o *Subscriber* pode enviar ordens com base nas informações recebidas do *Publisher* para o *Broker*, como é demostrado na Figura 6, que representa a troca de mensagens entre os vários elementos do protótipo MQTT.

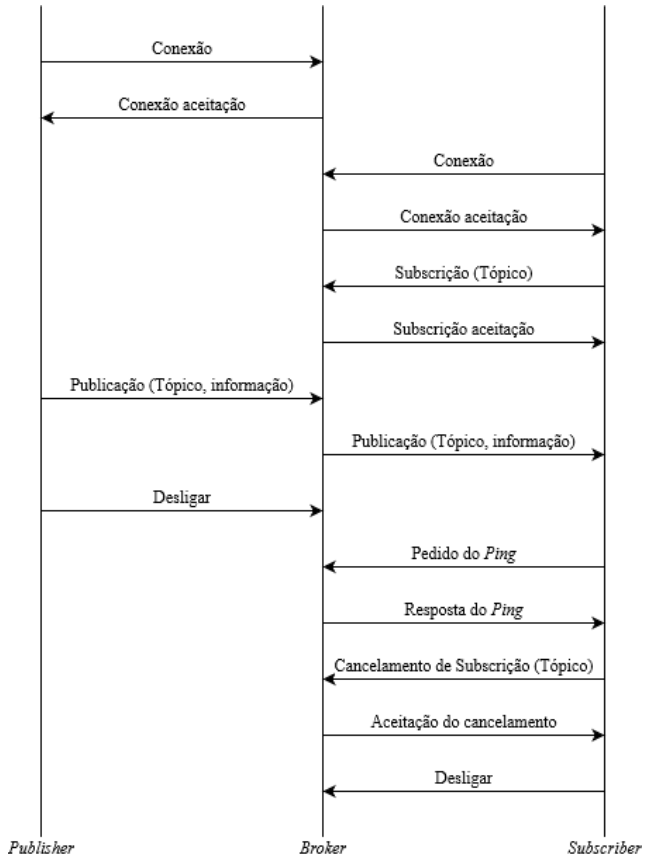

Figura 6 - MQTT fluxo de mensagens

Características do pacote MQTT

O pacote MQTT identifica o tipo de mensagens trocadas entres elementos, na qual a sua estrutura é constituída por três partes: o cabeçalho fixo, o cabeçalho variável e o *payload*. O pacote MQTT tem o tamanho mínimo de dois *bytes*, sendo que o primeiro *byte* define o tipo de mensagem MQTT, o *Duplicate and Damaged Packets* (DUP), o nível de *Quality of Service* (QoS) e a *flag* de *RETAIN*, e o segundo e seguintes *bytes* incluem mensagens MQTT que contêm a parte variável, por exemplo, a publicação de uma mensagem ou a subscrição de um tópico. A Figura 7 representa o pacote MQTT, onde se descreve cada um dos elementos.

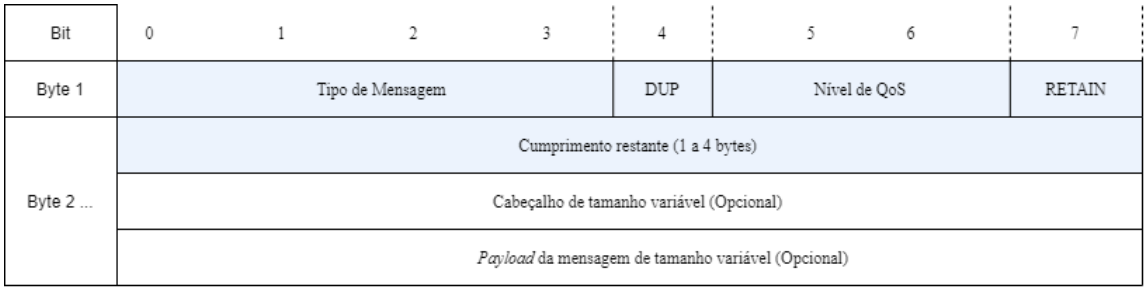

Figura 7 - Pacote MQTT

De seguida, são explicadas as características presentes no primeiro *byte*.

- Tipo de mensagem - existem vários tipos de mensagens. A Tabela 4, adotada de Koneski (2018), apresenta os vários tipos de mensagens existentes, o seu valor, a sua direção do tráfego e a sua descrição, respetivamente.

Tabela 4 - MQTT tipo de mensagem

Tipo	Valor	Direção	Descrição
Reservado	0	Não disponível	Reservado
CONNECT	1	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*	O *Publisher* ou o *Subscriber* pede para ligar com o *Broker*
CONNACK	2	*Broker* para o *Publisher* ou do *Broker* para o *Subscriber*	O *Broker* aceita a ligação vinda do *Publisher* ou do *Subscriber*
PUBLISH	3	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*. Ou do *Broker* para o *Publisher* ou do *Broker* para o *Subscriber*	Publica a mensagem
PUBACK	4	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*. Ou do *Broker* para o *Publisher* ou do *Broker* para o *Subscriber*	Aceitação da mensagem publicada
PUBREC	5	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*. Ou do *Broker* para o *Publisher* ou do *Broker* para o *Subscriber*	Publicação recebida
PUBREL	6	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*. Ou do *Broker* para o *Publisher* ou do *Broker* para o *Subscriber*	Publicação recebida
PUBCOMP	7	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*. Ou do *Broker* para o *Publisher* ou do *Broker* para o *Subscriber*	Publicação recebida
SUBCRIBE	8	*Subscriber* para o *Broker*	O *Subscriber* faz um pedido de subscrição de tópico
SUBACK	9	*Broker* para o *Subscriber*	Subscrição aceite
UNSUBSCRIBE	10	*Subscriber* para o *Broker*	Deixar de um subscrever um tópico
UNSUBACK	11	*Broker* para o *Subscriber*	Aceitação de deixar de subscrever um tópico
PINGREQ	12	*Subscriber* para o *Broker*.	Pedido de um *ping*
PINGRESP	13	*Broker* para o *Subscriber*	Resposta do pedido do *ping*
DISCONNECT	14	*Publisher* para o *Broker* ou do *Subscriber* para o *Broker*.	*Publisher* ou *Subscriber* desliga-se do *Broker*
Reservado	15	Não disponível	Reservado

- DUP - o campo DUP indica se o pacote é duplicado ou foi corrompido;
- Nível de QoS - existem três níveis de QoS:
    - QoS 0 - o mecanismo de QoS está desligado, sendo que a mensagem é entregue com o melhor esforço possível (*best effort*);
    - QoS 1 - confirma a entrega da mensagem, mesmo quando a mesma é enviada várias vezes;
    - QoS 2 - confirma que recebe apenas uma mensagem.
- *RETAIN* - informa o *Broker* para reter a última mensagem recebida e envia essa mensagem para os novos *Subscribers*, como se fosse a primeira mensagem.

2.4.6. MQTTS

A falta de segurança no MQTT levou ao desenvolvimento de uma variante de segurança, denominada de MQTTS. Esta variante de segurança é a adição da camada TLS ao tráfego MQTT e permite garantir ao tráfego gerado um sistema de autenticação, de autorização e de auditoria (Al-Fuqaha, et al. 2015). Esta solução é chamada de MQTTS, mas também é conhecida por MQTT *over* TLS (HiveMQ, 2020).

O TLS funciona sobre o TCP, de maneira a garantir uma troca segura de dados entre os intervenientes (Koneski, 2018).

A Figura 8 mostra o formato da mensagem TLS.

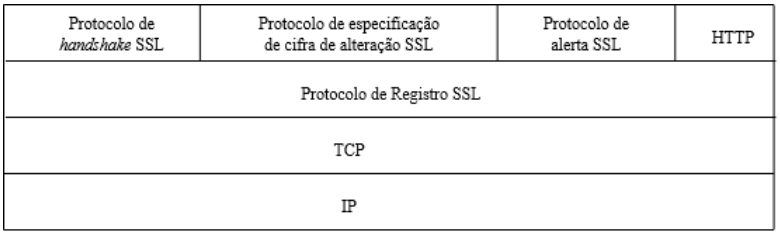

Figura 8 - Mensagem TLS

Os protocolos fundamentais do TLS são os descritos abaixo:

- Protocolo de *handshake Secure Sockets Layer* (SSL) - é o *handshake* entre o cliente e o servidor;
- Protocolo de especificação de cifra de alteração SSL - informa se existem alterações na ligação;

- Protocolo de alerta SSL - se existir algum tipo de problema, como por exemplo, uma cifra estar comprometida, a ligação é terminada.

A Figura 9 mostra como é efetuada a troca de mensagens do TLS entre os intervenientes, o servidor TLS e o cliente TLS. No MQTT, o *Broker* é o servidor e o cliente tanto pode ser o *Publisher* ou o *Subscriber*. De uma forma genérica, o fluxo de dados da aplicação é a mensagem MQTT sobre TLS e, se a camada de TLS for removida, somente o fluxo de dados da aplicação é o MQTT.

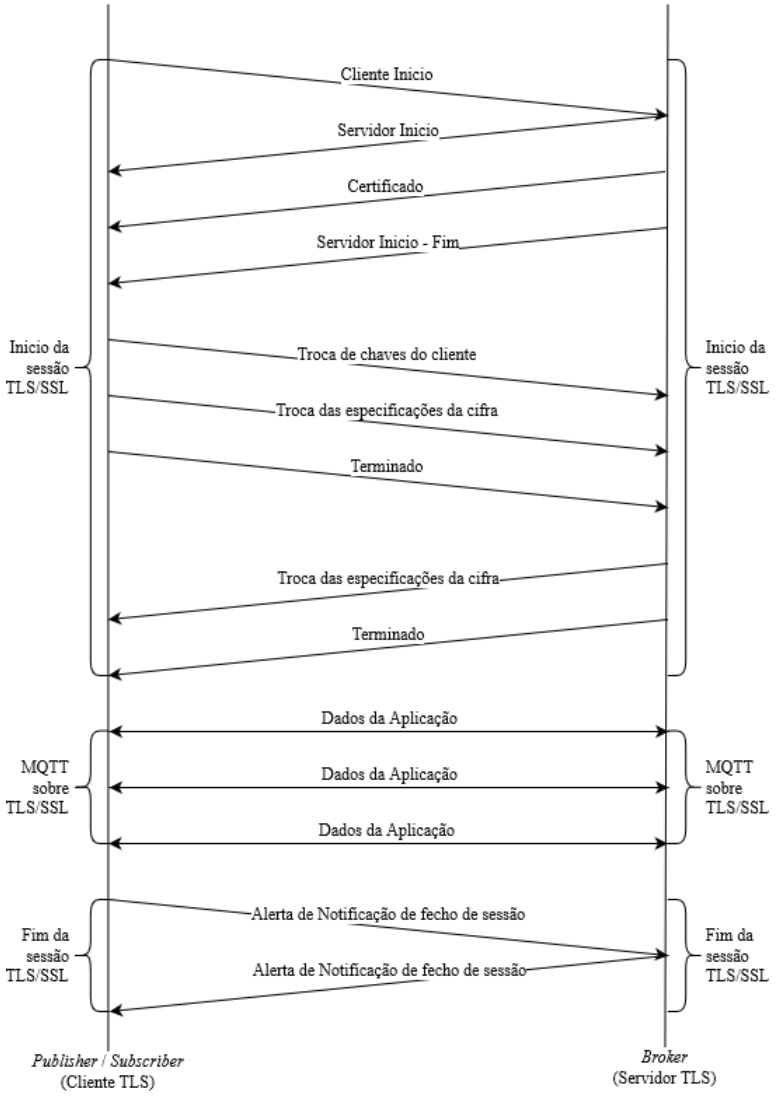

Figura 9 - Troca de mensagens MQTTS

2.5.Segurança IoT

As questões de segurança, que incluem a privacidade e a integridade, têm cada vez mais importância, já que se verifica um aumento de ataques e vulnerabilidades detetadas

(bankinfosecurity, 2020, infosecurity-magazine, 2020, darkreading, 2020). Esta situação é ainda agravada pelo facto de um dispositivo ou um sistema IoT, por si só, não ter uma camada de segurança, o que pode aumentar ainda mais os ataques e as vulnerabilidades.

As questões de segurança usualmente são abordadas em três categorias: confidencialidade, integridade e disponibilidade. Estas três categorias também são conhecidas por *Confidentiality, Integrity and Availability* (CIA) (Cherdantseva & Hilton, 2013), contudo, esta abordagem apresenta falhas quando está a lidar com novas vulnerabilidades ou ataques.

Assim, para que existem garantias de segurança no momento do desenvolvimento de um sistema IoT, são necessários cumprir requisitos, tais como (Al-Fuqaha, et al. 2015) (Sain, et al. 2017): a confidencialidade, a integridade, a disponibilidade, a identificação e a autenticação e a privacidade, a seguir descritos.

- Confidencialidade - a informação apenas está disponível para entidades, serviços ou utilizadores autorizados. O desafio no contexto IoT é a quantidade de dispositivos e de sistemas que podem ser integrados, já que, a confidencialidade é importante e é uma questão complexa, devido à heterogeneidade dos sistemas IoT;
- Integridade - garantir que a informação entre os elementos de um sistema IoT não seja adulterada. Num sistema IoT, devido aos dispositivos estarem exposto a ambientes exteriores, é fácil a sua adulteração, já que é possível remover um dispositivo de forma não autorizada e substituir o *firmware* desse dispositivo, e voltar a introduzir no sistema de forma a enviar a informação errada;
- Disponibilidade - significa que a informação e os dispositivos estão disponíveis para os utilizadores e para os serviços autorizados. O desafio num sistema IoT é a disponibilidade da informação ser apresentada em tempo real;
- Identificação e autenticação - a identificação é a garantia que apenas os dispositivos autorizados podem ligar-se ao sistema. A autenticação é a garantia de que apenas os dispositivos autorizados podem enviar informações e receber informações. Num sistema IoT, a identificação e a autenticação dos dispositivos são questões complicadas, já que, um sistema IoT pode ter muitos e variados dispositivos, sendo que é necessário que existam mecanismos que identifiquem, de forma única, um dispositivo e autenticá-lo;
- Privacidade - num contexto IoT é importante garantir que a informação recebida e enviada são visualizadas pelos serviços ou utilizadores correspondentes, sendo

assim, é necessário garantir que a informação seja apenas vista e manipulada pelo utilizador correspondente;

- Confiança - combinação entre a confidencialidade, a integridade, a disponibilidade, o controlo de acesso e a privacidade ao longo dos diferentes elementos de um sistema IoT.

De modo geral, as vulnerabilidades e os ataques são direcionados para as três camadas da arquitetura IoT (camada de perceção, camada de rede e camada de aplicação) (Al-Fuqaha, et al. 2015), sendo que, podem afetar uma ou várias camadas ao mesmo tempo. De seguida, descrevem-se, em tabelas, as vulnerabilidades e os ataques, assim como as respetivas defesas para cada uma das três camadas.

2.5.1. Camada de perceção

Na camada de perceção, que tem a responsabilidade de recolher informações, a informação recolhida é facilmente adulterada, já que esta está normalmente exposta a ambientes exteriores.

A Tabela 5 descreve as vulnerabilidades ou os ataques desta camada, assim como as respetivas soluções de defesa. (Al-Fuqaha, et al. 2015).

Tabela 5 - Camada de precessão vulnerabilidades, ataques e contramedidas

Vulnerabilidade ou Ataque	Descrição	Defesa
Captura do dispositivo	Retirada ilegal do dispositivo e a substituição por outro dispositivo modificado	Monotorização e deteção de manipulação dos dispositivos
Injeção de código adulterado	Captura de um dispositivo e a instalação ou modificação do *firmware*	Gestão automática do *software* dos dispositivos
Injeção de informação errada	Após a captura de um dispositivo ou alteração do *firmware* pode fazer com que este dispositivo envie informação errada	Deteção e ignorar da informação enviada pelo dispositivo afetado
Criptoanálise e canal lateral	Forma de obter a chave de autenticação das mensagens trocadas entre os dispositivos IoT	Algoritmos de encriptação e gestão de chaves
Escutas e interferências	Tentativas de escutar o tráfego e ou introduzir interferências no tráfego entre os dispositivos IoT.	Filtragem do ruído, de forma a filtrar a informação de ruído e restaurar os dados originais
Ataque à bateria do dispositivo	Fazer com que os dispositivos fiquem ativos o maior tempo possível, de forma a consumir a bateria	Monotorização e deteção de manipulação dos dispositivos

2.5.2. Camada de rede

O objetivo desta camada é enviar informação recolhida, na maior parte dos casos através de uma rede sem fios.

A Tabela 6 descreve as vulnerabilidades ou os ataques desta camada, assim como as respetivas soluções de defesa. (Al-Fuqaha, et al. 2015).

Tabela 6 - Camada de rede vulnerabilidades, ataques e contramedidas

Vulnerabilidade ou Ataque	Descrição	Defesa
Ataques *Denial Of Service* (DoS)	Um ou vários terminais atacam de forma coordenada um dispositivo de formar a esgotar os seus recursos	IDS para poder identificar padrões.
Ataques de *Spoofing*	Ter o total acesso a um dispositivo IoT e enviar informação errada	Gestão de acessos, autenticação e identificação dos utilizadores
Ataques *Sinkhole*	Um dispositivo malicioso faz com os dispositivos vizinhos enviem todo tráfego para o dispositivo malicioso. Esse dispositivo malicioso envia o tráfego para outro ponto não autorizado	Robustez dos protocolos de encaminhamento
Ataques *Wormhole*	Dois dispositivos malicioso separados fisicamente, trocam informação numa ligação privada. Fazendo acreditar aos outros dispositivos que os dois nos são fidedignos. No final o objetivo é similar ao ataque *Sinkhole*.	Robustez dos protocolos de encaminhamento
Ataques *Man in the middle*	Um atacante escuta a comunicação entre dispositivos fazendo o tráfego seja encaminhado para um ponto não autorizado	TLS
Ataques de encaminhamento	Modificação do protocolo de encaminhamento entre os dispositivos IoT	Robustez dos protocolos de encaminhamento e TLS
Ataques *Sybil*	Um dispositivo malicioso é introduzido na rede com vários perfis. De forma que o outros dispositivos aceitem na rede.	Mecanismos de identificação e autenticação
Acessos não autorizados	Um dispositivo malicio tem acesso não autorizado ao dispositivo	Mecanismos de identificação e autenticação

2.5.3. Camada de aplicação

Esta camada fornece serviços que são necessários e requisitados pelos utilizadores.

A Tabela 7 descreve as vulnerabilidades ou os ataques desta camada, assim como as respetivas soluções de defesa. (Al-Fuqaha, et al. 2015).

Tabela 7 - Camada aplicação, vulnerabilidades, ataques e contramedidas

Vulnerabilidade ou Ataque	Descrição	Defesa
Ataques de *phishing*	Um atacante obtém dados confidenciais através de site similar a site fidedignos	Vigilância dos utilizadores
Vírus, *Worns*	Instalação de aplicações maliciosa nos dispositivos	Verificação de antivírus, instalação de *firewall* e IDS
Script maliciosos	Alteração *firmware* dos dispositivos	Mecanismos de deteção de código modificado

2.6.Síntese

No presente capítulo, destaca-se o conceito e os paradigmas do IoT, sendo que, a visão de divergência de vários autores, de acordo com os seus estudos de investigação, permite que este tema não tenha uma definição concreta.

De acordo com estes estudos de investigação, são ainda descritos dois modelos de arquitetura para um sistema IoT: um modelo composto por três camadas e outro composto por cinco camadas. Salienta-se que, o modelo de três camadas está englobado no modelo de cinco camadas e, para a finalidade deste trabalho de projeto, o modelo de três camadas é o mais ajustado e o mais comum.

Este capítulo faz também referência aos vários *standards* e protocolos para os sistemas IoT, com destaque para os protocolos MQTT e MQTTS, já que são objeto de estudo do presente trabalho. Estes protocolos, apesar de não serem específicos para sistemas IoT, são largamente utilizados, devido às suas arquitetura simples M2M e aos seus baixos consumos de recursos.

O MQTT e o MQTTS é o mesmo protocolo aplicacional, sendo que o MQTTS adiciona uma camada TLS entre o MQTT e o TCP com o intuito de cifrar os dados.

Ainda relativo ao modelo de arquitetura de três camadas, no que se refere às questões de segurança, são expostos os ataques e as vulnerabilidades existentes e a forma de como estas adversidades podem ser mitigadas.

3. *Intrusion Detection System*

Nos anos 80, no decorrer do Project 6169 - *Statistical Techniques Development For An Audit Trail System* surgiu o conceito de IDS, que consiste basicamente no uso de um algoritmo que analisa o perfil do comportamento de utilizadores em alta velocidade (Evangelista, 2008).

Na maioria dos casos, um IDS é composto por um ou mais sensores que recolhem informações, sendo que estes sensores podem ser colocados em pontos diferente na rede ou em terminais. As informações recolhidas pelo sensor, ou sensores, são enviadas para um ponto centralizado que tem a responsabilidade de analisar, gerar relatórios e emitir alarmes (Raza, et al. 2013, Zarpelão & Miani, 2017).

A Figura 10 representa a arquitetura genérica de um IDS. O IDS contém uma base de conhecimento, que se baseia numa base de dados com armazenamento de comportamentos de ataques ou vulnerabilidades já conhecidos e ou em especificações de tráfego de *standards* ou protocolos.

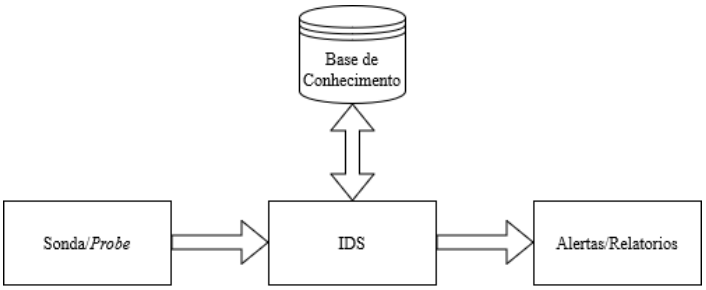

Figura 10 - Arquitetura genérica de um IDS

A Figura 11 demonstra como um IDS deve ser implementado de acordo com o seu tipo, a sua localização, a sua metodologia de deteção e a sua fonte de dados.

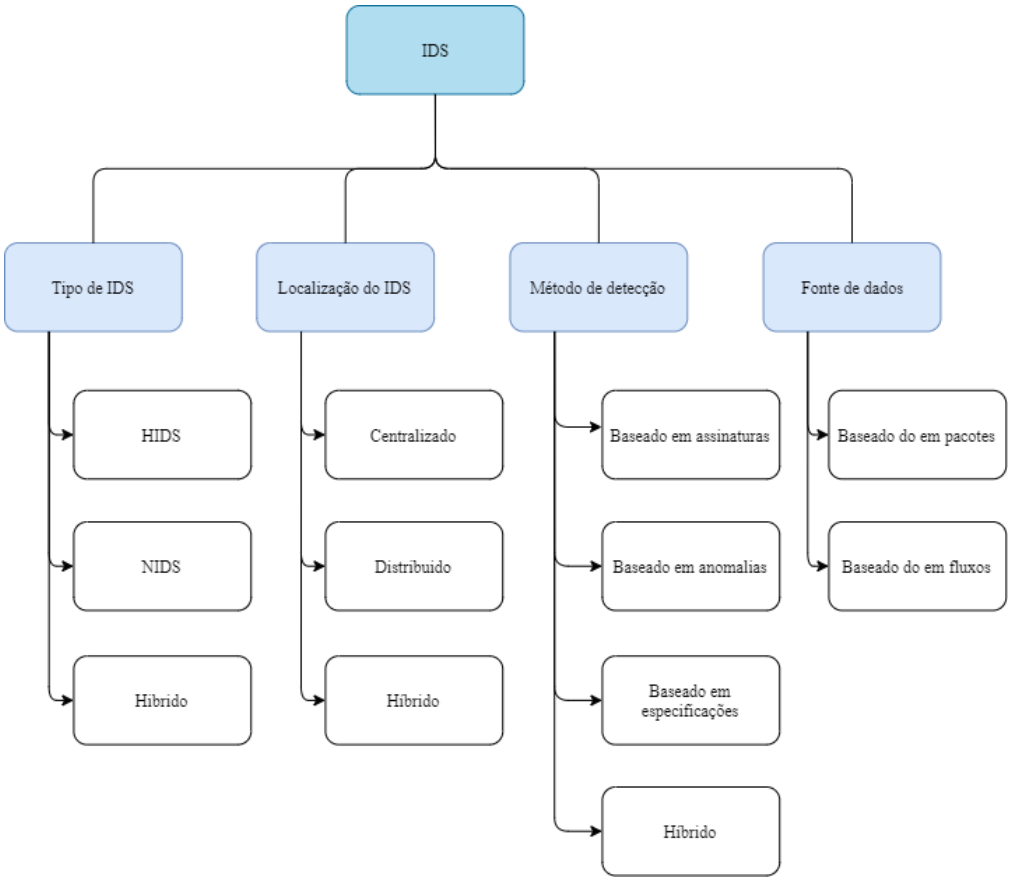

Figura 11 - Implantação de um IDS

3.1. Tipos de *Intrusion Detection System*

Existem vários tipos de IDS que podem ser implementados num sistema IoT. A sua escolha depende da finalidade, do tipo de ataques que se pretendem detetar, assim como, dos recursos disponíveis. Os tipos de IDS mais comuns são: o *Network* IDS (NIDS), o *Host* IDS (HIDS) e o Híbrido (Anwar, et al. 2017). De seguida explica-se cada um dos tipos.

3.1.1. NIDS

Instalado no sistema, este tipo de IDS tem a função de monitorizar o tráfego entre vários pontos, de forma discreta e sem interferir no mesmo.

A Figura 12 representa um exemplo de como um NIDS está integrado no sistema.

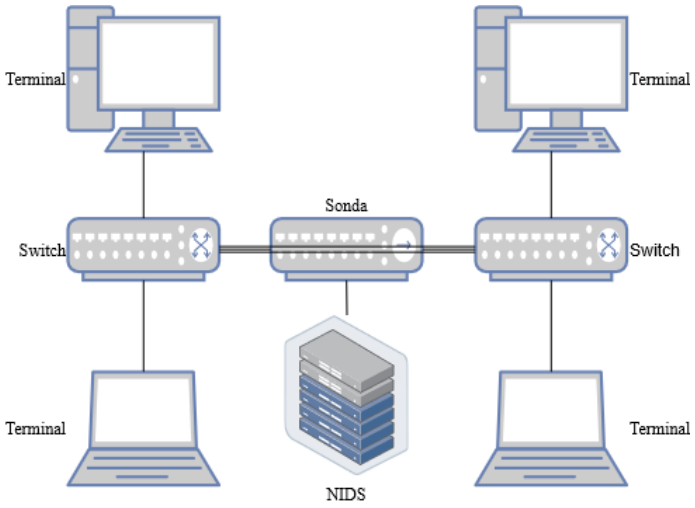

Figura 12 - NIDS diagrama

A monitorização do tráfego é efetuada em modo *mirroring* ou em modo *Test Access Port* (TAP). Em modo *mirroring*, o tráfego que atravessa um *switch* ou um encaminhador faz uma duplicação do tráfego para outra porta do *switch* ou para outra porta do encaminhador, a qual tem de estar ligada ao NIDS. Em modo TAP, também conhecido por modo *in-line*, é ligado um dispositivo específico na rede, sendo que esse dispositivo faz a duplicação do tráfego para o IDS (Jakub, et al. 2015).

A Tabela 8, adaptada do artigo Jakub, et al. (2015), explica as vantagens e as desvantagens de cada um dos tipos de monitorização.

Tabela 8 - Comparativos entre os vários métodos de monitorização

Tipo	Vantagens	Desvantagens
mirroring	Disponível nos *swtich* e encaminhador	Não confiável
TAP	Suporta redes de alta velocidade	Dispendioso, porque requer um elemento de rede adicional

3.1.2. HIDS

Instalado nos terminais, este tipo de IDS, também conhecido por agente, tem a capacidade de supervisionar o equipamento em que está instalado através da monitorização dos registos de ficheiros, dos recursos e do tráfego enviado e recebido.

A Figura 13 mostra um exemplo de como um HIDS está integrado no sistema, sendo que cada terminal pode ou não ter um agente instalado.

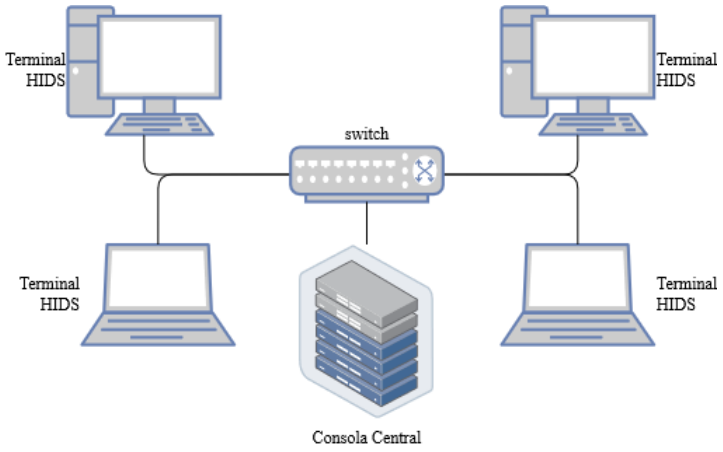

Figura 13 - HIDS diagrama

3.1.3. Híbrido

Corresponde à junção do NIDS e do HIDS, na qual os diferentes terminais têm um agente instalado (HIDS) e existem também um ou vários elementos que fazem a monitorização da rede (NIDS).

A Figura 14 exemplifica a implementação de um IDS híbrido.

