MOBILE FIREWALL SYSTEM FOR DISTRIBUTED DENIAL OF SERVICE DEFENSE IN INTERNET OF THINGS NETWORKS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Lorenzo Gardea

June 2023

COMMITTEE MEMBERSHIP

TITLE: Mobile Firewall System for Distributed Denial of Service Defense in Internet of Things Networks

AUTHOR: Lorenzo Gardea

DATE SUBMITTED: June 2023

COMMITTEE CHAIR: Dev Sisodia, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Dongfeng Fang, Ph.D.
Professor of Computer Engineering

COMMITTEE MEMBER: Stephen Beard, Ph.D.
Professor of Computer Engineering

ABSTRACT

Mobile Firewall System for Distributed Denial of Service Defense in Internet of Things Networks

Lorenzo Gardea

Internet of Things (IoT) has seen unprecedented growth in the consumer space over the past ten years. The majority of IoT device manufacturers do not, however, build their products with cybersecurity in mind. The goal of the mobile firewall system is to move mitigation of network-diffused attacks closer to their source. Attack detection and mitigation is enforced using a machine that physically traverses the area. This machine uses a suite of security tools to protect the network. Our system provides advantages over current network attack mitigation techniques. Mobile firewalls can be deployed when there is no access to the network gateway or when no gateway exists, such as in IoT mesh networks. The focus of this thesis is to refine an explicit implementation for the mobile firewall system and evaluate its effectiveness. Evaluation of the mobile firewall system is analyzed using three simulated distributed denial of service case studies. Mobility is shown to be a great benefit when defending against physically distant attackers – the system takes no more than 131 seconds to fully nullify a worst-case attack.

ACKNOWLEDGMENTS

Thanks to all of my teachers and professors for encouraging me to study my passion and to be present in class each day. To all professors, especially the committee, who have advised me in my academic and professional pursuits. Thanks to my fellow students for their assistance when learning and for motivating me to strive for more. Thanks to my close friends who I have grow with throughout my years in university. Finally, thanks to my family for their everlasting support.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

Internet of Things (IoT) has become a new and growing paradigm for computing. This can make users' lives much easier by automating routine tasks such as turning off the lights at night. However, most IoT device manufacturers do not take cybersecurity into consideration when designing their devices [6]. The priority is typically to optimize for power and cost constraints. The networking protocols used by IoT devices are insecure as well. Novel protocols tailored to IoT device constraints have been designed [27], but this only corrects a fraction of the vulnerabilities used in common IoT network protocols.

Research has proven that although improvements are being made to IoT device integrity and confidentiality, there are still methods that attackers can leverage to expose information. Ren et al. found that poor privacy practices resulted in IoT devices contacting 3rd parties unnecessarily [21]. The authors also trained a machine learning algorithm and found that they could successfully infer device activities. In addition to exposing user information, IoT device vulnerabilities are capable of enabling massive damage to computer networks.

It is important to define technical terms before we explain their place in the context of this thesis. **IoT** refers to the greater concept of Internet of Things – the idea of extending Internet connectivity to machines that interact with their physical environment. An **IoT device** is a computer that fits the IoT paradigm. IoT devices are typically Internet-connected machines with constraints to their size, power draw,

battery life, and/or processing strength. IoT devices interact with their physical environment and perform tasks that require data acquisition or automation.

The definition for **IoT network** is flexible. IoT networks refer to collections of interconnected IoT devices that do not require connection to the internet to communicate with each other. This may imply that the devices are contained within the same subnet or autonomous system. The term IoT network may also be used to describe the connections and topologies the collective IoT devices use for transmission. **IoT-enabled** refers to anything that uses IoT devices to achieve its goal. We consistently use this term in the context of DDoS attacks that are made possible by the large number of IoT devices controlled in botnets.

The scope of computer security is broad, so we must identify its meaning in the context of this thesis. The process of preventing unwanted access, disclosure, interruption, and modification of computer systems is known as computer **security** in this thesis. It encompasses a variety of methods intended to protect the privacy, accuracy, and accessibility of data in computer systems. We strive to achieve the best security possible, although these methods can never guarantee that a system is fully resistant to threats. In the subject of computer security, the CIA triad is a frequently recognized paradigm. It stands for three fundamental security principles of confidentiality, integrity, and availability.

According to the CIA triad, a computer system or asset is **secure** if it has taken the necessary precautions to safeguard its confidentiality, integrity, and availability. It makes sure that unauthorized people cannot access private information, that data is correct and undisturbed, and that systems work properly when needed. Security measures and security functions refer to the actions that systems take to defend their confidentiality, integrity, and availability.

Following are three examples of scenarios when security is affected:

- A computer user begins encrypting their plaintext. This results in unintended viewers being unable to see the data. Confidentiality is improved.
- A bad actor intercepts communications and modifies them to pose as someone that they are not. Integrity is diminished.
- A bad actor unplugs the router that a legitimate computer is communicating with. The computer can no longer perform its required functions. Availability is diminished.

The security objective that we will be focusing on in this thesis is availability. We use distributed denial of service as a case study for how the mobile firewall system can be applied to help secure collections of devices from network-diffused threats. Distributed denial of service is a textbook case of an attack on availability. We will also examine in Section 4.4 how Internet Protocol (IP) address spoofing detrimentally affects the availability of benign computers. Although we are using distributed denial of service as a case study and focusing on availability, it is important to understand that the mobile firewall system targets any network-diffused threat. Examples of network-diffused threats include port scanning, brute-force attacks, command and control channels, and distributed denial of service.

1.1 Motivation

Distributed denial of service (DDoS) is not a brand-new problem in the field of computer networks. Denial of service attacks have been recorded to happen since the 1990's. The emergence of IoT has only made the situation worse [6, 22]. IoT devices

can be used as part of a botnet, or they can be used to spread worms into other IoT networks [9]. IoT devices have proved to be most impactful when participating in large-scale botnets. In 2017, Google was the target of a DDoS attack with a peak throughput of 2.54 Tbps [14]. Google expects the global DDoS throughput to scale exponentially over time as depicted in Figure 1.1. One of the largest IoT exploits to affect the Internet, the Mirai botnet, took advantage of common IoT device vulnerabilities to infect an estimated 250,000 devices [23].

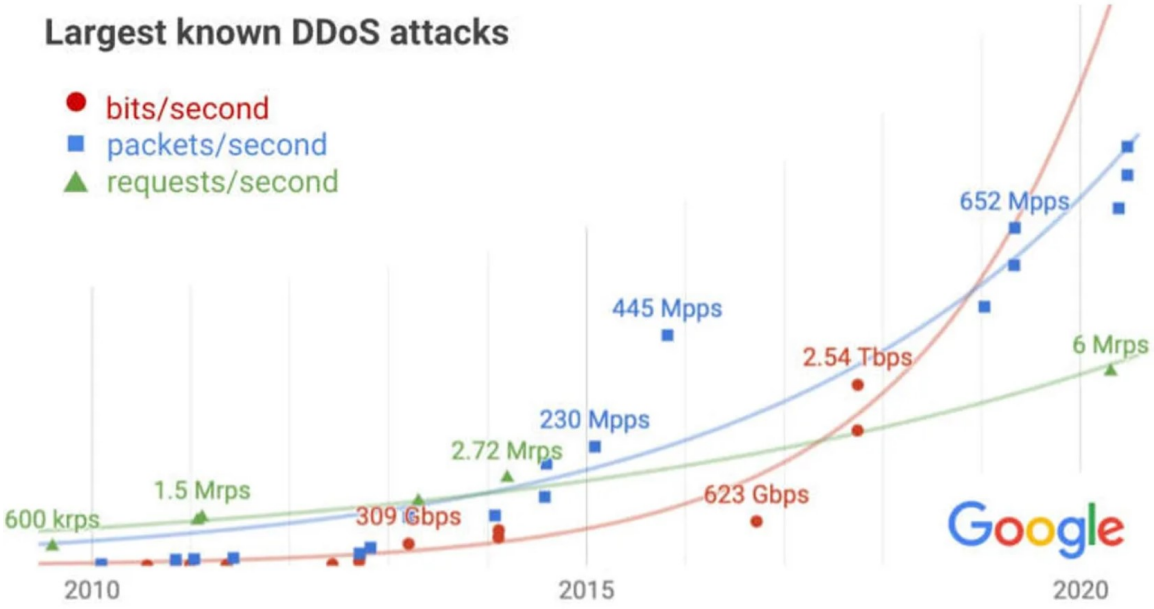

Figure 1.1: DDoS Growth Prediction [14]

The Mirai botnet was used multiple times to attack different targets including the cloud provider OVH, and DNS provider Dyn. However, the most influential event was the attack on Lonestar Cell. Lonestar Cell is a Liberian telecommunications company whose victimization by Mirai resulted in significant Internet service degradation for the entire country of Liberia. Sales of IoT devices, including wide-area and cellular IoT devices, are expected to grow. Figure 1.2 is an estimation of how many IoT devices will enter the Internet ecosystem by 2028 [31]. Because of the historical damage caused by DDoS attacks originating from IoT devices, and the growth of IoT

device sales, we find it imperative to solve the looming threat of IoT-enabled DDoS attacks. The best place to stop these attacks is at the source (within the IoT networks themselves), since this averts links from wasting their resources on malicious traffic.

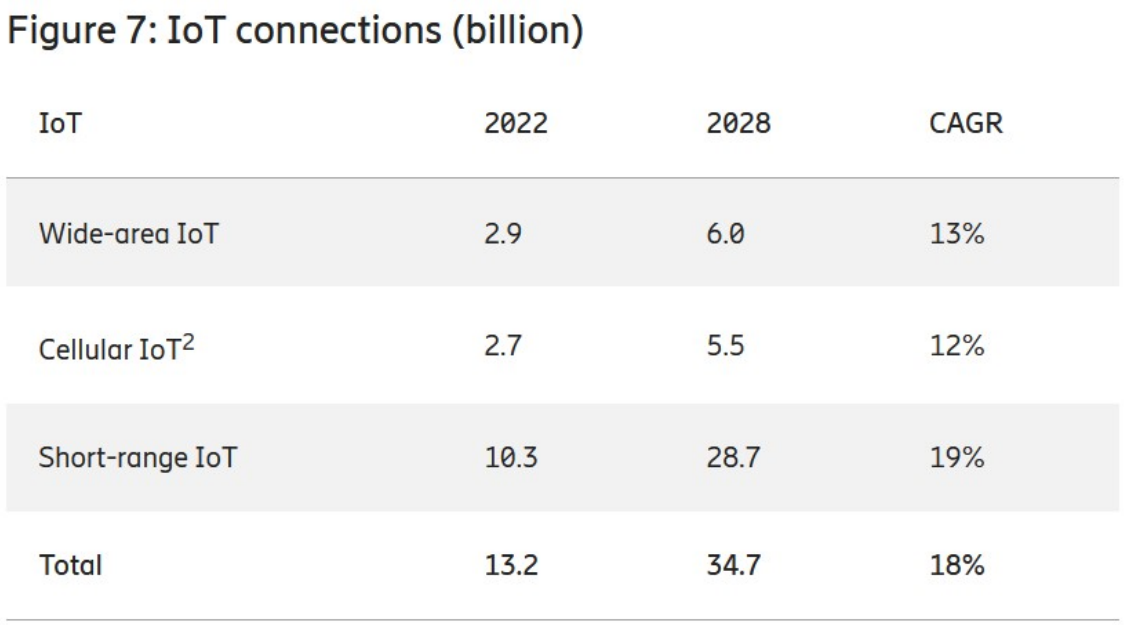

Figure 7: IoT connections (billion)

IoT	2022	2028	CAGR
Wide-area IoT	2.9	6.0	13%
Cellular IoT[2]	2.7	5.5	12%
Short-range IoT	10.3	28.7	19%
Total	13.2	34.7	18%

Figure 1.2: IoT Device Growth Forecast [31]

1.2 Characteristics of IoT Networks

Internet of Things networks are unique when compared to other types of computer networks and because of this, require distinctive solutions. The first special attribute of IoT networks is the uniqueness in communication protocols that they leverage. Unique communication protocols result in tailored security needs for each device. IoT devices use a variety of (often wireless) communication protocols to transmit data, such as Z-Wave, Zigbee, 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks), and cellular [27]. Cellular protocols in particular are special in that they allow any device with service to transmit data outside of the network, evading traditional network gateways.

DDoS will take different forms along with the different protocols. For example, one network resilient to SYN flood may not be resilient to low-rate DDoS. Similar to supporting different kinds of protocols, the diversity in the types of IoT devices commercially available is very wide. A second unique characteristic of IoT networks is that they consist of a wide variety of devices, such as sensors, actuators, cameras, and other types of smart devices [21]. These devices have assorted sizes, power requirements, and traffic throughput demands. We recognize that heterogeneity applies to both protocols and device type/demands. We keep these two characteristics separate because each result in unique solution requirements.

A third quality of IoT networks is that they have the potential to be very large-scale. IoT networks can be broad, consisting of hundreds or even thousands of machines spread over a wide geographic area. Scalability is an important factor when IoT is used for industrial automation or agricultural purposes [44]. The distribution and connectedness of the devices will vary as a result. Connectedness is further complicated by the fact that devices may be migrant, periodically joining and exiting the network. One factor that may contribute to dynamic topology and routing is that devices duty-cycle to conserve energy. Just as the network changes, so can the usages of the devices.

