ISEL
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Projeto – Relatório Final

Home Security – Network-based Intrusion Detection System for domestic IoT

Semestre de Verão 2019/2020

Licenciatura em Engenharia Informática, Redes e Telecomunicações

Alunos:
Guilherme Cardoso, nº 45223
Tiago Domingues, nº 45219

Orientador:
Prof. Vítor Almeida

Responsável:
Prof. Paulo Marques

Arguente:
Prof. Nuno Cruz

Presidente do júri:
Prof. Jorge Martins

14 de julho de 2020

Resumo

Confrontados com a adoção exponencial de novos equipamentos associados ao conceito de *Internet of Things* por parte de milhões de famílias espalhadas por todo o mundo, surgiu a irrefutável necessidade de se desenvolver uma aplicação para ajudar a combater e mitigar a fraca ou, por vezes, inexistente segurança implementada por este tipo de dispositivos através do policiamento da rede e, mais concretamente, dos equipamentos associados a este conceito relativamente recente.

Este relatório documenta e resume todo o processo de estudo, preparação e investigação que foi necessário realizar para que se pudessem reunir as condições essenciais à implementação dessa aplicação.

Posto isto, ao longo deste documento, são analisados os diversos passos e fatores associados a esse processo e são relatadas, da forma mais clara e realista possível, todas as decisões e pareceres que tiveram de ser tomados durante a realização da aplicação.

Palavras-chave

Ataques; Cibersegurança; *Internet of Things*; *Intrusion Detection System*; Linux; Python; Raspberry Pi; Redes domésticas; Vulnerabilidades.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

ACL – *Access Control List*

AMQP – *Advanced Message Queuing Protocol*

AP – *Access Point*

API – *Application Programming Interface*

ARP – *Address Resolution Protocol*

ASCII – *American Standard Code for Information Interchange*

CoAP – *Constrained Application Protocol*

CVE – *Common Vulnerabilities and Exposures*

DDoS – *Distributed Denial of Service*

DDS – *Data Distribution Service*

DHCP – *Dynamic Host Configuration Protocol*

DNS – *Domain Name System*

DNS-SD – *DNS-based Service Discovery*

DNSSEC – *Domain Name System Security Extensions*

DoS – *Denial of Service*

DTLS – *Datagram Transport Layer Security*

EAP – *Extensible Authentication Protocol*

HIDS – *Host-based Intrusion Detection System*

HMAC – *Hash-based Message Authentication Code*

HTTP – *Hypertext Transfer Protocol*

HTTPS – *Hypertext Transfer Protocol Secure*

IANA – *Internet Assigned Numbers Authority*

ICMP – *Internet Control Message Protocol*

IDS – *Intrusion Detection System*

IEEE – *Institute of Electrical and Electronics Engineers*

IoT – *Internet of Things*

IP – *Internet Protocol*

IPS – *Intrusion Prevention System*

IPv4 – *Internet Protocol version* 4

IPv6 – *Internet Protocol version* 6

IRC – *Internet Relay Chat*

ISP – *Internet Service Provider*

IV – *Initialization Vector*

JSON – *JavaScript Object Notation*

LAN – *Local Area Network*

MAC – *Message Authentication Code*

mDNS – *Multicast Domain Name System*

MITM – *Man-in-the-Middle*

MQTT – *Message Queuing Telemetry Transport*

MQTT-SN – *Message Queuing Telemetry Transport for Sensor Networks*

NAT – *Network Address Translation*

NIDS – *Network-based Intrusion Detection System*

PKI – *Public Key Infrastructure*

QoS – *Quality of Service*

RAM – *Random Access Memory*

RPC – *Remote Procedure Call*

SASL – *Simple Authentication and Security Layer*

SMS – *Short Message Service*

SSDP – *Simple Service Discovery Protocol*

SSL – *Secure Sockets Layer*

TCP – *Transmission Control Protocol*

TLS – *Transport Layer Security*

UDP – *User Datagram Protocol*

uPnP – *Universal Plug and Play*

WAN – *Wide Area Network*

WEP – *Wired Equivalent Privacy*

WPS – *Wi-Fi Protected Setup*

XML – *Extensible Markup Language*

XMPP – *Extensible Messaging and Presence Protocol*

Introdução

O aumento exponencial na utilização de equipamentos associados à *Internet of Things* contribui para o aumento do problema relacionado com o nível de confiança que se pode ter nestes dispositivos no que se refere à segurança que implementam. Apesar de começar a aparecer legislação que pretende colocar alguma ordem na maneira como os equipamentos de IoT são usados e na forma como a segurança é implementada (caso da Califórnia, por exemplo), a verdade é que existem milhões de dispositivos em que o controlo em termos de segurança é, no mínimo, praticamente inexistente.

Desta forma, os equipamentos associados ao IoT podem ser utilizados como cavalos de Tróia para fornecerem acesso a estranhos às redes onde são instalados, nomeadamente, às domésticas. Estas redes encontram-se protegidas pelos *firewalls* existentes nos *routers* domésticos, cuja eficiência, na presença de um cavalo de Tróia, é bastante discutível.

Nesse sentido, a criação de um *Network-based Intrusion Detection System* (NIDS) doméstico pode contribuir para melhorar a segurança das redes domésticas, impondo algum controlo na forma de atuar dos dispositivos associados ao IoT. Assim, o objetivo consistiu no desenvolvimento de uma ferramenta do tipo *plug--and-play*, evitando que o utilizador final tenha de lidar com configurações técnicas que pode não dominar. Para além disso, pretende-se que o impacto desta ferramenta no desempenho da rede local existente seja o mínimo possível.

Assim sendo, tem-se como objetivo criar uma ferramenta que policie uma rede doméstica, alertando para casos de tráfego considerados fora do habitual para o tipo de equipamentos associados ao IoT. Esta ferramenta gerará relatórios de tráfego, que serão compostos por uma descrição do momento em que os dados foram trocados, quem os trocou e em que quantidade, para cada um dos equipamentos IoT na rede doméstica. Isto ativará alarmes de segurança se, eventualmente, forem detetados comportamentos não compatíveis com a funcionalidade normal deste tipo de dispositivos.

Motivação

A realização de um Projeto Final de Curso de Licenciatura tem em vista a integração e consolidação das diversas competências que foram adquiridas ao longo do ciclo de estudos e, se possível, a obtenção de conhecimentos sobre áreas que não constam no Plano Curricular do curso. Assim sendo, objetiva-se a implementação de um projeto concreto baseado em tecnologias atuais ou emergentes que permitam preparar os alunos para o mercado de trabalho, desenvolvendo características profissionais.

Desta forma, o nosso intuito residia na realização de um projeto que abrangesse, simultaneamente, as áreas das Redes de Computadores e da Cibersegurança. Áreas estas que foram abordadas no decorrer da Licenciatura e que sempre despertaram um forte e especial interesse da nossa parte.

Tendo isto em consideração, o professor Vítor Almeida propôs o tema deste projeto, aliando a temática emergente da *Internet of Things*. Deste modo, permitiu-nos não só consolidar e desenvolver competências nas áreas anteriormente mencionadas, como obter conhecimentos numa temática totalmente nova no nosso percurso superior.

Posto isto, sabe-se que a Cibersegurança e a *Internet of Things* são duas áreas em ascensão e de grande importância nas Tecnologias de Informação. Com este projeto, pretende-se não só abordar essas duas ramificações cruciais de Informática, como procurar resolver um problema que será cada vez mais preponderante nos dias de hoje.

1 Contextualização e Estado da arte

Este primeiro capítulo pretende introduzir e contextualizar os conceitos de *Internet of Things* e de *Intrusion Detection System*, bem como fornecer uma visão geral sobre o estado da arte, as tecnologias presentes nestes tópicos e os principais componentes que estão envolvidos ou se apresentam como relevantes à realização deste projeto.

1.1 *Internet of Things*

O termo *Internet of Things* ou, traduzido à letra para português, Internet das Coisas ouve-se cada vez mais nos dias de hoje e, por vezes, sem se ter uma noção concreta do conceito a que está associado.

A *Internet of Things* é um sistema de dispositivos associados, principalmente, a objetos e atividades do quotidiano e que se interligam de forma digital para trocar dados sem que a interação humana seja necessária. A definição de IoT é, recorrentemente, associada a diversas áreas e tecnologias, como os sistemas embebidos, os sensores e controladores *wireless*, a automação, etc. [1]

Existem diversos equipamentos associados à *Internet of Things*. Alguns foram inventados com o surgimento deste conceito (por exemplo, colunas inteligentes e capazes de se interligarem a outros dispositivos para os controlarem através de comandos de voz – Alexa da Amazon, Google *Assistant*, etc.) e outros consistem em objetos que já existiam no nosso dia-a-dia e que, agora, evoluíram e passaram a possuir a capacidade de se ligarem a outros dispositivos (como o telemóvel, por exemplo) e à Internet em geral.

Desta forma, alguns dos equipamentos associados à *Internet of Things* são:

- *Smart* TVs;
- Frigoríficos inteligentes (com ligação à Internet);
- Máquinas de lavar loiça/roupa inteligentes;
- Balanças inteligentes;
- *Robots* de cozinha (Bimby, etc.);
- *Robot Vacuum Cleaners* (aspiradores inteligentes);
- AVRs (*Audio*/*Video Receivers*);
- Colunas inteligentes (Alexa da Amazon, Google *Assistant*, etc.);
- Lâmpadas inteligentes e, em alguns casos, as respetivas *bridges* (Hue da Philips, etc.);
- Tomadas e interruptores com ligação à Internet;
- Câmaras de vídeo via IP;
- Sensores e sistemas de alarme;
- entre outros…

Ou seja, basicamente todos os equipamentos associados a tarefas ou objetos que simplifiquem a atividade humana, ou que a eliminem por completo. Dentro destes, existem equipamentos mais complexos (como é o caso das *smart* TVs e da Alexa, por exemplo) e equipamentos mais simples (lâmpadas, interruptores, controladores, etc.) que, geralmente, são os que poderão levantar maiores preocupações relativamente à segurança que implementam.

Para além do uso doméstico, também existem diversos equipamentos de IoT no âmbito empresarial, industrial, militar, entre outras áreas. Contudo, esses não são objeto de estudo neste projeto.

Contrariamente, dispositivos como computadores pessoais (*laptops* ou *desktops*), telemóveis, *tablets*, consolas de jogos, impressoras e *routers* não são, habitualmente, associados ao conceito da Internet das Coisas.

1.1.1 Constante crescimento

Esta tecnologia emergente tem vindo a conquistar cada vez mais utilizadores e, consequentemente, tem ocorrido um crescimento exponencial no número de dispositivos utilizados mundialmente, ano após ano. Em 2018, estimava-se que existissem à volta de 7 mil milhões de dispositivos de IoT ativos em todo o mundo e perspetivava-se que, neste ano de 2020, este número atingisse os 10 mil milhões. As projeções indicavam que, a partir de 2021, passarão a existir mais dispositivos de IoT ativos do que os restantes equipamentos que não estão associados a esta tecnologia. [2]

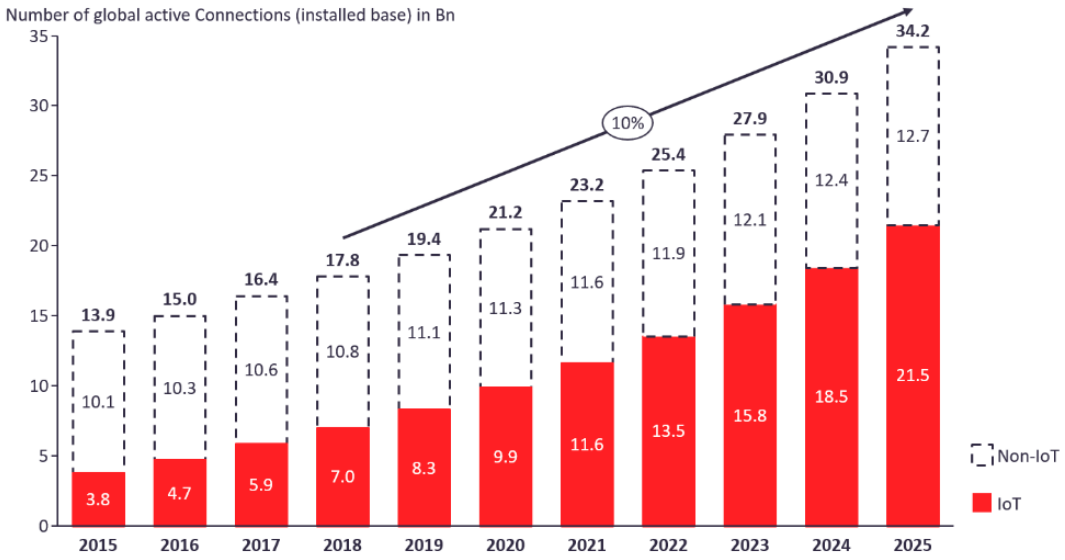

Figura 1 – Estimativa da utilização de dispositivos de IoT ao longo dos anos [2]

Contudo, estas estimativas e previsões estão longe de estar completamente certas, existindo diversos dados e estatísticas provenientes de diferentes instituições que apontam para números diferentes. Por exemplo, há quem projete que, em 2020, o

número de dispositivos IoT atinja os 31 mil milhões [3] e aponte para 64 mil milhões em 2025 [4].

Dito isto, facilmente se percebe que não é fácil prever quantos dispositivos deste tipo existem pelo mundo fora. Apenas se tem duas certezas absolutas: o número está na ordem dos milhares de milhão e, ano após ano, esse número tende a aumentar constantemente.

Posto isto, levantam-se, naturalmente, grandes preocupações relativamente aos perigos existentes neste "ecossistema" em ascensão, especialmente, nas áreas da segurança, incluindo a privacidade, das redes e dos equipamentos.

1.1.2 *Stack* e camadas

Apesar do principal foco deste projeto serem os dispositivos IoT que assentam sobre ligações e comunicações IP, existem equipamentos que utilizam outros tipos de protocolos para comunicar ao nível da camada de *Data Link*. Desta forma, estes equipamentos podem ser subdivididos em duas categorias: baseados em IP e não baseados em IP [5]. Os primeiros consistem em dispositivos que comunicam por WiFi ou Ethernet e os segundos em equipamentos que utilizam outras tecnologias, como, por exemplo, Bluetooth, Zigbee, 6LoWPAN, ZWave, Thread, LoRaWAN ou Sigfox.

Desta forma, a figura 2 apresenta o *stack* das comunicações associadas à *Internet of Things* e aos protocolos mais comuns.

Figura 2 – *Stack* das comunicações IoT [6]

Tal como foi referido anteriormente, existem bastantes tecnologias usadas ao nível da camada de *Data link*. Nas camadas de Rede e de Transporte, utilizam-se as tecnologias já bastante estabelecidas, nomeadamente: IPv4, IPv6, TCP e UDP. Para implementar segurança no transporte, tira-se partido do TLS e do DTLS. Por fim, ao nível da Aplicação, tal como na camada *Data link*, existem diversas e diferentes

tecnologias, que serão abordadas de maneira aprofundada no Capítulo 3 deste relatório.

1.2 *Intrusion Detection System*

Um *Intrusion Detection System* ou, traduzido para português, Sistema de Deteção de Intrusões é um sistema que, tal como o seu nome indica, tem o objetivo de detetar comportamentos suspeitos ou fora do comum através da análise de tráfego, gerando os devidos alertas no caso de surgirem anomalias.

Estes sistemas podem ser divididos, essencialmente, em dois tipos distintos no que toca à atividade que analisam:

- *Host-based* IDS (HIDS);
- *Network-based* IDS (NIDS).

Enquanto que o primeiro consiste num IDS que apenas examina eventos num equipamento específico da rede, o segundo analisa todo o tráfego que circula na rede [7]. Naturalmente, para o efeito que se pretende com o desenvolvimento deste projeto, o *Network-based* é o IDS que permite atingir os objetivos pretendidos, dado que, usualmente, não existe acesso ao *software* (*source*) incluído nos muitos equipamentos IoT.

Relativamente à forma como realiza a deteção de intrusões, os IDS também podem ser separados em dois tipos diferentes:

- *Signature-based*;
- *Anomaly*-based.

O primeiro consiste na deteção de ataques ou intrusões através da análise de padrões específicos e concretos como, por exemplo, sequências de *bytes* ou de pacotes no tráfego da rede. Embora este tipo de IDS seja bastante eficiente na deteção de ataques conhecidos, é-lhe difícil detetar novos ataques, enquanto ainda não existem padrões disponíveis para os mesmos. [8]

O segundo surgiu com a necessidade de se detetarem novos ataques, devido ao rápido aparecimento de novo *malware*. Esta abordagem baseia-se em *Machine Learning* para criar um modelo de atividade confiável e, depois, comparar o tráfego da rede usando este modelo. Apesar deste método permitir a deteção de ataques desconhecidos, sofre um problema de alerta para falsos positivos: o sistema assume a existência de um ataque quando, na verdade, não ocorreu nenhuma anomalia. [9]

Desta forma, a melhor alternativa passa, frequentemente, por adotar um sistema híbrido. Ou seja, um IDS que é, ao mesmo tempo, *Signature-based* e *Anomaly-based*, tirando partido da existência de padrões para detetar ataques conhecidos e da utilização de *Machine Learning* para detetar ataques desconhecidos e para os quais ainda não se estabeleceram padrões.

1.2.1 *Network-based* IDS

Geralmente, este tipo de sistemas funciona à base de regras que podem ser livremente definidas pelo administrador da rede ou regras disponibilizadas pelo próprio fabricante do *software* ou pela comunidade de utilizadores.

Uma vez que um *Network-based Intrusion Detection System* necessita de ter acesso ao tráfego da rede para o poder examinar e, se necessário, gerar alertas, é preciso que este seja colocado de forma estratégica na arquitetura da rede. Uma solução tipicamente utilizada nos NIDS reside na realização de *packet sniffing* para poder fazer a análise do tráfego dos *hosts* da rede. Este *packet sniffer* pode correr diretamente no *software* do IDS, num programa que se execute paralelamente ou, alternativamente, pode até correr noutro dispositivo que, depois, encaminha o tráfego para o NIDS.

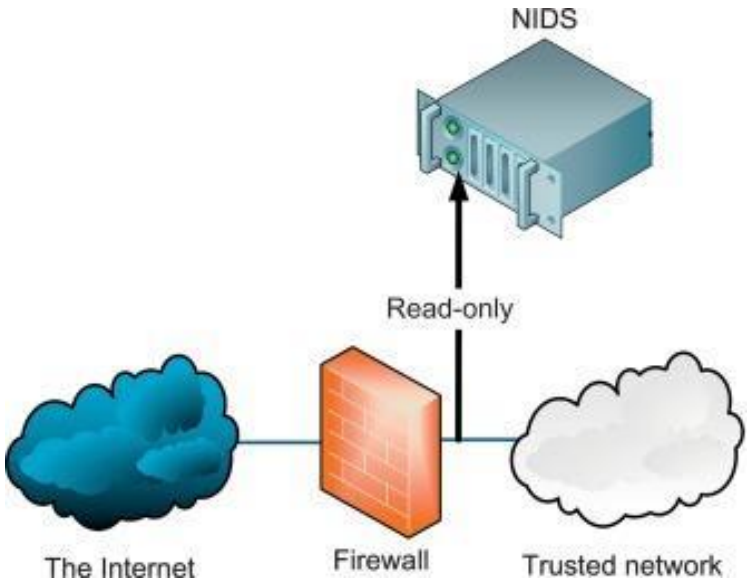

Figura 3 – Cenário 1 de integração do NIDS na rede [10]

Outra possibilidade (ainda que menos utilizada), que dispensa a realização de *packet sniffing*, passa por colocar o NIDS "em linha" com a rede, forçando a que todo o tráfego que circule da LAN para a WAN, e vice-versa, passe pelo mesmo. Para isso, o equipamento também precisa de ser responsável por encaminhar tráfego.

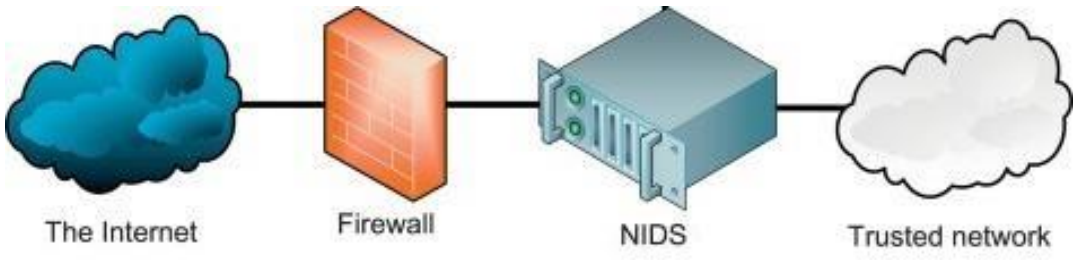

Figura 4 – Cenário 2 de integração do NIDS na rede [10]

É importante referir que um NIDS se trata de um equipamento passivo e, por isso, não altera o tráfego que monitoriza [10]. A sua função é gerar alertas e relatórios dos resultados para o administrador da rede, não realizando qualquer ação para prevenir as anomalias ou ataques detetados na rede. Para esse efeito, existem sistemas designados de *Intrusion Prevention System* (IPS).

1.2.2 Ferramentas e *software* de NIDS

Nos dias de hoje, os *Network-based Intrusion Detection Systems* estão mais do que assentes na indústria e são considerados importantes complementos à segurança das redes e dos equipamentos. Desta forma, existem diferentes soluções e ferramentas gratuitas e *open-source* que desempenham esta funcionalidade:

- Snort;
- Suricata;
- Zeek (antes conhecido como Bro);
- Sagan;
- ...

O Snort é, atualmente, o líder deste mercado. Apesar de ter sido comprado, em 2013, pela Cisco, este *software* ainda permanece gratuito. Para além de realizar deteção de intrusões, esta ferramenta também pode realizar *packet sniffing* e funcionar como um IPS. [11]

O Suricata é, geralmente, a principal alternativa ao Snort. Tal como este, também funciona como *packet sniffer* e *Intrusion Prevention System*. Para além disso, apresenta a vantagem de permitir a análise de dados ao nível da camada de Aplicação e de ser compatível com o Snort relativamente à sintaxe das regras que ditam o funcionamento do NIDS. [12]

O Zeek, assim como os anteriores, também funciona como IPS e realiza *packet sniffing*. Tal como o Suricata, permite realizar análises ao nível da camada de Aplicação. Este programa inclui suporte para importar padrões (*signatures*) do Snort e é o IDS mais antigo dos mencionados. [13]

Por fim, o Sagan, contrariamente aos restantes, não inclui a funcionalidade de *packet sniffing*, pelo que é necessário que esta seja realizada por outra ferramenta ou equipamento. As suas regras utilizam a mesma sintaxe do Snort, garantindo a compatibilidade entre os dois e permitindo que sejam usados em simultâneo. Desta forma, o Sagan tem a capacidade de correlacionar dados com o Snort. [14]

1.3 Lacunas

O grande problema associado ao conceito da *Internet of Things*, tal como já foi referido, reside nas sérias questões e preocupações que se levantam relativamente à fraca ou inexistente implementação de segurança na generalidade destes

equipamentos. Apesar de algumas entidades governamentais e a própria indústria já se estarem a movimentar no sentido de endereçar estes problemas [1] e até já ter começado a aparecer alguma legislação com o intuito de colocar uma certa ordem na maneira como os equipamentos de IoT são usados e na forma como a segurança é implementada, estes perigos estão incrivelmente longe de serem resolvidos ou mitigados.