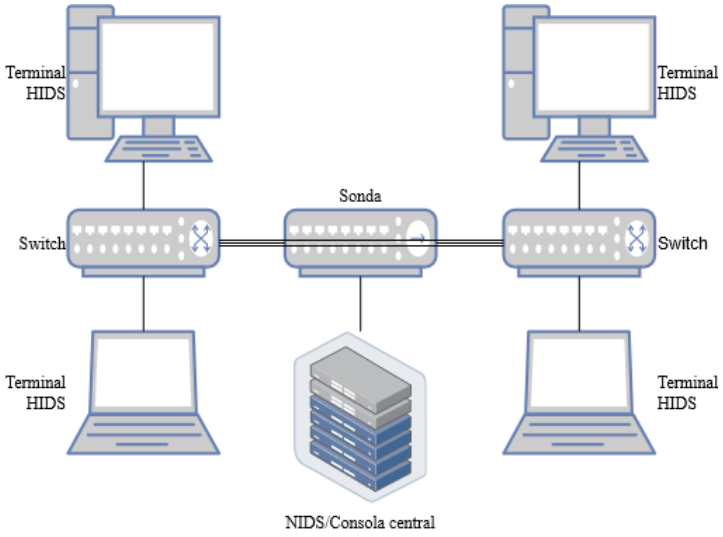

Figura 14 - IDS híbrido

A Tabela 9 apresenta as vantagens e as desvantagens dos três tipos de IDS (NIDS, HIDS e o Híbrido).

Tabela 9 - Comparativo entre NIDS, HIDS e Híbrido

Tipo	Vantagens	Desvantagens
NIDS	Baixo custo	Não tem visibilidade ao nível do dispositivo
	Capacidade detetar ataques que o HIDS não deteta	Não tem a capacidade de analisar tráfego encriptado
	Rápida resposta	Tem de estar sempre a monitorizar o tráfego
	Fácil instalação e tem um baixo impacto na rede em que está instalado	Sensível ao encaminhamento de tráfego
HIDS	Monitorizar todas camadas do modelo OSI	Instalação em cada dispositivo pode causar impacto na rede
	Não é necessário mais *hardware* adicional	
	Não tem problemas com encaminhamento de tráfego	A instalação no dispositivo pode causar impacto na integridade do *software*.
	Deteção simples de vírus, *malware* ou *trojans*	
Híbrido	Tráfego de várias fontes	O fluxo de dados poderá ser enorme
	Controlado e motorizado por um ponto central	
	Deteção e resposta também feita por um ponto central	O sistema pode ser perturbado por um atacante.
	Vantagens na monotorização, análise de incidentes e ataques em tempo real	

3.2.Estratégias de localização do *Intrusion Detection System*

A escolha da localização de um IDS deve ter em conta vários fatores, tais como: o tipo de rede que vai receber a informação, o tipo de IDS a ser implementado e o tipo de informação a ser recolhida. O artigo Yang, et al. (2018) define as três diferentes localizações possíveis para um IDS, a seguir mencionadas e explicadas.

3.2.1. Centralizado

Consiste num servidor centralizado com grandes recursos de processamento, de memória e de espaço em disco. Este tipo de servidor também tem a capacidade de fazer a monitorização da rede em tempo real, de forma a detetar ataques, ou seja, tem a capacidade de ser em simultâneo o sensor (aquele que faz a monitorização do tráfego) e o servidor, que tem a responsabilidade de analisar, de fazer relatórios e de emitir alarmes.

3.2.2. Distribuído

Um sistema distribuído é quando o sensor e o servidor (que tem a responsabilidade de analisar relatórios e enviar alarmes) estão em dispositivos separados, sendo que, desta forma, existem vários sensores instalados no sistema que enviam a informação recolhida para um ponto central, que analisa, cria relatórios e envia alarmes.

3.2.3. Híbrido

Consiste na junção entre o IDS centralizado e o IDS distribuído, sendo que esta estratégia de localização permite uma melhor gestão de recursos, tais como, de processamento e de memória. Geralmente, o dispositivo que tem a componente de sensor requer de mais recursos.

A Tabela 10 contém um comparativo das vantagens e das desvantagens entre as diferentes localizações de IDS.

Tabela 10 - Comparativo entre as diferentes localizações

Localização	Vantagens	Desvantagens
Centralizado	Fácil instalação	Necessita de grandes recursos
	Deteções externas	
Distribuído	Robusto	Custos mais elevados
	Deteções internas	Necessita de métodos de deteção mais leves
Híbrido	Robusto	Necessário mais processamento para os *clusters*

3.3. Metodologias de deteção de um *Intrusion Detection System*

A metodologia de deteção do IDS tem de detetar o tráfego, de forma a conseguir distingui-lo como normal ou anormal, sendo que este último pode significar que existe um ataque. A forma de medir a qualidade da metodologia utilizada é através da quantidade de falsos positivos ou de falsos negativos detetados, por outras palavras, quanto menor os falsos positivos ou negativos mais preciso é o método de deteção. Diz-se um falso positivo quando a deteção considera tráfego normal como anormal e um falso negativo quando é detetado tráfego anormal como normal.

Os métodos de deteção de um IDS podem ser efetuados através de quatro metodologias de funcionamento diferentes baseados em: anomalias, assinaturas, especificações e em híbrido (Inayat, et al. 2016, Liao, et al. 2013).

3.3.1. Método baseado em anomalias

O IDS monitoriza o tráfego de forma a analisar o comportamento deste, ou seja, através da observação do tráfego, este método cria um perfil do tráfego normal que flui no sistema, sendo que quando ocorre uma derivação do perfil é emitido um alerta.

3.3.2. Método baseado em assinaturas

Neste método é especificado previamente um conjunto de assinaturas ou de regras numa base de dados, que analisa o tráfego, compara com as assinaturas que tem definidas e quando existe uma correspondência é gerado um alerta.

3.3.3. Método baseado em especificações

Este método contém especificações de um comportamento normal de um protocolo, *standard* ou de uma aplicação. Essas especificações são previamente carregadas para uma base de conhecimento e quando o tráfego que está a ser observado pelo IDS não corresponde às especificações do tráfego normal é gerado um alerta.

3.3.4. Método híbrido

Este método é a combinação entre dois ou mais métodos de deteção, anteriormente referidos, sendo que utiliza as características dos métodos utilizados.

A Tabela 11 apresenta um comparativo das vantagens e das desvantagens entre os vários tipos de métodos de deteção.

Tabela 11 - Comparativos entre métodos de deteção

Método de deteção	Vantagens	Desvantagens
Baseado em anomalias	Padrões normais podem ser feitos através de *marchine learning*	Elevada taxa de falsos positivos
	Não requer assinaturas	Elevado consumo de recursos
	Tem a capacidade de fazer deteção de ataques não conhecidos	
Baseado em assinaturas	Baixa taxa de falsos positivos	Dificuldade na deteção de novos ataques
Baseado em especificações	Deteção de ataques conhecidos	Depende do fornecedor do protocolo ou aplicação
	Baixa taxa de falsos positivos	
Híbrido	Deteção de ataques conhecidos	Pode causar elevados consumos de recursos

3.4.Fonte de dados de um *Intrusion Detection System*

Um IDS tem ligado a ele uma fonte de dados, sendo que esta fonte pode ser baseada em pacotes de rede ou baseada em fluxos de rede. Esta fonte de dados contém o tráfego que o IDS analisa de forma a detetar intrusões (Alaidaros, et al. 2011). De seguida são explicados estas duas fontes.

3.4.1. Fonte de dados baseada em pacotes de rede

A fonte de dados baseada em pacotes, ou *Deep Packet Inspection* (DPI), analisa todo o pacote desde a camada dois (camada de dados) até à camada sete (camada de aplicação) do modelo *Open System Interconnection* (OSI). A fonte de dados baseada em pacotes faz a análise do cabeçalho (endereço IP de origem e destino e porto de origem e destino) e do *payload* que contém todos os dados que vêm no pacote.

A Figura 15 representa a arquitetura baseada em pacotes (Alaidaros, et al. 2011).

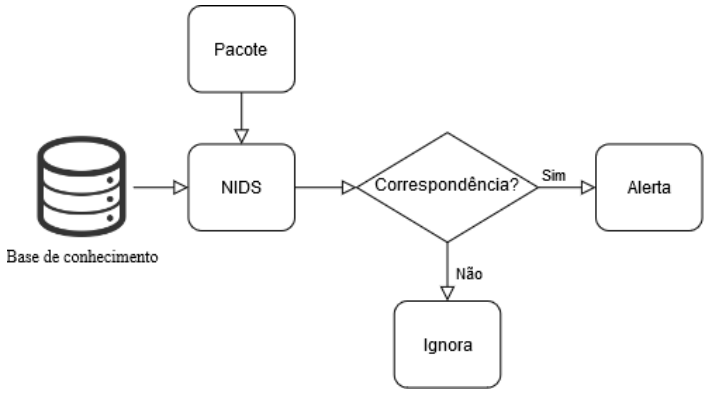

Figura 15 - Fonte de dados baseado em pacotes de rede

3.4.2. Fonte de dados baseada em fluxos de tráfego de rede

A fonte de dados baseada em fluxos de tráfego de rede apenas analisa o cabeçalho dos pacotes de rede (endereço IP de origem e destino e porto de origem e destino), ou seja, até à camada quatro (camada de transporte) do modelo OSI, sendo que, portanto, a parte do *payload* é ignorada.

A Figura 16 representa o funcionamento do fluxo de pacotes (Alaidaros, et al. 2011).

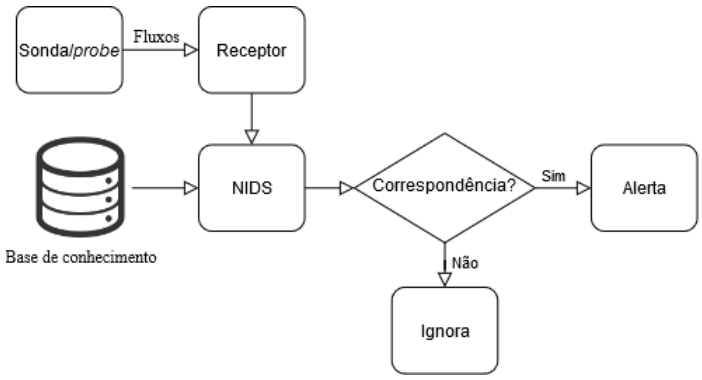

Figura 16 - Fonte de dados baseado em fluxos de tráfego de rede

A Tabela 12 apresenta um comparativo das vantagens e das desvantagens entre as fontes de dados baseadas em fluxos de tráfego de rede e em pacotes.

Tabela 12 - Comparativo entre fontes de dados

Tipo	Vantagens	Desvantagens
Baseado em fluxos de tráfego de rede	Os pacotes encriptados não influenciam a performance	Analisa apenas desde a camada quatro do modelo OSI
	Menos recursos necessários	Maior probabilidade de gerar falsos positivos
	Pouco sensível à privacidade dos dados	É necessária uma sonda
	Fluxos de tráfego de rede são enviados por uma sonda IP *Flow Information Export* (IPFIX), por exemplo, para uma filtragem previa	Sensível ao tempo entre pacotes enviado pela sonda para o NIDS
		O tráfego não pode estar sujeito a protocolos de encaminhamento
Baseado em pacotes	Analisa o pacote desde a camada dois até a camada sete do modelo OSI	Pacotes encriptados causam degradação na performance
	Menor probabilidade de geração de falsos positivos	Requer hardware para ser feito uma pré filtragem dos pacotes enviado para IDS
	Não requer *hardware* adicional para a coleta dos pacotes	Pode gerar uma grande quantidade de dados a ser processados
	Análise em tempo real	Como analisa todo o pacote está sujeito a questões de privacidade

3.5.Monitorização de fluxos de tráfego

Um registo de fluxo de rede, antes de ser exportado, só tem acesso aos cabeçalhos dos pacotes. No entanto, quando o fluxo é exportado muitas propriedades ficam disponíveis, tais como, o tempo de início do registo de fluxo de rede, a duração e se é *User Datagram Protocol* (UDP) ou TCP, entre outras. Todas estas propriedades são conhecidas por *Information Element* (IE) (Vieira, et al. 2019).

Uma forma de representar e transmitir um conjunto de IE é através do protocolo IPFIX, na qual a RFC 5101 (Claise & Bryant, 2008) contém a sua especificação. O IPFIX foi desenvolvido a partir do NetFlow versão 9 e é também designado de NetFlow versão 10 (ittsystems, 2020).

O IPFIX é suportado por vários fabricantes de equipamentos e foi concebido para analisar o tráfego de rede, de modo a fornecer um modelo de dados extenso e flexível, que possa ser ajustado à medida, confiável e possível de transferir os fluxos de tráfego de rede de forma segura (Claise & Bryant, 2008, Li, et al. 2013).

Cada IE tem um identificador único, sendo este gerido pela *Internet Assigned Number Authority* (IANA), na qual esta entidade tem a responsabilidade de garantir a compatibilidade entres os vários fabricantes.

A Tabela 13 contém alguns exemplos de IE e a sua respetiva descrição.

Tabela 13 - Exemplos de IE

Identificação do elemento	Descrição	Definição
4	protocolIdentifier	Identificação do protocolo
7	sourceTransportPort	Porto TCP ou UDP de origem do fluxo.
8	sourceIPv4Address	Endereço IPv4 origem do fluxo
11	destinationTransportPort	Porto TCP ou UDP de destino do fluxo.
12	destinationIPv4Address	Endereço IPv4 destino do fluxo
86	packetTotalCount	Total de pacotes trocados
136	flowEndReason	Razão do fim do fluxo
150	flowStartSeconds	Tempo de inico do fluxo

A arquitetura da monotorização dos fluxos de tráfego de rede apresenta quatro etapas, isto é, a observação dos pacotes, a medição e exportação de fluxos de tráfego de rede, a coleção de fluxos de tráfego e a análise de fluxos de tráfego de rede, que são ilustradas através da Figura 17 (Vieira, et al. 2019).

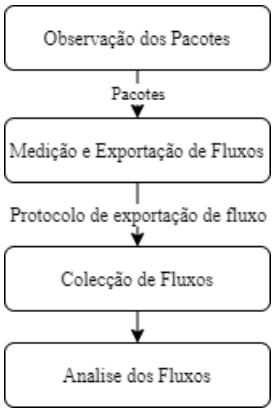

Figura 17 - Arquitetura da monotorização dos fluxos de tráfego de rede

3.5.1. Observação dos pacotes

Nesta etapa, os pacotes são capturados para um ponto de observação instalado na rede em modo *in-line* ou em modo espelho (*mirroring*).

Para a observação dos pacotes são utilizadas várias bibliotecas que estão disponíveis na maior parte dos SO, como por exemplo, o *libpcap* (Jacobson, et al. 1994) e o *libtrace* (Alcock, et al. 2012).

A observação dos pacotes está subdividida em cinco passos, isto é, captura do pacote, marcação do dia e da hora (*timestamping)*, seleção, amostragem de pacotes e filtragem de pacotes.

A amostragem de pacotes e a suas filtragens permitem reduzir a quantidade necessária, de memória utilizada e de processamento. Esta redução de volume dos pacotes facilita posteriormente o envio dos pacotes para a etapa da medição e da exportação do fluxo.

3.5.2. Medição e Exportação dos fluxos de tráfego de rede

Nesta etapa, os pacotes são transformados em fluxos de tráfego de rede. A medição e exportação dos fluxos de tráfego de rede é subdividida em três passos, isto é, processo de medição, amostragem de fluxo, sendo que esta passo é opcional, e processo de exportação.

Tal como referido anteriormente, um fluxo tem na sua definição algumas propriedades, tais como: os endereços IP, os portos, o protocolo UDP ou TCP e as medições, por exemplo, o número total de *bytes* ou de pacotes. Os fluxos de tráfego de rede são agregados pelo processo de medição através da lista de IE.

Após o processo de medição, é efetuada a amostragem de fluxos de tráfego de rede e as funções de filtragem são aplicadas, de modo a reduzir o volume de fluxos de tráfego de rede exportados.

O último passo é a exportação de fluxos de tráfego de rede. Neste passo é utilizado o protocolo IPFIX, através de vários aplicativos *open-source*, tais como: o *Yet Another Forum.net* (YAF) (Inacio & Trammell, 2010), o nProbe (Deri & SpA, 2003), o *Quality of Flow* (QoF) (ct-mplane.eu, 2020) e o *Vermont* (Lampert, et al. 2006).

3.5.3. Coleção de fluxos de tráfego de rede

Os fluxos de tráfego de rede no formato IPFIX exportados na etapa anterior, são armazenados nesta etapa. Os fluxos de tráfego de rede podem ser guardados de duas formas: numa memória volátil (por exemplo: em memória RAM) ou numa memória persistente (por exemplo: disco rígido). Se a forma de guardar os fluxos de tráfego de rede for em memória persistente, existem vários tipos de soluções, como: arquivos simples, base de dados de ficheiros e base de dados orientadas às colunas.

A escolha da forma de recolher os fluxos de tráfego de rede deve ter em conta a performance, as características do exportador de fluxos de tráfego de rede, o atraso no processamento, o formato de gravação dos fluxos de tráfego, a duplicação dos fluxos de tráfego de rede e a integração com outros sistemas.

3.5.4. Análise dos fluxos

Nesta etapa os fluxos de tráfego de rede armazenados são analisados, sendo que esta análise pode ser aplicada a três áreas: na análise e relatórios de fluxos de tráfego de rede, na deteção de vulnerabilidades e na monotorização da performance.

- Análise e relatórios dos fluxos de tráfego de rede - esta área é aplicada nas pesquisas, nas filtragens, nas visualizações de estatísticas e nos relatórios relacionados com os fluxos de tráfego de rede;
- Deteção de vulnerabilidades - esta área analisa os fluxos de tráfego de rede entre dois terminais, ou seja, nesta área são analisados os números de pacotes trocados entre terminais, o volume de *bytes*, o número de ligações, entre outros aspetos, com o objetivo de verificar se existe alguma vulnerabilidade;

- Monotorização de performance - esta área verifica se os serviços estão a funcionar adequadamente no sistema, obtendo-se o *Service Level Agreement* (SLA) de um serviço e a experiência do utilizador final. As métricas que são observadas são: o tempo de resposta, o atraso, o *jitter* ou a largura de banda utilizada.

3.6.Monitorização de fluxos de tráfego de rede para IoT

No artigo Santos, et al. (2019a) é proposto, para um sistema IoT, uma arquitetura para a monitorização de fluxos de tráfego de rede. Através das quatro etapas referidas no capítulo 3.5 (observação de pacotes, medição e exportação do fluxo, coleção de fluxos e análise dos fluxos de tráfegos) é efetuada uma adaptação das mesmas para um sistema IoT, que a seguir é explicada:

- Observação de pacotes no ambiente IoT - é o ponto de observação ou monitorização de um sistema IoT, onde os pacotes são enviados para a próxima etapa;
- Medição e exportação do fluxo de tráfego de rede - os pacotes capturados na etapa anterior são agregados em fluxos de tráfego, sendo que, posteriormente, esses fluxos de tráfego são enviados para o coletor de fluxos de tráfego de rede;
- Coleção de fluxos de tráfego de rede - recebe os fluxos de tráfego de rede gerados pela etapa anterior e armazena-os de forma persistente, de forma a ficarem disponíveis para a próxima etapa;
- Análise dos fluxos de tráfego de rede - nesta etapa ocorre a emissão de relatórios e de alertas.

3.7.Soluções existentes de IDS para IoT

Nos últimos anos foram efetuados vários estudos sobre o tema de qual poderá ser a melhor solução IDS para um ambiente IoT. Os artigos Santos, et al. (2019a) e Zarpelão et al. (2017) fazem uma compilação dos diferentes estudos existentes.

De seguida, de forma sucinta, encontram-se alguns dos artigos e trabalhos científicos estudados sobre este tema e as suas conclusões sobre a localização ou o método a utilizar na deteção de anomalias.

Cho, et al. (2009) sugere que um IDS deve estar centralizado no encaminhador de fronteira, isto é, entre a camada física e o domínio de rede. A análise dos ataques *botnet* usa o método

de deteção baseado em anomalias e este estudo conclui que, caso se tente aplicar um IDS a um ambiente IoT, é necessário considerar que poderão existir muitos dispositivos e que têm de ter suporte para várias tarefas.

Le, et al. (2011) usa a localização híbrida (NIDS e HIDS) do IDS, de forma a criar um monitor por região. Os monitores observam a rede, de forma a detetar dispositivos que estejam comprometidos, utilizando o método de deteção baseado em especificações.

Liu, et al. (2011) sugere utilizar o método de deteção baseado em assinaturas, sem fazer referência à localização do IDS.

Misra, et al. (2011) sugere utilizar o método de deteção baseado em anomalias, sem fazer referência à localização do IDS.

Gupta, et al. (2013) indica que a deteção deve ser efetuada através da análise do comportamento normal dos dispositivos de rede, sendo que, quando o comportamento passa a ter alguma anormalidade é gerado um alerta. Este artigo não faz referência à localização do IDS.

Kasinathan, et al. (2013a) e Kasinathan, et al. (2013b) são dois estudos efetuados pelos mesmos autores, que propõem um sistema centralizado de IDS, utilizando o método de deteção baseado em assinaturas.

Raza, et al. (2013) sugere que a localização do IDS seja híbrida, usando o encaminhador de fronteira e os dispositivos de rede. O encaminhador de fronteira tem um componente do IDS e analisa o tráfego. Os dispositivos de rede enviam informações para o encaminhador de fronteira. A análise é híbrida, fazendo um equilíbrio entre o consumo de recursos (processador e memória) do método baseado em anomalias e o consumo de espaço em disco do método baseado em assinatura. Este sistema tem o nome de SVELTE.

Wallgren, et al. (2013) propõe que a localização do IDS deve ser centralizada. A deteção é efetuada no encaminhador de fronteira, não fazendo referência de qual é o método de deteção que deverá ser empregue.

Amaral, et al. (2014) propõe que a localização do IDS seja híbrida, através de elementos selecionados, designados de *watchdogs*, que têm a capacidade de observar os pacotes trocados numa área. Os *watchdogs* utilizam o método de deteção baseado em especificações para a deteção de anomalias. Quando é detetado uma anomalia é enviado um alerta para um

sistema de *Event Management System* (EMS), que tem a caraterística de não ter constrangimento de recursos.

Jun & Chi, (2014) sugere que a localização do IDS deve ser centralizada. O IDS está instalado nos encaminhadores de fronteira, que vão fazer a monotorização da rede, sendo que utiliza método de deteção baseado em especificações. Este sistema é chamado de *Complex Event-Processing* (CPE).

Lee, et al. (2014) apresenta um estudo baseado no consumo de energia dos dispositivos a serem analisados. É proposto um IDS com localização distribuída e o método de análise utilizado é o baseado em anomalias.

Oh, et al. (2014) propõe que o IDS seja localizado de forma distribuída e que se utilize o método de deteção baseado em assinaturas.

Cervantes, et al. (2015) utilizada o IDS localizado de forma distribuída e o método de deteção empregue é o híbrido, através da combinação do método baseado em especificações e em anomalias.

Pongle & Chavan, (2015) faz a proposta de um IDS com localização híbrida, sendo que, quando os dispositivos detetam alterações nos dispositivos vizinhos, enviam essas anomalias para um ponto centralizado, que está localizado no encaminhador de fronteira. O método utilizado de deteção é baseado em anomalias.

Summerville, et al. (2015) utiliza o método de deteção baseado em anomalias e não faz referência à localização do IDS.

Le, et al. (2016) propõe um IDS localizado de forma híbrida, sendo que a rede é dividida em vários grupos e cada grupo analisa uma parte do tráfego. Quando existe uma ocorrência, esta é enviada para o encaminhador de fronteira e o método de deteção empregue é baseado em especificações.

Thanigaivelan, et al. (2016) sugere um IDS com localização híbrida e são atribuídas várias tarefas ao encaminhador de fronteira. Os dispositivos verificam o tráfego e enviam eventos para o encaminhador de fronteira, sendo que este elemento tem a capacidade de enviar um alerta se for uma anomalia. O método utilizado para a deteção é o baseado em anomalias.

Midi, et al. (2017) cria um sistema IDS, designado por *Knowledge-driven Adaptable Lightweight Intrusion Detection System* (Kalis), onde a localização do IDS é centralizada no encaminhador de fronteira e é utilizado o método híbrido para a deteção de eventos.

Shreenivas, et al. (2017) desenvolve uma extensão do SVELTE (Raza, et al. (2013)) que assenta num sistema com IDS localizado de forma híbrida e com o método de deteção também híbrido.

Santos, et al. (2019b) propõe uma arquitetura IDS para um sistema IoT. O IDS tem uma localização distribuída, dividida em várias etapas, e o método de deteção é o baseado em especificações.

Na Tabela 14 é apresentada a compilação dos diferentes trabalhos referidos anteriormente. Na primeira coluna está descrito o trabalho, na segunda coluna a localização do IDS e na terceira coluna o método de deteção.

Tabela 14 - Comparativo dos diferentes estudos

Estudo	Localização do IDS	Método de deteção
Cho, et al. (2009)	Centralizado	Baseado em anomalias
Le, et al. (2011)	Híbrida	Baseado em especificações
Liu, et al. (2011)	-	Baseado em assinaturas
Misra, et al. (2011)	-	Baseado em anomalias
Gupta, et al. (2013)	-	Baseado em anomalias
Kasinathan, et al. (2013) e Kasinathan, et al. (2013)	Centralizado	Baseado em assinaturas
Raza, et al. (2013)	Híbrido	Híbrido
Wallgren, et al. (2013)	Centralizado	-
Amaral, et al. (2014)	Híbrido	Baseado em especificações
Jun & Chi. (2014)	Centralizado	Baseado em especificações
Lee, et al. (2014)	Distribuído	Baseado em anomalias
Oh, et al. (2014)	Distribuído	Baseado em assinaturas
Cervantes, et al. (2015)	Distribuído	Híbrido
Pongle & Chavan. (2015)	Híbrido	Baseado em anomalias
Summerville, et al. (2015)	-	Baseado em anomalias
Le, et al. (2016)	Híbrido	Baseado em especificações
Thanigaivelan, et al. (2016)	Híbrido	Baseado em anomalias
Midi, et al. (2017)	Centralizado	Híbrido
Shreenivas, et al. (2017)	Híbrido	Híbrido
Santos, et al. (2019)	Distribuído	Baseado em especificações

Após leitura dos vários trabalhos, que propõem IDS num sistema IoT, verifica-se que não existe um consenso acerca da localização e dos métodos de deteção a utilizar. Este acontecimento pode dever-se ao facto dos sistemas IoT serem relativamente recentes, onde existem muitos projetos *Do-It-Yourself* (DIY) e também por ainda existir uma falta de standards definidos especificamente para este tipo de sistemas. No entanto, estes artigos não

devem ser ignorados e devem ser tidos em consideração no campo dos sistemas IoT (Zarpelão & Miani, 2017). De um modo geral, os artigos referem que a escolha de um IDS deverá ter em conta o baixo consumo de recursos (disco, memória e processamento), o baixo consumo de energia, assim como, a possibilidade de existir a geração de um elevado volume de informação.

3.8.Síntese

O presente capítulo inicia-se com a caraterização das soluções de IDS, onde são apresentados os diferentes tipos, as localizações possíveis, os métodos de deteção existentes e as fontes de dados possíveis. Quanto à monitorização dos fluxos de tráfego de rede, são explicadas as quatro etapas da arquitetura de monotorização de fluxos de tráfego de rede e de como as mesmas etapas podem ser adaptadas para um sistema IoT.

Relativamente aos trabalhos propostos pela comunidade acerca de IDS para sistemas IoT, verifica-se, entre os vários autores, que não existe um consenso relativo à melhor solução de IDS a ser aplicado. Estas divergências de soluções ocorrem devido às várias capacidades de recursos dos dispositivos IoT, à quantidade elevada e variada de dispositivos existentes e à não adoção de *standards*.

Importa ainda destacar que, tendo em conta o que foi apresentado e discutido no presente capítulo, é possível propor um IDS adaptado para um sistema IoT, o qual será descrito no capítulo seguinte.

4. Solução proposta de IDS para IoT

Nos capítulos anteriores foi exposto o estado da arte sobre os sistemas IoT, bem como, os seus problemas de segurança e as respetivas formas de mitigar os mesmos para cada uma das três camadas presentes na arquitetura IoT. Também se descreveram os vários sistemas de IDS, as suas localizações possíveis e os métodos de deteção existentes.

Com base no que foi descrito nos capítulos anteriores, no presente capítulo será apresentada uma proposta de um IDS para um sistema IoT. A referida proposta será descrita nos seguintes subcapítulos, que descrevem o respetivo modelo de arquitetura, o seu funcionamento e a descrição de cada um dos elementos que compõem a arquitetura.

4.1. Requisitos e características

Os IDS utilizados nos sistemas IoT têm de possuir requisitos que contribuem para uma eficiente deteção de intrusões nestes sistemas, de modo a abranger a diversidade existente nos ambientes IoT.

