DISTRIBUTED ATTACKS DETECTION AND MITIGATION FOR THE INTERNET OF THINGS

METONGNON LIONEL MAYEUL A.

Thesis submitted in partial fulfilment of the requirements for the degree of:

- *Doctor of Sciences of Engineering and Technology from UCL*
- *Doctor of Sciences of Engineering from UAC*

March 2020

Institute of Information and Communication Technologies,
Electronics and Applied Mathematics (ICTEAM)
Université catholique de Louvain (UCL)
Louvain-la-Neuve, Belgium

Institut de Formation et de Recherche en Informatique (IFRI)
École Doctorale des Sciences de l'Ingénieur (ED-SDI)
Université d'Abomey-Calavi (UAC)
Abomey-Calavi, Bénin

Thesis Committee

Prof. Ramin Sadre (Advisor)	ICTEAM, UCL, Belgium
Prof. Eugène C. Ézin (Advisor)	IFRI, UAC, Benin
Prof. Anna Sperotto	DACS, UTwente, Netherlands
Prof. Bruno Quoitin	Institut d'Informatique, UMons, Belgium
Prof. Olivier Bonaventure (Secretary)	ICTEAM, UCL, Belgium
Prof. Charles Pêcheur (Chair)	ICTEAM, UCL, Belgium

To my parents, my girlfriend, my brothers, and my friends for all the support.

ABSTRACT

The term Internet of Things (IoT) refers to the increasing number of devices that are connected to communication networks and are able to exchange data autonomously. The proliferation of such devices of various types (refrigerator, sensor, television, camera, etc.) with permanent access to the Internet has led to new security problems. In fact, their large number and the lack of state-of-the-art protection mechanisms have made IoT devices interesting targets of cyber-attacks. An early example was the Mirai malware that infected poorly protected IoT devices and misused them for powerful attacks against other Internet hosts. In order to understand the nature of attacks related to the Internet of Things, the thesis starts with a study of the current IoT threat landscape. To this end, we have collected and studied data from a network telescope and various honeypots for IoT-specific application protocols. Our analysis has shown that although most attackers still target the unsecured telnet protocol, attempts against newer IoT protocols can be also observed. To help network operators to discover IoT devices in their networks, we have developed a new scan algorithm tailored to resource-constrained networks. We have shown that our approach reduces the time required to scan an IPv6 network by more than 35%, while maintaining a discovery rate of more than 95%. Finally, we have developed an intrusion detection and protection system in the form of a distributed architecture that detects abnormal behaviour and limits machine-to-machine communication according to policies defined by the device owners. In particular, our solution enables to stop attacks at their sources thanks to a hierarchical organization of inter-connected middleboxes. Traditional Internet services such as web servers are also protected against attacks originating from the monitored IoT devices.

RÉSUMÉ

Le terme "Internet des objets" (IoT) fait référence aux nombres croissants d'appareils connectés à des réseaux de communication et capables d'échanger des informations de manière autonome. La prolifération de ces appareils de divers types (réfrigérateur, capteur, télévision, caméra, etc.) ayant en prime l'accès à l'internet a entraîné de nouveaux problèmes de sécurité. En effet, vu leur grand nombre couplée à l'absence de mécanismes de protection, ces appareils connectés sont devenus la cible favorite pour les cyber-attaques. Un exemple récent est le malware Mirai qui a infecté beaucoup appareils connectés mal protégés et les a utilisés pour commettre de puissantes attaques contre d'autres hôtes sur Internet. Afin de comprendre la nature des attaques liées à l'internet des objets, nous avons commencé notre thèse par une étude des menaces actuelles liées aux appareils connectés. Nous avons à cette fin recueilli et étudié les données d'un télescope rśeau couplé à divers honeypots utilisant des protocoles applications spécifiques aux IoT. Notre analyse a montré que malgré que les attaquants ciblent encore en majorité le protocole non sécurisé telnet, des tentatives contre les nouveaux protocoles de l'IoT sont tout de même présentes. Dans le but d'aider les opérateurs de réseau à découvrir les appareils connectés dans leurs réseaux, nous avons ensuite mis au point un nouvel algorithme d'analyse adapté aux réseaux à ressources limitées. Nous avons montré que notre approche réduit de plus de 35% le temps nécessaire pour l'analyse d'un réseau IPv6, tout en maintenant un taux de découverte de plus de 95% des appareils présents. Enfin, nous avons développé un système de détection et de protection contre les intrusions sous la forme d'architecture distribuée de middleboxes qui détecte les

comportements anormaux et qui limite les communications de machine à machine selon des politiques prédéfinies par les propriétaires d'appareils connectés. Notre solution permet notamment de stopper les attaques depuis leurs sources grâce à une organisation hiérarchique de middleboxes. Notre système permet également de protéger les services Internet traditionnels, tels que les serveurs web contre les attaques provenant des appareils connectés surveillés.

ACKNOWLEDGEMENTS

A thesis is the combined efforts of a group. I cannot fail to mention the help, support and various contributions without which this work would not have been possible. I start by thanking my advisors Eugene C. Ezin and Ramin Sadre.

Eugene is the beginning of everything. He fought so that I could find a thesis and helped me with all his being to face the difficulties throughout my journey.

Ramin is the main reason for this work. He gave me his trust, supported me day and night (especially the day before the deadlines). He taught me how to focus on my goals, how to write good papers and enhanced my English vocabulary.

I would like to thank professors Marc Lobelle and Norbert Hounkonnou for allowing me to take advantage of the experience of CeFRI.

I would like to thank Roland van Rijswijk-Deij (SURFnet and University of Twente) for giving us access to the SURFnet infrastructure and for helping us with the configurations.

I spent 4 years in a wonderful department and I thank all their members. I am not very talkative but I can assure you that the pleasant atmosphere allowed me to blossom and to finish this thesis. Special thanks to Jawad, Igor and Gorby for the constructive exchanges, to François, Mickael and Olivier G. for the good moments spent together.

I would like to thank all Beninese PhD students and PhD namely Emery, Meton-Meton, Ratheil, Parfait, John, Gaël and Harold for the warm atmosphere and our philosophical discussions.

I thank all the members of INGI, the researchers as well as the administrative members. A special thanks to Vanessa who was always quick to help day and night.

I thank my family for tolerating my absence all these years and for all the support and encouragements despite the distance. I thank my brothers and sisters Orphée, Carine, David, Grace and Vladimir, my friends Alvic, Yannick, Marius, Jean de Dieu, Jaurès, Armand, Jorès, Elwis, Thierno, Christel, etc. for their encouragements.

Finally, I thank my fiancée Alida who has been a constant source of motivation, who believed in me and supported me throughout this thesis. Above all, I thank her for all the sacrifices made for me.

REMERCIEMENTS

Une thèse est la conjugaison d'efforts d'un ensemble. Je ne saurais passe sous silence l'aide, le soutien et les apports divers sans lesquels ce travail n'aurait pas vu le jour. Je commence par remercier mes promoteurs Eugene C. Ezin et Ramin Sadre.

Eugene est le commencement de tout. Il a bataillé pour que je puisse trouver une thèse et m'a aidé de tout son être à affronter les difficultés tout au long de mon périple.

Ramin est la raison principale de ce travail. Il m'a accordée sa confiance, m'a soutenu de jours comme de nuits (surtout les veilles de deadlines). Il m'a appris à me concentrer sur mes objectifs, à écrire de bons papiers et a enrichi mon vocabulaire en anglais.

Je remercie les professeurs Marc Lobelle et Norbert Hounkonnou pour m'avoir permis de faire partir de cette expérience qu'a été le CeFRI.

Je remercie Roland van Rijswijk-Deij (SURFnet et Université de Twente) de nous avoir donné accès à l'infrastructure SURFnet et de nous avoir aidés pour les configurations.

J'ai passe 4 ans dans un merveilleux département dont je remercie tous ses membres. Je ne suis pas très bavard mais je peux vous assurer que la plaisante ambiance m'a permis de m'épanouir et d'arriver à bout de cette thèse. Un merci tout particulier à Jawad, Igor et Gorby pour les échanges constructifs, à François, Michael, Olivier G. pour les bons moments passé ensemble.

Je remercie tous doctorants et docteurs Béninois à savoir : Emery, Meton-Meton, Ratheil, Parfait, John, Gaël et Harold pour l'ambiance et nos discussions philosophiques.

Je remercie tous les membres d'INGI, les chercheurs ainsi que les membres administratifs. Un merci particulier a Vanessa qui a toujours été prompt à aider de jours comme de nuits.

Je remercie ma famille pour avoir toléré mon absence et pour tout le soutien et les encouragements malgré la distance. Je remercie mes frères et sœurs Orphée, Carine, David, Grace et Vladimir, mes amis Alvic, Yannick, Marius, Jean de Dieu, Jaurès, Armand, Jorès, Elwis, Thierno, Christel, etc pour leurs encouragements.

Enfin, je remercie ma fiancée Alida qui a été une source de motivation constante, qui a cru en moi et m'a soutenu durant tout le long de cette thèse. Je la remercie surtout pour tous les sacrifices consenties pour moi.

ACRONYMS

6LBR	6LowPAN Border Router.
6LoWPAN	IPv6 over Low Power Wireless Personal Area Networks.
ACC	Aggregate-based Congestion Control.
AD	Attack Diagnosis.
AITF	Active Internet Traffic Filtering.
AS	Autonomious System.
C&C	Command and Control.
CoAP	Constrained Application Protocol.
COSSACK	COordinated Suppression of Simultaneous Attacks.
CSS	Chirp Spread Spectrum.
CTS	Clear To Send.
DDoS	Distributed Denial of Service.
DEFCOM	Defensive Cooperative Overlay Mesh.
DTLS	Datagram Transport Layer Security.

EPMAP	End Point Mapper.
FTP	File Transfer Protocol.
HNAP	Home Network Administration Protocol.
HTTP	HyperText Transfer Protocol.
ICMPv6	Internet Control Message Protocol versiion 6.
IID	Interface Identifier.
IMEI	International Mobile Equipment Identity.
IoT	Internet of Things.
IPv6	Internet Protocol version 6.
ISM	Industrial, Scientific and Medical.
ISN	Initial Sequence Number.
LoRa	Long RAnge.
LoRaWAN	Long RAnge Wide Area Network.
M2M	Machine-to-Machine.
MAC	Medium Access Control.
MITM	Man In The Middle.
MQTT	Message Queuing Telemetry Transport.
MTD	Moving Target Defense.
MTU	Maximum Transmission Unit.
NAT	Network Address Translation.
OS	Operating System.
OUI	Organizational Unique Identifier.

PAD	Parallel Attack Diagnosis.
PKI	Public Key Infrastructure.
PPTP	Point-to-Point Tunneling Protocol.
RFID	Radio-frequency Identification.
RIR	Regional Internet Registry.
RPL	Routing Protocol for Low-Power and Lossy Networks.
RTS	Request To Send.
RTT	Round-Trip Time.
SDN	Software-Defined Networking.
SGX	Software Guard Extensions.
SIFF	Stateless Internet Flow Filter.
SIP	Session Initiation Protocol.
SMB	Server Message Block.
SSDP	Simple Service Discovery Protocol.
TCP	Transmission Control Protocol.
TFTP	Trivial File Transfer Protocol.
TLS	Transport Layer Security.
TSCH	Time-Slotted Channel Hopping.
TTL	Time To Live.
TVA	Traffic Validation Architecture.
UDP	User Datagram Protocol.
UPnP	Universal Plug and Play.
VPN	Virtual Private Network.

WSN Wireless Sensors Network.

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

CONTENTS

INTRODUCTION

The rise of the Internet of Things has created the appearance of vulnerable devices connected 24/7 [GWZ, Kol+17]. These devices often have minimal resources, are often neglected by their users (not updated regularly), are not updated and use default passwords for their applications. Apart from the lack of updates, the vulnerabilities present in IoT devices are also due to their limited resources to implement existing security solutions of traditional devices. As a result, they fall prey to bot-masters and are used to perform distributed attacks. A relatively recent example are the Mirai attacks [Ant+17, Kol+17]. Record figures in terms of attack power have emerged in recent years. Protecting IoT devices is not easy with the currently existing solutions (firewall, IDS and proxy). Indeed, many systems for detection and mitigation have been proposed in the literature [YPS04, AC09, YWA05, YWA08, LYL08], but they are not suitable for IoT and they are often hard to implement. That is why mitigation of large distributed attacks is still a problem today since it requires the victim to deploy very significant resources.

Depending on their position inside the network, we have different methods for detecting and mitigating the attack's effect on their victims. However, these methods have significant limitations and often require changes to the core of the Internet, namely routing, to work. Hybrid methods located at multiple locations in the network allow the use of techniques specific to their position and combine their strengths.

Although used in many attacks against servers, Internet of Things (IoT) devices are not immune to the attacks themselves. Primarily because of their low resources, they are more easily damaged when attacked.

In this thesis, we propose a solution to dynamically control access to devices' resources based on their characteristics. This control allows to prevent the misuse of their resources in a first step. In a second step, it allows the detection of an attack early and stops it at the source of the attack via a distributed system. The advantage of this mitigation at the source is that it limits the resources needed by the victim to stop large scale attacks.

The system we propose is composed of three different parts, each with specific roles. Any communication between user devices is first validated by our system at the destination to control the flow of information to the protected devices. In this way, we can limit incoming traffic depending on the current state of the protected device. Second, once an attack is detected, the system present at the attacker side stops the malicious traffic. The larger the number of attackers, the more resources are saved by the victim to protect itself. Finally, a hierarchical infrastructure enables the efficient communication between the different sub-systems. Our system can easily be set up by institutions that want to collaborate and share resources without changing their environment too much, allowing them to specify security policies. Federations of IoT devices can thus be set up and easily maintained and administered.

CONTRIBUTIONS

Our first contribution is a study on the prevalence of IoT protocols in telescope and honeypot measurements. We had noticed a lack of information related to IoT attack signatures and the usage of IoT protocols by attackers. Most existing studies only look at telnet traffic. In Chapter 2, we have collected data from a network telescope and also adapted and operated several honeypots. The collected data has allowed us to better understand how attackers target IoT devices and what protocols they use.

IoT devices are designed to exchange information automatically. Consequently, their interaction with humans is limited and they are deployed in such a way that they are easily forgotten by network managers and rarely receive updates and patches against vulnerabilities. Network scans can help to find deployed devices but they can easily overwhelm resource-constrained devices when done too aggressively. In Chapter 3, we present an approach that allows us to find and identify resource-limited devices more than 35% faster than traditional scans and with a discovery rate of at least 95%.

Finally, in Chapter 5 and Chapter 6 we have developed a system to protect IoT devices as well as any system that can be attacked by them once they are compromised. Due to the resource constraints of many IoT devices, our system consists of collaborating middleboxes that can protect networks containing IoT devices. The middleboxes can be easily integrated into existing networks and do not require modifications of the IoT devices. Furthermore, they are able to stop attacks at the source without consuming the target's resources.

PUBLICATIONS

The three main publications and their extended versions discussed in this thesis are the following.

WORKSHOP PUBLICATIONS

[MES17] L. Metongnon, E. C. Ezin, and R. Sadre. "Efficient probing of heterogeneous IoT networks." In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 1052–1058.

[MS18a] L. Metongnon and R. Sadre. "Beyond telnet: Prevalence of IoT protocols in telescope and honeypot measurements." In: *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*. ACM. 2018, pp. 21–26.

CONFERENCE PUBLICATIONS

[MSE19] L. Metongnon, R. Sadre, and E. C. Ezin. "Distributed Middlebox Architecture for IoT Protection." In: *Proceedings of the 15th International Conference on Network and Service Management*. CNSM. 2019.

JOURNAL PUBLICATIONS

[MS18b] L. Metongnon and R. Sadre. "Fast and efficient probing of heterogeneous IoT networks." In: *International Journal of Network Management* 28.1 (2018), e1997.

[MS19] L. Metongnon and R. Sadre. "Prevalence of IoT Protocols in Telescope and Honeypot Measurements." In: *Journal of Cyber Security and Mobility* 8.3 (2019), pp. 321–340.

In addition to those, another paper has been written during this thesis. This paper deals with reputation rating of BGP links and can help for the detection of large-scale attacks on Internet. While in the wider scope of this thesis due to its detection ability, this paper is left out of this thesis because of the difficulty to use it in real-time scenarios.

CONFERENCE PUBLICATIONS

[AML17] H. A. Arouna, L. Metongnon, and M. Lobelle. "Reputation Rating Algorithm for BGP Links." In: *International Conference on e-Infrastructure and e-Services for Developing Countries*. Springer. 2017, pp. 352–357.

OUTLINE

The core of this thesis is divided into two parts. In Part I, Chapter 1 presents the Internet of Things and the benefits of this technology. We give an introduction to important concepts related to IoT devices and the security concerns that follow from their omnipresence. Then in Chapter 2, we present the current state of threats against the IoT, the different protocols involved, the types of communication, and the areas where we find high rates of malicious users. Chapter 3 provides a scan-based approach to efficiently detect IoT devices in heterogeneous networks despite the size of the IPv6 address range. We show that our approach reduces the time to discover devices even when scanning slow constrained networks. This can help network administrators to locate forgotten devices in their networks.

Next, Part II contains chapters dedicated to the detection and mitigation of attacks from and against IoT devices. First, Chapter 4 presents the state of the art in the detection and mitigation of high-volume denial-of-service attacks. We argue that this type of attacks are particularly dangerous in the context of IoT. We classify existing defense methods depending on their position in the network and discuss possible and feasible solutions to stop such attacks. Then, in Chapter 5, we present a distributed middleboxes architecture for the protection of IoT

devices. We demonstrate its capability to detect attacks at the destination and to mitigate them at the source to save the target's resources. Chapter 6 presents the hierarchical communication system used in our distributed solution that allows, amongst others, its components to exchange information on ongoing attacks and to initiate mitigation actions. We evaluate the scalability of our design and the effectiveness of its collaborative aspect using experiments in a network emulator.

After these two parts, we give a conclusion of our work and some perspectives for future research activities.

Part I

UNDERSTANDING THE STATE OF IOT SECURITY

1

INTERNET OF THINGS AND CHALLENGES

Progress is not an illusion, it happens, but it is slow and invariably disappointing.

—George Orwell

INTRODUCTION

The term IoT was likely introduced by Kevin Ashton in a presentation in 1999 where he linked the new Radio-frequency Identification (RFID) technology to the idea of a world where large numbers of physical objects can be tracked and managed by computers [Ash]. That original vision has evolved and by today, IoT has become an umbrella term for various kinds of networks of devices exchanging information such as sensor data without human intervention.

The success of the IoT has been the result of advances in the past two decades in several fields, such as Wireless Sensors Network (WSN), Ubiquitous Computing, and data mining. Low-power electronics have enabled the development of cheap and small energy-efficient embedded systems and radio interfaces. New network protocols for low-power short-range and long-range communication have been designed that are robust to noise and interferences in license-free radio bands and allow, unlike RFID, bi-directional active communication. As on the data side, cloud computing and light-weight Machine-to-Machine (M2M) protocols have allowed to aggregate and process large amounts of sensor data in real-time.

From a security point of view, the IoT faces many challenges. Compared to modern PCs and smartphones, IoT field devices are significantly constrained in terms of computation and communication capabilities, resulting in old security issues that have been considered as "solved" in conventional ICT systems to resurface. For example, small IoT devices often do not have the necessary resources to host sophisticated security software, such as antivirus software. Furthermore, although secure communication is available on many IoT platforms it is not always used in practice because of the associated overhead (key management etc.). IoT devices also share many security issues with traditional networks, such as the usage of weak passwords. Last but not least, privacy is a big concern in IoT, since many IoT applications more or less directly handle highly privacy-sensitive information, such as user habits, location information, or health-related information.

It should be noted that the exact definition of IoT devices varies among authors. Some of them consider as IoT devices only small embedded systems equipped with sensors. We use a broader definition here and also include devices such as home routers and Smart TVs that do not collect sensor data but are constrained, networked and can

operate without human intervention, sometimes without the knowledge of their owners.

In this chapter, we will give a quick introduction to important concepts related to IoT devices and their communication networks. We will also discuss security threats directed against or originating from IoT devices.

1.1 OPERATING SYSTEMS FOR IOT DEVICES

Different operating systems are available for IoT devices, depending on the available computation and energy resources. On one end of the range, we find extremely resource-constrained devices with just a few bytes of RAM that rely on primary batteries or energy harvesting. Such devices run very specialized operating systems or, in some cases, no operating system at all. On the other end, we see stationary systems, such as Smart TVs that are connected to the main grid. Those devices are sometimes powerful enough to run lean versions of a general-purpose operating system like Linux.

Several popular operating systems for resource-constrained IoT devices were originally developed for Wireless Sensor Networks. The first such constrained OS was Tiny OS [Lev+05]. Its development started in 1999 under the BSD license. It has been designed for embedded systems with low power consumption and limited resources and only needs 400 bytes of RAM. A typical Tiny OS application takes around 15 KBytes of storage. Tiny OS applications are programmed in NesC (a subset of C) following a component-based and event-driven programming model. The current version Tiny OS 2.1.2 comes with support for cooperative threads and a low-power Internet stack called BLIP that implements Internet Control Message Protocol versiion 6 (ICMPv6), Internet Protocol version 6 (IPv6), User Datagram Protocol (UDP), Transmission Control Protocol (TCP), IPv6 over Low Power Wireless Personal Area Networks (6LoWPAN), Routing Protocol for Low-Power and Lossy Networks (RPL) and Constrained Application Protocol (CoAP).

The younger operating systems Contiki [DGV04] and RIOT [Bac+13] target the same device class as Tiny OS. Contiki was first presented by the Swedish Institute of Computer Science (SICS) in 2004. Unlike in Tiny OS, applications are written in standard C. Contiki supports event-driven programming as well as cooperative multi-threading through light-weight stack-less threads that can be interrupted by the OS and device drivers. RIOT follows a more modern design with priority-based

scheduling for user threads and a programming interface that is (partially) POSIX compliant. Both RIOT and Contiki support lightweight network and application protocols, such as 6LoWPAN, RPL and CoAP.

On less resource-constrained devices, general-purpose operating systems can be deployed. Many home routers, Smart TVs, and home automation hubs run Linux. The devices of the Raspberry Pi series can run Linux as well as Microsoft's Windows IoT (formerly Windows Embedded). Nowadays, such systems are even used in domains that were traditionally dominated by simple and reliable embedded systems, for example in industrial control systems [Kun].

1.2 WIRELESS COMMUNICATION NETWORKS FOR IOT DEVICES

Similar to operating systems, different network technologies have been developed for various use cases. Again, many of them pre-date the IoT and were already in use in WSN and industrial sensor and control networks.

For *long-range* wireless communication, GSM/GPRS was a popular choice for many years. However, it's being slowly phased out in many countries, forcing users to look for alternatives. Its modern successor LTE provides high bandwidth and low latency, but it is too complex and energy-hungry to be directly used in small and resource-constrained devices. Several LTE-based technologies for IoT have been proposed by researchers and the 3GPP. Operators have started to deploy two of them recently: LTE-M and NB-IoT. LTE-M is close to the original LTE standard and shares, to a certain degree, its complexity and power consumption. NB-IoT has a much lower power consumption, but also lower bandwidth (100 kbit/s for NB-IoT and 384 kbit/s for LTE-M)[1] and higher latency than LTE-M [Rat+16, Che+17, SWH17]. Both technologies compete with the new 5*G* standard that claims to reduce power consumption by a factor of 100 compared to the original LTE standard. The first commercial launches began in 2018 resp. 2019 in the USA, Germany, South Korea and China (see for example [Xin]), therefore there is not much practical experience with large-scale IoT deployments using 5G.

LTE-based networks for IoT have the advantage that the existing LTE infrastructure can be used. However, it requires that all devices

1 https://www.pietcallemeyn.be/sentineo/2018/04/30/nb-iot-or-lora-which-technology-to-choose-for-your-next-iot-device/

with LTE radio interfaces have to be standard compliant. A different approach is followed by Long RAnge (LoRa). Its design goals are to eliminate repeaters and increase the battery lifetime of the devices [Wix+16, KIAM17, BKL16, CS+17] by using Chirp Spread Spectrum (CSS) modulation techniques that are very robust to noise and interferences even at very low signal strengths. It uses unlicensed Industrial, Scientific and Medical (ISM) radio bands at 433 MHz, 868 MHz or 915 MHz and a single antenna can cover a range of 2 to 5 km in urban areas, 15 km in sub-urban and 45 km in rural areas. The transmission power in Europe is limited to 14 dBm. LoRa PHY defines the physical layer [N+18, BKL16] while Long RAnge Wide Area Network (LoRaWAN) provides a Medium Access Control (MAC) layer with a throughput per device that can go down to around 7850 bytes/hour depending on the number of active devices and the signal quality [Ade+17].

When it comes to *short-range* communication, an obvious choice is IEEE 802.11 (WiFi). However, while being fast and flexible, IEEE 802.11 is not very suitable for resource-constrained devices due to its complexity and power consumption. IoT devices using WiFi are therefore typically devices connected to the main grid, for example, Smart TVs. An alternative is the IEEE 802.15.4 standard [Bar+07] that defines short-range low-power communication with data rates ranging between 20 to 250 kbit/s depending on the used frequency bands [Mon+07]. The maximum frame size is 127 bytes, with just 72 to 116 bytes of payload available after link-layer framing, addressing, and optional security [HT11].

A desired property for IoT devices is addressability, i.e. the ability of a device (or the services it provides) to be found and being the target of message exchanges. Indeed, we are assuming in this thesis that IoT devices are reachable from outside their local network. IPv6 with its large 128-bit addresses quickly attracted the interest of researchers since an IPv6-based protocol stack would not only provide addressability but would also greatly simplify the exchange of messages between IoT devices and hosts in IP networks. While LTE and WiFi naturally support IP protocols, running IPv6 over low-power networks such as LoRa or IEEE 802.15.4 is much more challenging because IPv6 requires a minimum Maximum Transmission Unit (MTU) of 1280 bytes and does not support fragmentation [Dee98]. In constrained networks with smaller MTUs, a compression and encapsulation layer is needed, as provided by 6LoWPAN [Mon+07] for IEEE 802.15.4 and other networks

[TBS17]. Figure 1.1 depicts a comparison between the IP and 6LoWPAN protocol stacks.

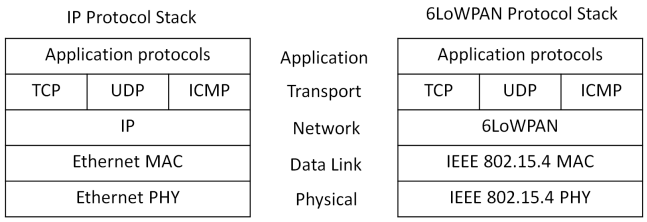

Figure 1.1: IP and 6LoWPAN protocol stacks (after [SB11])

In practice, a 6LowPAN Border Router (6LBR) is required to connect a 6LoWPAN network to a normal IPv6 network. Typically, the border router has also other tasks, including the generation of a compact 16-bit Interface Identifier (IID) for the nodes, the (de-)fragmentation of packets, and the management of the routing between nodes, for example by using RPL [Win12]. RPL allows multi-hop routing inside 6LoWPAN networks and defines control messages that are used by the border router and the nodes to construct and maintain a logical routing graph.

For the sake of completeness, we may underline that various other wireless protocol specifications exist for constrained devices, such as Thread [UTL18, KKC19], ZigBee, Z-Wave, INSTEON and Wavenis [GP10]. Most noteworthy, like 6LoWPAN, ZigBee is based on IEEE 802.15.4. ZigBee is a specification of protocols defined by the ZigBee alliance[2] for embedded applications requiring low-power consumption and low data-rates. The specification comprises network protocols as well as high-level application protocols based on a component model. However, we will use 6LoWPAN for constrained devices in the following because it is well supported by tools and IoT software/hardware, and because hosts in a 6LoWPAN network can easily inter-operate with hosts on the Internet.

1.3 IOT APPLICATION PROTOCOLS

In this section, we will discuss popular IoT protocols used with constrained devices, namely CoAP, Message Queuing Telemetry Transport (MQTT), Universal Plug and Play (UPnP), telnet and HyperText Transfer Protocol (HTTP).

2 http://www.zigbee.org/

CoAP is a protocol for M2M communication, running on top of UDP [SHB14]. Encryption with Datagram Transport Layer Security (DTLS) is optional. CoAP servers, i.e. the IoT devices, provide a REST interface that can be used by applications to retrieve sensor data or to change configuration parameters. Like in HTTP, resources are addressed by URIs. Clients can obtain a list of the available resources of a CoAP server by requesting the resource `/.well-known/core`.

MQTT [BG14] is another protocol for M2M communication. In contrast to CoAP's client-server model, MQTT uses a publisher-subscriber model where a *broker* server forwards the data received from the publishers (i.e. the IoT devices) to the subscribers. MQTT uses TCP as transport protocol with optional usage of SSL. Most noteworthy, connections to the broker are always initiated by the publishers and subscribers. In theory, this means that IoT devices using MQTT do not need to accept incoming connections and only the broker is of interest for MQTT-based attacks. MQTT-SN is a simplified version of MQTT that can directly run on top of a link protocol or over UDP and therefore is more suitable for constrained networks. It requires a gateway that translates MQTT-SN messages to MQTT.

UPnP is a set of network protocols used in home automation to discover the presence of devices in a network and to establish services, such as data sharing and streaming [JW03]. UPnP is enabled by default on many home network devices like home routers, webcams, printers, and cameras. UPnP's Simple Service Discovery Protocol (SSDP) uses UDP.

Telnet is a text-oriented interaction protocol based on TCP providing a bidirectional communication to a device. It is typically used to send commands to a terminal service. While not used anymore on servers and workstations because of security concerns (message exchanges are not encrypted), it is still very popular on embedded systems because of its simplicity. The access to the command line is granted after authentication, however owners of IoT devices often do not change the default credentials as demonstrated by the Mirai attacks (see Section 1.5).

Finally, *HTTP* is used to exchange hypertext messages. It is relatively heavy (the header of an HTTP message can grow to several hundred bytes) and therefore mostly used on less-constrained devices, such as routers, that provide a web interface for configuration and management or a REST interface for their services. HTTP is also the underlying protocol of many other application protocols. For example, the Home Network Administration Protocol (HNAP) by Cisco is used for the

management of network devices. A device supporting HNAP will reply with a valid response to HTTP requests for the resource `/HNAP1` [Sys09].

1.4 SECURITY CONCERNS

All devices connected to the Internet are under threat from worms, viruses, Trojan programs, and malicious users. Whether they are devices for private individuals or professionals, they are regularly attacked. The motivations for security attacks are usually [Wea+03]:

- Disruption of a business continuity;
- Information theft;
- Data manipulation;
- Propaganda;
- Damaging reputation;
- Taking revenge

An attack scenario often begins with a reconnaissance phase in which the attacker seeks to gather information about the target to plan the attack phase. We have two types of reconnaissance, namely active and passive reconnaissance. The latter consists in acquiring information without interacting with the target, such as searching online records or using a packet sniffer to collect traffic information. Active recognition consists of direct interaction with the target [UP13]. The second phase is often an analysis phase where the attacker analyzes the network, tests ports to find IP addresses used in the target network, services running, operating system details, device type and possibly system uptime. Many tools can be found online to automate this phase and perform it fast and also stealthily [Hoq+14]. The third and fourth phases consist mainly of obtaining and maintaining access to the target. Through privilege escalation, password hacking, buffer overflow and session hijacking, the attacker can gain access and then use backdoors, a Trojan horse and data manipulation to maintain and secure that access [Sim+14]. The final and last phase consists of erasing the traces of misdeeds to avoid suspicion, hide malicious acts but above all hide the identity of the attacker.

Naturally, the above also applies to IoT devices. A well-known example is the Mirai malware [Pa+15] and its variants that target IoT devices

and infect them with botnet software. However, the IoT also poses new challenges in cyber-security for various reasons.

First, the heterogeneity of deployed hardware and software is increased by the IoT. While the classical Internet confronts us with two or three dominating operating systems, CPU families and network technologies, IoT devices depict a much larger variety, as shown in the previous sections. This heterogeneity makes the deployment of IoT security solutions harder.

Second, some devices are deployed in such a way that they are easily forgotten by users and network managers, or they are difficult to update compared to servers or end-user personal computers [Yu+15]. This has resulted in several successful attacks that used default passwords or well-known exploits against home routers and CCTV cameras [GWZ, Con, Cid].

Third, the large number of IoT devices can also pose a threat to *other* Internet hosts. Although IoT devices are less powerful than, for example, a desktop PC or a web server, they can be misused to launch effective DDoS attacks against other Internet participants [DD+17, KKS17, GWZ, Con, Cid].

Finally, due to their lack of resources, i.e. small memory, limited computation resources and often limited battery, IoT devices are sensible to resource starvation attacks. This also applies to networking resources. In networks for constrained devices, such as IEEE 802.15.4, a single powerful device can easily jam communication channels (When Time-Slotted Channel Hopping (TSCH) is used, the attacker needs to target all available channels at the same time) and consume the available bandwidth, and thus effectively perform a Denial of Service attack that can affect many devices at the same time.

Unfortunately, constrained IoT devices often do not have enough resources to carry the overhead of an intrusion protection system or complex security protocols. This is all the more deplorable when we consider that IoT devices often handle sensitive information, as already noted nearly two decades ago in the context of Ubiquitous computing [Lano1]. Often, the user is not even aware that sensor data is collected by a third party or can be used to identify them [Nae+17]. Due to the limited resources of the IoT devices, encryption is not always used so the data is sent in clear text and can be accessed or tempered by malicious users.

The absence of strong security has also led to attacks that exploit vulnerabilities in communication protocols. For example, the routing

protocol RPL is subject to *spoofing attacks* [WRV13] where a malicious node sends wrong routing information that disrupts the normal operation of the network or that causes packets to be routed through the malicious node where their content can be eavesdropped or altered. Another type of attack is the *Sybil attack*, where nodes take multiple identities in a network. This is a huge threat to routing protocols or any system using geographical data to operate [SN11]. In multi-hop IoT networks, the majority of routing protocols use some kind of distance or position information to choose the best route. By sending different geographical data to its neighbours, an attacker can disrupt routing or cause IoT nodes to waste energy. Communication can be also eavesdropped or disturbed with *Wormhole* attacks, where an attacker creates a tunnel between two distant devices in a network [HPJ06, WW07], effectively reducing the number of hops between them. In wireless adhoc networks that use hop counts as routing metric, this causes neighbour nodes to choose the tunnel controlled by the attacker as preferred path.

1.5 THE MIRAI MALWARE

In the previous section, we discussed some important security concerns with IoT devices. Among the examples that we gave, one is particularly interesting: the *Mirai* malware. Mirai is a malware for IoT devices discovered in August 2016 by researchers at *MalwareMustDie*[3]. It is standing out among the large number of existing malwares because of its effectiveness despite its simplicity. Mirai scanned the Internet for devices with poorly secured telnet services, infected them with bot malware and then used them to perform Distributed Denial of Service (DDoS) attacks[4] against other hosts on the Internet. Its success was based on the enormous amount of devices it was able to infect. As it turned out, most of those devices were IoT or IoT-like devices, such as IP cameras and home routers, that provided telnet access with default credentials set by their manufacturers.

The success of Mirai has demonstrated one of the greatest dangers posed by IoT: Although IoT devices are less powerful than desktop PCs or servers, an attacker who manages to get even a small part of the

3 https://www.malwaremustdie.org/

4 DDoS attacks are attacks where a large number of devices perform an attack against a victim to exhaust its computation or network resources and disrupting its normal operation. Nowadays, most DDoS attacks are volumetric attacks attempting to disrupt normal traffic of a targeted server or network.

billion existing poorly protected devices under their control can carry out very effective attacks against other Internet hosts. For this reason, one of the goals of the security architecture that we design in Part II is to prevent or mitigate attacks against IoT devices and servers *as well as attacks originating from them*.

The initial version of Mirai scanned the Internet for devices with telnet default login credentials and then exploited vulnerabilities in old Linux distributions for privilege escalation. Mirai's main target were video security cameras and home routers connected to the Internet. Its source code was released on September 30, 2016. Since then, many attacks have been attributed to it, among then:

- an attack on the website of a security researcher KrebsOnSecurity on September 20, 2016, with a bandwidth of around 600 Gbps;
- an attack on OVH with a bandwidth of about 1 Tbps on September 20, 2016;
- an attack on the DNS Dyn server on October 21, 2016, impacting many sites including Paypal, Github, Amazon, Spotify and others. The attack rate was about 1.2 Tbps;
- an attack on 27 November 2016, resulting in the loss of Internet access for nearly 900,000 German customers of Deutsche Telekom;
- an attack resulting in the loss of Internet access for 55,000 TalkTalk customers on November 27, 2016;

After the code was published, researchers discovered how simple the attack procedure was. First, Mirai scans all IPv4 addresses to find new bots[5] while avoiding certain subnetworks belonging to Hewlett-Packard, General Electric, and the US Department of Defense. The target of the scans are the telnet ports 23 and 2323. This protocol is still used by many devices, such as home routers, IP cameras, etc. Since the default password is often not changed by the owners, Mirai can simply

5 Bots are compromised devices controlled by a malicious user called botmaster. A device is first infected with malware and can be then used without the consent of the actual owner. The general attack phases presented in Section 1.4 are the roadmap for a botmaster to recruit new bots for their botnet. When the botnet is new, these phases are performed by the botmaster, but this responsibility is later assigned to the bots themselves. Once a botnet has reached a sufficient size, it can be used for DDoS attacks, spamming and phishing attacks.

use a list of known login/password combinations to take control of them.