Associado a este problema, surge também a ausência de *Intrusion Detection Systems* especializados neste tipo de equipamentos e no tráfego que eles geram e recebem. Tal como foi referido anteriormente, existem diversas soluções no mercado e que são, de modo geral, competentes e bem aceites na comunidade. Contudo, todas sofrem da mesma lacuna: são demasiado generalizadas para serem diretamente aplicadas à deteção de intrusões ao nível dos dispositivos associados ao conceito da *Internet of Things*.

Face a estes factos, é necessário procurar aplicar as soluções de NIDS que existem ao problema da falta de segurança e privacidade destes equipamentos. Dado que não existe nenhum IDS focado em IoT, nem nenhum que apresente uma funcionalidade para esse efeito, é preciso, de alguma forma, integrar estes sistemas com outras ferramentas e aplicações, no sentido de endereçar direta e convenientemente a deteção de ataques e anomalias que possam existir nestes dispositivos e que, consequentemente, podem colocar toda a rede vulnerável a invasões.

2 Objetivos e foco do Projeto

Dada a ascensão da utilização de equipamentos e utensílios inteligentes, que são ligados à Internet, num contexto doméstico e familiar por indivíduos com pouco ou nenhum conhecimento de Informática, existe uma crescente necessidade de proteger os utilizadores e as suas redes domésticas contra as falhas de segurança e vulnerabilidades existentes nos múltiplos dispositivos de IoT espalhados pelas casas de milhões e milhões de pessoas por todo o mundo. Desta forma, o desenvolvimento de um NIDS para policiar e, com isso, proteger as redes das famílias tende a ser algo inovador e também muito necessário, o mais rapidamente possível.

Dito isto, o foco deste Projeto está centrado nos dispositivos IoT utilizados num âmbito doméstico e pretende-se, essencialmente, que possa ser implementado na rede doméstica de qualquer pessoa ou família. Ou seja, é necessário ter em mente que esta ferramenta tem de ser do tipo *plug & play* e a sua utilização tem de envolver o mínimo de configurações necessárias por parte do utilizador. Para além disso, é importante ter em consideração a arquitetura utilizada em redes domésticas e o tipo de *routers* que são usados para este efeito, bem como as suas funcionalidades, de modo a integrar esta ferramenta da melhor forma possível.

Tal como explicado anteriormente, existem diferentes tipo de dispositivos associados à *Internet of Things* e distintas tecnologias que podem ser usadas pelos mesmos. Assim, o objetivo desta aplicação de policiamento da rede é focar-se nos dispositivos IoT que utilizam ligações IP, nas comunicações que realizam para fora da rede e nas comunicações com origem no exterior da rede que sejam destinadas a estes equipamentos. Contudo, isto não invalida que, ao longo deste relatório, se possam abordar outras tecnologias e protocolos que não sejam o principal foco do projeto ou outro tipo que tráfego que não aquele que se pretende capturar, analisar e policiar através do NIDS, mas que, por qualquer motivo, se mostrem pertinentes de abordar, enriquecendo o relatório e o estudo necessário para o desenvolvimento do projeto.

Posto isto, é oportuno realçar que não se pretende reinventar as aplicações associadas a estas tecnologias ou revolucionar as áreas abordadas, mas sim pegar em *software* e ferramentas já existentes e devidamente aceites e testadas pela comunidade e desenvolver, através da integração das mesmas, algo que ainda não existe e que se tem revelado como cada vez mais essencial.

3 Estudo de Protocolos associados a IoT

O desenvolvimento deste projeto iniciou-se com um estudo conciso dos protocolos mais utilizados nos equipamentos associados à *Internet of Things*, num âmbito doméstico.

O objetivo desta fase inicial consiste na realização de uma pesquisa acerca desses protocolos, de modo a obter uma ideia geral sobre o seu funcionamento e sobre a forma como implementam segurança nos dispositivos que os utilizam para comunicar.

Nos dias de hoje, existem diversas tecnologias implementadas com diferentes funcionalidades e aplicações e que são altamente usadas ao nível da camada *Data link* do *stack* das comunicações IoT: *Low-rate* WPAN (IEEE 802.15.4), *Bluetooth* (e *Bluetooth Low Energy*), Zigbee, Z-Wave, Thread, Sigfox, NFC, 6LoWPAN, LoraWAN, entre outras.

Naturalmente, estes *standards* e protocolos tentam implementar alguma segurança nas comunicações entre os equipamentos que os utilizam, mas, ainda assim, existem falhas e vulnerabilidades nos mesmos.

Contudo, estas tecnologias não constituem o principal propósito da realização deste projeto e, por essa mesma razão, não serão desenvolvidas e elaboradas ao longo deste relatório. Em vez disso, focar-se-á principalmente nas tecnologias e protocolos associados à *Internet of Things* que assentam sobre ligações TCP/IP e que se enquadram na camada de Aplicação, tal como foi visto no Capítulo 1.1.2.

Posto isto, ao longo deste capítulo, serão explorados sete protocolos bastante utilizados no universo das comunicações associadas a IoT que diferem na sua estrutura e modelo de interação. Assim sendo, objetiva-se explorar os seguintes protocolos de *messaging*:

- MQTT;
- CoAP;
- AMQP;
- DDS;
- XMPP.

E os seguintes protocolos de *service discovery*:

- mDNS;
- SSDP.

Com especial foco no MQTT, CoAP, mDNS e SSDP, dado que são os principais protocolos de eleição para a implementação de dispositivos e aplicações destinadas a ambientes domésticos de *Internet of Things*.

3.1 *Message Queuing Telemetry Transport*

Arquitetura

O *Message Queuing Telemetry Transport* (MQTT) possui um modelo de cliente/servidor em que cada sensor funciona como um cliente que se liga a um servidor (conhecido como um *broker*) sobre TCP. O MQTT é orientado à troca de mensagens e, por isso, todas as mensagens são tratadas como um conjunto discreto de dados, opaco para o *broker*.

As mensagens são publicadas com um determinado endereço, conhecido como tópico. Os vários clientes podem inscrever-se em diversos tópicos e receberão todas as mensagens publicadas nos tópicos a que estão inscritos [15].

A título de exemplo, pode-se imaginar um cenário elementar de uma rede simples com apenas três clientes e um *broker* central. Todos os clientes irão estabelecer ligações TCP com o *broker*. Como é descrito na Figura 5, os clientes B e C subscrevem o tópico "*temperature*", através do envio de uma mensagem de *subscribe* para o *broker*.

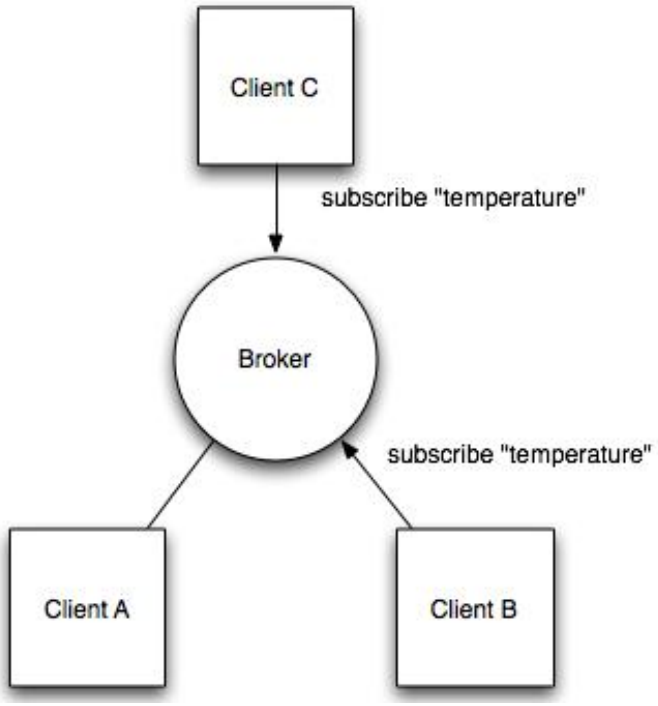

Figura 5 – Subscrição de um tópico em MQTT [15]

Posteriormente, como se observa na Figura 6, o cliente A publica um valor de 22,5 no tópico, através do envio de uma mensagem de *publish*. Consequentemente, o *broker* encaminha esse novo valor publicado no tópico para todos os clientes nele inscritos.

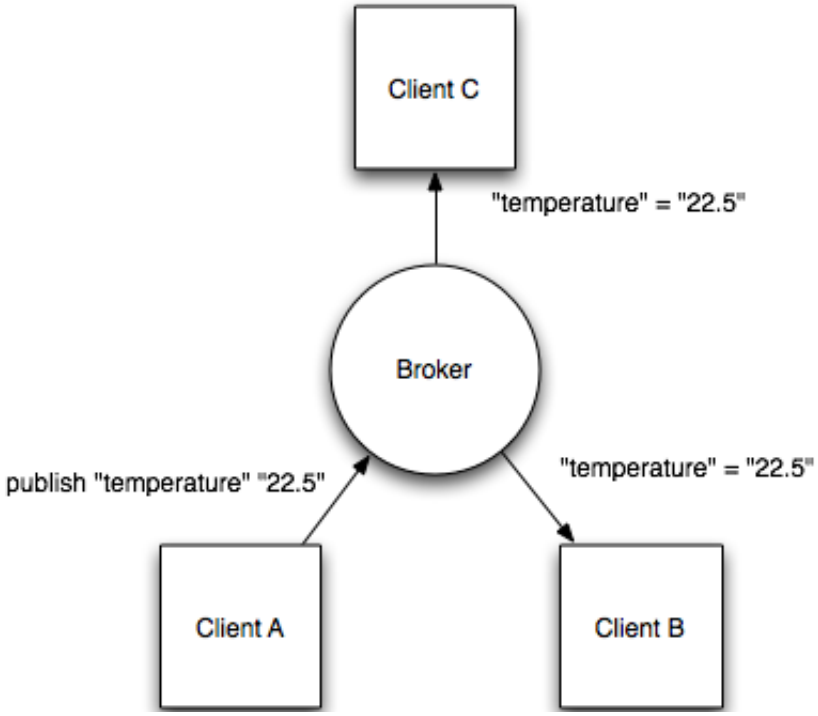

Figura 6 – Propagação de uma mensagem em MQTT [15]

O modelo de *publish-subscribe* permite que os clientes do MQTT comuniquem de um para um, de um para muitos e de muitos para um [16].

QoS

O MQTT suporta três níveis de qualidade de serviço: "*at most once*", "*at least once*" e "*exactly once*".

No primeiro (também conhecido por "*fire and forget*"), a mensagem é enviada e não existe qualquer tipo de *acknowledge*. No segundo (também conhecido por "*acknowledged delivery*"), a mensagem é reenviada até ser recebido um *aknowledge*. No último (também conhecido por "*assured delivery*"), ocorre um *handshake* para garantir que apenas uma cópia da mensagem é recebida [16].

Last Will and Testament

Os clientes MQTT podem registar uma mensagem personalizada de "*last will and testament*", a ser enviada pelo *broker* se eles se desconectarem. Essas mensagens podem ser usadas para sinalizar os restantes assinantes de um tópico quando esse dispositivo é desconectado [15].

Persistência

O MQTT possui suporte para mensagens persistentes, armazenadas no *broker*. Ao publicar mensagens, os clientes podem solicitar que o *broker* armazene a mensagem. Apenas a mensagem persistente mais recente é que fica guardada.

Quando um novo cliente assina esse tópico, qualquer mensagem persistente ser-lhe--á enviada [15].

MQTT-SN

Embora tenha sido projetado para ser um protocolo "leve", o MQTT tem duas desvantagens para dispositivos de muito baixo poder computacional:

- Todos os clientes MQTT devem suportar TCP e, normalmente, manter uma ligação aberta para o *broker*. Para alguns ambientes em que a perda de pacotes é alta ou os recursos de computação são escassos, isto poderá ser um problema.
- Os nomes dos tópicos do MQTT são, geralmente, cadeias longas, o que os torna impraticáveis em 802.15.4 (*standard* IEEE), dado que os pacotes deste protocolo precisam de ter a dimensão mínima possível e, por isso, a utilização de nomes extensos vai contra esse princípio.

Essas duas desvantagens são resolvidas pelo *Message Queuing Telemetry Transport For Sensor Networks* (MQTT-SN), que define um mapeamento UDP do MQTT e inclui suporte ao *broker* para indexar nomes de tópicos [15]. Esta variação do protocolo também pode ser utilizada em redes não baseadas em TCP/IP, como, por exemplo, o Zigbee. [16]

Segurança

Os agentes do MQTT podem requisitar autenticação dos clientes por meio de utilizador e *password*. Por omissão, estas credenciais são enviadas em claro (*plaintext*) na rede, não garantindo qualquer segurança! Ou seja, qualquer atacante com acesso ao tráfego que circule entre o cliente e o servidor pode intercetá-lo e obter as credenciais de autenticação.

Para garantir privacidade, a ligação TCP pode ser realizada sobre SSL/TLS e, desta forma, as mensagens serem todas cifradas. Assim, a informação transferida passa a estar protegida contra interceção, modificação ou falsificação.

Por omissão, o MQTT utiliza o porto 1883, mas, no caso da utilização de TLS, o porto *standard* é o 8883. [17]

3.2 *Constrained Application Protocol*

Arquitetura

Tal como o HTTP, o *Constrained Application Protocol* (CoAP) é um protocolo orientado à comunicação de dados. Por outro lado, o CoAP é projetado para as necessidades de dispositivos mais restritos e de menor capacidade.

Os pacotes do CoAP são muito menores que os fluxos de HTTP e, por isso, são simples de gerar e de serem analisados localmente, sem consumir RAM extra em dispositivos pequenos e de baixo poder computacional.

Este protocolo corre sobre UDP e, consequentemente, os clientes e servidores comunicam através de datagramas *connectionless*. O reenvio e a reorganização dos pacotes são implementados na camada de aplicação. O CoAP permite a utilização de *broadcast* e *multicast*.

O CoAP segue um modelo de cliente/servidor, onde os clientes realizam pedidos aos servidores e estes enviam as respostas. Os clientes podem realizar operações de GET, PUT, POST e DELETE sobre os recursos. O CoAP foi projetado de forma a ser compatível com o HTTP e com toda a *web* [15].

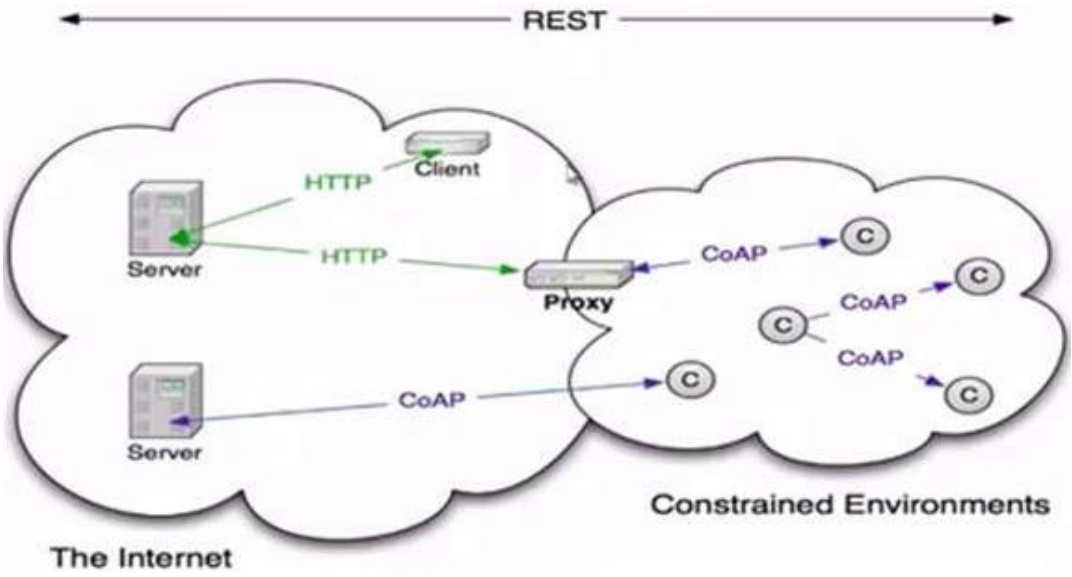

Figura 7 – Sistema de funcionamento do CoAP [18]

QoS

As mensagens dos pedidos e respostas podem ser marcadas como "confirmáveis" ou "não confirmáveis". As mensagens confirmáveis devem ser confirmadas pelo recetor com um pacote de *acknowledge*. As mensagens não confirmáveis são do tipo "*fire and forget*" [15].

Observe

O CoAP estende o modelo de pedidos do HTTP com a capacidade de observar um recurso. Quando a *flag* de *observe* é colocada num pedido GET, o servidor pode continuar a responder após ter sido enviada a resposta. Isto permite que os servidores transmitam as alterações de estado para os clientes, à medida que estas ocorrem. Qualquer extremidade pode cancelar a observação [15].

Segurança

Uma vez que o CoAP corre sobre UDP e não sobre TCP, o SSL/TLS não se encontra disponível para fornecer a camada de segurança, como acontece no caso do protocolo MQTT, visto anteriormente.

Assim sendo, o CoAP tira partido do *Datagram Transport Layer Security* (DTLS) para garantir a privacidade e a integridade das mensagens. O DTLS baseia-se no TLS com o objetivo de fornecer as mesmas garantias de segurança, mas aplicadas às transferências de dados sobre UDP. [19]

É oportuno referir que este protocolo não possui qualquer mecanismo de autenticação dos clientes.

Por omissão, o CoAP utiliza o porto 5683, mas, no caso da utilização de DTLS, o porto *standard* é o 5684. [20]

3.3 Outros protocolos de *messaging*

Para além do MQTT e do CoAP, essencialmente usados em dispositivos e sistemas domésticos associados à *Internet of Things*, existem outros protocolos de *messaging* também bastante utilizados neste emergente conceito, mas mais orientados para outro tipo de ambientes e setores de implementação, como o empresarial, o industrial e até o militar.

Apesar de estes protocolos não se enquadrarem diretamente com o principal foco e objetivo da realização deste projeto, acredita-se que a sua compreensão e conhecimento do seu mínimo funcionamento são cruciais para se adquirir uma melhor perceção sobre IoT, na sua generalidade, e sobre a forma como a segurança é ou não implementada nestes protocolos e nos equipamentos que os utilizam para comunicar.

Posto isto, pretende-se abordar alguns dos protocolos mais utilizados para este efeito: AMQP, DDS e XMPP.

3.3.1 *Advanced Message Queuing Protocol*

No *Advanced Message Queuing Protocol* (AMQP), é utilizado um mecanismo de *publish-subscribe*, onde existe um *broker* ou um espaço partilhado de *queues* (filas de espera) a que todas as aplicações cliente podem aceder. Os clientes enviam as mensagens através da ação de *publish* (publicação) para uma *exchange* dentro do *broker*. Cada mensagem contém uma chave de *routing* que será usada pelo *broker* para realizar as seguintes operações: atribuir a mensagem a uma fila (*queue*) específica; distribuir a mensagem por várias filas; duplicar cada mensagem para múltiplas filas ou distribuir várias mensagens para as filas escolhidas, conforme definido pela chave [21].

Um *subscriber* recebe, desta forma, as mensagens autorizadas das filas de espera para proceder à entrega das mesmas ao seu cliente final. Uma vez que cada mensagem é atribuída a um subscritor específico, mesmo que a mesma mensagem possa ter sido duplicada para outros, apenas um subscritor pode receber uma mensagem específica de qualquer uma das filas presentes.

As mensagens individuais são contabilizadas e "localizadas" (*tracked*) para garantir que todas são entregues tal como pretendido pelo *publisher*. Para que isso ocorra, todos os destinatários das mensagens publicadas devem confirmar a receção das mesmas.

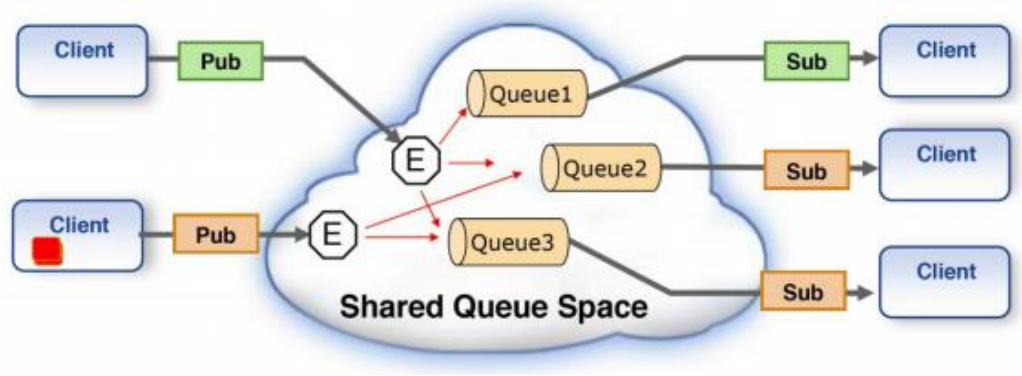

Figura 8 – Sistema de *publish-subscribe* de AMQP [22]

Relativamente à segurança implementada, o AMQP suporta autenticação dos clientes através da *framework* de *Simple Authentication and Security Layer* (SASL) e utiliza o TLS para assegurar a integridade e confidencialidade das comunicações. É importante referir que, contrariamente ao MQTT e ao CoAP, estas medidas de segurança encontram-se, geralmente, ativadas por omissão, reduzindo os potenciais riscos de segurança. [23]

O AMQP usa o porto 5671 em TCP e UDP para a utilização de TLS e DTLS, respetivamente, e o porto 5672 quando estes protocolos de segurança não são utilizados. [24]

3.3.2 *Data Distribution Service*

O *Data Distribution Service* (DDS) é um protocolo *standard* centrado nos dados, geralmente utilizado para gerir trocas de informação entre dispositivos de baixo poder computacional e redes de sensores de alto desempenho. A sua principal aplicação no universo da *Internet of Things* reside em ambientes industriais, como controlo de tráfego aéreo, veículos autónomos, serviços de transporte, etc. [25]

O DDS é descentralizado e implementa uma arquitetura de *publish-subscribe* para enviar e receber dados, eventos e comandos entre os nós, em tempo real. Os nós que produzem informações (*publishers*) criam "tópicos" (por exemplo, temperatura, local, pressão) e publicam "*samples*". O DDS entrega as *samples* aos *subscribers* que declaram interesse nesse tópico. As aplicações comunicam publicando e assinando tópicos identificados pelo nome. [26]

O DDS fornece *data-sharing* controlado por QoS. Os *subscribers* podem especificar filtros de tempo e conteúdo e obter apenas um subconjunto dos dados publicados no tópico. [26]

Realça-se que diferentes domínios de DDS são completamente isolados e independentes uns dos outros e não ocorrem trocas de dados entre domínios distintos.