Os requisitos necessários são descritos de seguida, bem como as suas características associadas:

- Deteção em todas as camadas - o IDS deve ter a capacidade de detetar ataques e intrusões em todas as camadas da arquitetura dos sistemas IoT, perceção, rede e aplicação;
- Deteção de intrusões internas e externas - capacidade de detetar intrusões originadas a partir de dispositivos internos do sistema IoT, assim como em *hosts* externos ao sistema IoT;
- Deteção em tempo útil - capacidade de detetar as intrusões que possam estar a decorrer e dentro de um intervalo de tempo razoável para os diversos tipos de intrusões;
- Escalabilidade - capacidade de acomodar um aumento da quantidade de atividades a analisar e do número de dispositivos a monitorizar, resultante da expansão do sistema IoT, em termos do número de dispositivos IoT e das correspondentes comunicações;
- Interoperabilidade e extensibilidade - deve ser garantido o suporte a diferentes tecnologias de comunicação, protocolos e standards, por forma a garantir uma melhor

integração com os restantes sistemas e dispositivos IoT. Para além disso, o IDS deve ser extensível facultando a inclusão ou integração de novos mecanismos de deteção de intrusões, novos padrões de intrusões, assim como novas tecnologias e protocolos à medida que os mesmos venham a surgir;

- Reconfigurável - deve ser garantida a inclusão e a modificação das políticas de intrusão durante a vida útil do sistema IoT;
- Sem alterações de *software* IoT - o sistema IDS deve minimizar a modificação ou alteração de software aplicacional utilizado nos dispositivos e nos serviços IoT disponibilizados;
- Reduzida implicação no desempenho - o sistema IDS deve minimizar o uso dos recursos computacionais dos dispositivos e dos serviços IoT, mantendo uma baixa pegada computacional, nomeadamente em termos de armazenamento, processamento, memória volátil e ocupação de largura de banda;
- Proteção das comunicações internas do sistema IDS - deve ser assegurada a proteção e segurança das comunicações internas entre os diversos componentes do sistema IDS.

O modelo de arquitetura apresentado de seguida tem em conta os requisitos descritos anteriormente, de forma a apresentar uma arquitetura IDS adaptada para um sistema IoT.

4.2.Modelo Arquitetural

Considera-se um IDS eficiente para um sistema IoT, o sistema que consome o mínimo de recursos dos dispositivos IoT e que também utiliza, caso seja necessário, os recursos disponíveis de outros dispositivos associados (encaminhador de periferia ou sistemas na nuvem).

Deste modo, e em concordância com os requisitos apresentados no capítulo 4.1, é proposto um IDS do tipo NIDS baseado na análise dos fluxos de tráfego de rede IP, com localização distribuída, e que assenta sobre uma arquitetura no modelo de três camadas, conforme ilustrado na Figura 18.

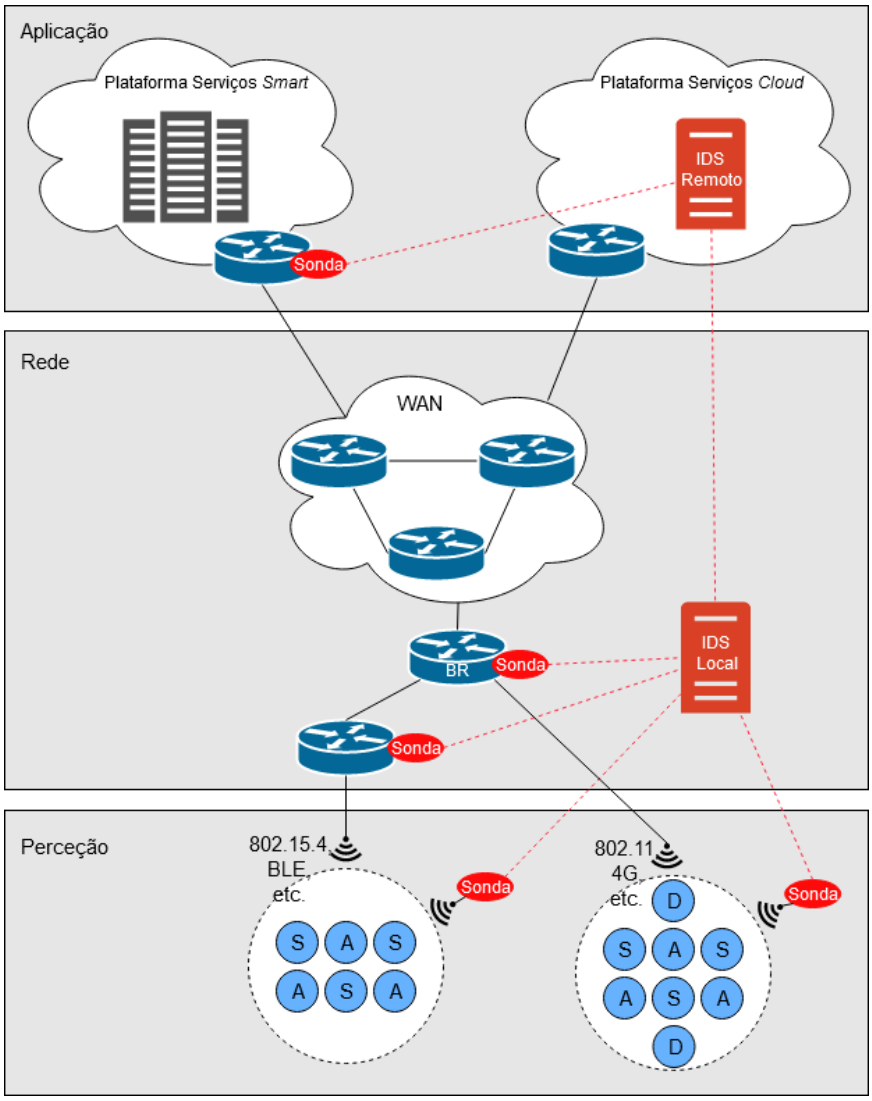

Figura 18 - Proposta de arquitetura para um IDS para IoT de análise fluxos de tráfego de rede

Descrição da arquitetura

A arquitetura proposta é do tipo NIDS, com localização distribuída, baseada na análise dos fluxos de tráfego de rede IP e recorre a sondas para capturar tráfego IP nas camadas de aplicação, rede e perceção. Esta arquitetura assegura o acesso ao tráfego entre os dispositivos IoT, em cada camada separadamente ou entre várias camadas em simultâneo.

O tráfego capturado pela sonda é encaminhado para um elemento do IDS, designado de IDS Local ou de IDS Remoto. Este elemento tem a responsabilidade de armazenar e de analisar o tráfego, está localizado nas camadas de rede e de aplicação e dispõe de menos limitações em relação a recursos computacionais, por exemplo: processamento, memória ou consumo energético.

O acesso ao tráfego interno ou externo em tempo útil permite detetar intrusões de forma rápida e eficiente, sendo que, no caso de intrusões internas permite detetar, por exemplo, ataques de *sleep deprivation*, e no caso de intrusões externas permite, por exemplo, detetar ataques DoS e *Distributed Denial of Service* (DDoS).

O IDS proposto apresenta uma estratégia de localização distribuída. A monitorização e a respetiva captura do tráfego é distribuída pelas diferentes camadas da arquitetura, já a análise do tráfego capturado é efetuada de forma centralizada em cada dispositivo dedicado para esse efeito. Esta estratégia permite pouca interferência no desempenho dos serviços e nos sistemas IoT.

O método de deteção de intrusões proposto é o método baseado em especificações, na qual apresenta vantagens como: não são necessários muitos recursos em termos de processamento, têm baixas taxas de falsos positivos e não é necessário um período de aprendizagem das especificações, já que as especificações são previamente carregadas, permitindo começar a funcionar de imediato.

A especificação do comportamento normal do tráfego dos dispositivos e dos serviços IoT é atualizada de maneira a acomodar mudanças nos dispositivos e nos serviços IoT, sendo que, quando existe uma derivação ao tráfego normal, quer seja uma intrusão ou um ataque, é emitido um alerta.

4.3.Funcionamento da arquitetura

O IDS proposto opera de acordo com a análise de fluxos de tráfego de rede IP e é composto por dois elementos: a sonda e o módulo do IDS centralizado.

Relativamente à sonda, nesta ocorre a observação de pacotes, a agregação de fluxos de tráfego de rede e a exportação dos fluxos de tráfego de rede, sendo que a sonda, posteriormente, envia os registos de fluxos de tráfego de rede para o módulo central do IDS. Após a receção dos registos dos fluxos de tráfego de rede por parte do módulo central do IDS, este analisa-os, com base em especificações, e gera e armazena os alertas de intrusões numa base de dados.

A Figura 19, adaptada de Santos, (2020), ilustra o esquema descrito acima para os dois elementos: a sonda e o IDS.

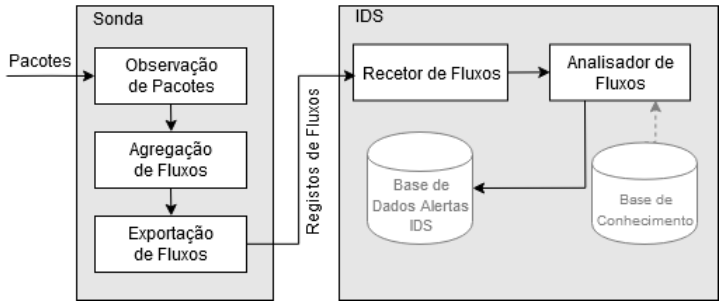

Figura 19 - Arquitetura proposta para um IDS para IoT baseado em análise de fluxos de tráfego de rede

4.3.1. Sonda

Este elemento tem a responsabilidade de monitorizar e de capturar o tráfego IP, que flui na rede ou no segmento de rede. Na arquitetura proposta existe a possibilidade de várias sondas funcionarem em simultâneo, cada uma com a sua função específica, sendo que podem estar integradas num outro dispositivo ou dispositivos, como por exemplo, encaminhadores de fronteira.

Para que esta situação seja possível, o SO ou o *firmware* desses dispositivos tem de suportar as funções da sonda (ou das sondas), na qual esta pode estar localizada nas camadas IoT a seguir mencionadas.

- Camada de perceção - a sonda localizada nesta camada deve suportar as tecnologias e os protocolos IoT, por exemplo: 802.15.4, BLE, o 802.11, etc.;
- Camada de rede - a sonda localizada nesta camada faz a captura do tráfego de entrada e de saída das interfaces dos encaminhadores, sendo que é necessário que os encaminhadores tenham suporte para esta função, sem a necessidade de modificar o *firmware* instalado no encaminhador;
- Camada de aplicação - a sonda faz um pré-processamento do tráfego, de modo a gerar os respetivos registos de fluxos tráfego de rede e enviar os mesmos, quase em tempo real, para o módulo central IDS, permitindo assim que seja efetuada a geração de alertas em tempo útil.

4.3.2. Módulo central IDS

O módulo central IDS, localizado na camada de rede e de aplicação IoT, é responsável pela recolha dos registos de fluxos de tráfego de rede (anteriormente enviados por uma ou mais

sondas) e pelo armazenamento dos mesmos registos em memória persistente, para futura análise.

Este elemento contém uma componente com a função de analisar os registos de fluxos de tráfego de rede recebidos e armazenados, de forma a que esta análise detete anomalias no comportamento do tráfego IP. A deteção de anomalias ocorre com recurso às especificações que estão guardadas na base de dados de especificações do IDS, sendo que, quando é detetado uma anomalia durante a análise, é gerada uma mensagem de alerta que é armazenada numa base de dados de alertas do IDS.

Todas as comunicações efetuadas entre as sondas e o módulo central IDS são efetuadas através de canais seguros, com cifragem e, se possível, através de rede cablada. A comunicação entre sondas (camada de perceção e de rede) e o módulo central IDS (camada de rede) deve efetua-se através de uma *Virtual Local Area Network* (VLAN) dedicada para esta troca de comunicações. Com isto, cria-se mais uma camada de segurança na comunicação interna entre os diferentes elementos do IDS.

4.4.Descrição dos elementos

De acordo com o capítulo 3.5, que refere as quatro etapas de monitorização de fluxos de tráfego de rede, e o capítulo 3.6, que sugere uma proposta para a monitorização de fluxos de tráfego de rede em ambiente IoT, é efetuada a proposta de uma adaptação de um IDS para um sistema IoT.

A Figura 20, adaptada de Santos et al. (2019a) representa uma solução para a monitorização dos fluxos de tráfego de rede num ambiente IoT.

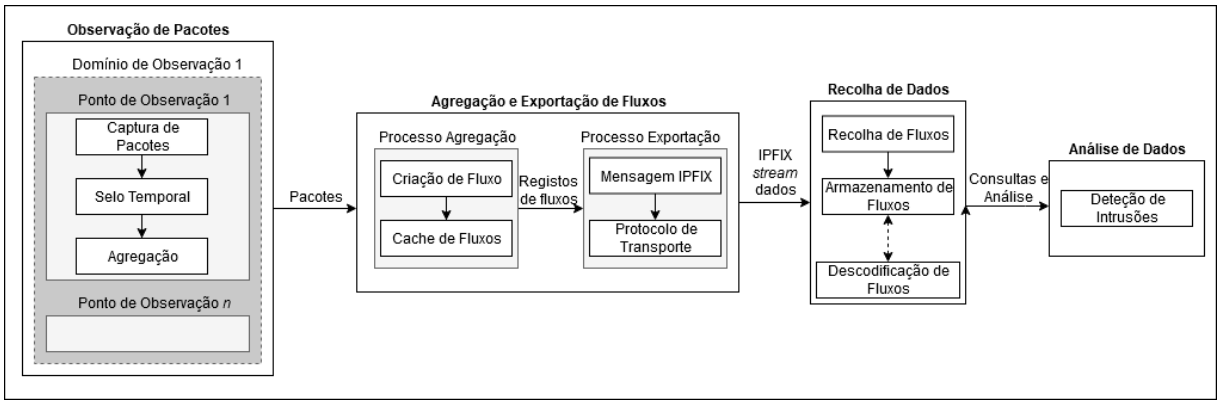

Figura 20 - Monitorização de fluxos tráfego de rede para IoT

4.4.1. Observação de Pacotes no ambiente IoT

O sistema IDS proposto apresenta um ou mais domínios de observação, sendo que cada domínio contém parte de um sistema IoT ou a monitorização do meio de comunicação, por exemplo: o BLE ou o *Wi-Fi*. Cada domínio de observação contém um ou mais pontos de observação.

Para aumentar a performance no processo de captura, Santos, et al. (2019a) propõe que seja utilizada a biblioteca *PF_RING* (ntop, 2020) e que deverá também ser utilizado o *Network Time Protocol* (NTP), de forma a garantir o sincronismo do tempo em todos os pontos de observação.

4.4.2. Medição e Exportação do Fluxo

Nesta etapa, os pacotes capturados são agregados em fluxos de tráfego de rede e, por consequência, são obtidos os IE.

Os IE mais importantes são os de transporte, nomeadamente: o endereço *Media Access Control* (MAC) de origem e de destino, o endereço IP de origem e de destino, o porto de origem e de destino, assim como, o protocolo de transporte (TCP ou UDP).

Contudo, Santos, et al. (2019a) sugere que devem ser selecionados os fluxos de tráfego de rede bidirecionais de modo a que, com isso, seja possível a obtenção de IE relativos à direção do fluxo, ao início e ao fim do fluxo, ao total de *bytes* trocados, total de pacotes trocados, razão do fim do fluxo e estatísticas TCP (tais como *flags* TCP e números de sequência). Todas estas características permitem aumentar a quantidade e qualidade das informações que devem ser extraídas de cada fluxo e permitem uma melhor análise dos fluxos de tráfego.

Com a seleção dos IE é definido o *layout* de *cache* de fluxos de tráfego de rede (deverá ser permanente) que o exportador de fluxos de tráfego de rede deverá utilizar. Esses fluxos de tráfego de rede vão integrar mensagens IPFIX que são exportadas para a próxima etapa, isto é, a recolha de fluxos de tráfego de rede. A exportação deve acontecer através da utilização do protocolo IPFIX sobre TLS, de forma a garantir confidencialidade dos fluxos de tráfego de rede.

4.4.3. Recolha de fluxos

A recolha de fluxos de tráfego de rede é escalável e pode ser localizada num serviço na *clould*, devido às limitações dos dispositivos IoT e dos encaminhadores de fronteira.

A recolha de fluxos de tráfego de rede deve considerar que os fluxos são recebidos por via IPFIX sobre TLS e que são guardados de forma persistente. Santos, et al. (2019a) sugere utilizar *flat files*, devido ao facto de serem otimizados na utilização de espaço em disco e devido à sua performance na inserção da informação, apesar de apresentarem limitações ao nível da flexibilidade das pesquisas e performance das mesmas.

4.4.4. Análise dos fluxos

Santos, et al. (2019a) sugere que, para detetar a vulnerabilidades e as falhas de segurança nos sistemas IoT, se deve ter em conta as pesquisas efetuadas em bases de dados do tipo *flat files*, bem como, as pesquisas na informação contida nos fluxos de tráfego de rede.

Na Figura 21, adaptada de Santos (2019b) encontra-se representado o funcionamento da análise dos fluxos de tráfego de rede, isto é, de como é analisado um registo de fluxo de tráfego de rede e o que é efetuado no caso de se detetar uma anomalia num registo de fluxo tráfego de rede.

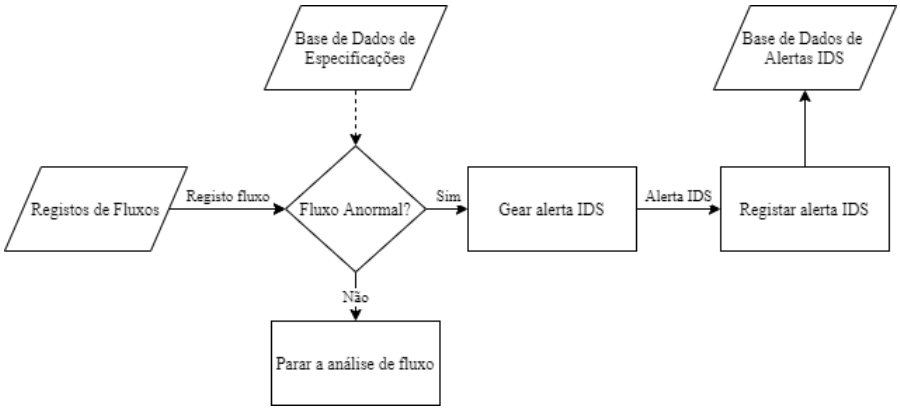

Figura 21 - Esquema do processo de análise dos fluxos de tráfego

4.4.5. Seleção dos IE

A necessidade de existir uma análise de registos de fluxos de tráfego de rede com maior exatidão, levou a que fosse selecionado um conjunto alargado de IE com determinadas características. Assim, sendo, foram selecionados os IE relativos ao início e ao fim do fluxo, a endereços IP, aos portos, aos endereços MAC, a protocolo de transporte (UDP ou TCP), bem como, aos IE relativos ao volume de pacotes e *bytes* trocados e à razão do término dos fluxos. Também foram selecionados os IE que fornecem estatísticas relativas ao fluxo de tráfego, como por exemplo: o volume de pacotes vazios, o desvio padrão e o tamanho máximo de pacotes nos fluxos.

A Tabela 15 expõe os IE selecionados.

Tabela 15 - IE selecionados

ID	Descrição	Definição
152	flowStartMilliseconds	Tempo do início do fluxo em milissegundos desde 1970-01-01 00:00:00 *Universal Time Coordinate* (UTC).
153	flowEndMilliseconds	Tempo do fim do fluxo em milissegundos desde 1970-01-01 00:00:00 UTC.
21	reverseFlowDeltaMilliseconds	Diferença de tempo em milissegundos entre o primeiro pacote de saída e do primeiro pacote de entrada de resposta ao anterior.
4	protocolIdentifier	Protocolo de transporte IP do fluxo.
8	sourceIPv4Address	Endereço IPv4 origem do fluxo
7	sourceTransportPort	Porto TCP ou UDP de origem do fluxo.
2	packetDeltaCount	Número de pacotes no sentido origem do fluxo.
1	octetDeltaCount	Número de octetos dos pacotes no sentido origem do fluxo.
40	flowAttributes	Atributos do fluxo no sentido de origem do fluxo.
56	sourceMacAddress	Endereço MAC origem do primeiro pacote no sentido origem do fluxo.
12	destinationIPv4Address	Endereço IPv4 destino do fluxo
11	destinationTransportPort	Porto TCP ou UDP de destino do fluxo.
2	reversePacketDeltaCount	Número de pacotes no sentido inverso do fluxo.
1	reverseOctetDeltaCount	Número de octetos dos pacotes no sentido inverso do fluxo.

16424	reverseFlowAttributes	Atributos do fluxo no sentido inverso do fluxo.
80	destinationMacAddress	Endereço MAC origem do primeiro pacote no sentido inverso do fluxo.
14	initialTCPFlags	*Flags* TCP do pacote inicial no sentido origem do fluxo.
5	unionTCPFlags	União das *flags* TCP de todos os outros pacotes, com exceção da inicial, no sentido origem do fluxo.
16398	reverseInitialTCPFlags	*Flags* TCP do pacote inicial no sentido inverso do fluxo.
16399	reverseUnionTCPFlags	União das *flags* TCP de todos os outros pacotes, com exceção do inicial, no sentido inverso do fluxo.
184	tcpSequenceNumber	Número de sequência inicial no sentido origem do fluxo.
185	reverseTcpSequenceNumber	Número de sequência inicial no sentido inverso do fluxo.
10	ingressInterface	O índice da interface IP onde os pacotes de rede do fluxo estão a ser recebidos.
14	egressInterface	O índice da interface IP onde os pacotes de rede do fluxo estão a ser recebidos.
58	vlanId	*Tag* VLAN 802.1q do primeiro pacote no sentido origem do fluxo.
33	silkAppLabel	Etiqueta da aplicação, definida como a primeira associação de uma aplicação e um porto.
5	ipClassOfService	Para pacotes IPv4, é o valor do campo *Type Of Service* (ToS) do cabeçalho IPv4.
136	flowEndReason	Código de fim do fluxo, conforme definido no IPFIX.
223	tcpUrgTotalCount	Número de pacotes TCP que têm a *flag* urgente ativa.
500	smallPacketCount	Número de pacotes que contêm menos de 60 *bytes* de *payload*.
501	nonEmptyPacketCount	Número de pacotes que contêm pelo menos de um *byte* de *payload*.
502	dataByteCount	Total de bytes transferidos como *payload*.
503	averageInterarrivalTime	Média do número de milissegundos entre pacotes.
505	firstNonEmptyPacketSize	Tamanho do *payload* do primeiro pacote não vazio.
510	largePacketCount	Número de pacotes que contêm pelo menos de 220 *bytes* de *payload*.
506	maxPacketSize	O maior tamanho de *payload* transferido no fluxo.
507	firstEightNonEmptyPacketDirections	Representa a direccionalidade dos primeiros oito pacotes não vazios. Zero para sentido origem, um para sentido inverso.
508	standardDeviationPayloadLength	O desvio padrão do tamanho do *payload* para os dez primeiros pacotes não vazios.
504	standardDeviationInterarrivalTime	O desvio padrão do *Round-Trip Time* (RTT) para os dez primeiros pacotes.
223	reverseTcpUrgTotalCount	Número de pacotes TCP que têm a *flag* urgente ativa no sentido inverso.
16884	reverseSmallPacketCount	Número de pacotes que contêm menos de 60 *bytes* de *payload* no sentido inverso.
16885	reverseNonEmptyPacketCount	Número de pacotes que contêm pelo menos de um *byte* de *payload* no sentido inverso.
16886	reverseDataByteCount	Total de *bytes* transferidos como *payload* no sentido inverso.
16887	reverseAverageInterarrivalTime	Média do número de milissegundos entre pacotes no sentido inverso.
16889	reverseFirstNonEmptyPacketSize	Tamanho do *payload* do primeiro pacote não vazio no sentido inverso.
16894	reverseLargePacketCount	Número de pacotes que contêm pelo menos de 220 *bytes* de *payload* no sentido inverso.
16890	reverseMaxPacketSize	O maior tamanho de *payload* transferido no fluxo no sentido inverso.

Recorre-se a uma seleção alargada de IE devido à necessidade de obter mais informações sobre os pacotes que foram capturados pela sonda, de maneira a efetuar-se uma análise mais precisa. Esta seleção de IE permite fazer a definição do *layout* da *cache* de fluxos de tráfego de rede que são capturadas na sonda. Esta *cache* de fluxos de tráfego de rede permite que os

fluxos de tráfego de rede não tenham tempo limite de validade até serem exportados posteriormente pela sonda.

Estes fluxos de tráfego de rede armazenados e que são exportados, pela sonda, são designados por registos de fluxos de tráfego de rede. Os registos de fluxos de tráfego de rede estão integrados nas mensagens IPFIX e são exportados posteriormente pelo exportador de fluxos de tráfego de rede para o elemento de recolha de fluxos de tráfego de rede através do protocolo de transporte TCP.

Esta proposta de arquitetura utiliza o protocolo de transporte TCP devido à sua capacidade de gestão de congestionamento, à sua disponibilidade, à sua confiabilidade e à sua garantia de segurança durante o transporte da informação. Este protocolo permite também recorrer ao IPFIX sobre TLS, de modo a permitir a adição de confidencialidade e de privacidade das mensagens IPFIX.

4.5.Síntese

No presente capítulo apresentou-se uma proposta de um IDS para um sistema IoT. A proposta baseia-se na análise do que é exposto nos capítulos anteriores, nos quais se abordam as características, os problemas e as limitações que um sistema IoT, assim como os problemas de segurança inerentes para um sistema IoT.

Sendo assim, é proposto um IDS do tipo NIDS baseado na análise dos fluxos de tráfego de rede IP, com localização distribuída, método de deteção baseado em especificações e que assenta sobre uma arquitetura no modelo de três camadas.

A localização distribuída do IDS permite que este seja constituído por mais que um elemento. No caso do IDS proposto, este é constituído por dois elementos: a sonda e o módulo central IDS. A sonda tem a função de recolher e de exportar os registos de fluxos de tráfego de rede IP, e o módulo central IDS tem a função de receber, de armazenar e de analisar esses mesmos registos. No módulo central IDS, quando num registo de fluxo de tráfego de rede é detetado uma anomalia, através da correspondência com as especificações guardadas na base de dados de especificações, gera-se um alerta que é armazenado na base de dados de alertas. O meio de comunicação entre a sonda e o módulo central IDS deve ser efetuado por canais seguros.

Neste capítulo também foram descritas cada uma das quatro etapas da arquitetura proposta para este IDS, isto é, a observação de pacotes no ambiente IoT, a medição e a exportação dos fluxos de tráfego de rede , a recolha de fluxos de tráfego de rede e a análise dos fluxos de tráfego de rede.

O presente capítulo termina com a descrição de IE relevantes, de acordo com determinadas características, a serem exportados pela sonda através do protocolo IPFIX.

5. Demonstração e validação da arquitetura

Este capítulo descreve a fase de demonstração e de validação do protótipo desenvolvido, de acordo com a arquitetura apresentada no capítulo 4. O protótipo funcionará sobre um ambiente real num sistema IoT, onde existirão sensores, atuadores, serviços IoT e serviços de rede. Estas características de ambiente permitem a criação de vários tipos de tráfegos IoT, de forma a que a arquitetura proposta consiga ser validada na recolha, no armazenamento e na análise do tráfego.

Este capítulo também descreve a forma de como se obtém as especificações de tráfego MQTT e MQTTS, isto é, a descrição de cada tipo de tráfego *Publisher* e *Subscriber*. Estas especificações serão utilizadas no IDS e para as mesmas apenas serão selecionados os IE relevantes para os protocolos MQTT e MQTTS, sendo que para estes IE serão definidos os valores considerados normais para o tráfego MQTT e MQTTS.

De seguida, será definido um plano de testes com a finalidade de validar a arquitetura proposta nos constituintes do funcionamento do IDS e o desempenho na deteção dos protocolos IoT, MQTT e MQTTS.

5.1. Protótipo desenvolvido

O protótipo pretende recriar o mais próximo possível a realidade de um sistema IoT e é composto por dispositivos que utilizam os protocolos MQTT e MQTTS. Estes dispositivos trocam mensagens entre sensores, atuadores e serviços IoT, com o principal objetivo de simular um cenário na qual se recolhe a leitura da temperatura do ar ou da luminosidade, e que envia comandos para um sistema de controlo de Aquecimento, Ventilação e Ar Condicionado (AVAC). Este sistema de controlo funciona com a informação recolhida da temperatura do ar, ou do estado do sistema de iluminação, na qual modifica a luz presente numa sala conforme a luminosidade existente.

Os dispositivos MQTT e MQTTS com maior destaque são os do tipo anunciante (*Publisher)*, intermediário (*Broker)* e subscritor (*Subscriber)*. Para além destes dispositivos MQTT e MQTTS, existem mais três dispositivos igualmente importantes, são eles : o *Raspberry PI*, que tem a função de encaminhador de periferia e que tem instalada a sonda IDS; o *Ubuntu Server 1*, que tem instalado o módulo central IDS; e, por último, um dispositivo atacante,

que tem a função de gerar tráfego com ataques e intrusões e que é utilizado para a validação da arquitetura proposta.

A Figura 22 representa o protótipo que foi utilizado para a arquitetura proposta (Santos, 2020).