A fourth quality of IoT networks is that they can be applied to many different purposes – with no guarantees about the centrality of the network. Some common IoT device use-cases are for smart homes, university campuses, industrial automation, and healthcare electronics. These environments may impose more restrictions on their IoT network infrastructure than typical computer networks [38]. This can result in inaccessibility or evasion of a traditional network gateway. Inaccessibility of a gateway and evasion of a gateway are two different concepts that both result in openness.

Mesh networking allows nodes to act as routers, creating spots in the network where egress traffic is unfiltered [62]. We define open IoT networks in this thesis to refer to these networks that have inaccessible or avoidable gateways. For example, IoT devices set up in a university environment may be connected to the Internet via 5G cellular communications. The devices can evade the centralized security provided by the university (in the form of a gateway). The university cannot deploy security measures on the 5G base station, so they would need solutions that remedy this inaccessibility.

To summarize, these are the four traits of IoT networks that we will be focusing on:

- Heterogeneity in communication protocols.
- Variety in device type and specification.
- Dynamic scale and distribution.
- Openness – inaccessibility or evasion of a traditional network gateway.

1.3 Solution Requirements

The uniqueness in traits presented by IoT networks causes a need for unique solutions. But how do these traits translate to engineering requirements? Firstly, heterogeneity in communication protocols results in a need for security solutions that are flexible in terms of protocol support [45]. The ideal solution can be deployed among many IoT devices, and it should not be protocol specific. The solution should still be able to defend against DDoS whether it is propagating through 6LoWPAN, Zigbee, or any other wireless protocol.

The fact that nodes may be able to join and leave the network at their own will means that new threats will be introduced [45]. Different security measures will need to be stored and ready for use as the attack surface increases. The second trait we focus on is variety in device type and specification. Security solutions should ideally be accessible for every device in the network. Each device must be protected using our ideal solution. This does not matter if the device is small and communicates infrequently under regular operation. Security must be provided when devices are power constrained and computationally restricted.

Scale and distribution are the third trait that we identify requirements for. The scale of IoT networks may range from a few devices to thousands. Environments such as agricultural tracts may have the additional trait of being geographically vast, causing devices to be distributed across multiple areas [44]. Deployment of the ideal solution must be scalable as the number of devices increases and physical area expands. The solution must be capable of handling the dynamic nature of routing and topology changes.

The fourth trait we are focusing on is inaccessibility or evasion of a traditional network gateway. The example presented earlier regarding the university campus being unable to deploy security measures on a base station is an example of inaccessibility. Cellular communications make it difficult to defend against DDoS in open IoT networks since some nodes may be able to forward traffic through 5G – all the while evading a gateway firewall. We recognize that cellular protocols have built-in security measures, but we would like to develop a solution that prevents devices in our network from transmitting DDoS traffic. The ideal security solution does not rely on a central gateway; it can protect all potential points in the network where traffic may exit [57]. The solution must function when the network gateway is inaccessible.

To summarize, these are the engineering requirements of a solution to IoT-enabled DDoS:

- Flexibility and extensibility for new protocols and security measures.
- All devices in the network are protected from network-diffused threats, regardless of their constraints.
- Resilient to changes in physical distance, number of nodes in-network, and routing/topology.
- Does not rely on a centralized or accessible gateway.

1.4 Prior Work

Sisodia et al. present a novel solution for IoT-enabled DDoS defense in *Defending Against IoT-Enabled DDoS Attacks at Critical Vantage Points on the Internet* [59]. The authors present a system called the mobile firewall system. The mobile firewall system consists of two new physical devices that are added to an existing collection of IoT devices – the mobile security node and the security controller. It is possible for more than one mobile security node to be integrated with the system to produce a more distributed deployment.

The role of the mobile security node is to physically move around an area of devices, gathering telemetry information for the security controller. We assume the mobile node has the ability to eavesdrop communications. The system can be made mobile through the use of drones, sneakernet, robots, or other methods. The mobile security node is allowed to be resource constrained. It does not have a connection to the Internet, although future work may reinvent the mobile firewall system with a *security*

as a service approach. The mobile node would then require an Internet connection to communicate with the cloud security controller; but for now there is no cloud communication.

The role of the security controller is to process the telemetry information and make intelligent decisions about threats present within the network. These decision are used to select functions that are passed back to the mobile node, whose physical mobility allows it to enforce security functions near the devices needed. The security controller is meant to operate as a framework. It allows for modular integration of off-the-shelf tools to be used as attack detectors. The idea of extensibility applies to security functions too, as these functions can be saved in the security controller and sent to the mobile node at will. The security controller is meant to be a higher power device. Refer to Section 4.5 for more detailed information on how the mobile firewall system operates.

The role of the mobile firewall system as a whole is to protect legitimate machines from network-diffused threats, as well as prevent any malicious machines from contributing to attacks. Examples of network-diffused threats include port scanning, brute-force attacks, command and control channels, and distributed denial of service. The network-diffused attack that we will be focusing on in this thesis is distributed denial of service. The mobile firewall system accomplishes bringing a heightened state of protection to all devices within the IoT network. Refer to Section 4.4 for the exact security objectives of the mobile firewall system. The target application of the mobile firewall system is for usage in any decentralized network or network without an accessible gateway. We study IoT networks in this thesis because of their usage of cellular routing and mesh routing, and the challenges associated with each.

Mesh routing removes the convenience of having a centralized gateway for monitoring traffic. Cellular routing results in inaccessible gateways. An example of inaccessible

gateways is that an average user cannot deploy their IDS in a 5G base station. The mobile firewall system allows for decentralized monitoring and enforcement of security functions. Monitoring is no longer centralized around a traditional gateway. It is possible for cellular traffic to go around the traditional border router, but not the mobile node. Monitoring is now conducted by the mobile node, which is not confined to one particular area of the network. Mobile firewall nodes can sniff traffic no matter where it exits. Multiple mobile nodes may be used for a more distributed solution.

We assume that all nodes may have cellular capabilities. The mobile firewall's goal is to come between nodes and perform firewall duties. It is not required for the mobile node to be connected in the same network as the devices it is monitoring. The only requirement is that it is within range to sniff communications. Depending on the security functions, it may be possible to detect and jam communications of devices with direct cellular capability. We leave this particular security function for future work. Another topic is scalability. How many mobile security nodes cover how many existing/legitimate nodes? We evaluate scalability in Section 6 and Section 7.

1.5 System Goals

The task of the mobile firewall system is to move the DDoS detection and mitigation closer to the source of the attack. Because of the unique challenges presented by defending IoT networks, there are multiple goals for this system. The first is that it must support many different communication protocols and security measures. This is taken into consideration when designing the modular attack detector threads present in the decision module of the security controller. Storage and deployment of security measures is provided by the security function database module.

The second goal of the mobile firewall system is that it protects all devices in the network, including those resource constrained, from bad actors. The mobile firewall system is not limited by weak hardware when compared to host-based intrusion detection (see Section 2.3 for more information). In addition, the security controller is what provides enhanced compute power to aid the mobile node.

A third goal of the mobile firewall system is that it continues to operate when physical distance between devices, number of devices in the network, and topologies change. This problem is alleviated by the distributed nature of the mobile firewall. As the area and number of devices increases, IoT network operators can add mobile firewalls to the network. Dynamic distribution allows the system to handle dynamic changes.

The fourth goal of the system is that it does not rely on a centralized or even accessible gateway. This goal is met because the system is mobile and distributed across the network. The attribute of mobility allows the firewall system to act as a gateway, routing and dropping traffic accordingly. The possibility of adding multiple firewall systems distributed across the network allows the system to enforce without relying on centralization.

1.6 System Advantages

While other solutions exist for network administrators to detect DDoS, it is important to understand the advantages that the mobile firewall methods have. One advantage is modularity. There is a great level of flexibility for users to add support for many attack detectors and as such, protocols. This gives the mobile firewall system a form of future-proofing as new standards emerge. Modularity also applies to the development of intrusion detection systems, which are currently being advanced by artificial intelligence. The extensibility of the security controller allows it to handle many dif-

ferent types of security measures for any threats that the network may encounter. The advantage of the mobile firewall system is that it is adaptable with the dynamic needs of the network.

A second advantage is that the mobile firewall helps secure resource-constrained devices. Host-based security may not even be able to run on IoT devices, which are often constrained in their compute capabilities. Administrators may be unable to install endpoint management software because the devices themselves are difficult to access (both physically and software-wise). An example of this is real-time embedded systems with limited operating system support. Changes to the device may require it to power-off, which is unacceptable in certain mission-critical scenarios. The mobile firewall system has the advantage of monitoring these smaller devices. An excellent quality is that it requires no modification to any of the IoT devices, nor does it require installing new software. It also takes away the need of configuring host-based security on devices that are unlimited in compute power. With the mobile firewall system, the network operator has the choice of what level of security they would like to configure. There are options for the trade-offs between security and ease of deployment.

Another advantage when compared to other forms of security is that this solution is much more scalable. Installing and upgrading host-based DDoS detection on every single device can be tiresome for a network administrator. With the mobile firewall system, the network administrator only needs to maintain the mobile firewalls to keep the ecosystem secure. Depending on how many devices are being secured by the mobile firewall, this may save an invaluable amount of time. The mobile firewall system may be used in conjunction with an endpoint management system. The mobile firewall protects devices when they are unable to run endpoint management software due to computation limitations. It protects devices that are unsupported by endpoint management software.

The two most powerful advantages of the mobile firewall over current DDoS mitigation techniques are its mobility and distribution. The first benefit is that mobile firewalls may be set up in mesh networks for IoT devices when there is no gateway or no means to deploy security solutions on one. For example, an average user cannot deploy an IDS in a 5G base station. Mobility provides it with the power to function as a gateway and assist in routing traffic for DDoS defense. Openness of the IoT network may inhibit the ability of the central gateway to collect all traffic. The mobile firewall system assists the gateway by mobilizing toward problematic nodes, allowing it to defend against DDoS through its own routing.

The next most influential advantage is that the system does not rely on network centrality. We acknowledge that the system is centralized around two machines, the mobile node and the security controller. When saying that the system does not rely on centrality, we mean that security functions and networking policies are enforced all around the topological area of devices, not just the gateway. The system does not need a single point of collection that traffic is routed through to defend against DDoS. Networks that allow traffic to exit anywhere are protected because the mobile firewall can be distributed all around the network. The mobile firewall catches traffic that is not picked up by a traditional border gateway. Mobile firewall nodes sniff traffic no matter where it exits. Distribution gives the mobile firewall system an advantage by easily adapting to changes in network topology.

Last but not least, placing DDoS detection downstream such that it is closer to the source-end DDoS attacker allows it to be identified earlier than upstream. As a result, the network is less congested and the DDoS assault victim isn't overloaded. It prevents links from wasting their energy on malicious traffic, and it saves legitimate services' time by reducing malicious throughput [52]. A larger percentage of DDoS

traffic can be blocked because the overall scale and volume of DDoS traffic is more manageable than on the victim-end.

1.7 Thesis Contributions

The implementation and evaluation of this thesis' mobile firewall system are an expansion of research presented by Dev Sisodia in his doctoral dissertation [59]. While the potential and high-level plan of the mobile firewall system is detailed in [59], the goal of this thesis is to produce a more explicit implementation and test it. Effectiveness of the system is evaluated in order to better understand the potential of the mobile firewall system. Significant attention is placed on a modular approach of implementation to easily support additional components. This allows for a high degree of flexibility when users would like to try different system components or when student researchers would like to modify the existing software architecture. The contributions of this thesis are as follows:

- We present an implementation of the mobile firewall system from [59].
- We investigate deployment of the mobile firewall in simulation, using three case studies.
- We evaluate the performance of the mobile firewall system.
- We identify areas of future work for the mobile firewall system.

1.8 Thesis Organization

The organization of this thesis is depicted in Table 1.1:

Table 1.1: Thesis Organization

Chapter	Title	Overview
1	Introduction	Introduction to IoT networks and their security issues. Introduction to distributed denial of service attacks. Motivation and advantages for designing the mobile firewall.
2	Background	In-depth discussion on IoT. In-depth discussion on distributed denial of service. In-depth discussion on intrusion detection and mitigation systems.
3	Literature Review	Review of current research in IoT intrusion detection. Focus on signature-based, machine-learning based, and their limitations. Review of distributed denial of service defense. Focus on critical defense points in the Internet.
4	Design	Explanation of our design decisions and advantages of IoT source-end detection. Explanation of our design decisions and advantages of firewall distribution. Threat modeling and explanation of security objectives. Overarching design of the mobile firewall system.
5	Implementation	Explanation on implementation of security controller. Focus on implementation choices for the decision module, including attack detector threads and security functions. Explanation on code base additions to RPLD for a viable RPL implementation. Explanation on the mobile security node.
6	Evaluation	Discussion on choices for network simulation. Explanation on why Mininet is chosen, and Mininet simulation setup. Focus on implementation of background traffic and DDoS scripts. Presentation of three case studies and their support to evaluation.
7	Results	Presentation of results from each case study evaluated. Explanation of effectiveness of system. Focus on timing measurements and traffic throughput measurements.
8	Conclusion	Summary of work. Discussion of successes, limitations, and future work.