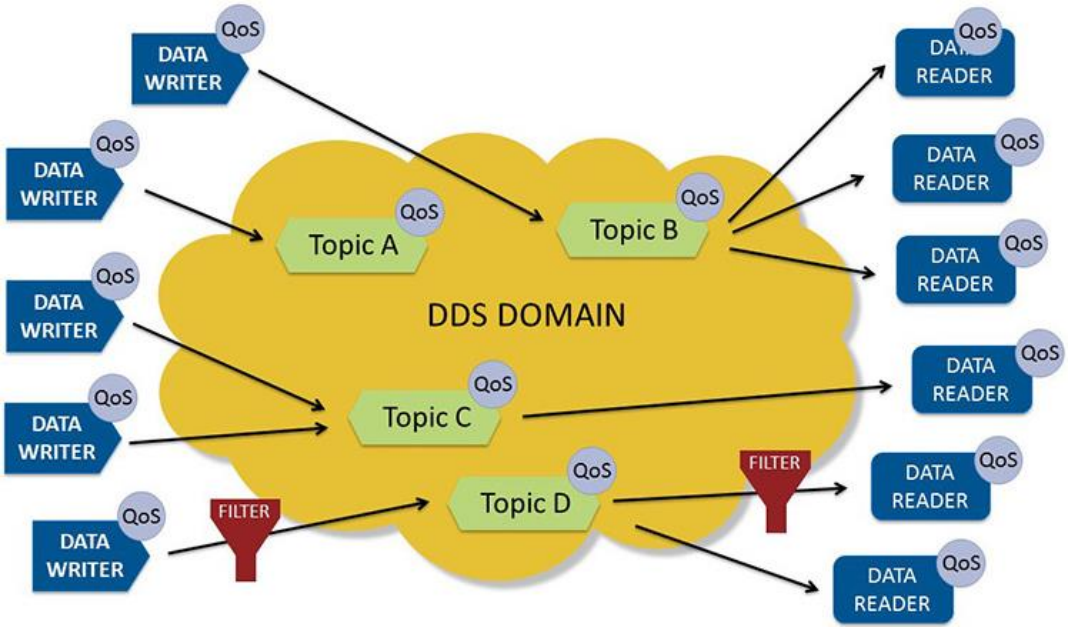

Figura 9 – Representação de um domínio DDS [26]

Relativamente à implementação de segurança, o protocolo DDS suporta quer TLS, quer DTLS. Para além disso, possui uma especificação de segurança própria que define uma arquitetura baseada em *plugins* para assegurar confidencialidade, integridade e autenticação. [27]

3.3.3 *Extensible Messaging and Presence Protocol*

O *Extensible Messaging and Presence Protocol* (XMPP) é um protocolo baseado em XML (*Extensible Markup Language*) para comunicações em tempo real entre duas ou mais entidades. [28]

Inicialmente, este protocolo foi desenvolvido para serviços de *instant messaging*, mas acabou por ser aproveitado para a implementação de sistemas que utilizem modelos de *publish-subscribe*, como aplicações associadas à *Internet of Things*, naturalmente. [29]

O XMPP garante robustez na segurança através do suporte de SASL para realizar autenticação e de TLS para garantir confidencialidade e integridade. Estes serviços estão assentes nas especificações do núcleo do protocolo e, por isso, estão ativados por omissão, tal como acontece no AMQP. [23]

Neste protocolo, os clientes e os servidores utilizam, respetivamente, os portos 5222 e 5269 de TCP. [24]

3.4 *Multicast Domain Name System*

O *Multicast Domain Name System* (mDNS) trata-se de um protocolo comummente utilizado em redes de pequena dimensão, onde não existem *name servers* locais (por exemplo, redes domésticas). O mDNS, juntamente com o DNS-SD (DNS-*based Service Discovery*) [30], tem a capacidade de realizar operações semelhantes ao serviço convencional de DNS e, desta forma, permitir a descoberta de serviços e a resolução de nomes no *link* local da rede. [31]

Ao contrário dos protocolos de *messaging*, o mDNS não providencia quaisquer serviços de segurança. Assim sendo, de forma semelhante ao DNS, este protocolo e os ambientes que o utilizam encontram-se expostos a ataques que explorem as suas falhas de segurança. Apesar de começarem a aparecer protocolos que procuram melhorar a segurança do DNS (por exemplo, DNSSEC [32], DNS *over* TLS [33] e DNS *over* HTTPS [34]), estes ainda se encontram numa fase bastante embrionária e são demasiado complexos para se aplicarem em ambientes *self-configuring*.

O mDNS utiliza o porto 5353 de UDP, mas, assim como acontece no DNS, também pode correr sobre TCP. Isto acontece quando uma mensagem de resposta *unicast* é demasiado grande para caber apenas num pacote [31]. Como endereço *multicast* de destino, são usados os endereços 224.0.0.251 e ff02::fb, em IPv4 e IPv6, respetivamente [35].

3.5 *Simple Service Discovery Protocol*

O *Simple Service Discovery Protocol* (SSDP) é um protocolo extremamente utilizado para a descoberta e o anúncio de serviços, sem a necessidade de recorrer a outros mecanismos baseados em servidores, tais como o DNS ou o DHCP. Este protocolo encontra-se incluído na arquitetura do UPnP (*Universal Plug-and-Play*) e a sua utilização é destinada a redes residenciais ou de pequenas empresas. [36]

No que toca à segurança, assim como o mDNS, o SSDP trata-se de um protocolo bastante fraco, uma vez que não dispõe de nenhum serviço ou mecanismo que a implemente. Assim sendo, existem diversos riscos de segurança associados a dispositivos que utilizem este protocolo.

Apesar da IANA ter alocado o porto 1900 de TCP e UDP [24], o SSDP utiliza apenas UDP. Como endereço de destino, são usados seguintes endereços *multicast* [36]:

- 239.255.255.250 (endereço IPv4 *site-local*);
- ff02::c (endereço IPv6 *link-local*);
- ff05::c (endereço IPv6 *site-local*);
- ff08::c (endereço IPv6 *organization-local*);
- ff0e::c (endereço IPv6 *global*).

3.6 Comparação

Posto isto, podem-se resumir as principais características dos protocolos acima expostos nos conteúdos da Tabela 1, de forma a realizar uma melhor e mais direta análise e comparação entre os mesmos.

Nota: Os círculos pretos simbolizam funcionalidades nativas dos protocolos, enquanto que os círculos sem fundo referem serviços adicionais que podem ser suportados pelos mesmos.

Protocol	Standard	Function		Architectural Model		Interaction Model		Transport Protocol	
		Messaging	Discovery	c/s	Decentralized	Pub/Sub	Req/Resp	TCP	UDP
MQTT	OASIS	●		●		●		●	
CoAP	IETF	●	○	●		○	●	○	●
AMQP	OASIS	●		●		●	○	●	
DDS	OMG	●	○		●	●	○	●	●
XMPP	IETF	●	○	●		●	●	●	
mDNS	IETF		●		●		●		●
SSDP	UPnP		●	●			●		●

Tabela 1 – Resumo das características dos protocolos [23]

Como se pode observar na Tabela 2, no que refere à segurança dos protocolos de *messaging* apresentados anteriormente, todos possuem mecanismos de encriptação, garantindo, deste modo, confidencialidade. Relativamente à autenticação, apenas o CoAP não suporta este serviço. Por fim, no que diz respeito à autorização, apenas o DDS e o XMPP foram desenvolvidos com esse objetivo.

Protocol	Authentication		Authorization	Confidentiality	
	SASL	Custom	Custom	TLS	DTLS
MQTT		●		●	
CoAP					●
AMQP	●			●	
DDS		●	●	●	●
XMPP	●		●	●	

Tabela 2 – Serviços de segurança dos protocolos de *messaging* [23]

O grande problema associado à implementação de segurança nestes protocolos de IoT, nomeada e principalmente, no MQTT e no CoAP, reside no defeituoso planeamento da segurança na estrutura do próprio protocolo. Contrariamente ao AMQP e ao XMPP, onde a utilização de TLS ocorre por omissão, estes protocolos obrigam a que sejam os *developers* a ativar explicitamente este serviço de confidencialidade, dado que o mesmo é considerado opcional. Como se não bastasse, os *developers* tendem a negligenciar o uso de TLS ou de DTLS na implementação e

configuração das suas aplicações. Consequentemente, os equipamentos de IoT ficam frequentemente expostos a falhas de segurança e arriscam suscetibilizar toda a rede a ataques provenientes de fontes mal-intencionadas.

Em suma, tendo de se escolher entre o MQTT e o CoAP para utilização em dispositivos IoT num âmbito doméstico, o MQTT será a melhor escolha, devido ao facto de realizar autenticação de clientes. Para além disso, a utilização de TCP também poderá representar uma vantagem sobre o CoAP (ou, em determinados casos, o UDP pode até ser mais vantajoso pela sua propriedade de *connectionless*). Contudo, este deve ser sempre utilizado sobre TLS para garantir a privacidade e a integridade dos dados!

Os restantes protocolos de *messaging* não possuem as características necessárias para serem candidatos válidos a uma implementação ao nível de um ambiente doméstico, dado serem mais exigentes computacionalmente e necessitarem de uma infraestrutura externa para assegurar o seu funcionamento.

3.7 Vulnerabilidades e Ataques

Após se ter percebido o funcionamento destes protocolos e a maneira como implementam segurança nas comunicações, é extremamente relevante perceber se existem falhas de segurança conhecidas e quais as razões para as potenciais ameaças existirem, expondo as boas práticas que devem ser utilizadas e as medidas a tomar para mitigar possíveis ataques.

Através da Figura 10, é possível constatar que, nos últimos três anos, têm sido reportados bastantes CVEs relativamente a estes protocolos de IoT, ainda que com algumas diferenças de números entre eles. Enquanto que, em 2015 e 2016, tinha ocorrido um decréscimo na frequência do aparecimento de vulnerabilidades relativamente a 2014, os anos de 2017, 2018 e 2019 mostram que houve uma subida bastante acentuada no número de casos reportados.

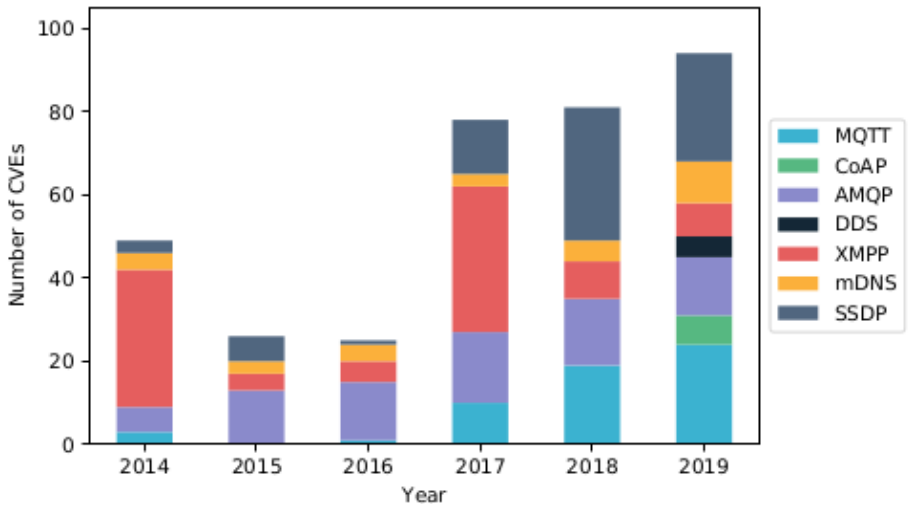

Figura 10 – Número de CVEs por ano e por protocolo [23]

Estes CVEs dizem respeito a diferentes tipos de vulnerabilidades associadas aos protocolos de IoT referidos e, dessa forma, podem ser categorizados e divididos consoante o tipo de ataque a que são vulneráveis. Assim sendo, a Tabela 3 apresenta, por protocolo, os principais ataques retratados pelos CVEs.

Protocol	Eavesdropping Attacks	IP Spoofing Attacks	DoS/DDoS Attacks	MiTM Attacks	Poisoning Attacks
MQTT			•	•	
CoAP		•	•	•	
AMQP			•		
DDS			•		
XMPP			•	•	
mDNS	•	•	•	•	•
SSDP	•	•	•	•	•

Tabela 3 – Principais tipos de ataques retratados em CVEs por protocolo [23]

Posto isto, pretende-se realizar uma abordagem mais profunda às vulnerabilidades e aos ataques que ocorrem nos protocolos MQTT, CoAP, mDNS e SSDP, dado que são aqueles que são mais utilizados em implementações de equipamentos associados à *Internet of Things* no âmbito doméstico.

3.5.1 MQTT

No caso do MQTT, já se viu que suporta cifra através do uso de TLS e autenticação dos clientes. Ainda assim, estes serviços não são suficientes para proteger os equipamentos que utilizam este protocolo para comunicar. Tal como demonstrado na DEFCON 24 [37], muitas falhas de segurança têm origem na má configuração dos *brokers* e em vulnerabilidades de *software*, tornando os dispositivos passíveis de serem explorados.

Desta forma, já está mais do que provado que os seguintes serviços do MQTT se podem apresentar vulneráveis [23]:

- Autenticação: nem sempre o *broker* verifica adequadamente a identidade dos clientes, nem bloqueia repetidas tentativas de autenticação. Estas falhas podem, naturalmente, permitir o acesso não autorizado de atacantes aos diversos dispositivos;
- Autorização: por vezes, o *broker* não define propriamente as permissões dos clientes relativamente às ações de *publish* e de *subscribe*, podendo permitir que atacantes ganhem controlo sobre dados ou funcionalidades dos equipamentos;
- Entrega de mensagens: a performance de um *broker* pode ser significativamente degradada através do envio de mensagens por um *publisher* mal-intencionado para um tópico sem *subscribers*, podendo, no pior cenário, causar negação de serviço;

- Validação de mensagens: quando os caracteres das mensagens não são devidamente validados, torna-se possível que um *publisher* envie mensagens com caracteres não expectáveis e, através disso, realizar ataques de injeção de código ou de negação de serviço, por exemplo;
- Geração de chaves: em determinadas implementações, a geração de chaves criptográficas não é suficientemente robusta e a sua utilização pode comprometer os equipamentos e as suas comunicações;
- Encriptação: sem o uso do TLS, os clientes e servidores trocam as mensagens em *plaintext* (inclusivamente, as credenciais), permitindo a realização de ataques de *Man-in-the-Middle* (MITM).

Como se observou anteriormente, existem vários CVEs (*Common Vulnerabilities and Exposures*) associados ao MQTT que mostram, de forma geral, a falta e a má implementação de segurança. Por exemplo, o CVE-2019-11779 [38] mostra que um atacante pode enviar uma mensagem de *subscribe* para um tópico cujo nome consista em 65400 ou mais carateres de '/' (*slash*) e causar um *buffer overflow*.

Da mesma forma, o CVE-2019-6241 [39] prova que o envio de uma mensagem de *connect*, seguida de uma mensagem malformada de *unsubscribe*, pode funcionar como um ataque de DoS (*Denial of Service*) contra o *broker*.

Relativamente à autenticação, por exemplo, o CVE-2017-7650 [40] retrata um cenário em que os clientes se podem autenticar através da utilização do carater '#' (cardinal) ou '+' (símbolo de somar) como *username*, contornando um mecanismo de autorização baseado em ACLs (*Access Control Lists*). Consequentemente, os atacantes podem ter acesso a todos os tópicos, tal como é mostrado na Figura 11.

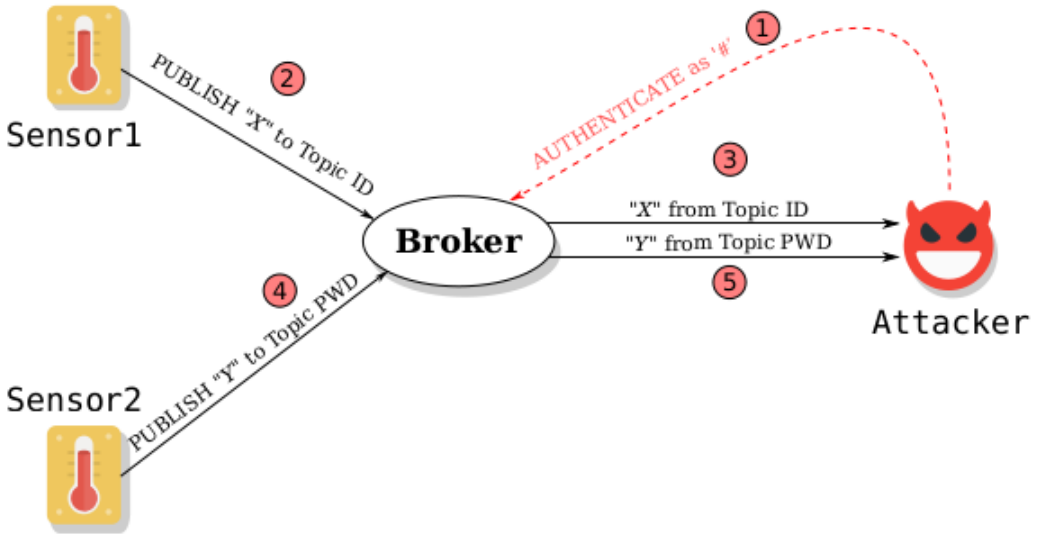

Figura 11 – Cenário do ataque descrito no CVE-2017-7650 [23]

Para mitigar estas falhas e ameaças à segurança dos equipamentos e das comunicações que realizam, o *standard* do MQTT recomenda alguns mecanismos que devem ser usados aquando da implementação do protocolo num dado serviço ou aplicação:

- Autenticação dos utilizadores e dispositivos;
- Controlo de acessos aos recursos;
- Integridade nos pacotes de controlo e de dados (utilização de TLS);
- Privacidade nos pacotes de controlo e de dados (utilização de TLS).

É o facto de alguns *developers* negligenciarem a integração destas medidas ou as configurarem erradamente nas suas aplicações e equipamentos que faz com que existam tantas falhas e problemas associados à segurança do MQTT.

Contudo, a utilização do TLS não garante totalmente a existência de comunicações seguras. Como é bem sabido, recorrer a versões antigas do TLS e a cifras consideradas inseguras também pode colocar em causa a confidencialidade das mensagens e expor o MQTT (ou, como é de realçar, qualquer outro protocolo que o utilize desta maneira) à possibilidade de ser alvo de ataques. [41]

Já existem propostas de versões mais seguras do MQTT que tiram partido da utilização de TLS com criptografia *lightweight* baseada em Curvas Elípticas, oferecendo maior segurança com menos *bits*. A par disso, também já surgiu uma proposta para melhorar o mecanismo de autenticação e autorização do protocolo através de uma arquitetura baseada numa versão modificada do OAuth. [23]

Em suma, pode-se afirmar que o MQTT suporta uma boa quantidade de serviços que implementam segurança. Contudo, o que acontece frequentemente é que esses acabam por não ser de todo utilizados nos produtos ou por serem mal configurados e, com isso, colocar o sistema propenso a riscos de segurança que afetem o protocolo.

3.5.2 CoAP

Relativamente ao CoAP, já se discutiu a sua utilização de DTLS e o facto de esta ser opcional e ficar ao critério dos *developers* a sua integração nas implementações que desenvolvem.

Desta forma, já está mais do que provado que os dispositivos que utilizam CoAP poderão estar suscetíveis às seguintes vulnerabilidades [23]:

- *Parsing* de mensagens: por vezes, as mensagens que chegam a um nó não são adequadamente tratadas e podem causar condições de *overload* ou até possibilitar a realização de ataques de *remote arbitrary code execution*.
- *Proxying and caching*: em determinadas situações, os mecanismos de controlo de acessos não são implementados apropriadamente, comprometendo a confidencialidade e integridade das mensagens;
- *Bootstrapping*: se o *setup* de novos nós (*nodes*) não for devidamente implementado, pode acontecer que nós não autorizados obtenham acesso indevido ao ambiente do CoAP;
- Geração de chaves: em determinadas implementações, a geração de chaves criptográficas não é suficientemente robusta e a sua utilização pode comprometer os nós e as suas comunicações;

- *Spoofing* de endereços IP: um atacante pode forjar os endereços IP dos nós e realizar diversos tipos de ataques, como, por exemplo, ataques de reflexão ou de amplificação.

Como foi possível verificar anteriormente, existem alguns CVEs que afetam produtos e serviços baseados no CoAP. Por exemplo, o CVE-2018-12679 [42] e o CVE-2018-12680 [43] incidem sobre o incorreto *parsing* das mensagens ou de certas exceções, levando a que ocorram negações de serviço nas aplicações. De forma semelhante, o CVE-2019-17212 [44] mostra que esse mesmo problema pode levar à ocorrência de *buffer overflows*, permitindo a injeção de código malicioso.

Também o protocolo de transporte UDP pode ser explorado como um vetor de ataque. Por exemplo, o CVE-2019-9750 [45] prova que é possível realizar ataques de DDoS (*Distributed* DoS) através de *spoofing* de endereços IP e de técnicas como *traffic amplification*.

Para mitigar estas falhas e ameaças à segurança dos equipamentos e das comunicações que realizam, o *standard* do CoAP recomenda dois mecanismos que devem ser usados aquando da implementação do protocolo num dado serviço ou aplicação:

- Controlo de acessos;
- Comunicações seguras (utilização de DTLS).

Tal como acontece no MQTT para a utilização do TLS, também o DTLS não garante totalmente a existência de comunicações seguras no CoAP. Como já foi referido, o uso de versões antigas e de cifras consideradas inseguras pode colocar em causa a confidencialidade das mensagens e, consequentemente, expor o protocolo e os dispositivos à possibilidade de serem alvo de ataques. [41]

A integração do DTLS no CoAP baseada em criptografia de Curvas Elípticas ajuda a minimizar o *overhead* nos equipamentos de baixo poder computacional e, ao mesmo tempo, oferece maior segurança (maior segurança com menos *bits*). [23]

Em suma, a utilização de DTLS no CoAP possibilita, de modo geral, comunicações seguras e confidenciais. Contudo, em ambientes de *Internet of Things*, as soluções *lightweight* integradas nos dispositivos de baixo poder computacional ficam sempre aquém das que são usadas em ambientes ditos “normais”, no que diz respeito à implementação de segurança.