A distinct characteristic of the original version of Mirai malware is the fact that it sends very efficiently "stateless" TCP SYN packets with identical Initial Sequence Number (ISN)s to the target IP addresses [Dul+17]. The IP addresses of compromised devices are sent to a Command and Control (C&C) server for later use. After obtaining access, the malware binary is downloaded on the compromised device and the telnet service is patched to prevent competing malwares to login. The new compromised device in turn starts scanning the Internet to find new targets and thus enlarge the botnet.

The release of the Mirai source code triggered a wave of many clones using the same general procedure. *Hajime* is a Mirai-like botnet using BitTorrent's DHT protocol for peer discovery and the uTorrent Transport Protocol (uTP) for data exchange [EP16]. Its purpose is still unknown. *BrickerBot* is a more exotic malware because it makes infected devices completely unusable. In that way, the authors wanted, so they claim, to protect the Internet from unsecure devices. BrickerBot contains attack vectors for telnet, SSH, HTTP, HNAP and SOAP [Ken17].

SUMMARY

Internet of Things is a technology brought to enhance every domains of human needs. IoT devices are used in automation, industrial manufacturing, logistics, transportation of people and goods, Smart Cities and domotics. The devices are constrained in terms of computation and communication capabilities, which has led to the development of operating systems, network technologies, and communication protocols that are very different to what we find in conventional networked systems.

These differences result in many security issues. Resource constraints and the heterogeneity of deployed hardware and software prevent the deployment of state-of-the-art solutions, such as secured protocols and intrusion detection systems. Access control is also poorly implemented on many devices or, where present, is easy to circumvent. As a result, IoT devices are not only easy to attack but they can be also misused to perform attacks against other hosts.

2

PREVALENCE OF IOT PROTOCOLS IN TELESCOPE AND HONEYPOT MEASUREMENTS

Why live? Life was its own answer. Life was the propagation of more life and the living of as good a life as possible.

—Ray Bradbury, *The Martian Chronicle*

INTRODUCTION

One of the most successful attacks misusing IoT devices started with an infection through telnet. Therefore, the majority of recent publications in this area have focused on this protocol. However, IoT installations often also involve other protocols, such as CoAP, MQTT, and the protocols used by UPnP for configuration and service discovery. These protocols are specialized lightweight protocols and provide by default only little security. Due to the enormous "success" of telnet-based attacks, the prevalence of these protocols in attack traffic has not been widely studied.

The goal of this chapter is to learn more about the usage of IoT-related protocols in attack traffic. We present a measurement study on attacks against common protocols used by IoT devices. To this end, we use data obtained from a large /15 network telescope[1] operated by SurfNet[2] and three honeypots with 15 IPv4 addresses. We find, not unexpectedly, that telnet-based malware is still widely used. Furthermore, we observe that the infected devices are employed not only for DDoS attacks but also for crypto-currency mining. However, we also see that attackers are looking for IoT installations using the above mentioned other protocols. Such attacks are much rarer than telnet-based ones, but it can be expected that their frequency will increase in the near future in light of the increasing popularity of IoT and M2M communication.

The rest of this chapter is organized as follows: We discuss related work in Section 2.1. We describe the experimental setup in Section 2.2 and discuss results in Section 2.3.

2.1 RELATED WORKS

The usage of network telescopes and honeypots for security research has a long tradition. By design, nearly[3] any traffic reaching them is malicious (e.g. a network scan), unintended (e.g. caused by misconfiguration), or the result of attacks with spoofed IP addresses, the so-called *backscatter* traffic. A general discussion on telescopes and honeypots

1 A network telescope is a set of routed IP address space that allow to observe malicious traffic [Moo+04]

2 https://www.surf.nl/en/about-surf/subsidiaries/surfnet

3 An exception is, for example, the traffic from benign network scans used by researchers for Internet-wide studies.

is out of the scope of this chapter and we refer the reader to seminal papers such as [Moo+06]. Instead, we will focus on IoT in the following.

The main intention of attacks against IoT devices is to turn them into bots for botnets. Kolias *et al.* [Kol+17] point out many reasons why IoT devices are interesting for botnets, such as the fact that they are online 24/7, their lack of security features, and their poor maintenance. Besides, their capability to launch powerful DDoS attacks has been underestimated for a long time and therefore their protection has not been given high priority by network administrators. The sheer number of available vulnerable devices allows to perform highly distributed DDoS attacks. This is a major difference to most non-IoT based DDoS attacks; in fact, Krämer *et al.* reported in [Krä+15] that major amplification-based DDoS attacks were short-lived and 96% came from single sources.

IP-based cameras are attractive victims for botnet operators because they often have good network connectivity and are poorly secured. Several powerful DDoS attacks performed by such cameras have been reported [GWZ, Con, Cid]. Pa Pa *et al.* [Pa+15] implemented a honeypot for telnet-based attacks and identified at least four distinct DDoS malware families supporting as many as nine different CPU architectures. The *Mirai* malware and alike described in Section 1.5 are the primary suspects for the above attacks.

Our findings confirm several of the observations made in the above publications, respectively update them. As an example for the latter, we find that only 10% of the Mirai-like telnet traffic respects the ISN pattern, which indicates that attackers have adapted the original source code of Mirai [Kre16]. More important, there are, to our knowledge, no existing studies on the prevalence of attacks against the other IoT-related protocols discussed in this chapter.

A completely different usage of network telescopes can be found in [Sha+18]. Shaikh et al. use a combination of passive and active measurement to discover and characterize unsolicited IoT devices. They applied the Context Triggered Piecewise Hashing (CTPH) algorithm to the malicious device's data using shodan and censys databases information through the devices IP addresses coupled with passive measurement session information.

2.2 EXPERIMENTAL SETUP

In this section, we describe our measurement setup to investigate the presence of IoT-related attacks in the Internet. The telescope using its large address range gives a better insight into what to expect in the number of attacks on a large scale, while our honeypots used 15 IP addresses (see Section 2.2.2) for the detailed analysis of the individual attacks. The telescope and the honeypots are in different addresses spaces in the following experiments.

2.2.1 *Telescope*

SURFnet is a company of SURF, the collaborative ICT organization for Dutch education and research. Its mission is to support innovation through their infrastructures. We use the tool *eemo*[4] to access their /15 telescope. We capture 10 minutes of packet data every hour from 2017-09-25 to 2018-02-28 with an interruption in October. Since the telescope is passive and therefore does not respond to incoming traffic, we use it only to record connection attempts as well as to identify the most used protocols and targeted ports.

2.2.2 *Honeypots*

A honeypot is a software mimicking specific services or devices and is used to attract malicious traffic. In our experiments, we are interested in IoT-related services and protocols. SURFnet provides us 15 IP addresses (separate from the IP of the telescope) that we use to deploy the three honeypots *Cowrie*, *dionaea* and *HoneyPy*.

Cowrie[5] is a medium interaction honeypot impersonating a server with SSH and telnet service and easily cracked login credentials. Attackers who connect to the honeypot see a fake file system. All tty message exchanges and attempts to download and install malware are recorded. We use version 1.1 (we started with commit 3d12c8c54 and upgraded to 4fofc85e02) of the honeypot.

Dionaea[6] is also a medium interaction honeypot. Similar to Cowrie, it records all message exchanges. We use version 0.6.0 (we started with

4 https://github.com/SURFnet/eemo
5 https://github.com/micheloosterhof/cowrie
6 https://github.com/DinoTools/dionaea

commit ac971c3ab and upgraded to 02492e2b973) with the following services: End Point Mapper (EPMAP), File Transfer Protocol (FTP), HTTP, Memcache, MQTT, MSSQL, MySQL, Point-to-Point Tunneling Protocol (PPTP), Session Initiation Protocol (SIP), Server Message Block (SMB), and UPnP.

HoneyPy[7] is a low-interaction honeypot. We use its services Trivial File Transfer Protocol (TFTP), TR-069.1 and TR-069.2 (a management protocol on port 5555 and 7574 used by some ISPs to configure the modems of their subscribers), and TelnetIoT (telnet on port 2323).

Since no module existed to emulate a CoAP server, we initiated a Master's thesis on that topic. **Dany Yarakou**, a student from Benin, chose HoneyPy to implement a first prototype with limited interactions. He then developed a module for Cowrie that has more features and a more advanced logging system. We used his HoneyPy version to emulate a CoAP server providing the following virtual resources [8]:

- a simple counter resource;
- a resource returning the current time;
- resources sending large data blocks;
- a separate large resource that sends intermediate ACKs to indicate that the processing of the request is delayed;
- the `/.well-known/core` resource that gives a list of all the existing resources on the server.

We did not operate the honeypots continuously over the measurement period. This was partially caused by technical difficulties that forced us to shut them down for maintenance. For example, the interruption in January 2018 was caused by an update to mitigate Meltdown/Spectre [Con+18].

To get more information about the hosts accessing the honeypots, we used Shodan[9], a well-known search engine for Internet-connected devices. Shodan periodically scans IP addresses in the Internet and collects information about the host behind the addresses, including information on their location, available services, and operating system. The Shodan databases can be queried programmatically through an API or manually through a web user interface.

7 https://github.com/foospidy/HoneyPy
8 https://github.com/DanMistyk/HoneyPy
9 https://www.shodan.io

2.3 RESULTS

We present the results of our analysis of the data obtained from the telescope (Section 2.3.1) and the three honeypots (Section 2.3.2).

2.3.1 *Telescope results*

Figure 2.1 shows the number of packets that the telescope received per day. Due to technical issues, no data was collected in the period from 2017-10-05 to 2017-11-01, on 2018-01-20, 2018-02-14, and 2018-02-19. Apart from that, the setup was not changed during our experiments.

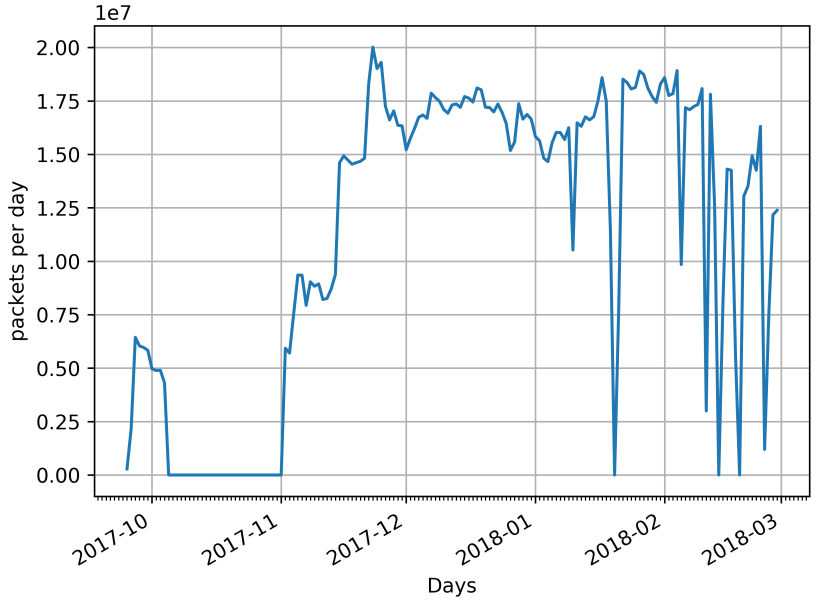

Figure 2.1: Number of packets per day reaching the telescope. Note the scaling factor of 10^7 for the y-axis.

With 96.1% of all observed packets, TCP is the dominant protocol, followed by UDP (2.8%) and ICMP (1.1%).

Figure 2.2 shows the disappearance of the ICMP traffic after 2017-12-08 without any specific reason known to us. The ICMP traffic was already very shallow on the telescope and present only on 47 days during the measurement. We identify 29 different universities probing with 137,760 packets send covering only 3.53% of the total probes received (Table 2.1). We deduced that only a small part of probes we

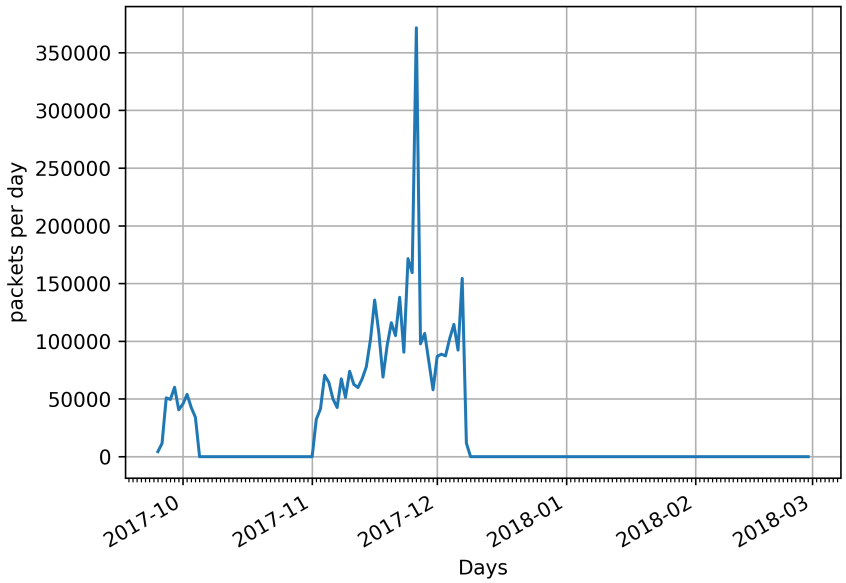

Figure 2.2: Number of ICMP packets per day reaching the telescope.

receive are for academic tests purposes. The University of Michigan is, however, the 11 top AS in total probes quantity with 74,061 on the period, that means the quantity is not negligible despite the percentage.

Figure 2.3 shows the correlation between Table 2.1 and Table 2.2. We see that scans coming from SuperNetwork s.r.o. and HLL LLC are coming from only one IP address followed by Los Nettos with 2 IP addresses, QuadraNet, Inc with 4 IP addresses and Hangzhou Alibaba Advertising Co.,Ltd. with 7 IP addresses send many ping request. We can assume that (a) we have powerful host responsible for all the scanning traffic coming those AS, (b) The addresses of these AS are spoofed so they are under reflections attacks and (c) We have many hosts in those AS attacking shield behind a Network Address Translation (NAT). The others AS No.31,Jin-rong Street, CHINA UNICOM China169 Backbone and PT Telekomunikasi Indonesia match the high number of packets with the quantities of their IP addresses.

Figure 2.4 shows the number of packets received by the telescope per destination port. We can see that port 23 (telnet) is the most active port with 29.37% of all packets followed by port 22 (SSH) and port 2323. Officially, port 2323 has been assigned by the IANA to the *3d-nfsd* service, but we know from Mirai that it also probes this port for telnet. Port 8545 is the default port of the JSON RPC protocol. Its popularity

Table 2.1: Top 10 of AS owners by involved IP addresses

Autonomous System Owner	Number of distinct IP addresses
No.31, Jin-rong Street	9,373
PT Telekomunikasi Indonesia	4,030
CHINA UNICOM China169 Backbone	2,652
VNPT Corp	1,509
China Unicom Beijing Province Network	903
National Internet Backbone	796
TELEFÔNICA BRASIL S.A	692
PT Excelcomindo Pratama (Network Access Provider)	635
PJSC Rostelecom	618
TOT Public Company Limited	597

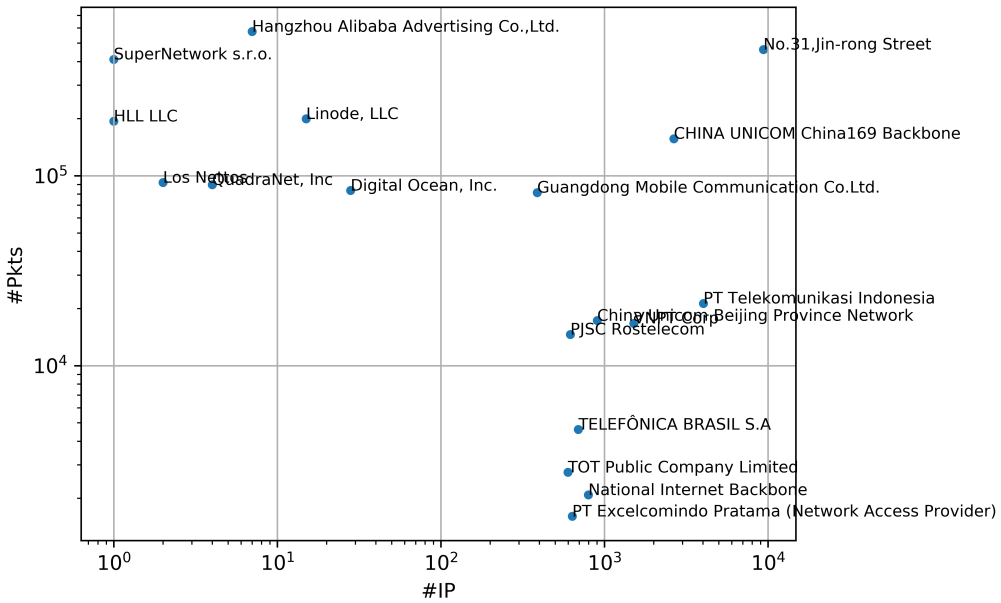

Figure 2.3: ICMP correlation on the telescope

Table 2.2: Top 10 of AS owners by incoming traffic

Autonomous System Owner	Number of incoming packets
Hangzhou Alibaba Advertising Co.,Ltd.	574,699
No.31,Jin-rong Street	462,393
SuperNetwork s.r.o.	410,297
Linode, LLC	199,525
HLL LLC	193,630
CHINA UNICOM China169 Backbone	156,772
Los Nettos	92,063
QuadraNet, Inc	89,633
Digital Ocean, Inc.	83,659
Guangdong Mobile Communication Co.Ltd.	81,609

could be explained by the fact that it is used by Ethereum clients. The ports associated with HTTP and HTTPS (port 80, 81, 8080, 8081, 8090, 443) and services such as SMB (port 445), FTP (port 21), and IMAP (port 993) also appear frequently.

Among IoT-specific protocols, UPnP's SSDP on port 1900 is the most popular one (0.4%). In contrast, only 0.017% of the packets target port 1883 (MQTT), 0.018% target port 8883 (MQTT over SSL), and 0.005% target port 5683 (CoAP). These protocols are not (yet) popular among attackers. Possible reasons for this could be that (a) only a few devices use these protocols, (b) effective attacks against the corresponding services are not known, or (c) most affected devices are hidden behind NAT and firewalls or use IPv6 instead of IPv4. Points (a) and (c) are partially confirmed by the Shodan search engine: a search for port 1900 returns 4,096,473 results, compared to only 48,190 for port 1883. The IoT constrained devices found online are very few in number. The shodan search was done on December 2018 and the majority of the malicious devices where standard nodes.

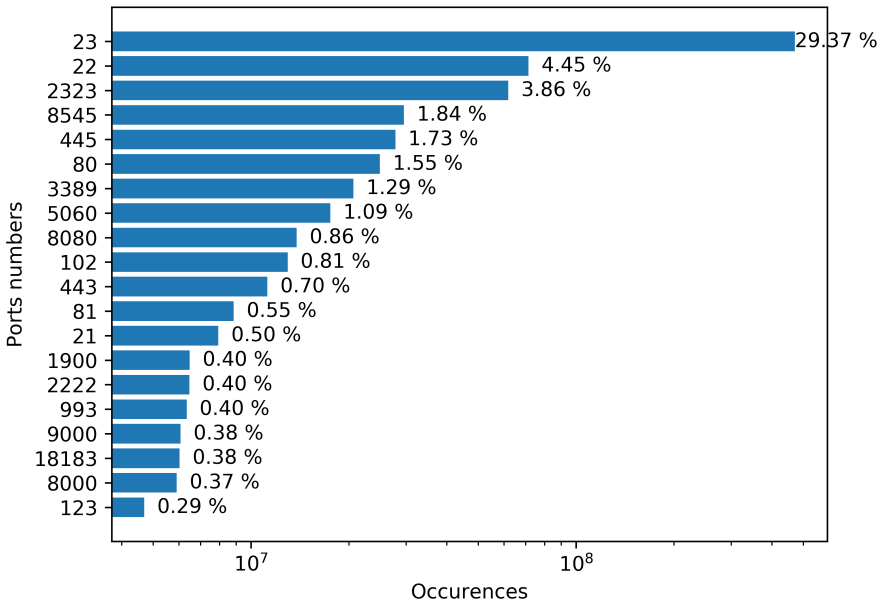

Figure 2.4: Incoming packets per port on the telescope (top 20). Note the scaling factor of 10^8 for the x-axis.

2.3.2 *Honeypot results*

General overview

Figure 2.5 shows the number of packets received by the honeypots per day. It should be noted that these (and the following) numbers are not directly comparable to those from the telescope since the honeypots are not passive and encourage connecting hosts to exchange more packets. The periods where the honeypots were not active are visualized as a zero baseline, for example in October 2017 and August 2018. Despite this gap in the data, we can see that the traffic intensity in 2018 is higher than the one in 2017. While receiving more than $3 \cdot 10^6$ per day was a rare event until March 2018, it has become the normal situation in the last nine months of the measurement period. The violent fluctuations appearing in the time series in the form of distinct peaks are caused by sharp and short raises in the number of long telnet connections. We will discuss them in Section 2.3.2. The honeypots were contacted by 765,207 distinct IP addresses during the experiment.

In total, around 74.5% of the received packets were targeting UDP and TCP services. The remaining 25.5% are discovery attempts with

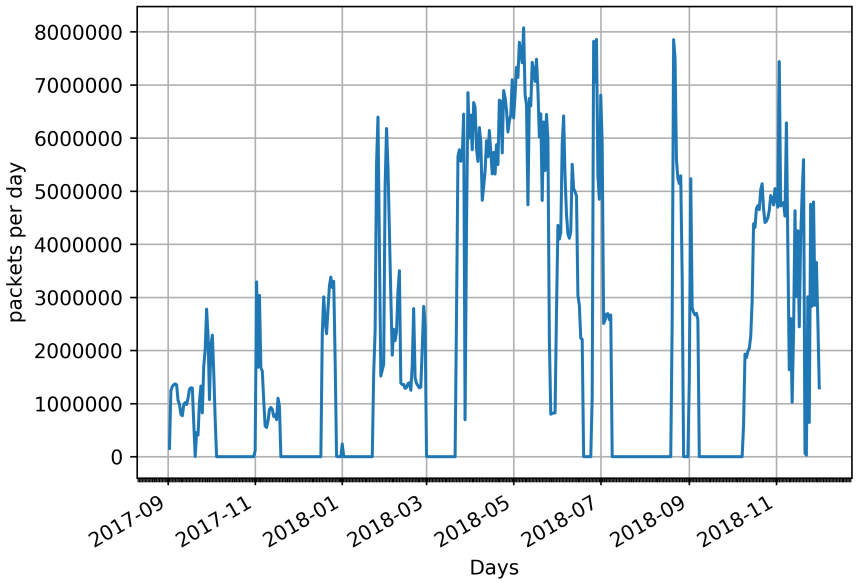

Figure 2.5: Incoming packets per day on the honeypots

ICMP echo request. We can see a huge disparity regarding the ICMP probes received on the honeypots compared to the telescope. Figure 2.6 shows the distribution of the ICMP packets on the honeypots. A total of 269,367,609 probes were sent from only 6,539 different source addresses to the 15 addresses of the honeypots. We find that only 31.38% of these addresses also sent TCP traffic and only 9.13% also sent UDP traffic. We find 87 sources sending more than a million probes, respectively. Table 2.3 shows the top 10 most active probers. We have an average of 962,027.175 probes per measurement day on the honeypots. The number-one prober is active 279 days out of 280 measurement days with a peak of 41,191 probes on 2017-12-22. This source IP did not send any other types of packets.

Table 2.4 shows the AS owners of the IP addresses contacting the honeypots. Those are not necessarily the most active IP addresses in terms of interactions with the honeypots because many of them only probe ports without further activities. Indeed, Table 2.5 shows a completely different set of top AS (except No.31,Jin-rong Street) if we sort them by the number of sent packets. Of course, it could be that attackers from those AS are behind NATs, in this way resulting in a smaller number of visible addresses.

Section 2.3.2 and **??**.

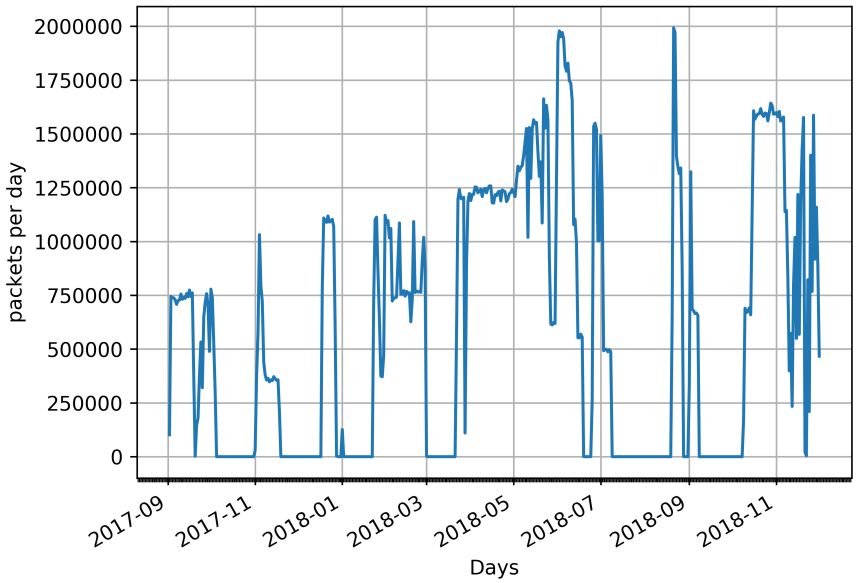

Figure 2.6: Incoming ICMP probes per day on the honeypots

Table 2.3: Top 10 sources sending ICMP probes. For privacy reasons, only the first 24 bits of the addresses are shown.

IP address	Number of probes
203.205.*.*	6,773,751
49.51.*.*	3,625,689
119.28.*.*	3,625,612
184.105.*.*	1,221,235
183.60.*.*	3,622,302
182.254.*.*	3,622,035
203.195.*.*	3,621,781
113.99.*.*	3,620,991
115.159.*.*	3,618,200
183.232.*.*	3,618,094

Figure 2.7 shows the number of received packets per port. Telnet is by far the most popular protocol, followed by SSH.

Table 2.4: Top 10 of AS owners by involved IP addresses

Autonomous System Owner	#distinct IP addresses
TELEFONICA BRASIL S.A	147,865
CHINA UNICOM China169 Backbone	70,181
No.31,Jin-rong Street	67,014
Magyar Telekom plc.	52,126
PJSC Rostelecom	45,987
Korea Telecom	40,536
Uninet S.A. de C.V.	35,066
OVH SAS	22,268
Turk Telekom	13,994
Data Communication Business Group	13,688

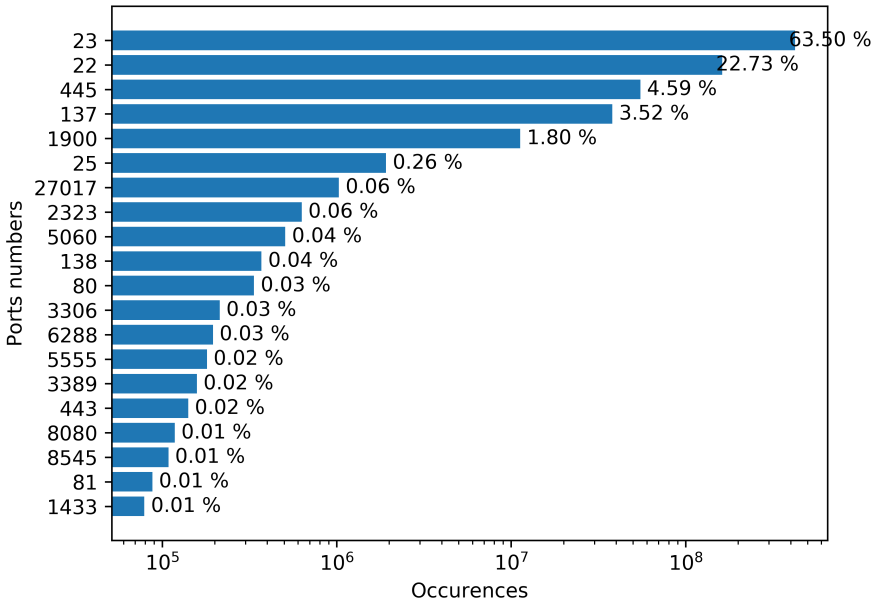

Figure 2.7: Incoming packets per port on the honeypots (top 20). Note the logarithmic x-axis.

Table 2.5: Top 10 of AS owners by incoming traffic

Autonomous System Owner	# incoming packets
Global Layer B.V.	107,121,191
FranTech Solutions	71,071,903
Digital Ocean, Inc.	69,130,300
Shenzhen Tencent Computer Systems Company Limited	52,137,841
Tencent Building, Kejizhongyi Avenue	43,822,755
Regionalnaya Kompaniya Svyazi Ltd.	38,828,910
Aruba S.p.A.	35,167,720
Hostio Solutions B.V.	23,124,288
SoftLayer Technologies Inc.	18,656,831
No.31, Jin-rong Street	18,093,637

SSH

Before presenting the results for the various IoT-related protocols, we briefly discuss the traffic received by Cowrie on port 22. In number of packets, SSH is the second most active protocol (Figure 2.7). One could expect that SSH sessions are used similarly as telnet. However, a manual inspection of the traffic reveals that it mostly consists of connection attempts without further communication. It can also consist of traffic reflected by servers originating from spoofed IP addresses as well. In both cases, we don't have enough information to decide whether they are related to IoT services. 50.68% (99,379,132 packets) of the SSH traffic come from the Autonomous System Owner Global Layer B.V. We also spotted 249 packets coming from well-known scanning projects like Shodan but they don't have a significant impact on our analysis due to their small number.

Telnet

All 85,788,366 login attempts on telnet follow the same pattern using a list of default credentials with the two most frequent ones being

`root/vizxv` (8.1% of all attempts) and root/12345 (3.7%). On average, attackers send 10 packets per telnet connection. This low number is mostly due to the presence of many short scans on ports 23 and 2323.

After a successful login, we see two different procedures: Some attackers write directly commands (or shellcode) on the terminal, the others first download a script file containing the commands. Attacks typically start with a series of `enable` commands to enable built-in functionalities of bash. Then, they create files with random content to check the existence of standard directories like /, /sys, /proc, /dev, /dev/pts, /run /bin, /dev/shm, /run/lock, /proc/sys/fs /proc/sys/fs/binfmt_misc/, /boot, /home. In this way, they try to identify honeypots. Finally, they download malware using tftp, curl or wget, that gives them root access to the system. They also download modified versions of services like ssh, telnet and ntpd to avoid future intrusions.

As mentioned in the general overview, the traffic on the honeypots is far from being evenly distributed over the measurement days. We saw two sharp rises on January 25 and February 1 where the average number of sent packets per telnet connection went up to 41 packets. On these days, the honeypots were mostly under attacks from the same IP address sending series of commands over the telnet connection before downloading binaries, whereas on the other days the script-based approach was dominant.

We see mostly Mirai and Mirai-like attacks. All these attacks show the typical sequence of commands targeting embedded systems with busybox[10] that gave Mirai its name [Dul+17]. However, Mirai's network signature, i.e. SYN packets with the ISN identical to the destination IP, is visible in only 10% of the attacks. We see variants that do not show this signature and that also include other changes (like a modified set of the IPs hardcoded in the original Mirai). Several of these attacks also try to install miner software (namely minerd, xmrminer, and cpuminer-multi) for crypto-currencies, we therefore conclude that attackers try to use IoT devices to harvest crypto-currencies. A deeper analysis of the different binaries shows miner software for bitcoin, ethereum and litecoin mostly.

HTTP

Only a few connections are targeting HTTP. Most of the 8,846 incoming HTTP requests are not specific enough to relate them to IoT. We see

10 https://busybox.net/

3,190 requests for the `/` resource and 2,417 requests with 61 different versions of attack URL against phpMyAdmin, the web-based administration tool for MySQL. We also spotted 122 requests targeting `bot.php`. Those are either attempts to exploit vulnerable chat bot scripts installed by normal users or checks to detect whether the host has been already successfully hacked by others and running a chat bot. We have filtered out those requests and manually analyzed the remaining ones for signs that the attackers were specifically targeting IoT systems.

0.76% of the HTTP requests try to access CGI resources of routers with known vulnerabilities, namely routers from Cisco (tmUnblock.cgi), Linksys (hndUnblock.cgi), and D-Link (hedwig.cgi). 0.85% of the attempts target getcfg.php to exploit vulnerabilities on D-Link DIR-6XX and DIR-8XX routers.

Much less frequent (0.12%) are attacks against Cisco's Home Network Administration Protocol (HNAP) for the management of home networks. The attackers request the resource `/HNAP1` in order to identify equipment supporting that protocol. 90% of those attacks came from only one IP address. The attacker used Firefox's user agent `Mozilla/5.0 (Windows NT 5.1; rv:9.0.1) Gecko/20100101 Firefox/9.0.1` in all attempts. Finally, 0.11% of the HTTP requests are targetting IP cameras with the Dahua backdoor (/current_config/passwd) and CVE-2017-8225 (/system.ini?loginuse&loginpas).

Other IoT constrained protocols

We will presented in the following the result about UPnP, CoAP and MQTT.

We see 726,978 attempts to get information on UPnP devices by using UPnP's service discovery protocol. As expected, most of those service discovery requests use the URI "*", but we also see 22 requests for http://www/. We assume that the latter does not use the standard implementation of the protocol anymore since they came all from the same IP address. It is reasonable to assume that these requests are caused by a misconfiguration and are not meant to be an attack.

The 702 received CoAP messages are requests to the standard resource `/.well-known/core` with which clients can obtain the list of available resources from a server. We did not received further requests to exploit the honeypots.

We received 1,277 MQTT messages of mainly the four types Connect, Subscribe, Disconnect, and Ping. We observe that 87.70% of the traffic

came from the same address and were concentrated on the period from 2018-01-26 to 2018-02-10. We note that the number of MQTT packets drastically decreased after February to only 81 messages exchanged in 10 days. However the number of source IP addresses is now 11.

Interestingly, we didn't observe any attempt to further exploit the above protocols beyond the first service-discovering interaction, although this is supported by the honeypots and many exploits of UPnP vulnerabilities are known. We assume this is the discovery phase of an automated mass scan to record IP and services running behind them.

2.3.3 *Shodan results*

As explained in Section 2.2.2, we used Shodan to obtain more information on the hosts behind the IP addresses that contacted the honeypots. The insight that can be gained by this or similar approaches is, of course, limited since hosts might be behind NATs.

We find different versions of Linux (3.x, 2.6.x, 2.4-2.6) but also many Windows versions, namely (Server 2012 R2 Standard 9600, 7 or 8, Server 2008 R2 Standard 7601 Service Pack 1, Server 2012 R2 Datacenter 9600, 6.1, XP, 10 Pro 17134 etc.), Unix and FreeBSD 9.x. We even find some Playstation 4 and HP-UX 11.x. However, Shodan is only able to provide this information for a small fraction of the observed IP addresses, therefore we will not discuss this further.

71.96% of the IP addresses connecting to the honeypot listen to the typical HTTP(S) ports 80, 8080 and 443, 8.51% have telnet (port 23), and 45.03% have ssh (port 23). After checking the type and version (TP-LINK WR740N WAP http config, Linksys wireless-G WAP http config, Netgear WNR3500L WAP http config, D-Link DCS-930L_46 webcam http interface), we conclude that many hosts are IoT devices (like IP cameras) or home routers, or are behind NATs running on such routers. Since those routers are mostly used by normal users, we can assume that the attacks probably come from infected devices in home networks.

SUMMARY

In this chapter, we deployed honeypots and used a telescope to observe attempts of attacks against IoT devices and infrastructures. Our theory was that since the IoT uses many different protocols with weak or

no authentication, many malicious acts are performed through them. However, our study shows that telnet-based attacks against IoT devices are still the most frequent ones, even raising during our measurements, probably caused by the enormous success of such attacks against surveillance cameras and home routers. However, while the original Mirai botnet was mostly used to perform DDoS, we see that new variants of Mirai-like malware also install miners for crypto-currencies. Furthermore, we see that those variants do not exhibit anymore Mirai's standard signature (SYN packets with the ISN identical to the destination IP address).