Chapter 2

BACKGROUND

In this chapter, we explain the background on technologies used in this thesis. The technologies covered in this chapter may be used for the implementation of the mobile firewall system or for the evaluation of the mobile firewall. We explain what the concepts are and give descriptions of their uses. 6LoWPAN and RPL are two important protocols in used IoT scenarios that we investigate. We present a brief study of distributed denial of service. A background on intrusion detection systems is presented.

2.1 Internet of Things

Internet of Things (IoT) is a network of physically connected objects with software and communication features built in [18]. Devices may consist of sensors, actuators, and beacons. These devices can interact with other machines all over the world thanks to their ability to gather and exchange data over the Internet. IoT extends the idea of Internet connectivity to everyday environments and objects beyond more conventional computing devices like computers and smartphones. Smart thermostats, smart lamps, industrial machinery, and even smart city infrastructure are a few examples of IoT.

Enabling these objects to connect and share data is the core idea of IoT. Sharing allows for increased convenience in various domains such as healthcare, agriculture, manufacturing, and more. New autonomy for industrial usage is introduced by IoT. IoT devices have the ability to acquire data, evaluate it, and then make wise judgments to decide what actions to take in response to that data.

2.1.1 IPv6 Over Low-Power Wireless Personal Area Networks

IPv6 over Low Power Wireless Personal Area Network (6LoWPAN) is a communication protocol designed specifically for low-power devices and networks in the context of IoT [27]. It enables the transmission of IPv6 (Internet Protocol version 6) packets over low-power wireless networks such as Zigbee, Thread, Matter, or Bluetooth Low Energy (BLE) [47].

The limitations that IoT devices frequently face are addressed by 6LoWPAN. The goal is to reduce processing power, memory, and energy requirements. 6LoWPAN's IPv6 adaptation to low-power networks enables these devices to take advantage of IP connectivity's advantages and seamlessly join the larger Internet.

The protocol achieves efficient packet transmission by compressing IPv6 headers and adapting them to the characteristics of low-power networks. Fragmentation is supported to keep compatibility with data link protocols that support larger maximum transmission unit (MTU) sizes. It also supports mesh networking, where devices can relay packets to extend the network coverage. Mesh networking is related to the threat model because it creates pockets in the network where DDoS data may exit unfiltered.

By facilitating communication amongst low-power devices, 6LoWPAN serves as a foundation of support for IoT. It enables them to interface with cloud services through the Internet, offering remote control and monitoring capabilities. IoT devices may take advantage of the vast array of infrastructure offered by the IP-based Internet ecosystem thanks to 6LoWPAN [27].

2.1.2 Routing Protocol for Low-Power and Lossy Networks

Routing Protocol for Low-Power and Lossy Networks (RPL) is a routing protocol specifically designed for low-power and lossy networks in the context of IoT [61]. RPL is a distance-vector routing protocol that builds and maintains a routing topology among the nodes in the network. It operates on a destination oriented directed acyclic graph (DODAG) structure, where each node in the network is represented as a vertex, and the edges represent the links between nodes. Figure 2.1 shows an example of a DODAG. Edges are oriented in the direction of the root node R.

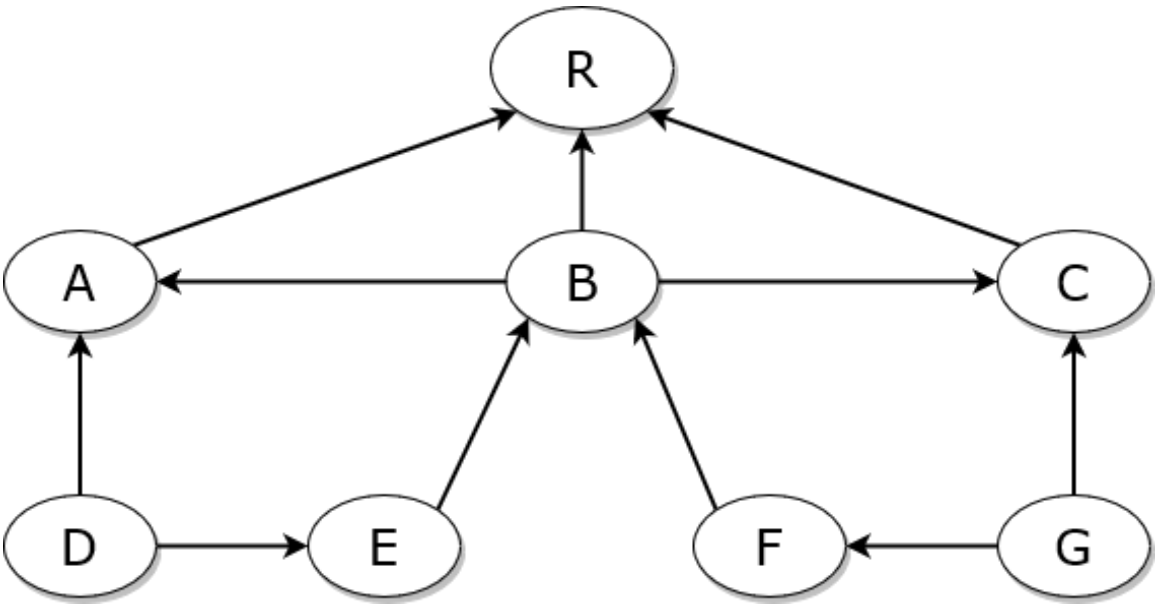

Figure 2.1: DODAG Example

RPL uses a concept of "parent" and "child" relationships between nodes in the DAG. Each node maintains information about its parent nodes and selects the optimal parent based on defined routing metrics. Data packets are then forwarded from child nodes to their parent nodes, eventually reaching the desired destination.

RPL uses a proactive approach to maintain routing information, allowing nodes to continuously update their routing tables and be aware of the network topology. It

leverages a Trickle algorithm to minimize control message overhead and network resource consumption [54]. The Trickle algorithm governs the frequency at which nodes exchange control messages based on a configurable interval and redundancy parameters.

There are three main types of control messages: DIS, DIO, and DAO [61]. DODAG Information Solicitation (DIS) messages are sent by a node that intends to join the RPL network and request information about the DODAG. DODAG Information Object (DIO) messages are sent by a node to distribute information about the DODAG, including the node's own rank. DIO messages are periodically transmitted and allow neighboring nodes to build and update their routing tables. Destination Advertisement Object (DAO) messages are used for route maintenance and are sent by a node to advertise available routes towards specific destinations.

2.2 Distributed Denial of Service

A cyberattack known as a distributed denial of service (DDoS) involves the use of numerous compromised devices, frequently those that are a part of a botnet, to flood a target system with a massive amount of malicious traffic. The intention is to exhaust the victim's resources and render its services unavailable to legitimate users. Organizations may experience disruptions or downtime as a result of DDoS attacks, which can have a significant impact on the availability of targeted systems. Figure 2.2 depicts a simple example of how a DDoS attack may be set up.

When it comes to IoT, DDoS attacks pose a particular threat due to the substantial number of interconnected devices. Many IoT devices have limited security measures, making them easy to compromise [51]. Attackers can exploit security weaknesses in these devices to exploit them as part of a botnet. Botnets form a powerful network

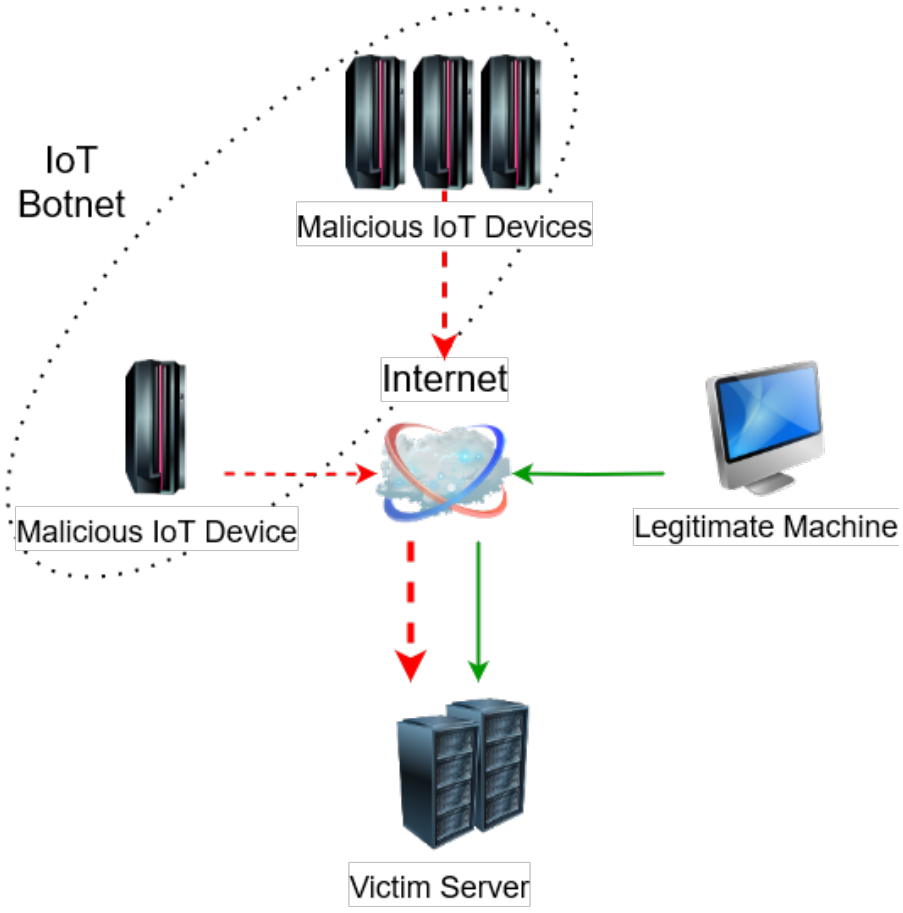

Figure 2.2: DDoS Diagram

of compromised devices that can be used to launch DDoS attacks. The volume of IoT machines and their interconnectedness amplifies the potential impact of DDoS attacks. Attackers can generate massive traffic volumes capable of overwhelming the target's infrastructure by harnessing the compute power and network bandwidth of numerous compromised IoT devices.

There are several distinct types of DDoS attacks that each employ different techniques to degrade system availability. Volumetric attacks aim to flood the target network or system with a massive amount of traffic, consuming its bandwidth and overwhelming its infrastructure [2]. Protocol attacks exploit weaknesses in network protocols to

exhaust system resources or disrupt communication [4]. Application layer attacks target specific applications or services rather than the network infrastructure. The objective is to overwhelm the target by exploiting vulnerabilities in the application layer, often using legitimate-looking requests [3].

UDP (User Datagram Protocol) flood is a volumetric attack that relies on sending many packets of raw bytes to the victim to overwhelm their processing capacity. This attack may target firewalls themselves as well as servers to incapacitate their resources. The victim of the attack must take time and resources to determine whether there are services utilizing the ports associated with the UDP traffic. The server is unable to serve legitimate requests when there is a high volume of malicious UDP packets inbound. SYN (SYNchronize sequence number) flood is a protocol attack that relies on an exploit in the TCP (Transmission Control Protocol) handshake process to make resources unavailable for legitimate users. By sending a SYN request, the victim server will temporarily allocate resources to the user requesting the sequence number synchronization. The DDoS attacker will send many SYN requests without actually making use of the allocated resources, resulting in less open ports for legitimate users.

2.3 Intrusion Detection System

Intrusion Detection Systems (IDSes) are security tools that monitor system activities to identify and respond to suspicious behavior [15]. They analyze network traffic, system logs, and other data sources to detect potential security threats, including DDoS attacks. Some intrusion detection systems are equipped with tools that help prevent attacks too.

IDSes record relevant information about detected DDoS attacks, including timestamps, source IP addresses, and attack characteristics. This data can be used for forensic analysis to understand the attack vectors. This knowledge ultimately strengthens future defenses against similar attacks. IDSes can generate real-time alerts when suspicious activity indicating a DDoS attack is detected [55]. The alerts help network administrators take immediate action, stopping the attacks at the earliest stages possible.

The two primary types of IDS are host-based IDS and network-based IDS. Host-based IDSes monitor the activities occurring on the host. Examples of activities are file system changes, processes, and system calls [15]. They analyze the host's behavior to detect potential signs of compromise. Host-based IDSes provide detailed insight into the security of individual hosts but are limited to the scope of the monitored system.

Network-based IDSes are positioned strategically throughout the network infrastructure, such as at the edge of the network. They analyze the headers and traffic patterns of network packets to spot any potential security risks [15]. In contrast to host-specific IDSes, network-based IDSes concentrate on spotting attacks that travel across the network.

IDSes are further categorized by their detection mechanism. Signature-based IDSes, also known as rule-based IDSes, rely on a database of patterns of known malicious activities [15]. The IDS compares the network behavior against these signatures and raises an alert if a match is found. Signature-based IDSes may struggle to detect novel attacks that do not match any existing signatures.