3.5.3 mDNS

Como já se viu anteriormente, o mDNS é um protocolo bastante fraco do ponto de vista da segurança, dado que não implementa nenhum serviço ou mecanismo que a garanta.

Deste modo, é sabido que o mDNS pode ser alvo dos seguintes ataques [23]:

- Ataques de DoS: um atacante pode inundar os equipamentos que estejam à escuta do endereço de *multicast* 224.0.0.251 com mensagens que explorem

características específicas do protocolo e, assim, torná-los irresponsivos ou completamente indisponíveis;
- Ataques de *poisoning*: atacantes podem realizar *spoofing* de mensagens de resposta de mDNS e anunciar falsos serviços aos mesmos;
- Ataques remotos: um atacante pode tentar explorar os equipamentos que estejam à escuta do endereço de *multicast* 224.0.0.251, fazendo com que respondam a *queries* efetuadas a partir do exterior da rede local para realizar ataques de DoS ou, entre outros, ganhar acesso a informação ou dados sensíveis.

O CVE-2015-1892 [46], o CVE-2017-6519 [47] e o CVE-2017-6520 [48] provam que existem equipamentos que, inadvertidamente, respondem a *queries unicast* com endereços de origem que não são *link-local*, possibilitando a realização de diversos tipos de ataques (nomeadamente, DoS) ou o acesso indesejado a informação. De forma semelhante, o CVE-2015-0650 [49] mostra que é possível que atacantes remotos provoquem negação de serviço nas vítimas através do envio de pacotes mDNS intencionalmente malformados ou maliciosamente elaborados.

Para além disso, a natureza *multicast* das comunicações e a ausência de qualquer mecanismo de cifra pode conduzir ao levantamento de questões relativamente à segurança e à privacidade que, normalmente, permanecem indetetadas. Na realidade, algumas mensagens podem revelar informações pessoais ou sensíveis sobre os utilizadores dos equipamentos e sobre os serviços que disponibilizam. [23]

Dado que este protocolo é afetado por várias ameaças e não implementa nenhum mecanismo de segurança, a utilização de medidas de mitigação é de extrema importância. Assim, aconselham-se as seguintes simples medidas:

- Desativar os serviços de mDNS quando não são necessários;
- Bloquear a passagem de tráfego mDNS (UDP porto 5353) com origem ou destino fora do segmento de rede.

Algumas medidas mais sofisticadas podem consistir na implementação dos seguintes requisitos de segurança:

- Autenticidade: mensagens de *query* e de *response* assinadas pelo remetente para permitir que os destinatários verifiquem a sua identidade;
- Confidencialidade: mensagens de *query* e de *response* encriptadas para prevenir o *leak* indesejado de informação ou qualquer outro possível abuso do seu conteúdo.

Para concluir, a segurança (nomeadamente, no que diz respeito à privacidade e à autenticação) representa a principal falha existente no protocolo de mDNS e o maior desafio associado à sua utilização, incluindo, naturalmente, em ambientes de *Internet of Things*.

3.5.4 SSDP

Relativamente à segurança, tal como já foi referido anteriormente, o SSDP é um protocolo consideravelmente fraco. Os riscos associados a este protocolo estão, geralmente, relacionados com a exploração das características de descoberta de serviços e da sua natureza *multicast*.

Uma principal ameaça que afeta os equipamentos que usam SSDP consiste na realização de ataques DDoS, com o intuito de os tornarem irresponsivos ou totalmente indisponíveis. Estes ataques exploram as características do UDP e do SSDP enquanto protocolos, bem como a má configuração dos dispositivos.

Mais precisamente, como é mostrado na Figura 12, um atacante pode gerar mensagens de M-SEARCH *request* com um endereço de IP *spoofed* (isto é, um endereço de IP falso, pertencente a uma vítima) e enviá-las para equipamentos vulneráveis que se encontrem a correr SSDP. Estes, por sua vez, irão inundar a vítima com mensagens de *response*. [23]

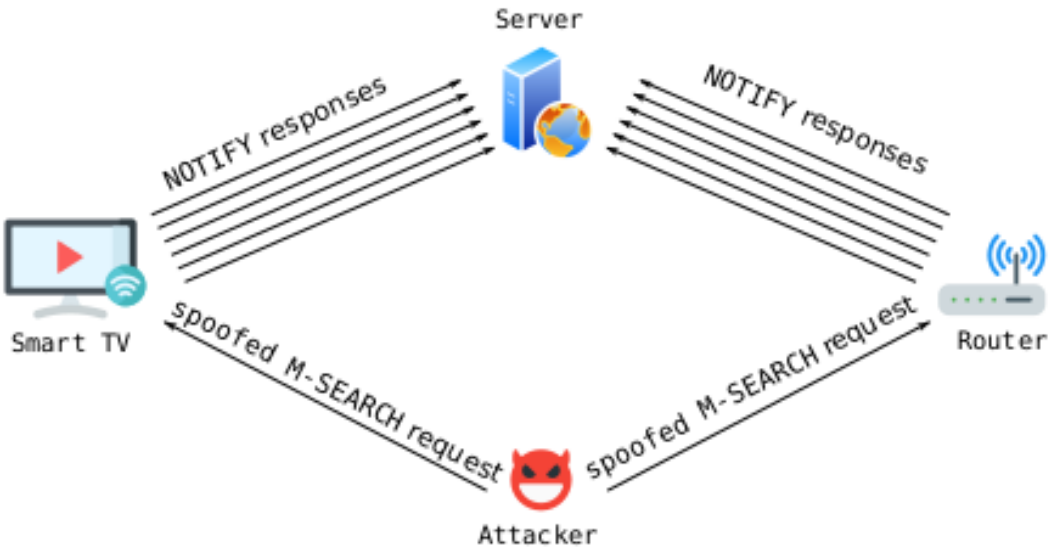

Figura 12 – Cenário de ataque DDoS ao SSDP [23]

Outra ameaça ocorrente em dispositivos que utilizam SSDP consiste em ataques passivos de *eavesdropping*, onde é possível obter as mensagens *multicast* trocadas na rede em *plaintext*. Desta forma, os atacantes poderão ganhar acesso a informação sensível e indesejada, conduzindo a sérias consequências do ponto de vista da privacidade e da confidencialidade. [23]

Para além disso, o CVE-2019-14323 [50] mostra que o SSDP pode ser vulnerável a um ataque de *buffer overflow* que provoca o *crash* do servidor e o CVE-2019-14363 [51] informa que esse mesmo tipo de ataque pode permitir a execução remota de *arbitrary code*.

Por outro lado, o CVE-2014-5406 [52] e o CVE-2015-4051 [53], por exemplo, referem falhas de segurança relacionadas com fracos mecanismos de autenticação e

autorização, permitindo que atacantes alterem configurações dos equipamentos ou os desliguem/reiniciem.

Posto isto, dado que o SSDP se trata se um protocolo extremamente inseguro, é crucial saber como mitigar eventuais ataques. Ainda que não seja uma tarefa fácil, devido às peculiaridades do SSDP, aconselham-se as seguintes medidas:

- Desativar os serviços de SSDP quando não são necessários, uma vez que, geralmente, se encontram ativados por omissão;
- Implementar mecanismos de cifra para garantir confidencialidade e integridade das mensagens nos serviços utilizados sobre este protocolo.

Para concluir, o SSDP, a par do mDNS, trata-se de um protocolo de *discovery* extremamente vulnerável e, por essa razão, a sua utilização requer ponderação e a implementação das devidas medidas de mitigação, principalmente ao nível da aplicação desenvolvida sobre este protocolo.

4 Estudo de equipamentos de IoT

Com a pesquisa dos protocolos terminada, procurou-se realizar um estudo sobre alguns equipamentos de IoT utilizados nas redes domésticas, principalmente, no que se refere aos portos e protocolos utilizados e na forma como estes efetuam as comunicações no interior da rede (LAN – *Local Area Network*) e para o exterior (WAN – *Wide Area Network*).

O objetivo desta segunda fase do projeto reside sobre a exploração de ferramentas de captura e análise de tráfego na rede, com o intuito de procurar perceber o funcionamento destes equipamentos, bem como o tráfego que por eles é gerado (quantidade de tráfego, destino do tráfego, etc.)

Para intercetar o tráfego entre dois equipamentos presentes na mesma rede (por exemplo, as comunicações entre um telemóvel e um dispositivo que este controla), tirou-se partido de uma técnica de ARP *Cache Poisoning*, através da ferramenta Ettercap. Para se proceder à captura e análise do tráfego, utilizou-se o reconhecido Wireshark. No Capítulo 6, estes procedimentos serão explicados de forma mais aprofundada e detalhada, dado que o intuito desta fase do relatório se prende com a realização de uma análise mais teórica sobre as observações e conclusões retiradas.

4.1 Interruptor SONOFF

O interruptor de estores é controlado através de uma aplicação móvel chamada eWeLink. Quando, na mesma, se efetua alguma ação (abrir ou fechar os estores), o telemóvel pode comunicar diretamente com o dispositivo ou pode fazê-lo através de um servidor remoto que, de seguida, comunica com o interruptor para anunciar a ação a realizar.

Quando o telemóvel que se encontra a correr a aplicação não está na mesma rede que o interruptor, eles comunicam por intermédio de um servidor remoto. O servidor consegue comunicar com o interruptor porque este estabelece uma ligação a cada 2 minutos (aproximadamente) e, por isso, o tráfego proveniente do servidor é considerado *established* quando atravessa a *firewall* do *router* doméstico. Todo o tráfego entre o interruptor e o servidor é feito sobre TLS (versão 1.2), encontrando-se, naturalmente, cifrado. Apesar de não ser possível ter acesso direto aos conteúdos que o TLS transporta, sabe-se que corresponde a tráfego HTTP. Isto porque o porto utilizado é o 443 (HTTPS) e a camada *Record* do TLS indica que o protocolo de *Application Data* é "HTTP-*over*-TLS".

Quando o telemóvel e o interruptor de estores se encontram na mesma rede, eles utilizam o protocolo mDNS (*multicast* DNS) para se descobrirem e, depois, comunicam diretamente (sem um servidor pelo meio a servir de *third party*) sobre TCP, através de HTTP. Contudo, também é enviada informação para o servidor

acerca do estado do interruptor (isto é, se os botões estão ou não ativos), aquando da ocorrência de qualquer ação que provoque uma mudança.

Nesta situação, apesar de os dispositivos não utilizarem nenhum protocolo de segurança no transporte (TLS, etc.), alguns dos dados trocados são cifrados. Por exemplo, as respostas do interruptor às *queries* de mDNS têm um campo de *data* que se encontra cifrado. A existência de um outro campo com um valor de IV (*Initialization Vector*) aponta para a possível utilização de cifra simétrica. Da mesma forma, as mensagens HTTP enviadas pelo telemóvel seguem exatamente o mesmo princípio.

Relativamente ao estabelecimento ou à troca da chave (ou chaves) utilizada, não foi possível apurar de que modo é que esse procedimento ocorre. Contudo, seria extremamente interessante averiguar se esta é trocada de forma assimétrica entre os equipamentos ou se se encontra *hardcoded* no código da aplicação e do interruptor de estores.

4.2 Tomada SONOFF

Visto que se tratam de produtos da mesma marca (SONOFF) e com uma arquitetura relativamente semelhante, a tomada funciona exatamente da mesma maneira que o interruptor de estores.

4.3 Lâmpada

A lâmpada inteligente é controlada através de uma aplicação móvel chamada SmartLife. Tal como nos dispositivos da SONOFF, quando, na mesma, se efetua alguma ação (acender, apagar, mudar a cor, etc.), o telemóvel pode comunicar diretamente com o dispositivo ou pode fazê-lo através de um servidor remoto que, de seguida, comunica com a lâmpada para anunciar a ação a realizar.

Quando a lâmpada e o telemóvel que a controla não se encontram na mesma rede, estes comunicam através de um servidor remoto. Para isso, a lâmpada utiliza o protocolo MQTT para comunicar com um *broker* (servidor). A conexão é estabelecida através de uma mensagem de CONNECT, definida no protocolo, e é utilizado um mecanismo de *keep-alive*, de 60 em 60 segundos, em que o dispositivo manda uma mensagem de PINGREQ ao *broker* que, de seguida, responde com um PINGRESP, terminando o processo com um ACK. O telemóvel, quando se efetua uma alteração no estado da lâmpada, comunica com o servidor para anunciar a ação desejada e este, por sua vez, envia essa ação à lâmpada com mensagens do tipo *publish-subscribe*.

Apesar de o MQTT ter suporte para correr diretamente sobre TLS (MQTT *over* TLS no porto 8883), a lâmpada utiliza o porto por omissão (porto 1883). Contudo, a

informação sobre a ação ou estado circula cifrada nas mensagens de *publish* e apenas é possível saber o identificador do dispositivo e o sentido da mensagem (*in*/*out*).

Quando o telemóvel onde se encontra a correr a aplicação que controla a lâmpada e esta se encontram na mesma rede, as atualizações de estado do dispositivo (como cor, luz e intensidade) são comunicadas diretamente entre eles e, depois, enviadas pela lâmpada ao *broker* através do mesmo tipo de mensagens (*publish-subscribe*). Para o dispositivo se anunciar, gera tráfego UDP *broadcast* com o seguinte conteúdo, que aparenta consistir num objeto JSON (esta mensagem é enviada de 3 em 3 segundos):

```
{"ip":"192.168.88.243","gwId":"032001806001947 3193d","active":2,
"ability":0,"mode":0,"encrypt":true,"productKey":"keycxv8dsyfegq
kw","version":"3.1"}
```

Quando se abre a aplicação no telemóvel, este estabelece uma ligação TCP com a lâmpada e este tráfego UDP *broadcast* deixa de ser enviado pela mesma. Assim, constata-se que este serve para que os dispositivos se descubram na rede. Desta forma, torna-se possível que comuniquem diretamente (sem uma *third party*) e, para isso, utilizam TCP, sendo que o porto usado pela lâmpada é o 6668.

Quando a aplicação é aberta e os equipamentos se descobrem, a lâmpada envia uma mensagem para informar o seu atual estado (se está ou não ligada, qual a cor da luz, etc.). Este conteúdo circula em claro (*plaintext*). Contudo, quando o telemóvel realiza uma ação sobre a lâmpada, os dados transportados pelo TCP encontram-se cifrados (remetendo para o valor *true* da *flag* de *encrypt* no tráfego UDP acima exposto).

Relativamente à chave (ou chaves) utilizada para cifrar o tráfego e ao mecanismo de cifra, não foi possível, com as capturas efetuadas, retirar qualquer conclusão.

4.4 Alexa da Amazon

Os equipamentos abordados até aqui consistem em dispositivos associado à *Internet of Things* com um extremamente reduzido poder computacional e que, individualmente, geram relativamente pouco tráfego na rede. Por outro lado, a Alexa tem um poder computacional consideravelmente superior e gera enormes quantidades de tráfego, dificultando a realização de uma análise detalhada sobre o mesmo e sobre o próprio equipamento. Contudo, foi possível perceber que este dispositivo realiza, entre outras coisas, bastantes *pings* periódicos ao *router* da LAN e a servidores remotos, aparentando funcionar como um mecanismo de *keepalive*.

Também se constatou que, apesar da Alexa utilizar TLS para várias comunicações (provavelmente, na maioria), existe tráfego que circula em claro. Nomeadamente, tráfego HTTP de pedidos realizados a *endpoints* de APIs. Para além disso, foi surpreendente a quantidade de pedidos de resolução de DNS que se observou que a Alexa realiza. Estes são feitos, maioritariamente, a domínios pertencentes à Amazon (AWS – Amazon *Web Services*).

Quando, por exemplo, se pede à Alexa para colocar uma música a tocar através de um comando de voz, surge, numa primeira instância, um pedido de resolução de DNS para o Spotify. De seguida, é feito um pedido GET à sua API para se obter a música em questão. Depois disto, a Alexa vai recebendo os diversos troços correspondentes ao conteúdo da música. É relevante realçar que este tráfego é HTTP e, por isso, circula em claro (*plaintext*). Ainda assim, notou-se que é utilizado um mecanismo de integridade dos dados, através de um MAC (*Message Authentication Code*) com chave: HMAC (*Hash-based* MAC).

4.5 Resumo

Equipamento	**Interruptor**	**Tomada**	**Lâmpada**	**Alexa**
Marca	SONOFF	SONOFF	- - - - - - - -	Amazon
Protocolos Utilizados	mDNS para *discovery*. HTTP para *messaging*.	mDNS para *discovery*. HTTP para *messaging*.	UDP *broadcast* para *discovery*. MQTT e TCP para *messaging*.	DNS, HTTP, HTTPS, ICMP, etc.
Portos Utilizados	As comunicações dentro da rede são feitas por HTTP e é usado o porto 8081. As comunicações para o exterior são feitas sobre TLS, do porto 3667 para o porto 443 (HTTPS).	As comunicações dentro da rede são feitas por HTTP e é usado o porto 8081. As comunicações para o exterior são feitas sobre TLS, do porto 3667 para o porto 443 (HTTPS).	O tráfego UDP *broadcast* usa 6666 como porto de destino. As comunicações na LAN usam o porto 6668. Para o exterior, as comunicações são feitas do porto 14873 para o porto 1883.	Múltiplos portos de origem e de destino.
Observações e Conclusões	- Utiliza *multicast* DNS para ser descoberto; - Utiliza HTTP para comunicar com o telemóvel na mesma rede e realiza cifra sobre um específico campo de dados; - Utiliza TLS (HTTPS) para comunicações com servidores fora da LAN; - Estabelece uma ligação para o servidor remoto a cada 2 minutos.	- Utiliza *multicast* DNS para ser descoberto; - Utiliza HTTP para comunicar com o telemóvel na mesma rede e realiza cifra sobre um específico campo de dados; - Utiliza TLS (HTTPS) para comunicações com servidores fora da LAN; - Estabelece uma ligação para o servidor remoto a cada 2 minutos.	- Não utiliza TLS; - Envia mensagens UDP *broadcast* a cada 3 segundos para se anunciar, até que a aplicação móvel seja aberta; - As comunicações na rede são feitas diretamente sobre TCP e as ações enviadas pelo telemóvel são cifradas; - O MQTT é usado nas comunicações para fora da LAN e utiliza um *keep-alive* com o *broker* de 60 em 60 segundos.	- Utiliza TLS na maioria das comunicações; - Realiza *pings* periódicos ao router da rede e a servidores remotos; - Efetua diversos pedidos de resolução de DNS; - Realiza pedidos HTTP a *endpoints* de APIs; - Nem sempre o tráfego é cifrado (caso do HTTP).

Tabela 4 – Resumo do estudo sobre os dispositivos de IoT

5 Funcionalidades dos *home routers* associadas à segurança

Nesta etapa da componente de preparação para o desenvolvimento do projeto, teve-se o intuito de estudar as funcionalidades dos *home routers* e das suas facilidades relativamente à segurança que implementam numa rede doméstica, principalmente, no que diz respeito ao *Internet of Things*.

Objetiva-se, com este estudo, explorar as capacidades dos *routers* dos ISP (*Internet Service Providers*) que permitem que os equipamentos associados ao IoT comuniquem de forma segura para fora da rede doméstica. Para além disso, pretende-se descobrir se existem limitações que possam colocar em causa a segurança destes dispositivos (e, consequentemente, da rede) ou dificultar a deteção de intrusões.

5.1 *Firewall* e NAT

A principal defesa contra intrusões que existe nos *routers* domésticos é a *firewall*. Apesar de serem algo limitadas e não permitirem um controlo absoluto por parte do utilizador, as *firewalls* presentes nos *routers* que os ISP instalam em casa dos clientes filtram o tráfego proveniente da Internet, implementando, com isso, alguma segurança nas redes locais.

Para além disso, as redes domésticas são redes privadas (geralmente, com endereços de rede do tipo 192.168.X.0/24) e, por isso, existe a necessidade de o *router* realizar operações de NAT (*Network Address Translation*) sobre o tráfego que entra e sai da rede. Consequentemente, os equipamentos que se encontram dentro da rede local não são diretamente acessíveis a partir da Internet, o que também representa uma vantagem do ponto de vista da segurança da rede.

Desta forma, equipamentos que se encontrem fora da rede doméstica e que necessitem comunicar com os dispositivos de IoT dentro da rede não o conseguem fazer, a menos que sejam estes últimos a iniciar as comunicações.

Naturalmente, isto levanta algumas questões relativamente ao funcionamento destes aparelhos tais como: “Como é que então é possível acender ou desligar uma lâmpada através do telemóvel estando-se fora da rede de casa?”

O que acontece é que os equipamentos de IoT estão constantemente a iniciar ou a manter comunicações abertas para os servidores da empresa que desenvolveu ou que dá suporte ao produto. Deste modo, quando se acende uma lâmpada através de uma aplicação no telemóvel, é enviada uma mensagem para um servidor que, por sua vez, a envia para a rede onde se encontra esse dispositivo. Uma vez que a comunicação foi iniciada pelo mesmo, o tráfego proveniente do servidor não é bloqueado pela *firewall* e consegue atingir o seu destinatário.

5.2 Possíveis portas de entrada

Contudo, os *routers* não têm apenas funcionalidades que protegem e oferecem segurança à rede. Existem diversas ferramentas que, quando usadas pelo utilizador "comum" ou por um técnico inexperiente, podem contornar a segurança implementada pelos *routers* e colocar a rede doméstica em risco e suscetível a intrusões indesejadas.

5.2.1 *Port Forwarding*

A funcionalidade de *Port Forwarding* permite ter um determinado serviço a correr na rede doméstica e disponibilizar o seu acesso a partir da Internet. Ao se fazer isto, está-se a permitir o acesso direto a um equipamento da rede.

Se esse equipamento não possuir as devidas ferramentas de segurança (por exemplo, *firewall*, antivírus, etc.) ou se o *software* do serviço que presta estiver desatualizado e tiver *exploits* conhecidos, esse equipamento poderá ser facilmente comprometido e deixar toda a rede vulnerável e à mercê do atacante.

Contudo, esta funcionalidade é bastante útil e, por isso, deve ser utilizada de forma doseada e consciente.

5.2.2 *Bridging*

A funcionalidade de *Bridging* oferece a possibilidade de se ter um equipamento fora da rede local. Ao ser ativada, uma das interfaces do *router* passa a possuir um endereço público (disponibilizado pelo ISP) e acessível a partir de qualquer ponto da Internet.

Apesar desta funcionalidade ter grandes vantagens, se não for utilizada de forma consciente e se o equipamento para o qual se está a realizar o *bridging* não estiver devidamente protegido e atualizado, poderá ser atacado com bastante facilidade.

5.2.3 DMZ *host*

Um DMZ *host* consiste num equipamento na rede interna para o qual o *router* encaminha todo o tráfego que habitualmente não encaminharia [54]. Desta forma, esse equipamento fica diretamente exposto à Internet e propenso a ataques.