Attacks relying on IoT-specific protocols, such as MQTT, UPnP's SSDP and CoAP, are still much rarer. In our data, connection attempts using these protocols were limited to service discovery interactions without further attempts to find or exploit vulnerabilities. Nevertheless, we expect to see more attacks in the future using these or other IoT protocols. The Mirai attacks have drawn a lot of attention to telnet-related vulnerabilities. Once those are fixed, attackers will turn to other protocols to infect IoT devices.

3

HOW TO FIND RESOURCE-CONSTRAINED DEVICES?

Woodstock was both a peaceful protest and a global celebration.

—Richie Havens

INTRODUCTION

In computer security, networks scans are a valuable tool equally used by network managers as well as malicious parties. With the increasing popularity of IoT, we expect that network administrators will use scan algorithms like the one presented in this chapter to keep track of all the devices deployed in their network. Depending on its nature, a network scan allows to discover the IP addresses used by the devices. Ideally, a scan also involves some kind of finger-printing to identify the type of hardware, operating system and applications behind those addresses. For the reasons explained above, the initiator of the scan might be particularly interested in finding constrained devices, assuming that they are easier targets for attacks with their weaker security mechanisms.

In general, one wants to perform the scan as efficient as possible in terms of duration and number of exchanged packets, since scans with long duration or high bandwidth can have a significant impact on the network. This is also especially important when IPv6 is used because of its large address space. However, performing a scan with a high discovery rate can be a challenging task in an IoT environment, since slow networks such as IEEE 802.15.4 are easily overloaded.

To help network administrators to search and distinguish IoT devices in their networks, we propose a new scan algorithm tailored to resource-constrained networks in this chapter. For this, we assume that the IoT is a heterogeneous environment hosting a mix of devices with different capabilities and using different communication technologies. In particular, we consider an infrastructure of constrained devices communicating over low-power wireless IEEE 802.15.4 links and more powerful devices using faster IEEE 802.11 (WiFi) links.

Our approach is based on Round-Trip Time (RTT) measurements with a variation of probe size (i.e. the size of the probing packet) and probing speed (i.e. the number of probes sent per time unit). The measurements allow the scanner to differentiate nodes by the used network technology. We show that the knowledge gained from this differentiation can be used to control the scan strategy to reduce probe losses and, hence, increase the efficiency of the scan.

To validate the feasibility of our approach, we have built an ns3 model of a mixed IoT infrastructure connected to the Internet and simulated network scans using our measurement scheme. Our experiments show that our approach indeed allows more efficient scans of heterogeneous IoT networks. As a secondary goal, our chapter illustrates that designers

of intrusion detection systems will have to face new attack techniques adapted to IoT environments.

The remainder of this chapter is organized as follows: we present related work in Section 3.1.In Section 3.2, we introduce our approach. We present the experimental methodology in Section 3.3 and discuss results in Section 3.4.

3.1 RELATED WORKS

In this section, we discuss publications related to this chapter. Apart from the obvious connection to existing works on network scans, our approach makes use of techniques that are related to bandwidth estimation and fingerprinting.

3.1.1 *Network scans*

Network scan implementations can be differentiated by the targeted network layer, the used protocol, the scanning speed, the number of scanners, and the scan targets [BIo8]. For large Internet-wide scans, trending tools are ZMap [DWH13] and MASSCAN [DBH14], which can probe the entire IPv4 address range under 45 minutes with 97% of the theoretical maximum speed of gigabit Ethernet. In [Dai+12], the authors point out new scanning procedures using botnets. By employing a large number of scanning hosts, the detection by simple intrusion detection systems can be avoided since the individual scanners can operate at slow speed and still achieve together a high probing speed. Recently, the usage of compromised IoT nodes by scanners has become popular, too [Pa+15].

3.1.2 *Available Bandwidth Estimation*

Available bandwidth estimation (ABE) injects probes into the network for measurement and is commonly used in wired or wireless networks to determine the throughput of a particular network link or of an entire end-to-end path. Different approaches exist based on the network type. Since the goal of ABE is to obtain an accurate estimation of the bandwidth, most approaches send a series of equal-sized probing packets and increase the emission rate of the packets gradually [Sar+08] or use special flight patterns [Rib+03].

In wireless networks, bandwidth measurements depend also on the probe packet size and cross-traffic rate [JMB06]. The added probing traffic and the end-to-end technique sometimes influence the measurement and that is more noticeable on multi-hop networks like 802.11 or 802.15.4. In [FK13], Farooq and Kunz proved that the MAC layer overhead leads also to congestion and influences the bandwidth estimation.

In this chapter, our goal is to obtain a fast and light-weight distinction between nodes using IEEE 802.11 and 802.15.4. In contrast to ABE approaches, we achieve this by performing an initial RTT measurement and then adapting the probing properties to be able to quickly identify the network technology.

3.1.3 *OS Fingerprinting*

In the introduction, we have argued that motivation for scanning a network can be to identify constrained devices. In this chapter, we propose an indirect approach to achieve this based on the identification of the used network technology. A more direct approach is OS Fingerprinting. OS Fingerprinting has as goal to identify the OS on a particular node. In passive approaches, one monitors the network and analyses the packets to find clues on the OS. This is possible because of the differences in TCP/IP stack implementations [Bev04]. Other passive approaches are using machine learning to learn useful features for the distinction [AG19]. In an active approach, the node is probed with malformed packets coupled with carefully chosen payloads. All OS have specific ways to handle such packets, which allows to identify them easily provided that a response is received [BSE18].

Various ways exist to defeat fingerprinting methods [Tal03]. In this chapter, our starting point is, therefore, that the scanner has no reliable way to identify the OS with sufficient precision.

3.2 PROPOSED APPROACH

In this section, we explain our approach to increase the efficiency of network scans. We describe the considered scenario in Section 3.2.1. The scan procedure is described in Section 3.2.2 and 3.2.3.

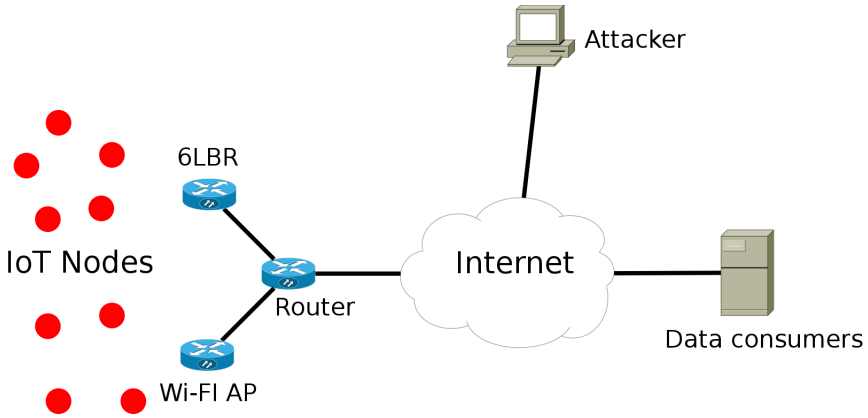

Figure 3.1: Scenario considered in this chapter

3.2.1 *Considered Scenario*

We consider the scenario depicted in Figure 3.1: An IoT network with a large number of nodes is connected to the Internet, allowing the exchange of information between IoT nodes and users/servers that consume the data produced by them. To connect those nodes to the Internet, various protocols are used. As explained, we focus on IEEE 802.11 and IEEE 802.15.4 (with 6LoWPAN) in the following. In practice, the operator of such a heterogeneous IoT will divide the network into smaller sub-networks based on the used network technology. In the example shown in the Figure 3.1, we see two such sub-networks connected via a WiFi access point and a 6LoWPAN border router (6LBR) to a router.

IoT nodes connected to the Internet are exposed to many attacks. We focus on network scans, a *reconnaissance* attack type that is very frequent in the Internet, but also used by network administrators to discover mis-configured or forgotten hosts. Once the scanner has guessed the address of the IoT network of interest, for example through information obtained from DNS servers or from passive monitoring of the data exchange with the data consumers in the Internet, the attacker will scan the corresponding entire address range. We assume that the scanner probes for services that can be typically expected in an IoT, such as CoAP.

Scans on the Internet are usually performed very fast. Tools like ZMap can scan the entire public IPv4 address space in less than 5 minutes at 10 Gbit/s [Adr+14]. When scanning an IPv6 network, the

speed becomes even more important: Even the smallest network in IPv6 has more addresses than the entire IPv4. While the high scan rate of ZMap is suitable for the Internet, it would be too aggressive for a network containing a mix of IoT nodes connected via IEEE 802.11 or IEEE 802.15.4. A scan rate adapted to the speed of modern Ethernet or even IEEE 802.11 is far too high for IEEE 802.15.4. The result would be a low *hit rate* or *discovery rate* (i.e. the fraction of discovered nodes) due to many lost probes.

Note that the huge address space of IPv6 might give an illusion of safety against scans. However, by using techniques such as in [GC16] i.e. guessing the Network ID, and IID, the potential target address range can be reduced. Since typical IoT node manufacturers and their Organizational Unique Identifier (OUI) are known, an IoT network can be probed in a reasonable amount of time.

3.2.2 *General description*

Our proposed approach is based on round-trip time (RTT) measurements. Our method consists of overloading the destination router to identify the devices and technologies that are used in their networks. However, from an ethical perspective, it is unacceptable for a measurement campaign to prevent the proper functioning of the various target devices. For this reason, it is recommended that, before using our approach, administrators of the target networks should give their favourable opinions, set the maximum duration of the scan and the period during which it can be carried out so as not to hinder the proper functioning of the target devices. Any breach of this prior agreement would be a serious violation and could be subject to prosecution. On the other hand, our method is adaptive and therefore more gentle than a brute-force high-speed scan using existing tools. It could be therefore an alternative for researchers and administrators.

Our method is describe in the following three steps.

Step 1 – Fast probing

The first step is to probe the entire address range of interest. To save time, we start with a small probe size and a high scan rate adapted to the fastest network technology we expect behind those addresses, i.e. a variant of IEEE 802.11 in our case. Naturally, for the scenario depicted in Figure 3.1, such a fast scan will only work reliably for the

IEEE 802.11 sub-network, while resulting in many lost probes in the slower IEEE 802.15.4 sub-network. Nevertheless, this first scan gives us a first insight into what parts of the address range are in use.

Step 2 – Probing with size variation

This step's goal is to identify what network technologies are used by the nodes discovered in step 1. To this end, we perform a slow scan against a randomly chosen small subset of those nodes.

As already explained, IEEE 802.15.4 with a speed of 250 kbit/s is much slower than IEEE 802.11 (11 Mbit/s for IEEE 802.11b). This results in a noticeable difference between the RTTs of probes. To amplify this difference, we modify the probe size during the second step. The key is the fragmentation performed by the 6LBR: IEEE 802.15.4 only supports 127 bytes as maximum physical layer service data unit (PSDU) while IEEE 802.11 supports the minimum MTU of 1280 bytes required by IPv6. This means that probes larger than 127 bytes are fragmented by the 6LBR and result in a clear increase of the RTT for IEEE 802.15.4 nodes while the RTT for IEEE 802.11 nodes will stay relatively stable even for larger packets. If nodes in the IEEE 802.15.4 are more than one hop away from the border router, the RTT is even further increased.

Step 3 – Slow re-probing

In this last step, we re-scan at a lower speed and with small probes only those parts of the address range where we have identified IEEE 802.15.4 nodes in step 2. The assumption is that our fast scan in step 1 has missed many nodes belonging to the IEEE 802.15 sub-network. By re-scanning with a lower speed we avoid probe losses and achieve a higher hit rate.

By starting with a fast scan (step 1) and slowly re-scanning (step 3) only those parts of the address range that were identified as (likely) containing IEEE 802.15.4 nodes (step 2), our approach achieves a good compromise between scan duration and discovery rate. A non-adaptive scan strategy would have to scan the entire address range slowly to achieve a comparable rate.

Algorithm 3.2.1: Basic scanning

Input: *listAddr* : addresses to scan, *pSize* : probe size, *pSpeed* : probe speed, *ratio*

```
1  as ← queue() // queue with all the address we will scan
2  if ratio == 1 then
3  |  as ← all addresses from the sub-network listAddr
4  end
5  else
6  |  as ← addresses randomly picked from listAddr with ratio
   |    ratio (but minimum 10)
7  end
8  send probes of size pSize with probe speed pSpeed to all
     addresses in as sequentially;
9  wait at most 5 seconds for replies;
10 received ← all addresses that replied;
11 rtt ← average measured RTT over all addresses in received;
12 return (received, rtt)
```

3.2.3 *Algorithm*

Algorithm 3.2.1 shows the basic building block of the implementation, a (non-adaptive) scan procedure. It sends probes of size *pSize* with probing speed *pSpeed* to a list of addresses. This list consists either of all addresses of a sub-network (indicated by $ratio = 1$; line 3) or of individual addresses sampled from the provided address list with ratio *ratio*. We try to take at least 10 addresses (line 6).

The main procedure is shown in Algorithm 3.2.2. It obtains a list of sub-networks *ns* to scan. To simplify the explanations in the following, the depicted algorithm processes sub-networks one by one (line 1), although a concrete implementation does not have to follow this order. It first performs a fast scan with small probes for all addresses in a sub-network (line 2), in this way obtaining a list of addresses $received_1$ that responded and an average RTT rtt_1. Some (with *ssize* ratio) of the addresses that responded are scanned again slowly with large probes (line 3). Since protocols like UDP are unreliable, the second scan doesn't always receive a satisfying number of replies for the following steps. Therefore, we repeat the slow scan up to three times (lines 4–7). With empirical experiment results, we find that *mvalue* should be 10 IPv6

Algorithm 3.2.2: Main procedure

Input: ns : sub-networks to scan, $pSize$: probe size, $pSpeed$: probe speed, $ssize$: ratio, $mvalue$: minimum number of responses, $tlimit$: rtt threshold, $factor$: Factor we except between rtt1 and rtt2

1 **for** each sub-network n in ns **do**
2 $(received_1, rtt_1) \leftarrow scan(n, psize_{small}, pspeed_{fast}, 1)$;
3 $(received_2, rtt_2) \leftarrow scan(received_1, psize_{large}, pspeed_{slow}, ssize)$;
4 **while** $size(received_2) < mvalue$ and number of attempts < 3 **do**
5 $(received', rtt') \leftarrow scan(received_1, psize_{large}, pspeed_{slow}, ssize)$;
6 add $(received', rtt')$ to $(received_2, rtt_2)$
7 **end**
8 $activeAddrList \leftarrow received_1 \cap received_2$;
9 **while** $rtt_2 < rtt_1$ and number of attempts < 3 **do**
10 $(received_1, rtt_1) \leftarrow scan(activeAddrList, psize_{small}, pspeed_{fast}, ssize)$;
11 $(received_2, rtt_2) \leftarrow scan(activeAddrList, psize_{large}, pspeed_{slow}, ssize)$;
12 **end**
13 **if** $(rtt_2 >= rtt_1)$ **then**
14 **if** $(rtt_2 >= tlimit$ and $rtt_2 >= factor.rtt_1)$ **then**
15 $type[n] \leftarrow$ 802.15.4
16 **end**
17 **else**
18 $type[n] \leftarrow$ 802.11;
19 $received[n] \leftarrow received_1$
20 **end**
21 **end**
22 **else**
23 `// No classification possible for this sub-network`
24 abort
25 **end**
26 **end**
27 `// rescan sub-networks identified as 802.15.4`
28 **for** each sub-network n with $type[n] ==$ 802.15.4 **do**
29 $(received[n], rtt) \leftarrow scan(n, psize_{small}, pspeed_{slow}, 1)$
30 **end**
31 **return** $(received, type)$

addresses RTT for each type of networks to make the distinction of the network type.

A comparison is done with the mean from the first and the second phase to trigger error handling (line 9). If an error occurred (unexpected delays in the first scanning), the two first phases are redone but only on *ssize* (typically 10% of the nodes) of the alive hosts previously found. The final classification of the sub-networks is based on two criteria: (i) the mean of the second phase is greater than *tlimit* (typically 0.5 second is the optimal value found after 50 tests), and (ii) we have at least a certain *factor* (typically 4 is the optimal value found after 50 tests) between the mean of the first and the second phase. Finally, the network type is saved and used in the last phase to rescan and find more nodes in the 802.15.4 networks.

In our experience, the probability of false positives (i.e. sub-networks wrongly classified as 802.15.4) is quite low due to the dual condition to determine the type of networks (line 14 of algorithm 3.2.2). The values for the threshold and the factor work for a very wide range of scenarios and have been verified by us in simulation and with real devices. We have noticed in our experiments (see Section 3.4) that the probe size doesn't affect much the 802.11 networks in contrast to the 802.15.4 ones where factors of 4 or greater between the RTT of the first step and second step are measured.

On the other hand, there is a significant chance for false negatives related to the amount of nodes available for the second phase and the number of replies collected. We have observed that when less than 5 nodes are used in the second phase, we may get a very small RTT, especially if we don't get all the replies in one go due to losses. To handle such situations, we redo the scans when the number of obtained nodes is too low or when the measured RTT behaves not in the expected way (lines 4 and 9). In the worst case, a final test to detect strange behaviour between the first and second step (line 13) will abort the classification.

3.3 EXPERIMENTAL METHODOLOGY

We use ns3 (www.nsnam.org), which is one of the few simulators supporting heterogeneous networks with IEEE 802.11 and 6LoWPAN over IEEE 802.15.4. Ns3 is written in C++, very powerful and complete [BO+13]. The current ns3 version 3.26 still lacks support for RPL. We

have used an experimental version[1] with RPL support for our experiments with multi-hop networks.

Some parameters of the algorithm 3.2.2 have been set based on multiple experiments to reassure the best compromise between time consumption and the efficiency of our method. We choose $ssize = 10\%$ representing the minimum amount of nodes needed for the second phase. $tlimit = 0.5s$ is the time limit and $factor = 4$ the ratio used to differ between 802.15.4 and 802.11 nodes.

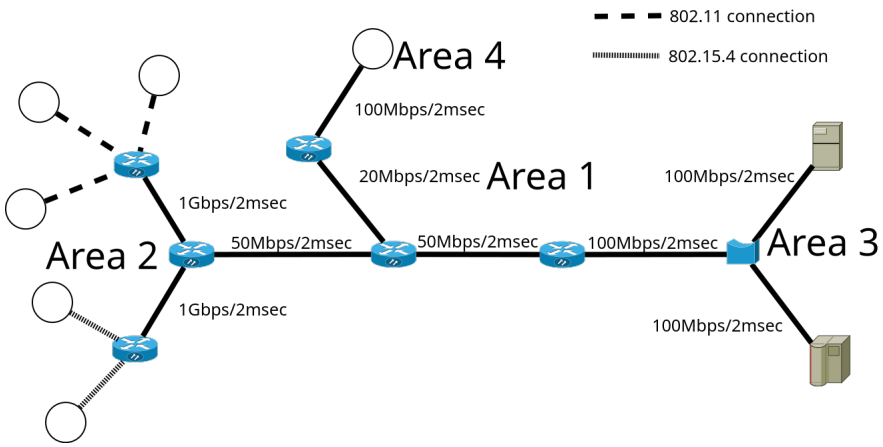

Figure 3.2: Simulated network topology

Figure 3.2 shows the simulated network. It consists of:

- *Area 1* contains the Internet and the border routers of the different sub-networks.
- *Area 2* represents the IoT network with its sub-networks for 802.11g and 802.15.4 nodes. Each node hosts a CoAP client on UDP port 5683.
- *Area 3* contains two servers that periodically collect data from the IoT nodes.
- *Area 4* contains the host that scans the IoT nodes by sending CoAP requests.

We call the legitimate traffic exchange between area 2 and area 3 *background* traffic. We perform our experiments with *light* and *normal* background traffic.

1 This version was provided to us by the Assistant professor Tommaso Pecorella https://unifi.academia.edu/TommasoPecorella, maintainer of NS3

We let the simulation perform a warm-up time of several minutes to initialize the routing protocols and to populate all the routing tables. We divide our validation into two parts:

3.3.1 *Tests with one network technology*

The goal of the first part is to verify our assumptions on the behaviour of IEEE 802.11 and IEEE 802.15.4 when facing a simple scan that does *not* use our approach. To this end, we study the two network technologies separately. For these tests, area 2 consists of three sub-networks with 100, 50 and 20 nodes, respectively. The links between the access points (IEEE 802.11) or 6LBR (IEEE 802.15.4) of the sub-networks and the main network have a bandwidth/delay of 1 Gbps/2ms, 100 Mbps/2ms and 50 Mbps/2ms, respectively. We perform scans by sending CoAP probes with different rates and sizes and estimate the RTT by measuring the time until a response (if any) is received. The following experiments are performed:

IEEE 802.11 NETWORK

The IEEE 802.11 devices form a circle of radius 80 meters with the access point at its center. The probing starts around 1000 seconds (time to adjust the Request To Send (RTS)/Clear To Send (CTS) mechanism with the background traffic) after the beginning of the experiment with different inter-probing times. The IEEE 802.11 nodes send data packets of 512 bytes every 20 seconds for the light background traffic and every 10 seconds for the normal traffic to one of the servers which reply automatically with a 512 byte packet. In addition to this push-style communication, all nodes also have the CoAP port open.

IEEE 802.15.4 NETWORK WITHOUT RPL ROUTING

The IEEE 802.15.4 nodes also form a circle with radius 80 meters with the 6LBR as the center. The probing starts 5000 seconds after the beginning of the experiment to match the RPL experiment which needs this time for RPL to converge. For the background traffic, one of the servers sends a 30 bytes CoAP request to the IEEE 802.15.4 nodes consecutively every 20 seconds for the light background traffic and every 10 seconds for the normal traffic. The nodes reply with a 512-byte response. Note that all nodes are in the communication range of the 6LBR and no routing protocol is used.

IEEE 802.15.4 NETWORK WITH RPL ROUTING

For this test, we modify the above IEEE 802.15.4 setup: We place the nodes on two different circles of radius 80 cm and 160 cm around the 6LBR and use RPL for multi-hop routing. To compare the results, we also test the original setup (one circle) with RPL. The background traffic is generated as described in the test without RPL. RPL was configured with the *ETX metric* and *objective function 0* [Win12].

3.3.2 *Tests with mixed network technologies*

In this second set of tests, we validate our proposed approach for efficient scanning. We simulate a heterogeneous network in area 2, consisting of two sub-networks: The first contains 100 IEEE 802.15.4 nodes and the second contains 50 IEEE 802.11 nodes. The up-links of the 6LBR router and the access point have a bandwidth/delay of 1 Gbps/2ms. We first perform a fast scan (step 1), then probe random nodes out of the found nodes to identify IEEE 802.15.4 nodes (step 2), and finally perform a slower scan on that part of the address range that contained those nodes (step 3).

3.4 RESULTS

Following the methodology described in the previous section, we first study the two network technologies separately in Section 3.4.1 (IEEE 802.11) and Section 3.4.2 (IEEE 802.15.4), before testing our scan approach on a heterogeneous network in Section 3.4.3.

3.4.1 *Behaviour of IEEE 802.11 nodes*

In our first experiment, we scan area 2 with a constant and moderate inter-probing time of $T = 80$ ms and three different probe sizes. The measured RTTs of the scanned nodes are shown in Figure 3.3 and Figure 3.4 for respectively light and normal background traffic. The three sub-networks with 100, 50 and 20 nodes are visible as three distinct groups (from left to right) in the result. Since they are in a different range of IP addresses, we have gaps without any measurement results.

We observe that the amount of background traffic does not influence the RTTs since it is relatively small compared to the available network

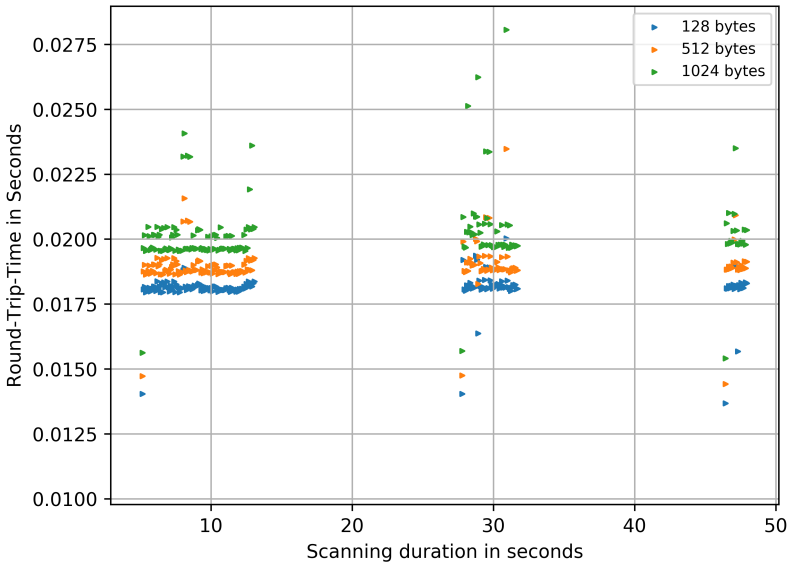

Figure 3.3: RTT for IEEE 802.11; light background, $T = 80$ ms.

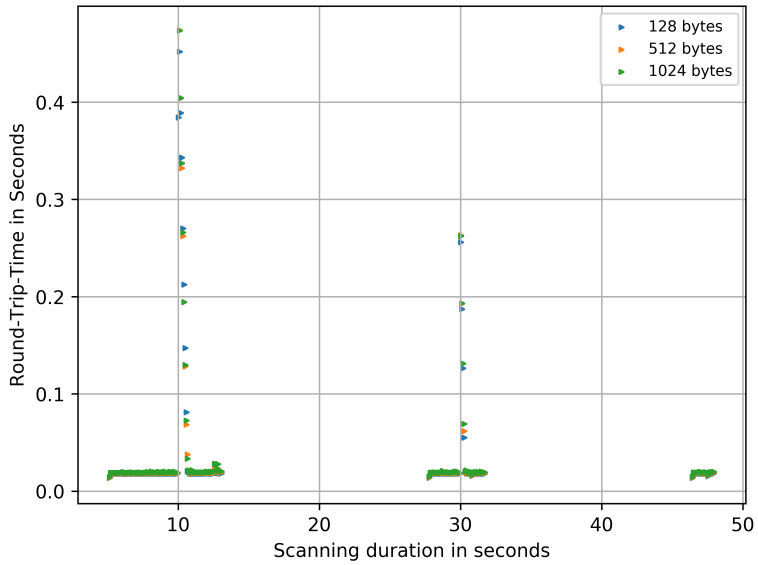

Figure 3.4: RTT for IEEE 802.11; normal background; $T = 80$ ms.

bandwidth. Furthermore, we observe that the probe size only has a small effect. Also note that Figure 3.4 shows a few outliers. We believe that these are simulation glitches or caused by environment background

processes. We have decided to keep them for the sake of completeness and repeatability.

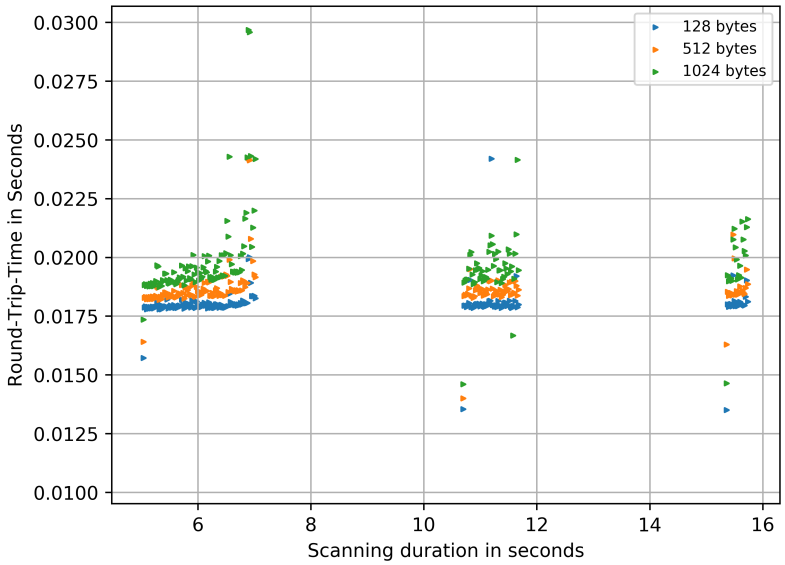

Figure 3.5: RTT for IEEE 802.11; normal background, $T = 20$ ms.

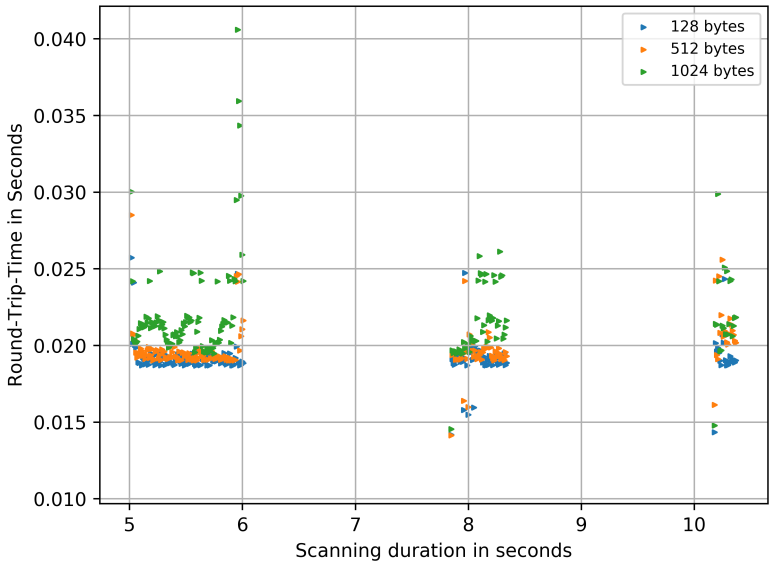

Figure 3.6: RTT for IEEE 802.11; normal background; $T = 10$ ms.

Figure 3.5 and Figure 3.6 show the obtained results when scanning with a shorter inter-probing time $T = 20$ ms and 10 ms, respectively, and normal background traffic. The results show a slightly increased RTT compared to the test with $T = 80$ ms and stable results for probes of 128 bytes and 512 bytes. We can also see that the 1024-byte probes suffer from a small queuing effect in the largest (left-most) sub-network when sent with $T = 10$ ms.

Independently from the probe size and probing speed, the scans in the above experiments were always able to find 100% of the nodes, i.e. no probes were lost.

3.4.2 *Behaviour of IEEE 802.15.4 nodes*

In our second set of experiments, we simulate area 2 with three IEEE 802.15.4 sub-networks. We present the results when the nodes are directly connected to the 6LBR without RPL, followed by the results with RPL in this section.

IEEE 802.15.4 nodes without RPL

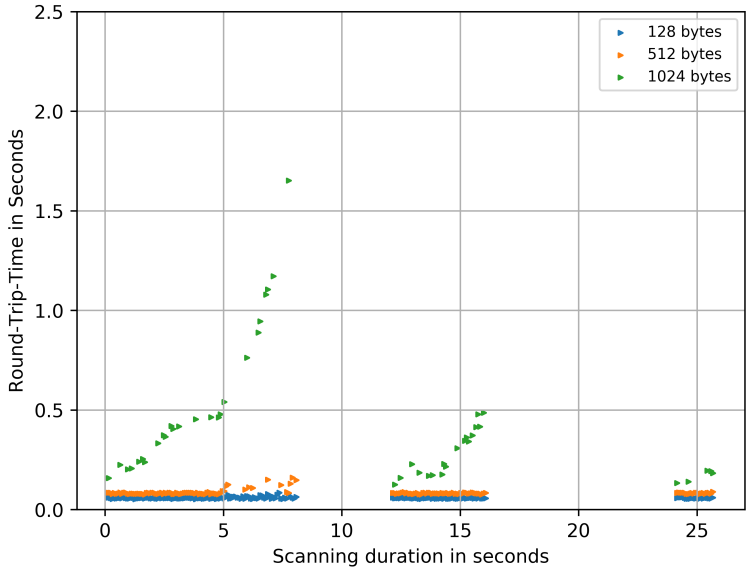

Figure 3.7: RTT for IEEE 802.15.4 without RPL; light background, $T = 80$ ms.

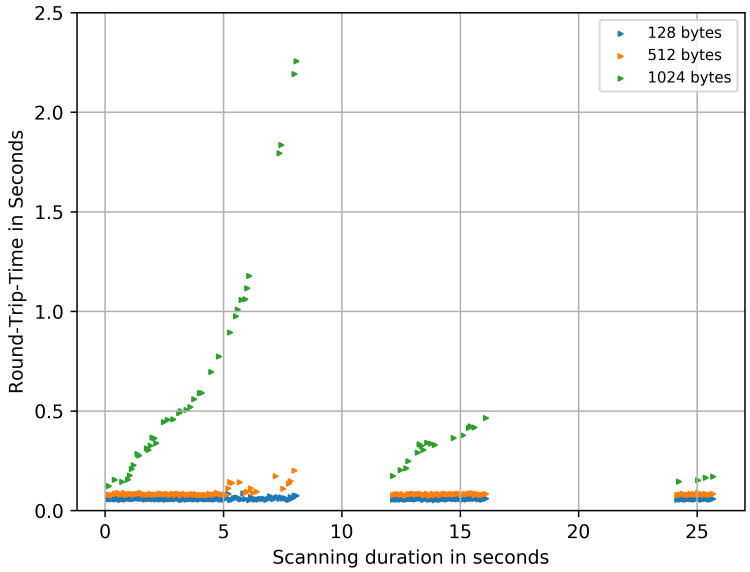

Figure 3.8: RTT for IEEE 802.15.4 without RPL; normal background, $T = 80$ ms.

Again, we begin with a constant and moderate inter-probing time of $T = 80$ ms and three different probe sizes. The results for light and normal background traffic are show in Figure 3.7 and Figure 3.8. The corresponding box-plot in Figure 3.9 shows that the probe size has a great impact on the measured RTT.

Even at this slow scan speed, the results for 128-byte probes and 1024-byte probes differ by one order of magnitude and the queuing effect is much more pronounced than in the experiments with IEEE 802.11. This is due to the low bandwidth and small MTU of IEEE 802.15.4, as explained in Section 3.2.2.

Figure 3.10 and Figure 3.11 show the results for $T = 20$ ms and $T = 10$ ms, respectively. We observe a very high RTT with large variations for all probe sizes.

In addition, a second effect becomes visible: a large number of probes are lost because of the high scan speed. Figure 3.12 shows the percentage of nodes discovered by the scan with normal background traffic. For small probe sizes ($\leq$ 200 bytes) and slow scan speed ($T \geq 80$ ms), more than 95% of the nodes are found. This number drops quickly for faster scans and larger probes. For example, less than 30% of the nodes are found with 512-byte probes and $T = 40$ ms.

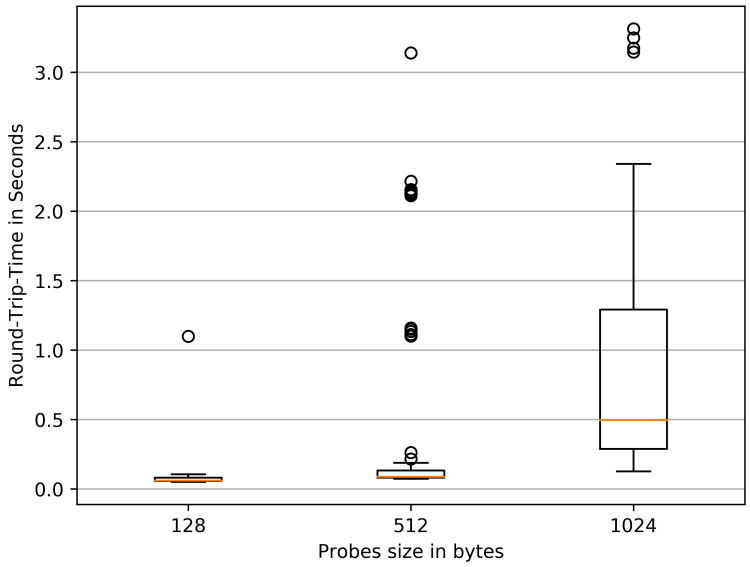

Figure 3.9: Probe size impact on IEEE 802.15.4 without RPL; normal background, $T = 80$ ms.