Anomaly-based IDSes establish a baseline of normal behavior by monitoring network activities over time [15]. They then compare the observed behavior against the base-

line to identify any anomalies that may indicate potential intrusions. Anomaly-based IDSes are effective at detecting zero-day attacks, as they can identify abnormal behavior that does not match any known attack pattern. However, there is a risk of higher false positives rates due to legitimate variation in network behavior.

Chapter 3

LITERATURE REVIEW

In this chapter, we discuss current research related to the mobile firewall system. We use DDoS as a case study in this thesis for how the mobile firewall system may be used to defend networks. Thus, we present literature review on recent DDoS mitigation research. We cover victim-end, in-network, and source-end DDoS mitigation schemes. It is important to note that the mobile firewall system may be used to counter any threats to the network, not just DDoS. We focus on source-end defenses because of their relevance to the mobile firewall system. We examine IoT intrusion detection and prevention solutions. Signature-based IDSes and ML-based IDSes are two important traits of intrusion detection that we examine to fully understand how the mobile firewall system may be implemented.

3.1 DDoS Mitigation

Two Decades of DDoS Attacks and Defenses, by Shi et al. at the University of Oregon, provides a comprehensive review of the latest DDoS attacks and defense solutions [22]. Some of the defenses investigated include rate limiting, whitelisting, blacklisting, and software filtering. The author highlights advancements and gaps in DDoS research. The author stresses that there is a need for deployable solutions. We take the requirements of deployable into consideration when designing the mobile firewall system. Another need the author highlights is scalability. We seek to satisfy these scalability demands in the Design chapter.

A Privacy-Aware Collaborative DDoS Defence Network, published by Fung et al. at the 2020 IEEE/IFIP Network Operations and Management Symposium, proposes a solution to victim-end DDoS mitigation while respecting user privacy [43]. The solution works by redirecting excessive connection requests to a federation of other servers. These servers (located in different domains) filter traffic during a DDoS attack. The federated model has its advantages, and it lends itself to the mobile firewall system. One of the qualities we replicate is the distributed nature of the solution. Distributedness provides a good characteristic for the mobile firewall system, allowing it to respond when there are multiple attackers.

In-Network Filtering of Distributed Denial-of-Service Traffic with Near-Optimal Rule Selection, published by Sisodia et al. at the 2020 ACM Asia Conference on Computer and Communications Security, introduces a new operational model for in-network DDoS defense and offers a solution to the rule selection problem [60]. The authors' offer-based model for in-network DDoS defense allows victims to use source IP prefix-based rules on networks other than their own. The rules express their filtering needs. The mitigation process (generation of DDoS traffic filtering rules) begins once a DDoS attack is detected. This research presents an alternative to source-end defenses like the mobile firewall. Although this research is not directly related to the mobile firewall system, it is still important to understand how rules can be used to filter traffic – an action the mobile node takes to defend against DDoS.

3.1.1 Source-End Defense

Defending Against IoT-Enabled DDoS Attacks at Critical Vantage Points on the Internet, by Sisodia et al. at the University of Oregon, details the potential and high-level plan of the mobile firewall system [59]. The goal of this thesis is to refine a more explicit implementation and assess it. Significant attention is placed on a modular

approach of implementation to easily support additional components. This allows for a high degree of flexibility when users would like to try different system components or when student researchers would like to modify the existing software architecture. Sisodia et al. includes analysis on how the system might behave when travelling using simulation [59]. The author found that while travelling, the amount of DDoS traffic dropped rose as the number of infected devices rose. The ratio of dropped to allowed DDoS traffic rose as the number of infected devices rose.

ML-DDoS: A Blockchain-Based Multilevel DDoS Mitigation Mechanism for IoT Environments, published by Hayat et al. in the 2022 IEEE Transactions on Engineering Management journal, proposes a device-based verification using blockchain to prevent malicious devices from participating in IoT environments [46]. This research solution is enforced at the edge of the source network, blocking DDoS from exiting to the open Internet. The mobile firewall system utilizes a blocking system with a similar result to stop DDoS. In the case of the mobile firewall though, the blocking is engaged by the RPL black hole mitigation. This research presents a solution that may be used by the mobile firewall as a security function in the future. The benefit of this research is that it uses blockchain to maintain an agreement of trust among nodes. This solution works in situations where network integrity may be compromised, such as when malicious devices spoof their IP addresses.

DDoS Defense for IoT: A Stackelberg Game Model-Enabled Collaborative Framework, published by Xu et al. in the 2022 IEEE Internet of Things Journal, proposes a multipoint DDoS defense framework to address IP spoofing in 5G networks [39]. The defense is based on a proposed novel packet sampling strategy, which is deployed near the DDoS source network and maximizes IoT utility compared with existing methods. The packet sampling investigated in the research may be applied to the mobile firewall node as a reconnaissance security function. Similarly, we investigate

other packet-sampling technologies in Section 5.2. Another interesting point of this research is the fact that the authors took cellular services into consideration. Cellular poses a big threat in IoT applications by creating gaps where traffic may exit the network. We aim to reduce the threat of DDoS traffic travelling through cellular nodes by using the mobile firewall.

The mobile firewall system that we propose in this thesis has advantages over the related work discussed. Fung et al. [43] and Sisodia et al. [60] do not provide source-end DDoS defense. Our system benefits from the advantages of source-end defense discussed in the Design chapter. Hayat et al. [46] and Xu et al. [39] provide source-end defense with drawbacks that our system does not have. Hayat et al. is enforced at the edge of the network, and is susceptible to holes created by cellular and mesh routing. Xu et al. is enforced outside of the source network, resulting in later attack detection and mitigation. The research gap that our system fills is the need for a scalable and flexible defense solution catered toward IoT that mitigates assaults early.

3.2 IoT Intrusion Detection

In this section, we discuss recent work in intrusion detection measures. We focus on IoT use-cases since that is the goal of the mobile firewall system. It is important to keep in mind that the mobile firewall system is not meant to directly compete with IDSes. Our system is a framework that allows multiple IDSes to collaborate in detecting assaults. One of our primary objectives is to provide extensibility for defense against novel attacks. Thus, we implement the mobile firewall to easily incorporate multiple IDSes.

An Intrusion Detection System Against DDoS Attacks in IoT Networks, published by Monika et al. at the IEEE Annual Computing and Communication Workshop and Conference, proposes an IDS using multi-objective optimization and Convolutional Neural Network integrating Long Short-Term Memory deep learning techniques for classification [56]. It was trained and tested on IoT datasets and performs with an F1 score of 99.36%. Monika et al. [56] is an example of a machine-learning based IDS that could be integrated with the mobile firewall system as an attack detector thread to provide robust DDoS detection. One of the main benefits of this research is the use of the convolutional neural network. CNNs allow for the IDS to identify network patterns that may not be evident to other types of IDSes by filtering traffic through pooling and activation layers.

DeepCoin: A Novel Deep learning and Blockchain-based Energy Exchange Framework for Smart Grids, published by Ferrag et al. in the 2020 IEEE Transactions on Engineering Management journal, uses a blockchain-based scheme and a deep learning-based scheme to prevent smart grid attacks [40]. The deep learning-based scheme is an IDS which employs recurrent neural networks. This research is an example of how IDSes can be used to effectively monitor the status of IoT networks. A similar IDS, perhaps consisting of an RNN, may be used as an attack detector thread in the decision module. The blockchain scheme provides added integrity to the network, which is good when considering that machines within may be compromised.

HADES-IoT: A Practical and Effective Host-Based Anomaly Detection System for IoT Devices, published by Breitenbacher et al. in the 2022 IEEE Internet of Things Journal, provides tamper-proof protection and can be deployed on a wide range of Linux-based IoT devices with low overhead [37]. HADES is host-based, meaning that it must be installed on each device to be monitored. We included HADES in the

literature review to identify other examples of IoT intrusion detection that do not rely on strategic placement in the network.

Chapter 4

DESIGN

In this chapter, we explain the design behind the mobile firewall system. We explain the reasons why we investigate source-end DDoS detection and mitigation. Mobility and distribution are two fundamental traits of the mobile firewall system that we examine to fully understand how the mobile firewall system is to be used. We present the threat model and identify security objectives. Design of the mobile firewall system is presented. This is broken down into two parts – design of the security controller, and design of the mobile node.

4.1 Theory and Advantages of Source-End Detection

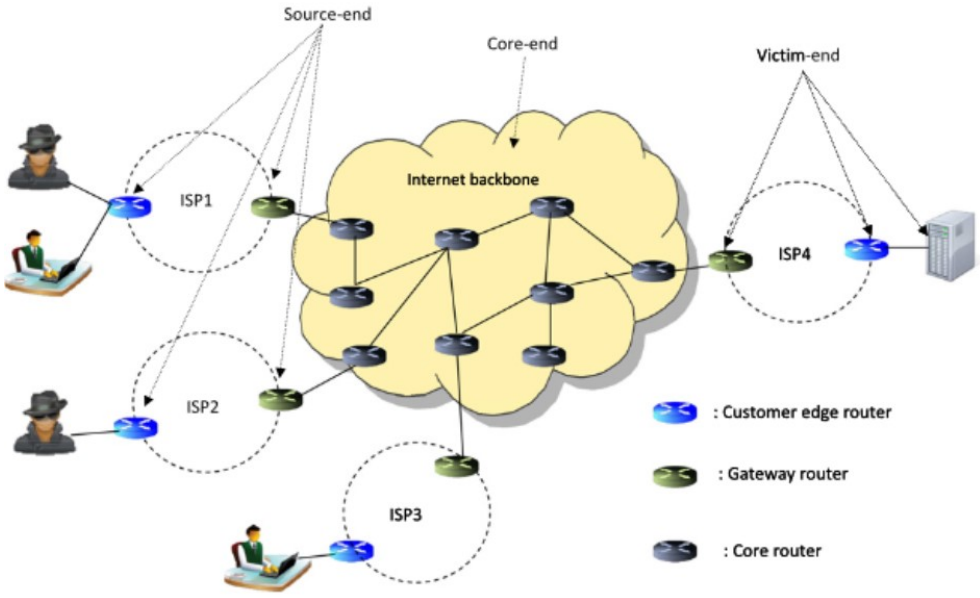

Figure 4.1: DDoS Defense Points [36]

Figure 4.1 details where three critical DDoS defense points lie. Victim-end consists of networks close to the victim server. In-network or core-end describes any number of routers in the Internet. Source-end may describe networks close to the source of the DDoS attack. It may describe defense within the sourcing network itself.

Traditional DDoS detection mechanisms are deployed at the victim-end of a DDoS attack and in-network. Victim-end DDoS detection techniques are simple to implement since the traffic has already reached its target. Characteristics of DDoS attacks are easy to detect at the victim-end, such as spikes in network traffic and unavailability of TCP ports due to a high rate of SYN packets. The disadvantage with victim-end detection is that by the time it is detected, the attack has already impacted the service. Volumetric DDoS attacks have already exhausted hardware resources and congestion to disrupt services. Attacks targeting protocols, such as the SYN flood, have already begun wasting ports. Mitigation is difficult at the victim-end. Performance of the system will degrade because so many resources are allocated to processing packets [22]. Mitigation functions such as blocking IP addresses that are contributing to the attack will help but ultimately, legitimate traffic will be lost.

In-network detection techniques are more complicated to deploy, but they can more effectively mitigate the DDoS attack. DDoS traffic may be caught enroute to the destination and mitigated through the use of offer-based filtering at scrubbing-centers [60]. A benefit of this is that these centers may be deployed across various geographic locations. This helps in mitigating traffic when the source of the traffic is spread out across a vast geographic area, such as in the case of IoT-enabled DDoS attacks. The load of the DDoS will be distributed among different scrubbing centers rather than congesting the victim-end.

In-network detection has the benefit of being able to detect more complicated DDoS attacks, such as the crossfire attack. This is because the DDoS may never be intended

to reach the victim server in the first place. Instead, decoy servers will be targeted, making detection at the source-end difficult [48]. In-network detection remedies this weakness by observing traffic patterns not just at the victim-end, but all around the network.

The third critical vantage point at which DDoS may be detected is the source of the attack. Source-end detection comprises of multiple different methods. One detection mechanism is the use of host-based malware detection. This involves installing security software, such as HADES-IoT, to monitor the behavior of the device itself [37]. If malicious traffic is determined to be generated by the device, then the host-based software will be able to prevent it from leaving the device.

The source-based detection mechanism that the mobile firewall system employs is different. It does not require software to be installed on the source devices. This is a great benefit considering how large IoT networks may scale. Networks may consist of hundreds of devices, and installing software on each may be time consuming. If the installation is completed correctly, then the network administrator must spend time ensuring that all the devices are running the latest software update.

These issues of deployment are solved by the mobile firewall system. The mobile firewall system is designed to monitor the source network, rather than each individual device. This results in a solution that is much more scalable and flexible. Because the detection and mitigation of the DDoS attack is located closer to the source of the attack, this enables the mobile firewall system to detect the attack sooner than upstream. Packets are dropped sooner, which results in less waste of network resources. There will be less congestion, and legitimate packets will be dropped at a lower rate [52]. A larger percentage of DDoS traffic can be blocked because the overall scale and volume of DDoS traffic is more manageable than on the victim-end.