Uma vez que esse *host* continua a ter acesso à rede local, caso o equipamento seja comprometido, todos os dispositivos da rede doméstica ficam expostos ao atacante que se conseguiu apoderar do DMZ *host*.

Esta funcionalidade é extramente desaconselhada e, possivelmente, é a mais vulnerável das três apresentadas. Infelizmente, é frequentemente utilizada em detrimento do *Port Forwarding* por ser mais simples e prática de configurar [55].

5.2.4 *Universal Plug and Play*

Por fim, existe uma outra funcionalidade ainda mais perigosa do que as acima mencionadas: *Universal Plug and Play* (UPnP).

O UPnP é uma funcionalidade suportada por praticamente todos os *routers* de cariz doméstico e que, infelizmente, se encontra ativada por omissão em grande parte dos *routers* colocados pelos ISPs nas casas dos clientes, em milhares de milhões de *routers* em todo o mundo. [56]

O UPnP consiste numa forma simples e conveniente de permitir que um equipamento da rede local abra portos no *router* e que o tráfego que chegue aos mesmos lhe seja direcionado. Funciona de forma relativamente semelhante ao *Port Forwarding*, mas de modo totalmente automático e sem que o utilizador autorize ou, sequer, tenha conhecimento disso. Desta forma, qualquer aplicação que se encontre a correr num computador ou qualquer dispositivo associado à *Internet of Things*, seja ele bastante simples ou mais complexo, pode realizar estes pedidos ao *router* e, com isso, passar a estar acessível a partir da Internet.

Tal como já foi referido, esta funcionalidade vem, muitas vezes, ativada por omissão para facilitar o trabalho dos utilizadores e fazer com que não tenham de inserir configurações manualmente. Contudo, esta conveniência tem um custo bastante elevado que se reflete nos inúmeros problemas de segurança que acarreta.

Sabe-se, por exemplo, que esta foi uma das funcionalidades exploradas pelo reconhecido *botnet malware* Mirai para infetar dispositivos de IoT que se encontravam em redes privadas e protegidos por *firewalls*. [56] [57]

6 *Pentesting* aos equipamentos de IoT

Chegando a esta fase do projeto, procurou-se realizar *Penetration Testing* aos equipamentos de IoT estudados e explorados anteriormente, seguindo o princípio de "procurar saber como atacar para se saber como defender".

Desta forma, os objetivos propostos para este capítulo consistem na tentativa de encontrar vulnerabilidades nos dispositivos de IoT, nos protocolos que estes utilizam e na exploração das mesmas, na eventualidade de existirem.

6.1 *Scanning* e *Enumeration*

Para isso, começou-se por utilizar a ferramenta NMAP [58] para fazer *scan* aos equipamentos e enumerar os portos que têm abertos.

Nos equipamentos da marca SONOFF (interruptor e tomada inteligentes), descobriu-se que apenas possuem o porto TCP 8081 aberto. Apesar do NMAP associar este porto a um serviço designado de *blackice-icecap*, descobriu-se na documentação do fabricante que se trata de um porto utilizado pela API (do tipo RESTful) que disponibilizam para *developers* [59], confirmando a análise que se realizou ao tráfego gerado e recebido por estes equipamentos no Capítulo 4 deste relatório. Para além disso, foi possível descobrir que o fabricante da placa de rede Wi-Fi é a empresa Espressif.

```
Nmap scan report for 192.168.88.251
Host is up (0.0011s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
8081/tcp open  blackice-icecap
MAC Address: CC:50:E3:15:E4:16 (Espressif)
```

Figura 13 – Relatório de *scan* da tomada da SONOFF

Na lâmpada inteligente, detetou-se que apenas o porto TCP 6668 se encontra aberto. Uma vez mais, o NMAP associou este porto a um serviço (neste caso, de IRC), mas acredita-se que será uma situação semelhante à dos equipamentos da SONOFF. Como se viu no Capítulo 4, este porto é utilizado pela lâmpada para realizar as comunicações diretamente com o telemóvel que o controla através da aplicação móvel, quando se encontram na mesma rede. Novamente, o fabricante da placa de rede Wi-Fi é a empresa Espressif.

```
Nmap scan report for 192.168.88.252
Host is up (0.0012s latency).
Not shown: 999 closed ports
PORT     STATE SERVICE
6668/tcp open  irc
MAC Address: 60:01:94:73:19:3D (Espressif)
```

Figura 14 – Relatório de *scan* da lâmpada inteligente

Por fim, na Alexa da Amazon, a ferramenta NMAP detetou os portos TCP 1080 e 8888. O primeiro está associado a um serviço de *proxy* e o segundo a um serviço de *web server*. Apesar de estes portos estarem abertos e de poderem não estar necessariamente associados a esses serviços, é algo absurdamente suspeito que a Alexa possua estes portos abertos e não tenha qualquer tipo de informação ou documentação sobre o assunto! Principalmente, no que diz respeito ao porto 1080 que, geralmente, é utilizado por servidores de SOCKS *proxy* [60].

```
Nmap scan report for 192.168.88.253
Host is up (0.0033s latency).
Not shown: 957 filtered ports, 41 closed ports
PORT     STATE SERVICE
1080/tcp open  socks
8888/tcp open  sun-answerbook
MAC Address: CC:F7:35:39:D5:26 (Amazon Technologies)
```

Figura 15 – Relatório de *scan* da Alexa

Para além disso, no Capítulo 4, capturaram-se pacotes gerados e recebidos por este equipamento durante algum tempo e perante determinadas ações e, nesse tráfego, estes portos nunca foram utilizados, o que ainda traz maior relevância para o quão suspeito e invulgar é o facto destes se encontrarem abertos sem que os utilizadores tenham qualquer acesso a motivos ou justificações por parte da Amazon.

É importante referir que, ao contrário da lâmpada, da tomada e do interruptor, a Alexa filtra a maioria dos portos testados (957 em 1000, para ser preciso). Enquanto que os restantes dispositivos tinham 1 porto aberto e os outros fechados, a Alexa apenas tem 41 portos fechados. A diferença entre um porto fechado e um porto filtrado (*firewalled*) reside na resposta dada pelo alvo. Se o porto estiver fechado, é enviado um TCP RST; se o porto estiver a ser filtrado, não é obtida qualquer resposta.

Scan Type →	TCP Full Connect Scan	TCP Stealth Scan	TCP Full Connect or Stealth Scan	TCP Full Connect or Stealth Scan
Status →	Host Up Port Up	Host Up Port Up	Host Up Port Down	Host Down or Host Firewalled
Example →	Client Server SYN → ← SYN ACK ACK → ← RST	Client Server SYN → ← SYN ACK RST →	Client Server SYN → ← RST	Client Server SYN →
Number of Packets →	4 packets	3 packets	2 packets	1 packet
Nmap Syntax →	nmap -sT	nmap -sS	nmap -sT or nmap -sS	nmap -sT or nmap -sS

Tabela 5 – Diferentes tipos de *scan* e deteção do estado dos portos

6.2 Ataque de *Replay*

Após se ter realizado o *scanning* dos diversos dispositivos de IoT e a respetiva enumeração, percebeu-se que estes não aparentam ser passíveis de ataques ao nível dos portos porque, ao contrário do que se poderia pensar, não possuem uma grande quantidade de portos abertos.

Desta forma, optou-se por tentar atacar os protocolos que estes dispositivos utilizam através de um ataque de *Replay*.

O ataque de *Replay* (também conhecido por ataque de *Playback*) consiste na captura de pacotes trocados entre dois equipamentos para, mais tarde, serem retransmitidos [61].

Deste modo, tentou-se realizar este ataque a dois dispositivos:

- Interruptor de estores da SONOFF;
- Lâmpada inteligente.

Para isso, começou-se por realizar um ataque de *Man-in-the-Middle* por ARP *Cache Poisoning* (ou, simplesmente, ARP *spoofing*) ao telemóvel e aos equipamentos enunciados. O ataque de ARP *spoofing* consiste no envio de mensagens *spoofed* de ARP para as vítimas, fazendo-se passar por cada uma delas, perante a outra. Desta forma, as vítimas irão associar o endereço MAC do atacante ao endereço IP da máquina para a qual querem comunicar e pela qual este se está a fazer passar. A Figura 16 representa o cenário deste ataque, tendo, como vítimas, um dispositivo de IoT e o telemóvel que o controla.

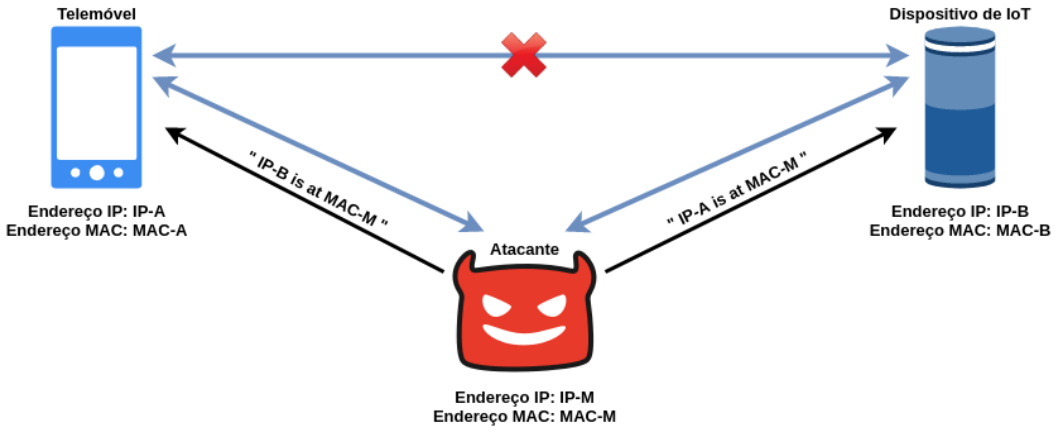

Figura 16 – Cenário de ataque MITM por ARP *Spoofing*

No primeiro caso, através deste ataque de ARP *Cache Poisoning*, capturou-se o tráfego enviado pelo telemóvel para o interruptor, aquando de uma ação de fechar os estores. De seguida, procedeu-se ao reenvio desses pacotes no computador (atacante). Contudo, o ataque não foi bem-sucedido e o interruptor não aceitou o comando de fechar os estores.

Para o segundo caso, fez-se algo bastante semelhante. Realizando este mesmo ataque, capturou-se o tráfego enviado do telemóvel para a lâmpada associado a uma

ação de acender e, depois, reenviou-se o mesmo pelo computador. Uma vez mais, o ataque não foi bem-sucedido e a lâmpada não acendeu.

Existem várias medidas de prevenção contra estes ataques (por exemplo, o número de sequência do TCP, a utilização de um *Session* ID, etc.) e, felizmente (ou infelizmente, para o propósito destes testes), estes dois equipamentos testados apresentaram-se invulneráveis a ataques triviais de *Replay*. Contudo, o universo de dispositivos de IoT é incontavelmente vasto e é impossível garantir que todos sejam invulneráveis a este tipo de ataque. Certamente, o mais provável é nem todos o serem. É igualmente possível, com mais tempo, afinar este tipo de ataques.

É importante referir que, para a realização deste ataque de *Replay* aos dois dispositivos, se utilizaram diferentes ferramentas para as diferentes fases envolvidas no processo:

- O Ettercap [62] foi utilizado para a realização de ARP *Spoofing*;
- A captura dos pacotes trocados entre as vítimas foi realizada através do Wireshark [63];
- Por fim, o reenvio (*replay*) dos pacotes capturados provenientes do telemóvel foi efetuado através da ferramenta Tcpreplay [64].

6.3 *Password Leaks*

Em 2019, houve um *leak* dos dados (*username*/*email* e *password*) de várias contas da aplicação eWeLink, que controla diversos dispositivos de várias marcas associados ao conceito de *Internet of Things*, nomeadamente, da SONOFF [65].

Este *leak* permitiu que atacantes pudessem fazer *login* na aplicação com os dados desses utilizadores e controlar os dispositivos das suas casas (por exemplo, apagar e acender luzes, subir e descer estores, ligar e desligar equipamentos de Ar Condicionado, etc.) e, caso os utilizadores tivessem o número de telemóvel associado à sua conta, ter acesso ao mesmo.

O mais grave é que esta aplicação também permite controlar *smart cameras* (pequenas câmaras de vigilância inteligentes) e, dessa forma, os atacantes com acesso a estes dados poderiam, se fosse o caso, observar as suas vítimas e o interior dos seus domicílios.

Esta vulnerabilidade não está associada aos equipamentos nem aos protocolos, mas sim às empresas e entidades nas quais os utilizadores confiam a sua segurança e a dos seus equipamentos. Apesar de, ao se ter acesso aos equipamentos das vítimas via aplicação móvel, não ser possível ter acesso à própria rede local, não deixa de ser extremamente preocupante que exista a possibilidade de alguém poder controlar dispositivos de IoT alheios.

Para além disso, *leaks* deste género podem permitir que atacantes consigam entrar nas contas de outros serviços das vítimas, caso estas reutilizem as suas *passwords* em diferentes *websites* e aplicações.

6.4 Ataques aos *chipsets* de Wi-Fi da Espressif

Nos dias de hoje, os *chipsets* de Wi-Fi produzidos pela empresa Espressif foram adotados mundialmente e são extremamente utilizados, praticamente monopolizando o mercado dos dispositivos de baixo poder computacional que se ligam à Internet.

Estima-se que cerca de 40% a 50% dos milhares de milhões de dispositivos associados a *Internet of Things* em todo o mundo utilizem o *chipset* ESP32 ou o *chipset* ESP8266. Estes são uma opção muito escolhida para o desenvolvimento deste tipo de equipamentos, dado o seu baixo valor monetário, o seu tamanho reduzido e o seu relativamente bom desempenho. Desta forma, estas características representam a principal razão pela qual a escolha recai sobre estes dois *chipsets* da Espressif.

Contudo, é fundamental realçar que existem vulnerabilidades e ataques conhecidos a estes dispositivos de *hardware* que afetam qualquer equipamento que os utilize. Dada a imensamente larga escala da sua utilização, é crucial referir a sua existência, bem como o tipo de *exploit* que pode ser realizado.

Por exemplo, o CVE-2018-18558 [66] enuncia uma vulnerabilidade existente no *bootloader* que permite que um atacante fisicamente próximo do dispositivo execute *arbitrary code*, através do desenvolvimento de uma aplicação que faça *overwrite* do código do *bootloader*.

O CVE-2019-12586 [67] e o CVE-2019-12587 [68] referem falhas na implementação do EAP (*Extensible Authentication Protocol*). A primeira permite que um atacante no alcance rádio do equipamento cause negação de serviço através do envio de uma mensagem maliciosamente elaborada. A segunda possibilita que um atacante em alcance rádio do equipamento faça *replay*, *decrypt* ou *spoof* de tramas, através de um *rogue* AP (*Access Point*).

Ao nível da implementação *Wireless* (IEEE 802.11), também já se sabe que existem *bugs*. O CVE-2019-12588 [69] mostra uma vulnerabilidade que permite que um atacante no alcance rádio do equipamento cause o *crash* do mesmo, através do envio de uma mensagem maliciosamente elaborada, provocando, assim, a negação de serviço.

O CVE-2019-15894 [70] alerta para uma vulnerabilidade de *fault injection.* Através desta, um atacante com acesso físico ao equipamento pode executar *arbitrary code* e, por exemplo, introduzir um *backdoor* que permita realizar o acesso remoto ao mesmo. [71]

Da mesma forma, o CVE-2019-17391 [72] aponta para uma vulnerabilidade que possibilita que um atacante com acesso físico ao dispositivo obtenha a chave da *flash encryption* e a chave do *secure boot*. Com a primeira, consegue ter acesso aos dados guardados na *flash* e ao *firmware*. Com a segunda, é possível modificar o código do *bootloader*. [73]

É mais do que evidente que existem diversas e diferentes vulnerabilidades associadas a estes *chipsets* e, possivelmente, muitas outras ainda por se descobrir. Desta forma, é essencial tentar reduzir ao máximo o impacto destas falhas de segurança e dos riscos que representam. Para isso, devem ser tomadas medidas de prevenção e mitigação por parte dos *developers* de equipamentos e aplicações e por parte dos próprios utilizadores.

No caso dos *developers*, é crucial que estes se mantenham devidamente informados e atualizados relativamente ao surgimento de novas vulnerabilidades e *exploits* para que possam implementar medidas de segurança e atualizar as suas aplicações, de acordo com as recomendações da própria Espressif.

Os utilizadores, por outro lado, devem procurar manter sempre as suas aplicações móveis atualizadas com a versão mais recente de *software* e, ainda mais importante, atualizar o *firmware* dos equipamentos assim que sejam lançadas novas atualizações.

6.5 Outros vetores de ataque

Ao longo deste capítulo, tem-se vindo a discutir e a abordar diferentes maneiras de explorar possíveis vulnerabilidades existentes nos dispositivos associados à *Internet of Things* com o intuito de, depois destes serem invadidos, se poder ter acesso à rede doméstica em que se encontram e, a partir daí, se poder explorar os restantes equipamentos a que ela se encontram ligados (isto é, computadores, telemóveis, etc.).

No entanto, existem outros vetores de ataque possíveis que também permitem, de alguma forma, que estes dispositivos sejam explorados por indivíduos mal-intencionados.

Ao invés de se obter acesso à rede a partir da invasão a um dispositivo de IoT, é possível que se invada, primeiro, a própria rede doméstica e, depois, se ganhe acesso e controlo sobre os equipamentos a ela ligados.

A título de exemplo, um atacante pode invadir a rede Wi-Fi de uma vítima, através do *crack* da palavra-chave utilizada na mesma. Este processo pode ser muito moroso ou, pelo contrário, pode ser relativamente simples, caso sejam usadas palavras-chave extremamente comuns ou de dicionário ou protocolos de baixa segurança (WEP e WPS, por exemplo). Depois de adquirir acesso à rede, o atacante fica em contacto "direto" com os equipamentos da vítima, nomeadamente, os associados a IoT. Nesta fase, o atacante pode tirar partido desta invasão para procurar explorar potenciais falhas de segurança nestes equipamentos e, tal como se tem vindo a descrever ao longo deste relatório, elas são bastante frequentes.

Ao se ganhar acesso aos dispositivos inteligentes da casa da vítima, podem-se realizar ações mais triviais como acender/apagar as luzes, descer/baixar os estores, ligar/desligar os aparelhos de Ar Condicionado, alterar alarmes, etc., ou, entre

outras coisas, causar uma simples negação de serviço. Contudo, estes acessos não autorizados podem possibilitar ao atacante realizar ações comparativamente mais graves.

Por exemplo, no limite, uma Alexa (ou outro dispositivo de semelhante utilização) pode ser explorada para garantir acesso ao atacante a tudo o que ela "ouve", funcionando como uma escuta, sem que os utilizadores possam sequer desconfiar dessa enorme invasão de privacidade. Da mesma forma, uma câmara de CCTV pode ser explorada de modo relativamente semelhante para garantir ao atacante o acesso a vídeo em vez de som/voz. O limite reside na imaginação e nas próprias funcionalidades suportadas pelos equipamentos inteligentes espalhados pelas casas de famílias em todo o mundo.

7 Desenvolvimento de uma aplicação de NIDS para IoT doméstico

Por fim, tendo-se dado por terminada a componente de estudo e análise dos protocolos e equipamentos associados ao conceito de *Internet of Things*, bem como das vulnerabilidades e ataques associados à sua utilização, partiu-se para a fase final da realização deste Projeto: implementação da aplicação de *Network-based Intrusion Detection System* para IoT, num âmbito doméstico.

Ao longo desta etapa, foi necessário planear o desenvolvimento do código associado à aplicação de policiamento da rede, projetar as diversas funcionalidades que se planeavam implementar e tomar múltiplas e diferentes decisões ao longo do caminho, sobre a forma como essas seriam elaboradas, com que recursos e através de que ferramentas já existentes.

Assim sendo, o principal objetivo deste capítulo prende-se com a exposição transparente e suficientemente aprofundada das várias funcionalidades desenvolvidas na aplicação, referindo, sobretudo, qual o intuito das mesmas e de que forma foram implementadas. Para além disso, pretende-se apresentar uma justificação para cada uma das decisões que foram tomadas durante este trajeto, nomeadamente, na escolha da plataforma de *harware*, da linguagem de programação e do Sistema Operativo e na preferência por determinadas ferramentas de *software* utilizadas em detrimento de outras alternativas.

7.1 Raspberry Pi

A escolha sobre a plataforma de *hardware* sobre a qual se desenvolveu a aplicação recaiu, de forma natural, sobre a Raspberry Pi [74]. O seu reduzido tamanho, a sua capacidade de processamento, o seu baixo custo, a sua portabilidade e a sua flexibilidade foram os principais fatores que pesaram nesta decisão. Quando se começou a idealizar este projeto, a Raspberry Pi integrou-se automaticamente nos requisitos e características que se pretendiam implementar.

Tal como foi enunciado no início deste relatório, pretende-se desenvolver uma ferramenta do tipo *plug & play*, que evite que o utilizador final tenha de lidar com configurações técnicas que pode não dominar. A Raspberry Pi permite exatamente isso! O objetivo é que o utilizador apenas tenha de ligar este dispositivo ao seu *router* de casa e permitir o acesso à sua rede doméstica Wi-Fi para que este fique totalmente operacional. A simplicidade e a comodidade associadas a este processo trivial representam uma enorme vantagem.

A Raspberry Pi consiste num pequeno computador de arquitetura ARM produzido, no Reino Unido, por uma organização sem fins lucrativos chamada Raspberry Pi Foundation. Este equipamento tornou-se rapidamente popular devido ao facto de

ser consideravelmente potente para o seu reduzido tamanho. Por esta razão, este dispositivo tem vindo a ser cada vez mais adotado em diferentes áreas de Informática para a realização de estudos e de projetos.

Naturalmente, a Raspberry Pi tem evoluído cada vez mais nos últimos anos e, por isso, existem diversos modelos que satisfazem as várias necessidades ao nível do preço, tamanho e poder computacional. No caso específico deste projeto, adotou-se o modelo 4 B, presente na Figura 17, com 4GB de RAM.

(Nota: À data do início da realização deste projeto, este era o melhor modelo existente no mercado. Contudo, foi recentemente lançado um novo modelo de 8GB de RAM que seria ainda mais adequado.)