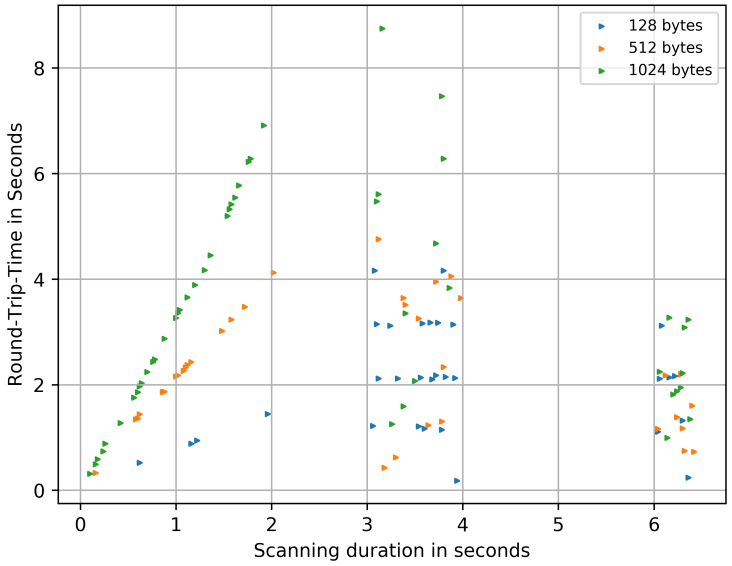

Figure 3.10: RTT for IEEE 802.15.4 without RPL; normal background, $T = 20$ ms.

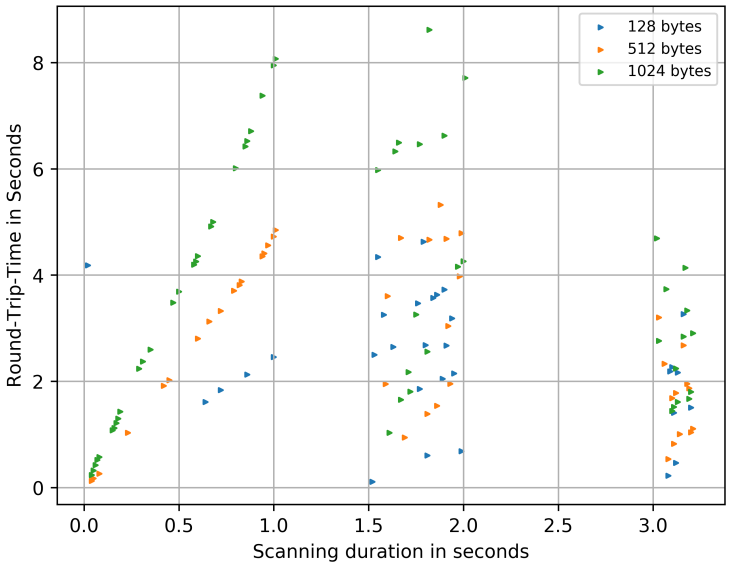

Figure 3.11: RTT for IEEE 802.15.4 without RPL; normal background, $T = 10$ ms.

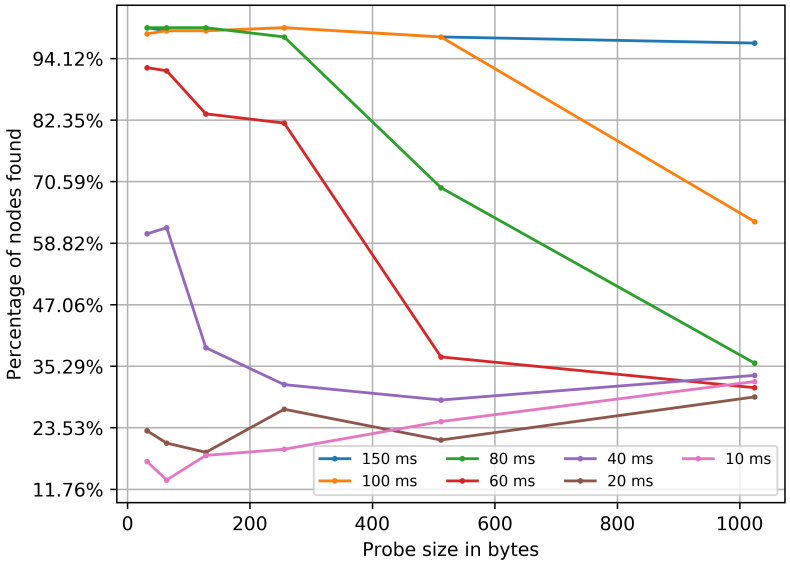

Figure 3.12: Number of IEEE 802.15.4 nodes found (without RPL); normal background.

IEEE 802.15.4 nodes with RPL

We begin with the same single-hop configuration as in the previous test, but this time we activate RPL. The results for light and normal

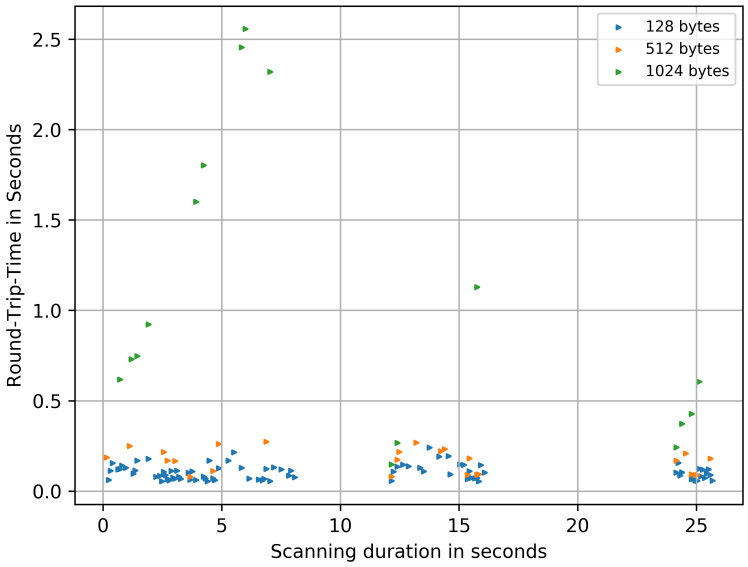

Figure 3.13: RTT for IEEE 802.15.4 with RPL and 1 hop; light background, $T = 80$ ms.

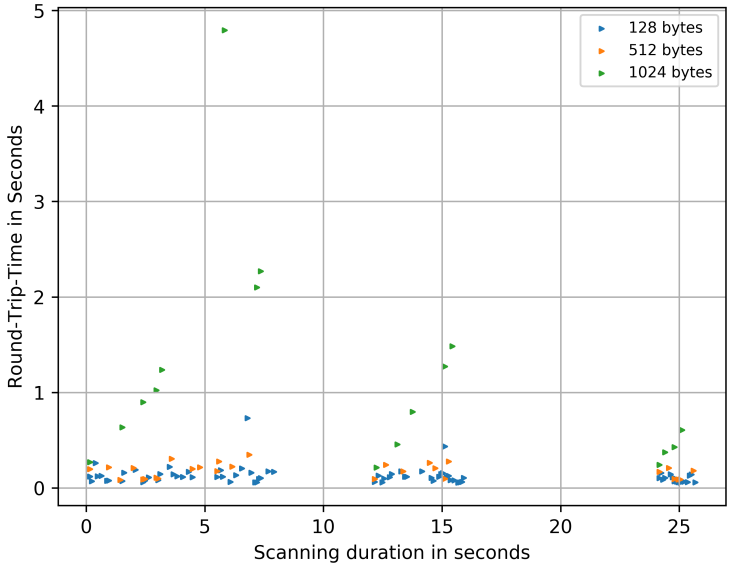

Figure 3.14: RTT for IEEE 802.15.4 with RPL and 1 hop; normal background, $T = 80$ ms.

background traffic are shown in Figure 3.13 and Figure 3.14. The dis-

parity between 128-byte and 1024-byte probes becomes larger, which is also clearly visible in the box-plot in Figure 3.15.

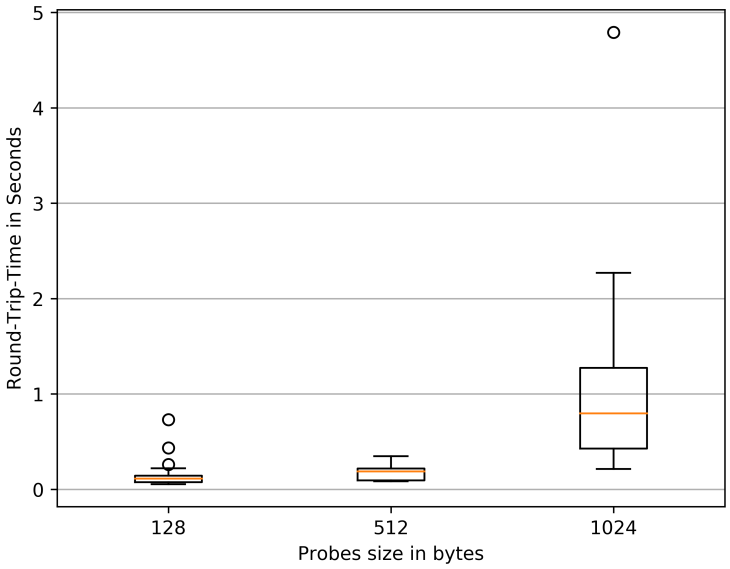

Figure 3.15: Probes size impact on IEEE 802.15.4 with RPL and 1 hop; normal background, $T = 80$ ms.

Figure 3.16 and Figure 3.17 show the impact of faster probing. Many losses are present with fast probing, leading to less than 10% nodes found as shown in Figure 3.18. We need $T = 150$ ms to be able to find more than 90% of the nodes.

Furthermore, Figure 3.12 and Figure 3.18 show the loss rate is higher than in the tests without RPL although both scenarios simulate a one-hop network. The reason for this are collisions with RPL control messages.

Figure 3.19 and Figure 3.20 show the results for light and normal background traffic and $T = 80$ ms in a two-hop IEEE 802.15.4 network with RPL. The box-plot in Figure 3.21 shows that the variation of the measured RTT has increased due to the additional hop. However, note that many large probes are lost in this scenario and therefore the results are not directly comparable.

Figure 3.22 shows that we find only around 55% of the nodes when probing with 128-byte messages and $T = 80$ ms, compared to a discovery rate of nearly 100% when not using RPL. We need a very low scan speed ($T \geq 100$ ms) and small probes (≤ 100 bytes) to discover 90% of the nodes in the two-hop scenario.

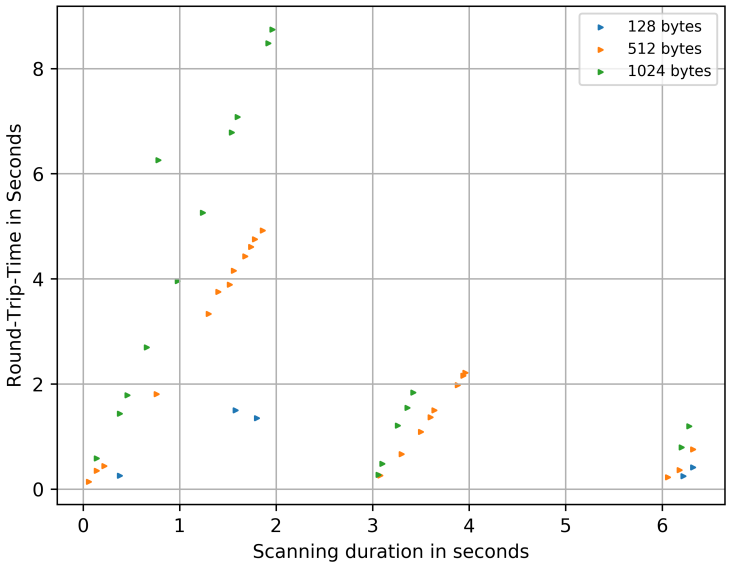

Figure 3.16: RTT for IEEE 802.15.4 with RPL and 1 hop; normal background, $T = 20$ ms.

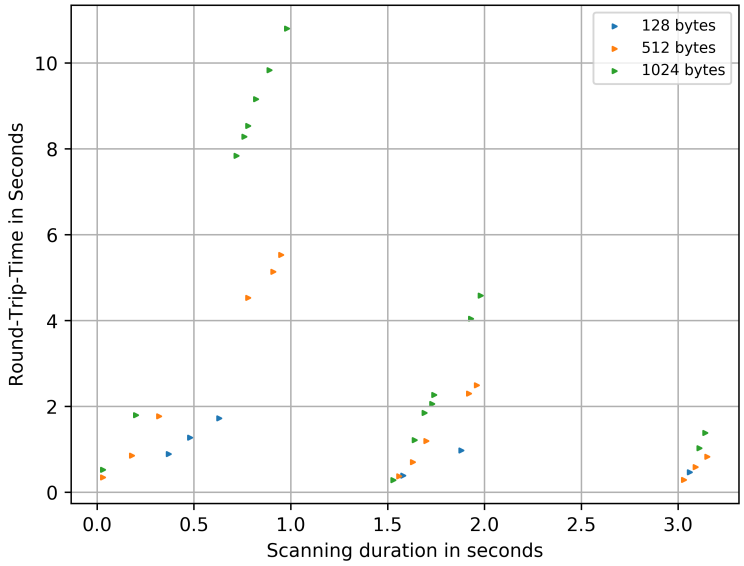

Figure 3.17: RTT for IEEE 802.15.4 with RPL and 1 hop; normal background, $T = 10$ ms.

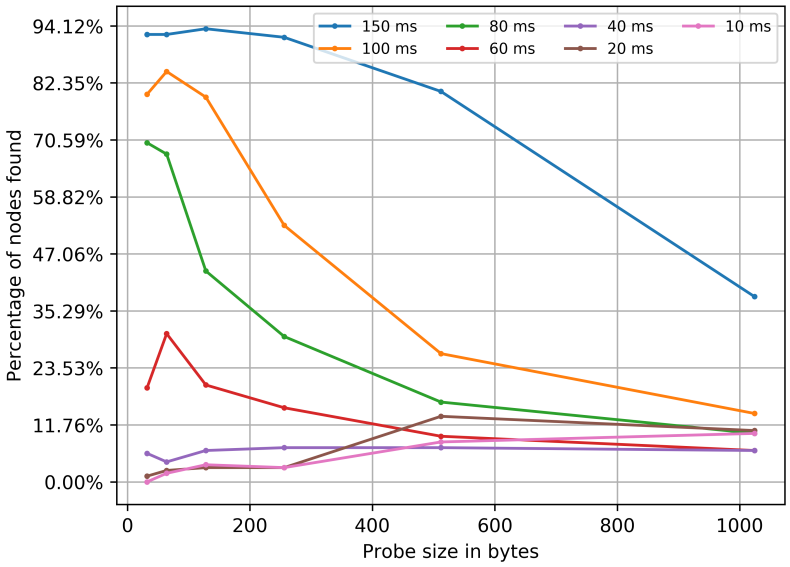

Figure 3.18: Number of IEEE 802.15.4 nodes found with RPL and 1 hop; normal background.

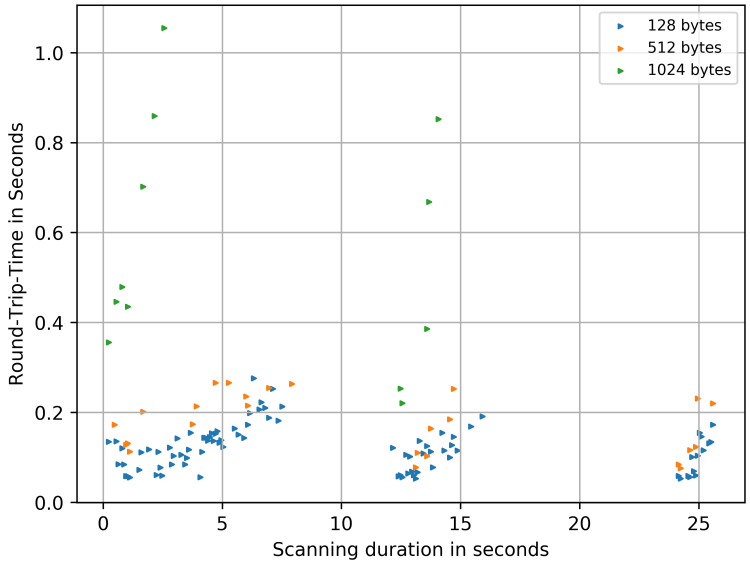

Figure 3.19: RTT for IEEE 802.15.4 with RPL and 2 hops; light background, $T = 80$ ms.

3.4.3 *The complete heterogeneous scenario*

Our previous experiments have shown that IEEE 802.11 and 802.15.4 networks behave quite differently when probed at fast speed or with

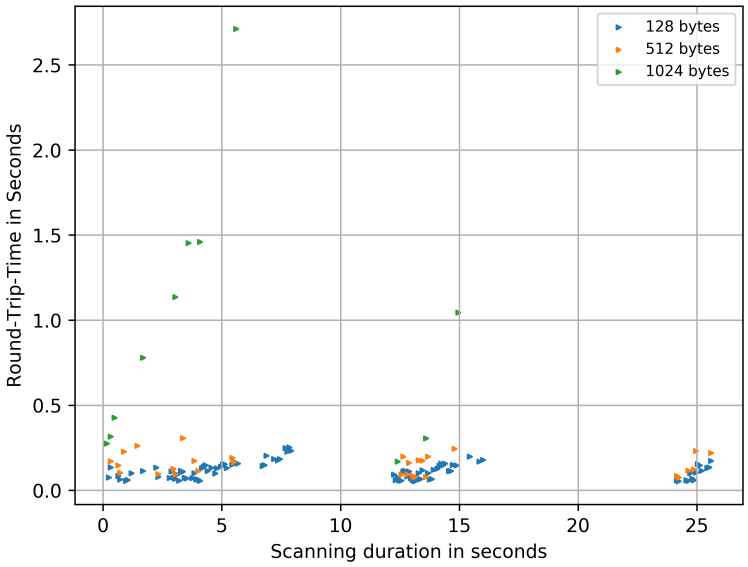

Figure 3.20: RTT for IEEE 802.15.4 with RPL and 2 hops; normal background, $T = 80$ ms.

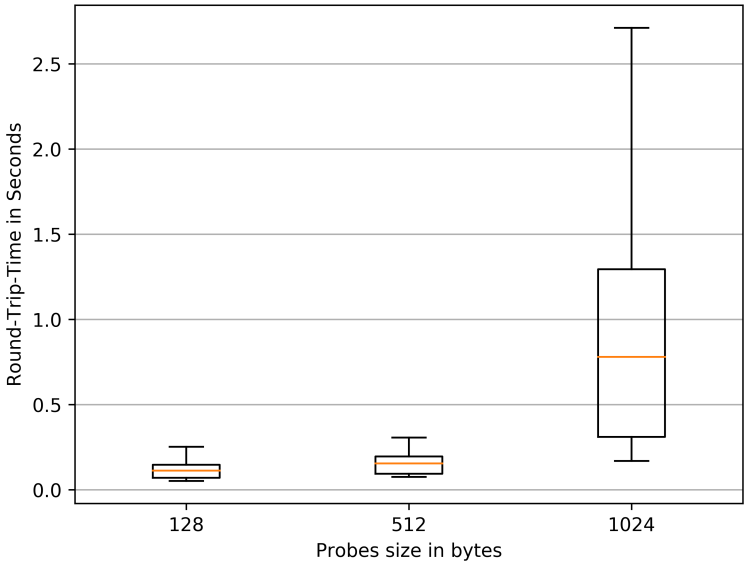

Figure 3.21: Probes size impact on IEEE 802.15.4 with RPL and 2 hops; normal background, $T = 80$ ms.

large probe packets, resulting in large RTTs and high probe losses in 802.15.4 networks. In the following, we measure how our scan approach

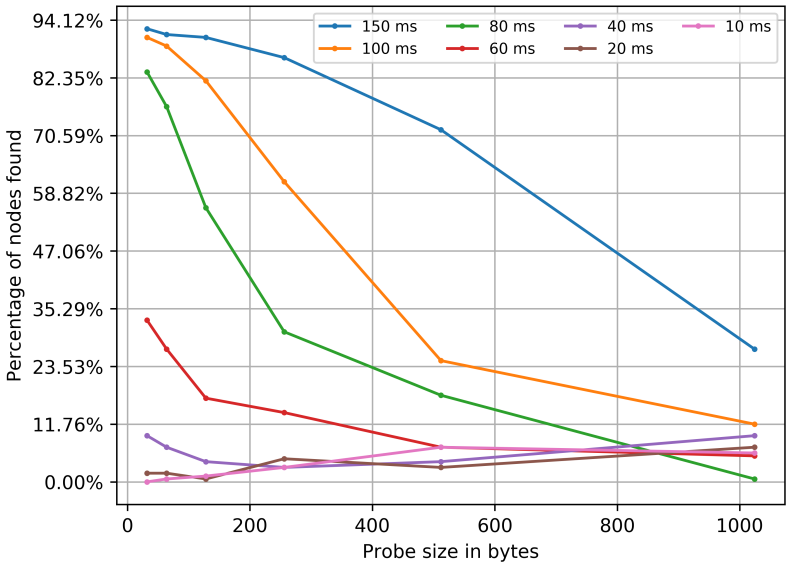

Figure 3.22: Number of IEEE 802.15.4 nodes found with RPL and 2 hops; normal background.

makes use of these effects to improve scan efficiency. As described in Section 3.3, we simulate an IoT network with two sub-networks, containing 100 IEEE 802.15.4 nodes resp. 50 IEEE 802.11 nodes. For our first test, we do not use RPL and all IEEE 802.15.4 are close to the 6LBR.

The first step of our approach is a fast scan of the entire area 2. Figure 3.23 shows the measured RTT with $T = 10$ ms and a 32-bytes probe size under normal background traffic. The low inter-probing time results in the desired very short scan duration, which works well for the second sub-network with IEE 802.11 nodes (and would also work well for wired hosts), but results in many probe losses in the first sub-network containing the IEEE 802.15.4 nodes.

At this point, one might be tempted to directly classify the first sub-network as an IEEE 802.15.4 network and argue that step 2 of the approach is not really needed since Figure 3.23 clearly shows a difference between IEEE 802.15.4 and 802.11 nodes. However, one should not forget that an IoT installation does not necessarily contain two types of networks and there might be other reasons for a large RTT, for example a generally slow network connection, slow reaction speed of nodes, etc.

In step 2, the scan algorithms selects randomly a few nodes and sends slow probes to them. Figure 3.24 and Figure 3.25 show the measured

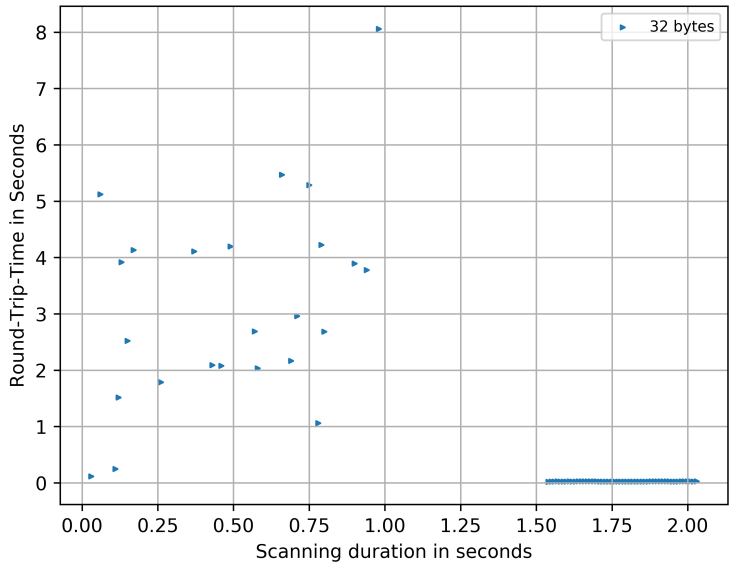

Figure 3.23: Mixed network; no RPL, normal background, $T = 10$ ms, 32-byte probes.

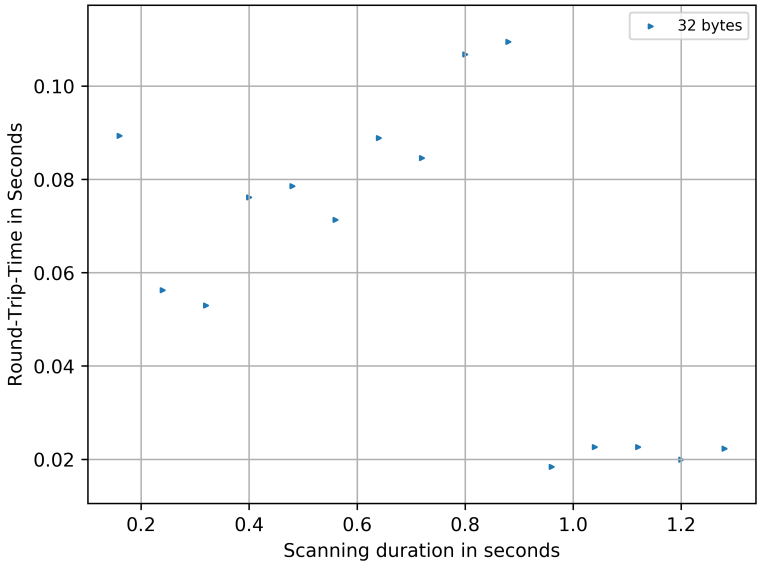

Figure 3.24: Selected nodes; no RPL, normal background, $T = 80$ ms, 32-byte probes.

RTTs for probes of size 32-bytes and 512-bytes and $T = 80$ ms. The

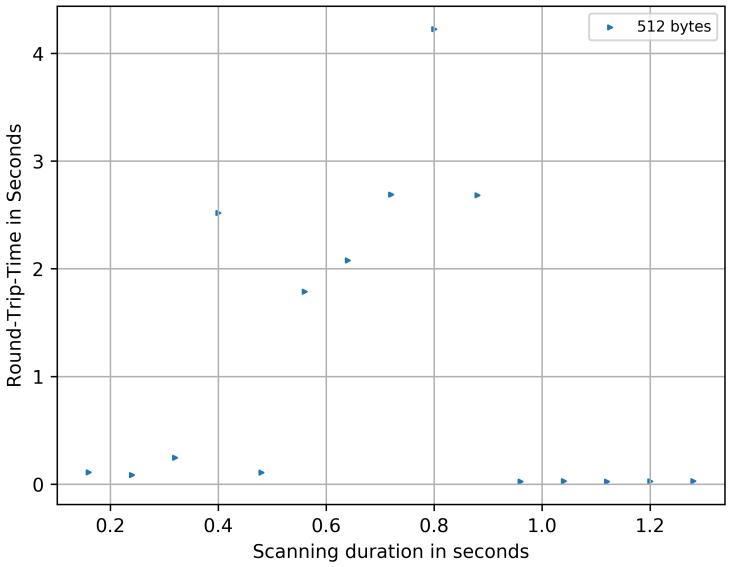

Figure 3.25: Selected nodes; no RPL, normal background, $T = 80$ ms, 512-byte probes.

large increase of the RTT for 512-byte probes despite the slow scan speed clearly identifies the IEEE 802.15.4 nodes.

Finally, in step 3 the part of the address range that contains IEEE 802.15.4 nodes is re-scanned at slow speed ($T = 80$ ms) and small probes (32-bytes). The results are shown in Figure 3.26. In contrast to the fast scan in step 1, we are able to find 96% of the IEEE 802.15.4 nodes. This means that we have achieved a discovery percentage comparable to a slow scan (see Figure 3.12), however without having to scan slowly the *entire* IoT network. Our approach takes 2 seconds for the fast scan in step 1, less than 1.5 seconds for the selective probing in step 2, and 8 seconds for the partial slow scan in step 3, i.e., 11.5 seconds in total, compared to 16 seconds for a slow scan ($T = 80$ ms) of the entire IoT network.

We have omitted figures for the heterogeneous scenario with RPL since they are very similar to the ones already shown. In order to be able to achieve a discovery rate of significantly more than 90% in a two-hop IEEE 802.15.4 network, the scan speed in step 3 has to be decreased to $T = 150$ ms. The duration of step 3 increases by 7 seconds. Consequently, the total scan duration of our approach increases to 18.5 seconds, compared to 30 seconds for a slow scan of the entire IoT network with $T = 150$ ms.

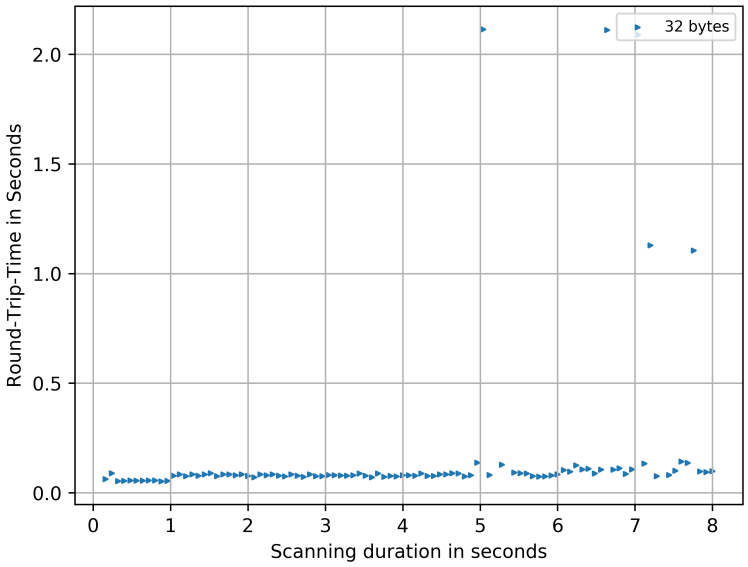

Figure 3.26: Mixed network; no RPL, normal background, $T = 80$ ms, 32-byte probes.

SUMMARY

In this chapter, we have presented a light-weight approach for the efficient probing of heterogeneous IoT networks consisting of IEEE 802.11 and IEEE 802.15.4 sub-networks. Our three-phased approach combines RTT measurements and variations of the probe size and speed to identify the communication technology used by the network hosts. Using this information, we can achieve a high discovery rate by slowly scanning those parts of the IoT network where IEEE 802.15.4 nodes have been detected. We have validated our approach by simulating an IoT network with 170 nodes and two servers that create legitimate (non-probing) background traffic.

The gained knowledge on the network technology used by a particular host can help to fingerprint it. For example, the usage of IEEE 802.15.4 indicates a resource-constrained node, and therefore allows certain conclusions on its OS and application software. This insight could be exploited by the probing party in various ways. We have shown that one of the main characteristics of the Internet of Things, namely its heterogeneity in terms of software and hardware, can lead to new attack patterns.

As we discussed attack description in Section 1.4, we can see that our approach improves the analysis phase since the scanning only can help to find the working service but also lead to the discovery of the type of devices. We expect many approaches, like that to emerge in the near future and the scanning can already harmful to constrained devices so we design our solution to limit this scanning process.

Part II

DETECTION AND MITIGATION

4

STATE OF THE ART IN DDOS MITIGATION

An error does not become truth by reason of multiplied propagation, nor does truth become error because nobody sees it.

—Mahatma Gandhi

INTRODUCTION

In Section 1.5, we discussed the Mirai malware and explained the danger posed by hacked IoT devices when used as bots to launch DDoS attacks against other Internet participants. In Chapter 5 and 6, we will propose a distributed architecture for the protection of IoT systems. The architecture allows to detect and mitigate various attacks such as scanning, DoS, DDoS and reflection attacks directed against and originating from IoT devices and services in a transparent way, i.e. it does not require changes to existing field devices or back-end servers.

Of course, DDoS attacks have existed long before the Internet of Things and the mitigation of DDoS attacks has been the topic of many publications. Before we explain the design of our architecture in the next two chapters and how it can help to defend against DDoS and other types of attacks, we want to summarize the existing work in this area. This chapter is based on the surveys in [MR04, PLR07, ZJT13, DM04]. It should be noted that some of the solutions presented in the following are complementary to our architecture. In the following, we will classify them by the location of the mitigation namely:

- close to the attacker side (source-based mitigation);
- across the network;
- close to the victim side (destination-based mitigation);
- hybrid solutions.

4.1 SOURCE-BASED MITIGATION

4.1.1 *Resource access limitation*

Solutions based on resource access limitation use criteria to authorize or deny communication attempts made by a client. They can be applied to a single communication flow or a set of communication flows. The filtering happens on the client host itself or on a (border) router close to the client.

In the so-called resource consumption regulation approach [GN02], the destination shares its resources equally among all its customers to ensure a good quality of service. All communication flows or packets are thus subject to a strict regulatory policy based on the state of the

target's resources maintained by a QoS server. This system ensures the equitable sharing of resources among all customers. However, it cannot prevent the cooperation of malicious users to monopolize everything.

A client bottleneck approach is proposed by Aura et *al.* in [ANL00] with the idea of sending puzzles to all clients when the destination suspects a DoS or DDoS attack. Clients must then solve the puzzle using a hash function with a parameter k sent by the target. The level of this k parameter is constantly increased as long as the threat is still active. Unfortunately, this leads legitimate users to a point where they are unable to solve the puzzle. This flaw is pointed out by Bocan et *al.* in [Boco4, BFC05] and a time estimatation is proposed as a threshold to limit the value of k. Another problem is that all clients have the same amount of time to solve the puzzle regardless of their respective computation capabilities. This can become a problem if malicious nodes are more powerful than legitimate nodes. Hwang and *al.* [HCO16] improve the client puzzle approach by adding the constraint of equitable computation time based on client characteristics so that even the least powerful nodes can access the service without restriction.

In the repression approach [IB02], the target of the attacks warns the last router on the path to it by sending it the signature of the malicious traffic (mainly the protocol used). The congestion is identified using an Aggregate-based Congestion Control (ACC) where the malicious traffic represented the aggregation traffic going to the destination. Incoming traffic to the upstream router is checked, and the source with the most attack traffic is subject to a speed restriction. This "pushback" continues from the last router before the target to the source of the attack.

MANAnet [Man] is a system that uses an inverted firewall that only filters outgoing traffic. MANAnet differentiates between high-load attacks and flood attacks by examining the second communication flow. MANAnet's authors assume that in authentic communications, hosts respond to incoming flows and therefore classify as malicious all traffic flows that do not have a correspondent.

4.1.2 *Traffic selection*

Traffic selection is a method where the genuine traffic has a signature known by specific users. This method allows to discriminate between good users and reject traffic from other sources. In the filtering method (see next section), we try to determine malicious traffic and remove it

blindly, whereas with traffic selection we distinguish good users and remove all other traffic.

In [Bru02], Brustoloni introduces the idea of VIP customers. VIP tokens are given to customers and are used to prioritize their traffic whether it is malicious or not. This solution assumes that the token cannot be counterfeited and that only paying users will have access to the service. But the authors do not cover the scenario where paid users are compromised and are part of the attack. This solution is interesting when there are only a few VIP customers, but if it is adopted worldwide, the risk to have DDoS attacks from compromised VIP customers increases proportionally.

Khattab et *al.* based their solution on proactive server roaming. The active server changes its location periodically and can only be accessed by legitimate clients [Kha+03, San+04]. When disabled servers receive traffic, they identify it as malicious and drop it directly. Clients periodically update the server location and the selection of the right server is done by a formula in which the server sends the parameters used to determine the next location.

4.1.3 *Traffic Filtering*

The traffic filtering method consists of using ingress or egress filtering solution to dispose of malicious traffic. The filtering rules are found by monitoring the traffic and detecting malicious traffic or removing spoofed communications.

Ingress router filtering is used to block packets with spoofed source IP address close to the source of the attack [Fer00]. This prevents reflection attacks where the spoofed IP address is the address of the target in a different network. However, it cannot prevent attacks with source IP addresses matching the source network (e.g. SYN flooding attacks) in IPv6 because even the smallest network has around 2^{64} possible IP addresses.

The D-WARD system [MPR02, MPR03, MR05] is an inbound and outbound traffic monitoring system for the detection of attacks and malicious packets. It aggregates statistical information on traffic observations and performs traffic throttling. To detect malicious behaviour, the monitored traffic is compared against predefined normal flow patterns. The authors of [Mer+17] notice that accidental throttling of benign traffic penalizes constrained IoT devices that don't have the necessary network and energy resources to retransmit dropped segments. Their

proposed solution FR-WARD sends a series of signals to the sender to shrink the size of the sender's TCP congestion window. If the sender shrinks the window, its connection is relabelled as benign.

MULTOPS [GP01] is a monitoring tool that analyzes flows by examining the disproportionate rate of incoming and outgoing packets. An heuristic targets asymmetric communication flows, assuming that a normal flow has a proportional number of packets in both directions. TOPS [Abd+03] solves a major weakness of MULTOPS that can be exploited by memory depletion attacks and that occur when the behaviour of a large number of hosts must be tracked. TOPS also uses asymmetric heuristics but introduces a threshold to reduce the false-positive rate when the packet rate is low.

4.2 MITIGATION ACROSS THE NETWORK

4.2.1 *Traffic selection*

Gonzalez et *al.* [GAJ11] assume that intermediate devices such as firewalls and routers can also be compromised and play an active role in DDoS attacks. They then introduce a special *judge* as an independent entity to monitor the network without adding ingress/egress overhead costs. The judge uses metrics such as communication delay and the number of packets to identify compromised routers. A confidence value is calculated based on a Bayesian inference model. When an malicious router is found, the judge warns all its neighbours to remove it from the routing paths.