4.2 Theory and Advantages of Mobility and Distribution

DDoS attacks are most commonly enacted by IoT devices [51]. In this section, we discuss source-end DDoS detection and mitigation within the constraints of an IoT network. Typical DDoS defense within a network relies on a centralized gateway that filters all traffic entering and exiting the network. While this defense is viable in traditional networks, it has weaknesses when applied to IoT networks. The first dilemma that may occur in IoT networks is that the gateway may be inaccessible. This is often the case when defending an IoT network that includes nodes connected to the Internet via cellular communications. A 5G base station is inaccessible to the network administrator, thus the administrator cannot deploy security solutions such as an IDS.

The next issue that IoT networks face is the porous nature of IoT routing. Not only may malicious traffic exit the network through cellular connected devices, but it may also exit the network through devices that participate in mesh networks. Mesh networks utilize physical distance to other nodes to route traffic [50]. The issue of traffic exiting the network due to both cellular and mesh routing results in many holes in the traditional network security architecture. The mobile firewall system addresses these two issues by exercising the concepts of mobility and distribution.

Mobility addresses the problem presented by inaccessible gateways. A mobile node has the opportunity to get as close to the gateway as possible. It leverages its physical closeness to the gateway to advertise a strong connection to the Internet for other nodes. In terms of an IoT network implementing RPL routing, this means that the mobile security node sends a DIO message to all its peers in the network [61]. The mobile security node can become the root of the RPL DODAG through legitimate means. Once the mobile node is closest to the gateway, it can then function as the

gateway itself. The mobile node sniffs the traffic that is exiting the network and sends periodic updates to the security controller.

There may be multiple points in the IoT network where traffic can exit, circumventing a traditional network gateway. This problem is solved by the trait of being distributed in the mobile firewall system. If multiple mobile security nodes are available to the security controller, then the security controller may assign a mobile node to monitor the egress point in the network. When there are no available nodes distributed close to the egress point, one of the mobile nodes may use its mobility to its advantage. The mobile node may move physically closer to the egress point and monitor its traffic.

Mobile nodes can be distributed in such a way that all egress points are monitored by mobile nodes. The advantage provided in this scenario is that mobile nodes do not need to be routing the traffic to monitor the egress point. They only need to be in close enough range to sniff the packets. In the case of our RPL routed network, the mobile node will enact a black hole RPL attack to reroute packets for mitigation [35].

4.3 Threat Model

Although the system in this chapter is described as having only one security controller and one mobile node, it is possible for a deployment of the system to include multiple mobile nodes and controllers. We explain the threat model consisting of only one security controller and one mobile node for the sake of simplicity.

4.3.1 Network

The threat model we are considering for the mobile firewall is based on an IoT mesh network. The network consists of multiple IoT devices, which may not be of the same type. There may be different devices that are commonly found in smart-homes – devices such as cameras, smart speakers, or televisions. The devices are connected via the same RPL instance, with the root node at rank zero. Multiple devices may be in range of each other, without being considered RPL parents or children. RPL control messages are broadcasted throughout the network, and the nodes respond according to these messages [61]. We assume that there are multiple egress points for traffic in the network. This may be due to mesh network protocols or cellular communications.

We assume that there is at least one malicious node in the network generating DDoS traffic. There is at least one benign node generating legitimate traffic. We do not assume the class of DDoS traffic this node is generating. There is no centralized gateway in the model, and no way to deploy a centralized IDS that can collect all traffic. The malicious devices degrade network availability using DDoS. This applies to the availability of the network which contains the devices. Degradation of availability also applies to the victim network which is likely in a different subnet across the Internet. Figure 4.2 shows an example of an attacker in our threat model.

We design our threat model in an approach that emphasizes the ability of the mobile security node to stop DDoS traffic before exiting the network. This threat model demonstrates the advantages of effective deployment of our mobile firewall system. The advantages provided by our proposed method are that DDoS traffic is stopped early and there is no requirement for access to a gateway. We assume that the mobile security node can eavesdrop on communications. Even if the communications are

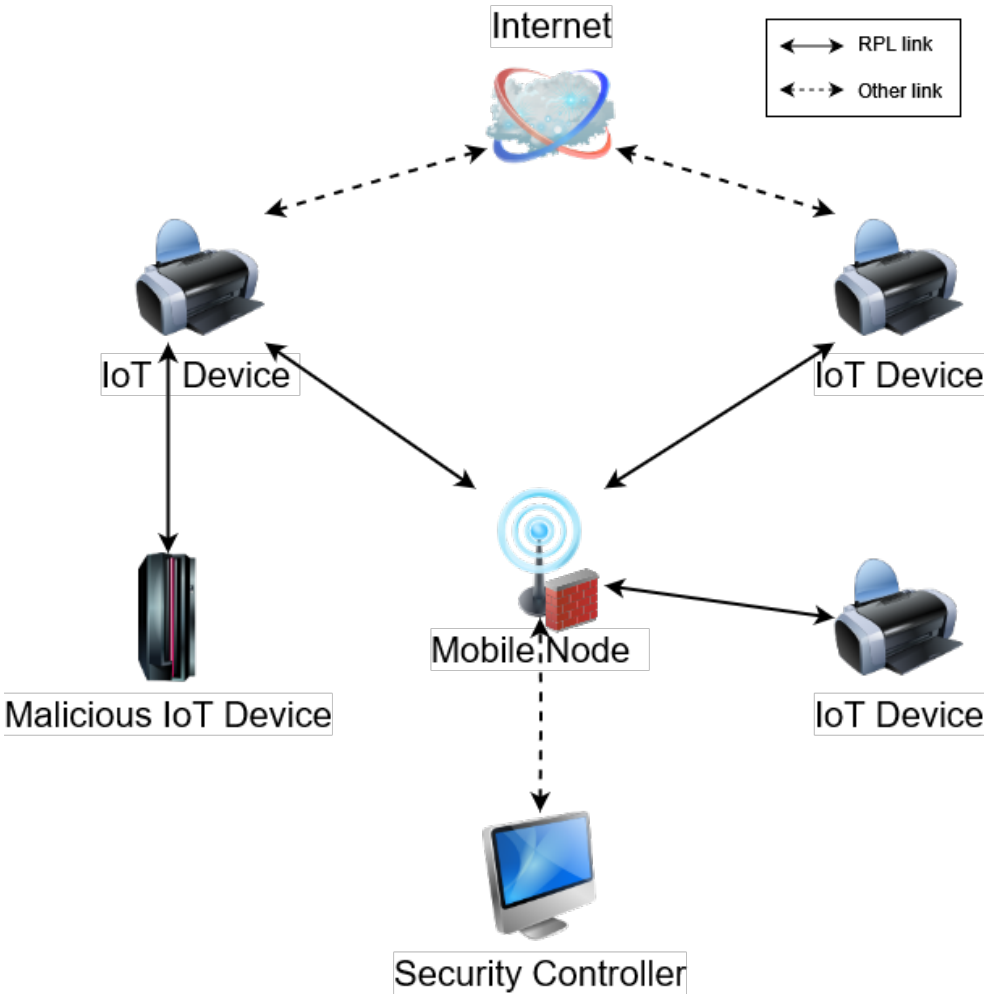

Figure 4.2: Threat Model

encrypted, it is still possible for the mobile firewall system to detect DDoS attacks. In the threat model, the mobile node may be in transmission radius of benign devices or malicious devices. We assume that the mobile node may lose connection to the security controller. Temporary connection loss may be due to leaving transmission radius of the controller to gather data. Batches of data may be collected then sent to the controller periodically.

4.3.2 Distributed Denial of Service

IoT networks are vulnerable to a plethora of different attacks. Some attacks include sinkhole attacks, battery-draining attacks, and voice-command injection attacks [59]. For the purposes of this thesis, we assume that the only attack present in the network is a DDoS attack. The DDoS attack may be implemented through different techniques. We focus on UDP (User Datagram Protocol) flood and SYN (SYNchronize sequence number) flood. The malicious nodes are the only nodes in the network generating DDoS traffic. The benign devices, mobile node, and security controller do not contribute to the attack.

Our threat model makes the assumption that authentication may be compromised. We recognize that bots can easily spoof their IP address and MAC address. Because of this, we assume that the security controller uses detection methods that rely on the source IP address of traffic. That way, legitimate traffic from an IP address may be blocked from reaching the Internet. DDoS connections from that same address will be dropped as well. The convenience of having per-packet intrusion detection is one that may not be present in every network. Thus, we make no assumptions about allowing legitimate traffic to pass while the network is under the influence of a DDoS attack.

The mobile firewall system does not require the mobile node to be connected to the same network as the DDoS attackers. Thus, it may not be possible to target the mobile node in a DDoS attack. In our experiments however, we assume that the mobile node is a part of the same RPL network that the DDoS nodes are a part of. The mobile node may be a target of the DDoS attack. It is possible to overwhelm the mobile node, at which point it may be unable to effectively capture all traffic for

processing. In this thesis, we assume that the security controller is not part of the network and therefore not a potential DDoS target.

4.3.3 Countermeasures

The mobile security node and security controller are capable of enacting countermeasures within the network. These countermeasures are used for the purposes of DDoS detection and mitigation. The first countermeasure is eavesdropping. Eavesdropping is the act of intercepting and listening to communications between two nodes without their consent [42]. The goal of eavesdropping is typically to reduce the confidentiality of the network. The mobile security node leverages eavesdropping to gain a wider context on the behavior of the network. Eavesdropping can be conducted through a variety of techniques, with the most common network sniffing being packet capturing. Packet capturing involves monitoring network traffic to intercept communications that are transmitted over the network. We assume the mobile node has the ability to packet capture. This can be achieved through the use of a wireless capture card in monitor mode. It may also use an appropriate software, such as Tcpdump, to save these packets in the form of PCAP files. It is still possible for the mobile firewall system to detect DDoS attacks even if the communications are encrypted.

The second countermeasure that the mobile security node utilizes is selective forwarding. Selective forwarding is a type of routing attack that involves allowing some packets to travel to their next hop, while denying others from reaching their destination [63]. If the attacker can approximate the contents of the packets, they can choose to forward non-critical data. They can choose to drop critical data at the same time. The ultimate goal of the attack is to compromise availability of the network. In this threat model, the mobile node uses selective forwarding to defend the network. The mobile node protects the network by dropping all packets originating from malicious

nodes. The dropping of malicious packets results in an increase in availability for legitimate traffic. This specific instance of a selective forwarding attack is known as a black hole attack. A black hole attack is when a router refuses to forward any traffic from a blacklisted source, no matter if the data is considered critical or not.

4.4 Security Objectives

The target attack surface of the mobile firewall system is network-diffused threats and specifically for this thesis DDoS. We consider availability and IP spoofing objectives for the mobile firewall system in this thesis.

4.4.1 Availability

The primary goal of the mobile firewall system is to detect and mitigate DDoS traffic at its source. DDoS attacks impact the availability of the network by preventing machines from servicing requests. During a DDoS attack, a large number of compromised devices flood the target with traffic, overwhelming its capacity to process requests and causing it to become slow or unresponsive. This can lead to service disruption or even complete downtime, making it impossible for legitimate users to access the resource. DDoS attacks can also consume significant amounts of network bandwidth, which can affect the performance of other network resources and cause them to become unavailable.

On the contrary, the mobile security node uses selective forwarding to open up bandwidth to legitimate users. This is achieved by degrading the availability of the network for the malicious nodes. All traffic from the malicious nodes is dropped in the event of a black hole mitigation [63]. The objective of availability is preserved for benign

machines thanks to the selective forwarding countermeasure. Availability applies to both devices within the source-end IoT network, as well as the victim-end network. Availability is improved for both environments.

4.4.2 IP Spoofing

One objective of the mobile firewall system is to protect against IP spoofing. To protect against IP spoofing means to ensure that all data is originating from their authentic sources. We focus on integrity in the context of authentication. Packets originating from a malicious device should be determined as malicious and packets from legitimate devices should be determined as benign. DDoS attackers usually use spoofing to conceal their identity. It is the objective of the security controller to uncover which nodes are spoofing their data.

Spoofing has its consequences. It may result in legitimate traffic being blocked, and it makes DDoS detection more difficult. The security controller can communicate with the mobile node to request telemetry data about the network. Through analyzing this data, the attack detector threads within the security controller's decision module will be alerted of a malicious node present. An objective of the security controller is that it can enact security functions even when network authentication is reduced.

4.5 Mobile Firewall System Design

The system model is described as follows: There are two separate entities within the mobile firewall system. One is the mobile security node. The mobile security node collects traffic within the network, stores it, and sends it to the security controller for processing. The mobile node receives security functions from the security controller to

enforce within the network. Security functions include intrusion mitigation techniques such as black hole routing.