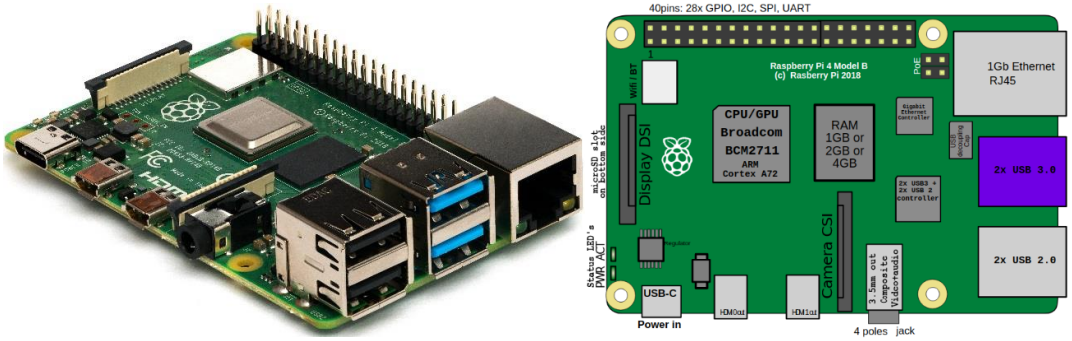

Figura 17 – Raspberry Pi 4 Model B [75]

7.2 Linguagem Python

Para o desenvolvimento desta aplicação de *Network-based Intrusion Detection System*, a escolha na linguagem de programação a utilizar recaiu sobre Python. Isto deve-se, não só ao facto de ser uma linguagem altamente atrativa e poderosa, mas também por ser extremamente utilizada em ferramentas de *networking* e de *pentesting*, existindo um conjunto de bibliotecas especializadas com elevado potencial nesta área.

A linguagem Python é recorrentemente associada ao desenvolvimento deste tipo de programas, devido à sua grande versatilidade e, principalmente, ao facto de existirem boas bibliotecas (por exemplo, a Scapy) que facilitam o trabalho aos *developers*, adicionando uma camada de abstração e permitindo que programem a um nível mais alto.

É uma linguagem que, infelizmente, não consta no Plano Curricular do curso de Licenciatura em Engenharia Informática, Redes e Telecomunicações e, por isso, nunca foi abordada ao longo do nosso percurso académico. Nesse sentido, considerou-se que seria bastante benéfico e uma excelente oportunidade aproveitar o Projeto para utilizar e aprender uma nova linguagem de programação, que sempre nos suscitou curiosidade e um especial interesse.

Apesar de ter várias vantagens, a linguagem Python pode, por vezes, deixar um pouco a desejar no que toca ao seu desempenho. Devido ao facto de ser uma linguagem de alto nível, não consegue ter uma eficiência tão elevada quanto outras linguagens de nível mais baixo, como, por exemplo, o C. Ainda assim, entendeu-se que, para o âmbito deste projeto, se trata de uma escolha sólida e totalmente capaz de realizar as operações necessárias num espaço de tempo aceitável.

7.3 Sistema Operativo

Relativamente ao Sistema Operativo a instalar e utilizar na Raspberry Pi e sobre o qual se vai desenvolver e correr a aplicação de policiamento da rede, a escolha incidiu sobre o Kali Linux [76], da Offensive Security. Esta distribuição de Linux é orientada para *Penetration Testing* e *Ethical Hacking* e é uma das favoritas e das mais utilizadas pela comunidade interessada nestes assuntos da cibersegurança.

O motivo pelo qual se optou por este Sistema Operativo reside no facto de já possuir, de base, algumas das ferramentas que se pretendem explorar e integrar na aplicação. Para além disso, funciona, por omissão, com o utilizador *root* e isso permite que não se tenha de lidar com a falta de permissões suficientes para realizar certas operações ou executar determinado *software*.

Alternativamente, poder-se-ia utilizar outro Sistema Operativo semelhante ao Kali (por exemplo, Parrot OS) ou qualquer outra distribuição de Linux. Desta forma, optou-se pelo Kali Linux por três principais razões:

- Boa familiarização com o Sistema Operativo;
- Possui, de base, várias ferramentas instaladas que se tencionam explorar;
- É disponibilizada uma versão para a Raspberry Pi que é *lightweight* e especificamente desenvolvida para esta plataforma.

7.4 Arquitetura da aplicação

Por fim, chega-se finalmente à última etapa deste Projeto Final de Licenciatura: a fase, propriamente dita, da implementação e do desenvolvimento da aplicação de *Network-based Intrusion Detection System* de equipamentos associados ao conceito de *Internet of Things* em redes domésticas.

Ao longo deste subcapítulo, pretende-se expor da melhor maneira possível a arquitetura da aplicação desenvolvida, bem como explicar de forma relativamente pormenorizada os módulos que a constituem e as principais funcionalidades que implementam.

Nesse sentido, o diagrama apresentado na Figura 18 representa um esboço da aplicação desenvolvida, subdividindo as suas tarefas em diferentes módulos, que permitem uma melhor organização e compreensão do código elaborado. Posto isto, segue-se uma concisa explicação sobre cada um dos módulos que integram esta

aplicação, bem como uma justificação para cada uma das decisões tomadas ao longo do processo do seu desenvolvimento.

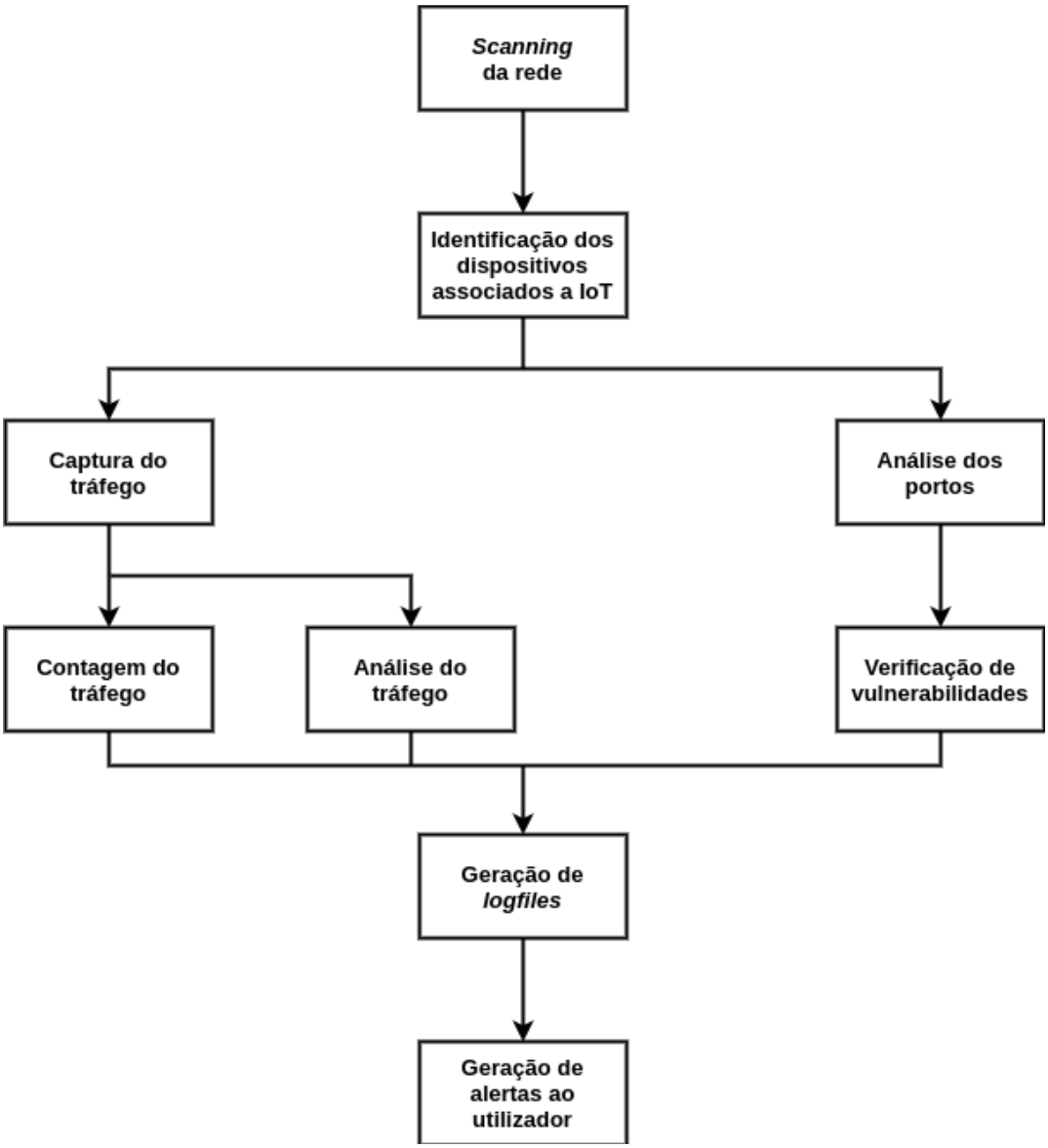

Figura 18 – Diagrama de representação da aplicação de NIDS

7.4.1 *Scanning* da rede

Com o intuito de se tomar conhecimento sobre a rede, o primeiro passo da aplicação de policiamento deverá passar por realizar um *scan* ativo da mesma. Com isto, pretende-se descobrir e identificar todos os *hosts* que se encontrem ligados à rede local, seja por Wi-Fi ou Ethernet.

Para isso, este módulo começa por obter o endereço da rede a que a Raspberry Pi se encontra ligada e a respetiva máscara. De seguida, realiza o *scanning* de todos os endereços da rede (à exceção, naturalmente, dos endereços de rede e *broadcast*)

para descobrir os *hosts* e, com a informação recolhida, gera um ficheiro JSON. Este ficheiro encontra-se organizado por dispositivo encontrado e armazena, para cada um, o seu endereço IP, endereço MAC e fabricante da placa de rede (obtido pelo Nmap).

Para implementar a funcionalidade de *scanning* da rede, tirou-se partido da ferramenta Nmap. Esta ferramenta foi preferida relativamente a outras alternativas pelas seguintes razões:

- Boa familiarização com a ferramenta em questão;
- Vem instalada de origem com o Sistema Operativo adotado;
- Possui uma interface de linha de comandos concisa e multifuncional (algumas alternativas apenas possuem interface gráfica);
- Permite realizar *host discovery* sem se ter de fazer *scanning* aos portos dos equipamentos (algumas alternativas não disponibilizam esta opção).

Posto isto, utilizou-se o seguinte comando na execução do Nmap:

```
nmap -sn -n <range>
```

Onde as opções utilizadas têm o seguinte significado:

- `-sn`: especifica que se pretende apenas realizar *host discovery* e desativa o *port scaning*;
- `-n`: especifica que não se pretende realizar resoluções de DNS;
- `<range>`: corresponde à gama de endereços IP que se pretende que seja utilizada no *scan* (por exemplo, 192.168.24.1-254).

7.4.2 Identificação dos dispositivos associados a IoT

Seguindo o *runtime* da aplicação, após a conclusão do módulo anterior, fica-se perante um ficheiro JSON com os dados adquiridos relativamente a cada *host* presente na rede. Assim sendo, é agora necessário identificar os dispositivos que estão associados ao conceito de *Internet of Things* e separá-los dos restantes equipamentos que se encontram na rede local.

Para atingir este objetivo, recorreu-se à elaboração de um ficheiro de texto com conhecidos fabricantes de placas de rede usadas neste tipo de dispositivos. Esta lista conta com perto de 15 destas principais empresas, como, por exemplo, a Amazon e a Espressif, que já foram referidas ao longo deste relatório. No caso da última sabe-se, como já foi mencionado, que é responsável pelas placas de rede utilizadas por uma grande percentagem (algures entre 40% e 50%) destes equipamentos.

Desta forma, este módulo da aplicação gera um novo ficheiro de texto com os dispositivos que tinham sido descobertos anteriormente e cuja placa de rede foi produzida por um dos fabricantes presentes nesta lista. O ficheiro é, então, gerado com o seguinte formato:

```
<EndIP_Dispositivo_IoT_1> <EndMAC_Dispositivo_IoT_1> -
<EndIP_Dispositivo_IoT_2> <EndMAC_Dispositivo_IoT_2> -
```

```
<EndIP_Dispositivo_IoT_3> <EndMAC_Dispositivo_IoT_3> -
...
<EndIP_Dispositivo_IoT_n> <EndMAC_Dispositivo_IoT_n> -
<EndIP_Router> <EndMAC_Router> -
```

No módulo seguinte, entender-se-á o motivo desta formatação.

Apesar de se ter adotado esta implementação para este módulo, tem-se plena consciência de que ela apresenta algumas falhas. Nomeadamente, o facto de ocorrer a possibilidade de existirem fabricantes que produzem placas de rede para equipamentos associados e não associados a IoT. Desta forma, a introdução de *Machine Learning* para realizar a identificação dos dispositivos de IoT constitui um dos tópicos presentes nos desejos de trabalho futuro, apresentados no Capítulo 8.

7.4.3 Captura do tráfego

Neste módulo da aplicação, é necessário fazer com que o tráfego que circula pelos equipamentos identificados passe pela Raspberry Pi para que, nos módulos que se seguem, seja possível realizar a contagem e a análise do mesmo.

Naturalmente, em condições normais, as comunicações que estes dispositivos realizam para o exterior da rede passam pelo *router* (que funciona como um *default gateway*) e nunca pela Raspberry Pi. Contudo, é necessário forçar a que isso aconteça.

Para esse efeito, realiza-se um ataque de *Man-in-the-Middle*, através da técnica de ARP *Cache Poisoning*, já abordada no Capítulo 6. Para isso, voltou-se a utilizar a ferramenta Ettercap. Desta forma, a Raspberry Pi vai-se fazer passar pelo *router* perante os dispositivos de IoT e vice-versa. A Figura 19 apresenta o cenário deste ataque MITM.

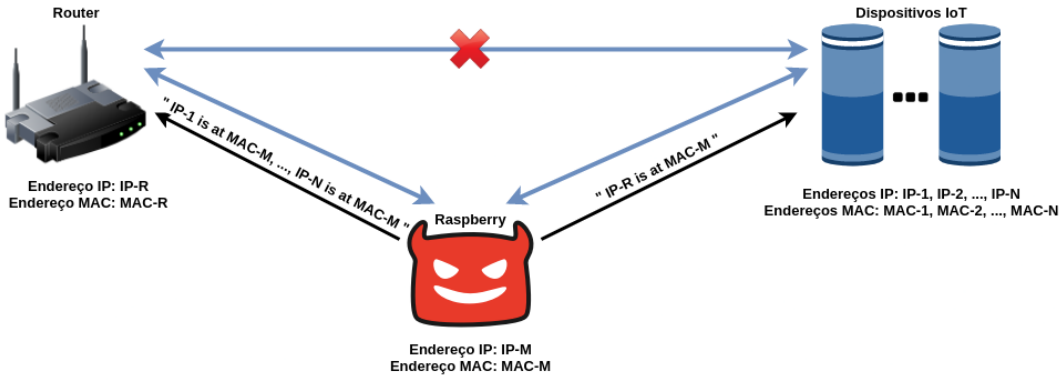

Figura 19 – Cenário de ARP *Spoofing* na aplicação de NIDS na Raspberry Pi

Posto isto, utilizou-se o seguinte comando na execução do Ettercap:

```
ettercap -i wlan0 -T -q -M arp:remote -j <hosts_file>
```

Onde as opções utilizadas têm o seguinte significado:

- `-i wlan0`: especifica que a interface a utilizar é a de Wi-Fi;
- `-T`: indica que se quer utilizar a interface de linha de comandos;
- `-q`: especifica que se pretende que não seja gerado qualquer *output* para o terminal dos pacotes capturados;
- `-M arp:remote`: indica que se pretende realizar um ataque MITM de ARP *Spoofing*;
- `-j <hosts_file>`: serve para indicar os *hosts* que vão servir de vítimas.

O ficheiro utilizado para este efeito é aquele que é gerado pelo módulo que foi explicado no ponto 7.4.2 (identificação dos dispositivos de IoT). A formatação utilizada corresponde àquela que é suportada e exigida pelo Ettercap.

Apesar de se ter optado por esta estratégia para se ter acesso ao tráfego que passa pelos dispositivos de IoT na rede doméstica, existem outras possibilidades que foram, de igual forma, analisadas e devidamente ponderadas. Assim sendo, a Tabela 6 apresenta todas as opções tidas em conta aquando da implementação deste módulo, bem como as suas vantagens, desvantagens e potenciais motivos pelos quais foram descartadas.

Técnica	Vantagens	Desvantagens	Motivos para exclusão
DHCP *Spoofing*	- Não exige que o utilizador adicione configurações ou faça alterações nos seus equipamentos (*plug & play*).	- É uma técnica um bocado "invasiva", ainda que o seu propósito não seja malicioso.	- Esta técnica foi alvo de testes e mostrou-se algo suscetível a falhas; - A técnica de ARP *Spoofing* é um pouco semelhante e dá mais garantias de funcionamento.
ARP *Spoofing*	- Não exige que o utilizador adicione configurações ou faça alterações nos seus equipamentos (*plug & play*).	- É uma técnica um bocado "invasiva", ainda que o seu propósito não seja malicioso.	- - - - - - - - - - - - - -
Port Forwarding	- Não é invasivo;	- Não permite que a aplicação seja do tipo *plug & play*; - Geralmente, os *routers* fornecidos pelos ISPs aos clientes domésticos não possuem essa funcionalidade.	- Esta estratégia vai contra aquele que é um dos principais princípios da aplicação: ser do tipo *plug & play*; - Não é aplicável, dado que a grande maioria dos *routers* das redes alvo não suportam esta técnica.

Nova rede Wi-Fi implementada na Raspberry Pi, apenas para dispositivos de IoT	- Não é invasivo; - Deixa de existir a necessidade de fazer a distinção entre dispositivos IoT e equipamentos "normais".	- Não permite que a aplicação seja do tipo *plug & play*, dado que o utilizador tem de reconfigurar os dispositivos de IoT.	- Esta estratégia vai contra aquele que é um dos principais princípios da aplicação: ser do tipo *plug & play*.
Apanhar o *handshake* dos equipamentos de IoT com o *router* e decifrar todo o tráfego que geram e que recebem	- Não é invasivo; - Não exige que o utilizador adicione configurações ou faça alterações nos seus equipamentos (*plug & play*).	- Não permite fazer análise e contagem de tráfego em tempo real; - É difícil integrar esta técnica com os restantes componentes da aplicação (com o Snort, mais concretamente).	- As duas desvantagens apresentadas têm um peso muito forte e dificultam a implementação da aplicação.

Tabela 6 – Comparação das técnicas consideradas para a captura do tráfego

Assim sendo, optou-se por adotar a estratégia de ARP *Spoofing* porque, apesar de ser um pouco "invasiva", é aquela que melhor se enquadra na perspetiva e nos princípios da aplicação e a que permite realizar uma melhor integração dos diferentes módulos e das ferramentas que neles são exploradas. Para além disso, esta técnica foi alvo de uma quantidade considerável de testes e, tanto quanto se apurou, apresenta um desempenho aceitável e adequado às condições e ao ambiente específico desta aplicação, fornecendo garantias suficientes para que seja escolhida para a implementação deste módulo.

7.4.4 Contagem do tráfego

Com o intuito de se averiguar a quantidade de tráfego e de dados que os dispositivos de IoT geram e recebem nas suas comunicações para o exterior da rede, construiu-se este módulo. A sua finalidade é realizar *sniffing* sobre a interface de rede Wi-Fi, onde terá acesso ao tráfego intercetado entre o *router* e os equipamentos de IoT, pelo módulo anterior.

Para isso, utiliza a biblioteca Scapy da linguagem de programação Python, que permite analisar, pacote a pacote, todo o tráfego que passa na placa de rede. Com isto em mente, este módulo cria um ficheiro JSON em que, para cada um dos dispositivos de IoT identificados, efetua o registo das comunicações realizadas, armazenando as seguintes informações:

- Sentido da comunicação (transmissão ou receção);
- Quantidade de pacotes trocados;
- Endereços IP das máquinas com quem comunica;
- Protocolos usados nas comunicações.

Através deste ficheiro, torna-se possível gerar relatórios para o utilizador com o intuito de lhe fornecer informação relativamente à quantidade de dados recebidos e transmitidos por cada equipamento associado ao conceito de *Internet of Things*, bem como os intervenientes e os protocolos utilizados para realizar essas comunicações.

7.4.5 Análise do tráfego

Neste módulo da aplicação, pretende-se realizar uma análise sobre todas as comunicações com origem ou destino nos equipamentos de IoT. Com isso, objetiva-se identificar tráfego suspeito e que possa ser, potencialmente, malicioso e prejudicial para a segurança da rede.

Para implementar esta funcionalidade, recorreu-se ao *software* Snort, o NIDS líder do mercado. Esta ferramenta faz *sniffing* do tráfego que circula sobre uma interface de rede e gera alertas (*warnings*), quando algum pacote ou conjunto de pacotes coincide com uma das regras configuradas e associadas a padrões típicos de ataques e *exploits*.

Este tipo de abordagem é baseado em *signatures* e, por isso, foi necessário configurar um conjunto de regras com os diversos padrões dos tipos de ataques conhecidos. Para isso, aproveitaram-se as *community rules* existentes no Snort. Com base no tráfego intercetado, o Snort percorre a lista de regras em busca de uma correspondência.

Assim, é possível integrá-lo nos restantes módulos da aplicação e, consequentemente, adicionar funcionalidades e ferramentas que este não suporta. Para além disso, fica especificamente orientado à deteção de intrusões associadas aos equipamentos de IoT que, tal como já foi referido, é extremamente inovador e, atualmente, ausente no mercado.

Posto isto, utilizou-se o seguinte comando na execução do Snort:

```
snort -d -h <network_address> -i <interface_name> -A -c <config_file>
-l <logging_dir> -K ascii
```

Onde as opções utilizadas têm o seguinte significado:

- `-d`: indica que não se pretende toda a informação quando é gerado o alerta no terminal;
- `-h <network_address>`: indica o endereço de rede para ser utilizado nas regras como variável $HOME_NET;
- `-i <interface_name>`: indica a interface em que vai estar à escuta;
- `-A`: indica que se um pacote bater numa regra gera um alerta;
- `-c <config_file>`: indica qual o ficheiro de configuração deve ser usado;
- `-l <logging_dir>`: especifica a diretoria onde serão guardados os ficheiros de *logging* gerados pelo Snort;
- `-K ascii`: especifica que os ficheiros de *logging* serão gerados de acordo com a codificação de carateres ASCII.

Apesar de se ter optado pelo Snort, existem outras ferramentas semelhantes que já foram abordadas no Capítulo 1 deste relatório e que se apresentam como alternativas válidas. Entre elas, encontram-se o Suricata, o Zeek e o Sagan.

A escolha acabou por recair sobre o Snort por ser, atualmente, o líder destacado do mercado dos *Intrusion Detection Systems* e, ainda mais importante, por disponibilizar uma muito boa documentação acerca da utilização desta aplicação. Para além disso, é também aquela que possui mais informação *online*, o que se traduz numa grande vantagem, visto que se pretende explorar uma ferramenta com a qual nunca se interagiu.

7.4.6 Análise dos portos

Para desenvolver esta funcionalidade da aplicação, voltou-se a tirar partido da ferramenta Nmap. Aqui, pretende-se saber quais os portos que cada equipamento de IoT tem abertos e, se possível, tentar perceber os serviços a estes associados. Para além disso, objetiva-se reconhecer a ocorrência de mudanças no estado dos portos dos diversos dispositivos, dado que isso pode ser relevante para detetar intrusões ou *backdoors* nos mesmos.