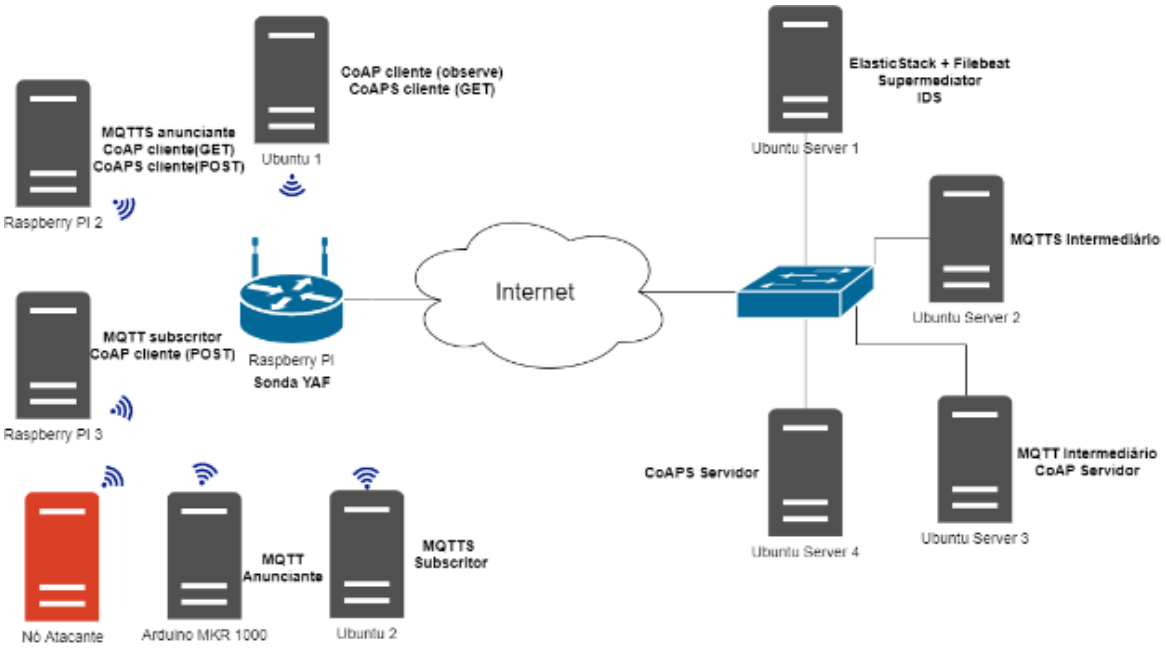

Figura 22 - Protótipo para testar a arquitetura proposta

O protótipo tem o objetivo de simular um sistema IoT e recorre a duas redes: a rede interna e a rede externa. A rede interna é uma rede sem fios *Wi-Fi* (IEEE 802.11n) que contém sensores e atuadores. A rede externa é a uma rede cablada *Fast Ethernet* (IEEE 802.3u) que tem o propósito de garantir uma ligação à Internet e aos serviços baseados na nuvem.

Apesar do protótipo apresentado também conter dispositivos CoAP e CoAPS, estes não foram utilizados no presente trabalho de projeto, pois foram abordados e utilizados no trabalho de projeto de outro estudante, tal como foi referido na nota prévia do presente documento. Assim, os resultados produzidos acerca dos protocolos CoAP e CoAPS estão referidos no trabalho de Canuto, et al. 2019. Relativamente ao presente trabalho, este apenas aborda protocolos MQTT e MQTTS, logo, apenas os dispositivos do protótipo referentes a esses protocolos serão referenciados e abordados.

.

Posto isto, apresentam-se os elementos MQTT e MQTTS existentes na rede interna: o *Raspberry PI 2*, com a função de anunciante MQTTS (*Publisher*); o *Raspberry PI 3*, com a função de subscritor MQTT (*Subscriber*); o *Arduino MKR 1000*, com a função de anunciante MQTT (*Publisher*); e o *Ubuntu 2*, com a função de subscritor MQTT (*Subscriber*). Contudo encontram-se mais dois elementos utilizados: o *Raspberry Pi*, com a função de encaminhador e de sonda YAF (IDS); e o Nó atacante, com a função de gerar ataques aos dispositivos MQTT e MQTTS.

Em relação aos elementos MQTT e MQTTS existentes na rede externa, os mesmos são os seguintes: o *Ubuntu Server 2*, com a função de intermediário MQTTS (*Broker*), e o *Ubuntu Server 3*, com a função de intermediário MQTT (*Broker*). Para além destes, na rede externa ainda se encontra presente o *Ubuntu Server 1*, com a função de módulo central IDS.

O *Raspberry Pi* é um dispositivo com a função de encaminhador de periferia e tem a responsabilidade de fazer a "ponte" entre a rede interna e a rede externa, através de duas interfaces: a interface *Wi-Fi* e a interface *Fast Ethernet*.

A interface *Wi-Fi* faz a ligação entre os sensores e os atuadores (rede interna), através de rede sem fios com endereçamento IPv4 privado, permitindo assim fornecer o acesso à Internet e aos serviços IoT remotos. A interface *Fast Ethernet*, com a função de *Wide Area Network* (WAN), está ligada à Internet (rede externa).

No dispositivo *Raspberry Pi 3*, do tipo *Model B+*, é utilizado o SO *Raspbian versão Kernel 4.14*, que fornece o serviço de *Dynamic Host Configuration Protocol* (DHCP) para a rede interna. Este dispositivo tem também instalado o elemento de sonda para um sistema IDS, que tem como responsabilidade capturar os pacotes IP que fluem entre a rede interna e a Internet.

Relativamente ao dispositivo *Ubuntu 2*, este está instalado sobre uma plataforma de *hardware Intel x86-64* e utiliza como SO o *Ubuntu Desktop 18.04 LTS*. Este dispositivo está interligado com o encaminhador de periferia através da interface *Wi-Fi*, sendo que o encaminhador de periferia fornece o serviço de DHCP para ele.

Nos dispositivos *Raspberry Pi 2* e *Raspberry Pi 3*, do tipo *Model B+*, é utilizado o SO *Raspbian Versão Kernel 4.14*. Ambos utilizam a interface *Wi-Fi* para interligarem com o encaminhador de periferia, sendo que o encaminhador de periferia também fornece o endereço IPv4 através do serviço de DHCP.

Por último, o dispositivo *Arduino MKR 1000*. Este dispositivo funciona sobre a plataforma de *hardware Arduino Yun* e tem uma interface *Wi-Fi*, que tem o propósito de interligar com o encaminhador de periferia. O endereçamento IPv4 é fornecido através do encaminhador de periferia recorrendo ao serviço de DHCP.

Em relação à rede externa, como mencionado, esta contém os dispositivos *Ubuntu Server 1*, *Ubuntu Server 2* e *Ubuntu Server 3*. Todos estes dispositivos utilizam a mesma plataforma de *hardware Intel x86-64* e o SO *Ubuntu Server 18.04 LTS*. Em termos de conectividade, estes elementos estão interligados com o sistema IoT através da interface *Fast Ethernet*. Igualmente para todos os dispositivos, o endereçamento IPv4 é fornecido através do encaminhador de periferia recorrendo ao serviço de DHCP.

Os dipositivos *Raspberry PI 2*, *Raspberry PI 3*, *Ubuntu 2* e o *Arduino MKR 1000* têm as funções de sensores e atuadores do sistema IoT. Estes dispositivos utilizam os protocolos MQTT e MQTTS e funcionam como *Publisher* ou *Subscriber* para os serviços MQTT e MQTTS.

O dispositivo *Ubuntu Server 1* contém o módulo central IDS e recebe, armazena e analisa os fluxos de tráfego de rede IP que são recolhidos e exportados pela sonda IDS, localizada no encaminhador de periferia.

Os dispositivos *Ubuntu Server 2* e *Ubuntu Server 3* têm as funções de *Broker* MQTT e MQTTS, de forma a armazenar e a enviar atualizações das informações recebidas e enviadas pelos *Publisher* MQTT e MQTTS.

Quanto aos dispositivos IoT MQTT e MQTTS que funcionam como *Publisher*, *Subscriber* e *Broker*, todos recorrem ao *software Eclipse Mosquito* (Eclipse Foundation, Inc, 2020). Este pacote de *software* permite desenvolver aplicações que façam com que os dispositivos funcionem como *Publisher* MQTT e MQTTS, na qual recolhem e enviam a temperatura do ar para um *Broker* MQTT ou MQTTS, através da atualização de um tópico, por exemplo. Quando o *Broker* MQTT ou MQTTS recebem a informação vinda do *Publisher*, enviam uma atualização para o *Subscriber* MQTT ou MQTTS, que tem subscrito o tópico correspondente ao tópico que o *Publisher* envia as informações sobre a temperatura do ar, por exemplo.

Na rede interna existe também um dispositivo com a função de dispositivo atacante, sendo denominado de Nó atacante. Este dispositivo está instalado sobre o *hardware Intel x86-64* e

utiliza como SO o *Ubuntu Desktop 18.04 LTS*. O Nó atacante tem apenas uma interface *Wi-Fi* para ligação à rede interna do sistema IoT e, à semelhança dos restantes dispositivos, recebe o endereçamento IPv4 via DHCP, fornecido pelo encaminhador de periferia.

O Nó atacante foi desenvolvido com o intuito de gerar ataques e intrusões aos serviços e ao sistema IoT. Os ataques e as intrusões foram criadas através das ferramentas de *net scan nmap* (nmap, 2020) e *hping* (hping, 2020), fazendo ataques de análise de rede, inundação e DoS. O objetivo destes ataques e intrusões é fazer um uso abusivo ou utilizar informações inválidas quando são efetuados pedidos ao *Broker* MQTT ou ao *Broker* MQTTS. Esta situação tem como consequência a produção de tráfego IP anormal e malicioso, que é posteriormente analisado pelo módulo central IDS.

As Tabelas seguintes são uma sumarização do mencionado sobre as redes e os seus dispositivos associados.

A Tabela 16 identifica os dispositivos MQTT e MQTTS que estão presentes na rede interna, e a Tabela 17 identifica os dispositivos MQTT e MQTTS que estão presentes na rede externa. Ambas as tabelas descrevem a descrição do dispositivo, o *hardware* utilizado, o respetivo SO utilizado, os serviços, a sua função e as interfaces de rede e onde se encontram ligados.

Tabela 16 - Elementos da rede Interna

Dispositivo	***Hardware***	**SO**	**Serviço(s)**	**Função**	**Interface(s) de rede**	**Ligado a**
Encaminhador de preferia	*Raspberry Pi 3 Model B+*	*Raspbian Versão Kernel 4.14*	Servidor DHCP	Ponto de saída para a internet	*Wi-Fi*	A todos os elementos via *Wi-Fi*
			Sonda YAF		*Fast Ethernet*	ligado a internet (rede externa) via *Fast Ethernet*
Raspberry PI 2	*Raspberry Pi 3 Model B+*	*Raspbian Versão Kernel 4.14*	Cliente DHCP	MQTTS *Publisher*	*Wi-Fi*	Encaminhador de periferia
Raspberry PI 3	*Raspberry Pi 3 Model B+*	*Raspbian Versão Kernel 4.14*	Cliente DHCP	MQTT *Subscriber*	*Wi-Fi*	Encaminhador de periferia
Arduino MKR 1000	*Arduino Yu*		Cliente DHCP	MQTT *Publisher*	*Wi-Fi*	Encaminhador de periferia
Ubuntu 2	*Intel x86-64*	*Ubuntu Desktop 18.04 LTS*	Cliente DHCP	MQTTS *Subscriber*	*Wi-Fi*	Encaminhador de periferia
Nó Atacante	*Intel x86-64*	*Ubuntu Desktop 18.04 LTS*	Cliente DHCP	Gerador de ataques e intrusões	*Wi-Fi*	Encaminhador de periferia

Tabela 17 - Elementos da rede Externa

Dispositivo	*Hardware*	SO	Serviço(s)	Função	Interface(s) de rede	Ligado a
Switch				"Ponte" entre a rede Interna e a rede Externa	*Fast Ethernet*	A todos os elementos via *Fast Ethernet* ligado ao Encaminhador de periferia via *Fast Ethernet*
Ubuntu Server 1	*Intel x86-64*	*Ubuntu Server 18.04 LTS*	Cliente DHCP	Módulo Central IDS	*Fast Ethernet*	*Switch*
Ubuntu Server 2	*Intel x86-64*	*Ubuntu Server 18.04 LTS*	Cliente DHCP	MQTTS *Broker*	*Fast Ethernet*	*Switch*
Ubuntu Server 3	*Intel x86-64*	*Ubuntu Server 18.04 LTS*	Cliente DHCP	MQTT *Broker*	*Fast Ethernet*	*Switch*

De seguida são descritas as caraterísticas implementadas pela sonda e pelo módulo central do sistema IDS.

5.1.1. Sonda IDS

A sonda monitoriza todo o tráfego que circula entre a rede externa e a rede interna, isto é, entre os clientes do serviço IoT e os servidores do serviço IoT. A Figura 23 (Santos, 2020) ilustra as diferentes etapas e os diferentes elementos que a sonda IDS tem incorporado (a observação de pacotes, a agregação das informações dos pacotes em fluxos de tráfego de rede e a exportação dos registos de fluxos de tráfego de rede IP recorrendo ao protocolo IPFIX).

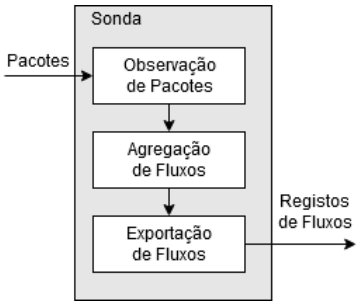

Figura 23 - *Workflow* com as etapas e elementos da sonda do IDS

O cenário de testes apresentado tem um domínio de observação e um ponto de observação, que se localizam na rede interna. O encaminhador de periferia (*Raspberry PI 3 Model B+*) tem a função de sonda IDS. A sonda recolhe os pacotes que são trocados entre a rede interna e a rede externa.

A sonda está implementada em modo espelho (*mirroring*) e implementada através do YAF, solução *open source* desenvolvida pelo *Carnegie Mellon University* (CERT), que exporta os registos de fluxos de tráfego de rede utilizando os IE selecionados para serem utilizados pela arquitetura proposta, descritos no ponto 4.4.5.

O método de integração do YAF na sonda IDS efetua-se através de um *daemon*. O YAF tem a função de realizar as três etapas na sonda IDS: a observação de pacotes, a agregação de fluxos de tráfego de rede e a exportação de fluxos de tráfego de rede. Também de forma a que a sonda IDS tenha melhor performance na recolha dos pacotes de rede foi instalado adicionalmente a biblioteca *PF_RING* (ntop, 2020).

O YAF é configurado de acordo com as seguintes características:

- Captura de pacotes de tráfego entre a rede interna e a rede externa;
- Captura de pacotes em tempo real com a utilização do *PF_RING*;
- Exportação dos fluxos de tráfego de rede através do IPFIX sobre TLS para um servidor que se encontra na rede externa, através do porto 4740;
- Os fluxos de tráfego de rede são removidos da cache e exportados quando atingem o tempo de inatividade de um minuto ou o tempo de atividade de três minutos;
- Inclusão das informações relativas aos endereços MAC, tal como as estatísticas de fluxos de tráfego de rede;
- Utiliza a lista dos IE presentes na Tabela 15.

5.1.2. Módulo central IDS

Posteriormente, os fluxos de tráfego de rede exportados pela sonda são enviados para o módulo de análise, denominado de módulo central IDS (Santos, 2020). Este módulo está localizado na rede externa, mais propriamente no *Ubuntu Server 1*, e tem como funções receber, armazenar e analisar os fluxos de tráfego de rede IP que são enviados pela sonda IDS.

A Figura 24 (Santos, 2020) ilustra as diferentes etapas e os elementos que estão presentes no módulo central IDS (recolha dos registos de fluxos de tráfego enviados pelo protocolo IPFIX, análise dos registos de fluxos de tráfego de rede IP armazenados com a base de dados de especificações, emissão e disseminação de mensagens de alerta de intrusão).

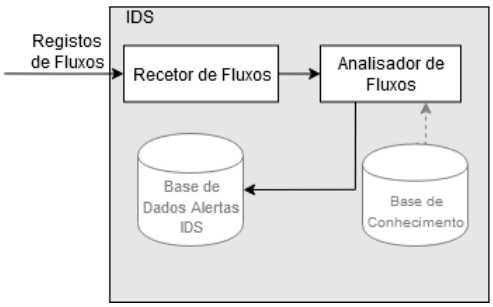

Figura 24 - *Workflow* com as etapas e elementos do módulo central do ID

Dentro do módulo central IDS existe um elemento denominado *super_mediator* (Carnegie Mellon University, 2020). Este elemento é uma solução *open source*, desenvolvido pelo CERT, que tem as funções de recolher, analisar e armazenar os fluxos de tráfego de rede.

O *super_mediator* recebe e descodifica as mensagens vindas do protocolo IPFIX, sendo utilizado em simultâneo com o YAF. À semelhança do YAF, o *super_mediator* é executado em modo *daemon*, sendo configurado com as opções de escuta e receção dos registos de fluxos de tráfego de rede IP com as caraterísticas abaixo descritas:

- Monitoriza a interface *Fast Ethernet*;
- Recebe os fluxos de tráfego de rede enviados pela sonda através do protocolo IPFIX sobre TLS no porto 4740;
- Os fluxos de tráfego de rede são armazenados em disco em formato *JavaScript Object Notation* (JSON);
- A cada 240 segundos é executado a rotação dos ficheiros JSON.

Nas funções de recolha e nos respetivos armazenamentos dos registos de fluxos de tráfego de rede são criados e adicionados ficheiros JSON numa diretoria presente no *Ubuntu Server 1*.

A Figura 25 é um exemplo do conteúdo de um ficheiro que contém vários registos de fluxos de tráfego de rede. Sendo que a Figura 26 mostra o conteúdo de um desses registos de fluxos de tráfego de rede com maior detalhe.


```
{"flows":{"flowStartMilliseconds":"2019-06-06 17:09:05.807","flowEndMilliseconds":"2019-06-06 17:12:05.431","flowDurationMilliseconds":179.624,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:12:35.370","flowEndMilliseconds":"2019-06-06 17:15:34.998","flowDurationMilliseconds":179.628,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:22:34.067","flowEndMilliseconds":"2019-06-06 17:25:33.696","flowDurationMilliseconds":179.629,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:32:02.878","flowEndMilliseconds":"2019-06-06 17:35:02.507","flowDurationMilliseconds":179.629,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:39:01.995","flowEndMilliseconds":"2019-06-06 17:42:01.651","flowDurationMilliseconds":179.656,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:42:31.590","flowEndMilliseconds":"2019-06-06 17:45:31.224","flowDurationMilliseconds":179.634,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:49:30.711","flowEndMilliseconds":"2019-06-06 17:52:30.334","flowDurationMilliseconds":179.623,
{"flows":{"flowStartMilliseconds":"2019-06-06 17:55:59.916","flowEndMilliseconds":"2019-06-06 17:58:59.565","flowDurationMilliseconds":179.649,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:02:59.059","flowEndMilliseconds":"2019-06-06 18:05:58.691","flowDurationMilliseconds":179.632,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:06:28.624","flowEndMilliseconds":"2019-06-06 18:09:28.230","flowDurationMilliseconds":179.606,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:13:27.707","flowEndMilliseconds":"2019-06-06 18:16:27.341","flowDurationMilliseconds":179.634,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:19:56.917","flowEndMilliseconds":"2019-06-06 18:22:56.562","flowDurationMilliseconds":179.645,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:26:56.054","flowEndMilliseconds":"2019-06-06 18:29:55.684","flowDurationMilliseconds":179.630,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:30:25.619","flowEndMilliseconds":"2019-06-06 18:33:25.232","flowDurationMilliseconds":179.613,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:37:24.700","flowEndMilliseconds":"2019-06-06 18:40:24.305","flowDurationMilliseconds":179.605,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:43:53.914","flowEndMilliseconds":"2019-06-06 18:46:53.548","flowDurationMilliseconds":179.634,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:50:52.999","flowEndMilliseconds":"2019-06-06 18:53:52.618","flowDurationMilliseconds":179.619,
{"flows":{"flowStartMilliseconds":"2019-06-06 18:54:22.558","flowEndMilliseconds":"2019-06-06 18:57:22.191","flowDurationMilliseconds":179.633,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:01:21.686","flowEndMilliseconds":"2019-06-06 19:04:21.341","flowDurationMilliseconds":179.655,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:07:50.915","flowEndMilliseconds":"2019-06-06 19:10:50.580","flowDurationMilliseconds":179.665,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:17:49.729","flowEndMilliseconds":"2019-06-06 19:20:49.363","flowDurationMilliseconds":179.634,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:21:19.303","flowEndMilliseconds":"2019-06-06 19:24:18.955","flowDurationMilliseconds":179.652,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:34:17.781","flowEndMilliseconds":"2019-06-06 19:37:17.436","flowDurationMilliseconds":179.655,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:40:47.037","flowEndMilliseconds":"2019-06-06 19:43:46.684","flowDurationMilliseconds":179.647,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:50:45.864","flowEndMilliseconds":"2019-06-06 19:53:45.553","flowDurationMilliseconds":179.689,
{"flows":{"flowStartMilliseconds":"2019-06-06 19:54:15.495","flowEndMilliseconds":"2019-06-06 19:57:15.156","flowDurationMilliseconds":179.661,
{"flows":{"flowStartMilliseconds":"2019-06-06 20:01:14.707","flowEndMilliseconds":"2019-06-06 20:04:14.384","flowDurationMilliseconds":179.677,
```


Figura 25 - Ficheiro JSON com registos de fluxos de tráfego de rede

```
{"flows":
{"flowStartMilliseconds":"2019-06-06
17:06:06.162",
"flowEndMilliseconds":"2019-06-06 17:06:28.118",
"flowDurationMilliseconds":21.956,
"reverseFlowDeltaMilliseconds":0.001,
"protocolIdentifier":6,
"sourceIPv4Address":"10.42.0.94",
"sourceTransportPort":50936,
"packetTotalCount":6,
"octetTotalCount":284,
"flowAttributes":"00",
"sourceMacAddress":"b8:27:eb:fc:b4:95:",
"destinationIPv4Address":"192.168.111.22",
"destinationTransportPort":1883,
"reversePacketTotalCount":6,
"reverseOctetTotalCount":248,
"reverseFlowAttributes":"00",
"destinationMacAddress":"08:00:27:6f:48:8a:",
"initialTCPFlags":"S",
"unionTCPFlags":"APF",
"reverseInitialTCPFlags":"AS",
"reverseUnionTCPFlags":"APF",
"tcpSequenceNumber":"0xa968782b",
"reverseTcpSequenceNumber":"0x4f9c13cf",
"ingressInterface":0,
"egressInterface":0,
"vlanId":"0x000",
"silkAppLabel":0,
"ipClassOfService":"0x00",
"flowEndReason":"",
"collectorName":"C1",
"tcpUrgTotalCount":0,
"smallPacketCount":2,
"nonEmptyPacketCount":2,
"dataByteCount":40,
"averageInterarrivalTime":4391,
"firstNonEmptyPacketSize":27,
"largePacketCount":0,
"maxPacketSize":27,
"firstEightNonEmptyPacketDirections":"02",
"standardDeviationPayloadLength":7,
"standardDeviationInterarrivalTime":8771,
"bytesPerPacket":20,
"reverseTcpUrgTotalCount":0,
"reverseSmallPacketCount":1,
"reverseNonEmptyPacketCount":1,
"reverseDataByteCount":4,
"reverseAverageInterarrivalTime":4391,
"reverseFirstNonEmptyPacketSize":4,
"reverseLargePacketCount":0,
"reverseMaxPacketSize":4,
"reverseStandardDeviationPayloadLength":0,
"reverseStandardDeviationInterarrivalTime":8746,
"reverseBytesPerPacket":4}}
```


Figura 26 - Estrutura de um registo de fluxo de tráfego de rede IPFIX

De forma a analisar os registos de fluxos de tráfego de rede que foram recebidos, descodificados e armazenados no módulo central IDS no formato JSON, foi desenvolvido um *script* em *Python*, (aplicação IDS) .

Adaptado de Santos, (2020), a Figura 27 ilustra o *workflow* do *script* utilizado para a validação.

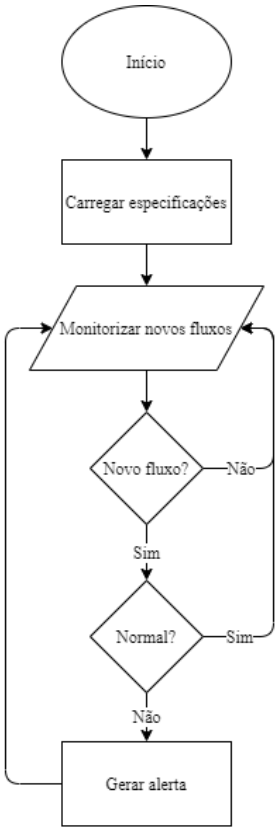

Figura 27 - *Workflow* do processo de análise dos fluxos de tráfego de rede no IDS

De acordo com a Figura 27, a aplicação IDS carrega primeiramente a base de dados de especificações do comportamento normal do tráfego MQTT e MQTTS, que estão armazenadas no formato JSON.

As especificações do comportamento normal do tráfego MQTT e do MQTTS consideram os vários tipos de mensagens geradas como, por exemplo, as subscrições de tópicos ou publicações de mensagens.

As especificações foram determinadas considerando o comportamento normal do tráfego em ambiente real MQTT e MQTTS, associando às capturas de pacotes de rede e às recolhas de registos de fluxos de tráfego de rede. As especificações foram desenvolvidas para um nível de QoS igual a 0, onde o *Publisher* e o *Subscriber* são os tipos de mensagens mais comuns. O MQTT e o MQTTS recorrem ao protocolo de transporte TCP, na qual o porto por defeito é o 1883 para o tráfego MQTT e para o tráfego MQTTS o porto por defeito é o 8883.

O módulo central IDS recebe os novos registos de fluxos de tráfego de rede recolhidos, descodificados e armazenados pelo *super_mediator*. A aplicação IDS averigua se existem alterações nos registos de fluxos de tráfego de rede armazenados no módulo central IDS e,

caso existam novos registos de fluxos de tráfego de rede, então a aplicação IDS compara os novos registos de fluxos com as especificações carregadas no início da aplicação IDS, sendo que pode acontecer uma das duas situações: a aplicação IDS classifica o registo como normal ou a aplicação IDS classifica o registo como anormal.

De acordo com a classificação de cada registo de fluxo de tráfego de rede, o *script* apresenta comportamentos distintos que a seguir são explicados:

- Para um registo de fluxo de tráfego de rede classificado como normal - O *script* termina a análise e reinicia o processo, de modo a verificar novos registos;
- Para um registo de fluxo classificado como anormal - O *script* gera uma mensagem de alerta através do protocolo *syslog*, sendo que o mesmo é armazenado num ficheiro de *log* na base de dados de alertas do IDS. Posteriormente, os alertas podem ser enviados para um *Security Information and Event Management* (SIEM) (varonis, 2020) através de um canal seguro.

De seguida, são descritas as especificações dos protocolos MQTT e MQTTS que serão utlizadas na aplicação IDS, com o objetivo de fazer a deteção de intrusões.

5.2.Especificações dos protocolos MQTT e MQTTS

O protótipo utilizado tem o QoS definido com o valor 0 (sem QoS). O valor 0 para QoS influência o volume de mensagens MQTT e MQTTS trocadas, ou seja, com o QoS igual a 0 não existem mensagens MQTT a confirmar os envios ou as receções de mensagens MQTT ou MQTTS, sendo que, para QoS diferentes de 0, esta confirmação já aconteceria.

As especificações dos protocolos MQTT e MQTTS, que serão utilizadas para análise dos fluxos de tráfego de rede, foram desenvolvidas a partir dos fluxos de tráfego sem anomalias. Os fluxos de tráfego de rede sem anomalias foram gerados pelo protótipo de testes, que gerou os *Packet CAPture* (PCAP) produzidos pela sonda IDS e os JSON produzidos pelo *super_mediator*.

Dos IE descritos na Tabela 15 (capítulo 4.4.5) e após análise dos fluxos de tráfego de rede sem anomalias obteve-se uma lista mais curta de IE relevantes para as especificações do tráfego dos protocolos MQTT e MQTTS, que estão descritos na Tabela 18.

Tabela 18 - Lista dos IE selecionados para o MQTT e MQTTS

IE	Descrição
ProtocolIdentifier	Identificação do protocolo, será 6 (TCP)
DestinationTransportPort	Porto de destino. Para o MQTT: 1883 e para o MQTTS: 8883
FlowEndReason	Código de fim do fluxo, conforme definido no IPFIX. Valores possíveis: *Active* ou vazio.
NonEmptyPacketCount	Número de pacotes com *payload* MQTT ou MQTTS, vindos do *Publisher* ou do *Subscriber* para o *Broker*.
ReverseNonEmptyPacketCount	Número de pacotes com *payload* MQTT ou MQTTS, vindos do *Broker* para *Publisher* ou para *Subscriber*.
DataByteCount	Somatório do *payload* dos pacotes com conteúdo MQTT ou MQTTS, vindos do *Publisher* ou do *Subscriber* para o *Broker*.
ReverseDataByteCount	Somatório do *payload* dos pacotes com conteúdo MQTT ou MQTTS, vindos do *Broker* para *Publisher* ou para *Subscritor*.
FirstNonEmptyPacketSize	Tamanho do *payload* do primeiro pacote com conteúdo MQTT ou MQTTS, vindos do *Publisher* ou do *Subscriber* para o *Broker*.
ReverseFirstNonEmptyPacketSize	Tamanho do *payload* do primeiro pacote com conteúdo MQTT ou MQTTS, vindos do *Broker* para *Publisher* ou para *Subscriber*.
unionTCPFlags	União das *flags* TCP de todos os outros pacotes, com exceção da inicial, no sentido origem do fluxo.

No capítulo 2.4.5 foi especificado o método de funcionamento do protocolo MQTT. O protocolo MQTT funciona através do protocolo TCP e através do porto por defeito de 1883. Os registos dos fluxos de tráfego de rede ou das sequências de mensagens geradas têm em conta o tamanho mínimo de um pacote MQTT, que é de dois *bytes*.

No capítulo 2.4.6, relativamente ao protocolo MQTTS, o protocolo MQTTS funciona através do protocolo TLS e através do porto por defeito de 8883. Os registos dos fluxos de tráfego de rede ou das sequências de mensagens geradas têm em conta o tamanho mínimo de um pacote TLS, isto é, quando a sessão está a ser estabelecida o tamanho mínimo é de 1049 *bytes* e quando a sessão já está estabelecida o tamanho mínimo é 22 *bytes*.