4.2.2 *Traffic Filtering*

Packet filtering in the network is similar to source filtering, but happens on intermediate routers. In [PL01b], Park et *al.* propose filters inside Autonomious System (AS). An undirected graph algorithm based on Internet AS graph is used to transfer the packet to the next router if it comes from the usual set of source addresses, otherwise, the packet is dropped.

Probabilistic packet tagging [PL01a] is an algorithm used to track packets with spoofed addresses to find the real sender. An evaluation of this method was conducted in [CTK16] for IoT traffic. The authors

highlighted the difficulty inherent in such traceback methods when the size of the address range changes from IPv4 to IPv6.

Another approach is to monitor routers instead of end hosts [Bra+98, HABoo, MSM08]. Monitors detect DDoS traffic by checking for routing errors, dropped packets and packet flooding and by comparing the behaviour of a router with its neighbours. This can be implemented in a centralized or distributed way. One of the main disadvantages of this approach is the potential overloading of the network by monitoring messages. Another major disadvantage is the need to upgrade all routers on the Internet to support this feature.

In [Ste+18], the authors propose to mitigate DDoS attacks on Internet Service Provider (ISP) level Moving Target Defense (MTD) and Software-Defined Networking (SDN) techniques. The MTD is done at network-level referring to changes of network graph (e.g., IP-hopping, random port numbers, extra open ports, fake listing hosts) and at host-level by changing their Operating System (OS) configuration. Their approach requires ISP to share their networks and to allow the modification of their routing rules. This level of trust between the ISP can be difficult to reach and the effectiveness of the solution will depend on the number of ISP participating.

4.3 DESTINATION-BASED MITIGATION

Many DDoS attacks use spoofed source IP addresses to reflect traffic from healthy machines to the target. Several authors have proposed techniques to determine the legitimacy of the packet source. In [JWS03, WJS07], the authors look for inconsistencies between the source IP address of an incoming packet and its hop-count as determined from the TTL field in the IP header. However, this technique does not work well in the presence of NAT since the TTL is no more consistent for every hosts inside the same network. Unfortunately the NAT decrease the TTL value which lead to false positive with hop-count.

Packet marking is another method to identify malicious traffic. Yaar et *al.* [YPS03] propose a path identification approach where a fingerprint of the network path is added to each packet. This allows the target to filter incoming packets according to their path taken through the network. Attack traffic sent from the same source can be detected, even if spoofed IP addresses are used. However, adding a complete path fingerprint to each packet is very expensive. The authors have proposed

coding schemes to reduce the size of the fingerprint. Unfortunately, this increases the number of false positives.

Tracking mechanisms can also be used to mark packets from malicious users, so that the victim can identify their source [JS09]. In [BC00, Ram+17], a traceback technique is used to trace the origin of the packets from the router near the victim to the source. These solutions need the majority of the routers to support the mechanism, though. In [Che+16], Chen et *al.*, instead of tracking an IP address, follow a cluster of IPs with the same behaviour.

In [PLR03], the authors use the victim's communication history to identify malicious traffic. The IP addresses and the number of packets previously exchanged are stored in a database and used when the system is under attack. Only information from regular users is stored to minimize space consumption. The drawback of the system is that an attacker could also register their bots on the target as regular users before launching the attack.

Checking the level of congestion on routers can also help to detect incoming flooding attacks and to start mitigation measures before the target is brought down. Packetscore [Yoo+04] assigns a score (based on Bayesian-theoretic metric of each packet namely the Conditional Legitimate Probability) to each packet and if the score is below a threshold, the packet is dropped. The strategy is to maintain the CDF of the score of all incoming packets and calculate the cut-off threshold for each packet entering in a defined interval. The stability of the model depends on the duration of the analysis, the number of parameters used for the score and the behaviour of the network. Inconsistencies in the results of the method can be overlooked for short intervals but they make the method unreliable overall.

In commercially available DDoS protection services for web sites, the mitigation is done at the target side and consists in redirecting, filtering and absorbing the attack traffic by proxies and CDNs. The authors of [Jak+17] propose the usage of network function virtualization (NFV) to allocate dynamically resources to deal with DDoS traffic. Using a dispatcher and on-demand agents, they distribute the traffic to avoid congestion.

4.4 HYBRID MITIGATION SOLUTIONS

Hybrid solutions are combinations of several of the approaches presented above. They are often distributed solutions, making an inband

or out of band communication channel necessary for the exchange of information between all involved entities.

In [CP05], Chen et *al.* create a solution using pushback, packet marking and filtering concepts to mitigate DDoS attacks. The Attack Diagnosis (AD) system consists of three components at the source, router and destination level. The source components start the marking process when an attack has been detected by the victim end where all the packets are analysed. The marked packets is then used by a pushback mechanism to find the attack source. AD is used only to handle DoS attacks while Parallel Attack Diagnosis (PAD) is used to stop DDoS attacks by managing the marking and filtering of many attack source. Both methods require modifications of the routers. Furthermore, the approach uses the Time To Live (TTL) field for the packet marking and will not work correctly if the field is modified unexpectedly by other network nodes.

Defensive Cooperative Overlay Mesh (DEFCOM) [MRR03, Oik+06] is also a solution with components present at the source, the network, and the destination. Nodes exchange various types of messages containing attack alerts, rate limitation instructions and traffic classification information via a P2P channel. Packets are marked by routers along the path from source to destination. The mark is placed in the fragmented traffic field. Since marking only begins after an attack has been detected, it does not significantly affect communications using this field.

In [Pap+03], Papadopoulos et *al.* present a distributed solution located on peripheral network routers. Many assumptions are made in the COordinated Suppression of Simultaneous Attacks (COSSACK) approach, namely the existence of methods to apply ingress/esgress filtering, the prevention of spoofed addresses and the existence of signatures for every attacks. *Watchdogs* are placed on the routers to monitor the traffic. The watchdogs use various inputs such as SNMP statistics and netflow records. When an attack is detected, the victim's watchdog broadcasts a notification to the other watchdogs whose nodes are part of the attack. These watchdogs then filter traffic in order to stop the attack close to the source.

Stateless Internet Flow Filter (SIFF) is a mitigation solution based on flow filtering to stop traffic at the source. It uses a marking approach called *capability* in [YPS04]. A SIFF server provides a capability token to legitimate clients. Network traffic is divided into privileged and unprivileged traffic. Privileged channels are established through a capability exchange handshake. Capabilities are verified by the routers

in the network and can be revoked when an attack has been detected, resulting in the offending traffic to be dropped by the border routers.

Active Internet Traffic Filtering (AITF) [AC09] is a defense service located in the network. Targets can contact a misbehaving source and ask it to stop sending it traffic. The source's ISP must ensure its compliance, otherwise it risks losing all access to the target. AITF verifies the legitimacy of a target's filter request using a three-way handshake. The authors of [LYL08] notice that the handshake becomes difficult to execute if the target is under attack and they propose an independent infrastructure service called StopIt that handles filter requests. When a target detects an attack from a certain source, it sends a filtering request to its access router which forwards it to the StopIt server of the AS the target is located in. That StopIt server then contacts the StopIt server responsible for the AS of the source, which on its part forwards the filtering request to the source's access router. The router finally delivers the request to the source host.

Traffic Validation Architecture (TVA) [YWA05, YWA08] is located at the source, router and destination and requires the modification of the current Internet infrastructure. Similar to Sniff, TVA uses a capability-based approach. However, the marking here is not only used for tracing purposes but also to represent the state of the packet, i.e. whether it is legitimate or malicious. In this approach, clients obtain short-term authorization before starting their communication. The capability is represented here by a tag added to the packets in form of an IP protocol extension. The capability is verified by each router and is also used to limit the level of bandwidth usage by messages using TVA to less than 5% of the general traffic. One of the problems of this method is the spoofing of source IP addresses, but in [LYL08], Liu et *al.* propose an improved version TVA+ that adds source authentication.

SUMMARY

DDoS attacks are devastating to all services on the Internet. The major challenges are to detect them early and to distinguish legitimate communications from malicious ones. Many approaches have been proposed in the literature. Mitigation solutions mainly consist of filtering unwanted traffic or deploying more resources to absorb attacks.

The location of the detection and mitigation has a significant impact on the efficiency and performance of the proposed techniques. The best position to *detect* an attack is at the victim side [ZJT13] because a DDoS

attack consists of traffic flows originating from many different sources, each one looking harmless if seen individually. However, *mitigation* at the victim side has several drawbacks. Methods such as resource limitation or traffic selection often add segregation when they only allow specific customers to access their architecture. Even more important, the mitigation can become ineffective if the attacker manages to exhaust network resources in front of the target. Finally, it does not prevent the attack traffic to harm the Internet backbone. For that reason, DDoS attacks should be ideally stopped as close as possible to the attacker side, as concluded in [DM04, PLR07]. Indeed, many of the approaches presented in the previous sections perform the detection at the target and use techniques such as pushback to do the mitigation close to the source.

Solutions that involve the core network have the advantage that mitigation actions can be handled on the core routers which are typically more powerful than the equipment found at the source or target. However, such solutions involve changes in the Internet infrastructure or require a certain degree of cooperation between ISPs or AS owners.

Hybrid mitigation solutions have many advantages of the other solutions and use techniques such as tagging to only affect malicious traffic. Furthermore, the detection and mitigation process can be distributed to reduce the load on individual hosts or routers. As a drawback, they sometimes require extensive communication between the different components and therefore add burden to the network. A summary can be found in Table 4.1.

Table 4.1: Summary of the solutions discussed in this chapter

Position	Techniques	Pros	Cons
Source-based	Resource access limitation	Resources shared equally	Malicious users can monopolize
	Traffic selection	Only VIP traffic authorized	Danger of compromised VIP devices
	Traffic filtering	Prevents reflection attacks	Sensitive to false positives
Across the network	Traffic selection	Efficiently detects malicious routers	Only targets routers
	Traffic filtering	Bandwidth attacks efficiently detected	More load on the routers
Destination-based	*many*	Attacks are accurately detected and classified	The mitigation difficult if the throughput is big
Hybrid	*many*	Accurate detection and powerful mitigation	Lot of information exchange to coordinate

5

A DISTRIBUTED MIDDLEBOXES ARCHITECTURE FOR IOT NETWORK PROTECTION

Men are mortal. So are ideas. An idea needs propagation as much as a plant needs watering. Otherwise both will wither and die.

—B. R. Ambedkar

INTRODUCTION

The rise of the Internet of Things (IoT) has resulted in a world-wide deployment of billions of small connected devices. Security-wise, these devices pose a major challenge. As explained in Chapter 1, they are often too resource constrained to run sophisticated intrusion prevention software. In addition, by their nature, these devices are easily "forgotten"; once deployed, their firmware is not updated regularly and they do not receive the same attention by network managers as, for example, a desktop PC. Their vulnerability and their number make them an attractive target for attackers since, when hacked and infected with botnet software, they can be misused for powerful large DDoS attacks against other Internet participants [DD+17, KKS17, GWZ, Con, Cid], as shown by recent DDoS attacks achieving 600 Gbps [BI17] and 1.2 Tbps [Hilce]. The state of the art in defending against such DDoS attacks is attack mitigation at the target. Companies like Arbor Networks, Renesys, Cloudflare, or Akamai use CDN networks to filter and absorb malicious traffic. However, although this approach is effective from the viewpoint of the target, the attack traffic still consumes resources on the path from the source to the mitigation point and the mitigation does not prevent a botnet from attacking other, potentially less protected hosts.

This situation is further complicated by the increasing complexity of the IoT ecosystem, which does not only consist of sensor devices but also encompasses intermediate data handling nodes, servers to store and process data, and devices hosting the end-user interfaces (e.g. smartphones). All these components are managed independently and by different operators and can be combined to build new applications. For example, an application might need real-time sensor data from the IoT networks of two Smart Cities A and B, historical data stored in a data-center C, and the data mining capacities of a cloud D. Protecting such a complex multi-party environment from abuse becomes a very challenging task. New difficulties arise everyday when policies are updated or new collaborations and federations appear between entities. On the other hand, a pragmatic approach where all devices and services are allowed to mutually access each other without restrictions can have dramatic consequences in the case an attacker manages to gain control over a sufficient number of hosts, as demonstrated by the recent DDoS attacks.

To address the above challenges, in this chapter we propose a permission-based system of cooperating middleboxes that monitor and control the communication between IoT networks (or sub-networks). The middleboxes are interconnected through a management layer that allows to implement different access policies and to propagate information on detected attacks. Since the middleboxes operate on network level they can be easily deployed in existing IoT networks without requiring modifications on the end hosts. Traffic monitoring and filtering by the middleboxes is bidirectional and flow-based. We validate our approach by experiments in a virtual environment where we demonstrate how different types of attacks can be stopped at the source and effectively mitigated.

This chapter is organized as follows: We present related work in Section 5.1. The architecture of our system is detailed in Section 5.2. The experiment methodology is described in Section 5.3, followed by the presentation and discussion of experimental results in Section 5.4.

5.1 RELATED WORKS

In this section, we discuss works related to this chapter in the fields of access control for IoT networks and the mitigation of DDoS attacks.

Access control in IoT networks

By default, many resource-constrained IoT devices only implement a minimum level of access control, for example in the form of a password request, because complex access control systems are resource consuming. In addition, the typical activity of an IoT device, i.e. providing sensor data, is short and sparse and discourages the use of security protocols with a large overhead.

Researchers have proposed more advanced techniques for access control. Liu *et al.* [LXC12] implement the idea of a register authority (RA) to authenticate users and then to control device accesses inside the IoT network to reduce the impact of vulnerability exploits. Their solution requires that nodes that are contacted by other hosts first communicate with the RA to check whether access should be granted. Cruz-Piris *et al.* [CP+18] proposed to extend the MQTT protocol by user authentication and access rules. Since MQTT is push-based, the access control is implemented on the servers, which are, in general, more powerful than the sensor nodes. No solution is proposed for

connection attempts toward the sensor nodes, as done, for example in pull-based protocols like CoAP or during management activities. Novo [Nov18] proposes an architecture using a block-chain to create a distributed access solution where IoT nodes send their traffic through management hubs providing access control. In the opposite direction, i.e. when receiving traffic, a node has to ask its hub to know if it should answer. In Xu *et al.* [Xu+18], a host has to request a capability token from a Local Coordinator before it can access other hosts. Policies are stored in the cloud where the coordinators can access them.

All the above approaches require an active participation of the sending or the receiving device in the access control. This makes them unsuitable for scenarios where the existing software on the IoT nodes cannot be modified or where constrained nodes do not have enough resources to implement sophisticated protocols. For this reason, we propose to enforce access control in the network without relying on the end nodes.

Another aspect is the management of the access control itself which can become complex in multi-party and federated IoT environments. We have many types of devices, added and removed from time to time with different access rules depending of the destination. With its architecture of local coordinators and higher-level management implemented in the cloud, the previously mentioned publication by Xu *et al.* [Xu+18] addresses federated systems. We follow a similar hierarchical design. However, in our system, access control is implemented in middleboxes that monitor and filter the network communication between the collaborating systems. In contrast to [Xu+18], the end nodes are not directly involved.

DDoS mitigation

DDoS attacks are currently one of the most powerful attacks to bring down networked services. In a DDoS attack, traffic generated by a large number of hosts and amplified through different techniques floods the target. As already noted in Section 4.4, the best location to detect an attack is at the victim side [ZJT13], and the attack should be ideally stopped at the attacker side [DM04, PLR07]. We follow this idea and have designed a system such that the attack traffic is stopped at the attacker's network border. This mitigation close to the source prevents that the attack travel across the Internet and that the attacker can continue attacking others, less protected hosts. However, our system

also allows blocking unwanted traffic at the destination's network border if it cannot be stopped at the source.

Like StopIt [LYL08] and AITF [AC09] (see Section 4.4), our system allows a target system to contact a misbehaving source and ask it to stop sending it traffic. However, in our design, individual end hosts are not involved in the attack detection and mitigation process. As explained in Section 5.1, we cannot assume that the software on existing IoT devices can be adapted or that constrained nodes have enough resources to implement the necessary protocols. Instead, we let middleboxes handle all detection and mitigation activities. Unlike the StopIt servers in [LYL08], our middleboxes do not require the collaboration of different AS thanks to a hierarchical organization.

In Yang et *al.* [YWA08], clients obtain short-term authorization before starting their communication. Permissions are granted per end device and recorded directly inside the packets. All routers on the path to the target verify the permissions of a packet before forwarding it. We also use short-term authorizations in our system. However, the authorizations are negotiated between the middleboxes responsible for the source and destination host, respectively. Again, our motivation is to provide a transparent security service to IoT systems without requiring modifications of existing IoT devices or core routers.

5.2 DISTRIBUTED MIDDLEBOX ARCHITECTURE

In this section, we describe our approach toward distributed permission-based access control and attack detection and mitigation. We first motivate our approach in Section 5.2.1, followed by a system overview in Section 5.2.2. The resulting architecture is discussed in Section 5.2.3 and 5.2.4.

5.2.1 *Motivation*

The Internet of Things is not a single network of homogeneous devices and services. Its various components, that are sensors, servers, end-user devices, etc., are hosted in small and large networks owned and managed by different organizations. Building complex IoT applications means to aggregate and process data from different sources and at different places on the Internet. Thanks to Machine-to-Machine communication this can happen without human intervention. However, it

requires that the different entities have agreements on who is allowed to communicate with whom. An example is depicted in Figure 5.1. In this scenario, we have various IoT networks providing different types of data and services and users wanting to access them. To collaborate safely, security policies have to be carefully defined and implemented. For example, we want to avoid IoT cameras sending video streams directly to users instead of to the streaming servers.

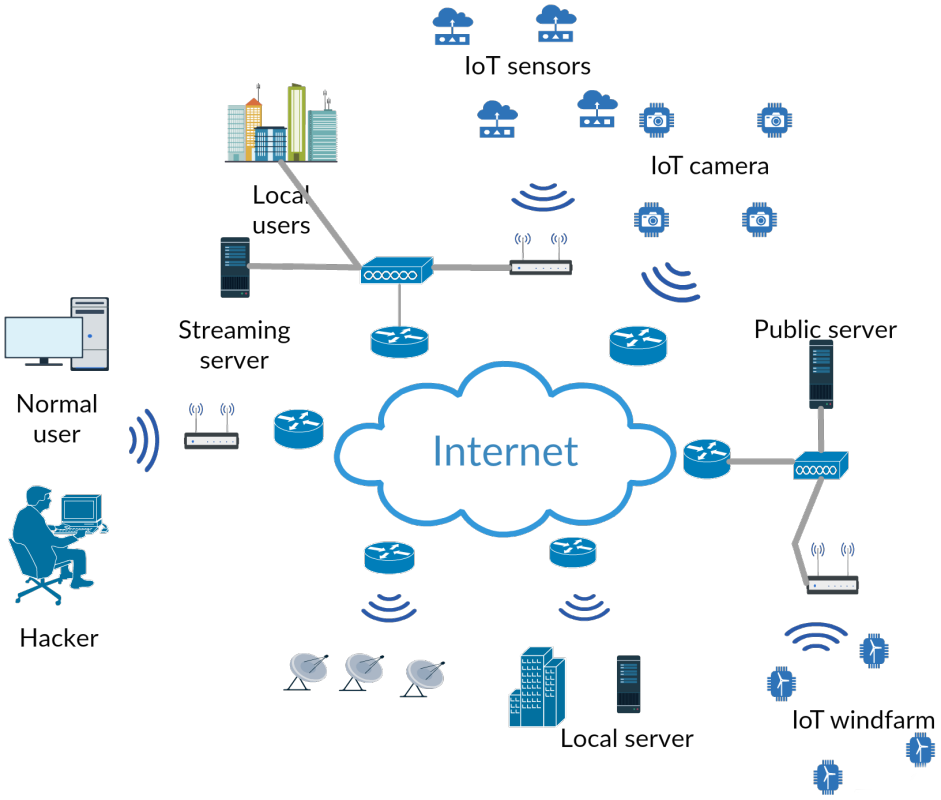

Figure 5.1: Collaboration between different IoT networks

Implementing such policies on all sites is cumbersome and error-prone. Small mistakes can take months to be detected and will expose the affected subsystems to attacks. Although similar problems exist in other types of networked systems, they are particularly serious in the context of IoT with its billions of poorly managed devices. Sometimes, network administrators or, in the case of Smart Homes, end users might not be even aware that vulnerable devices exist in their networks. It is realistic to assume that sooner or later an attacker will gain control

over them and use them to perform attacks against other parts of the system.

5.2.2 *System overview*

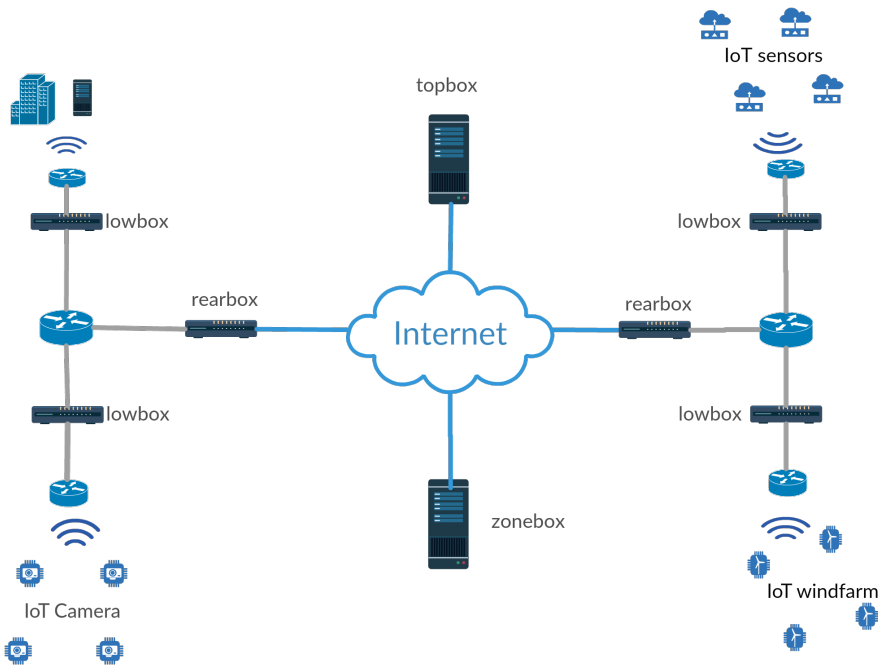

Figure 5.2: Distributed middleboxes architecture

The goal of this chapter is to provide a security system that helps to collaborate with IoT networks to communicate and exchange information without exposing them to attacks. We define the following requirements for such a security system:

1. It should allow us to define and enforce access control between all participating networks.

2. It should operate on network-level since we cannot expect that all vulnerable devices are known in advance and that their vulnerabilities can be fixed.

3. It should be independent of the concrete communication protocols used between IoT components.

4. It should allow to detect DDoS and similar attacks.

5. Once an attack is detected, the system should stop it at the source to prevent that the attack traffic strains the Internet and that other collaborating networks are attacked.

The general idea is shown in Figure 5.2. Our security system consists of middleboxes that are deployed inside or at the border of networks hosting IoT components. These middleboxes (labelled *RearBoxes* in the figure) are placed such that they can monitor and filter the traffic between the network and the Internet. In larger IoT networks consisting of several sub-networks, more middleboxes (labelled *LowBoxes* in the figure) can also be deployed in front of the individual sub-networks because only one RearBox is allow per organization. LowBoxes have the same role as a RearBox except they are under the main RearBox management and their communications are restricted with it. A RearBox/LowBox has two tasks:

1. When a host in a network tries to access a host in another network, the RearBox/LowBox of the target's network checks whether the source host is authorized to access the target host. If not, the traffic is blocked. Similarly, other policies, such as limitations on the number of messages or maximum packets size can be enforced.

2. The RearBox/LowBox looks for attack traffic signatures entering/leaving the network. Since the box only monitors network traffic, our focus in this chapter is on attacks that can be detected by flow-based intrusion detection, such as DDoS attacks, scans, etc.

To achieve both tasks, the RearBox/LowBox are forming the lower layer of a hierarchical structure of security devices. This structure allows to easily implement security policies between all collaborating (or federating) networks. General security policies are defined as rules by the network administrators. They determine which and how much communication is allowed between the IoT components hosted by the different networks, in this way guaranteeing on one hand, a smooth operation of legitimate IoT application and preventing. On the other

hand, unauthorized accesses attempted by hacked devices are quickly detect and stop.

The hierarchical structure also allows middleboxes to exchange information on identified threats. The TopBox/ZoneBox purpose is to allow/ease the communication between the Rearboxes of the system. Concretely, if a RearBox/LowBox detects attack traffic directed to the network under its protection, it will block the traffic and then check if the source of the attack is located in one of the collaborating networks. If this is the case, it will notify the RearBox of the source, which will then filter the attack traffic right in front of the attack source before it can leave the source network. The higher part of the hierarchical security structure (TopBox, ZoneBox), which we call management layer, is described in Section 5.2.3. We explain how the middleboxes work in Section 5.2.4.

5.2.3 *The management layer*

We only presented here an overview of the communication system. More details about the management layer and the messages exchange are shown in Chapter 6. Our system used an hybrid approach to detect and mitigate IoT attacks using middleboxes. The RearBoxes/LowBoxes are the agent responsible of detecting and stopping the attacks when occurred. Since our system stopped the attack at their sources, they need to locate them when needed. We propose a hierarchical design inspired by DNS for scalability. Rearboxes connect to a *ZoneBox* which operates on country level. The registration of a RearBox to its ZoneBox is mandatory. During registration, the RearBox provides a list of IP networks its protects. The communication between the boxes is encrypted, and the keys are exchanged during the registration. The ZoneBoxes are managed by the country or AS they protect. When RearBoxes register under different ZoneBox need to exchange information, they needed, another level to the hierarchy which interconnects the ZoneBoxes (indicated as *TopBox* in Figure 5.2). The TopBox/ZoneBox in the management level had two purposes:

- To provide information about other Rearboxes when needed: A Rearbox SR who needs to find the Rearbox DR responsible for a target or source address (the look-up operation at line 3 in Algorithm 5.2.1 and 5.2.2 respectively) contacts its ZoneBox which might, similar to DNS, forward the request to the TopBox.

- To provide mitigation on large scale: If mitigation at a RearBox fails or cannot be done because the attack source is in a network without a Rearbox, the ZoneBox stop the attack inside the ISPs.

The algorithms in Section 5.2.4 to find the optimal mitigation point, agreements between the operators of IoT networks and their ISPs, etc. It should be noted that such a setup is only required for large scale treatment of attacks. If IoT networks want to collaborate on small scale, only RearBoxes and a ZoneBox are needed. The latter is the scenario that we will evaluate in Section 5.3 and Section 5.4.

5.2.4 *The middleboxes*

In this section, we give details regarding the operations of the RearBox for the protection of the network. We disregarded all the communication processes (registration, lookup) with higher boxes for simplification purpose. All the communications are described in Chapter 6. Those communication processes are only used for the first communication with an unknown box, they are cached for futures exchanges to reduce the communication with ZoneBox/TopBox. We will assume in the following that the RearBoxes already knew each other.

To protect an IoT network, there is at least one box (the RearBox) needed at the network border. Depending on the size of the network, its topology and the security policies the owner of the network wants to enforce there could be one or more LowBoxes deployed inside the network. Only the RearBox communicates with the outside world and it is in charge of configuring the LowBoxes.

The RearBox/LowBox implement permission-based access policies on network flow level. When a source host S sends a packet to a destination host D, the RearBox of the source network first checks its list of permissions if it has a permission matching the flow identifier[1] of the packet. If there is no such permission and there exists a RearBox protecting D, the RearBox of S requests a permission from D's RearBox. If the destination RearBox declines the request the traffic is blocked at the source. Similarly, the destination RearBox will block flows for which no permission has been requested beforehand. Note that cached permissions have a (configurable) Time-To-Live permission.

1 We define the flow identifier as the classic five-tuple (source address, destination address, source port, destination port, protocol).

This two-party permission system allows implementing dynamic access control. For example, a destination network that is experiencing heavy load can decide to temporarily decline new permission requests or issue permissions with a short time-to-live. On the other hand, permissions with a long duration reduce the communication overhead and the latency introduced by the verification process when a new permission is requested from a remote RearBox. Network administrators might also decide to permanently install permissions on a RearBox to make it accept traffic from networks that do not have a RearBox.

Our permission system is related to devices policies. Those policies are used to decide if a permission can be granted if the device is not under attack. The policies for the individual devices contain the following information :

- Information that is used when the RearBox has to decide whether a permission request should be accepted. For example, the network owner could define that only incoming connections towards port 80 are allowed for device 1.2.3.4 or that device 1.2.3.5 does not accept more than 10 simultaneous connections.
- Information used when sending a permission request to another RearBox. For example, the policy could specify that a device 1.2.3.4 needs permissions of 15 seconds when communicating with other hosts.
- What to do with flows from/to the device that are on the alert list, i.e. identified as suspicious? For example, the policy could define that suspicious flows should be blocked or rate limited.

There is also a *default policy* that is used when no specific policy exists for a device or when devices are communicating with a network without a RearBox/LowBox. Policies are installed by the network administrator on the RearBox which duplicates it on the LowBoxes it manages.

A RearBox has four different lists updated in real time. Entries in the permission list, the flow list and the alert list have a timestamp and a duration after which they expire and are removed.

1. The *permission list* contains all permissions granted or denied. A permission contains the flow identifier (the five-tuple), the duration of this permission and finally the time when the permission

was issued. The type of the permission (granted or denied) is also save.

2. The *policy list* represents the policies that the owner of the network protected by RearBox/LowBox wants to enforce for the individual devices inside the network.

3. The *flow list* contains information on all flows currently active, such as its size in bytes, the number of packets, and the duration. This list is scanned by the detection methods for attack patterns.

4. The *alert list* contains the list of flows that have been identified as suspicious. Each entry contains the flow identifier of the affected flow and the type of offense detected for that flow.

Algorithm 5.2.1: Handling of outgoing packets

Input: *permList* : permission list, *policyList* : policy list, *packet*: packet

```
1  f ← flow identifier of packet;
2  if f is not in permList then
3  |   box ← look up RearBox of packet.destination;
4  |   if no box then
5  |   |   apply default policy
6  |   else
7  |   |   policy ← policyList[packet.source];
8  |   |   perm ← requestPermission(f, policy, box);
9  |   |   add perm for f to permList
10 |   end
11 else
12 |   perm ← permList[f]
13 end
14 if perm.type is granted and perm.duration is valid then
15 |   forward packet
16 else
17 |   drop packet
18 end
```

Algorithm 5.2.1 is executed for every outgoing packet. If the packet is part of a new flow, i.e. no permission exists yet (line 2), a request is sent to the RearBox responsible for the destination host (otherwise the default policy is applied in line 5). The obtained permission is added to the permission list (line 9). If the flow already has a permission it is just retrieved from the list (line 12). Then it is checked whether the

permission is granted and still valid (line 14) to decide whether the packet can be forwarded or must be dropped. Note that permissions are stored in the permission list independently of whether they are granted or denied.

Algorithm 5.2.2: Handling of incoming packets

Input: *permList* : permissions list, *alertList*: alerts list, *packet*: packet

```
1  f ← flow identifier of packet;
2  if f is not in permList then
3      box ← look up RearBox of packet.source;
4      if no box then
5          apply default policy
6      else
7          raise alert for f;
8          send alert to box;
9          add permission of type Denied with DefaultDuration for f to
             permList;
10         add f to alertList for offense NoPermission;
11         drop packet
12     end
13 else
14     perm ← permList[f];
15     if perm.type is granted and perm.duration is valid then
16         forward packet
17     else
18         drop packet
19     end
20 end
```

Algorithm 5.2.2 describes the handling of incoming packets. Packets that arrive without permission are either subject to the default policy (if they come from an unmanaged network; line 4) or raise an alert (line 7 for the log) and are dropped (line 11). In addition, an entry is added to the permission list that blocks further packets from that flow for a configured duration (line 9) and the source middle box is notified (line 8). Packets with a permission are treated as usual.

Algorithm 5.2.3 shows the treatment of permission requests. First, we check whether the network administrator has defined a policy for the target of the permission (line 2). We only accept the request if it is compatible with that policy (lines 4–10). Among others (not shown in the algorithm), we check whether the requested duration is below the maximum allowed duration specified in the policy and whether the

Algorithm 5.2.3: Handling of permission requests

Input: *request* : permission request

```
1  policy ← policyList[request.targetDevice];
2  if policy exists then
3  |   perm ← new permission for f with requested duration;
4  |   if request follows policy then
5  |   |   perm.type ← Granted
6  |   else
7  |   |   perm.type ← Denied
8  |   end
9  |   send perm to requester;
10 |   add perm for f to permList
11 else
12 |   apply default policy
13 end
```

device has not reached its maximum allowed number of connections. For devices without a specific policy, the default policy applies (line 12).

In addition to the procedures described above, the RearBox/LowBox performs two additional tasks on every flow:

1. *Intrusion detection:* Traffic is analysed for attack patterns. If suspicious behaviour is identified by the target RearBox, the source RearBox is notified, so it can add the flow to its alert list, too.

2. *Attack mitigation:* Flows on the alert list are subject to mitigation actions. We can have different mitigation strategies depending on the devices and the network administrators. We can completely block malicious flow as a hard approach or follow a softer approach, such as applying a rate limitation. The latter prevents false positives to completely block the normal functioning of an IoT application.

We consider the concrete algorithms for intrusion detection as out of scope of this chapter and refer to the vast literature on this topic [Lun93, ZLH03, GT+09, Zar+17, Cha+19]. We have implemented some exemplary algorithms in our proof-of-concept that we describe in Sections 5.3 and 5.4.

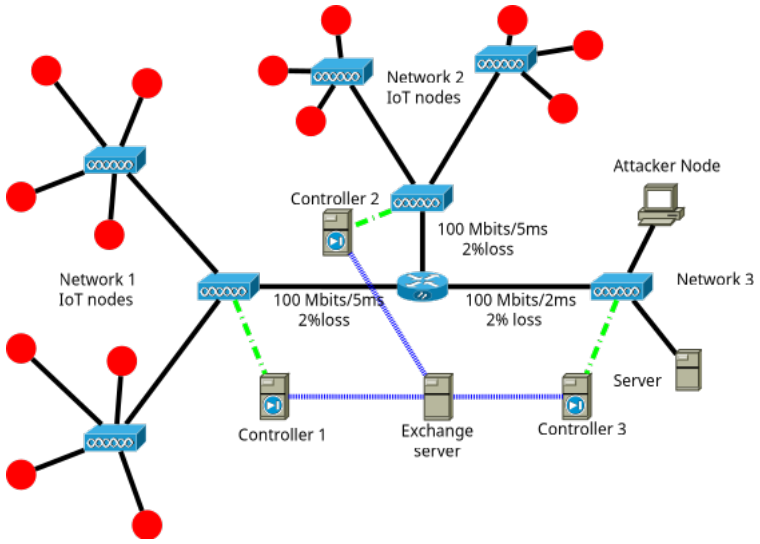

Figure 5.3: Network topology used in this chapter

5.3 EXPERIMENTAL METHODOLOGY

We validate our approach on a network topology emulated with mininet (`http://www.mininet.org`). The topology contains three IoT networks (as shown in Figure 5.3) that are interconnected through a central router representing the Internet:

- *Network* 1 contains 20 nodes representing an IoT network consisting of two sub-networks;
- *Network* 2 is similar to *Network* 1, but with only 10 nodes;
- *Network* 3 hosts two nodes representing a data storage or data treatment server and a malicious node, respectively;

The hosts in *Network* 1 and *Network* 2 represent resource-constrained IoT devices with a wireless connection. To simulate the latter, we set a packet loss rate of 2%, a bandwidth of 100 Mb/s and 5 ms delay. For the server and the attacker in *Network* 3, we choose the same bandwidth but a delay of 2 ms. We do not simulate a TopBox since our main goal is to show the advantage of the mitigation at the attack source in the following. The RearBoxes are emulated by programmable switches at the borders of each of the three networks controlled with the Pox controller [Pox].

Background traffic represents normal interactions between the different networks. The IoT devices in the two networks emulate IoT services that send respectively UDP message of 20 bytes for the data and get UDP reply message of 50 bytes from the server. For simplicity the same default policy is applied to all the nodes inside the system. The policy treats all the nodes as equal in term of priority, i.e. it does not impose limitations on the amount of data they are allowed to exchange nor on the number of connections. The default duration, as used when requesting permissions, is 10 seconds.