Desta forma, utilizou-se o seguinte comando na execução do Nmap para cada um dos dispositivos que foram identificados como associados a IoT:

```
nmap -sT -Pn -n <IPaddress>
```

Onde as opções utilizadas têm o seguinte significado:

- `-sT`: especifica que se pretende realizar um *port scan* do tipo TCP *Connect*;
- `-Pn`: indica que se pretende apenas realizar *port scanning* e desativa o *host discovery*;
- `-n`: especifica que não se pretende realizar resoluções de DNS;
- `<IPaddress>`: corresponde ao endereço IP de cada um dos dispositivos.

Sempre que ocorre uma mudança no estado dos portos dos vários dispositivos, é gerado um novo ficheiro JSON que armazena, para cada um deles, o estado dos seus portos e os serviços que, possivelmente, se poderão encontrar a correr nos mesmos.

Para além do Nmap, existem outras ferramentas similares que podiam ser exploradas para este efeito como, por exemplo, o Masscan e o Nikto. Ainda há outras alternativas, mas foram diretamente descartas por apenas possuírem uma interface gráfica e não uma interface de linha de comandos (SPARTA, por exemplo).

Relativamente a estas três possibilidades, acabou-se por optar pelo Nmap que, tal como o Snort é líder entre as ferramentas de IDS, este é líder entre as de *port scanning*. Para além disso e assim como já foi referido, já se tinha explorado esta ferramenta fora do âmbito deste projeto e, por essa razão, já existe uma boa familiarização com a mesma.

7.4.7 Verificação de vulnerabilidades

Para verificar as vulnerabilidades associadas aos portos e serviços dos equipamentos de IoT anteriormente identificados, explorou-se a ferramenta SearchSploit [77]. Esta ferramenta permite pesquisar, através de resultados obtidos pelo Nmap, vulnerabilidades conhecidas e presentes na base de dados Exploit Database [78], da Offensive Security.

Desta forma, voltou-se a utilizar o Nmap para se realizar uma análise mais morosa, minuciosa e completa dos portos dos dispositivos e dos serviços associados aos mesmos. Com base nos resultados, esta ferramenta gera um ficheiro XML para cada um deles. Estes são, depois, usados como *input* no programa SearchSploit para efetuar a pesquisa na base de dados.

Naturalmente, caso sejam detetadas vulnerabilidades nos equipamentos ou nos serviços associados aos seus portos, são gerados ficheiros de alerta para reportar ao utilizador os dados críticos encontrados.

O Nmap é executado, neste módulo, de forma semelhante ao módulo do ponto 7.4.6 (análise dos portos), adicionando-lhe o seguinte argumento:

- `-sV`: sonda e examina os portos da máquina alvo, de forma mais minuciosa, para determinar que serviço se encontra a correr em cada um deles, bem como a versão de *software* associada aos mesmos;

Como se viu, para a verificação de vulnerabilidades nos equipamentos de IoT da rede doméstica, optou-se por utilizar a ferramenta SearchSploit, em conjunto com o Nmap. Contudo, poder-se-iam explorar outras formas de atingir este objetivo. Nomeadamente, existem APIs para bases de dados de CVEs, que produziriam um efeito semelhante.

Deste modo, optou-se pelo SearchSploit, principalmente, pela possibilidade de, de forma simples e direta, se poder integrar com o Nmap. Para além disso, já se tinha alguma experiência com a utilização desta ferramenta e, por isso, a adoção da mesma na implementação da aplicação de NIDS revelou-se um processo rápido e elementar.

7.4.8 Geração de *logfiles*

Até aqui, discutiram-se os módulos de reconhecimento, deteção e análise. Contrariamente a estes, o módulo de geração de *logfiles* é responsável por gerar os devidos ficheiros de extensão `.log` para reportar ao utilizador aquilo que os módulos anteriores auferiram.

Desta forma, sempre que seja detetada alguma irregularidade, uma mudança não expectável ou um acontecimento de risco, cabe a este módulo gerar os respetivos relatórios que apresentem, de maneira detalhada, os riscos que podem estar presentes na rede ou nos equipamentos de IoT a ela ligados.

Enquanto que este módulo se responsabiliza pela elaboração de ficheiros que recolham e indiquem a informação passada pelos anteriores, o módulo que se segue no ponto 7.4.9 (geração de alertas) é responsável por alertar, de forma direta, o utilizador para a ocorrência destes potenciais problemas, levando, por este modo, a que o utilizador consulte os *logfiles* criados.

A geração de *logfiles* ocorre ao longo dos módulos apresentados nos pontos 7.4.5 (análise do tráfego), 7.4.6 (análise dos portos) e 7.4.7 (verificação de vulnerabilidades) para guardar e destacar comportamentos fora do comum por parte dos equipamentos de IoT da rede, seja relativamente ao seu tráfego, portos ou vulnerabilidades em serviços.

7.4.9 Geração de alertas ao utilizador

Este último módulo da aplicação de NIDS tem o objetivo de alertar o utilizador para a possível ocorrência de eventos de risco na rede e que possam colocar a sua segurança e a dos seus dispositivos em risco.

Ao receber este alerta, é expectável que o utilizador vá consultar os *logfiles* gerados pelo módulo apresentado no ponto 7.4.8 (geração de *logfiles*) para averiguar a natureza do problema, verificando se se trata de uma ameaça real ao equipamento e à rede ou apenas de um falso positivo ou um acontecimento do qual o utilizador estava consciente e que não representa, de qualquer maneira, uma ameaça para os seus dispositivos ou para a sua rede doméstica.

Posto isto, estes alertas podem ser gerados de duas maneiras possíveis:

- Através do envio de um *email* para um endereço de correio eletrónico a ser especificado pelo utilizador;
- Através do envio de uma mensagem de texto (SMS) para um número de telemóvel a ser indicado pelo utilizador.

Assim sendo, este módulo comunica diretamente com um servidor remoto, através do mecanismo de chamadas remotas RPC (*Remote Procedure Call*), que disponibiliza estes dois serviços como forma de alertar o utilizador.

Mediante a criação de *logging files* por parte da aplicação, estes são enviados ao servidor remoto juntamente com uma mensagem de alerta para que seja enviado um *email* ao utilizador a reportar essa circunstância. Para além disso, é também enviada outra mensagem mais pequena e menos detalhada com o intuito de advertir o utilizador via mensagem de texto (SMS), para que este vá consultar o ficheiro de *logging* enviado por *email* ou armazenado na Raspberry Pi.

8 Trabalho futuro

A realização deste relatório e o desenvolvimento da aplicação de NIDS para equipamentos domésticos associados ao conceito de *Internet of Things*, deixaram-nos plenamente conscientes de que, apesar de sentirmos que fizemos um bom trabalho de estudo e de implementação, ainda existe trabalho a realizar no futuro que pode enriquecer a nossa pesquisa e aperfeiçoar e complementar as funcionalidades da aplicação desenvolvida. Desta forma, este capítulo serve para expor as nossas ideias de trabalho futuro.

De um ponto de vista mais teórico, objetiva-se estudar uma maior gama de equipamentos associados a IoT e analisar, pormenorizadamente, o tráfego que geram e recebem no seu normal funcionamento. Por motivos de escassez de recursos, apenas se estudaram quatro tipos de dispositivos, o que representa apenas uma gota de água num enorme oceano. Por essa razão, tem-se o intuito de alargar o espetro e proceder à análise de muitos mais equipamentos para que se possa testar, em primeira mão, a qualidade da segurança que implementam ou a total ausência da mesma.

Para além disso, estamos determinados em realizar mais *pentesting* a este tipo de dispositivos e em explorar outras ferramentas e vetores de ataque. Não só tínhamos poucos equipamentos em nossa posse, como estivemos sempre limitados pelo tempo associado à realização deste projeto. Por estas razões, não nos foi possível aprofundar este tema, que constitui uma das principais áreas onde temos maior interesse. Por exemplo, é revelante referir que seria bastante interessante averiguar que tipo de verificações são feitas nos equipamentos de IoT que utilizam canais TLS para comunicar com servidores remotos – se apenas é conferido o *hostname* do certificado ou se são feitas validações ao nível da PKI – e se seria possível abrir esses canais e realizar um ataque MITM para obter o conteúdo em claro das mensagens trocadas.

Relativamente ao desenvolvimento da aplicação de NIDS, tem-se a ambição de integrar *Machine Learning* na sua implementação, nomeadamente, no módulo de identificação dos dispositivos de IoT que se encontram ligados à rede local. Com isto, pretende-se realizar uma melhor deteção destes equipamentos e menos propensa a erros e falhas. Ao mesmo tempo, estar-se-ia a englobar um tópico emergente e cada vez mais essencial, nos dias de hoje.

No que diz respeito ao uso do Snort, pretende-se aprofundar mais esta ferramenta de IDS e ir além da utilização das *community rules* como método de deteção de anomalias. Desta forma, o nosso objetivo neste ponto prende-se com o desenvolvimento de regras mais específicas ao universo abrangido por este projeto e que se apliquem diretamente sobre as intrusões, os ataques e as vulnerabilidades ao nível dos equipamentos associados ao conceito de *Internet of Things*, num ambiente doméstico.

Por fim, deseja-se desenvolver uma interface gráfica via *web browser* para disponibilizar ao utilizador o acesso HTTP (ou, eventualmente, HTTPS) à Raspberry Pi e, mais concretamente, à aplicação. A finalidade desta funcionalidade reside na possibilidade de se efetuar uma melhor e mais simples gestão sobre os recursos gerados pela mesma (isto é, ficheiros com a informação obtida e relatórios de riscos e vulnerabilidades) e, potencialmente, de permitir pontuais e opcionais configurações por parte do utilizador, que possam melhorar ou personalizar o serviço prestado pela ferramenta.

Conclusão

Ao se iniciar este projeto com uma exposição do contexto do mesmo, foi possível construir uma ideia geral sobre o conceito de *Internet of Things* e sobre a deteção de intrusões. O levantamento do estado da arte permitiu perceber o tipo de ferramentas de IDS que existem e a forma como se distinguem entre elas. Para além disso, teve-se a possibilidade de verificar as lacunas que existem e, consequentemente, as razões para haver a necessidade de se desenvolver uma aplicação deste tipo.

Com a realização de um estudo conciso e sucinto acerca dos principais protocolos utilizados pelos dispositivos domésticos associados ao conceito de *Internet of Things*, foi possível perceber quais se aplicam e são usados em ambientes domésticos, conforme o âmbito deste projeto. Fora isso, conseguiu-se analisar que tipo de mecanismos de segurança são implementados por estes protocolos e quais são as maiores vulnerabilidades a que estão sujeitos e os ataques conhecidos contra os mesmos.

A realização de um estudo sobre os equipamentos de IoT utilizados nas redes domésticas permitiu-nos fazer uma análise prática dos portos e protocolos usados para efetuar as comunicações na *Local Area Network* e para a *Wide Area Network*. Nesta fase, foi feita uma exploração de ferramentas de captura e análise de tráfego na rede, com o objetivo de procurar perceber o tráfego gerado e recebido por este tipo de dispositivos em duas principais situações: quando o telemóvel que controla o equipamento se encontra na mesma rede; quando o telemóvel que controla o equipamento se encontra fora da rede.

Posteriormente, com a exploração das diversas funcionalidades dos *routers* domésticos, foi possível constatar que tipos de facilidades implementam segurança na rede e nos equipamentos a esta ligados, bem como aquelas que podem colocar a rede e os seus dispositivos em risco, quando utilizadas de forma errada ou inconsciente pelos utilizadores.

Para finalizar a fase de pesquisa do projeto, procurou-se realizar *Penetration Testing* aos equipamentos de IoT estudados e explorados anteriormente, seguindo o princípio de "procurar saber como atacar para se saber como defender". Desta forma, tentou-se explorar alguns vetores de ataque para se encontrar vulnerabilidades nesses dispositivos e testar a segurança oferecida pelos protocolos que utilizam. Para além disso, foi feito um levantamento dos principais ataques conhecidos aos *chipsets* Wi-Fi da Espressif, que são largamente adotados para implementar este tipo de equipamentos e, por isso, representam um tópico crucial no desenvolvimento deste projeto.

Por fim, a implementação propriamente dita da aplicação de *Network-based Intrusion Detection System* para equipamentos de *Internet of Things* em ambientes domésticos revelou-se um processo consideravelmente difícil e trabalhoso, mas extremamente enriquecedor e satisfatório. Com a sua realização, não só foi possível

consolidar conhecimentos mais teóricos, como, principalmente, desenvolver capacidades de programação numa nova linguagem. Ao longo de todo este trajeto, foram-se explorando vários tipos de programas e ferramentas que, em alguns casos, foram integrados na aplicação desenvolvida, de forma a atingir os objetivos a que nos propusemos com a realização deste Projeto Final de Curso de Licenciatura.

Para concluir, resta agradecer ao Instituto Superior de Engenharia de Lisboa, à Área Departamental de Engenharia Eletrónica e Telecomunicações e de Computadores e a todos os professores que, ao longo do ciclo de estudos, contribuíram com conhecimento e ferramentas que nos ajudaram a atingir esta determinante etapa das nossas vidas. Um especial agradecimento ao professor Vítor Almeida, que idealizou este projeto e que nos orientou no sentido de atingirmos, da melhor forma possível, os nossos ambiciosos objetivos.

Referências

[1] "Internet of Things," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things.

[2] K. L. Lueth, "State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating," IoT Analytics, 2018. [Online]. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[3] N. G., "How Many IoT Devices Are There?," TechJury, 2020. [Online]. Available: https://techjury.net/blog/how-many-iot-devices-are-there/.

[4] "How IoT devices & smart home automation is entering our homes in 2020," Business Insider, 2020. [Online]. Available: https://www.businessinsider.com/iot-smart-home-automation.

[5] M. Haroon, "Building Affordable Intrusion Detection System for Internet of Things," *University of Oslo,* 2018.

[6] A. Kliarsky, "Detecting Attacks Against The Internet of Things," *SANS Institute,* 2017.

[7] S. Cooper, "Intrusion Detection Systems Explained," CompariTech, 2020. [Online]. Available: https://www.comparitech.com/net-admin/network-intrusion-detection-tools/.

[8] C. Douligeris e D. Serpanos, Network Security: Current Status and Future Directions, John Wiley & Sons, 2007.

[9] "Intrusion detection system," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Intrusion_detection_system.

[10] "Network Based Intrusion Detection System," ScienceDirect, [Online]. Available: https://www.sciencedirect.com/topics/computer-science/network-based-intrusion-detection-system.

[11] "Snort (software)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Snort_(software).

[12] "Suricata (software)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Suricata_(software).

[13] "Zeek," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Zeek.

[14] "Sagan (software)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Sagan_(software).

[15] T. Jaffey, "MQTT and CoAP, IoT Protocols," Eclipse Foundation, 2014. [Online]. Available: https://www.eclipse.org/community/eclipse_newsletter/2014/february/article2.php.

[16] "MQTT," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/MQTT.

[17] "Frequently Asked Questions," MQTT, [Online]. Available: https://mqtt.org/faq.

[18] "What is CoAP protocol IoT," RF Wireless World, [Online]. Available: https://www.rfwireless-world.com/IoT/CoAP-protocol.html.

[19] "Datagram Transport Layer Security," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security.

[20] "The Constrained Application Protocol (CoAP)," IETF - RFC 7252, [Online]. Available: https://tools.ietf.org/html/rfc7252.

[21] "Advanced Message Queuing Protocol," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol.

[22] I. N. McAteer , M. I. Malik, Z. Baig e P. Hannay, *Security vulnerabilities and cyber threat analysis of the AMQP,* 2017.

[23] G. Nebbione e M. C. Calzarossa, "Security of IoT Application Layer Protocols: Challenges and Findings," *Future Internet,* 2020.

[24] "Service Name and Transport Protocol Port Number Registry," IANA, [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[25] "Data Distribution Service," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Data_Distribution_Service.

[26] "What is DDS?," DDS Foundation, [Online]. Available: https://www.dds-foundation.org/what-is-dds-3/.

[27] "About the DDS Security Specification," OMG, [Online]. Available: https://www.omg.org/spec/DDS-SECURITY/.

[28] "Extensible Messaging and Presence Protocol (XMPP): Core," IETF - RFC6120, [Online]. Available: https://tools.ietf.org/html/rfc6120.

[29] "XMPP," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/XMPP.

[30] "DNS-Based Service Discovery," IETF - RFC6763, [Online]. Available: https://tools.ietf.org/html/rfc6763.

[31] "Multicast DNS," IETF - RFC 6762, [Online]. Available: https://tools.ietf.org/html/rfc6762.

[32] "DNS Security Introduction and Requirements," IETF - RFC 4033, [Online]. Available: https://tools.ietf.org/html/rfc4033.

[33] "Specification for DNS over Transport Layer Security (TLS)," IETF - RFC 7858, [Online]. Available: https://tools.ietf.org/html/rfc7858.

[34] "DNS Queries over HTTPS (DoH)," IETF - RFC 8484, [Online]. Available: https://tools.ietf.org/html/rfc8484.

[35] "Multicast DNS," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Multicast_DNS.

[36] "Simple Service Discovery Protocol," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Simple_Service_Discovery_Protocol.

[37] "DEF CON 24 Hacking Conference," DEF CON, [Online]. Available: https://www.defcon.org/html/defcon-24/dc-24-index.html.

[38] "CVE-2019-11779," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-11779.

[39] "CVE-2019-6241," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-6241.

[40] "CVE-2017-7650," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-7650.

[41] "Summarizing Known Attacks on Transport Layer Security (TLS) and Datagram TLS (DTLS)," IETF - RFC 7457, [Online]. Available: https://tools.ietf.org/html/rfc7457.

[42] "CVE-2018-12679," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-12679.

[43] "CVE-2018-12680," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-12680.

[44] "CVE-2019-17212," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17212.

[45] "CVE-2019-9750," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-9750.

[46] "CVE-2015-1892," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-1892.

[47] "CVE-2017-6519," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6519.

[48] "CVE-2017-6520," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-6520.

[49] "CVE-2015-0650," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0650.

[50] "CVE-2019-14323," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-14323.

[51] "CVE-2019-14363," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-14363.

[52] "CVE-2014-5406," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-5406.

[53] "CVE-2015-4051," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-4051.

[54] "DMZ (computing)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/DMZ_(computing).

[55] "What is DMZ and how to configure DMZ host?," Port Checker, [Online]. Available: https://www.portcheckers.com/what-is-dmz-in-router-dmz-configuration.

[56] A. Green, "What is UPnP & Why is it Dangerous?," Varonis, 2019. [Online]. Available: https://www.varonis.com/blog/what-is-upnp/.

[57] "Mirai (malware)," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Mirai_(malware).

[58] "NMAP," [Online]. Available: https://nmap.org/.

[59] "DIY Mode API Protocol," SONOF, [Online]. Available: http://developers.sonoff.tech/sonoff-diy-mode-api-protocol.html.

[60] "New network ports listening on Alexa," reddit, [Online]. Available: https://www.reddit.com/r/AmazonEchoDev/comments/cyk24g/new_network_ports_listening_on_alexa/.

[61] "Replay attack," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Replay_attack.

[62] "Ettercap," Ettercap Project, [Online]. Available: https://www.ettercap-project.org/index.html.

[63] "Wireshark," Wireshark, [Online]. Available: https://www.wireshark.org/.

[64] "Tcpreplay," GitHub, [Online]. Available: https://github.com/appneta/tcpreplay.

[65] "Fears about Sonoff Ewelink hacking might be true," reddit, [Online]. Available: https://www.reddit.com/r/homeautomation/comments/atcime/fears_about_sonoff_ewelink_hacking_might_be_true/.

[66] "CVE-2018-18558," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2018-18558.

[67] "CVE-2019-12586," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-12586.

[68] "CVE-2019-12587," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-12587.

[69] "CVE-2019-12588," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-12588.

[70] "CVE-2019-15894," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-15894.

[71] "Espressif Security Advisory Concerning Fault Injection and Secure Boot (CVE-2019-15894)," Espressif, 2019. [Online]. Available: https://www.espressif.com/en/news/Espressif_Security_Advisory_Concerning_Fault_Injection_and_Secure_Boot.

[72] "CVE-2019-17391," Common Vulnerabilities and Exposures, [Online]. Available: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-17391.

[73] "Security Advisory concerning fault injection and eFuse protections (CVE-2019-17391)," Espressif, 2019. [Online]. Available: https://www.espressif.com/en/news/Security_Advisory_Concerning_Fault_Injection_and_eFuse_Protections.

[74] "Raspberry Pi," Raspberry Pi Foundation, [Online]. Available: https://www.raspberrypi.org/.

[75] "Raspberry Pi," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi.

[76] "Kali Linux," Offensive Security, [Online]. Available: https://www.kali.org/.

[77] "SearchSploit," Exploit Database, [Online]. Available: https://www.exploit-db.com/searchsploit.

[78] "Exploit Database," Offensive Security, [Online]. Available: https://www.exploit-db.com/.

[79] "Ping of death," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Ping_of_death.

[80] "Constrained Application Protocol," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Constrained_Application_Protocol.

[81] "MQTT Version 5.0," OASIS, 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[82] "Port forwarding," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Port_forwarding.

[83] "Sabe o que é port forwarding e qual a sua utilização?," pplware, 2017. [Online]. Available: https://pplware.sapo.pt/tutoriais/networking/sabe-port-forwarding-qual-utilizacao/.

[84] "What is a DMZ and how to configure DMZ host," TP-Link, [Online]. Available: https://www.tp-link.com/en/support/faq/28/.

[85] "Replay a tcp packet captured by wireshark," Share Your Thoughts to Blog, 2016. [Online]. Available: https://shareyourthoughtstoblog.blogspot.com/2016/10/replay-tcp-packet-captured-by-wireshark.html.

[86] "Nmap Reference Guide," NMAP, [Online]. Available: https://nmap.org/book/man.html.

[87] "Ports Database," Speed Guide, [Online]. Available: https://www.speedguide.net/ports.php.

[88] C. Hoffman, "Is UPnP a Security Risk?," How-To Geek, [Online]. Available: https://www.howtogeek.com/122487/htg-explains-is-upnp-a-security-risk/.

[89] "Universal Plug and Play," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Universal_Plug_and_Play.

[90] "Simple Authentication and Security Layer," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Simple_Authentication_and_Security_Layer.

[91] "Simple Authentication and Security Layer," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Simple_Authentication_and_Security_Layer.

[92] "Arbitrary code execution," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Arbitrary_code_execution.

[93] "Reflection attack," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Reflection_attack.

[94] "Denial-of-service attack," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Denial-of-service_attack.