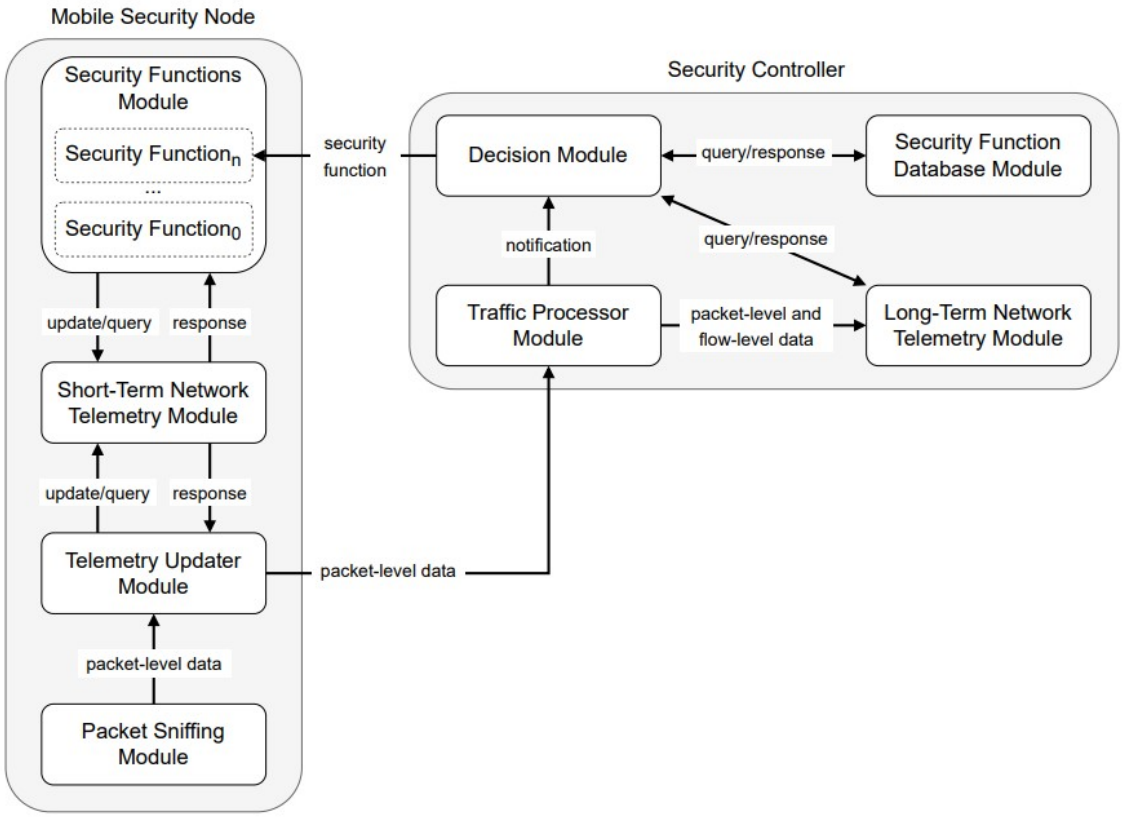

Figure 4.3: Mobile Firewall System Diagram [59]

The other entity within the mobile firewall system is the security controller. The security controller receives traffic data from the mobile node, stores it, and processes it. Processing can include intrusion detection and anomaly detection. The security controller decides what security function the node should enforce based on the traffic data and sends this function to the mobile node. Figure 4.3 depicts an overview of the mobile security node and the security controller.

Although the system in this chapter is described as having only one security controller and one mobile node, it is possible for a deployment of the system to include multiple mobile nodes and controllers. We explain the design of the system consisting of only one security controller and one mobile node for the sake of simplicity.

4.5.1 Security Controller

The security controller consists of four main components – the traffic processor module, long-term network telemetry module, security function database module, and decision module. The purpose of the traffic processor module is to receive and manage packet-level or flow-level data that is being transmitted by the mobile node. Packet-level data in the form of PCAP files is converted to flow-level data in the event that the decision module requires flow data. The traffic processor module notifies the decision module when new data has arrived to keep DDoS detection as up to date as possible. The traffic processor module hands off its data to the long-term network telemetry module once the transmission from the mobile node is complete.

The long-term network telemetry module saves the data that is passed to it from the traffic processor module. The data is stored on the security controller and may be stored in a local database. The long-term network telemetry module is always ready to process requests from the decision module. The decision module may send the long-term network telemetry module queries for data from earlier in the lifetime of the network. The queried data may be useful to detect long-lived attacks and stealthy attacks such as low-rate DDoS or pulsing DDoS. Recognition of stealthier attacks may require network pattern observation over an extended period of time. An advantage of the long-term network telemetry module is that the packets do not need to be stored in memory (as they are in the decision module). The data can be stored on disk. Disk storage is a great benefit that prevents the security controller from exhausting its memory while powered on. We assume that the security controller is not resource constrained, therefore it is capable of storing long-term outlooks on network patterns.

The security function database module is responsible for storing security functions. Security functions may be classified as reconnaissance tasks, such as gathering more network traffic for extra extensive inspection. Security functions may also be classified as mitigation tasks. Mitigation tasks are countermeasures that aim to reduce the threat of attacks. In terms of RPL routing, a mitigation security function may be to initiate an RPL black hole attack against malicious nodes. The security function database module should be able to store all mitigations and reconnaissance tasks for all threats that the network may encounter. Based on the threat model, the security function database module stores functions for the purposes of DDoS mitigation because no other threats are assumed to be present within the network.

4.5.2 Decision Module

The decision module is the component within the security controller that provides DDoS detection. It does so by deciding which security functions to select. Figure 4.4 depicts an overview of the decision module. The decision module is comprised of three key threads. First is the detection initiation thread. The detection initiation thread receives data from the traffic processor module and queries data from the long-term network telemetry module for older data when needed.

Data is then sent to the attack detector threads. Multiple attack detector threads run concurrently for granular intrusion detection. The attack detector threads are the core of the decision module. These threads use the traffic data to determine whether an attack on the network is occurring. The threads may also return inconclusive results, requiring more data to be collected to make a confident decision. This knowledge of the network is expressed through values known as detection scores. The benefit of having multiple attack detector threads is that each thread can be configured to detect a particular attack. Multiple threads can be used to cater towards different protocols

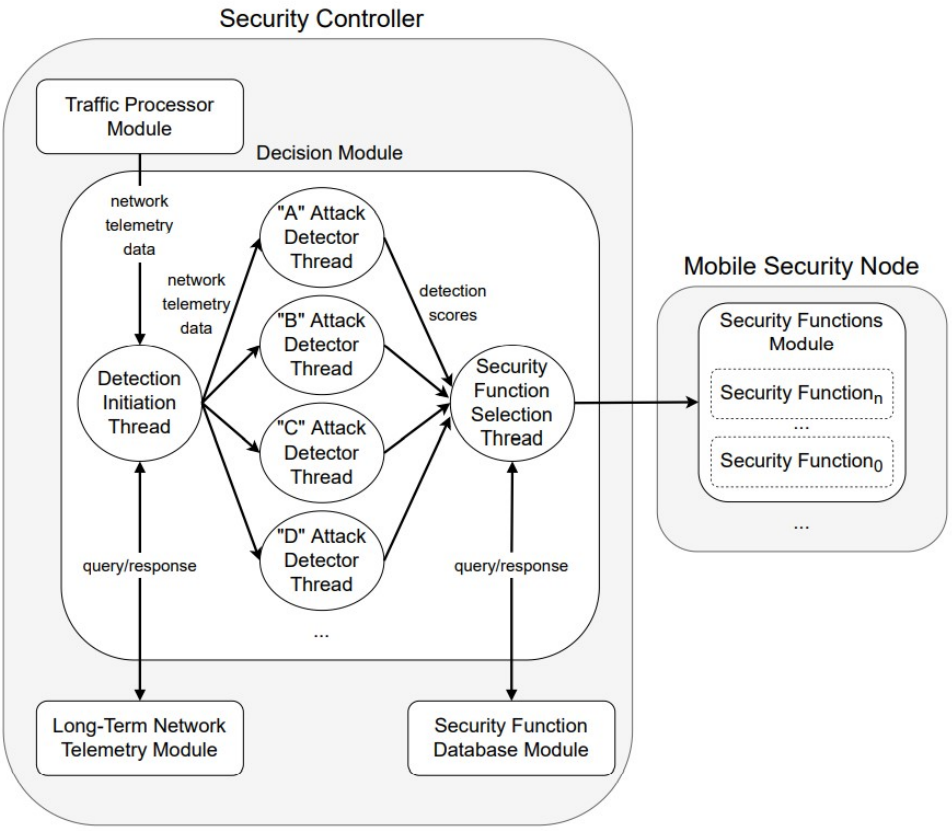

Figure 4.4: Decision Module Diagram [59]

or intrusion detection systems, resulting in great flexibility for mobile firewall system users. Flexibility is of particular importance in the field of network security since machine learning and other technologies are being developed for intrusion detection purposes.

Detection scores are aggregated by the security function selection thread to determine the appropriate actions to take. If results are inconclusive, the function selection thread will request the mobile node for more data. The mobile node will then perform extra reconnaissance tasks. When attacks are identified within the network, the function selection thread will query the security function database module for a security function to send to the mobile node. The mobile node enforces these security functions actively within the network.

4.5.3 Mobile Security Node

The mobile security node has four main components – the packet sniffing module, telemetry updater module, short-term network telemetry module, and security function module. The purpose of the packet sniffing module is to collect all packets and log the patterns of the network traffic. This requires the mobile security node to be equipped with a wireless interface card. The card must support monitor mode to eavesdrop on all traffic [64]. Once the traffic is collected, it is passed to the telemetry updater module.

The telemetry updater module works with the short-term network telemetry module to record the immediate status of the network. Traffic is saved to the short-term network telemetry module once it receives data from the packet sniffing module. The telemetry updater module is also responsible for receiving data from the short-term module. It queries data to send to the security controller periodically.

The short-term network telemetry module stores data within the mobile node for a small period of time. This is because it is assumed that the mobile node is resource constrained. The mobile node may not have enough storage to hold a complete view of network behaviors over a prolonged period of time. The short-term network telemetry module works with the telemetry updater module to unload PCAP files to the security controller for long-term storage. It can then delete the files that it does not need, freeing up space for new data.

The purpose of the security functions module is to execute and enforce security functions within the network. The security controller sends the security function module countermeasures to deploy. The countermeasures are stored in the security function module and can be enacted at any time. After a certain period of time defined by the security controller, the mobile node will stop enforcing a particular security function.

The enforcement ends when the security controller has presumed that the attack is finished. The security functions module has multiple threads of security functions running in parallel to mitigate multiple attacks at the same time. A benefit of having the short-term network telemetry module is that the security function module can query it for information about the network that may be important to the security functions.

4.5.4 Mitigation Functions

We focus on a black hole attack to counter DDoS in the example of a mesh IoT network routed through RPL. Other mitigation functions may be added to the security functions module based on the type of attacks present. In RPL routing, a black hole attack is a type of attack in which a malicious node advertises itself as having the shortest path to the destination node, and then drops all data packets that it receives. As a result, all traffic that is intended for the destination node is lost, effectively creating a "black hole" in the network.

The black hole attack is conducted by intentionally modifying the RPL routing table in the mobile node to advertise false routing information. The mobile node specifically crafts false DIO messages that include a low rank. The rank of the DIO advertised must be lower than the rank of the node that is forwarding malicious traffic. The malicious node will identify that the mobile node has a shorter connection to the victim and switches its route accordingly. No use of selective forwarding (which involves forwarding only certain types of data packets while discarding others) is required since we assume all traffic originating from the malicious node is malevolent.

Figure 4.5 shows an example of the RPL black hole mitigation in action. Legitimate traffic is portrayed in solid lines, DDoS traffic is portrayed in dashed lines, and the

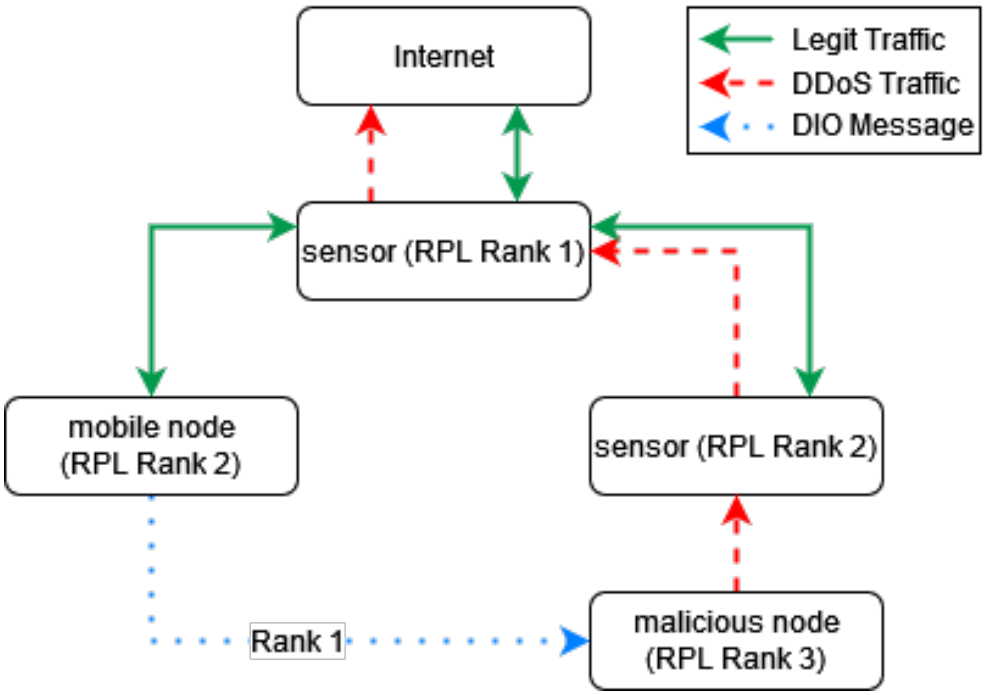

Figure 4.5: DIO Mitigation Example

DIO message is portrayed in a dotted line. In this example, the malicious node (of RPL rank three) is generating DDoS traffic. The sensor of RPL rank two is forwarding the DDoS traffic to the sensor of rank one. The DDoS traffic is exiting the network and entering the Internet through the rank one sensor. The mobile security node (of RPL rank two) initiates the RPL black hole mitigation by sending the malicious node a DIO message. The DIO message advertises the mobile node as rank one.