Algorithm 5.3.1: Mitigation

Input: *alertList* : alerts list, *policyList* : policies list, *permList* : permission list

```
1  for each alert a in alertList do
2      f ← a.flow;
3      policy ← policyList[a.deviceAddr];
4      if a.offense == DoS then
5          if policy.dos exists then
6              apply policy.dos to a
7          apply default DoS policy to a
8      else if a.type == DDoS then
9          if policy.ddos exists then
10             apply policy.ddos to a
11         apply default DDoS policy to a
12     else if a.type == Scanning then
13         if policy.scanning exists then
14             apply policy.scanning to a
15         else
16             apply default Scanning policy to a
17     if a.deviceAddr == f.destination then
18         box ← look up middlebox of f.source;
19         send alert to box if it exists;
20     end
21     remove permission for f from permList;
22     remove a from alertList ;
23 end
```

We perform experiments with different types of attacks. We describe for each experiment the concrete background traffic used in that experiment, the nature of the attack, and the method used by the RearBoxes/LowBoxes to detect it. It should be noted that our goal is not to develop new detection methods. For example, we are using, as a

proof of concept, a simple threshold-based detection method for DDoS attacks in the experiments, but our system would also work with other, more sophisticated network-based detection methods, for example the one proposed in [DAF18].

The basic mitigation procedure run by the RearBoxes/LowBoxes is identical in all experiments and shown in Algorithm 5.3.1. The controller scans the alert list, i.e. the list of flows marked as suspicious by the detection method, every 500 ms and executes for each alert the mitigation action depending on the offense type indicated in the alert. In our experiments, the default policy is to block the flow. It should be noted that the detection and mitigation procedures run on all RearBoxes/LowBoxes, i.e. an attack flow can be potentially detected and mitigated by the RearBox/LowBox of the source as well as by the RearBox/LowBox of the destination according to their policies. If the destination RearBox/LowBox detects the attack it will send an alert to the source box (lines 17–20). Finally, the permission of the affected flow is revoked (line 21) and the alert is remove from it list as well (line 22).

5.4 EXPERIMENTAL VALIDATION

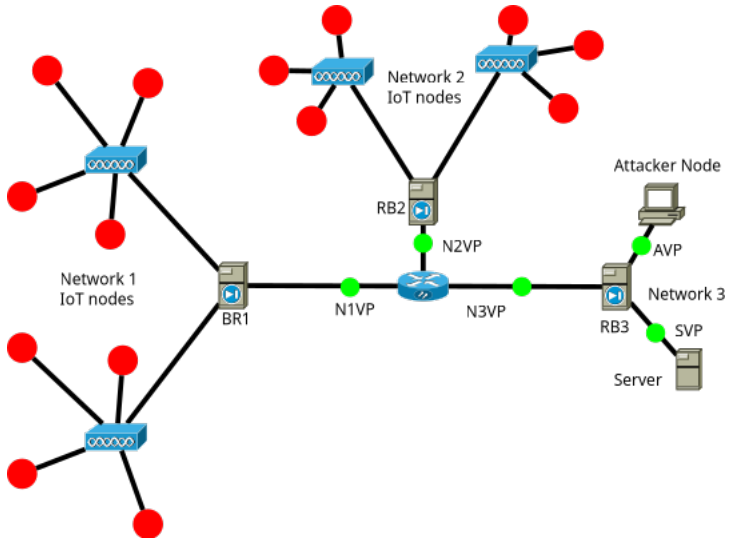

Figure 5.4: Network topology used in this chapter

For the following experiments, we deploy RearBoxes in the network from Figure 5.3 and collect traffic data at different vantage points. Their locations are shown in Figure 5.4. Network 1 is protected by RearBox

RB1, Network 2 by RB2 and Network 3 by RB3. The vantage points used to record the data are N1VP for Network 1, N2VP for Network 2, N3VP for Network 3, AVP for the attacker device and SVP for the server.

5.4.1 *Scenario 1: DoS attack*

In our first experiment, the malicious host in *Network* 3 sends UDP packets of 500 bytes with an inter-packet time of 3 s against a set of 15 random nodes of *Network* 1. This simulates a generic DoS attack where an attacker tries, for example, different messages to exploit a vulnerability. For the background traffic, 10 nodes in *Network* 1 send UDP packets of 20 bytes to random nodes in *Network* 2 with an inter-packet time of 5 s. The nodes in *Network* 1 reply with a UDP packet of 50 bytes to each incoming packet. We record the traffic at N1VP, N2VP, N3VP and AVP for this experiment. The system is configured to detect a DoS attack when a node exchanges at least 5 packets with a destination and $\frac{size\ of\ flow}{flow\ duration} > 100$. We implement a hard mitigation approach and permanently stop the malicious flows.

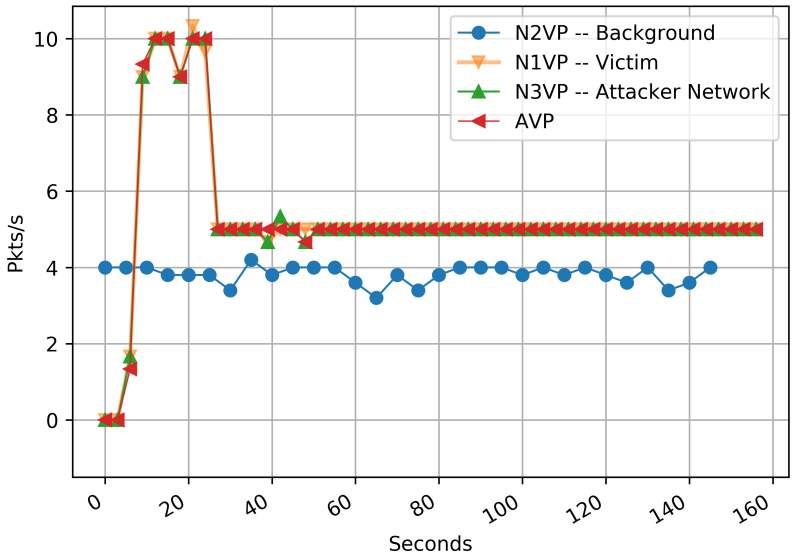

Figure 5.5: DoS attack with hard mitigation strategy by target RearBox

Figure 5.5 shows the number of packets per second observed at the different vantage points. The background traffic is mostly constant,

except for a few losses caused by the configured packet loss rate (the emulated IoT application does not resend lost packets). The attack starts at time 9 s and is mitigated when the detection conditions are reached at around time 27 s. To better illustrate the functioning of the system, we first let only the RB1 run the detection and mitigation process. We can see, the traffic observed at AVP drops from around 10 packets/s to 5 packets/s since RB1 stops the attack packets before they can reach the IoT nodes. The legitimate background traffic is not affected as shown by N2VP.

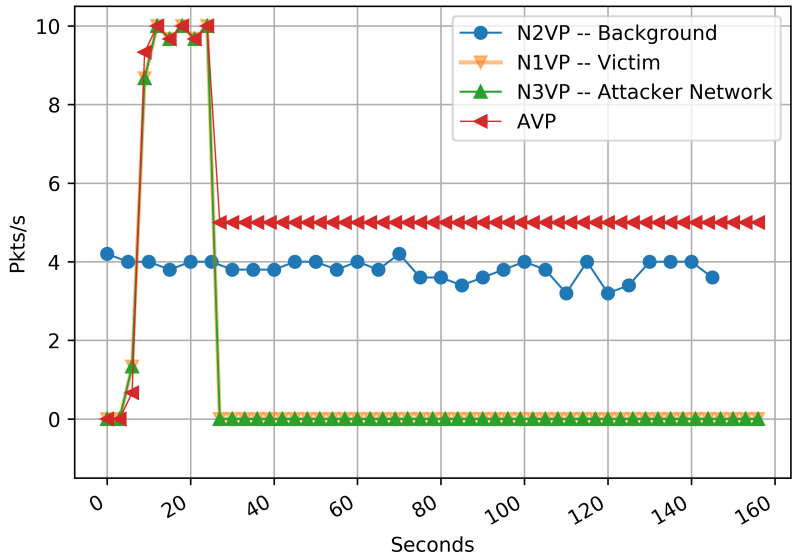

Figure 5.6: DoS attack with d hard mitigation strategy by source RearBox

For Figure 5.6, the RearBox on the attacker side is enabled, too. Once the attack is detected, the RearBox blocks the attack traffic at the source for an infinite duration, hence the drop to 0 packet/s after mitigation.

Alternatively, we could use a policy that blocks attack traffic only for 60 seconds. This is shown in Figure 5.7 to illustrate the flexibility of the RearBoxes.

5.4.2 *Scenario 2: DDoS attack*

In the second experiment, 10 nodes in *Network* 1 and 10 nodes in *Network* 2 send UDP packets of 50 bytes every second to the server in *Network* 3. The goal is to simulate a small DDoS attack against

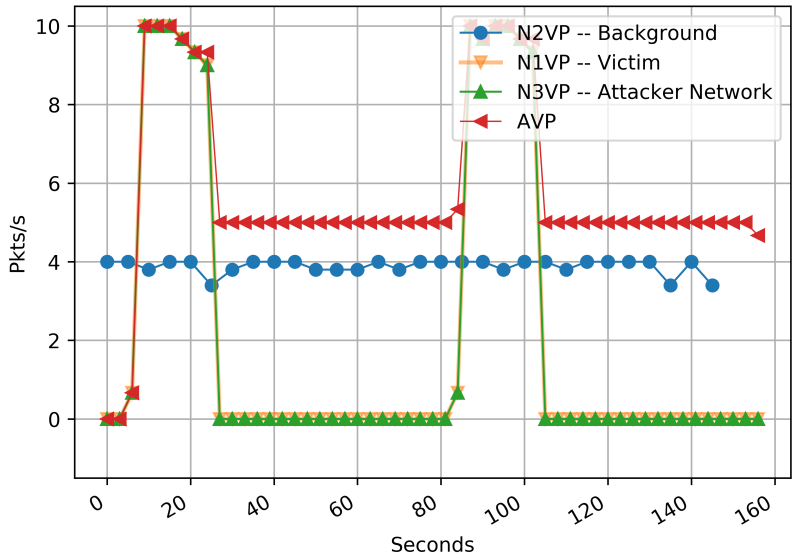

Figure 5.7: DoS attack with soft mitigation strategy (60 s blocking) by source RearBox

the server. We are using the same background traffic as in the first experiment with the DoS attack. The RearBoxes detect a DDoS attack when a host receives more than 5 kBytes (size of all the flow) and 4/5 of the incoming flows exceed a duration of 15 s. Instead of completely blocking the attack source like in the DoS attack, we implement a less aggressive mitigation policy on the target RearBox that still allows to provide a basic service to legitimate requests coming from the source even when under attack: The mitigation consists in blocking one third of the flows every time the thresholds for the total flow size ($\sum_{f \in flows} f_{size}$ > 5 KBytes) and flow durations (4/5 of all incoming flows have a duration > 15 s) are exceeded at the target box. This is repeated until the attack traffic falls below the detection thresholds.

Figure 5.8 show the results when the traffic is only mitigated by the victim side (we do not show the background traffic separately in order to keep the figure readable). After mitigation, traffic is still sent from *Network* 1 and 2, but it does not reach the target.

In contrast, no attack traffic leaves *Network* 1 and *Network* 2 if the source RearBoxes participate in the mitigation (Figure 5.9).

The experiment demonstrates how network administrators can reach very specific goals in terms of quality of service by adapting the mitigation policy. For example, the system on the victim side could also apply

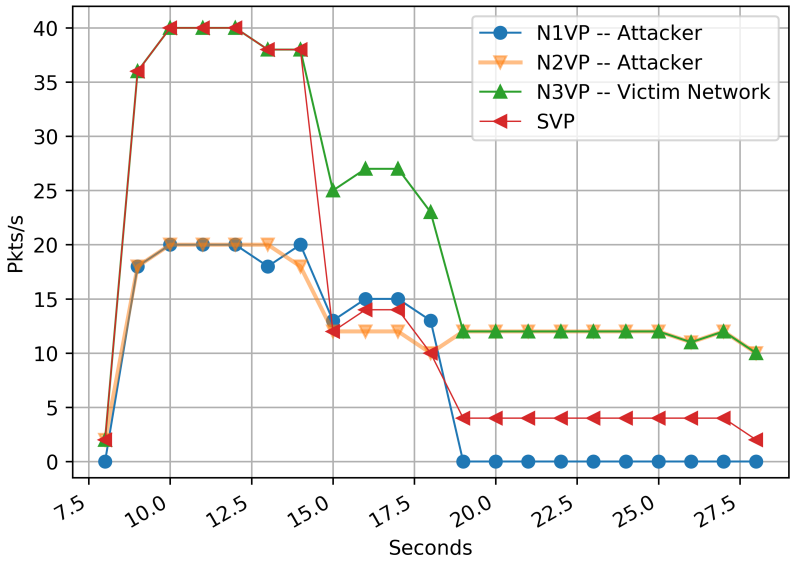

Figure 5.8: DDoS attack with different mitigation strategies by target RearBox

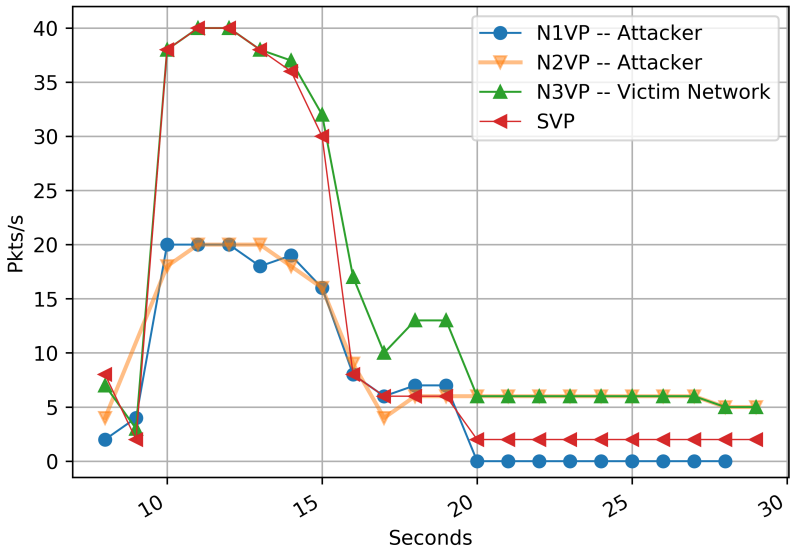

Figure 5.9: DDoS attack with different mitigation strategies by source RearBox

a mitigation policy that is aggressive at the beginning (block everything when the attack is detected) and rolled back once the source RearBoxes confirm that they have activated their mitigation procedures.

5.4.3 *Scenario 3: Scanning attack*

We consider an attacker in *Network* 3 that scans the IoT network by sending UDP packets of 10 bytes every two seconds. The scanner first scans *Network* 1 at time 9 s, then *Network* 2 at time 50 s. The background traffic consists of UDP packets sent from 10 nodes in *Network* 1 to 5 nodes in *Network* 2 with an inter-packet time of 5 s. The detection method raises an alert if it finds that a host contacts more than 5 IoT devices. The specific mitigation action on the target consists in blocking all flows from the scanning host and blacklisting it for 3 minutes. The results are shown in Figure 5.10. As expected, the background traffic is not affected since the mitigation specifically targets the scanning host.

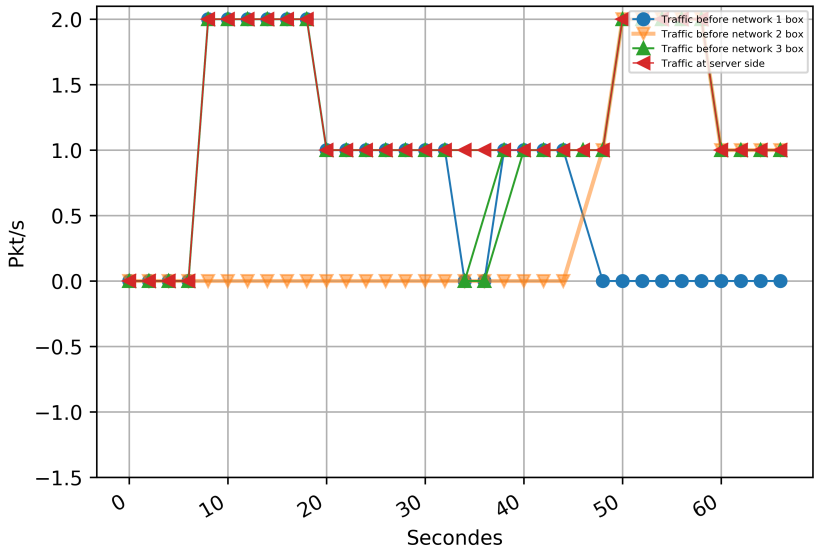

Figure 5.10: Scanning attack

In Figure 5.11, we show the result of a more complex detection and mitigation scheme. We have modified the RearBox software such that basic alerts (without flow details) are broadcasted to all RearBoxes with the help of the ZoneBox. In that way, the RearBox of *Network* 2 lowers its detection threshold from 5 to 2 nodes as soon at it hears from the first scan of *Network* 1. As a result, the scan is blocked faster when the attacker tries to scan *Network* 2 at time 48 s. Here, the collaboration between RearBoxes helps to lower the reaction time of the network to the attack.

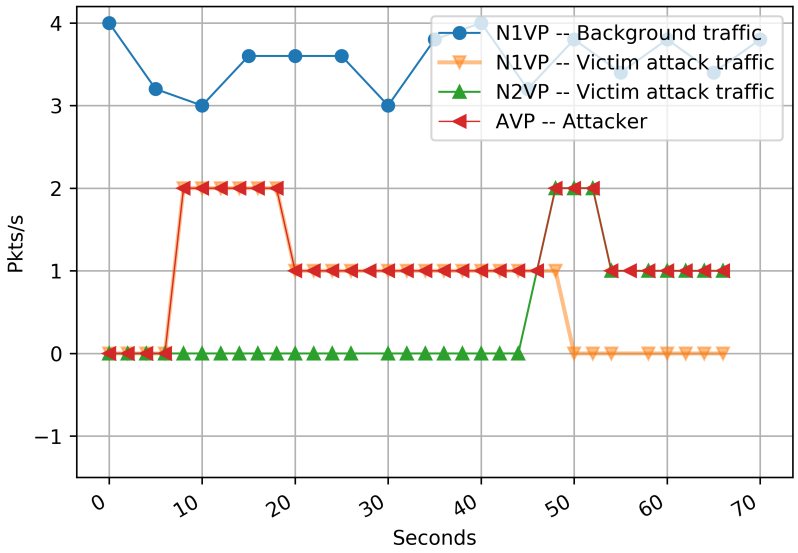

Figure 5.11: Scanning attack with adaptation of detection threshold after first scan

5.4.4 *Scenario 4: Reflection attack*

In our last experiment, a malicious user in *Network* 3 spoofs IP addresses of random nodes of *Network* 2 and tries to attack them with an amplified reflection attack by sending messages to the devices in *Network* 1 with an inter-packet time of 3 s. We also have background traffic from random nodes of *Network* 2 that send packets to nodes in *Network* 1.

Figure 5.12 shows that the reflection attack traffic doesn't pass the RearBox of *Network* 3 because the latter did not receive permission requests from the RearBox of *Network* 1 for that traffic. In fact, the attacker tried to request permissions on behalf of *Network* 1 but was rejected by the authentication procedure. The legitimate background traffic is not affected. Although an attacker could guess the IP addresses and port numbers of an existing permitted UDP flow and successfully pass the RearBox in this way, the other rules specified by the network administrator in the policies (permission duration, limitation of packet rate etc.) would still apply.

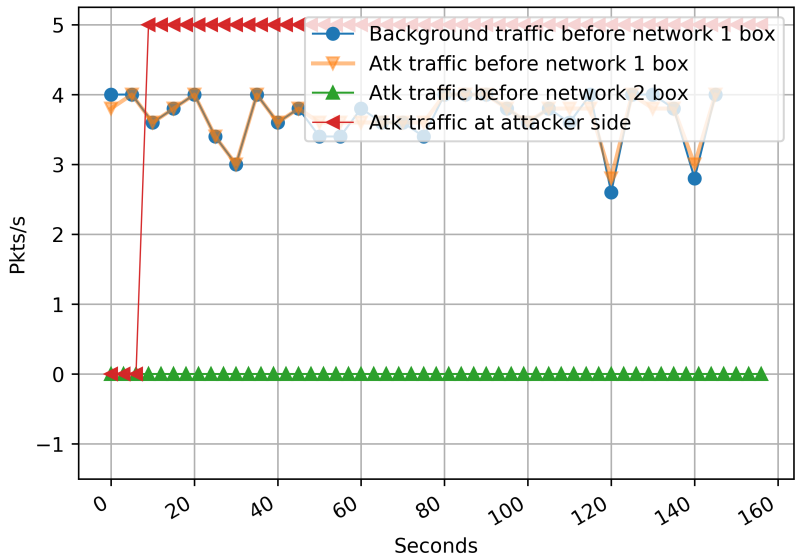

Figure 5.12: Reflection attack

SUMMARY

We have presented a distributed approach for the protection of collaborating IoT networks. Our system consists of middleboxes that are deployed on network borders and that enforce security policies. Communication between the networks is monitored and controlled by a permission-based policy system implemented on flow level. In addition, the middleboxes detect attack patterns and automatically execute mitigation actions. Attacks originating from collaborating networks are stopped at the attack side or near it thanks to the ability of the middleboxes to exchange information on detected threats and access policies. Default policies regulate the communication with unmanaged networks. We show in our experiments that different types of attacks can be completely mitigated with our approach, often without disruption of regular traffic. We also demonstrate how the distributed architecture can lower the reaction time to attacks.

A key characteristic of our approach is that it is purely network based and, therefore, does not require modifications on existing IoT end hosts nor their active participation in the protection scheme.

6

MIDDLEBOX COMMUNICATION SYSTEM FOR IOT PROTECTION

Man is condemned to be free; because once thrown into the world, he is responsible for everything he does.

—Jean-Paul Sartre

INTRODUCTION

The mitigation system presented in Chapter 5 uses middleboxes especially RearBoxes or LowBoxes, to monitor communication and to accurately stop the attack traffic at the attackers side. In this way, traffic flows detected as malicious by the destination do not leave their local source networks and do not put the destination system under stress. Our system is composed of multiple categories of middleboxes that exchange information on normal communication attempts (in form of permission requests) and on alerts. The focus of the boxes is on enforcing the communication rules for "their" devices and on protecting them from other networks. The rules can be specific for each device or group of devices and defined by their owners based on their usual communication behaviour. These rules are also used to decide which of the nodes is involved in malicious activity.

For this system to act fast and effectively, we need an efficient communication model. In this chapter, we describe the design of this communication model and the different message types needed to cover the different functionalities of the system. We focus on the message protocols and their impact on the quality of service experienced by the IoT devices.

Our approach does not only protect IoT devices but also systems that can become victims of attacks originating from them, i.e servers and other online services. The mitigation system only targets malicious flows, i.e. any communication the destination does not want at a specific point due to its policies or being overloaded. Our solution can be integrated directly in an existing network without modifications on the end hosts. As already explained in the previous chapter, this has the advantage that we can also protect devices that cannot be upgraded or that don't have the necessary resources to implement a protection system. Finally, a lookup system is implemented which allows our middleboxes to find each other and to cooperate, for example by broadcasting alerts in case of large-scale attacks.

We identify some weaknesses of our system and propose solutions for better protection. Those solutions only add a small overhead to the global system but can become an issue with time-constrained applications. It can be solved however using the appropriate policy. All those features will be presented in the following. The remainder of this chapter is organized as follows: In Section 6.1, we present the communication system, the different message types, and their purposes. Issues

of the initial system design are discussed and solved in Section 6.2. We present the experimental methodology in Section 6.3 and discuss results in Section 6.4. We conclude the chapter in Section 6.4.2.

6.1 MIDDLEBOX COMMUNICATION

In this section, we describe our communication protocol for middlebox lookups, alert propagation, synchronization, complaint when compromise middleboxes are found, activation of the RearBox, permission requests and replies, policy management, and flow information exchange. Some of these processes were already briefly mentioned in Section 5.2 without providing details.

All the middleboxes in the system need to authenticate themselves before any communications to avoid spoofing. The communications need to be encrypted and authenticated in order to avoid, amongst others, Man In The Middle (MITM) attacks. We decide to use TCP on port 9955 coupled with Transport Layer Security (TLS). The authentication will be done by middleboxes at both sides of the communication using their certificate. We will need a Public Key Infrastructure (PKI) to manage the keys, signatures and certificates [Mau96, Hög+20]. The management of the PKI is out of the scope of this thesis.

We build a hierarchical system to improve the scalability of the system so that an increase of end-devices does not affect the upper boxes' work. In our system, end-devices only interact with their RearBox and the latter's performance is only affect by their number. The ZoneBoxes of our system only interact with the RearBoxes for the first lookup and when reporting specific attacks that need collaborative actions, e.g. scans. The TopBoxes are used to register new RearBoxes inside the system and for large scale lookup as well.

In the following, we first describe the general message format in Section 6.1.1 and then explain how the different message types are used in Section 6.1.2.

6.1.1 *Message format*

As explained in the previous chapter, the RearBoxes request permissions from every communication's destination on behalf of the devices they protect. They enforce the devices' traffic policies and protect them against attacks coming from outside by monitoring the incoming traffic.

To ensure the functioning of the system, the RearBoxes need to exchange permissions, alerts and mitigation messages. We have therefore set up a communication protocol between the various RearBoxes of the system. The Figure 6.1 shows the message format.

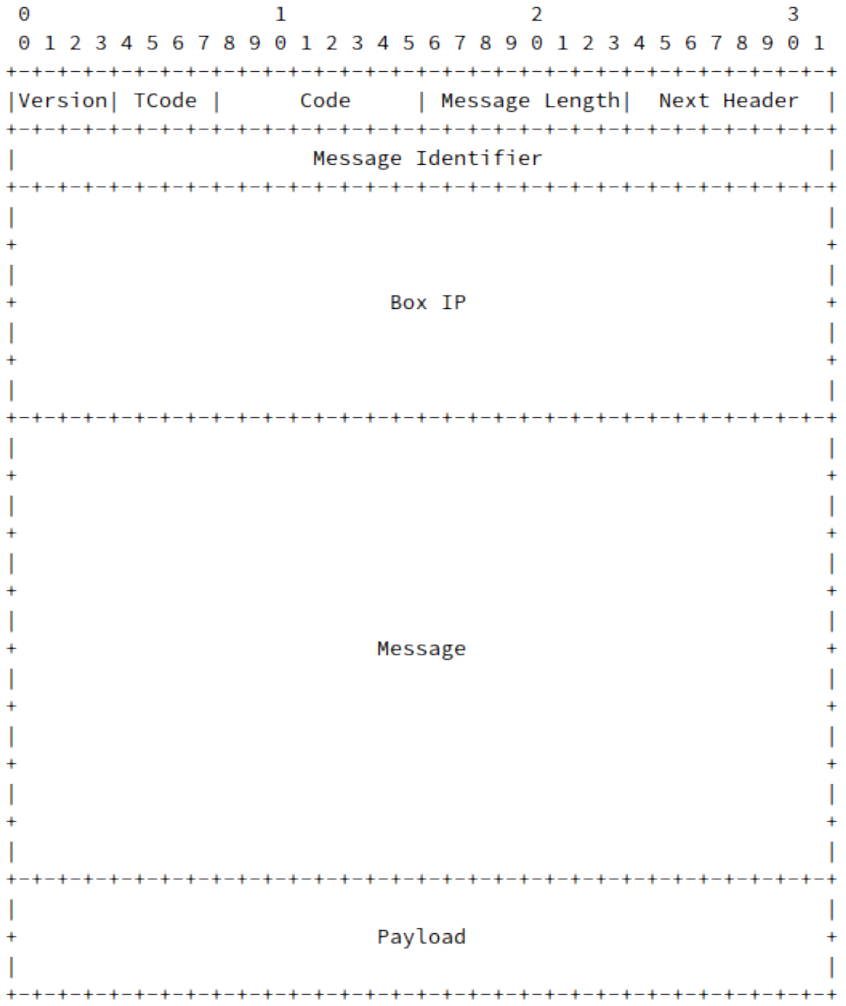

Figure 6.1: Message format

The fields are:

- A *Version* field (4 bits);
- The *TCode* field (4 bits) indicates the general messages type. We currently have 8 different types of messages: permission, alert, policy, flow, lookup, activation, synchronization and complaint.

- The *Code* field (8 bits) indicates the specific message sub-type for *TCode*. For example, for messages of TCode=permission, *Code* specifies whether the message is a request or reply;
- The *Message length* (8 bits) identifies the end of the header since the header size is variable;
- The *Next Header* field (8 bits) is designed for future integration at the routing level. It is used to link to other IPv6 headers. We plan to use our protocol coupled with routing information to mitigate traffic inside ISPs;
- The *Message Identifier* (32 bits) is an integer used to identify the messages and responses belonging to the same communication since two boxes can have several ongoing communications at the same time. This number is also used to verify that the communication protocol is not tampered but also for retransmission when a packet is lost. The field is coupled with the *TCode*, *Code*, and destination address and port to uniquely identify a communication flow.
- The *Box IP* (128 bits) field is the IPv6 address of the box that originally created the message;
- The *Message* (variable) contains the message details.
- The *payload* (variable) field is used to encapsulate other packets when needed. We will show an example in the next section.

The boxes in our system each have specific roles [MSE19]. A ZoneBox manages the RearBoxes of the same country. The RearBoxes protect a network's (e.g. a company's) servers and IoT devices. The TopBox is used for all exchanges with a global reach. The RearBox and the LowBox protect the network. Since the TopBox is the point of failure of our system, we can have several of them distributed over the Internet, similar to DNS root servers. Their addresses are stored in all other boxes. They are contacted through anycast and when a new one is added, the RearBoxes have to be updated with the new addresses.

Since we expect the number of IoT devices to grow exponentially in the near future and with them the number of RearBoxes, we have minimized the interactions between RearBoxes and TopBoxes to avoid overloading our system. The different interactions are limited to boxes

of the same level and their direct higher-level box only (with an exception being the activation phase of the RearBoxes). The different management messages used by our system use a priority system to fight against flood request attacks. The priority of the messages is as follows:

1. synchronization;
2. alert;
3. complaint;
4. lookup;
5. activation;
6. permission;
7. policy;
8. flow.

6.1.2 *Inter-box communication*

Procedure for the registration of a middlebox

In total, we have the following message types for the registration process:

1. A *unique identity check* message is sent by the RearBox at boot time to determine origin, uniqueness, and owner information.
2. A *key initialization* message is used for the secured registration of a RearBox at a TopBox.
3. *Synchronization* messages are used between TopBoxes.
4. A *ZoneBox request* is sent by a RearBox to a TopBox with the identity of the largest network to protect. If several networks are protected by the RearBox, all networks need to be reported to the ZoneBox.
5. A *ZoneBox reply* is sent by the TopBox with the IP address of the ZoneBox managing the region to the RearBox.

6. A *registration request* is sent by a RearBox to a ZoneBox with the box IP address and the networks it manages.

7. The ZoneBox replies with a *registration reply*.

The messages 1, 2, 4, 5, 6 and 7 are sub-types (specified in *Code* field) of the activation message type. Message 3 is a sub-type of the synchronisation message type.

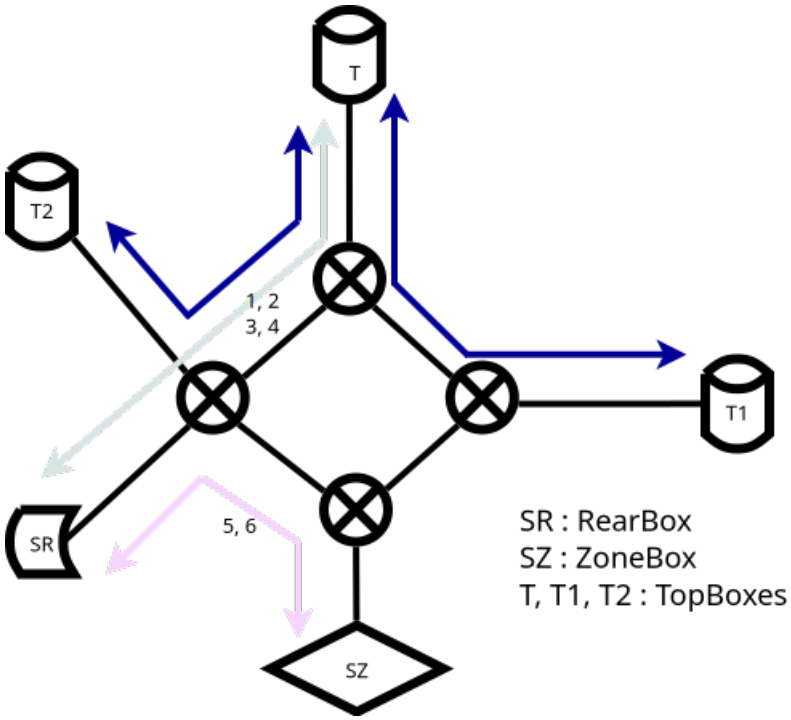

Figure 6.2: Message exchanges for registration

The first phase in the lifetime of a middlebox is its registration in the worldwide network of our system. This phase consists of initialization and registration of ZoneBox and RearBox information. ZoneBoxes need to provide their IP address, the IP addresses of the countries or regions they manage, their public key, and the identity of the manager. The registration of ZoneBoxes is a sensitive operation because it can compromise the security of an entire country, and, is therefore done manually with a TopBox. It will therefore not be presented in this document.

We show the registration of a RearBox *SR* in Figure 6.2. This procedure requires not only the cooperation of all types of boxes (RearBox,

ZoneBox and, TopBox) in the system, but also the participation of third parties. It is crucial for the correct functioning and security of our system that a RearBox is tied to the sub-network(s) it protects, i.e. it only sends permission requests and alerts for the address range(s) it is responsible for. Furthermore, collisions due to configuration errors where multiple RearBoxes claim to be protecting the same network should be avoided. This is more an administrative than technical issue and we will not discuss it further here since similar problems have been already solved before (domain name registration, IP address assignment, etc.). We imagine an approach where a RearBox is linked to its owner using a International Mobile Equipment Identity (IMEI)-type of system and Regional Internet Registry (RIR) or a similar organization will help verify that the box is only registered for the sub-network of its owner.

The RearBox chooses randomly the TopBox T that it will contact for its registration from the list stored in its firmware. As shown in Figure 6.2, RearBox SR starts the procedure by establishing a secure connection to T. As explained, both sides have to be authenticated, using the certificates provided by a PKI system. This is handled by the *unique identity check* and *key initialization* messages. Once this step is completed, T sends a *synchronization* message to inform the other TopBoxes (here: $T1$ and $T2$) about the new RearBox. The synchronization messages are only used by the TopBoxes to synchronize information related to new boxes and large scale attacks. SR then sends a *ZoneBox request* to T to obtain the address of the ZoneBox SZ that manages its country. T answers with a *ZoneBox reply* message. This closes the interaction between the TopBox T and the RearBox SR. The RearBox SR will finally contact its ZoneBox SZ to register itself with a *registration request* message. The ZoneBox SZ will verify that SR is genuine and that the network protected by SR does not overlap with other networks protected by other RearBoxes, then SZ answers with a *registration reply*.

A particular case of registration concerns the registration of a LowBox. A LowBox only has interaction with the RearBox. The LowBox initiates a *key initialization* with its Rearbox, followed by *registration request* and *registration reply* exchanges to determine the sub-networks protected by the LowBox.

Policy management

We have four types of messages used for policy management:

1. The *policy update* message is used by a RearBox or ZoneBox to send policies to a LowBox or RearBox, respectively. The message contains the parameters of the policy, such as maximum communication duration, port number, etc.

2. The *policy deactivation* message is sent by a RearBox to LowBoxes to permanently deny permission to a device and stop all its communication.

3. The *policy persistent* message is sent by a RearBox to LowBoxes to make a device's policy permanent. The communications of this device are the last stopped when a DDoS attack occurred.

4. The *default policy* message is sent by a RearBox to LowBoxes to remove existing policies for a device.

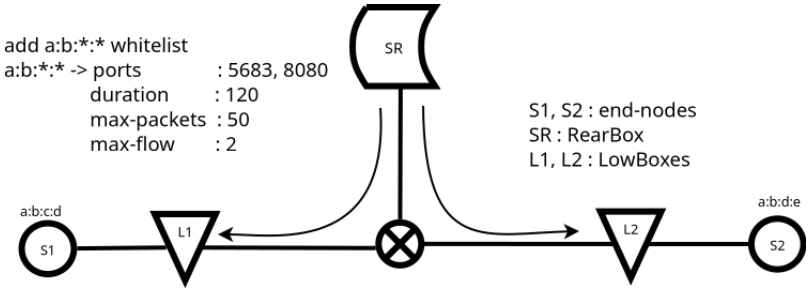

Figure 6.3: Message exchanges for policy management

One of the key functionalities of the system is the management of the policy lists on the LowBoxes and RearBoxes. Policies on the RearBoxes are set by their owners according to their needs. The RearBoxes configure the policies on the LowBoxes using *policy* messages. This message type allows to define policies for individual IoT devices (identified by their IP addresses) but also for groups of similar devices (using their OUI and the port number they are using) as shown in Figure 6.3. The OUI helps to specify policies for devices from the same manufacturer. This might not be sufficient in situations where users or administrators want to have higher-level classifications of devices, for example in order to specify policies based on device functionalities. In that case, we envisage the usage of management software that can be added on top of our system and that is able to generate individual device policies based on profiles assigned to the devices.