No mesmo capítulo também são descritas as caraterísticas e as sequências de trocas de fluxos de tráfego de rede de mensagens entre os vários dipositivos (*Publisher*, *Broker* e o *Subscriber*). O tráfego é cifrado, através do recurso à encriptação efetuada pelo TLS.

O protocolo MQTTS funciona da mesma forma que o protocolo MQTT e com as mesmas mensagens trocadas, contudo, tem uma camada adicional denominada de TLS, como referido no capítulo 2.4.6.

Os fluxos de tráfego de rede MQTT e MQTTS dividem-se em vários tipos de tráfego:

- tráfego MQTT - tráfego entre o *Publisher* e o *Broker*, também descrito como tráfego do tipo *Publisher*;
- tráfego MQTT - tráfego entre o *Subscriber* e o *Broker*, também descrito como por tráfego do tipo *Subscriber*;
- tráfego MQTTS - tráfego entre o *Publisher* e o *Broker*, também descrito como tráfego do tipo *Publisher*;
- tráfego MQTTS - tráfego entre o *Subscriber* e o *Broker*, também descrito como tráfego do tipo *Subscriber*.

Os tipos de tráfego MQTT e MQTTS possuem IE e especificações comuns a todos, que estão apresentadas na Tabela 19.

Tabela 19 - Especificações comuns para os IE

IE	Valor
protocolIdentifier	6
destinationTransportPort	MQTT: 1883 MQTTS: 8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
nonEmptyPacketCount	!= *null*
unionTCPFlags	!= apr

De seguida, é efetuada a descrição de cada um dos tipos de tráfego MQTT e MQTTS.

5.2.1. MQTT- tráfego entre o *Publisher* e o *Broker*

No tráfego entre o *Publisher* e o *Broker*, o *Publisher* estabelece, em primeiro lugar, a conexão com o *Broker*. Após a aceitação da conexão pelo *Broker*, o *Publisher* envia a publicação (tópico, informação) para o *Broker*, na qual, após o envio da publicação é efetuado o término da conexão com o *Broker*.

Este descritivo é esquematizado na Figura 28, onde é ilustrado o fluxo de tráfego entre o *Publisher* e o *Broker* mencionado.

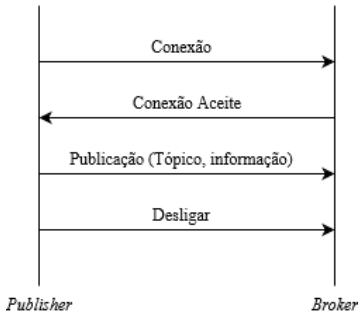

Figura 28 - Fluxo de tráfego entre o *Publisher* e o *Broker* sem TLS

Neste tipo de tráfego existem três pacotes trocados do *Publisher* para o *Broker*. A soma mínima dos tamanhos dos três pacotes é de 16 *bytes*, sendo que o tamanho global dos três pacotes pode aumentar de acordo com o tamanho da publicação. Nota-se que o tamanho mínimo de um pacote MQTT é de dois *bytes*, esta informação é importante para a determinação do tamanho mínimo de *payload* de um fluxo MQTT.

De seguida, apresenta-se a descrição de cada um dos pacotes enviados, que são respetivamente: o pedido de conexão (*Connect Command*), a publicação da mensagem (*Publish Message*) e o desligar (*Disconnect Req*).

O pedido de conexão (*Connect Command*) é o primeiro pacote trocado neste tipo de tráfego e transmite o pedido de conexão a um tópico. O tamanho mínimo deste pacote é de nove *bytes*, sendo que este tamanho pode aumentar consoante o tamanho do tópico e da mensagem enviada do *Publisher* e o *Broker*.

A Figura 29 apresenta o conteúdo do *payload* do pacote MQTT para o pedido de conexão.

```
MQ Telemetry Transport Protocol, Connect Command
  Header Flags: 0x10, Message Type: Connect Command
  Msg Len: 35
  Protocol Name Length: 4
  Protocol Name: MQTT
  Version: MQTT v3.1.1 (4)
  Connect Flags: 0x02, QoS Level: At most once delivery (Fire and Forget), Clean Session Flag
  Keep Alive: 60
  Client ID Length: 23
  Client ID: mosq-6fFwLdc79ZtvWVPruc
```

Figura 29 - MQTT *Connect Command*

A publicação da mensagem (*Publish Message*) é o segundo pacote trocado neste tipo de tráfego e transmite a mensagem do *Publisher* para o *Broker*. O tamanho mínimo deste pacote é de cinco *bytes* e também pode aumentar consoante o tamanho do tópico e da mensagem.

A Figura 30 mostra o conteúdo do *payload* do pacote MQTT para a publicação da mensagem.

```
MQ Telemetry Transport Protocol, Publish Message
> Header Flags: 0x30, Message Type: Publish Message,
  Msg Len: 9
  Topic Length: 4
  Topic: test
  Message: 4f4646
```

Figura 30 - MQTT *Publish Message*

O desligar (*Disconnect Req*) é o terceiro pacote trocado neste tipo de tráfego e transmite o pedido para terminar a conexão do *Publisher* para o *Broker*. Este pacote tem o tamanho fixo de dois *bytes*.

A Figura 31 mostra o conteúdo do *payload* do pacote MQTT para o desligar.

```
MQ Telemetry Transport Protocol, Disconnect Req
> Header Flags: 0xe0, Message Type: Disconnect Req
  Msg Len: 0
```

Figura 31 - MQTT *Disconnect Req*

Neste tipo de tráfego existe somente um pacote trocado do *Broker* para o *Publisher* e transmite a aceitação da conexão (*Connect Ack*). Este pacote tem o tamanho fixo de quatro *bytes*.

A Figura 32 mostra o conteúdo do *payload* do pacote MQTT para a aceitação da conexão.

```
MQ Telemetry Transport Protocol, Connect Ack
> Header Flags: 0x20, Message Type: Connect Ack
  Msg Len: 2
> Acknowledge Flags: 0x00
  Return Code: Connection Accepted (0)
```

Figura 32 - MQTT *Connect Ack*

De seguida, é efetuada a especificação do tipo de tráfego entre o *Publisher* e o *Broker* para cada um dos IE relevantes para o MQTT.

Os IE relevantes para a identificação do tráfego MQTT entre o *Publisher* e o *Broker* são mencionados e descritos de seguida:

- nonEmptyPacketCount - são enviados neste sentido três pacotes com *payload*;

- reverseNonEmptyPacketCount - são enviados neste sentido apenas um pacote com *payload*;
- dataByteCount - neste sentido são enviados três pacotes com *payload* de diferentes tamanhos. O somatório destes três pacotes é de 16 *bytes*, na qual abaixo é especificado o tamanho do *payload* de cada pacote:
    - o primeiro pacote tem o tamanho mínimo de nove *bytes*;
    - o segundo pacote tem o tamanho mínimo de cinco *bytes*;
    - e o terceiro pacote tem o tamanho fixo dois *bytes*.
- reverseDataByteCount - é enviado um pacote com o tamanho fixo de *payload* de quatro *bytes*;
- firstNonEmptyPacketSize - o primeiro pacote enviado tem o tamanho mínimo de nove *bytes*;
- reverseFirstNonEmptyPacketSize - o primeiro pacote enviado tem o tamanho fixo de quatro *bytes*.

A Tabela 20 contém um resumo da caracterização genérica para o tipo de tráfego MQTT entre o *Publisher* e o *Broker*.

Tabela 20 - MQTT- Especificação do tráfego entre o *Publisher* e o *Broker*

IE	Valor
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e == 3
reverseNonEmptyPacketCount	== 1
dataByteCount	>= 16
reverseDataByteCount	== 4
firstNonEmptyPacketSize	>= 9
reverseFirstNonEmptyPacketSize	== 4

5.2.2. MQTT - tráfego entre o *Subscriber* e o *Broker*

No tráfego entre o *Subscriber* e o *Broker*, o S*ubscriber* estabelece, em primeiro lugar, a conexão com o *Broker*. Após a aceitação da conexão pelo *Broker*, o *Subscriber* subscrever um tópico e recebe publicações relativas a esse tópico, que são enviadas pelo *Broker*.

Periodicamente o *Subscriber* envia mensagens de *ping* para o *Broker*, de modo a manter a conexão ativa. Quando o *Subscriber* tenciona deixar de receber publicações, faz o cancelamento da subscrição do tópico e, de seguida, o respetivo término de conexão.

Este descritivo é esquematizado na Figura 33, onde é ilustrado o fluxo de tráfego entre o *Subscriber* e o *Broker* sem o TLS.

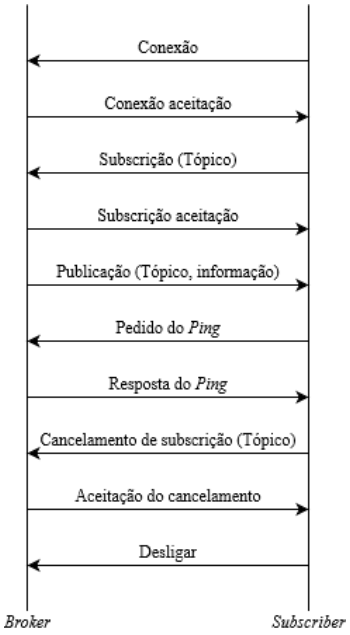

Figura 33 - MQTT Fluxo de mensagem entre o *Subscriber* e o *Broker* sem TLS

Este tipo de tráfego é exportado em vários tipos de fluxos de tráfego de rede de mensagens. Esta separação em tipos de fluxos de tráfego de rede tem em conta os tempos de exportação da sonda YAF, sendo que, poderão ser exportados os fluxos tráfego de rede incompletos, ou seja, fluxos de tráfego de rede que só contenham parte do tráfego. Por exemplo, foram observados registos de fluxos de tráfego de rede que só continham uma publicação e mensagens de *ping*.

De seguida, é separado e explicado cada tipo de tráfego de rede MQTT entre o *Subscriber* e o *Broker*:

- Conexão, *Connect Command* e *Connect Ack*;
- Subscrição de um tópico, *Subscribe Request* e *Subscribe Ack*;
- Publicação de uma mensagem, *Publish Message*;
- Mensagens de *ping*, *Ping Request* e *Ping Response*;
- Cancelamento de uma subscrição a um tópico, *Unsubscribe Request* e *Unsubscribe Ack*;
- Término de conexão, *Disconnect Req*.

Conexão

A conexão é composta por dois pacotes trocados entre o *Subscriber* e o *Broker*, que são respetivamente: o *Connect Command* e o *Connect Ack.*

Do *Subscriber* para o *Broker* há um pacote trocado (*Connect Command*), com o tamanho mínimo de nove *bytes*. A Figura 34 mostra o conteúdo do *payload* do pacote MQTT.

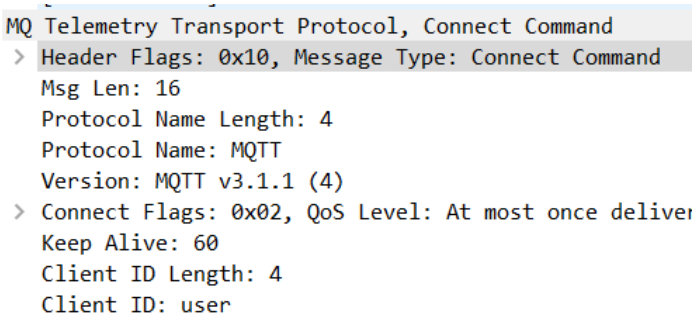

Figura 34 - MQTT *Connect Command*

Do *Broker* para o *Subscriber* existe o envio de um pacote com a aceitação da conexão (*Connect Ack*) e tem o tamanho fixo de quatro *bytes*. A Figura 35 mostra o conteúdo do *payload* do pacote MQTT.

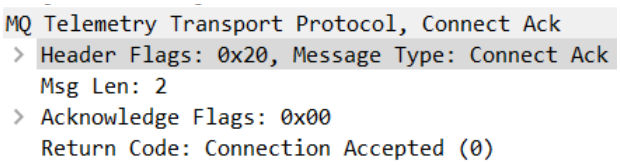

Figura 35 - MQTT *Connect Ack*

Subscrição de um tópico

O *Subscriber* faz a subscrição de um tópico presente no *Broker*, sendo a subscrição constituída por dois pacotes: o *Subscribe Request* e o *Subscribe Ack*:

Do *Subscriber* para o *Broker* há um pacote trocado (*Subscribe Request*) com o tamanho mínimo de seis *bytes*, o qual o tamanho pode aumentar de acordo com o tamanho do tópico a subscrever. A Figura 36 mostra o conteúdo do *payload* do pacote MQTT.

```
MQ Telemetry Transport Protocol, Subscribe Request
 > Header Flags: 0x82, Message Type: Subscribe Request
   Msg Len: 9
   Message Identifier: 1
   Topic Length: 4
   Topic: test
   Requested QoS: At most once delivery (Fire and Forget) (0)
```

Figura 36 - MQTT *Subscribe Request*

Do *Broker* para o *Subscriber* existe o envio de um pacote com aceitação da subscrição (Subscribe Ack) que tem o tamanho fixo de quatro *bytes*. A Figura 37 mostra o conteúdo do *payload* do pacote MQTT.

```
MQ Telemetry Transport Protocol, Subscribe Ack
 > Header Flags: 0x90, Message Type: Subscribe Ack
   Msg Len: 3
   Message Identifier: 1
   Granted QoS: At most once delivery (Fire and Forget) (0)
```

Figura 37 - MQTT *Subscribe Ack*

Publicação de uma mensagem

Do *Subscriber* para o *Broker* existe o envio de um pacote trocado (*Publish Message*) que tem o tamanho mínimo de cinco *bytes*, na qual o tamanho pode aumentar de acordo com o tamanho da mensagem enviada pelo *Broker*. A Figura 38 mostra o conteúdo do *payload* do pacote MQTT.

```
MQ Telemetry Transport Protocol, Publish Message
 > Header Flags: 0x30, Message Type: Publish Mess
   Msg Len: 9
   Topic Length: 4
   Topic: test
   Message: 4f4646
```

Figura 38 - MQTT *Publish Message*

Mensagens de *ping*

De forma a manter a conexão ativa entre o *Subscriber* e o *Broker*, são enviados periodicamente dois pacotes que são respetivamente: o *Ping Request* e o *Ping Response*.

Do *Subscriber* para o *Broker* existe o envio de um pacote trocado (*Ping Request*), sendo que este pacote tem o tamanho fixo de dois *bytes*. A Figura 39 mostra o conteúdo do *payload* do pacote MQTT.

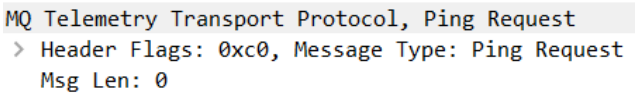

Figura 39 - MQTT *Ping Request*

Do *Broker* para o *Subscriber* existe o envio de um pacote trocado (*Ping Response*) que tem o tamanho fixo de dois *bytes*. A Figura 40 mostra o conteúdo do *payload* do pacote MQTT.

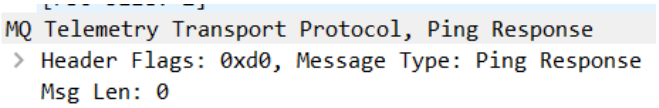

Figura 40 - MQTT *Ping Response*

Cancelamento de uma subscrição a um tópico

Quando o *Subscriber* pretende terminar a subscrição de um tópico, que se encontra no *Broker*, são trocados dois pacotes, que são respetivamente: o *Unsubscribe Request* e o *Unsubscribe Ack*.

Do *Subscriber* para o *Broker* existe o envio de um pacote trocado (*Unsubscribe Request*) que tem o tamanho mínimo de cinco *bytes*, na qual o tamanho pode aumentar de acordo com o tamanho do tópico. A Figura 41 mostra o conteúdo do *payload* do pacote MQTT.

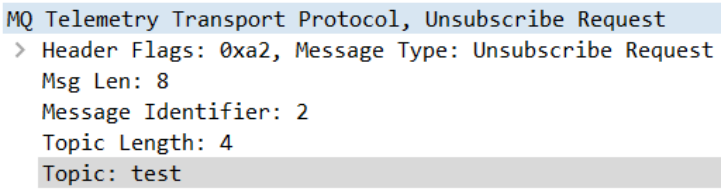

Figura 41 - MQTT *Unsubscribe Request*

Do *Broker* para o *Subscriber* existe o envio de um pacote trocado (*Unsubscribe Ack*) que tem o tamanho fixo de três *bytes*. A Figura 42 mostra o conteúdo do *payload* do pacote MQTT.

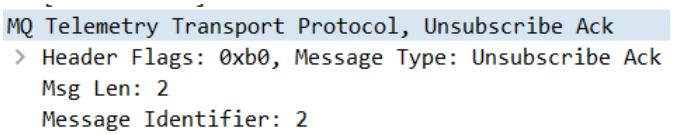

Figura 42 - MQTT *Unsubscribe Ack*

Término de conexão

Do *Subscriber* para o *Broker* existe o envio de um pacote trocado (*Disconnect Req*) que tem o tamanho fixo de dois *bytes*. A Figura 43 mostra o conteúdo do *payload* do pacote MQTT.

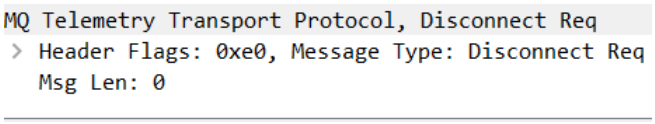

Figura 43 - MQTT *Disconnect Req*

De seguida, é explicada a especificação do tipo de tráfego entre o *Subscriber* e o *Broker*. Após análise da exportação dos fluxos de tráfego de rede e da análise dos registos fluxos de tráfego de rede, caraterizou-se o tráfego de forma generalista e de maneira a abranger as máximas situações encontradas durante a exportação do registo de fluxos de tráfego de rede.

É de realçar que a exportação dos fluxos de tráfego de rede pelo YAF, que ocorre num determinado intervalo de tempo (passado um minuto de inatividade ou três minutos de atividade), pode produzir fluxos que não contenham todo o fluxo de tráfego de tráfego de rede.

Na caracterização do tráfego, verifica-se que os registos de fluxos de tráfego de rede contêm as seguintes trocas de mensagens:

- O *Subscriber* inicia a sessão com o *Broker*;
- O *Subscriber* subscreve um tópico;
- O *Broker* envia as mensagens relativas a esse tópico para o *Subscriber*;
- O *Subscriber* para manter a sessão ativa envia mensagens de *ping* e o *Broker* responde a essas mesmas mensagens de *ping*.

Na Tabela 21 encontram-se as especificações para o tráfego MQTT de forma generalista, que resultam da combinação das mensagens identificadas anteriormente (a conexão, a subscrição de um tópico, a publicação de uma mensagem e as mensagens de *ping*).

Tabela 21 - Especificações generalistas para o tráfego MQTT - *Subscriber* para o *Broker*

IE	Descrição
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 3
reverseNonEmptyPacketCount	>= 3
dataByteCount	== 0 ou == 2 ou >= 6
reverseDataByteCount	== 2 ou >= 4
firstNonEmptyPacketSize	== 0 ou == 2 ou >= 6
reverseFirstNonEmptyPacketSize	== 2 ou >= 4

Durante a análise dos registos dos fluxos de tráfego de rede exportados pelo protótipo de testes observou-se que alguns fluxos de tráfego de rede não continham todo o tráfego. De acordo com esta situação, existiu a necessidade de criar especificações adicionais de forma a considerar estes tipos de fluxos de tráfego de rede.

Sendo assim, foram identificados fluxos de tráfego de rede que contêm somente as seguintes partes de tráfego:

- Conexão e Subscrição de um tópico;
- Publicação de uma mensagem;
- Mensagens de *ping;*
- Publicação de uma mensagem e mensagens de *ping*;
- Cancelamento de uma subscrição e o término de conexão.

Conexão e Subscrição de um tópico

A sequência de mensagens dos registos de fluxos de tráfego de rede com o início de sessão e a subscrição de um tópico é a seguinte:

- O *Subscriber* inicia a sessão para o *Broker*;
- O *Subscriber* subscreve um tópico presente no *Broker*.

Os IE relevantes para a identificação do tráfego MQTT entre o *Subscriber* e o *Broker* para a estabelecimento da conexão e a subscrição de um tópico estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;

- reverseNonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;
- dataByteCount - é enviado um pacote com *payload*, com o tamanho mínimo de seis *bytes*;
- reverseDataByteCount - é enviado um pacote com *payload*, com o tamanho mínimo de quatro *bytes*;
- firstNonEmptyPacketSize - o pacote com o tamanho mínimo de seis *bytes*;
- reverseFirstNonEmptyPacketSize - o pacote com o tamanho mínimo de quatro *bytes*;

Na Tabela 22 encontram-se as especificações para estes fluxos de tráfego de rede MQTT.

Tabela 22 - MQTT - Entre *Subscriber* e o *Broke*r, conexão e subscrição de um tópico

IE	Descrição
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 1
reverseNonEmptyPacketCount	>= 1
dataByteCount	>= 6
reverseDataByteCount	>= 4
firstNonEmptyPacketSize	>= 6
reverseFirstNonEmptyPacketSize	>= 4

Publicação de uma mensagem

Na análise dos registos de fluxos de tráfego de rede exportados verificou-se que existem fluxos de tráfego de rede que contêm apenas publicações de mensagens. Estas publicações têm origem no *Broker* e destino no *Subscriber*.

Os IE relevantes para a identificação do tráfego MQTT para este tráfego estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - não são enviados pacotes com *payload*;
- reverseNonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;
- dataByteCount - não são enviados pacotes com *payload*;
- reverseDataByteCount - é enviado um pacote com *payload*, com o tamanho mínimo de cinco *bytes*;
- firstNonEmptyPacketSize - não são enviados pacotes com *payload*;
- reverseFirstNonEmptyPacketSize - o pacote com o tamanho mínimo de cinco *bytes*.

Na Tabela 23 encontram-se as especificações para estes fluxos de tráfego de rede MQTT.

Tabela 23 - MQTT - Entre o *Subscriber* e o *Broker*, publicação de uma mensagem

IE	Descrição
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e == 0
reverseNonEmptyPacketCount	== 1
dataByteCount	== 0
reverseDataByteCount	>= 5
firstNonEmptyPacketSize	== 0
reverseFirstNonEmptyPacketSize	>= 5

Mensagens de *ping*

Existem registos de fluxos de tráfego de rede que apenas contêm as mensagens de *ping*, trocadas entre o *Subscriber* e o *Broker*, sendo que a sequência é a seguinte:

- *Ping Request*;
- *Ping Response*.

Os IE relevantes para a identificação do tráfego MQTT entre o *Subscriber* e o *Broker* para a troca de mensagens de *ping* estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado um pacote com *payload*;
- reverseNonEmptyPacketCount - é enviado um pacote com *payload*;
- dataByteCount - com o tamanho fixo de dois *bytes*;
- reverseDataByteCount - com o tamanho fixo de dois *bytes*;
- firstNonEmptyPacketSize - com o tamanho fixo de dois *bytes*;
- reverseFirstNonEmptyPacketSize - com tamanho fixo de dois *bytes*.

Na Tabela 24 encontra-se as especificações para as mensagens de *ping*.

Tabela 24 - MQTT - Entre o *Subscriber* e o *Broker*, mensagens de *ping*

IE	Descrição
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e == 1
reverseNonEmptyPacketCount	== 1
dataByteCount	== 2
reverseDataByteCount	== 2
firstNonEmptyPacketSize	== 2
reverseFirstNonEmptyPacketSize	== 2

Publicação de uma mensagem e mensagens de *ping*

Existem registos de fluxos de tráfego de rede que apenas contêm a publicação de mensagens e a respetiva troca de mensagens de *ping* entre o *Subscriber* e o *Broker*. A sequência das mensagens é a seguinte:

- O *Broker* envia as mensagens relativas a esse tópico para o *Subscriber*;
- *Ping Request*;
- *Ping Response*.

Os IE relevantes para este tipo de fluxos de tráfego de rede entre o *Subscriber* e o *Broker* estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;
- reverseNonEmptyPacketCount - são enviados pelos menos dois pacotes com *payload*;
- dataByteCount - com o tamanho mínimo de dois *bytes*;
- reverseDataByteCount - com o tamanho mínimo de quatro *bytes*;
- firstNonEmptyPacketSize – com o tamanho mínimo de dois *bytes*;
- reverseFirstNonEmptyPacketSize – com o tamanho mínimo de dois *bytes*.

As especificações são a combinação das mensagens anteriormente caraterizadas para o tráfego MQTT. Estas especificações são descritas na Tabela 25.

Tabela 25 - MQTT - Entre o *Subscriber* e o *Broker*, publicação de uma mensagem e mensagens de *ping*

IE	Descrição
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 1
reverseNonEmptyPacketCount	>= 2
dataByteCount	>= 2
reverseDataByteCount	>= 4
firstNonEmptyPacketSize	>= 2
reverseFirstNonEmptyPacketSize	>= 2

Cancelamento de uma subscrição e término de conexão

Existem registos de fluxos de tráfego de rede que contêm apenas o cancelamento de uma subscrição e o término de conexão entre o *Subscriber* e o *Broker*. As sequências de mensagens são as seguintes:

- Cancelamento da subscrição de um tópico;
- Término da conexão.

Os IE relevantes para este tráfego entre o *Subscriber* e o *Broker* para o cancelamento de uma subscrição e término da conexão estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado um ou dois pacotes com *payload*;
- reverseNonEmptyPacketCount - não são enviados pacotes com *payload*;
- dataByteCount - o tamanho de dois *bytes* ou tem o tamanho mínimo de quatro *bytes*;
- reverseDataByteCount - não são enviados pacotes com *payload*;
- firstNonEmptyPacketSize - o tamanho mínimo de dois bytes;
- reverseFirstNonEmptyPacketSize - o tamanho de dois *bytes* ou tem o tamanho mínimo de quatro *bytes*;

A Tabela 26 representa este tipo de tráfego MQTT para o cancelamento de uma subscrição e o término da conexão.

Tabela 26 - MQTT - Entre o *Subscriber* e o *Broker*, cancelamento de subscrição e o término da conexão

IE	Valor
protocolIdentifier	6
destinationTransportPort	1883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e (== 1 ou == 2)
reverseNonEmptyPacketCount	== 0
dataByteCount	== 2 ou >= 4
reverseDataByteCount	== 0
firstNonEmptyPacketSize	== 2 ou >= 4
reverseFirstNonEmptyPacketSize	== 0

5.2.3. MQTTS - tráfego entre o *Publisher* e o *Broker*

O tráfego MQTTS funciona da mesma forma que o tráfego MQTT, com a adição da camada de segurança TLS. No tráfego MQTTS, os pacotes que têm *payload* apresentam a camada *Application Data*: MQTT.

Conhecendo os tamanhos mínimos para o MQTT é possível determinar os tamanhos mínimos para o MQTTS, sendo que para tal é necessário adicionar ao MQTTS o tamanho do *overhead* do TLS.

Quando a sessão TLS está a ser a estabelecida (*handshake*) o tamanho mínimo dos pacotes relativos ao estabelecimento da sessão TLS é de 1049 *bytes*, sendo este o tamanho do *overhead* do TLS. Quando a sessão TLS já está estabelecida, isto é, somente existe tráfego MQTTS (*Application Data*), o tamanho mínimo dos pacotes é de 22 *bytes*.

Na Figura 44 descreve-se o fluxo de mensagens trocadas entre o *Publisher* e o *Broker* para o tráfego MQTTS.

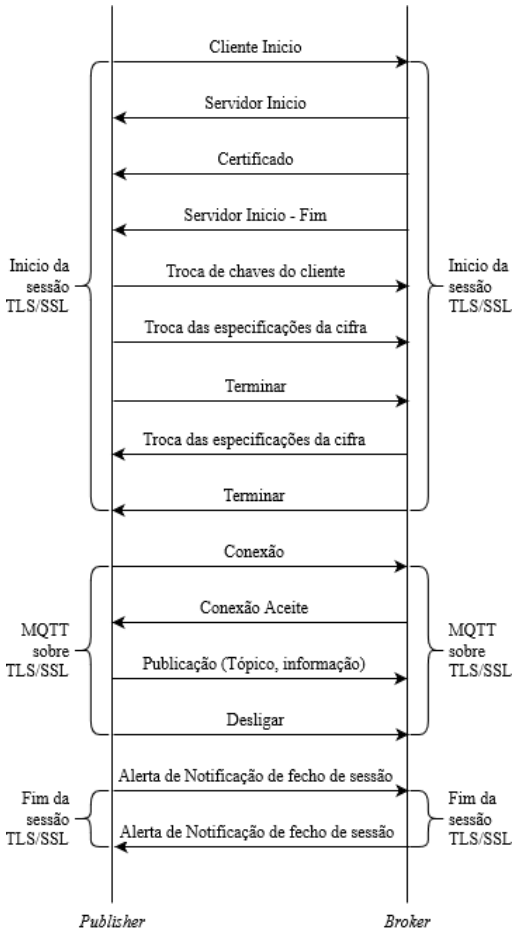

Figura 44 - Fluxo de mensagem entre o *Publisher* e o *Broker* com TLS

Neste tipo de tráfego existem três pacotes trocados do *Publisher* para o *Broker*. A soma mínima dos tamanhos dos três pacotes é de 82 *bytes*, sendo que o tamanho global dos três pacotes pode aumentar de acordo com o tamanho da publicação.

De seguida apresenta-se a descrição de cada um dos pacotes enviados, que são respetivamente: o pedido de conexão (*Connect Command*), a publicação da mensagem (*Publish Message*) e o desligar (*Disconnect Req*).