[95] "AMQP Version 1.0," OASIS, 2012. [Online]. Available: https://docs.oasis-open.org/amqp/core/v1.0/os/amqp-core-overview-v1.0-os.html.

[96] "ARP spoofing," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/ARP_spoofing.

[97] "Man-in-the-middle attack," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Man-in-the-middle_attack.

[98] "Extensible Authentication Protocol," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Extensible_Authentication_Protocol.

[99] "Fault injection," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Fault_injection.

Apêndices

Apêndice A: Código-fonte

O código-fonte (*source code*) correspondente ao desenvolvimento e à implementação da aplicação *Home Security* – NIDS *for domestic* IoT está disponível no seguinte repositório Git:

- https://github.com/Dr0g0n/HomeSecurity

Apêndice B: Exemplo de Demonstração

Para complementar este relatório sobre o Projeto Final de Curso desenvolvido, optou-se por colocar um exemplo simples de demonstração da aplicação implementada para que qualquer leitor possa testemunhar o seu funcionamento.

Tal como foi explicado ao longo do Capítulo 7.4, a primeira ação realizada pela aplicação passa por realizar um *scan* à rede e listar todos os equipamentos a ela ligados. Estes equipamentos são colocados no ficheiro `hosts/out.json`.

```
root@kali:~/fase1/finalProject/hosts# cat out.json
{
    "devices": [
        {
            "IP": "192.168.88.1",
            "MAC": "B8:69:F4:55:85:E2",
            "Vendor": "Routerboard.com"
        },
        {
            "IP": "192.168.88.241",
            "MAC": "00:0C:29:4F:08:B6",
            "Vendor": "VMware"
        },
        {
            "IP": "192.168.88.249",
            "MAC": "58:20:B1:62:B0:B6",
            "Vendor": "Hewlett Packard"
        },
        {
            "IP": "192.168.88.251",
            "MAC": "CC:50:E3:15:E4:16",
            "Vendor": "Espressif"
        },
        {
            "IP": "192.168.88.252",
            "MAC": "60:01:94:73:19:3D",
            "Vendor": "Espressif"
        },
        {
            "IP": "192.168.88.254",
            "MAC": "60:45:CB:31:B1:00",
            "Vendor": "Asustek Computer"
        },
        {
            "IP": "192.168.88.245",
            "MAC": "",
            "Vendor": ""
        },
        {
            "IP": "192.168.88.247",
            "MAC": "",
            "Vendor": ""
        }
    ]
}
```

Figura 20 – Ficheiro com os *hosts* detetados na rede

(Nota: Na Figura 20, os dois últimos endereços IP não se encontram identificados porque pertencem à própria Raspberry Pi.)

Depois de se ter feito o reconhecimento sobre a rede, é necessário identificar quais desses equipamentos estão associados ao conceito de *Internet of Things*. Como foi explicado na Capítulo 7.4, esta identificação é realizada através do endereço MAC. Desta forma, estes dispositivos (a par do *router* da rede doméstica) são colocados no ficheiro `hosts/hosts.txt`.

```
root@kali:~/fase1/finalProject/hosts# cat hosts.txt
192.168.88.241 00:0C:29:4F:08:B6 -
192.168.88.251 CC:50:E3:15:E4:16 -
192.168.88.252 60:01:94:73:19:3D -
192.168.88.1 B8:69:F4:55:85:E2 -
```

Figura 21 – Ficheiro com os dispositivos de IoT

Após os dispositivos IoT da rede estarem devidamente identificados, a aplicação começa a realizar a contagem de tráfego gerado e recebido pelos mesmos. Esta informação é armazenada num ficheiro do tipo JSON na diretoria `counters/`. É gerado um ficheiro semanalmente e o seu nome corresponde à data e hora da conclusão do mesmo. A Figura 22 mostra um exemplo deste ficheiro para dois equipamentos.


```
"192.168.88.241": {
    "Packets": 51,
    "Debits": {
        "TX": {
            "Number": 20,
            "Size": 2376
        },
        "RX": {
            "Number": 31,
            "Size": 3974
        }
    },
    "Protocols": [
        {
            "Protocol": "udp",
            "Quantity": 32
        },
        {
            "Protocol": "tcp",
            "Quantity": 19
        }
    ],
    "Outsiders": [
        {
            "Outsider": "54.76.30.11",
            "Quantity": 32
        },
        {
            "Outsider": "192.168.88.247",
            "Quantity": 19
        }
    ]
},
```

```
"192.168.88.251": {
    "Packets": 184,
    "Debits": {
        "TX": {
            "Number": 98,
            "Size": 5880
        },
        "RX": {
            "Number": 86,
            "Size": 6167
        }
    },
    "Protocols": [
        {
            "Protocol": "tcp",
            "Quantity": 184
        }
    ],
    "Outsiders": [
        {
            "Outsider": "192.168.88.247",
            "Quantity": 183
        },
        {
            "Outsider": "3.122.122.135",
            "Quantity": 1
        }
    ]
},
```


Figura 22 – Ficheiro com as estatítiscas de tráfego dos dispositivos de IoT

Depois do ficheiro semanal ser gerado, é enviado um *email* e um SMS ao utilizador para o informar sobre a estatística do tráfego dos seus equipamentos de IoT. Como se pode observar na Figura 23, o ficheiro é anexado ao *email*.

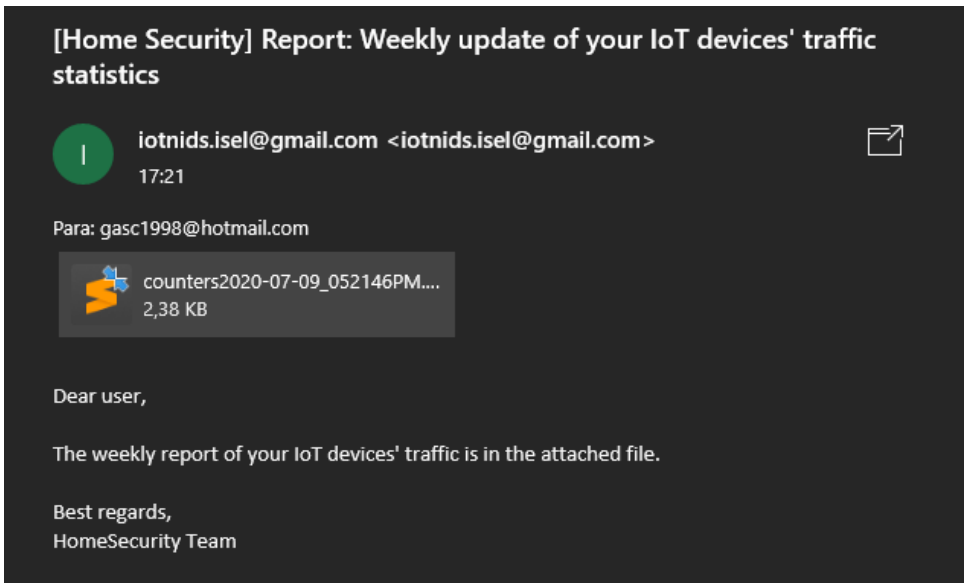

Figura 23 – *Email* de *update* semanal

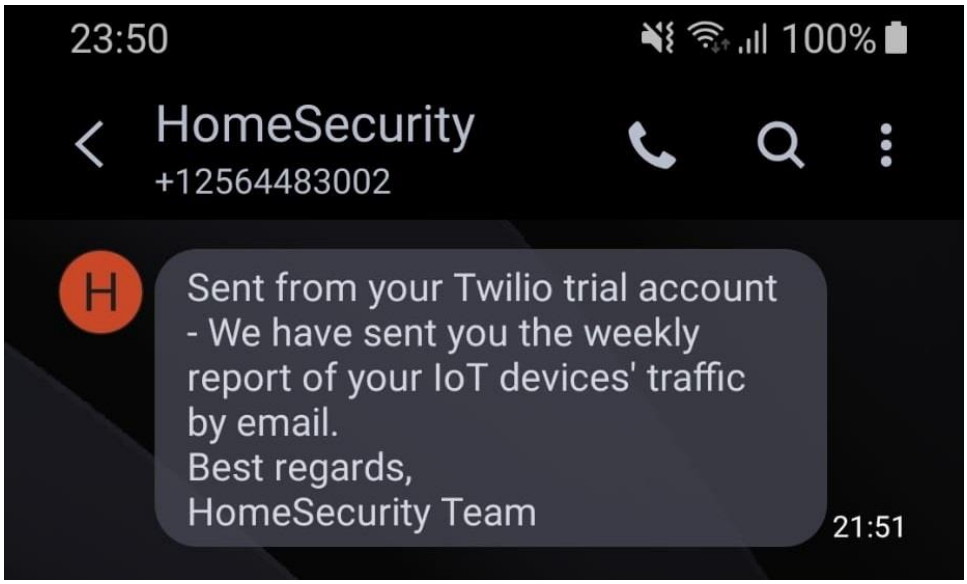

Figura 24 – SMS de *update* semanal

Para simular uma tentativa de intrusão à rede e verificar o comportamento do Snort, realizou-se um simples ataque de *Ping of death* [79]. Para isso, enviou-se um ICMP *Echo Request* com uma dimensão de 1300 bytes a um dos equipamentos de IoT. Como se pode verificar pela Figura 25, o Snort detetou esse pacote e gerou um *log* para alertar para esta circunstância.

```
[**] Ping of Death Detected [**]
07/08-15:36:45.128039 192.168.88.251 -> 192.168.88.241
ICMP TTL:128 TOS:0x0 ID:40844 IpLen:20 DgmLen:1328 DF
Type:0  Code:0  ID:11988  Seq:1  ECHO REPLY
FC D9 05 5F 00 00 00 00 D9 26 07 00 00 00 00 00  ..._.....&......
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   !"#$%&'()*+,-./
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F  0123456789:;<=>?
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F  @ABCDEFGHIJKLMNO
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F  PQRSTUVWXYZ[\]^_
60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F  `abcdefghijklmno
70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F  pqrstuvwxyz{|}~.
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F  ................
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F  ................
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF  ................
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF  ................
C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF  ................
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF  ................
E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF  ................
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF  ................
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  ................
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F  ................
```

Figura 25 – Ficheiro de *logging* do Snort

Estes ficheiros de *logging* gerados pelo Snort, aquando da deteção de pacotes que se enquadrem com as regras estabelecidas, são armazenados na diretoria `snortDir/`.

Consequentemente, é enviado um *email* ao utilizador para reportar esta situação crítica, onde é anexado o ficheiro de *logging* gerado pelo Snort. Para além disso, também é enviado um SMS semelhante ao da Figura 24, a informar o utilizador que deve de consultar esse mesmo ficheiro.

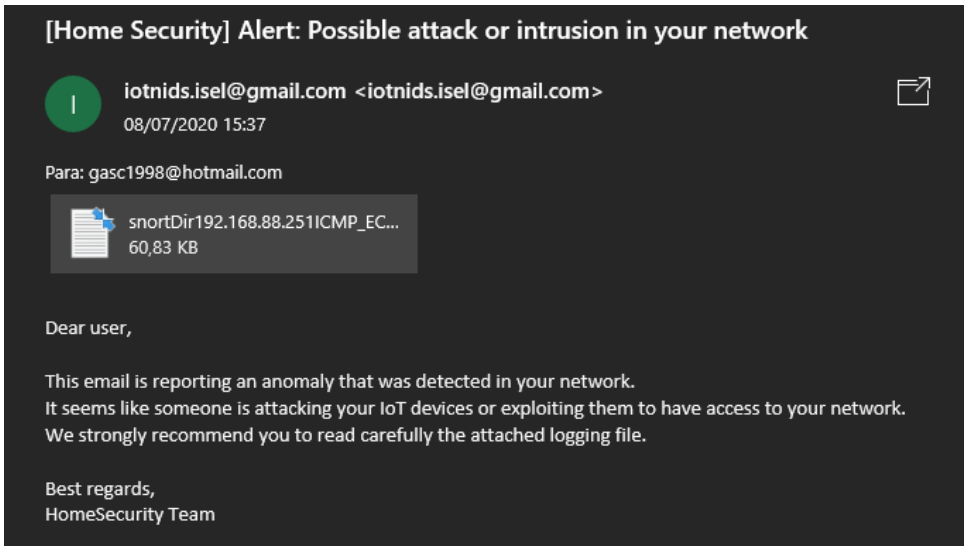

Figura 26 – *Email* a alertar para a deteção de uma possível intrusão

Relativamente à análise dos portos destes equipamentos, a aplicação gera dois tipos de ficheiros, JSON e TXT, que são guardados, respetivamente, nas diretorias `scans/json/` e `scans/text/`. O nome destes ficheiros baseia-se na data e hora da sua geração. A Figura 27 retrata um exemplo de um ficheiro TXT com os vários

dispositivos de IoT detetados na rede, bem como o estado dos seus portos e os serviços associados aos mesmos.

```
root@kali:~/fase1/finalProject/scans/text# cat 2020-07-09_05\:28\:45PM.txt
Device: 192.168.88.241
        1000/tcp  filtered  cadlock

Device: 192.168.88.251
        8081/tcp  open  blackice-icecap

Device: 192.168.88.252
        6668/tcp  open  irc

Device: 192.168.88.253
        1080/tcp  open  socks
        8888/tcp  open  sun-answerbook
```

Figura 27 – Ficheiro com o estado dos portos dos equipamentos de IoT

Caso surja alguma alteração no estado destes equipamentos, a aplicação gerará um ficheiro de *logging* a reportar esse acontecimento. Isso pode ocorrer mediante uma das seguintes situações:

- Um dos equipamentos de IoT identificados passa a estar *offline*;
- É adicionado um novo equipamento de IoT à rede;
- O estado de um ou mais portos de um dispositivo de IoT sofre alguma alteração.

Estes ficheiros são armazenados na diretoria `scans/logs/`.

Num cenário em que seja removido um equipamento da rede ou que seja adicionado um novo, são gerados ficheiros de *logging* semelhantes, respetivamente, aos apresentados pelas Figuras 28 e 29.

```
root@kali:~/fase1/finalProject/scans/logs# cat 2020-07-09_05\:17\:57PM.log
*************** LOGGING PORT SCAN REPORT ***************

Detected devices removed from the network (or currently down):
192.168.88.253
```

Figura 28 – *Logfile* gerado pela remoção de um dispositivo da rede

```
root@kali:~/fase1/finalProject/scans/logs# cat 2020-07-09_05\:28\:45PM.log
*************** LOGGING PORT SCAN REPORT ***************

Detected new devices in the network:
192.168.88.253
```

Figura 29 – *Logfile* gerado pelo surgimento de um novo dispositivo na rede

Da mesma forma, o mesmo se aplica para um cenário em que o estado de um ou mais portos de um dispositivo de IoT sofre alguma alteração. Neste caso, o ficheiro de *logging* parecer-se-á com o apresentado na Figura 30.

```
root@kali:~/fase1/finalProject/scans/logs# cat 2020-07-12_04\:15\:23PM.log
************** LOGGING PORT SCAN REPORT **************

It was detected that the following ports of device 192.168.88.251 are now closed:
81/tcp  filtered  hosts2-ns
11110/tcp  filtered  sgi-soap
4001/tcp  filtered  newoak
41511/tcp  filtered  unknown
```

Figura 30 – *Logfile* gerado pela mudança de estado de um ou mais portos

Para além da geração de *logs*, é ainda enviado um *email* ao utilizador a reportar estas anomalias (onde é anexado o respetivo ficheiro de *logging*), bem como um SMS.

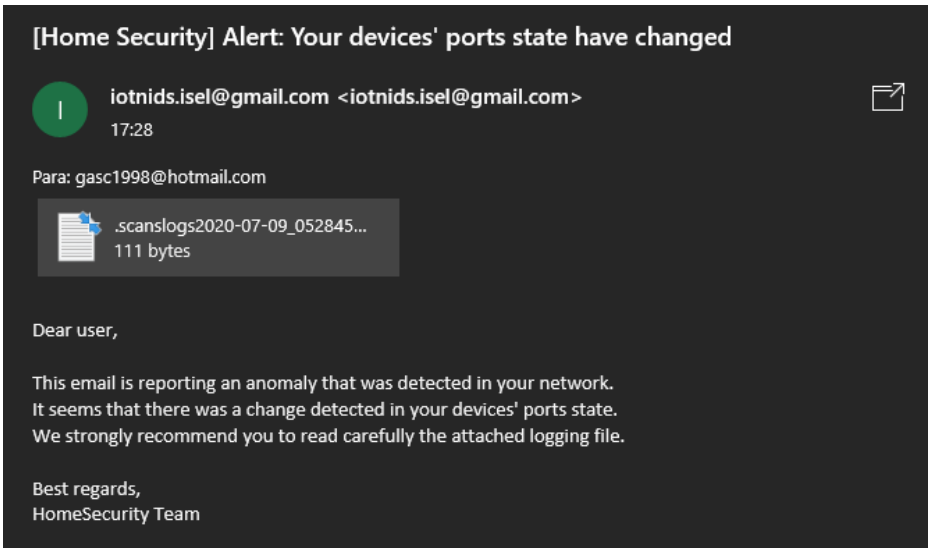

Figura 31 – *Email* a alertar para a deteção de uma alteração nos dispositivos

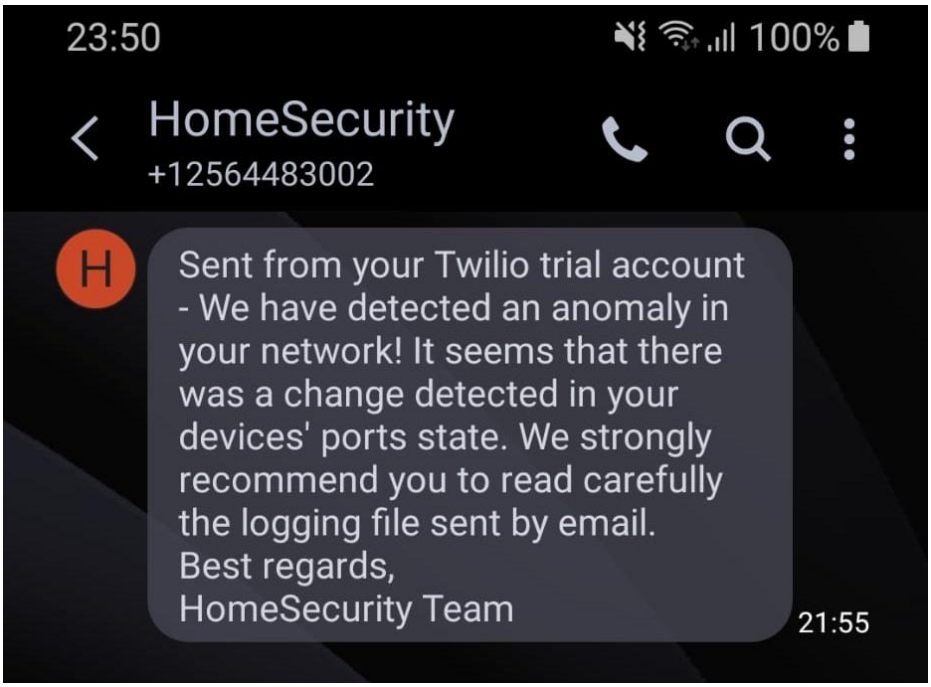

Figura 32 – SMS a alertar para a deteção de uma alteração nos dispositivos

Por fim, na verificação de vulnerabilidades, são gerados ficheiros de *logging* para cada um dos equipamentos de IoT. Estes ficheiros são armazenados na diretoria `vulns/logs/`.

Caso não seja reconhecida qualquer vulnerabilidade relacionada com um dispositivo concreto, nem com serviços associados aos portos que disponibiliza, é gerado um ficheiro semelhante ao apresentado pela Figura 33. Por outro lado, caso sejam identificadas uma ou mais vulnerabilidades, é gerado um ficheiro similar ao exposto pela Figura 34.

```
root@kali:~/fase1/finalProject/vulns/logs# cat 192.168.88.251.log
*************** LOGGING VULNERABILITY SCAN REPORT ***************

No vulnerabilities were found on this device!
```

Figura 33 – *Logfile* gerado pela ausência de vulnerabilidades detetadas

```
root@kali:~/fase1/finalProject/vulns/logs# cat 192.168.88.253.log
*************** LOGGING VULNERABILITY SCAN REPORT ***************

The table below presents the possible vulnerabilities found on this device:

------------------------------------------ ------------------------------------------
 Exploit Title                            |  URL
------------------------------------------ ------------------------------------------
XChat 1.8.0/2.0.8 socks5 - Remote Buffer Ov | https://www.exploit-db.com/exploits/296
------------------------------------------ ------------------------------------------
Shellcodes: No Results
```

Figura 34 – *Logfile* gerado pela deteção de uma possível vulnerabilidade

Para o segundo cenário, é então enviado um *email* ao utilizador a alertá-lo para a situação detetada, onde é anexado o respetivo ficheiro de *logging*. Para além disso, também é enviado um SMS semelhante ao da Figura 32, a informar o utilizador que deve de consultar esse mesmo ficheiro.

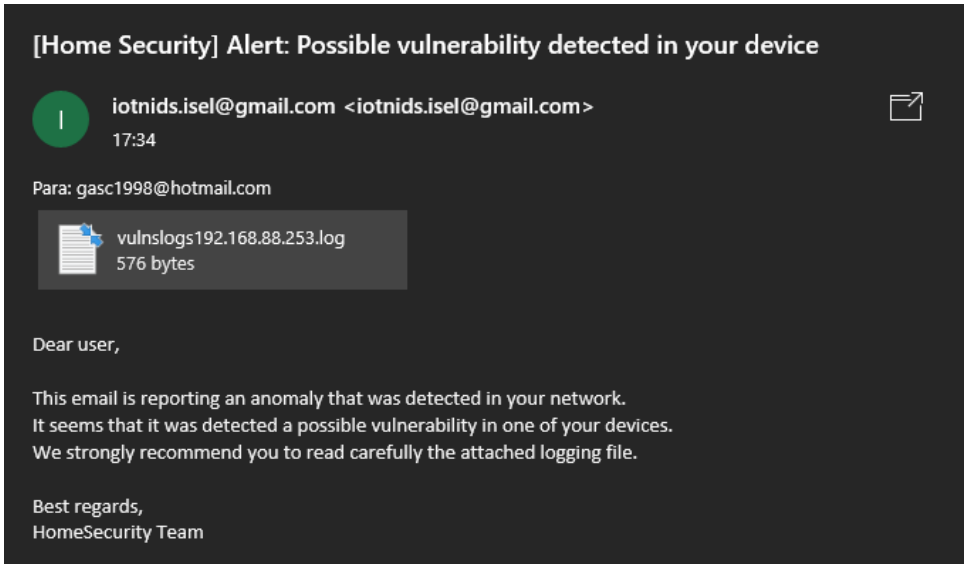

Figura 35 – *Email* a alertar para a deteção de uma possível vulnerabilidade