Since the DIO rank is lower than the malicious node's current parent rank, it begins routing traffic through the mobile node. Figure 4.6 shows the effect of the black hole mitigation. The mobile node receives all traffic from the malicious node and discards it. DDoS traffic is no longer being routed through the network and the attack has been mitigated. It is important to note that the malicious node now observes the mobile node as its RPL parent. The rest of the network may not acknowledge the change in RPL ranks if the DIO message was not sent to them. DIO messages are required to be broadcasted to the network. In order to preserve the routing policy of

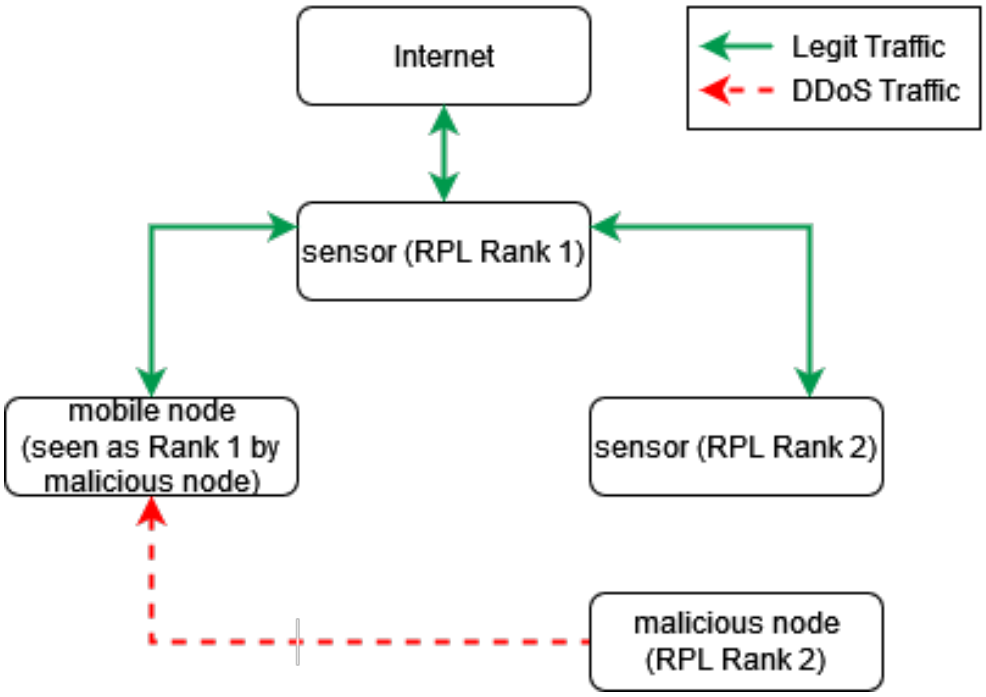

Figure 4.6: Black Hole Diagram

the network while initiating the black hole, the DIO message must be sent out to the malicious node without being broadcasted. A consequence of this may be that the network continues to treat the mobile node as RPL rank two, while only the malicious node treats it as rank one.

Chapter 5

IMPLEMENTATION

In this chapter, we discuss the implementation of the mobile firewall system. We explain the reasons why we chose the components for the security controller. Emphasis is placed on the implementation of the decision module. We present the system components in depth, with explanations as to why we have chosen them. Discussion of the attack detector threads is presented and how they relate to DDoS defense. The implementation of the security functions is explained. Contributions to the RPLD library are presented and explained. The system components for the implementation of the mobile security node are discussed.

5.1 Security Controller

The security controller consists of four main parts – the traffic processor module, the long-term network telemetry module, the decision module, and the security function database module. The traffic processor module is implemented through the use of SCP. SCP (Secure Copy) is a protocol used for transferring data securely over a network or networks. It provides a secure way to transfer files between systems, using an encrypted connection to protect the confidentiality and integrity of the data being transferred [53]. Another alternative, SFTP may be used but we chose SCP for its ease in automating tasks [49].

The use of SCP allows us to achieve our security objective of integrity through the mobile firewall system. SCP is based on the SSH (Secure Shell) protocol, which provides a secure way to access remote systems and execute commands [30]. The

advantages of SCP over other protocols are strong encryption, data integrity protection, and support for authentication mechanisms. It also supports file transfer resumption, allowing interrupted file transfers to be resumed from where they left off. These traits of SCP make it a excellent choice for a protocol to use. The utility of file transfer resumption is of particular importance in the case of DDoS defense since DDoS impacts the availability of the network.

SCP is installed on both the mobile security node and the security controller to establish a connection. SSH keys are used to manage authentication between the security controller and mobile node [53]. The security controller receives packet-level data from the mobile node using the SCP connection. This effectively acts as the traffic processor module for the system. A notification to the decision module is not necessary because of how the decision module is implemented. The decision module checks the save directory for the latest PCAP file. If the latest PCAP file in the directory is the same as the latest PCAP file processed, the decision module will continue to check for new files.

The long-term network telemetry module is implemented as a directory on the security controller. This directory is never changed, because the traffic processor module must always update the same directory and the decision module must always check the same directory for new files. We do not use a database in the implementation of the long-term network telemetry module because we assume that the security controller is not constrained in its disk resources. Another reason we do not use a database is because there is only one mobile security node in the model. The traffic data is easy to manage because all data originates from one mobile node, thus a database is not required.

The security function database module is implemented with similar reasoning to the long-term network telemetry module. The threat model assumes that the only attack

faced by the network is DDoS. It assumes that devices in the network are routed through RPL. This results in the need for a single security function. Thus, we opt not to use a full-fledged database for simplicity. We store the RPL black hole mitigation script in a file on the security controller. The security function is transmitted to the mobile node through methods similar to the traffic processor module (using SCP) [53].

5.2 Decision Module

The mobile firewall system is developed in the Go programming language and features the integration of multiple industry standard tools. Such tools include the Wireshark packet sniffer, the Snort intrusion detection system, the Fastnetmon intrusion detection system, and more. The Python programming language is used for the implementation of the security function.

5.2.1 Detection Initiation Thread

The decision module's threads are implemented using the Go programming language [12]. Go's clarity and readability are one of its main benefits. It contains clear syntax that is simple to understand and lowers the chance of mistakes. Furthermore, Go is quicker than interpreted languages like Python or Ruby since it is compiled. It features built-in concurrency support, enabling us to create code that is scalable and effective for multi-core machines. Go's robust type system, which detects problems early in the development process, is another benefit. The popularity of Go has grown recently, and the big community behind it means that we can access libraries and resources to aid with development of the mobile firewall system [13].

The built-in concurrency support that Go offers is its most significant benefit. The "go" keyword is used to generate goroutines, lightweight threads which are controlled by the go runtime [10]. It is simple to build a large number of them without negatively affecting performance since they are far less expensive to produce and manage than conventional threads. Goroutines are important in the context of the mobile firewall system because the decision module requires a detection initiation thread, security function selection thread, and multiple attack detector threads.

In addition to goroutines, Go also provides a set of primitives for sharing data between them, including channels and mutexes. Channels are used to pass data between goroutines in a safe and controlled way, while mutexes are used to synchronize access to shared data [10]. These primitives are simple to use; they help to ensure that decision module threads are free from race conditions and other common concurrency-related bugs. The advantages of channels and mutexes are important to the decision module since data must be passed between all three types of threads.

The detection initiation thread manages the attack detector threads and security function selection thread. All threads are implemented as goroutines. Network telemetry data is passed around goroutines using channels. The number of responses received on the shared channel is used to manage when goroutines have terminated. The attack detection is complete when the number of detection scores received on the channel is equivalent to the number of attack detector goroutines spawned.

5.2.2 Attack Detector Thread

One of the great benefits of the mobile firewall system is its flexibility. The goal is for any user to plug in an off-the-shelf component to serve as an attack detector thread. This extensibility allows the mobile firewall system to utilize the latest DDoS

detection and mitigation mechanisms. Two tools are chosen to be the basis of attack detectors threads in the mobile firewall system.

5.2.2.1 Snort

The first tool used is the Snort intrusion detection system maintained by Cisco. Network traffic patterns that match well-known attack signatures can be found by Snort using a rule-based detection approach. When Snort discovers a possible threat, it can produce alarms, log the relevant data, and even restrict traffic to prevent more attacks [32].

The ability to be very customized is one of Snort's best features. This flexible tool may be customized to the unique requirements of the network since users can add their own rules to identify specific sorts of threats. We use the Snort community DoS/DDoS rules and additional rules from Milad Soltanian [7]. A comprehensive security solution may be delivered by combining Snort with other cybersecurity tools, such as firewalls [32]. Real-time traffic analysis makes Snort a useful tool and is another one of its primary features for detecting and preventing DDoS attacks.

We leverage the flexibility of Snort by building multiple configuration files. We use different data processing rules for each configuration and run Snort as its own thread for each distinct configuration file. This allows Snort to detect diverse types of attacks occurring in the network – with an emphasis on DDoS detection. We decided to implement the attack detector threads using Snort because of its flexibility in rules, popularity in the cybersecurity realm, and wide community support. Although it is not a dedicated DDoS detection system, Snort provides a reliable source of detection because of its real-time capabilities and support for configurations with DDoS rules.

5.2.2.2 FastNetMon

The second DDoS detection tool we chose to form the basis of the attack detector threads is FastNetMon, developed by FastNetMon LTD Company. Fastnetmon is made to find and stop different kinds of DDoS attacks on a network. FastNetMon is made to be quick and effective, and it can recognize intricate DDoS attacks. It is capable of detecting a variety of DDoS attacks, including volumetric, protocol, and application layer attacks [55]. In order to lessen the effects of an attack, FastNetMon can automatically block the attacking IP addresses or reduce their traffic once an attack has been identified.

One of the key features of FastNetMon is its ability to detect DDoS attacks in real-time. This allows network administrators to respond to an attack quickly and prevent it from causing severe damage. Real-time detection is possible in part because of the way FastNetMon processes traffic. FastNetMon uses Netflow formatted data for processing. Netflow contains flow-level information about the network, allowing DDoS to be detected with less data than packet-level PCAP captures. The benefit of this is that FastNetMon can process flow-level data at a much faster rate than other intrusion detection systems process packet-level data [55]. FastNetMon can also be configured to send notifications when an attack is detected, allowing network administrators to take action immediately.

The attribute that both Snort and FastNetMon share is that both programs send notification alerts when attacks are detected. Figure 5.1 shows an example of these alerts from Snort. The alerts contain valuable information such as the classification of attack, protocol, and associated IP addresses. The attack detector threads parse these notifications using regular expressions. The source of the DDoS attack is reported by both Snort and FastNetMon. The IP addresses are reported to the security function

selection thread with full confidence – a 100% detection score. We recognize the confidence to be perfect since there is no way to determine a confidence level from either Snort or FastNetMon alerts. A study of the internals of the IDSes may reveal how to extract confidence levels, but that is not necessary for the purposes of this thesis.

```
12/13-12:33:26.015918  [**] [1:1852:3] WEB-MISC robots.txt access [**] [Classifi
cation: access to a potentially vulnerable web application] [Priority: 2] {TCP}
162.158.146.39:47780 -> 198.71.247.91:80
12/13-12:46:57.691950  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 104.244.72.81:5
3357 -> 198.71.247.91:1900
12/13-13:05:02.317868  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 104.244.74.242:
42890 -> 198.71.247.91:1900
12/13-13:19:21.907178  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Attem
pted Information Leak] [Priority: 2] {ICMP} 222.135.14.151 -> 198.71.247.91
lorenzo@lorenzo-VirtualBox:~/Documents/firewall$
```

Figure 5.1: Snort Alert

5.2.3 Security Function Selection Thread

RPLD is the RPL implementation that Mininet supports for RPL routing [41]. Upon experimentation, we discovered that the RPLD library was incomplete. It did not support correct processing of DIO messages. In this section, we discuss what changes were made to the RPLD library to process DIO messages, and how DIO messages are used to implement an RPL black hole security function.

5.2.3.1 RPLD Additions

RPLD is an open-source RPL implementation in C for Linux machines developed by Alexander Aring and maintained by UFRN professor Ramon Fontes [41]. The implementation supports manipulation of IPv6 routing tables and scheduled RPL messages. When experimenting with DIO messages to initiate an RPL black hole

attack, we found that the messages were not being processed correctly. In the experiment, we sent a falsified DIO message advertising root rank zero from a node that was not the root of the DODAG. The receiving node of the message did not update its routing tables to become the child of the node falsifying its rank. Upon further inspection of the source code, we discovered that the DIO messages were not being processed in a way that allowed the routing tables to change.