In Figure 6.3, we can see RearBox *SR* sending policies information i.e the open ports, the max-packets allow in a flow, the maximum number of flow.

Since we have many ways to define policies for a device (individual and group), we may have conflicting policies. For example, a network administrator might define a device policy allowing flows of long duration for a specific Huawei camera device and a group policy with lower allowed flow duration for all Huawei devices. We define that device policies have higher propriety than group policies for all devices, as shown in the algorithm of the policy lookup function in Algorithm 6.1.1. If no policy matches, the default policy is returned (line 12).

Algorithm 6.1.1: Policy lookup

Input: *policyList* : policy list, *packet*: packet, *direction*: incoming or outgoing packet

```
1  if direction is incoming then
2  |  id ← packet.destination;
3  else
4  |  id ← packet.source;
5  end
6  policy ← policyList[id];
7  if not policy then
8  |  id ← getDeviceOUI(packet);
9  |  policy ← policyList[id];
10 end
11 if not policy then
12 |  return default policy;
13 end
14 return policy ;
```

User-devices monitoring

Sometimes, a network administrator needs accurate traffic information for monitoring or analysis purposes. These information when take directly from a RearBox can be difficult to interpret because of its position behind the router since we can have the presence of many others middleboxes like NAT, Virtual Private Network (VPN), etc. altering the packet information. It is possible to retrieve the original flows from the network LowBoxes by directly asking them to send

their flows for analysis. We have two types of messages for monitoring purposes:

1. The *flow request* message is sent by a RearBox to its LowBoxes to get flow records. It contains a time interval and optionally, protocol and IP addresses.

2. The *flow reply* message contains the list of requested flow records.

Permission management

When an end-device starts a new communication, e.g. to send or query sensor data, a set of permission messages are exchanged to ensure that the destination agrees to the connection. In this process, we have two types of messages commonly used: *permission* and *lookup*. The permission message allows to exchange the set of thresholds to be agreed on for future communications between two end-devices. The lookup message is used when RearBoxes need to find each other for further communications.

The permission messages are only used by the RearBoxes:

1. The *permission request* is used to request access to a particular device or network. It contains its destination's address, and the requirements for the communication, e.g. the maximum duration and maximum size.

2. The *permission reply* contains the decision of the destination RearBox and the limitations (communication duration, size, etc.) granted by the destination.

On the other hand, the lookup message is used to query a ZoneBox (TopBox) about a RearBox (ZoneBox) address:

1. The *lookup request* message contains the IP address of a network or a device. The message is sent by a RearBox to its ZoneBox.

2. The *lookup reply* message is sent by the ZoneBox and contains a boolean specifying if the network is protected by our system and then the IP of the network's RearBox.

3. The *lookup delegate request* message is sent between ZoneBoxes or between ZoneBox and TopBox and contains the same information as the lookup request.

4. The *lookup delegate reply* message is a reply to the *lookup delegate* request with the requested information.

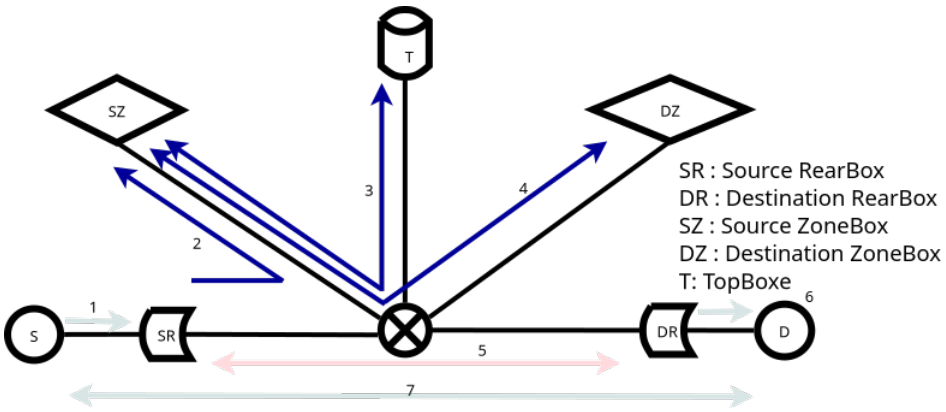

Figure 6.4: Message exchanges for permission management

As shown in Figure 6.4, when an end-device S wants to send a packet to end-device D, RearBox SR first checks if a permission for the communication exists. For a new communication, sending a *permission request* message is necessary, so SR checks whether it knows the identity of RearBox DR protecting end-device D.

If DR's IP address is not known to RearBox SR, SR sends a *lookup request* request to its ZoneBox SZ. If RearBox DR is under the authority of the same Zonebox SZ, a *lookup reply* is sent to RearBox SR with the address of DR. Otherwise, the ZoneBox SZ sends a *lookup delegate request* to T to get the address of DZ. Finally, the ZoneBox SZ contacts DZ to request the IP address of DR also using a *lookup delegate request* and *lookup delegate reply* exchange.

Now that the address of RearBox DR is known, a *permission request* is sent by RearBox SR to DR containing the original packet of S encapsulated inside as shown in Figure 6.5. If DR considers that the traffic can be accepted it transmits the original message from S to D and replies to SR with a packet containing a permission as well as D's response in the payload. We encapsulate the original packet inside the permission to not spend an extra exchange (1 extra RTT) when the permission is accepted. The permission is stored in SR and DR so that in the future the two final devices can communicate directly (labelled (7) in Figure 6.4). In case the permission is denied, DR responds directly with an empty payload and SR installs an entry in its permission list to block all packets from S to D for the time specified by DR in its answer.

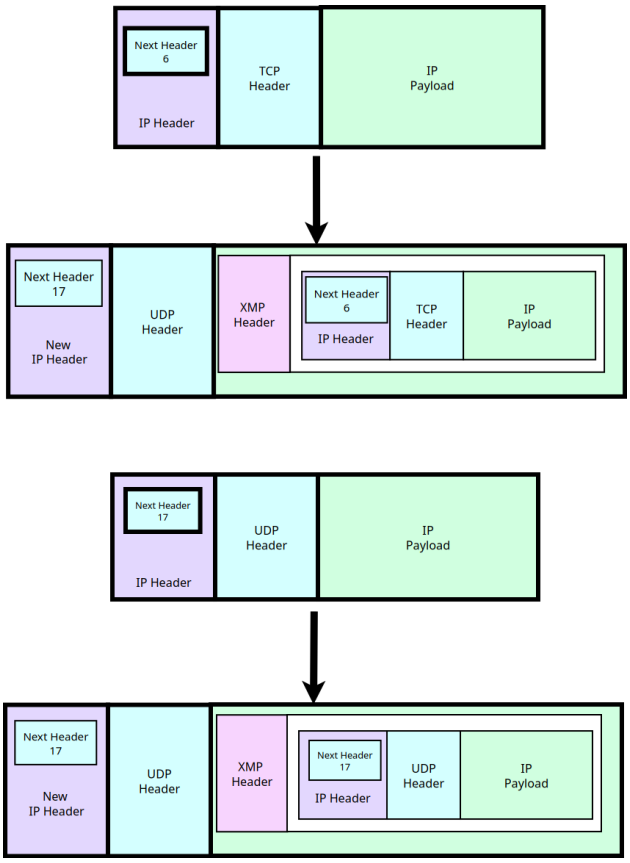

Figure 6.5: Protocol encapsulation

When the network is protect by a LowBox, it just transfers the message to its RearBox since the RearBox is the only one interacting with other RearBoxes.

Alert and mitigation

In the following, we extend the description given in Section 5.2.4 by focusing on the exchanged messages since we have already explained the basic approach. Since our system focuses on detecting attacks at the destination, the policies of the devices are used to distinguish the types of traffic. When an attack is detected, depending on its type, different types of messages are sent to one or more recipients. The alert and mitigation messages are:

1. The *alert notification* message is sent by a RearBox to its ZoneBox. The message contains the type of the detected attack (scanning, DoS, DDoS, or reflection), relevant information like the attack source and threshold to reduce the malicious communications.

2. The *alert acknowledgment* is used to ensure that the RearBoxes of the attacking devices are warned and will take mitigation measures. The mitigation being a sensitive part of our system, we need to be sure that alert message are received by the destination RearBox.

3. The *alert broadcast* is only used to send alert information to the TopBox for broadcasting.

When an attack is detected and an *alert notification* is sent to the attacker RearBox, a priority system is used to stop communications in the following order:

- Malformed traffic;
- Unusual traffic (Traffic not compatible with the usual behaviour of the device);
- Recent traffic (Traffic started a little while before receiving the request);
- Infrequent traffic (Traffic with a low frequency);
- Normal traffic.

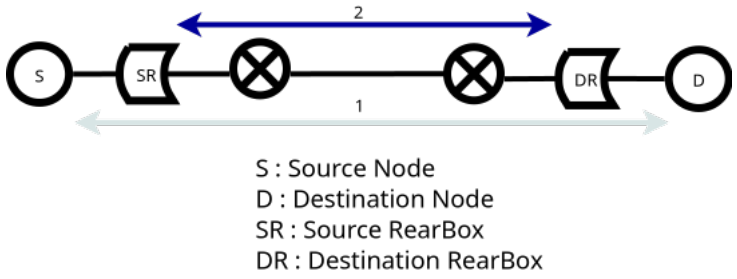

Figure 6.6: DoS alert scenario

As shown in Figure 6.6, a *DoS* attack launched by end-node *S* (labelled (1) in the figure) is detected by the RearBox *DR* protecting end-node *D*. An *alert notification* message is sent by *DR* to the RearBox

SR of end-node *S*. The RearBox *SR* of the attacker replies with an *alert acknowledgment* and stops the attack traffic. The alerts are labelled (2) in the figure.

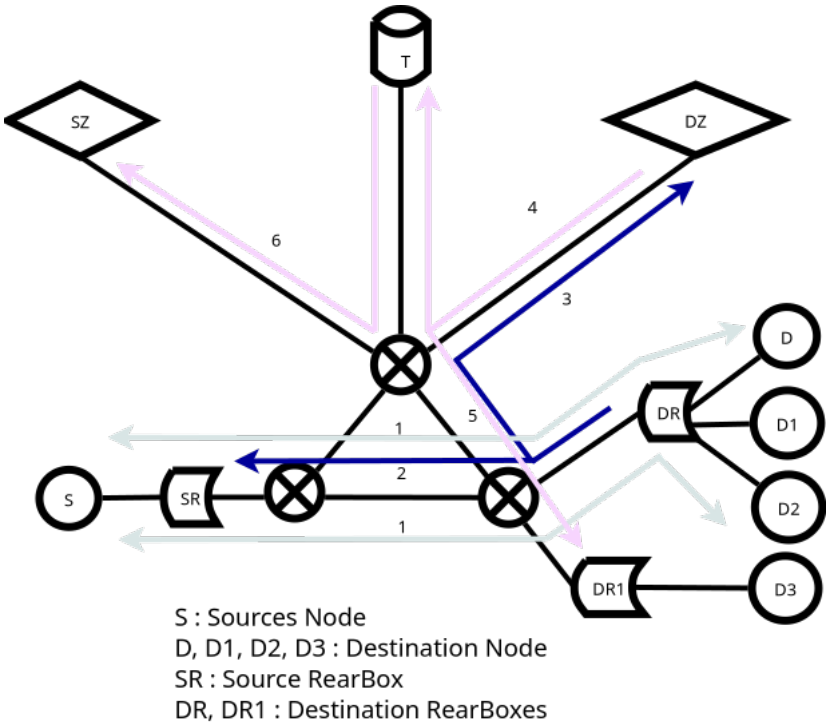

Figure 6.7: Scan alert scenario

When *DR* notices that the attack is a *scan*, i.e., *S* is trying to access several devices or unoccupied addresses in *DR*'s network, an *alert notification* is sent again by *DR* to *SR* which stops the outgoing traffic temporary (Figure 6.7). In addition, an *alert broadcast* is also simultaneously sent by *DR* to its ZoneBox *DZ*. *DZ* sends a *alert broadcast* to *T* which broadcasts it to all other ZoneBoxes, then each ZoneBox alerts their RearBoxes to reduce the threshold of their scan detection policy for the address of *S*.

In the case of a DDoS attack (Figure 6.8), i.e. several devices *S*, *S*1 and *S*2 attack *D*, *alert notification* messages are sent to the RearBoxes *SR*, *SR*1 and *SR*2 responsible for the malicious devices. The RearBoxes *S*, *S*1 and *S*2 reply to *DR* with an *alert acknowledgment*. The thresholds specified in the message are applied to communications by the RearBoxes. If after a period of 30 seconds, *DR* notices that *D* is still under attack the

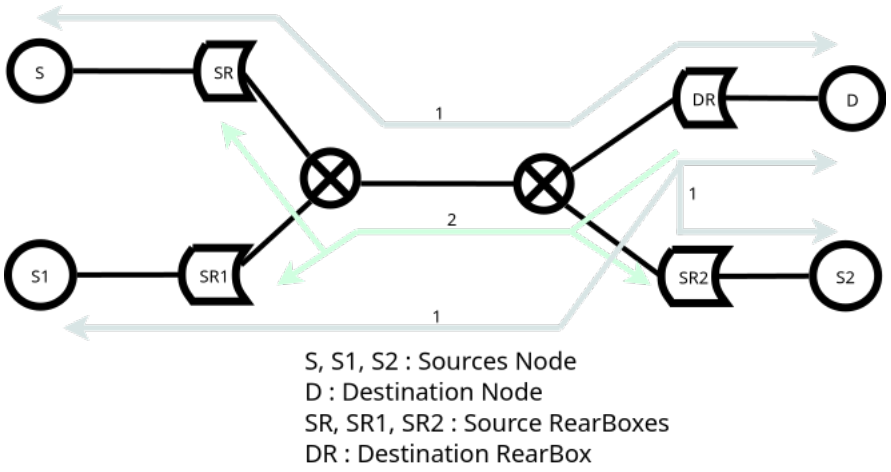

Figure 6.8: DDoS alert scenario

RearBox *DR* chooses the flows with the biggest impact on the end-user device and asks their respective RearBoxes to stop them.

Finally, misusing devices for amplification or reflection is not possible in our system because the permission system prevents the usage of spoofed IP addresses. In the case where end-device *S* has a RearBox, the spoofing is detected as soon as the packet is sent due to the source IP address not matching the network. If the network of the attacker does not has a RearBox but the host used for the reflection has a RearBox the attack packet will be stopped there.

In all previous scenarios, when no action is taken by the attacker's RearBox *SR* and the attack continues, a *complaint request* message is sent by the RearBox *DR* to the ZoneBox *DZ* to mitigate the attack. As explained in Section 5.2.3, the ZoneBox could block the attack traffic through the ISP, however, this is outside the scope of this thesis. This extreme measure ensures that the system can work even when some RearBoxes are saturated with traffic or for systems that do not have a RearBox but whose region is under the control of a ZoneBox. The ZoneBox *DZ* replies with a *complaint reply* sent to *DR* and ask the ZoneBox of *SR* to stop the attack. More details on complaint messages will be given in Section 6.2.2.

6.2 MISUSE OF THE MIDDLEBOX ARCHITECTURE

So far, we have only explained how the system protects other hosts. In this section, we are more interested in possible attacks against the system itself. We consider three scenarios:

1. A box makes a false claim of being under threat;
2. A box makes a false claim of doing the mitigation;
3. A box refuses to allow communication.

In the following, we will discuss these scenarios and then how to protect the system against them.

6.2.1 *Misuse scenarios*

False threat claim

In this scenario we have a compromised RearBox DR that falsely claims to be attacked by a device S. The RearBox SR of S can see that S has not sent malicious traffic and refuses to block S's traffic, but DR can send a complaint to DZ which takes mitigation actions. In this scenario, this mitigation is effectively equivalent to a DoS attack against S.

False mitigation claim

Here, we assume the attacker's S RearBox SR is compromised and after being warned of the malicious traffic sent by S, it does not mitigate it.

As a result, the RearBox DR will send a *complaint request* to its ZoneBox DZ. Similarly to the previous scenario, we have again the question whether a complaining RearBox can be trusted. It should be also noted that ZoneBox mitigation are costly in terms of time and resources because middleboxes must be contacted at the ISP backbone level, therefore we have to be sure that the complaint is justified.

Refusing all communication

In the last scenario, the RearBox DR is compromised and denies access to the devices under its supervision, making the services they provide unusable to others. This behaviour is difficult to identify as malicious because it could be also caused by a network problem or a fault host.

A possible way to handle this kind of situations is to report the non-responding devices to the ZoneBox of *DR*. If the number of reports reaches a certain threshold, the ZoneBox notifies the owner of *DR*.

6.2.2 *Protecting the system against abuse*

There are two general approaches to prevent false claims. First, we can try to secure the RearBox software such that its integrity can be verified by a ZoneBox when a complaint is received. Second, we can require the complaining RearBox to provide evidence (a signed packet showing an unauthorized communication is going on) of the ongoing attack against it before the ZoneBox performs mitigation actions.

Verifying software integrity

We can require that the RearBox software is executed in a Trusted Execution Environment, as provided for example by Software Guard Extensions (SGX) on Intel CPUs [CD16, Arn+16], that can be verified by the ZoneBox. This measure would protect against the three misuse scenarios described in the previous section. The main problem of this approach is the extra cost (functions overhead, and data deciphering) [Zha+16] that comes with SGX and that can turn the RearBox into a bottleneck.

Message signing

To prevent false threat and mitigation claims, we can require that a RearBox *SR* that sends a complaint to a ZoneBox *SZ* must send a sample of the offending traffic in order to prove that the complaint is justified. This can be implemented by signing all packets exchanged between end hosts by the RearBoxes. That means that all outgoing packets require the active participation of the sender's RearBox *SR*, even after the permission has been obtained from the destination Rearbox *DR* for the corresponding connection. The signature proves that the packet is really coming from the asserted source and has not been spoofed by the compromised RearBox. The ZoneBox *SZ* can then check the complaint content looking for the malicious source device and then determine that the proof is enough.

Control messages

Our final solution is to add a control message for traffic between RearBoxes. The RearBox that opens the communication will send a control message signed with its private key for each ongoing communication at a regular interval for the duration of the communication. Control messages must be acknowledged by the target RearBox, so that no RearBox can deny having received a control message.

Four types of messages exist to handle complaints:

1. The *complaint request* sent by a RearBox to its ZoneBox with the the IP address of the RearBox perpetrator and a proof of the wrongdoing.
2. The *complaint inquiry* sent to the RearBox accused of misuse to request a proof of innocence.
3. The *complaint reply* contains the acknowledgement of the ZoneBox to the RearBox complaining.
4. The *complaint mitigation* is a message sent to the ZoneBox of the malicious RearBox or a third party, i.e. an ISP or a another network manager organization to mitigate the attack.

The signed control message contains the timestamp and can be used as evidence for the ongoing communication. The clock synchronisation can be a problem if the clock difference on the middleboxes is more than 10 s (the minimum size of a permission), so an appropriate clock synchronization mechanism must be deployed. The control messages will be included in the complaint messages by the defender's RearBox and used to legitimize the complaint. The control message is a proof provided to the ZoneBox to identify the malicious ongoing communication. Furthermore, if a RearBox DR does not receive control messages from a RearBox SR, it can assume that SR is compromised and alert its ZoneBox DZ. On the other hand, a RearBox SR can also contact its ZoneBox SZ and report DR as malicious if not receiving acknowledgement for every control message sent.

In the false threat claim scenario, it becomes trivial to detect a false claim if the complainant cannot provide proof. In the false mitigation claim scenario, the ongoing mitigation can be proved with the control message as well. Using the algorithm shown in 6.2.1 we can find the compromised box.

Algorithm 6.2.1: Handling of complaint

Input: SR : Attacker box IP, DR : Victim box IP

```
proof_dr ← getProofFrom(DR);
if proof_dr is valid then
    proof_sr ← getProofFrom(SR);
    if proof_sr is valid then
        result ← DR is compromised;
        send message to DR manager;
    else
        result ← SR is compromised;
        send message to SR manager;
    end
else
    result ← DR is compromised;
    send message to DR manager;
end
```

6.3 EXPERIMENTAL METHODOLOGY

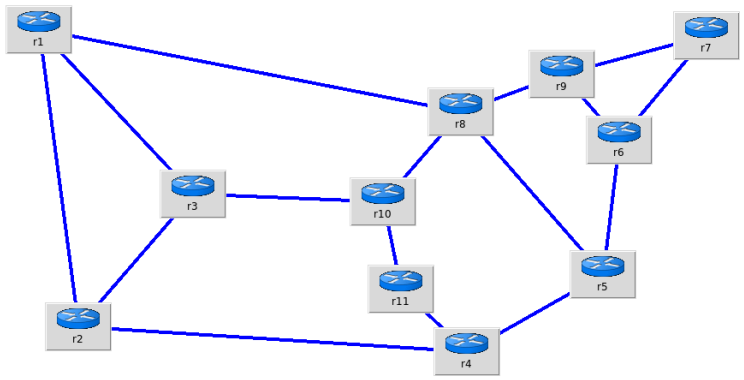

Figure 6.9: Network topology used in this chapter

We validate our approach on a network topology emulated with ipmininet (https://github.com/cnp3/ipmininet) instead of mininet since we use IPv6. The topology consists of 11 interconnected routers representing the Internet core (Figure 6.9). Ten sub-networks, each with their own RearBox and up to 20 resource-constrained IoT devices, are connected to that core.

- *Network* 0 represents the Internet core. It contains 11 routers interconnected as in Figure 6.9.
- *Network* 1 to *Network* 10 contain the 10 RearBoxes connected to $r1$, $r2$, $r4$, $r5$, $r8$ (2 per routers);
- *Network* 11 hosts one node representing a TopBox and one node for the ZoneBox connected to $r9$;

We use a packet loss rate of 2% and a bandwidth of 100 Mb/s with 5 ms delay on the link between the router and the RearBox. The TopBox and ZoneBox in *Network* 11 have more resources and a bandwidth of 250 Mb/s and a shorter delay of 2 ms on the link to $r9$. The IoT devices are not simulated individually in the following experiments. Instead, the RearBoxes simulate the activity of the devices in their sub-networks by exchanging permission requests and other types of messages with other RearBoxes. The IoT data traffic consists of messages sent over UDP and their respective replies.

We use the same threshold-based detection methods as in Section 5.2. In the following, background traffic represents normal interactions between the different networks. The focus of the experiments is to evaluate the scalability of the system and the overhead of the management messages.

6.4 VALIDATION

6.4.1 *General considerations on the system overhead*

In general, the response time of a middlebox to messages from other boxes is determined by:

$$T = T_{sec} + T_{smq} + T_{dmq} + T_{msgTCode} \tag{6.1}$$

- T_{sec} is the time needed to open a secure connection with the destination box;
- T_{smq} is the time needed to process the message at the sender;
- T_{dmq} is the time needed to process the message at the receiver;
- $T_{msgTCode}$ is the processing time for a message type.

In practice, we expect the response time to be dominated by T_{sec}, unless the boxes are under heavy load causing large queueing delays for outgoing and incoming messages. T_{sec} can be improved with the 0-RTT scheme of QUIC, while the queueing delays will depend on the computation resources of the boxes. In the case where all packets need to be signed by the RearBoxes as a proof of origin (see Section 6.2.2) the response time further increases by the time needed to sign the packets at the source box and to verify the signature at the destination box.

Furthermore, our permission-based approach adds an overhead to the communication between two end hosts because the first packet of the communication triggers a permission exchange. Instead of just sending the packet to the destination, the source RearBox has to encapsulate it into a permission request. At the destination RearBox, the destination device's policies and the existing entries in the permission list and the alert list have to be checked. A similar overhead occurs for the response packet of the destination device since it has to be encapsulated into a permission reply message before being forwarded to the source RearBox.

6.4.2 *Message type impact*

Normal traffic scenario

We first test our system with attack-free traffic. The goal here is to explore the scalability of the system under normal traffic. We have a scenario with IoT devices of Network 1 under $r1$ sending UDP packets of 200 bytes with 2 seconds of interval to IoT devices of Network 3 under $r3$. The default policy is to grand permissions of 10 s duration. Only Network 1 does not receive any traffic. Network 10 does not send any traffic and only receives it. We have a total of 200 IoT nodes present in this experiments separated in two scenarios.

In the first experiment, we have 10 RearBoxes with 20 devices each. Figure 6.10 shows at the beginning of the experiment a lot of traffic for the ZoneBox from the RearBoxes because of the activation of the 10 RearBoxes and the ZoneBox itself. The activation in this experiment consists of a registration request and the corresponding registration reply. We can see the 11 messages at the beginning, followed by 9 lookup message exchanges. Note that since the nodes of Network 10 are not sending any messages to other devices, their RearBox does not need to perform a lookup. No more communications are done by the

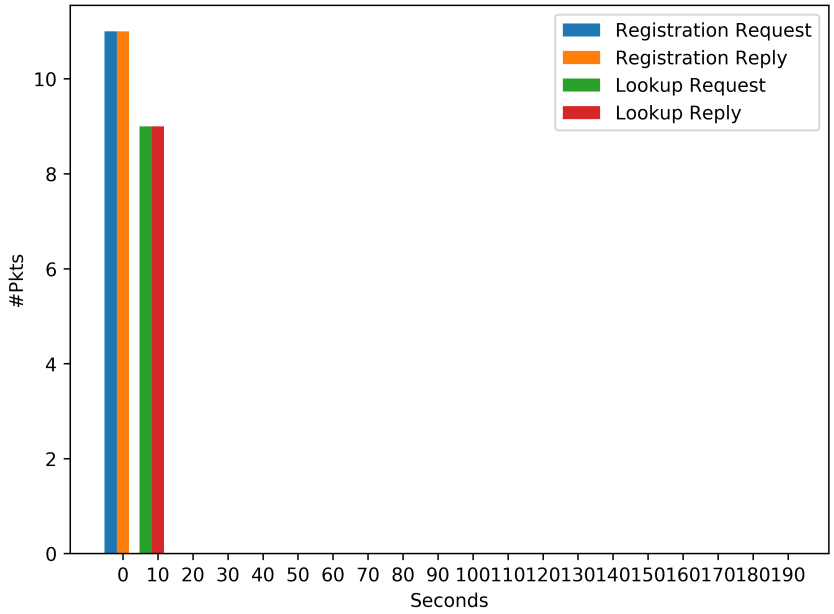

Figure 6.10: ZoneBox messages with 200 nodes and 10 RearBoxes

RearBoxes with the ZoneBox after that. Once the desired information has been obtained and cached by the RearBoxes, no further contact is needed with the ZoneBox.

Figure 6.11 shows that the traffic of the RearBoxes is constant throughout the rest of the experiment after the initial activation and lookup actions. We can see that the exchanges between the RearBoxes consist of requests and replies for permissions at regular intervals. We also notice that RearBox 1 protecting Network 1 sends only half of the number of packets of RearBox 2 protecting Network 2, because the former only receives permission requests and sends permission replies for the traffic it sends to RearBox 2. RearBox 2, on the other hand, manages the requests and replies for incoming messages from RearBox 1 and outgoing messages to RearBox 3. The traffic for all the RearBoxes except for RearBoxes 1 and 10 is similar to the traffic of RearBox 2.

Figure 6.12 shows the resulting traffic with 10 RearBoxes with 20 devices each when the permission duration is doubled. With this longer duration, the number of permission messages is reduced by a factor of two. The permission duration is therefore a parameter that plays an important role in the scalability of the system. However, a longer permission duration also increases the risk for a successful attack.

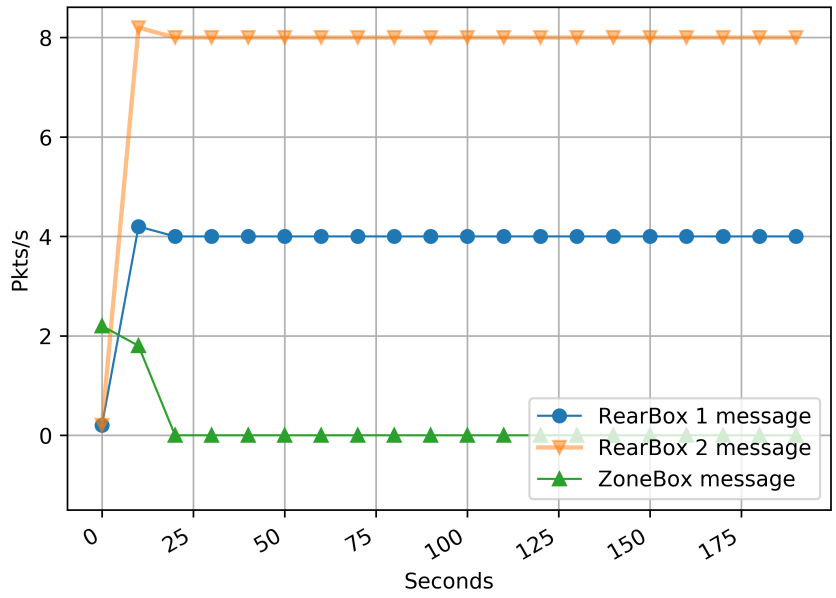

Figure 6.11: Normal communication with 200 nodes and 10 RearBoxes

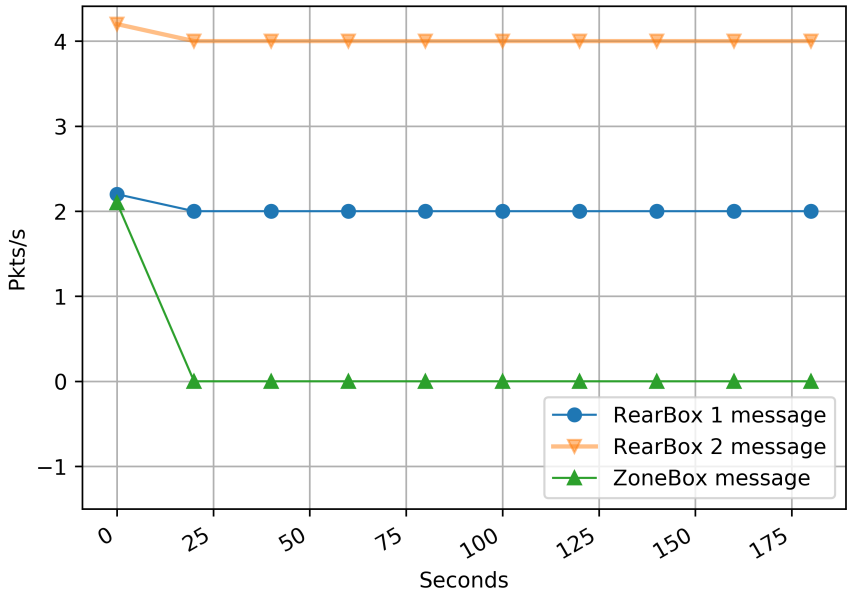

Figure 6.12: Normal communication with 200 nodes and 10 RearBoxes with 20 s permission duration

Although the traffic is still monitored and network attacks will be still detected, the longer duration gives a MITM a bigger attack window. Therefore, the permission duration should be tailored to the devices and services to be protected.

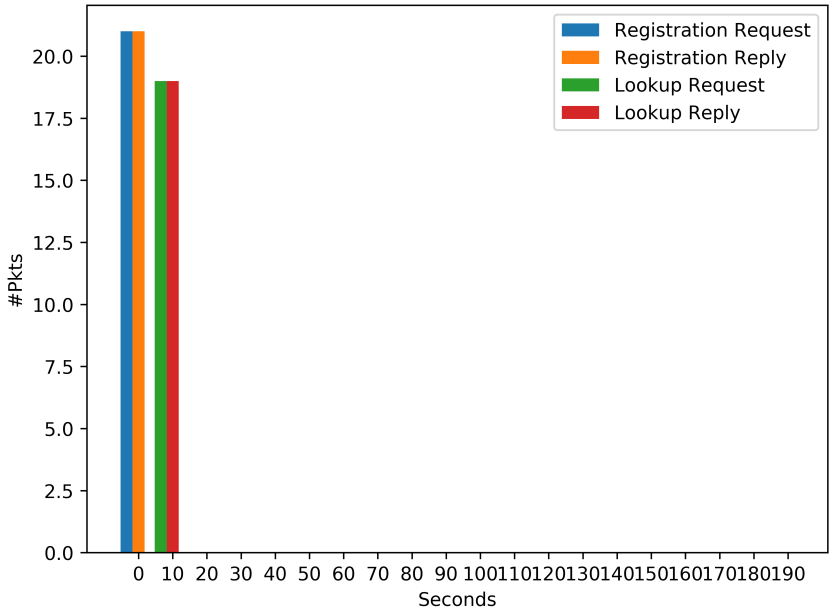

Figure 6.13: Normal communication with 200 nodes with 20 RearBoxes

The same experiment is repeated with 20 RearBoxes with 10 devices each, i.e. the number of end devices remains 200, however the number of RearBoxes is doubled. We can see in Figure 6.13 that the exchanges with the ZoneBox double as well: From 11 registrations to 21 and from 9 lookups to 19. We can conclude that the number of RearBoxes has a direct impact on the load on the ZoneBox.

When comparing the number of RearBox messages exchanged in the two setups (Figure 6.11 and Figure 6.14), we can see that they look identical, apart from the number of messages exchanged with the ZoneBox. In fact, they depend on the interaction of the IoT devices protected by the RearBoxes. With Network 1 only sending traffic and not receiving, the number of permission exchanges is half the amount of the other network.

To summarize, under normal conditions, the number of messages a ZoneBox receives mostly depends on the number of RearBoxes and not on the number of devices managed by a RearBox. Communication

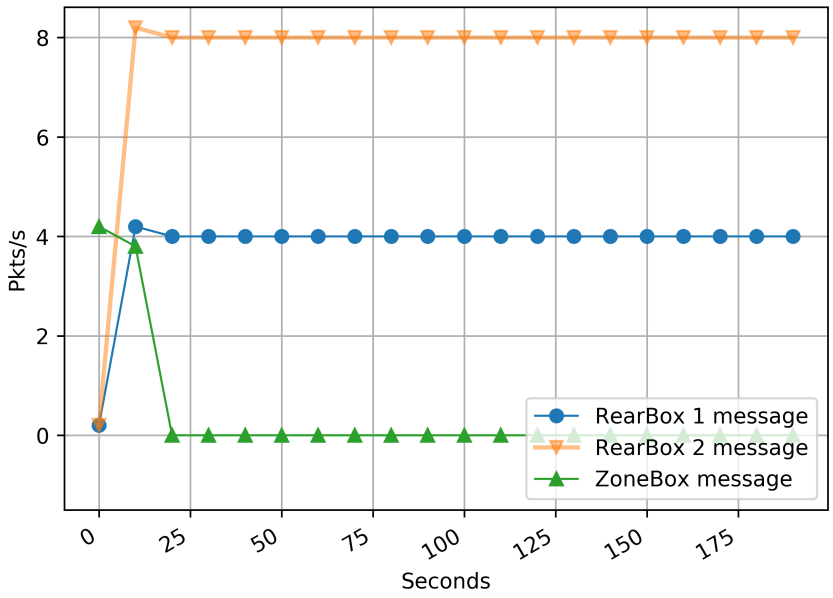

Figure 6.14: Normal communication with 200 nodes with 20 RearBoxes

attempts by end devices can result in lookup messages sent to the ZoneBox but the requested information is cached at the RearBoxes. If the communication patterns are relatively stable, i.e. end devices do not frequently try to communicate with new targets, the number of lookup messages will be small. This allows keeping the higher layers of the middlebox architecture, i.e. the TopBoxes and ZoneBoxes, relatively small compared to the number of RearBoxes and end devices and, therefore, ensure the scalability of the design.

Attack traffic scenarios

Figure 6.15 shows the impact of a scan attack. The traffic of the end devices are again UDP messages with a size of 200 bytes sent every second to their destination. The initial detection policy is to trigger an alert when a device contacts at least 5 nodes under the same RearBox. The attack is executed by one host of Network 1 trying to scan all the devices in the other networks. In the figure, we see at the beginning the registration process of the middleboxes. We have 22 packets exchanged with 20 for the RearBoxes and 2 for the ZoneBox itself.