O pedido de conexão (*Connect Command*) é o primeiro pacote trocado neste tipo de tráfego e transmite o pedido de conexão a um tópico. O tamanho mínimo deste pacote é de 31 *bytes*, sendo que este tamanho pode aumentar consoante o tamanho do tópico e da mensagem enviada do *Publisher* e o *Broker*.

A Figura 45 apresenta o conteúdo do *payload* do pacote MQTTS para o pedido de conexão.

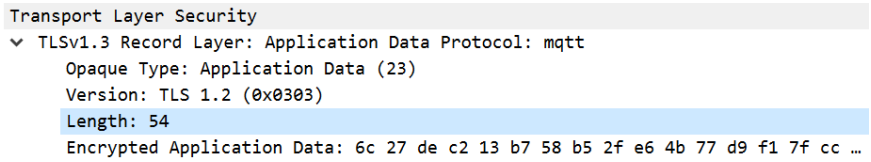

Figura 45 - MQTTS *Connect Command*

A publicação da mensagem (*Publish Message*) é o segundo pacote trocado neste tipo de tráfego e transmite a mensagem do *Publisher* para o *Broker*. O tamanho mínimo deste pacote é de 27 *bytes*, e pode aumentar consoante o tamanho do tópico e da mensagem.

A Figura 46 mostra o conteúdo do *payload* do pacote MQTTS para a publicação da mensagem.

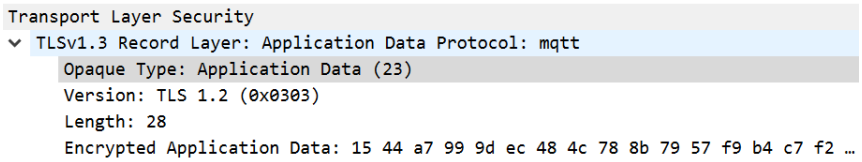

Figura 46 - MQTTS *Publish Message*

O desligar (*Disconnect Req*) é o terceiro pacote trocado neste tipo de tráfego e transmite o pedido para terminar a conexão do *Publisher* para o *Broker*. Este pacote tem o tamanho fixo de 24 *bytes.*

A Figura 47 mostra o conteúdo do *payload* do pacote MQTTS para o desligar.

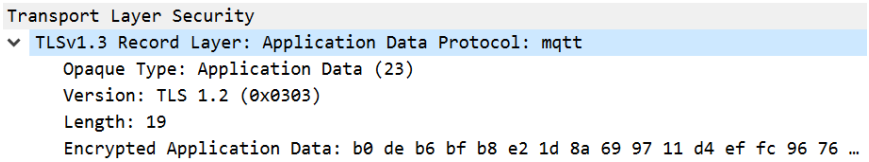

Figura 47 - MQTTS *Disconnect Req*

Neste tipo de tráfego existe somente um pacote trocado do *Broker* para o *Publisher* e transmite a aceitação da conexão (*Connect Ack*). Este pacote tem o tamanho fixo de 24 *bytes*.

A Figura 48 mostra o conteúdo do *payload* do pacote MQTTS para a aceitação da conexão.

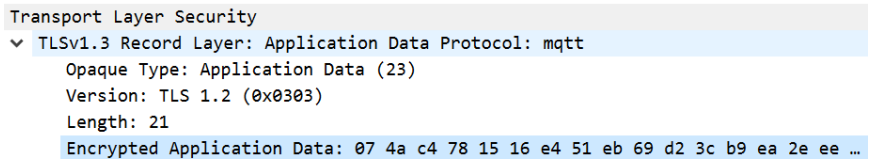

Figura 48 - MQTTS *Connect Ack*

De seguida, é efetuada a especificação do tipo de tráfego entre o *Publisher* e o *Broker* para cada um dos IE relevantes para o MQTTS.

Os IE relevantes para a identificação do tráfego MQTTS entre o *Publisher* e o *Broker* estão mencionados e descritos de seguida:

- nonEmptyPacketCount - neste sentido são enviados três pacotes com *payload*. Contudo, é também necessário contar com pacotes de início de sessão do TLS que, pelo menos, é um pacote. Portanto, desta maneira o número mínimo de pacotes enviados é de quatro pacotes com *payload*;
- reverseNonEmptyPacketCount - neste sentido apenas é trocado um pacote com *payload*. Contudo é também necessário contar com as trocas de mensagens TLS, ou seja, pelo menos mais três pacotes, dando um total de pelo menos quatro pacotes trocados com *payload*;
- dataByteCount - são enviados três pacotes com *payload* de diferentes tamanhos, na qual abaixo é especificado o tamanho do *payload* de cada pacote:
    - o primeiro pacote tem o tamanho mínimo 31 *bytes*;
    - o segundo pacote tem o tamanho mínimo de 27 *bytes*;
    - e o terceiro pacote tem o tamanho fixo 24 *bytes*.

  Os pacotes anteriores, que são pacotes que contém *payload* TLS, correspondem ao estabelecimento da sessão TLS e têm o tamanho mínimo de 410 *bytes*. O somatório dos tamanhos do *payload*, que corresponde ao estabelecimento da sessão TLS com a troca de pacotes MQTTS, tem um tamanho mínimo de 492 *bytes*;

- reverseDataByteCount - é enviado um pacote com o tamanho fixo de *payload* de 24 *bytes*. Contudo, são trocados pelos menos mais três pacotes com origem do *Broker* para *Publisher* com *payload* TLS, correspondendo ao estabelecimento da sessão TLS, sendo que o mínimo da soma desses três pacotes é de 1049 *bytes*. O tamanho mínimo para este IE é de 1073 *bytes*, sendo que este valor pode aumentar consoante o tamanho do *overhead* do TLS;

- firstNonEmptyPacketSize - é o pacote da sessão TLS e tem o tamanho mínimo de 306 *bytes*;
- reverseFirstNonEmptyPacketSize - é o pacote correspondente ao início da sessão TLS e tem o tamanho mínimo de 1049 *bytes*.

A Tabela 27 contém um resumo da caracterização genérica para o tráfego MQTTS para um fluxo entre o *Publisher* e o *Broker*.

Tabela 27 - MQTTS - Especificação do tráfego entre o *Publisher* e o *Broker*

IE	Valor
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 4
reverseNonEmptyPacketCount	>= 4
dataByteCount	>= 492
reverseDataByteCount	>= 1073
firstNonEmptyPacketSize	>= 306
reverseFirstNonEmptyPacketSize	>= 1049

5.2.4. MQTTS - tráfego entre o *Subscriber* e o *Broker*

O tráfego entre o *Subscriber* e o *Broker* no MQTTS funciona da mesma forma que o tráfego MQTT, com a adição de uma camada de segurança TLS. No tráfego MQTTS, os pacotes que têm *payload* apresentam a camada *Application Data*: MQTT.

O S*ubscriber* estabelece, em primeiro lugar, a conexão com o *Broker*. Após a aceitação da conexão pelo Broker, o *Subscriber* subscrever um tópico e recebe publicações relativas a esse tópico, que são enviadas pelo *Broker*.

Periodicamente o *Subscriber* envia mensagens de *ping* para o *Broker*, de modo a manter a conexão ativa. Quando o *Subscriber* tenciona deixar de receber publicações, faz o cancelamento da subscrição do tópico e de seguida o respetivo término de conexão.

Esta sequência é esquematizada na Figura 49, onde é ilustrado o fluxo de tráfego entre o *Subscriber* e o *Broker* com o TLS.

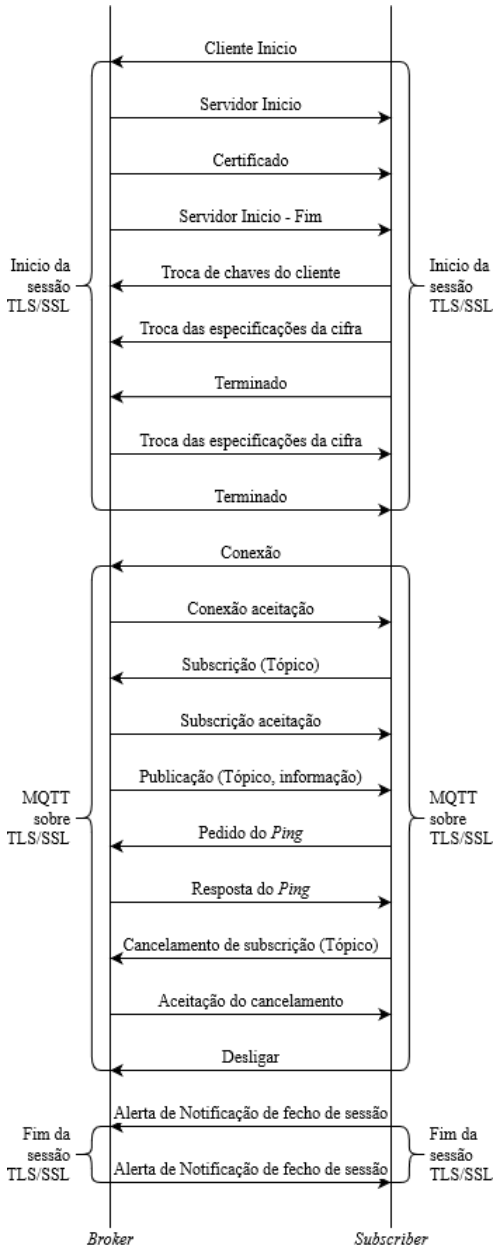

Figura 49 - MQTTS fluxo de tráfego entre o *Subscriber* e o *Broker* com o TLS

Este tipo de tráfego é exportado em vários tipos de fluxos de tráfego de rede de mensagens. Esta separação em tipos de fluxos de tráfego de rede tem em conta os tempos de exportação da sonda YAF, sendo que, poderão serem exportados fluxos incompletos, ou seja, fluxos de tráfego de rede que só contenham parte do tráfego. Por exemplo, foram observados registos de fluxos de tráfego de rede que só continham uma publicação e mensagens de *ping*.

De seguida, são apresentados e explicados os tipos de mensagens referentes ao tráfego MQTTS entre o *Subscriber* e o *Broker*:

- Conexão, o *Connect Command* e o *Connect Ack*;

- Subscrição de um tópico, o *Subscribe Request* e o *Subscribe Ack*;
- Publicação de uma mensagem, o *Publish Message*;
- Mensagens de *ping*, o *Ping Request* e o *Ping Response*;
- Cancelamento de uma subscrição de um tópico, o *Unsubscribe Request* e o *Unsubscribe Ack*;
- Término de conexão, o *Disconnect Req*.

Conexão

A conexão é caracterizada por dois pacotes trocados entre o *Subscriber* e o *Broker* que são, respetivamente: o *Connect Command* e o *Connect Ack*.

Do *Subscriber* para o *Broker* existe um pacote trocado (*Connect Command*) com o tamanho mínimo de 30 *bytes*. A Figura 50 mostra o conteúdo do *payload* do pacote MQTTS.

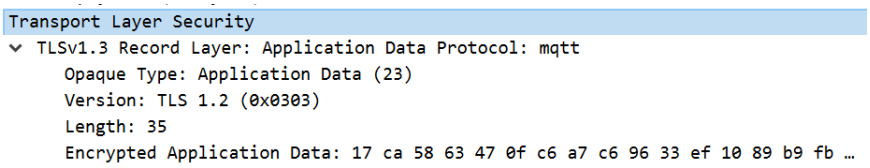

Figura 50 - MQTTS *Connect Command*

Do *Broker* para o *Subscriber* existe um pacote com aceitação da conexão (*Connect Ack*), com o tamanho fixo de 24 *bytes*. A Figura 51 mostra o conteúdo do *payload* do pacote MQTTS.

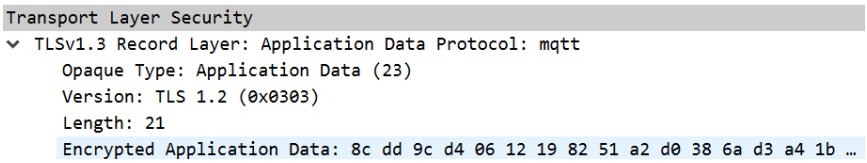

Figura 51 - MQTTS *Connect Ack*

Subscrição de um tópico

O *Subscriber* faz a subscrição de um tópico presente no *Broker*, sendo a subscrição constituída por dois pacotes: o *Subscribe Request* e o *Subscribe Ack*:

Do *Subscriber* para o *Broker* existe um pacote trocado (*Subscribe Request*) com o tamanho mínimo de 28 *bytes*, o qual o tamanho pode aumentar de acordo com o tamanho do tópico a subscrever. A Figura 52 mostra o conteúdo do *payload* do pacote MQTTS.

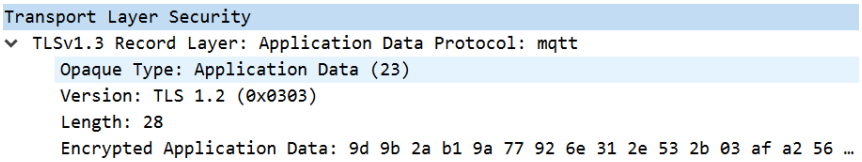

Figura 52 - MQTTS *Subscribe Request*

Do *Broker* para o *Subscriber* existe um pacote com aceitação da subscrição (*Subscribe Ack*) com o tamanho mínimo de 24 *bytes*. A Figura 53 mostra o conteúdo do *payload* do pacote MQTTS.

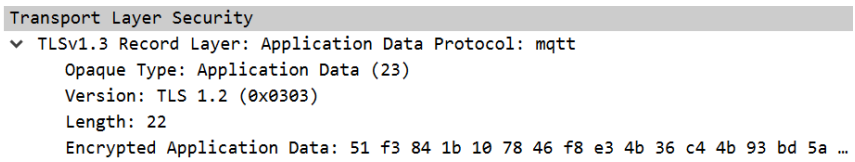

Figura 53 - MQTTS *Subscribe Ack*

Publicação de uma mensagem

Do *Subscriber* para o *Broker* existe um pacote trocado (*Publish Message*) com o tamanho mínimo de 27 *bytes*, o qual o tamanho pode aumentar de acordo com o tamanho da mensagem enviada pelo *Broker*. A Figura 54 mostra o conteúdo do *payload* do pacote MQTT.

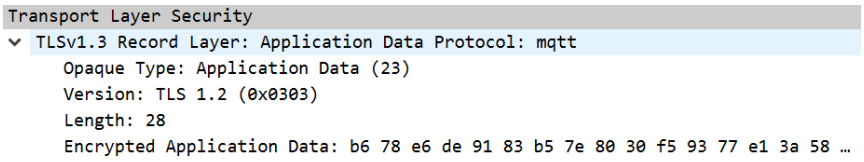

Figura 54 - MQTTS *Publish Message*

Mensagens de *ping*

De forma a manter a conexão ativa entre o *Subscriber* e o *Broker*, são enviados periodicamente dois pacotes, que são respetivamente: o *Ping Request* e o *Ping Response*.

Do *Subscriber* para o *Broker* existe um pacote trocado (*Ping Request*), sendo que este pacote tem o tamanho fixo de 24 *bytes*. A Figura 55 mostra o conteúdo do *payload* do pacote MQTTS.

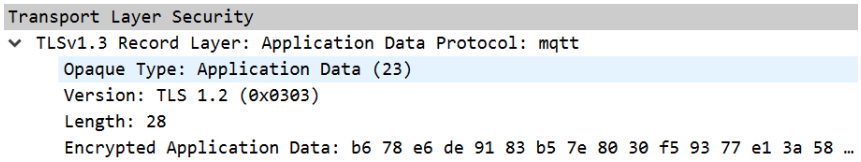

Figura 55 - MQTTS *Ping Request*

Do *Broker* para o *Subscriber* existe um pacote trocado (*Ping Response*) com o tamanho fixo de 24 *bytes*. A Figura 56 mostra o conteúdo do *payload* do pacote MQTTS.

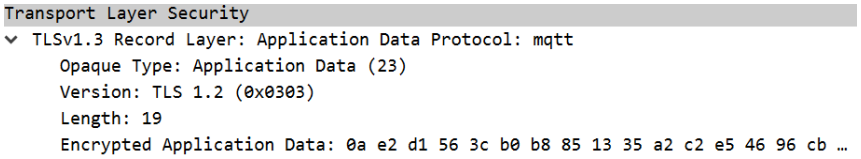

Figura 56 - MQTTS *Ping Response*

Cancelamento de uma subscrição de um tópico

Quando o *Subscriber* pretende terminar a subscrição de um tópico que se encontra no *Broker*, são trocados dois pacotes, que são respetivamente: o *Unsubscribe Request* e o *Unsubscribe Ack*.

Do *Subscriber* para o *Broker* existe um pacote trocado (*Unsubscribe Request*) com o tamanho mínimo de 27 *bytes*, o qual o tamanho pode aumentar de acordo com o tamanho do tópico. A Figura 57 mostra o conteúdo do *payload* do pacote MQTTS.

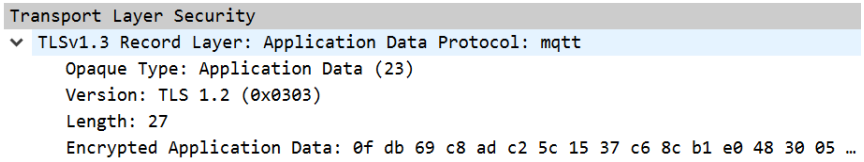

Figura 57 - MQTTS *Unsubscribe Request*

Do *Broker* para o *Subscriber* existe um pacote trocado (*Unsubscribe Ack*) com o tamanho mínimo de 26 *bytes*. A Figura 58 mostra o conteúdo do *payload* do pacote MQTTS.

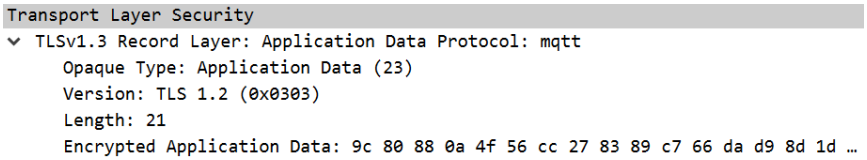

Figura 58 - MQTTS *Unsubscribe Ack*

Término de conexão

Do *Subscriber* para o *Broker* existe um pacote trocado (*Disconnect Req*) com o tamanho fixo de 22 *bytes*. A Figura 59 mostra o conteúdo do *payload* do pacote MQTTS.

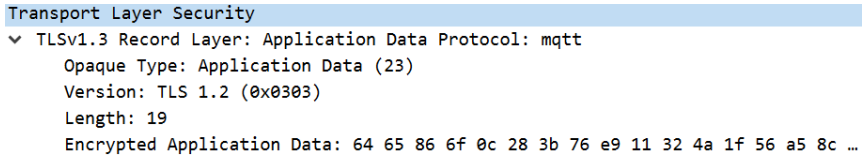

Figura 59 - MQTTS *Disconnect Req*

De seguida, é explicada a especificação do tipo de tráfego entre o *Subscriber* e o *Broker*. Após análise da exportação dos fluxos de tráfego de rede e da análise dos registos fluxos de tráfego de rede, caraterizou-se o tráfego de forma generalista e de forma a abranger todas as situações encontradas durante a exportação dos registos de fluxos.

É de realçar que a exportação dos fluxos pelo YAF, que ocorre num determinado intervalo de tempo (passado um minuto de inatividade ou três minutos de atividade), pode produzir fluxos que não contenham todo o fluxo de tráfego.

Na caracterização do tráfego, verifica-se que os registos de fluxos de tráfego de rede contêm as seguintes trocas de mensagens:

- O *Subscriber* inicia a sessão para o *Broker*;
- O *Subscriber* subscreve um tópico;
- O *Broker* envia as mensagens relativas a esse tópico para o *Subscriber*;
- para manter a sessão ativa, o *Subscriber* envia mensagens de *ping* e o *Broker* responde a essas mesmas mensagens de *ping*.

Na Tabela 28 encontram-se as especificações para o tráfego MQTTS de forma generalista. Estas resultam da combinação das mensagens identificadas anteriormente (a conexão, a subscrição de um tópico, a publicação de uma mensagem e as mensagens de *ping*).

Tabela 28 - Especificações generalistas para o tráfego MQTTS - *Subscriber* para o *Broker*

IE	Descrição
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 5
reverseNonEmptyPacketCount	>= 7
dataByteCount	== 0 ou >= 24
reverseDataByteCount	>= 24
firstNonEmptyPacketSize	== 0 ou >= 24
reverseFirstNonEmptyPacketSize	>= 24

Durante a análise dos registos dos fluxos de tráfego de rede exportados pelo protótipo observou-se que alguns fluxos de tráfego de rede não continham todo o tráfego. De acordo com esta situação, existiu a necessidade de criar especificações de forma a considerar estes tipos de fluxos de tráfego de rede.

Sendo assim, foram identificados fluxos de tráfego de rede que contêm somente as seguintes partes de tráfego:

- Conexão e Subscrição de um tópico;
- Publicação de uma mensagem;
- Mensagens de *ping*;
- Publicação de uma mensagem e mensagens de *ping*;
- Cancelamento de uma subscrição e o término de conexão.

Conexão e Subscrição de um tópico

A sequência de mensagens dos registos de fluxos de tráfego de rede com apenas início de sessão e a subscrição de um tópico é a seguinte:

- O *Subscriber* inicia a sessão para o *Broker*;
- O *Subscriber* subscreve um tópico presente no *Broker*.

Os IE relevantes para a identificação do tráfego MQTTS entre o *Subscriber* e o *Broker* para o estabelecimento da conexão e a subscrição de um tópico estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;

- reverseNonEmptyPacketCount - são enviados pelo menos quatro pacotes com *payload*;
- dataByteCount - é enviado um pacote com *payload*, com o tamanho mínimo de 28 *bytes*;
- reverseDataByteCount - é enviado um pacote com *payload*, com o tamanho mínimo de 24 *bytes*;
- firstNonEmptyPacketSize - pacote com o tamanho mínimo de 28 *bytes*;
- reverseFirstNonEmptyPacketSize - o pacote com o tamanho mínimo de 24 *bytes*.

Na Tabela 29 encontram-se as especificações para estes fluxos de tráfego de rede MQTTS.

Tabela 29 - MQTTS - Entre *Subscriber* e o *Broker*, conexão e subscrição de um tópico

IE	Descrição
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 3
reverseNonEmptyPacketCount	>= 4
dataByteCount	>= 28
reverseDataByteCount	>= 24
firstNonEmptyPacketSize	>= 28
reverseFirstNonEmptyPacketSize	>= 24

Publicação de uma mensagem

Na análise dos registos de fluxos de tráfego de rede exportados, verificou-se que existem fluxos de tráfego de rede que contêm apenas publicações de mensagens. Essas publicações têm origem no *Broker* e destino no *Subscriber*.

Os IE relevantes para a identificação do tráfego MQTTS para este tráfego estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - não são enviados pacotes com *payload*;
- reverseNonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;
- dataByteCount - não são enviados pacotes com *payload*;
- reverseDataByteCount - é enviado um pacote com *payload* com o tamanho mínimo de 27 *bytes*;
- firstNonEmptyPacketSize - não são enviados pacotes com *payload*;

- reverseFirstNonEmptyPacketSize - o pacote com o tamanho mínimo de 27 *bytes*.

Na Tabela 30 encontram-se as especificações para estes fluxos de tráfego de rede MQTTS.

Tabela 30 - MQTTS - Entre o *Subscriber* e o *Broker*, publicação de uma mensagem

IE	Valor
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e == 0
reverseNonEmptyPacketCount	== 1
dataByteCount	== 0
reverseDataByteCount	>= 27
firstNonEmptyPacketSize	== 0
reverseFirstNonEmptyPacketSize	>= 27

Mensagens de *ping*

Existem registos de fluxos de tráfego de rede que apenas contêm as mensagens de *ping* trocadas entre o *Subscriber* e o *Broker*, sendo que a sequência é a seguinte:

- *Ping Request*;
- *Ping Response*.

Os IE relevantes para a identificação do tráfego MQTTS entre o *Subscriber* e o *Broker* para a troca de mensagens de *ping*, estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado um pacote com *payload*;
- reverseNonEmptyPacketCount - é enviado um pacote com *payload*;
- dataByteCount - tamanho fixo de 24 *bytes*;
- reverseDataByteCount - tamanho fixo de 24 *bytes*;
- firstNonEmptyPacketSize - tamanho fixo de 24 *bytes*;
- reverseFirstNonEmptyPacketSize - tamanho fixo de 24 *bytes*.

Na Tabela 31 encontram-se as especificações para as mensagens de *ping*.

Tabela 31 - MQTTS - Entre o *Subscriber* e o *Broker*, mensagens de *ping*

IE	Valor
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e == 1
reverseNonEmptyPacketCount	== 1
dataByteCount	>= 24
reverseDataByteCount	>= 24
firstNonEmptyPacketSize	>= 24
reverseFirstNonEmptyPacketSize	>= 24

Publicação de uma mensagem e mensagens de *ping*

Existem registos de fluxos de tráfego de rede que apenas contêm a publicação de mensagens e a respetiva troca de mensagens de *ping* entre o *Subscriber* e o *Broker*. A sequência das mensagens é a seguinte:

- O *Broker* envia as mensagens relativas a esse tópico para o *Subscriber*;
- *Ping Request*;
- *Ping Response*.

Os IE relevantes para este tipo de fluxos de tráfego de rede entre o *Subscriber* e o *Broker* estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado pelo menos um pacote com *payload*;
- reverseNonEmptyPacketCount - são enviados pelo menos dois pacotes com *payload*;
- dataByteCount - tamanho mínimo de 22 *bytes*;
- reverseDataByteCount - tamanho mínimo de 24 *bytes*;
- firstNonEmptyPacketSize - tamanho mínimo de 22 *bytes*;
- reverseFirstNonEmptyPacketSize - tamanho mínimo de 24 *bytes*.

As especificações são a combinação das mensagens anteriormente caraterizadas, para o tráfego MQTTS. Estas especificações são descritas na Tabela 32.

Tabela 32 - MQTTS - Entre o *Subscriber* e o *Broker*, publicação de uma mensagem e mensagens de *ping*

IE	Descrição
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 1
reverseNonEmptyPacketCount	>= 2
dataByteCount	>= 22
reverseDataByteCount	>= 24
firstNonEmptyPacketSize	>= 22
reverseFirstNonEmptyPacketSize	>= 24

Cancelamento de uma subscrição e o término de conexão

Existem registos de fluxos de tráfego de rede que contêm apenas o cancelamento de uma subscrição e o término de conexão entre o *Subscriber* e o *Broker*. As sequências de mensagens são as seguintes:

- Cancelamento da subscrição de um tópico;
- Término da conexão.

Os IE relevantes para este tráfego entre o *Subscriber* e o *Broker* para o cancelamento de uma subscrição e término da conexão estão mencionados de seguida, onde também estão descritos cada um dos IE:

- nonEmptyPacketCount - é enviado pelos menos um pacote com *payload*;
- reverseNonEmptyPacketCount - não são enviados pacotes com *payload*;
- dataByteCount - tamanho mínimo de 22 *bytes*;
- reverseDataByteCount - não são enviados pacotes com *payload*;
- firstNonEmptyPacketSize - tamanho mínimo de 22 bytes;
- reverseFirstNonEmptyPacketSize - tamanho de dois *bytes* ou tem o tamanho mínimo de quatro *bytes*;

A Tabela 33 apresenta este tipo de tráfego MQTTS para o cancelamento de uma subscrição e o término da conexão.

Tabela 33 - MQTTS - Entre o *Subscriber* e o *Broker*, cancelamento de subscrição e o término da conexão

IE	Valor
protocolIdentifier	6
destinationTransportPort	8883
flowEndReason	!= *eof* ou != *idle* ou == *active* ou == vazio (“”)
unionTCPFlags	!= apr
nonEmptyPacketCount	!= *null* e >= 1
reverseNonEmptyPacketCount	>= 1
dataByteCount	>= 22
reverseDataByteCount	== 0
firstNonEmptyPacketSize	>= 22
reverseFirstNonEmptyPacketSize	== 0

De seguida, é descrito o plano de testes que será usado para validar a proposta do IDS para um sistema IoT.

5.3.Plano de testes

O plano de testes tem como base o protótipo descrito na Figura 22 (protótipo para testar a arquitetura proposta), bem como, as especificações de tráfego MQTT e MQTTS presentes no capitulo 5.2. Este plano de testes tem o objetivo de avaliar a arquitetura proposta no capítulo 4, onde se insere a atividade da aplicação IDS e a sua respetiva avaliação de performance, na deteção de registos de fluxos de tráfego de rede para os protocolos MQTT e MQTTS.

O plano de testes está divido em três fases. Na primeira fase serão efetuados testes ao funcionamento da aplicação IDS, na segunda fase serão efetuados testes sobre a capacidade de deteção de tráfego MQTT e na terceira e última fase serão efetuados testes ao protocolo MQTTS para determinar a capacidade da aplicação IDS para a deteção do tráfego MQTTS.

A atividade inicial do cenário de testes ocorre em duas fases. Em primeiro lugar, os dispositivos MQTT e MQTTS (*Publisher*, *Broker* e *Subscriber*) estão desligados. De forma a criar um sistema IoT, os dispositivos IoT são ligados e iniciam-se trocas de mensagens relativas à temperatura e à luminosidade do espaço em que os dispositivos estão inseridos. Na segunda fase, ativa-se a aplicação IDS e inicia-se a sonda IDS YAF, que captura o tráfego que trocado entre os dispositivos IoT. De seguida, a sonda IDS fez a agregação do tráfego capturado em registos de fluxos de tráfego de rede, sendo que, posteriormente, os mesmos registos de fluxos de tráfego de rede são exportados para o módulo central IDS, onde ficam armazenados para posterior análise.