The effect of the attack appears at 10 seconds where we see the ZoneBox traffic reaching 3.8 pkts/s instead of the normal 2 pkts/s for

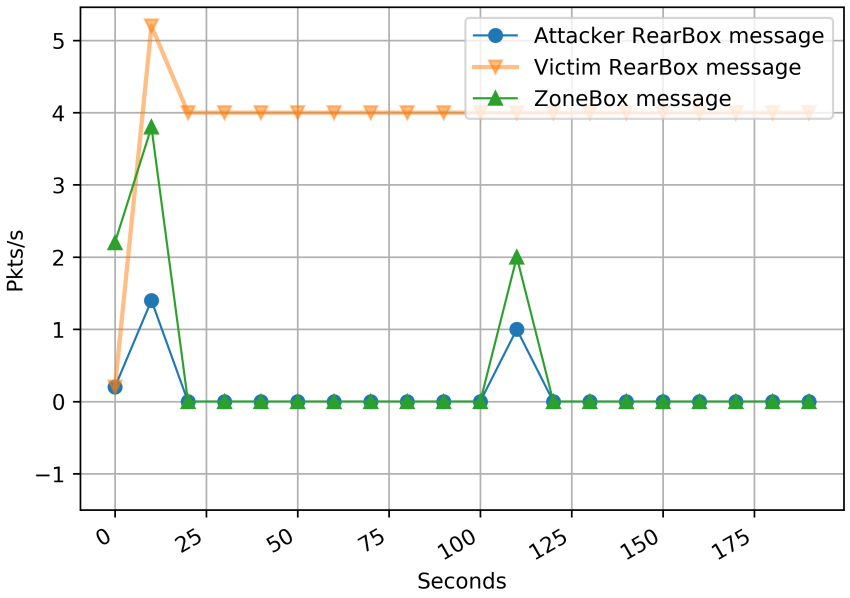

Figure 6.15: Scan attack scenario

the normal communication of 10 RearBoxes. The additional messages come from the *alert broadcast* sent to the others 9 RearBoxes under the ZoneBox. The attacker RearBox *SR* exchanges 1.4 pkts/s corresponding to the lookup, the 5 permissions and the alert communication. The victim RearBox *DR* reaches 5.2 pkts/s of which 1.2 represent the permissions and the raise alert. After the alert is received by the attacker's RearBox *SR*, its traffic is mitigated for 80 seconds.

After the mitigation period is over, the attacker can again send scanning packets to another network. However, this time, the attack traffic can only reach 1 pkt/s before the mitigation starts because the previously sent alarm has lowered the detection threshold of the target. The traffic to the ZoneBox consists of lookup messages and broadcasted alarm messages triggered by the scan. In general, a scan attack will trigger as many alert messages as RearBoxes. By increasing the mitigation duration, we can reduce the number of messages. Furthermore, since the threshold for the attack detection is halved every time the scan is detected, at some point the attacker's RearBox will completely block the scanner and no further alerts are broadcasted.

In the last scenario, we study the impact of a DDoS attack (Figure 6.16). The device policy in this experiment is to allow less than 10 communications to a node and stop unwanted traffic for 200 seconds.

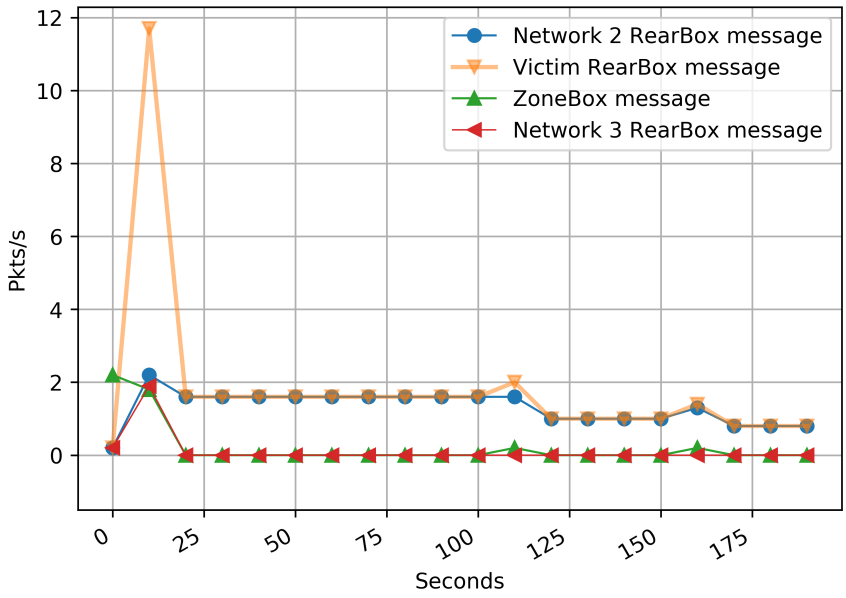

Figure 6.16: DDoS attack scenario

Here, network 1 is attacked by all the other 9 networks in the experiment. The attack traffic is composed of UDP messages with a size of 500 bytes send every 2 s as the normal traffic. When, at 10 s, the 90 devices send their traffic, less than 10 communications are allowed by the policies and the attack is mitigated before it can flood the victim.In the figure, the attack is detected at the 110th second and messages are sent to all attackers' RearBoxes to throttle their traffic. Every time the detection threshold is reached again, new alert messages are sent to all the RearBoxes with nodes involved as seen at second 160.

SUMMARY

In this chapter, we have presented the communication system of our distributed middleboxes solution that allows to stop attacks from IoT devices or against them at the source of the attack. We presented the different management messages needed to have a fully functional and scalable system. The system is composed of different types of middleboxes i.e. RearBox, ZoneBox and TopBox. The TopBox governs all global communication, the ZoneBox is the manager of all the RearBoxes

in a country. The RearBox is the one that ensures the protection and mitigation of the different attacks.

We have created a simple process to automate the integration of new RearBoxes into the global system. For security reasons, middleboxes are confined to their geographical area and interactions are limited to other middleboxes of the same hierarchical level or their direct hierarchical box. This ensures the robustness and scalability of the system.

RearBoxes use a permission system based on the resources of the nodes they protect. This ensures that despite their different resources, all IoT devices are well protected. An alert system is used to warn RearBoxes whose nodes are committing malicious actions and to stop their traffic. Communications between middleboxes are signed to resist man-in-the-middle attacks and we have also provided processes to identify them when compromised. In our experience, we have shown the scalability of our system by proving that only the traffic of a RearBox is dependent on the number of devices it protects and the number of communications these devices are allowed to initiate. Only Scan attacks require temporary the active participation of the ZoneBox to prevent all RearBoxes in the system and quickly terminate a scanner communication.

CONCLUSION AND PERSPECTIVES

CONCLUSION AND PERSPECTIVES

CONCLUSION

The rise of the Internet of Things has resulted in an exponential increase in the number of Internet-connected devices. These devices are often limited in resources because their tasks are often trivial and/or require cheap and maintenance-free embedded systems with low energy consumption. Unfortunately, their permanent connection to the Internet makes them prime targets for attacks. Several DDoS attacks have already been carried out using IoT devices and we expect their frequency to increase over time.

With this in mind, in Chapter 2, we used honeypots and a network telescope to make an inventory of the different services that are the most targeted on the Internet. Unsurprisingly, that inventory is headed by several services from the classic Internet, such as SSH and HTTP. The most targeted service is telnet because it is active on many IoT and home automation devices with default settings. We saw several versions of the Mirai malware attacking such devices. However, several IoT-specific services such as MQTT, CoAP and UPnP also appeared in our datasets.

One of the characteristics of the Internet of Things is the great diversity of devices and network technologies. In Chapter 3, we argued that traditional approaches to efficiently scan networks do not work well for networks containing resource-constrained IoT devices because their slow low-power networks are easily overloaded by the scanning process. We proposed an approach that differentiates nodes by the used network technology and adapts its scan strategy accordingly, resulting in a higher discovery rate.

Hacked IoT devices pose a danger to the Internet because they can be used as bots to launch powerful DDoS attacks against other Internet participants. In Chapter 4, we studied existing approaches to mitigate such attacks for general Internet services. Based on our observations, we proposed an architecture for the protection of IoT systems in Chapter 5. It is based on middleboxes that enforce communication policies defined by their owners. These middleboxes can exchange information on attacks, collaborate in mitigation actions and are therefore able to stop attack traffic at its source. One advantage of our approach is that it does not require the participation of the end hosts in the detection and mitigation mechanisms. In Chapter 6, we finally described in detail the hierarchical organization of the middleboxes and their communication system that allows them to manage and exchange policies, access permissions, and alerts and to detect if the system has been corrupted.

PERSPECTIVES

As future work, we envisage the following topics:

In our work on telescope and honeypot measurements in Chapter 2, we encountered various challenges especially in the search for services used by IoT devices. Existing interactive honeypots are still focused on traditional protocols such as telnet and neglect new protocols used by IoT systems. The CoAP honeypot developed by Master student Dany Yarakou[1] is an important step into the right direction. However, support for proprietary protocols as employed by several devices intended for home and industrial automation requires the cooperation with the industry.

In Chapter 3, we showed the usefulness of our scanning approach for heterogeneous systems consisting of IoT devices using IEEE 802.11 and IEEE 802.15.4 links. Network operators have recently started to deploy new network technologies based on LTE and the popularity of low-power long-range networks like LoRa is increasing. Their impact on the security of IoT systems and the Internet in general has to be studied carefully.

Finally, our distributed architecture for IoT protection presented in Chapter 5 and 6 has been only evaluated in the form of a proof of concept in an emulated environment. We see many opportunities to optimize and extend the system. For example, the regularity of typical

1 https://github.com/DanMistyk/HoneyPy

IoT traffic can be exploited by the middleboxes to anticipate permission requests and therefore reduce the latency of the permission mechanism. Another point is the message exchange between the middleboxes. In our original design, we have envisaged the usage of TLS resp. DTLS to secure the inter-box communication. The 0-RTT mechanism of QUIC could significantly reduce the communication cost.

BIBLIOGRAPHY

[Abd+03] S. Abdelsayed, D. Glimsholt, C. Leckie, S. Ryan, and S. Shami. "An efficient filter for denial-of-service bandwidth attacks." In: *GLOBECOM'03. IEEE Global Telecommunications Conference (IEEE Cat. No. 03CH37489)*. Vol. 3. IEEE. 2003, pp. 1353–1357.

[Ade+17] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. "Understanding the limits of LoRaWAN." In: *IEEE Communications magazine* 55.9 (2017), pp. 34–40.

[Adr+14] D. Adrian, Z. Durumeric, G. Singh, and J. A. Halderman. "Zippier ZMap: internet-wide scanning at 10 Gbps." In: *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. 2014.

[AG19] A. Aksoy and M. H. Gunes. "Automated iot device identification using network traffic." In: *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. IEEE. 2019, pp. 1–7.

[Ant+17] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. "Understanding the mirai botnet." In: *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 2017, pp. 1093–1110.

[AC09] K. Argyraki and D. R. Cheriton. "Scalable network-layer defense against internet bandwidth-flooding attacks." In: *IEEE/ACM Transactions on Networking (ToN)* 17.4 (2009), pp. 1284–1297.

[Arn+16] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. L. Stillwell, et al. "{SCONE}: Secure Linux Containers with Intel {SGX}." In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 689–703.

[AML17] H. A. Arouna, L. Metongnon, and M. Lobelle. "Reputation Rating Algorithm for BGP Links." In: *International Conference on e-Infrastructure and e-Services for Developing Countries*. Springer. 2017, pp. 352–357.

[Ash] K. Ashton. *That Intenet of Things Thing*. https://www.rfidjournal.com/articles/view?4986. Accessed: 2019-12-16.

[ANL00] T. Aura, P. Nikander, and J. Leiwo. "DOS-resistant authentication with client puzzles." In: *International workshop on security protocols*. Springer. 2000, pp. 170–177.

[Bac+13] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt. "RIOT OS: Towards an OS for the Internet of Things." In: *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2013, pp. 79–80.

[BSE18] P. Bajpai, A. K. Sood, and R. J. Enbody. "The art of mapping IoT devices in networks." In: *Network Security* 2018.4 (2018), pp. 8–15.

[BKL16] D. Bankov, E. Khorov, and A. Lyakhov. "On the limits of LoRaWAN channel access." In: *2016 International Conference on Engineering and Telecommunication (EnT)*. IEEE. 2016, pp. 10–14.

[BG14] A. Banks and R. Gupta. "MQTT Version 3.1. 1." In: *OASIS standard* 29 (2014).

[BI08] R. Barnett and B. Irwin. "Towards a taxonomy of network scanning techniques." In: *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries: riding the wave of technology*. ACM. 2008, pp. 1–7.

[Bar+07] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu. "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards." In: *Computer communications* 30.7 (2007), pp. 1655–1695.

[BI17] E. Bertino and N. Islam. "Botnets and internet of things security." In: *Computer* 50.2 (2017), pp. 76–79.

[Bev04] R. Beverly. "A robust classifier for passive TCP/IP fingerprinting." In: *International Workshop on Passive and Active Network Measurement*. Springer. 2004, pp. 158–167.

[BO+13] S. M. Bilalb, M. Othmana, et al. "A Performance Comparison of Network Simulators for Wireless Networks." In: *arXiv preprint arXiv:1307.4129* (2013).

[Boc04] V. Bocan. "Threshold puzzles: The evolution of DoS-resistant authentication." In: *Periodica Politechnica, Transactions on Automatic Control and Computer Science* 49 (2004), p. 63.

[BFC05] V. Bocan and M. Fagadar-Cosma. "Adaptive threshold puzzles." In: *EUROCON 2005-The International Conference on" Computer as a Tool"*. Vol. 1. IEEE. 2005, pp. 644–647.

[Bra+98] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson. "Detecting disruptive routers: A distributed network monitoring approach." In: *IEEE network* 12.5 (1998), pp. 50–60.

[Bru02] J. Brustoloni. "Protecting electronic commerce from distributed denial-of-service attacks." In: *Proceedings of the 11th international conference on World Wide Web*. ACM. 2002, pp. 553–561.

[BC00] H. Burch and B. Cheswick. "Tracing Anonymous Packets to Their Approximate Source." In: *LISA*. 2000, pp. 319–327.

[CS+17] J. de Carvalho Silva, J. J. Rodrigues, A. M. Alberti, P. Solic, and A. L. Aquino. "LoRaWAN—A low power WAN protocol for Internet of Things: A review and opportunities." In: *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*. IEEE. 2017, pp. 1–6.

[Cha+19] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki. "Network intrusion detection for IoT security based on learning techniques." In: *IEEE Communications Surveys & Tutorials* 21.3 (2019), pp. 2671–2701.

[Che+17] M. Chen, Y. Miao, Y. Hao, and K. Hwang. "Narrow band internet of things." In: *IEEE access* 5 (2017), pp. 20557–20577.

[CP05] R. Chen and J.-M. Park. "Attack Diagnosis: Throttling distributed denial-of-service attacks close to the attack sources." In: *Proceedings. 14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005.* IEEE. 2005, pp. 275–280.

[Che+16] Y. Chen, X. Chen, H. Tian, T. Wang, and Y. Cai. "A blind detection method for tracing the real source of DDoS attack packets by cluster matching." In: *2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*. IEEE. 2016, pp. 551–555.

[Cid] D. Cid. *Large CCTV Botnet Leveraged in DDoS Attacks*. https://blog.sucuri.net/2016/06/large-cctv-botnet-leveraged-ddos-attacks.html.

[Con] L. Constantin. *Thousands of hacked CCTV devices used in DDoS attacks*. http://www.pcworld.com/article/3089346/security/thousands-of-hacked-cctv-devices-used-in-ddos-attacks.html.

[Con+18] T. M. Conte, E. P. DeBenedictis, A. Mendelson, and D. Milojičić. "Rebooting computers to avoid meltdown and spectre." In: *Computer* 51.4 (2018), pp. 74–77.

[CD16] V. Costan and S. Devadas. "Intel SGX Explained." In: *IACR Cryptology ePrint Archive* 2016.086 (2016), pp. 1–118.

[CP+18] L. Cruz-Piris, D. Rivera, I. Marsa-Maestre, E. De La Hoz, and J. Velasco. "Access control mechanism for IoT environments based on modelling communication procedures as resources." In: *Sensors* 18.3 (2018), p. 917.

[CTK16] B. Cusack, Z. Tian, and A. K. Kyaw. "Identifying DOS and DDOS attack origin: IP traceback methods comparison and evaluation for IoT." In: *Interoperability, Safety and Security in IoT*. Springer, 2016, pp. 127–138.

[Dai+12] A. Dainotti, A. King, F. Papale, A. Pescape, et al. "Analysis of a/0 stealth scan from a botnet." In: *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM. 2012, pp. 1–14.

[DD+17] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi. "Analysis of DDoS-capable IoT malwares." In: *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2017, pp. 807–816.

[Dee98] S. E. Deering. ***RFC 2460*** *Internet protocol, version 6 (IPv6) specification*. 1998.

[DAF18] R. Doshi, N. Apthorpe, and N. Feamster. "Machine Learning DDoS Detection for Consumer Internet of Things Devices." In: *2018 IEEE Security and Privacy Workshops (SPW)*. 2018, pp. 29–35.

[DM04] C. Douligeris and A. Mitrokotsa. "DDoS attacks and defense mechanisms: classification and state-of-the-art." In: *Computer Networks* 44.5 (2004), pp. 643–666.

[Dul+17] A. Dulaunoy, G. Wagener, S. Mokaddem, and C. Wagner. "An extended analysis of an IoT malware from a blackhole network." In: *TNC17*. 2017.

[DGV04] A. Dunkels, B. Gronvall, and T. Voigt. "Contiki - a lightweight and flexible operating system for tiny networked sensors." In: *29th Annual IEEE International Conference on Local Computer Networks*. 2004, pp. 455–462.

[DBH14] Z. Durumeric, M. Bailey, and J. A. Halderman. "An Internet-Wide View of Internet-Wide Scanning." In: *USENIX Security*. 2014, pp. 65–78.

[DWH13] Z. Durumeric, E. Wustrow, and J. A. Halderman. "ZMap: Fast Internet-wide Scanning and Its Security Applications." In: *Usenix Security*. Vol. 2013. 2013.

[EP16] S. Edwards and I. Profetis. "Hajime: Analysis of a decentralized internet worm for IoT devices." In: *Rapidity Networks* 16 (2016).

[FK13] M. O. Farooq and T. Kunz. "Proactive bandwidth estimation for IEEE 802.15.4-based networks." In: *Vehicular Technology Conference (VTC Spring), 2013 IEEE 77th*. IEEE. 2013, pp. 1–5.

[Fero0] P. Ferguson. "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing." In: (2000).

[GT+09] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. "Anomaly-based network intrusion detection: Techniques, systems and challenges." In: *computers & security* 28.1-2 (2009), pp. 18–28.

[GN02] A. Garg and A. L. Narasimha Reddy. "Mitigation of DoS attacks through QoS regulation." In: *IEEE 2002 Tenth IEEE International Workshop on Quality of Service (Cat. No.02EX564)*. 2002, pp. 45–53.

[GWZ] O. Gayer, O. Wilder, and I. Zeifman. *Cctv botnet in our own back yard*. https://www.incapsula.com/blog/cctv-ddos-botnet-back-yard.html.

[GP01] T. M. Gil and M. Poletto. "MULTOPS: A Data-Structure for Bandwidth Attack Detection." In: *USENIX Security Symposium*. 2001, pp. 23–38.

[GP10] C. Gomez and J. Paradells. "Wireless home automation networks: A survey of architectures and technologies." In: *IEEE Communications Magazine* 48.6 (2010).

[GC16] F. Gont and T. Chown. ***RFC 7707*** *Network Reconnaissance in IPv6 Networks*. 2016.

[GAJ11] J. M. Gonzalez, M. Anwar, and J. B. Joshi. "A trust-based approach against IP-spoofing attacks." In: *2011 Ninth Annual International Conference on Privacy, Security and Trust*. IEEE. 2011, pp. 63–70.

[Hilce] S. Hilton. *Dyn Analysis Summary Of Friday October 21 Attack*. https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/. Accessed: 2018-02-11.

[Hög+20] J. Höglund, S. Lindemer, M. Furuhed, and S. Raza. "PKI4IoT: Towards public key infrastructure for the Internet of Things." In: *Computers & Security* 89 (2020), p. 101658.

[Hoq+14] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita. "Network attacks: Taxonomy, tools and systems." In: *Journal of Network and Computer Applications* 40 (2014), pp. 307–324.

[HPJ06] Y.-C. Hu, A. Perrig, and D. B. Johnson. "Wormhole attacks in wireless networks." In: *IEEE journal on selected areas in communications* 24.2 (2006), pp. 370–380.

[HABoo] J. R. Hughes, T. Aura, and M. Bishop. "Using conservation of flow as a security mechanism in network protocols." In: *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. IEEE. 2000, pp. 132–141.

[HT11] J. Hui and P. Thubert. ***RFC 6282** Compression format for IPv6 datagrams over IEEE 802.15. 4-based networks*. 2011.

[HCO16] M.-S. Hwang, S.-K. Chong, and H.-H. Ou. "The Moderately Hard DoS-Resistant Authentication Protocol on Client Puzzles." In: *Informatica* 27.1 (2016), pp. 31–48.

[IB02] J. Ioannidis and S. M. Bellovin. "Implementing pushback: Router-based defense against DDoS attacks." In: (2002).

[Jak+17] A. Jakaria, B. Rashidi, M. A. Rahman, C. Fung, and W. Yang. "Dynamic ddos defense resource allocation using network function virtualization." In: *Proceedings of the ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM. 2017, pp. 37–42.

[JW03] M. Jeronimo and J. Weast. *UPnP design by example*. 2003.

[JWS03] C. Jin, H. Wang, and K. G. Shin. "Hop-count filtering: an effective defense against spoofed DDoS traffic." In: *Proceedings of the 10th ACM conference on Computer and communications security*. ACM. 2003, pp. 30–41.

[JS09] A John and T Sivakumar. "Ddos: Survey of traceback methods." In: *International Journal of Recent Trends in Engineering* 1.2 (2009), p. 241.

[JMB06] A. Johnsson, B. Melander, and M. Björkman. "Bandwidth measurement in wireless networks." In: *Challenges in Ad Hoc Networking*. Springer, 2006, pp. 89–98.

[KKS17] G. Kambourakis, C. Kolias, and A. Stavrou. "The mirai botnet and the iot zombie armies." In: *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*. IEEE. 2017, pp. 267–272.

[Ken17] S. Kenin. *BrickerBot mod_plaintext Analysis*. https://www.trustwave.com/Resources/SpiderLabs-Blog/BrickerBot-mod_plaintext-Analysis/. Accessed: 2018-03-30. 2017.

[Kha+03] S. M. Khattab, C. Sangpachatanaruk, R. Melhem, D l Mosse, and T. Znati. "Proactive server roaming for mitigating denial-of-service attacks." In: *International Conference on Information Technology: Research and Education, 2003. Proceedings. ITRE2003.* IEEE. 2003, pp. 286–290.

[KIAM17] O. Khutsoane, B. Isong, and A. M. Abu-Mahfouz. "IoT devices and applications based on LoRa/LoRaWAN." In: *IECON 2017-43rd Annual Conference of the IEEE Industrial Electronics Society*. IEEE. 2017, pp. 6107–6112.

[KKC19] H.-S. Kim, S. Kumar, and D. E. Culler. "Thread/OpenThread: A compromise in low-power wireless multihop network architecture for the internet of things." In: *IEEE Communications Magazine* 57.7 (2019), pp. 55–61.

[Kol+17] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas. "DDoS in the IoT: Mirai and other botnets." In: *Computer* 50.7 (2017), pp. 80–84.

[Krä+15] L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, and C. Rossow. "Amppot: Monitoring and defending against amplification ddos attacks." In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2015, pp. 615–636.

[Kre16] B. Krebs. *Source Code for IoT Botnet Mirai Released*. https://krebsonsecurity.com/2016/10/source-code-for-iot-botnet-mirai-released/. Accessed: 2018-02-11. 2016.

[Kun] Kunbus. *Revolution Pi*. https://revolution.kunbus.com. Accessed: 2019-12-18.

[Lan01] M. Langheinrich. "Privacy by design—principles of privacy-aware ubiquitous systems." In: *International conference on Ubiquitous Computing*. Springer. 2001, pp. 273–291.

[Lev+05] P. Levis et al. "TinyOS: An Operating System for Sensor Networks." In: *Ambient Intelligence*. Ed. by W. Weber, J. M. Rabaey, and E. Aarts. 2005, pp. 115–148.

[LXC12] J. Liu, Y. Xiao, and C. P. Chen. "Authentication and access control in the internet of things." In: *2012 32nd International Conference on Distributed Computing Systems Workshops*. IEEE. 2012, pp. 588–592.

[LYL08] X. Liu, X. Yang, and Y. Lu. "To filter or to authorize: Network-layer DoS defense against multimillion-node botnets." In: *ACM SIGCOMM Computer Communication Review*. Vol. 38. 4. ACM. 2008, pp. 195–206.

[Lun93] T. F. Lunt. "A survey of intrusion detection techniques." In: *Computers & Security* 12.4 (1993), pp. 405–418.

[Man] *MANAnet: The reverse firewall: defeating DDOS attacks emanating from a local area network*. http://www.cs3-inc.com/MANAnet.html. Accessed: 2019-10-14. 2019.

[Mau96] U. Maurer. "Modelling a public-key infrastructure." In: *European Symposium on Research in Computer Security*. Springer. 1996, pp. 325–350.

[Mer+17] S. Mergendahl, D. Sisodia, J. Li, and H. Cam. "Source-End DDoS Defense in IoT Environments." In: *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*. ACM. 2017, pp. 63–64.

[MES17] L. Metongnon, E. C. Ezin, and R. Sadre. "Efficient probing of heterogeneous IoT networks." In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE. 2017, pp. 1052–1058.

[MS18a] L. Metongnon and R. Sadre. "Beyond telnet: Prevalence of IoT protocols in telescope and honeypot measurements." In: *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*. ACM. 2018, pp. 21–26.

[MS18b] L. Metongnon and R. Sadre. "Fast and efficient probing of heterogeneous IoT networks." In: *International Journal of Network Management* 28.1 (2018), e1997.

[MS19] L. Metongnon and R. Sadre. "Prevalence of IoT Protocols in Telescope and Honeypot Measurements." In: *Journal of Cyber Security and Mobility* 8.3 (2019), pp. 321–340.

[MSE19] L. Metongnon, R. Sadre, and E. C. Ezin. "Distributed Middlebox Architecture for IoT Protection." In: *Proceedings of the 15th International Conference on Network and Service Management*. CNSM. 2019.

[MPR02] J. Mirkovic, G. Prier, and P. Reiher. "Attacking DDoS at the source." In: *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* IEEE. 2002, pp. 312–321.

[MPR03] J. Mirkovic, G. Prier, and P. Reiher. "Source-end DDoS defense." In: *Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003.* IEEE. 2003, pp. 171–178.

[MR04] J. Mirkovic and P. Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." In: *ACM SIGCOMM Computer Communication Review* 34.2 (2004), pp. 39–53.

[MR05] J. Mirkovic and P. Reiher. "D-WARD: a source-end defense against flooding denial-of-service attacks." In: *IEEE transactions on Dependable and Secure Computing* 2.3 (2005), pp. 216–232.

[MRR03] J. Mirkovic, M. Robinson, and P. Reiher. "Alliance formation for DDoS defense." In: *Proceedings of the 2003 workshop on New security paradigms*. ACM. 2003, pp. 11–18.

[MSM08] A. T. Mizrak, S. Savage, and K. Marzullo. "Detecting compromised routers via packet forwarding behaviour." In: *IEEE network* 22.2 (2008), pp. 34–39.

[Mon+07] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. ***RFC 4944*** *Transmission of IPv6 packets over IEEE 802.15. 4 networks*. 2007.

[Moo+06] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage. "Inferring Internet Denial-of-service Activity." In: *ACM Trans. Comput. Syst.* 24.2 (May 2006), pp. 115–139.

[Moo+04] D. Moore, C. Shannon, G. M. Voelker, and S. Savage. *Network telescopes*. Tech. rep. Technical Report CS2004-0795, CSE Department, UCSD, 2004.

[N+18] S. N, A.yegin, B. A, and C. j. *LoRa Specification 1.0.3*. https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf. Accessed: 2020-02-14. 2018.

[Nae+17] P. E. Naeini, S. Bhagavatula, H. Habib, M. Degeling, L. Bauer, L. F. Cranor, and N. Sadeh. "Privacy expectations and preferences in an IoT world." In: *Thirteenth Symposium on Usable Privacy and Security ({SOUPS} 2017)*. 2017, pp. 399–412.

[Nov18] O. Novo. "Blockchain meets IoT: An architecture for scalable access management in IoT." In: *IEEE Internet of Things Journal* 5.2 (2018), pp. 1184–1195.

[Oik+06] G. Oikonomou, J. Mirkovic, P. Reiher, and M. Robinson. "A framework for a collaborative DDoS defense." In: *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*. IEEE. 2006, pp. 33–42.

[Pa+15] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. "IoTPOT: analysing the rise of IoT compromises." In: *EMU* 9 (2015), p. 1.

[Pap+03] C. Papadopoulos, R. Lindell, J. Mehringer, A. Hussain, and R. Govindan. "Cossack: Coordinated suppression of simultaneous attacks." In: *Proceedings DARPA Information Survivability Conference and Exposition*. Vol. 1. IEEE. 2003, pp. 2–13.

[PL01a] K. Park and H. Lee. "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack." In: *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*. Vol. 1. IEEE. 2001, pp. 338–347.

[PL01b] K. Park and H. Lee. "On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets." In: *ACM SIGCOMM computer communication review*. Vol. 31. 4. ACM. 2001, pp. 15–26.

[PLR03] T. Peng, C. Leckie, and K. Ramamohanarao. "Protection from distributed denial of service attacks using history-based IP filtering." In: *IEEE International Conference on Communications, 2003. ICC'03.* Vol. 1. IEEE. 2003, pp. 482–486.

[PLR07] T. Peng, C. Leckie, and K. Ramamohanarao. "Survey of network-based defense mechanisms countering the DoS and DDoS problems." In: *ACM Computing Surveys (CSUR)* 39.1 (2007), p. 3.

[Pox] *Pox controller*. https://openflow.stanford.edu/display/ONL/POX+Wiki. Accessed: 2018-12-07.

[Ram+17] K. R. Ramya, K. A. Gururaj, S Prasanth, K Gayathri, and A. Kumar. "PINPOINTING THE ATTACKER USING AN HYBRID IP TRACE BACK MECHANISM." In: *International Journal of Pure and Applied Mathematics* 115.8 (2017), pp. 451–456.

[Rat+16] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh. "NB-IoT system for M2M communication." In: *2016 IEEE wireless communications and networking conference*. IEEE. 2016, pp. 1–5.

[Rib+03] V. J. Ribeiro, R. H. Riedi, R. G. Baraniuk, J. Navratil, and L. Cottrell. "pathchirp: Efficient available bandwidth estimation for network paths." In: *Passive and active measurement workshop*. 2003.

[San+04] C. Sangpachatanaruk, S. M. Khattab, T. Znati, R. Melhem, and D. Mossé. "Design and analysis of a replicated elusive server scheme for mitigating denial of service attacks." In: *Journal of Systems and Software* 73.1 (2004), pp. 15–29.

[Sar+08] C. Sarr, C. Chaudet, G. Chelius, and I. G. Lassous. "Bandwidth estimation for IEEE 802.11-based ad hoc networks." In: *IEEE transactions on Mobile Computing* 7.10 (2008), pp. 1228–1241.

[Sha+18] F. Shaikh, E. Bou-Harb, N. Neshenko, A. P. Wright, and N. Ghani. "Internet of Malicious Things: Correlating Active and Passive Measurements For Inferring and Characterizing Internet-scale Unsolicited IoT Devices." In: *IEEE Communications Magazine* (2018).

[SB11] Z. Shelby and C. Bormann. *6LoWPAN: The wireless embedded Internet*. Vol. 43. John Wiley & Sons, 2011.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. "RFC 7252 - The constrained application protocol (CoAP)." In: (2014).

[Sim+14] C. Simmons, C. Ellis, S. Shiva, D. Dasgupta, and Q. Wu. "AVOIDIT: A cyber attack taxonomy." In: *9th Annual Symposium on Information Assurance (ASIA'14)*. 2014, pp. 2–12.

[SWH17] R. S. Sinha, Y. Wei, and S.-H. Hwang. "A survey on LPWA technology: LoRa and NB-IoT." In: *Ict Express* 3.1 (2017), pp. 14–21.

[Ste+18] J. Steinberger, B. Kuhnert, C. Dietz, L. Ball, A. Sperotto, H. Baier, A. Pras, and G. Dreo. "DDoS Defense using MTD and SDN." In: *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2018, pp. 1–9.

[SN11] D. N. Sushma and V. Nandal. "Security threats in wireless sensor networks." In: *IJCSMS International Journal of Computer Science & Management Studies* 11.01 (2011), pp. 59–63.

[Sys09] C. Systems. *Home network administration protocol (HNAP) whitepaper*. https://www.cisco.com/web/partners/downloads/guest/hnap_protocol_whitepaper.pdf. Accessed: 2018-03-30. 2009.

[Tal03] G. Taleck. "Ambiguity resolution via passive OS fingerprinting." In: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2003, pp. 192–206.

[TBS17] S. Thielemans, M. Bezunartea, and K. Steenhaut. "Establishing transparent IPv6 communication on LoRa based low power wide area networks (LPWANS)." In: *2017 Wireless Telecommunications Symposium (WTS)*. 2017, pp. 1–6.

[UP13] M Uma and G. Padmavathi. "A Survey on Various Cyber Attacks and their Classification." In: *IJ Network Security* 15.5 (2013), pp. 390–396.

[UTL18] I. Unwala, Z. Taqvi, and J. Lu. "Thread: An iot protocol." In: *2018 IEEE Green Technologies Conference (GreenTech)*. IEEE. 2018, pp. 161–167.

[WRV13] L. Wallgren, S. Raza, and T. Voigt. "Routing Attacks and Countermeasures in the RPL-Based Internet of Things." In: *International Journal of Distributed Sensor Networks* 9.8 (2013), p. 794326.

[WJS07] H. Wang, C. Jin, and K. G. Shin. "Defense against spoofed IP traffic using hop-count filtering." In: *IEEE/ACM Transactions on Networking (ToN)* 15.1 (2007), pp. 40–53.

[WW07] X. Wang and J. Wong. "An end-to-end detection of wormhole attack in wireless ad-hoc networks." In: *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*. Vol. 1. IEEE. 2007, pp. 39–48.

[Wea+03] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. "A taxonomy of computer worms." In: *Proceedings of the 2003 ACM workshop on Rapid malcode*. ACM. 2003, pp. 11–18.

[Win12] T. Winter. ***RFC 6550*** *RPL: IPv6 routing protocol for low-power and lossy networks*. 2012.

[Wix+16] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia, and N. Strachan. "Evaluation of LoRa and LoRaWAN for wireless sensor networks." In: *2016 IEEE SENSORS*. IEEE. 2016, pp. 1–3.

[Xin] Xinhua. *China Focus: Top mobile operators switch on commercial 5G services*. http://www.xinhuanet.com/english/2019-10/31/c_138518626.htm. Accessed: 2019-12-19.

[Xu+18] R. Xu, Y. Chen, E. Blasch, and G. Chen. "A federated capability-based access control mechanism for internet of things (IoTs)." In: *Sensors and Systems for Space Applications XI*. Vol. 10641. International Society for Optics and Photonics. 2018, 106410U.

[YPS03] A. Yaar, A. Perrig, and D. Song. "Pi: A path identification mechanism to defend against DDoS attacks." In: *2003 Symposium on Security and Privacy, 2003*. IEEE. 2003, pp. 93–107.

[YPS04] A. Yaar, A. Perrig, and D. Song. "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks." In: *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*. IEEE. 2004, pp. 130–143.

[YWA05] X. Yang, D. Wetherall, and T. Anderson. "A DoS-limiting network architecture." In: *ACM SIGCOMM Computer Communication Review*. Vol. 35. 4. ACM. 2005, pp. 241–252.

[YWA08] X. Yang, D. Wetherall, and T. Anderson. "TVA: a DoS-limiting network architecture." In: *IEEE/ACM Transactions on Networking (ToN)* 16.6 (2008), pp. 1267–1280.

[Yoo+04] Yoohwan Kim, Wing Cheong Lau, Mooi Choo Chuah, and H. J. Chao. "Packetscore: statistics-based overload control against distributed denial-of-service attacks." In: *IEEE INFOCOM 2004*. Vol. 4. 2004, pp. 2594–2604.

[Yu+15] T. Yu, V. Sekar, S. Seshan, Y. Agarwal, and C. Xu. "Handling a trillion (unfixable) flaws on a billion devices: Rethinking network security for the Internet-of-Things." In: *HotNets 2015*. 2015.

[ZJT13] S. T. Zargar, J. Joshi, and D. Tipper. "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks." In: *IEEE communications surveys & tutorials* 15.4 (2013), pp. 2046–2069.

[Zar+17] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga. "A survey of intrusion detection in Internet of Things." In: *Journal of Network and Computer Applications* 84 (2017), pp. 25–37.

[ZLH03] Y. Zhang, W. Lee, and Y.-A. Huang. "Intrusion detection techniques for mobile wireless networks." In: *Wireless Networks* 9.5 (2003), pp. 545–556.

[Zha+16] C. Zhao, D. Saifuding, H. Tian, Y. Zhang, and C. Xing. "On the Performance of Intel SGX." In: *2016 13th Web Information Systems and Applications Conference (WISA)*. 2016, pp. 184–187.