No cenário de testes proposto, o funcionamento da aplicação IDS considera os seguintes pressupostos:

- A sonda IDS é executada em modo *daemon*, com o objetivo de recolher os pacotes de rede, de forma contínua, e de exportar os fluxos de tráfego de rede;
- A sonda IDS agrega e exporta os fluxos de tráfego de rede que constam na memória *cache* do módulo de agregação e exportação, com a periodicidade de um minuto, no caso de inatividade ou de três minutos em caso de atividade interrupta. Estes períodos podem ser reduzidos, de forma a chegarem a uma exportação quase em tempo real. A exportação recorre ao protocolo IPFIX sobre TLS;
- A sonda IDS utiliza a lista de IE presente na Tabela 15, com o objetivo de ter a maior variedade de características de tráfego que é capturado, melhorando assim futuras análises;
- O módulo central IDS recebe os fluxos de tráfego de rede IP, descodifica e armazena em ficheiros no formato JSON, através da utilização da estrutura apresentada na Figura 26;
- O módulo central IDS verifica se existem novos fluxos de tráfego de rede armazenados no módulo central e, caso existam, estes são analisados através das especificações que estão guardadas na base de dados de especificações. O resultado da análise irá catalogar os fluxos de tráfego de rede como normais ou anormais;
- Quando o módulo central IDS deteta um registo de fluxo anormal, envia e armazena uma mensagem de alerta de intrusão, recorrendo ao protocolo *syslog*;
- No cenário de testes irá ser gerado tráfego anormal, usando para isso o nó atacante e em algumas situações os dispositivos MQTT e MQTTS. O nó atacante cria ataques de *net scan* e do tipo DoS e DDoS para os serviços MQTT e MQTTS, para além de, conseguir injetar informações erradas ao MQTT e ao MQTTS, fazendo com que estes criem ataques de publicações e subscrições inválidas.

Os pressupostos enumerados anteriormente têm o objetivo de ajudar na execução dos testes futuramente tratados, com o propósito de validar a performance das especificações que serão utilizadas para a deteção de intrusões.

5.3.1. Teste de funcionamento da aplicação IDS

O teste ao funcionamento da aplicação IDS tem o objetivo de validar o funcionamento da aplicação IDS. A aplicação foi desenvolvida com recurso à linguagem de programação *Python* e tem a finalidade de executar a leitura e análise do tráfego de fluxos de rede IP que estão armazenados no módulo central IDS.

No teste são apenas analisados fluxos de tráfego de rede IP armazenados no módulo central IDS. Os fluxos de tráfego de rede têm origem no tráfego que foi gerado pelos dispositivos MQTT e MQTTS, tendo sido armazenados pelo módulo central IDS após a exportação através da sonda IDS.

A aplicação IDS irá executar a leitura e análise em duas etapas. Na primeira etapa será executada a leitura das especificações MQTT e MQTTS de tráfego normal. Estas especificações encontram-se numa base de dados existente numa diretoria dedicada para esse efeito. Na segunda etapa, caso existem novos fluxos de tráfego de rede, serão comparados com as especificações para tráfego normal. Caso os fluxos de tráfego de rede sejam classificados como anormais será então criado um alerta de intrusão recorrendo ao protocolo *syslog*. O produto desta análise resulta num relatório que classifica cada registo de fluxo de tráfego de rede como normal ou anormal.

Deste modo, o resultado esperado da aplicação IDS é conseguido, já que se verifica que a mesma tem a capacidade de classificar os registos de fluxos de tráfego IP como normais ou anormais.

5.3.2. Teste de deteção para o tráfego MQTT

O teste à deteção do tráfego MQTT tem o objetivo de avaliar o funcionamento da análise dos registos de fluxos de tráfego de rede IP, com o intuito de se obter um relatório da performance das especificações para o tráfego do protocolo MQTT, mencionadas nos capítulos 5.2.1 e 5.2.2.

Este teste pretende validar duas situações:

1. Analisar os registos de fluxos de tráfego de rede MQTT normais;
2. Analisar os registos de fluxos de tráfego de rede MQTT normais e anormais.

Analisar os registos de fluxos de tráfego de rede MQTT normais

Este teste está dividido em quatro etapas. Na primeira etapa são ativados todos os dispositivos MQTT com o objetivo de criar tráfego entre os vários dispositivos MQTT (*Publisher*, *Broker* e *Subscriber*).

Na segunda etapa, a sonda YAF é ativada, com o intuito de capturar os pacotes de rede que são trocados entre os dispositivos MQTT, que se encontram quer na rede interna quer na rede externa.

Na terceira etapa, no módulo central IDS, o *super_mediator* carrega as especificações, mencionadas nos capítulos 5.2.1 e 5.2.2, para que se consiga efetuar as descodificações e o armazenamento dos registo de fluxos de tráfego de rede IP que são exportados pela sonda YAF através do IPFIX.

Por último, na quarta etapa, tem início a atividade da aplicação IDS, onde se verifica a existência de novos fluxos de tráfego armazenados no módulo central IDS, sendo que estes novos fluxos estão no formato JSON. Caso existam novos fluxos armazenados é efetuada uma análise dos mesmos com as especificações de tráfego MQTT mencionadas capítulos 5.2.1 e 5.2.2.

Após estas quatro etapas é elaborado um relatório que contém a classificação de cada registo de fluxo de tráfego, classificação essa que pode ser normal ou anormal.

Analisar os registos de fluxos de tráfego de rede MQTT normais e anormais

O segundo teste, à semelhança do primeiro, está também dividido em quatro etapas. Assim sendo, na primeira etapa são ativados todos os dispositivos MQTT com o objetivo de criar tráfego entre os vários dipositivos MQTT (*Publisher*, *Broker* e *Subscriber*). Após ativação dos dispositivos MQTT, ativa-se o nó atacante, descrito no capítulo 5.1, com o objetivo de criar tráfego IoT anormal e malicioso, recorrendo para isso aos dispositivos MQTT.

Na segunda etapa, tem início a execução da sonda YAF, com o intuito de fazer captura dos pacotes de redes que fluem entre os dispositivos MQTT, que se encontram quer na rede interna quer na rede externa.

Na terceira etapa, no módulo central IDS, o *super_mediator* carrega as especificações, mencionadas nos capítulos 5.2.1 e 5.2.2, para que se consiga efetuar as descodificações e o

armazenamento dos registo de fluxos de tráfego de rede IP que são exportados pela sonda YAF através do IPFIX.

Por último, na quarta etapa, tem início a atividade da aplicação IDS, onde se verifica a existência de novos fluxos de tráfego de rede armazenados no módulo central IDS, sendo que estes novos fluxos estão no formato JSON. Caso existam novos fluxos de tráfego de rede armazenados, é efetuada uma análise dos mesmos com as especificações de tráfego MQTT mencionadas nos capítulos 5.2.1 e 5.2.2.

Após estas quatro etapas também é elaborado um relatório que contém a classificação de cada registo de fluxo de tráfego de rede, classificação essa que pode ser normal ou anormal.

Em ambos os testes pretende-se que as especificações descritas nos capítulos 5.2.1 e 5.2.2 classifiquem corretamente os registos de fluxos tráfego de rede normais e anormais de MQTT. No entanto, acontece situações em que o nó atacante produz tráfego normal MQTT e também tráfego anormal MQTT gerado por dispositivos MQTT não fidedignos.

Em paralelo à geração de tráfego MQTT normal e anormal nestes dois testes, é importante realçar que também existe tráfego adicional a ser gerado, já que na rede interna existem dois serviços adicionais (o serviço de DNS e o serviço de NTP) que também produzem tráfego.

5.3.3. Teste de deteção para o tráfego MQTTS

Por último, o teste à deteção ao tráfego MQTTS tem o objetivo de avaliar o funcionamento da análise dos registos de fluxos de tráfego de rede IP, com o intuito de se obter um relatório da performance das especificações para o tráfego do protocolo MQTTS, mencionadas nos capítulos 5.2.3 e 5.2.4.

Este teste pretende validar duas situações:

1. Analisar dos registos de fluxos MQTTS normais;
2. Analisar os registos de fluxos MQTTS normais e anormais.

Analisar dos registos de fluxos de tráfego de rede MQTTS normais

Este teste está dividido em quatro etapas. Na primeira etapa são ativados todos os dispositivos MQTTS. É de notar que no MQTTS é estabelecido previamente uma sessão TLS entre os vários dispositivos MQTTS, sendo que o *Broker* tem a função de ser o servidor

para os restantes dispositivos MQTTS (*Publisher* e *Subscriber*), este processo permite garantir a privacidade do tráfego entre os dispositivos MQTTS.

Na segunda etapa, a sonda YAF é ativada, com o intuito de capturar os pacotes de rede que são trocados entre os dispositivos MQTT, que se encontram quer na rede interna quer na rede externa.

Na terceira etapa, no módulo central IDS, o *super_mediator* carrega as especificações, mencionadas nos capítulos 5.2.3 e 5.2.4, para que se consiga efetuar as descodificações e o armazenamento dos registos de fluxos de tráfego de rede IP que são exportados pela sonda YAF através do IPFIX.

Por último, na quarta etapa, tem início a atividade da aplicação IDS, onde se verifica a existência de registos de fluxos de tráfego de rede armazenados no módulo central IDS, sendo que estes novos fluxos de tráfego de rede estão no formato JSON. Caso existem novos fluxos de tráfego de rede armazenados é efetuada uma análise dos mesmos com as especificações de tráfego MQTTS mencionada nos capítulos 5.2.3 e 5.2.4.

Após este teste é obtido um relatório na qual que classifica cada registo de fluxo de tráfego de rede como normal ou anormal.

Analisar dos registos de fluxos de tráfego de rede MQTTS normais ou anormais.

O segundo teste, à semelhança do primeiro, está também dividido em quatro etapas. Assim sendo, na primeira etapa são ativados todos os dispositivos MQTTS com o objetivo de criar tráfego entre os vários dispositivos MQTTS (*Publisher*, *Subscriber* e *Broker*). No caso do MQTTS são estabelecidas previamente sessões TLS entres os vários dispositivos MQTTS, sendo que *Broker* tem o papel de ser o servidor, em que o nome do estabelecimento da sessão é designado por *handshake* TLS. Depois do estabelecimento da sessão TLS, tem o início a troca de tráfego MQTTS entre os vários dispositivos MQTTS.

Após ativação dos dispositivos MQTTS, ativa-se o nó atacante, descrito no capítulo 5.1, como o objetivo de criar tráfego IoT anormal e malicioso, recorrendo para isso aos dispositivos MQTTS, produzindo assim também algum tráfego anormal. Todo o tráfego TLS está cifrado de forma a ter a garantia da confidencialidade dos dados dos clientes.

Na segunda etapa, tem início a execução da sonda YAF, com o intuito de fazer a captura dos pacotes de rede que fluem entres os dispositivos MQTTS, que se encontram quer na rede interna quer na rede externa.

Na terceira etapa, no módulo central IDS o *super_mediator* carrega as especificações mencionadas nos capítulos 5.2.3 e 5.2.4, para que se consiga efetuar as descodificações e o armazenamento dos registos de fluxos de tráfego de rede IP que são exportados pela sonda YAF através do IPFIX.

Por último, na quarta etapa, terá inico a atividade da aplicação IDS, onde se verifica a existência de novos fluxos de tráfego de rede armazenados no módulo central IDS sendo que esses novos fluxos estão no formato JSON. Caso existam novos fluxos de tráfego de rede armazenados, é efetuada uma análise dos menos com as especificações de tráfego MQTTS mencionadas nos capítulos 5.2.3 e 5.2.4.

Após estas quatro etapas também é elaborado um relatório que contém a classificação de cada registo de fluxo de tráfego, classificação essa que pode ser normal ou anormal.

Em ambos os testes pretende-se que as especificações descritas nos capítulos 5.2.3 e 5.2.4 classifiquem corretamente os registos de fluxos tráfego de rede normais e anormais de MQTTS. No entanto, acontece situações em que o nó atacante produz tráfego normal MQTTS e também tráfego anormal MQTTS gerado por dispositivos MQTTS não fidedignos.

Em paralelo à geração de tráfego MQTTS normal e anormal nestes dois testes, é importante realçar que também existe tráfego adicional a ser gerado, já que na rede interna existem dois serviços adicionais (o serviço de DNS e o serviço de NTP) que também produzem tráfego.

5.4.Síntese

Neste capítulo, inicialmente, apresentou-se a arquitetura proposta e efetuou-se a descrição da rede interna e da rede externa, com as correspondentes caraterizações dos diferentes elementos.

Seguidamente, fez-se a caracterização da sonda. A mesma localiza-se na rede interna e recolhe o tráfego da rede interna para a rede externa e vice-versa. Também se efetuou a

caraterização do YAF, que é utilizado para exportar os registos de fluxos de tráfego de rede para o módulo central IDS.

A caraterização do módulo central IDS foi também efetuada. O mesmo localiza-se na rede externa e recebe os fluxos de tráfego de rede exportados pela sonda, para além disso, armazena e analisa os registos de fluxos de tráfego de rede.

Um dos elementos do módulo central IDS contém um *script* em *Python* que tem a finalidade de verificar a existência de novos registos de fluxos de tráfego de rede no módulo central IDS e, tendo as especificações MQTT e MQTTS pré-carregadas, classifica cada registo de fluxo de tráfego como normal ou anormal.

Neste capítulo também se descrevem as especificações do tráfego MQTT e MQTTS, com a caraterização das mensagens MQTT e MQTTS (*Subscriber*, *Broker* e o *Publisher*). Estas especificações têm em conta o tempo de exportação de tráfego de fluxos de tráfego de rede feita pelo YAF, já que os registos de fluxos de tráfego de rede são exportados passado um minuto, caso não exista atividade ou exportados passados três minutos, caso exista atividade. Estas exportações podem originar registos de fluxos de tráfego de rede incompletos, isto é, fluxos de tráfego de rede apenas com partes das mensagens, por exemplo, apenas mensagens de *ping*.

Após a caraterização do tráfego MQTT e MQTTS, apresentou-se um plano de testes, de forma a validar as especificações apresentadas anteriormente e a classificar corretamente os registos de fluxos do tráfego de rede normal e anormal.

No entanto, nem todos os IE são selecionados para a criação de especificações no tráfego, sendo que somente são escolhidos os IE necessários à caraterização do tráfego MQTT e MQTTS.

6. Resultados e avaliação do plano de testes

Este capítulo apresenta e discute os resultados dos testes anteriormente apresentados no capítulo 5.3. Como mencionado no capítulo anterior, o plano de testes tem o objetivo de avaliar a eficiência e eficácia da arquitetura proposta, no capítulo 4. O plano de testes foca-se no campo da atividade da aplicação IDS e à sua respetiva capacidade de deteção de intrusões. Para tal, recorrem-se às especificações definidas para o tráfego normal MQTT e MQTTS. Essas especificações foram apresentadas nos capítulos 5.2.1 e 5.2.2 para o tráfego MQTT e nos capítulos 5.2.3 e 5.2.4 para o tráfego MQTTS.

Através da produção de fluxos de tráfego de rede entre os vários dispositivos (*Publisher*, *Broker* e *Subscriber*) MQTT e MQTTS e o nó atacante foi possível criar um *dataset* composto por fluxos de tráfego de rede normal e anormal, para além de fluxos de tráfego de rede dos serviços DNS e NTP.

O tráfego normal foi capturado através do protótipo, de forma consecutivas e sem interrupções, durante sete dias de operação, sem a geração de tráfego anormal. O facto de não ter sido gerado tráfego anormal permite garantir que os fluxos de tráfego de rede gerados nesta fase sejam relativos a fluxos de tráfego de rede normais.

O tráfego anormal foi depois gerado pelo nó atacante do protótipo, recorrendo a ataques *net scan*, através das ferramentas *hping* e *nmap*, assim como a ataques de rajada.

Este capítulo está divido em quatro partes. Na primeira parte serão apresentados os resultados da validação do funcionamento da aplicação IDS. Depois, na segunda parte, serão expostos os resultados dos testes para o tráfego MQTT. De seguida, na terceira parte, serão descritos os resultados dos testes para o tráfego MQTTS. E, por último, na quarta parte, serão discutidos os resultados obtidos, tanto para o tráfego MQTT como para o tráfego MQTTS.

6.1. Teste de funcionamento da aplicação IDS

O resultado deste teste tem como objetivo a validação da aplicação IDS para os registos de fluxos de tráfego de rede referentes aos protocolos MQTT e MQTTS.

No desenvolvimento deste teste foram utilizados e selecionados registos de fluxos de tráfego de rede armazenados no módulo central IDS.

Estes registos de fluxos de tráfego de rede são o resultado de trocas de tráfego entre os vários dispositivos MQTT e MQTTS (*Publisher*, *Broker* e *Subscriber*). Sendo, posteriormente, capturados pela sonda YAF e exportados através do protocolo IPFIX para o módulo central IDS, onde são depois armazenados.

Após o armazenamento dos registos de fluxos de tráfego de rede no módulo central IDS, a aplicação IDS, também integrada no módulo central IDS, efetua a verificação dos novos registos de fluxos de tráfego de rede e a sua respetiva análise, tendo em conta os protocolos MQTT e MQTTS. A análise é efetuada através da comparação entre estes registos de fluxos de tráfego e as especificações de tráfego normal, que estão descritas nos capítulos 5.2.1 e 5.2.2 para o tráfego MQTT e nos capítulos 5.2.3 e 5.2.4 para o tráfego MQTTS.

O resultado desta análise produz um relatório que classifica qualitativamente e quantitativamente os fluxos de tráfego de rede MQTT e MQTTS como normais ou como anormais. Assim, de forma sucinta, o relatório especifica o número de registos de fluxos de tráfego de rede MQTT que são classificados como normais e o número de registos de fluxos de tráfego de rede MQTT que são classificados como anormais. O mesmo acontece para os registos de fluxos de tráfego de rede MQTTS, o relatório especifica o número de registos de fluxos de tráfego de rede MQTTS que são classificados como normais e o número de registos de fluxos de tráfego de rede MQTTS que são classificados como anormais.

Adicionalmente, o relatório também reporta o número de fluxos de tráfego de rede que não são classificados como tráfego TCP.

A Figura 60 demonstra um excerto da saída da execução da aplicação IDS, onde se consegue visualizar a forma de como são classificados os registos de fluxos de tráfego de rede armazenados no módulo central IDS.

```
$ python3 main.py flows
(…)
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:37:46.592',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:38:55.167',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:38:46.678',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:39:55.170',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:39:46.763',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:40:55.179',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:40:46.836',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:41:55.192',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:41:46.915',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:42:55.215',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:43:25.225',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:42:46.995',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:40:30.439',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:44:25.230',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:43:47.076',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:45:25.231',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:44:47.153',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:46:25.238',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:45:47.236',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:47:25.251',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:47:46.409',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:48:25.254',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:48:46.498',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:49:25.274',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:49:46.577',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:50:25.288',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:50:46.655',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:51:25.302',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:51:46.734',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:52:25.310',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:52:46.817',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:53:25.323',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:53:46.898',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:54:25.336',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:54:55.347',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:54:46.977',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:55:55.354',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:55:47.058',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:56:55.355',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:56:47.134',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:57:55.370',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:58:55.374',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:56:30.444',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:57:47.216',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:59:55.379',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 10:59:46.376',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:00:55.381',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:00:46.458',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:01:55.389',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 10:59:30.459',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:01:46.537',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:02:55.397',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:02:46.615',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:03:55.401',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:03:46.694',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:04:55.405',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:04:46.775',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:02:30.462',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:05:55.402',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:05:46.858',
MQTT-NORMAL - {'flowStartMilliseconds': '2019-06-15 11:06:55.403',
MQTTS-ATTACK - {'flowStartMilliseconds': '2019-06-15 11:06:46.938',
COUNTERS -
6031 MQTT attacks ----------
82 MQTT invalid flow (ATTACK) ----------
19140 MQTT normal flows ----------
9169 MQTTS attacks ----------
126 MQTTS invalid flow (ATTACK) ----------
29 MQTTS normal flows ----------
0 MQTT or MQTTS invalid flow (ATTACK) ----------
0 No TCP Flows
```

Figura 60 - Resultado da execução da aplicação IDS

A Figura 61 ilustra um excerto de um relatório obtido pela execução da aplicação IDS, onde se visualiza o número total de fluxos de tráfego de rede classificado como normais e anormais para os protocolos MQTT e MQTTS.

```
$grep COUNTERS output_FINAL_attack.txt
COUNTERS -
6031 MQTT attacks ----------
82 MQTT invalid flow (ATTACK) ----------
19140 MQTT normal flows ----------
9169 MQTTS attacks ----------
126 MQTTS invalid flow (ATTACK) ----------
29 MQTTS normal flows ----------
0 MQTT or MQTTS invalid flow (ATTACK) ----------
0 No TCP Flows
```

Figura 61 - Saída para o monitor da aplicação IDS - resumo

6.2.Testes e resultados para o protocolo MQTT

Este teste tem como principal objetivo avaliar a atividade da aplicação IDS, aquando da análise dos registos de fluxos de tráfego de rede, com o intuito de observar a performance das especificações para o tráfego MQTT, descritas nos capítulos 5.2.1 e 5.2.2.

Estes registos de fluxos de tráfego de rede são o resultado de trocas de tráfego entres os vários dispositivos MQTT (*Publisher*, *Broker* e *Subscriber*), assim como, do nó atacante, sendo posteriormente capturados pela sonda YAF e exportados através do IPFIX para o módulo central IDS, onde são armazenados.

Tal como descrito no capítulo 5.3.2, o plano de testes está divido em duas situações, que, por sua vez, estão subdividas em duas fases, como se descreve de seguida:

1. Analisar os registos de fluxos de tráfego de rede MQTT normal (situação 1):
    a. Analisar a coleção de registos de fluxos de tráfego de rede MQTT normal;
    b. Analisar o *dataset* de registos de fluxos de tráfego de rede MQTT normal gerado pelo protótipo durante sete dias consecutivos, sem adição de tráfego anormal.
2. Analisar os registos de fluxos de tráfego de rede MQTT com anomalias (situação 2):
    c. Analisar a coleção de registos de fluxos de tráfego de rede MQTT anormal;
    d. Analisar o *dataset* de registos de fluxos de tráfego de rede MQTT anormal gerado pelo protótipo durante sete dias consecutivos.

Os resultados obtidos através dos testes, considerando estas duas situações, estão descritos nos subcapítulos seguintes, nos quais são mencionados os seguintes pontos:

- O método utilizado nos fluxos de tráfego de rede (*Publisher* ou *Subscriber*);
- O número de registos de fluxos de tráfego de rede que compõem o *dataset* que foi utlizado;
- O número de registo de fluxos de tráfego de rede referente ao protocolo MQTT;
- O número de registos de fluxos de tráfego de rede que foram classificados como normais (RFN);
- O número de registos de fluxos de tráfego que foram classificados como anormais (RFA);
- Taxa de falsos positivos (FP);
- Taxa de falsos negativos (FN);
- Taxa de verdadeiros positivos (TP);
- Taxa de deteção (TD).

6.2.1. Teste e resultados aos fluxos de tráfego normal (situação 1)

Na fase a) da situação 1 foi selecionado apenas um conjunto de registos de fluxos de tráfego de rede contendo tráfego normal MQTT do tipo *Publisher* e do tipo *Subscriber*, com o propósito de validar um menor número de registos de fluxos de tráfego de rede tendo em conta as especificações definidas.

Na fase b) da situação 1 foram utilizados todos os registos de fluxos de tráfego de rede normais, sem a separação do tráfego do tipo *Publisher* ou do tipo *Subscriber*. Estes registos de fluxos de tráfego de rede estão presentes no *dataset*, que contém o tráfego MQTT gerado pelo protótipo durante sete dias consecutivos sem a introdução de tráfego anormal (nó atacante), de forma a validar o tráfego MQTT normal e com maior diversidade.

Os resultados das duas fases estão apresentados nas tabelas Tabela 34 e Tabela 35, sendo que nas mesmas está identificado o tipo de tráfego MQTT *Publisher* e *Subscriber* que correspondem aos registos de fluxos de tráfego de rede analisados, o número de registos de fluxos de tráfego de rede que estão presentes no *dataset*, o número de registo de fluxos de tráfego de rede MQTT que estão presentes no *dataset*, o número de registos de fluxos de tráfego de rede que foram classificados como normais (RFN), o número de registos de fluxos de tráfego de rede que foram classificados como anormais (RFA), a taxa de deteção (TD)

para o teste realizado, a taxa de falsos positivos (FP), assim como, a taxa falsos negativos (FN).

A Tabela 34 apresenta o resultado do teste para os registos de tráfego normal MQTT para a fase a).

Tabela 34 - Resultados da deteção de fluxos de tráfego de rede normais MQTT por mensagem

Tipo de mensagem MQTT	Número de fluxos do *dataset*	Número de fluxos MQTT	RFN	RFA	FP	FN	TD
Publisher	100	100	100	0	0	0	100%
Subscriber	100	100	100	0	0	0	100%

A Tabela 35 apresenta o resultado do teste para os registos de tráfego normal MQTT para a fase b).

Tabela 35 - Resultado da deteção de fluxos de tráfego de rede normais MQTT

Tipo de mensagem MQTT	Número de fluxos do *dataset*	Número de fluxos MQTT	RFN	RFA	FP	FN	TD
Publisher/ Subscriber	19147	19118	19118	0	0	0	100%

Em ambas as fases da situação 1, o volume de tráfego produzido originou uma taxa de deteção (TD) de 100% e uma taxa de falsos positivos e de falsos negativos de 0%. Estes resultados eram os pretendidos para a classificação do tráfego MQTT normal e demonstram que as especificações desenvolvidas estão adequadas para este tipo de tráfego.

6.2.2. Teste e resultados aos fluxos de tráfego de rede normais e anormais (situação 2)

Na fase c) da situação 2, os resultados do teste estão estruturados pelo tipo de anomalias, que são as seguintes: *net scan* (*nmap* e *hping*), inundação de pedidos válidos e inválidos MQTT, DoS e DDoS. Estes tipos de anomalias foram escolhidos com o propósito de conterem um menor volume de exemplares de registos de fluxos de tráfego de rede para a validação das especificações.

Na fase d) da situação 2, foram utilizados todos os registos de fluxos de tráfego de rede MQTT, quer de tráfego normal quer de tráfego anormal (*net scan*, inundação de pedidos validos e inválidos MQTT, DoS e DDoS). Estes registos de fluxos de tráfego de rede estão

presentes no *dataset*, que contém o tráfego MQTT gerado pelo protótipo durante sete dias consecutivos com a adição de tráfego anormal, permitindo assim que se efetue tanto a validação do tráfego MQTT normal como do anormal.

Os resultados das duas fases estão apresentados nas tabelas Tabela 36 e Tabela 37, sendo que nas mesmas está identificado o tipo de tráfego MQTT anormal que corresponde ao registos de fluxos de tráfego de rede analisados, o número de registos de fluxos de tráfego de rede que estão presentes no *dataset*, o número de registo de fluxos de tráfego MQTT que estão presentes no *dataset*, o número de fluxos de tráfego que foram classificados como normais (RFN), o número de registos de fluxos de tráfego de rede que foram classificados como anormais (RFA), a taxa de deteção (TD) para o teste realizado, a taxa de falsos positivos (FP) e a taxa falsos negativos (FN).

A Tabela 34 apresenta o resultado do teste para os registos de tráfego de rede anormal MQTT para a fase c).

Tabela 36 - Resultados da deteção de fluxos de tráfego de rede MQTT anormais por tipo de mensagem

Tipo de mensagem MQTT	Número de fluxos do *dataset*	Número de fluxos MQTT	RFN	RFA	FP	FN	TD
net scan	100	100	0	100	0	0	100%
Publisher / *Subscriber* tópico inválido	18	18	10	8	0	0	44%
Publisher Inundação pedidos validos e inválidos MQTT	100	100	0	100	0	0	100%
Subscriber Inundação pedidos válidos e inválidos MQTT	100	100	0	100	0	0	100%

Na Tabela 36, no tipo de mensagem MQTT "*Publisher* / *Subscriber* tópico inválido" não existe um grande volume de registos de fluxos de tráfego de rede anormal entre o *Publisher* e o *Broker* inválidos. Apenas são observados 18 registos de fluxos de tráfego de rede inválidos entre o *Subscriber* e o *Broker*.

A Tabela 37 apresenta o resultado do teste para os registos de tráfego anormal MQTT para a fase d).

Tabela 37 - Resultado da deteção de fluxos tráfego de rede anormais MQTT

Dataset	Número de fluxos do *dataset*	Número de fluxos MQTT	RFN	RFA	FP	FN	TD
Dataset	34577	25253	19140	6113	0	0	100%

De seguida, estão expostas as análises dos resultados obtidos para o tráfego MQTT normal e para o tráfego MQTT com anomalias.

Para a situação 2 na fase c), o volume de tráfego produzido originou uma taxa de deteção de tráfego anormal de 100% e uma taxa de falsos positivos e de falsos negativos de 0%. Estes resultados eram os pretendidos para a classificação do tráfego MQTT normal e desmontaram que as especificações desenvolvidas estão adequadas para este tipo de tráfego.

Relativamente ao tipo de mensagem MQTT "*Publisher* / *Subscriber* tópico inválido", salienta-se que existiram poucos registos de fluxos de tráfego de rede MQTT gerados pelo protótipo. Sendo que não existem registos de fluxos de tráfego de rede MQTT entre *Publisher* e o *Broker* válidos e apenas são observados registos de fluxos de tráfego de rede MQTT entre o *Subscriber* e *Broker* com anomalias. Assim, os resultados obtidos são de 44% de taxa de deteção, com 0% de falsos positivos e de falsos negativos.

Para a situação 2 na fase d), para o volume de tráfego existente no *dataset*, o teste resultou numa taxa de deteção de tráfego anormal de 100% e uma taxa de falsos positivos e de falsos negativos de 0%.

Em suma, para ambas as situações, verifica-se que os resultados dos testes definidos no capítulo 5.3.2. e que utilizam as especificações criadas nos capítulos 5.2.1 e 5.2.2, permitem demostrar que as especificações apresentam um grau de exatidão satisfatório para o protótipo e para as anomalias testadas.

6.3.Teste e resultados para o protocolo MQTTS

Este teste tem como principal objetivo avaliar a atividade da aplicação IDS, aquando da análise dos registos de fluxos de tráfego de rede, com o intuito de observar a performance das especificações para o tráfego MQTTS, descritas nos capítulos 5.2.3 e 5.2.4.

Estes registos de fluxos de tráfego de rede são o resultado de trocas de tráfego entres os vários dispositivos MQTTS (*Publisher*, *Broker* e *Subscriber*), assim como, do nó atacante, sendo posteriormente capturados pela sonda YAF e exportados através do IPFIX para o módulo central IDS, onde são armazenados.

Tal como descrito no capítulo 5.3.3, o plano de testes está divido em duas situações, que por sua vez estão subdividas em duas fases como se descreve de seguida:

1. Analisar os registos de fluxos de tráfego de rede MQTTS normal (situação 1):
    a. Analisar a coleção de registos de fluxos de tráfego de rede MQTTS normal;
    b. Analisar o *dataset* de registos de fluxos de tráfego de rede MQTTS normal gerado pelo protótipo durante sete dias consecutivos, sem adição de tráfego anormal.
2. Analisar os registos de fluxos de tráfego de rede MQTTS com anomalias (situação 2):
    c. Analisar a coleção de registos de fluxos de tráfego de rede MQTTS anormal;
    d. Analisar o *dataset* de registos de fluxos de tráfego de rede MQTTS anormal gerado pelo protótipo durante sete dias consecutivos.

Os resultados obtidos através dos testes, considerando estas duas situações, estão descritos nos subcapítulos seguintes, nos quais são mencionados os seguintes pontos:

- O método utilizado nos fluxos de tráfego de rede (*Publisher* ou *Subscriber*);
- O número de registos de fluxos de tráfego de rede que compõem o *dataset* que foi utlizado;
- O número de registo de fluxos de tráfego de rede referente ao protocolo MQTTS;
- O número de registos de fluxos de tráfego de rede que foram classificados como normais (RFN);
- O número de registos de fluxos de tráfego de rede que foram classificados como anormais (RFA);
- Taxa de falsos positivos (FP);
- Taxa de falsos negativos (FN);
- Taxa de verdadeiros positivos (TP);
- Taxa de deteção (TD).

6.3.1. Teste e resultados aos fluxos de tráfego de rede normal (situação 1)

Na fase a) da situação 1 foi selecionado apenas um conjunto de registos de fluxos de tráfego de rede contendo tráfego normal MQTTS do tipo *Publisher* e do tipo *Subscriber*, com o propósito de validar um menor número de registos de fluxos de tráfego de rede tendo em conta as especificações definidas.

Na fase b) da situação 1 foram utilizados todos os registos de fluxos de tráfego de rede normais, sem a separação do tráfego do tipo *Publisher* ou do tipo *Subscriber*. Estes registos

de fluxos de tráfego de rede estão presentes no *dataset*, que contém o tráfego MQTTS gerado pelo protótipo durante sete dias consecutivos sem a introdução de tráfego anormal (nó atacante), de forma a validar o tráfego MQTTS normal e com maior diversidade.

Os resultados das duas fases estão apresentados em tabelas Tabela 38 e Tabela 39, sendo que nas mesmas está identificado o tipo de tráfego MQTTS *Publisher* e *Subscriber* que correspondem aos registos de fluxos de tráfego de rede analisados, o número de registos de fluxos de tráfego de rede que estão presentes no *dataset*, o número de registo de fluxos de tráfego de rede MQTTS que estão presentes no *dataset*, o número de registos de fluxos de tráfego de rede que foram classificados como normais (RFN), o número de registos de fluxos de tráfego de rede que foram classificados como anormais (RFA), a taxa de deteção (TD) para o teste realizado, a taxa de falsos positivos (FP), assim como, e a taxa falsos negativos (FN).

A Tabela 38 apresenta o resultado do teste para os registos de tráfego de rede normal MQTTS para a fase a).

Tabela 38 - Resultados da deteção de fluxos tráfego de rede normais MQTTS por mensagem

Tipo de mensagem MQTTS	Número de fluxos do *dataset*	Número de fluxos MQTTS	RFN	RFA	FP	FN	TD
Publisher	10	10	10	0	0	0	100%
Subscriber	19	19	19	0	0	0	100%

É de notar que o protótipo não gerou um volume elevado de registos de tráfego de rede MQTTS normal.

A Tabela 39 apresenta o resultado do teste para os registos de tráfego de rede normal MQTTS para a fase b).

Tabela 39 - Resultado da deteção de fluxos tráfego de rede normais MQTTS

Tipo de mensagem MQTTS	Número de fluxos do *dataset*	Número de fluxos MQTTS	RFN	RFA	FP	FN	TD
Publisher/ Subscriber	19147	29	29	0	0	0	100%

Em ambas as fases da situação 1, o volume de tráfego produzido originou uma taxa de deteção (TD) de 100% e uma taxa de falsos positivos e de falsos negativos de 0%. Estes

resultados eram os pretendidos para a classificação do tráfego MQTTS normal e demonstram que as especificações desenvolvidas estão adequados para este tipo de tráfego.

É de notar que o protótipo não gerou um volume elevado de tráfego MQTTS sem anomalias.

No entanto, se o protótipo se prolongar no tempo é esperado que o tráfego continue a ser classificado de forma correta, visto que a continuação de geração de tráfego e o seu consequente aumento de volume de registos de fluxos de tráfego de rede capturados não influenciam a classificação dos registos de tráfego.

6.3.2. Teste e resultados aos fluxos de tráfego de rede normais e anormais (situação 2)

Na fase c) da situação 2 os resultados do teste estão estruturados pelo tipo de anomalias, que são as seguintes: *net scan* (*nmap* e *hping*), inundação de pedidos válidos e inválidos MQTTS, DoS e DDoS. Estes tipos de anomalias foram escolhidos com o propósito de conterem um menor volume de exemplares de registos de fluxos de tráfego de rede para a validação das especificações.

Na fase d) da situação 2, foram utilizados todos os registos de fluxos de tráfego de rede MQTTS, quer de tráfego normal quer de tráfego anormal (*net scan*, inundação de pedidos validos e inválidos MQTTS, DoS e DDoS). Estes registos de fluxos de tráfego de rede estão presentes no *dataset*, que contém o tráfego MQTTS gerado pelo protótipo durante sete dias consecutivos com a adição de tráfego anormal, permitindo assim que se efetua tanto a validação do tráfego MQTTS normal como do anormal.

Os resultados das duas fases estão apresentados nas tabelas Tabela 40 e Tabela 41, sendo que nas mesmas está identificado o tipo de tráfego MQTTS anormal que corresponde ao registos de fluxos de tráfego de rede analisados, o número de registos de fluxos de tráfego de rede que estão presentes no *dataset*, o número de registo de fluxos de tráfego de rede MQTTS que estão presentes no *dataset*, o número de fluxos de tráfego de rede que foram classificados como normais (RFN), o numero de registos de fluxos de tráfego de rede que foram classificados como anormais (RFA), a taxa de deteção (TD) para o teste realizado, a taxa de falsos positivos (FP) e a taxa falsos negativos (FN).

A Tabela 40 apresenta o resultado do teste para os registos de tráfego anormal MQTTS para a fase c).

Tabela 40 - Resultados da deteção de fluxos de tráfego de rede MQTTS anormais por tipo de mensagem

Tipo de mensagem MQTTS	Número de fluxos do *dataset*	Número de fluxos MQTTS	RFN	RFA	FP	FN	TD
net scan / Hping	100	100	0	100	0	0	100%
Publisher / *Subscriber* tópico invalido	100	100	0	100	0	0	100%
Publisher Inundação pedidos válidos e inválidos MQTTS	20	20	6	14	0	0	70%
Subscriber Inundação pedidos válidos e inválidos MQTTS	60	60	0	60	0	0	100%

É de notar que o protótipo não gerou um elevado volume de amostras para o tipo de mensagens "*Publisher* inundação pedidos válidos e inválidos MQTTS" e "*Subscriber* inundação pedidos válidos e inválidos MQTTS".

A Tabela 41 apresenta o resultado do teste para os registos de tráfego anormal MQTT para a fase d).

Tabela 41 - Resultado da deteção de fluxos de tráfego de rede anormais MQTTS

Dataset	Número de fluxos do *dataset*	Número de fluxos MQTTS	RFN	RFA	FP	FN	TD
Dataset	34577	9341	29	9312	0	0	100%

Para a situação 2 na fase c), o volume de tráfego produzido originou uma taxa de deteção de tráfego anormal de 100% e uma taxa de falsos positivos e de falsos negativos de 0%. Estes resultados eram os pretendidos e os esperados para a classificação do tráfego MQTTS normal, e demonstram que as especificações desenvolvidas estão adaptadas para este tipo de tráfego.

Relativamente ao tipo de mensagem "*Publisher* inundação de pedidos válidos e inválidos MQTTS" não foi gerado uma amostragem elevada para análise deste tipo de anomalia, sendo encontrados apenas 20 registos de fluxos de tráfego de rede MQTTS anormal. Alguns fluxos de tráfego de rede são de tráfego sem anomalias, com uma taxa de deteção de 70% e para os falsos positivos e falsos negativos de 0%.

Relativamente ao tipo de mensagem "*Subscriber* inundação de pedidos válidos e inválidos MQTTS" também se detetou poucos registos de fluxos de tráfego de rede gerados pelo protótipo para este tipo de anomalia, sendo que a taxa de deteção é de 100% e de 0% para os falsos positivos e falsos negativos.

No entanto, realça-se que se a amostragem fosse maior, a taxa de deteção (TD) continuaria a ter uma taxa elevada para ambos.

Para a situação 2 na fase d), para o volume de tráfego existente no *dataset*, o teste resultou numa taxa de deteção de tráfego anormal de 100% e uma taxa de falsos positivos e de falsos negativos de 0%.

Verifica-se que, apesar da amostragem elevada de registos de fluxos de tráfego de rede anormal MQTTS, o volume de registos de fluxos de tráfego de rede MQTTS normal foi baixo, não existindo, portanto, uma amostragem elevada para análise.

Em suma, para ambas as situações, verifica-se que os resultados dos testes definidos no capítulo 5.3.3 e que utilizam as especificações criadas nos capítulos 5.2.3 e 5.2.4, permitem demostrar que as especificações apresentam um grau de exatidão satisfatório para o protótipo e para as anomalias testadas, apesar de existirem baixos volumes de registos de fluxos de tráfego de rede normais MQTTS gerados pelo protótipo.

6.4.Síntese

No presente capítulo, apresentou-se inicialmente um resumo do plano de testes e das quatro partes que dividem o presente capítulo.

Na primeira parte foram apresentados os resultados que se obtiveram na validação do funcionamento da aplicação IDS. E, analisando os resultados deste teste, pode-se afirmar que o objetivo foi atingido com sucesso.

Na segunda parte pretendeu-se validar os resultados para classificar, como normal ou como anormal, os registos de fluxos de tráfego de rede MQTT. Para a classificação foram utilizadas as especificações que estão descritas nos capítulos 5.2.1 e 5.2.2.

Na terceira parte pretendeu-se validar os resultados para classificar, como normal ou como anormal, os registos de fluxos de tráfego de rede MQTTS. Para a classificação foram utilizadas as especificações que estão descritas nos capítulos 5.2.3 e 5.2.4.

Por fim, foram expostos e discutidos os resultados obtidos da segunda e terceira partes, isto é, os resultados obtidos na classificação dos registos de fluxos de tráfego de rede MQTT e do MQTTS, respetivamente.

Assim sendo, verificou-se que todos os tráfegos MQTT normal foram detetados e que portanto, a respetiva taxa de deteção foi de 100%. O mesmo aconteceu para o tráfego MQTT anormal, onde a taxa de deteção foi também de 100%.

Á semelhança do MQTT, o MQTTS normal e anormal também obtiveram taxas de deteção de 100%. Realça-se que o protótipo gerou poucos registos de fluxos de tráfego de rede MQTTS normal, contudo, a especificação tem a capacidade de classificar o tráfego de forma correta.

Em resumo, os resultados obtidos estão de acordo com o esperado na definição do plano de testes o que demostra que as especificações estão bem definidas.

7. Conclusões

A realização do presente trabalho permitiu aprofundar os conhecimentos sobre o conceito IoT, nomeadamente, sobre os equipamentos utilizados, os protocolos e os *standards* existentes, assim como, sobre os problemas de segurança inerentes ao conceito IoT e as possíveis soluções.

O levantamento do estado da arte sobre este tema, permitiu o desenvolvimento e definição de especificações para o tráfego MQTT e MQTTS, a fim de serem implementadas numa arquitetura também proposta no âmbito deste projeto. A arquitetura proposta está integrada num IDS para um sistema IoT e permite que, em tempo útil, seja efetuada a deteção de intrusões, oriundas da rede interna ou da rede externa, nas camadas de rede e de aplicação da arquitetura IoT.

A primeira parte de trabalho de projeto, Capítulo 2, faz referência ao estado atual dos sistemas IoT, dos diferentes elementos que contém um sistema IoT, das suas características e dos diferentes protocolos e *standards* existentes. Os problemas de segurança de um sistema IoT também são abordados, tal como as medidas que podem ser tomadas para resolver esses mesmos problemas.

Ainda na revisão de literatura, no Capítulo **Erro! A origem da referência não foi encontrada.**, foram estudados os IDS existentes, ou seja, os diferentes tipos (NIDS, HIDS e o híbrido), as localizações possíveis, os métodos de deteção, a origem dos dados e como é efetuada a monitorização dos fluxos de tráfego de rede. Posteriormente, fez-se a análise dos vários estudos e dos trabalhos realizados na área dos sistemas de IDS adaptados para um sistema IoT, sendo que se constata que ainda não existe uma solução ideal e consensual de um IDS para um sistema IoT.

No seguimento dos conhecimentos estudados e adquiridos, no Capítulo 4 apresentou-se uma proposta de um IDS para um sistema IoT, na qual é descrito a sua localização, o seu método de deteção e os elementos que constituem o IDS proposto, o funcionamento de cada elemento e as diferentes etapas da arquitetura IDS proposta. Este capítulo finaliza com a descrição dos IE relevantes para a caraterização do tráfego MQTT e MQTTS.

De seguida, no Capítulo 5, descreveu-se a aplicação IDS que está inserida no módulo central IDS e que tem o propósito de verificar a existência de novos registos de fluxos de tráfego de rede e de os classificar como normal ou anormal.

Para tal, neste capítulo foi descrita a forma de como foram produzidas as especificações de fluxos de tráfego de rede normal para o MQTT e para o MQTTS, com o objetivo de classificar os fluxos de tráfego de rede como normais ou como anormais. No final, foi apresentado um plano de testes com o intuito de validar as especificações que estão descritas neste capítulo.

Por fim, no Capítulo 6, foi executado o plano de testes para a validação da aplicação IDS e para a validação das especificações expostas nos capítulos anteriores, seguindo-se a análise dos resultados obtidos para todos os testes definidos no plano de testes.

As especificações produzidas para os registos de fluxos de tráfego de rede MQTT e MQTTS, que foram gerados pelo protótipo, demonstram resultados satisfatórios, com uma taxa de deteção de 100% de registos de fluxos de tráfego de rede MQTT, bem como, de registos de fluxos de tráfego de rede MQTTS.

Em relação ao tráfego MQTTS, apesar da taxa de deteção ser de 100%, o protótipo não gerou um volume elevado de registos de fluxos de tráfego MQTTS normal. Esta ocorrência tem como efeito uma reduzida amostra de volume considerável para análise e afinação das especificações para os registos de fluxos de tráfego MQTTS. Contudo, as especificações tiveram a capacidade de detetar de forma correta os poucos registos de fluxos de tráfego de rede MQTTS normal.

Tendo em conta os resultados que se obtiveram com a utilização da aplicação IDS, concluiu-se que a arquitetura proposta para IDS, cujo método de deteção é baseado em especificações, é aplicável e válida para um sistema IoT.

Por fim, realça-se que os objetivos definidos para a execução deste trabalho de projeto foram alcançados. A compreensão e a análise das vulnerabilidades mais relevantes possibilitaram o entendimento da forma de como um dispositivo IoT se encontra vulnerável, devido às limitações ao nível da computação, do consumo energético, assim como, da grande diversidade de dispositivos e sistemas IoT, causando dificuldades de integração de sistemas de segurança.

7.1. Principais Contribuições

Tendo em conta os objetivos definidos no início deste trabalho de projeto e considerando os resultados obtidos, as principais contribuições foram:

- Levantamento das soluções existentes para um IDS e as suas respetivas limitações num sistema IoT, contribuindo para um melhor entendimento do estado da arte sobre os tema dos IDS em sistemas IoT;
- Estudo e análise do tráfego MQTT e MQTTS, de forma a compreender o comportamento normal deste tipo de tráfego. Apesar de existirem mais protocolos utilizados em sistemas IoT, estes dois protocolos são amplamente utilizados devido a serem de fácil utilização;
- Elaboração das especificações para os registos de fluxos de tráfego de rede normal MQTT e MQTTS;
- Contribuição para a comunidade científica através da publicação de um artigo (Leal et al. 2019).

7.2. Tópicos para Trabalho Futuro

Após o desenvolvimento deste trabalho, verifica-se que ainda há muito conhecimento para adquirir no âmbito dos sistemas IoT, visto ser um ambiente bastante abrangente e heterogéneo.

Tendo em conta que os protocolos MQTT e MQTTS não foram desenvolvidos para serem utilizados em sistemas IoT, verifica-se, no entanto, que são largamente utilizados neste tipo de sistemas. Na realização deste trabalho de projeto foram adotadas algumas seleções que orientaram para a escolha dos protocolos aplicacionais MQTT e MQTTS a empregar nesta solução.

Neste sentido, sugere-se também como trabalho futuro a utilização de outros níveis de QoS, como o nível um ou o nível dois, já que estes níveis permitem a geração de outros tipos de registos de fluxos de tráfego de rede devido à utilização de um controlo no envio e entrega das comunicações entre o *Publisher*, o *Subscriber* e o *Broker*.

Relativamente à geração de ataques, efetuaram-se ataques de vários tipos, no entanto cada um em separado. A geração de diferentes tipos de ataques em simultâneo é outra proposta

de trabalho futuro para as especificações desenvolvidas no presente trabalho de projeto, visto que seria interessante averiguar se estas especificações têm a capacidade de lidar com outras situações de ataques e obterem igualmente resultados satisfatórios. A título de exemplo, seria interessante verificar se as especificações desenvolvidas têm a capacidade de atuar em ataques do tipo *slow rate*, sendo que estes ataques poderiam atuar separadamente ou em simultâneo com outros tipos de ataques.

Para além dos protocolos e *standards* referidos no presente trabalho é de realçar que existem outros, que também poderiam ser utilizados no protótipo proposto. O estudo e a análise de outros protocolos e *standards* poderiam levar à criação de outras especificações, mais adequadas, a serem utilizadas para detetar tráfego normal na arquitetura proposta.

Por fim, recomenda-se também como trabalho futuro, o estudo de outros métodos de deteção. A utilização do método baseado em assinaturas ou o método híbrido, combinação entre dois ao mais métodos de deteção, por exemplo, a junção dos métodos baseados em assinaturas e em especificações, possivelmente, poderia levar a um incremento da taxa de deteção e a um aumento do número e do tipo de ataques detetados. Outro método interessante de abordagem de deteção seria através do recurso à utilização de inteligência artificial.

Bibliografia

Alaidaros, H., Mahmuddin, M., & Al Mazari, A. (2011). An overview of flow-based and packet-based intrusion detection performance in high speed networks. *Proceedings of the International Arab Conference on Information Technology*, 1-9.

Alcock, S., Lorier, P., & Nelson, R. (2012). Libtrace: a packet capture and analysis library. *ACM SIGCOMM Computer Communication Review*, 42-48.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys \& tutorials*, 2347-2376.

Amaral, J., Oliveira, L., Rodrigues, J., Han, G., & Shu, L. (2014). Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks. *2014 IEEE International Conference on Communications (ICC)*, 1796-1801.

Anwar, S., Mohamad Zain, J., Zolkipli, M., Inayat, Z., Khan, S., Anthony, B., & Chang, V. (2017). From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions. *Algorithms*, 39.

Atzori, L., Iera, A., & Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, pp. 2787-2805.

bankinfosecurity. (28 de janeiro de 2020). *bankinfosecurity*. Obtido de bankinfosecurity: https://www.bankinfosecurity.com/massive-botnet-attack-used-more-than-400000-iot-devices-a-12841

Canuto, L., Santos, L., Vieira, L., Gonçalves, L., & Rabadão, C. (junho de 2019). CoAP Flow Signatures for the Internet of Things. *In 2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-6.

Carnegie Mellon University. (10 de fevereiro de 2020). *super_mediator*. Obtido de super_mediator: https://tools.netsa.cert.org/super_mediator/index.html

Cervantes, C., Poplade, D., Nogueira, M., & Santos, A. (2015). Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 606-611.

Cherdantseva, Y., & Hilton, J. (2013). A reference model of information assurance & security. *2013 International Conference on Availability, Reliability and Security*, pp. 546 - 555.

Cho, E., Kim, J., & Hong, C. (2009). Attack model and detection scheme for botnet on 6LoWPAN. *Asia-Pacific Network Operations and Management Symposium*, 515-518.

Claise, B., & Bryant, S. (2008). RFC 5101: Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information. *bep. notaroo. net*.

ct-mplane.eu. (5 de fevereiro de 2020). *Quality of Flow (QoF)* . Obtido de Quality of Flow (QoF) : https://www.ict-mplane.eu/public/qof

darkreading. (28 de janeiro de 2020). *darkreading*. Obtido de darkreading: https://www.darkreading.com/attacks-breaches/iot-attacks-up-significantly-in-first-half-of-2019/d/d-id/1336096

Deri, L., & SpA, N. (2003). nProbe: an open source netflow probe for gigabit networks. *TERENA Networking Conference*, 1-4.

Eclipse Foundation, Inc. (08 de fevereiro de 2020). *Eclipse Foundation, Inc*. Obtido de iot mosquitto: https://projects.eclipse.org/projects/iot.mosquitto

Evangelista, S. V. (2008). *Sistemas de detecção de intrusos e sistemas de prevenção de intrusos: Princípios e aplicação de entropia.*

Gupta, A., Pandey, O., Shukla, M., Dadhich, A., Mathur, S., & Ingle, A. (2013). Computational intelligence based intrusion detection systems for wireless communication and pervasive computing networks. *2013 IEEE International Conference on Computational Intelligence and Computing Research*, 1-7.

hping. (17 de marco de 2020). *hping*. Obtido de hping: http://www.hping.org/

Inacio, C. M., & Trammell, B. (2010). Yaf: yet another flowmeter. *Proceedings of LISA10: 24th Large Installation System Administration Conference*, 107.

Inayat, Z., Gani, A., Anuar, N., Khan, M., & Anwar, S. (2016). Intrusion response systems: Foundations, design, and challenges. *Journal of Network and Computer Applications*, 53-74.

infosecurity-magazine. (28 de janeiro de 2020). *infosecurity-magazine*. Obtido de infosecurity-magazine: https://www.infosecurity-magazine.com/news/over-100-million-iot-attacks/

iot-analytics. (2 de fevereiro de 2020). *iot-analytics*. Obtido de iot-analytics: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

ittsystems. (17 de março de 2020). *netflow-vs-ipfix*. Obtido de https://www.ittsystems.com/: https://www.ittsystems.com/netflow-vs-ipfix/

Jacobson, V., Leres, C., & McCanne, S. (1994). libpcap, Lawrence Berkeley Laboratory, Berkeley, CA. *Initial public release June*.

Jakub, S., Ibrahim, G., & Vaclav, P. (2015). Network monitoring approaches: An overview. *Int J Adv Comput Netw Secur*, 88-93.

Jun, C., & Chi, C. (2014). Design of complex event-processing ids in internet of things. *2014 Sixth International Conference on Measuring Technology and Mechatronics Automation*, 226-229.

Kasinathan, P., Costamagna, G., Khaleel, H., Pastrone, C., & Spirito, M. (2013a). An IDS framework for internet of things empowered by 6LoWPAN. *Proceedings of the 2013 ACM SIGSAC conference on Computer \& communications security*, 1337-1340.

Kasinathan, P., Pastrone, C., Spirito, M., & Vinkovits, M. (2013b). Denial-of-Service detection in 6LoWPAN based Internet of Things. *2013 IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob)*, 600-607.

Lampert, R. T., Sommer, C., Munz, G., & Dressler, F. (2006). Vermont-a versatile monitoring toolkit for IPFIX and PSAMP. *Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006)*.

Le, A., Loo, J., Chai, K. K., & Aiash, M. (2016). A specification-based IDS for detecting attacks on RPL-based network topology. *Information*, 25.

Le, A., Loo, J., Luo, Y., & Lasebae, A. (2011). Specification-based IDS for securing RPL from topology attacks. *2011 IFIP Wireless Days (WD)*, 1-3.

Leal, R., Santos, L., Vieira, L., Gonçalves, R., & Rabadão, C. (2019). MQTT flow signatures for the Internet of Things. *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-5.

Lee, T.-H., Wen, C.-H., Chang, L.-H., Chiang, H.-S., & Hsieh, M.-C. (2014). A lightweight intrusion detection scheme based on energy consumption analysis in 6LowPAN. *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*, 1205-1213.

Li, B., Springer, J., Bebis, G., & Gunes, M. (2013). A survey of network flow applications. *Journal of Network and Computer Applications*, 567-581.

Liao, H.-J., Lin, C.-H., Lin, Y.-C., & Tung, K.-Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, 16-24.

Liu, C., Yang, J., Chen, R., Zhang, Y., & Zeng, J. (2011). Research on immunity-based intrusion detection technology for the internet of things. *2011 Seventh International Conference on Natural Computation*, 212-216.

Midi, D., Rullo, A., Mudgerikar, A., & Bertino, E. (2017). Kalis—A system for knowledge-driven adaptable intrusion detection for the Internet of Things. *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 656-666.

Misra, S., Krishna, P., Agarwal, H., Saxena, A., & Obaidat, M. (2011). A learning automata based solution for preventing distributed denial of service in internet of things. *2011 international conference on internet of things and 4th international conference on cyber, physical and social computing*, 114-122.

mqtt.org. (28 de março de 2020). *mqtt.org*. Obtido de mqtt.org: http://mqtt.org/

nmap. (17 de março de 2020). *nmap*. Obtido de https://nmap.org/: https://nmap.org/

ntop. (08 de fevereiro de 2020). *https://www.ntop.org/guides/pf_ring/*. Obtido de https://www.ntop.org/guides/pf_ring/: https://www.ntop.org/guides/pf_ring/

Oh, D., Kim, D., & Ro, W. W. (2014). A malicious pattern detection engine for embedded security systems in the Internet of Things}. *Sensors*, 24188-24211.

Pongle, P., & Chavan, G. (2015). Real time intrusion and wormhole attack detection in internet of things. *International Journal of Computer Applications*, 9.

Raza, S., Wallgren, L., & Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. *Ad hoc networks*, 2661-2674.

Sain, M., Kang, Y., & Lee, H. (2017). Survey on security in Internet of Things: State of the art and challenges. *2017 19th International conference on advanced communication technology (ICACT)*, 699-704.

Santos, L. (2020). *Sistema de deteção de intrusões para a Internet das Coisas.* Vila Real: Universidade de Trás-os-Montes e Alto Douro.

Santos, L., Rabadao, C., & Gonçalves, R. (2018). Intrusion detection systems in Internet of Things: A literature review. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 1-7.

Santos, L., Rabadão, C., & Gonçalves, R. (2019a). Flow Monitoring System for IoT Networks. *World Conference on Information Systems and Technologies*, 420-430.

Santos,, L., Gonçalves, R., & Rabadão, C. (2019b). A Novel Intrusion Detection System Architecture for Internet of Things Networks. *in ECCWS 2019 18th European Conference on Cyber Warfare and Security*, p. 428.

Shreenivas, D., Raza, S., & Voigt, T. (2017). Intrusion detection in the RPL-connected 6LoWPAN networks. *Proceedings of the 3rd ACM international workshop on IoT privacy, trust, and security*, 31-38.

Summerville, D. H., Zach, K. M., & Chen, Y. (2015). Ultra-lightweight deep packet anomaly detection for Internet of Things devices. *2015 IEEE 34th international performance computing and communications conference (IPCCC)*, 1-8.

Thanigaivelan, N. K., Nigussie, E., Kanth, R. K., Virtanen, S., & Isoaho, J. (2016). Distributed internal anomaly detection system for Internet-of-Things. *2016 13th IEEE Annual Consumer Communications \& Networking Conference (CCNC)*, 319-320.

Valério, G. N. (2015). *Segurança e Privacidade na massificação da Internet dos Objectos.* Coimbra.

varonis. (17 de Fevereiro de 2020). *varonis*. Obtido de what-is-siem: https://www.varonis.com/blog/what-is-siem/)

Vieira, L., Santos, L., Gonçalves, R., & Rabadão, C. (2019). Identifying Attack Signatures for the Internet of Things: An IP Flow Based Approach. *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*, pp. 1-7.

Wallgren, L., Raza, S., & Voigt, T. (2013). Routing attacks and countermeasures in the RPL-based internet of things. *International Journal of Distributed Sensor Networks*, 794326.

Zarpelão, B. B., & Miani, R. S. (2017). A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications*, 25-37.