RPLD uses the Linux netlink libraries to communicate with the kernel for changes to the routing tables [28]. The benefit of using netlink is that system calls are not required, resulting in great application portability. The first addition to RPLD we implemented was a method that uses netlink to delete default routes from the routing tables. This function was a necessary addition because leaf nodes must be able to change the parent node that their traffic is routed through. Methods already exist in the RPLD source code to add default routes, as well as manipulate via routes.

Code was added to manage the case in which a DIO message arrives at the node advertising a lower rank than the current parent has. We made the addition to check the rank of DIO messages when incoming RPL messages are being processed. When an appropriate DIO message arrives, it enters the next method we added. The next method we added is a function to process the DIO message correctly. First, the route to the parent is deleted if the DIO rank is lower. The node's parent in the DODAG struct is updated to be the sender of the DIO message. The new route to the new parent is updated in the routing tables.

Pseudo code for the methods we added to RPLD is summarized in Algorithm 1.

Algorithm 1: RPLD DIO Processing

Result: Updated DODAG struct and routing tables

```
Incoming DIO messages;
while DIO messages arrive do
    if DIO Rank < Parent Rank then
        Delete old parent route;
        Update DODAG struct parent rank;
        Update DODAG struct self rank;
        Add new parent route;
    else
        Process next message;
    end
end
```

5.2.3.2 RPL Black Hole

We implemented the RPL black hole mitigation using a networking tool called Scapy. Scapy is a Python-based interactive library and software for packet manipulation [33]. Scanning, sniffing, and forging packets are just a few of the various network-related jobs that Scapy may assist with. It may be used to evaluate the security and dependability of network systems and supports a broad variety of network protocols, including TCP, UDP, ICMP, and DNS.

The Scapy script requires knowledge of the offending device. First, the security controller must uncover the source of the DDoS traffic. In the context of an RPL routed 6LoWPAN network, the mitigation requires the attacker's link-local IP address. The link-local IP address is required so that the forged DIO message can be sent directly to the malicious node, rather than broadcasted to the network. We prefer the forged

DIO message to unicast because we do not want to disrupt the routing policy of the network. Figure 5.2 shows the required fields of a DIO base object. The script requires the ID of the RPL DODAG. The DODAGID is required for all messages intended for the specific DODAG targeted [61]. If the DIO DODAGID does not match the DODAGID of the node's DODAG struct, then the message will not be processed.

Most importantly, we set the rank of the forged DIO message to be lower than the rank of the malicious node's parent. This causes the malicious node to recognize the mobile node as its parent. The security controller sends the mobile node the security function through SCP after all the required information is collected. Once the traffic begins routing through the mobile node, it can then be discarded to execute the black hole attack on the attacker. The process of baiting the attacker and then dropping its traffic is the RPL black hole mitigation against DDoS.

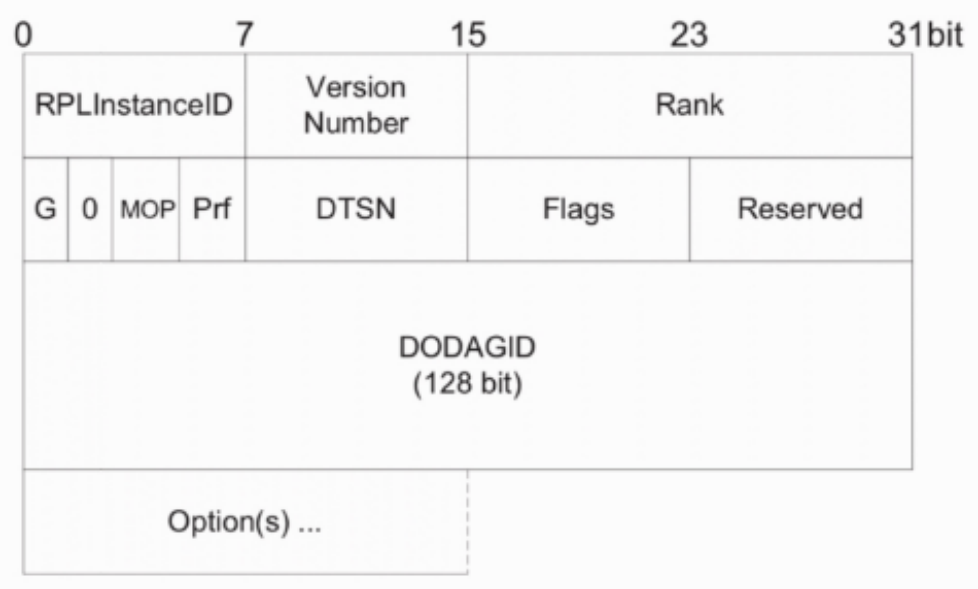

Figure 5.2: DIO Base Object [24]

5.3 Mobile Security Node

The four primary components of the mobile security node are the packet sniffing module, short-term network telemetry module, telemetry updater, and security functions module. In this section, we explain how each of these modules are implemented, and how they integrate into the overarching mobile firewall system. We utilize third party tools including Wireshark, SCP, and iptables.

5.3.1 Packet Sniffing Module

The packet sniffing module of the mobile node is implemented using Dumpcap. Dumpcap is a command-line packet capture tool that is part of the Wireshark network protocol analyzer. It is designed to capture network traffic and save it in a PCAP format file that can be analyzed later using Wireshark or other tools [8]. Dumpcap is available on multiple platforms, including Windows, macOS, and Linux, and it can be used to capture packets from a wide range of network interfaces. Support for many network interfaces is important in the context of the mobile firewall system since one of the goals is flexibility. The mobile firewall system must support any physical-layer wireless protocol carrying DDoS.

Dumpcap's simplicity and flexibility are one of its main benefits. With a variety of capture filters, including port numbers, IP addresses, and protocols, it enables users to capture packets. Additionally, it supports a variety of output file formats, such as PCAP, PCAPNG, and even plain text. Dumpcap can be used to record traffic both online and off. In offline mode, it reads packets from an existing PCAP file while in live mode, it captures packets in real-time as they are transmitted over the network. The mobile firewall system performs reconnaissance using Dumpcap in live mode.

Dumpcap's ring buffer mode is a feature that allows it to capture network traffic under storage constraints. In this mode, Dumpcap captures packets and writes them to a file in a circular buffer, overwriting old data as time progresses [8]. The ring allows for continuous capture of network traffic, even if the capture buffer becomes full. With ring buffer mode, the mobile firewall can set a maximum size for the capture buffer and configure Dumpcap to start overwriting old data once the buffer is full. This means that the most recent network traffic will always be available for usage by the security functions module, regardless of how long the observation has been running.

The mobile firewall implementation sets both a maximum size for the capture buffer, as well as a maximum allowed number of seconds. Capturing of a single PCAP file ends when either the time limit is reached, or when the maximum file size is reached. Each new PCAP file is then managed by the telemetry updater module. We set the time limit to three seconds and maximum file size to 30MB unless otherwise specified. These limits are imposed to manage an average traffic throughput of one megabit per second.

5.3.2 Short-Term Network Telemetry Module

Because Dumpcap manages both the capturing of traffic and the saving of traffic into PCAP files, we implement the short-term network telemetry module as a directory on the mobile node. We assume that the storage of the mobile node is constrained, therefore we use Dumpcap's ring buffer mode to preserve disk. This implementation is similar to that of the security controller's long-term network telemetry module. We use a directory in this case because we assume that all reconnaissance data is bound for the single security controller. A database may be required in an implementation where the mobile node communicates with multiple security controllers.

5.3.3 Telemetry Updater

The telemetry updater is implemented using SCP [53]. It must be implemented with SCP to maintain a stable connection with the security controller. SCP transfers data to the security controller whenever Dumpcap creates new PCAP files. The mobile security node shares an SSH key with the security controller to maintain the integrity of authentication. Multiple SSH keys may be required in mobile firewall implementations with multiple mobile nodes and security controllers. See Section 5.1 for more details about SCP and SSH authentication.

5.3.4 Security Functions Module

Since there is only one DDoS mitigation, we have one security function thread for the purposes of this thesis. We use the networking application iptables to implement the security functions module. Iptables is a firewall utility program that is included in most Linux distributions. It is used to configure and manage the network packet filtering rules in the mobile firewall system. Iptables filters and modifies network traffic to regulate which routed packets are permitted to enter or leave a system [20]. It operates by upholding a set of chains – organized rules.

Each chain contains a set of rules that apply to ingress or egress network traffic. The rules define what kind of traffic should be allowed or blocked based on various criteria, such as the source and destination IP address, port number, or protocol type. Iptables can be used to set up a variety of firewall configurations – from simple configurations that block or allow specific ports, to complex configurations that include network address translation, port forwarding, and traffic shaping. Iptables is used in a simple configuration for the mobile firewall's DDoS mitigation.

The security function received from the security controller contains a list of iptables rules to add to the chain. These rules are determined by the attacker IP addresses discovered by the decision module. One of the assumptions we make in the threat model is that all traffic originating from an attacker node is DDoS traffic. Because of this assumption, we do not have to worry about setting up complicated iptables chains. All traffic generated by an attacker IP address is blocked and prevented from routing through the mobile node.

It is possible to set more complex rules using iptables. A rule based on protocol may be used if we do not assume that all traffic from an IP address is DDoS [19]. The specification of a single port or protocol may be helpful in the event that a legitimate node is blocked. A legitimate node may be blocked if attackers spoof their IP address. For example, a malicious node generates volumetric UDP DDoS traffic under the disguise of a legitimate node. The legitimate node is transmitting TCP traffic. A new rule would allow legitimate TCP traffic to pass while denying UDP traffic.

Chapter 6

EVALUATION

In this chapter, we explain the tools used for evaluation. We discuss potential network simulators and elaborate on why Mininet is ultimately chosen. We explain how the simulation is set up to reflect a real IoT network. Three case studies are introduced to evaluate three use-cases of the mobile firewall system.

6.1 Ethical Statement

All network attacks are done purely for academic purposes. All tests are done in an isolated network which we own. No privacy was violated making this report.

6.2 Network Simulators

In this section, we study different network simulators that can provide us with a realistic environment for evaluating the mobile firewall system. Four of the ideal qualities of our network simulator are easy to set up, support for IoT sensor nodes, support for wireless mesh protocols, and granular detail. Granular detail means that we would like full control over the network devices. We should be able to measure the exact impact of each device – whether it be viewing the exact packets leaving the device, the applications running on the device, or some other state of the device requiring exact knowledge.

6.2.1 GNS3

GNS3 is a network emulator and virtualization software that allows users to design and simulate networks using virtual appliances [11]. With GNS3, users can create network topologies by dragging and dropping virtual devices, such as routers, switches, and firewalls, onto a canvas. They can then connect these gadgets together and configure their settings, just like they would in a real network. GNS3 also supports a wide range of network operating systems, including Cisco IOS, Juniper Junos, and VyOS, among others. This allows users to simulate different network environments and evaluate their configurations before deploying them in a production environment.

Comparing GNS3 to other network simulation tools reveals both its strengths and weaknesses. It is incredibly adaptable and may be used to simulate a variety of network topologies and devices, including virtual machines and appliances. It has great potential for evaluating the mobile firewall system because to the variety of devices. Specifically, simulations are more accurate when using sensor-type devices. GNS3 is an open-source tool that can be extended with plugins and add-ons. This enables the tool's functionality to be modified or new features to be added.

GNS3's potential resource requirements are one of its disadvantages, especially when emulating large and complex network topologies. To run GNS3 efficiently, users may need powerful hardware. The learning curve is another drawback. GNS3 has a steep learning curve particularly for people who are unfamiliar with virtualization technology. We want something simple to utilize for evaluation reasons and we do not want to worry about our hardware being unable to run GNS3. We decided not to use GNS3 for the evaluation due to these disadvantages.

6.2.2 OMNeT++

OMNeT++ (Objective Modular Network Testbed in C++) is a discrete event simulation framework that is primarily used for network simulation and research [29]. It is a versatile tool that can be used to simulate a wide range of network technologies, including wired networks, wireless networks, and cloud computing environments. OMNeT++ has a large and active user community, which provides documentation for unaccustomed users. This trait is important for us because 6LoWPAN may not be easy to set up in OMNeT++. Simulators with more documentation tend to be easier to use.

OMNeT++ has similar disadvantages to GNS3. One characteristic they share is a steep learning curve. OMNeT++ has a steep learning curve, especially for users who are not familiar with network simulation in C++ programming. It can be resource-intensive, especially when simulating expansive and complex network topologies. Similar to GNS3, we do not want to worry about having powerful hardware to run OMNeT++.

The lack of a graphical user interface for modeling and simulation in OMNeT++ is another drawback. Users must instead write C++ code to create models and simulations. This is a disadvantage when compared to the graphical tools that GNS3 provides. We decided against using OMNeT++ for the evaluation due to the learning curve and resource requirements.

6.2.3 Contiki Cooja

Contiki Cooja is a network simulator that is used to simulate and evaluate wireless sensor networks (WSNs). It is a part of the Contiki operating system, an open-source

operating system created for WSNs and IoT devices [17]. Users can more easily create and assess various WSN configurations thanks to Cooja's graphical user interface for network topology creation and simulation.

One of Cooja's main benefits is that it enables users to simulate WSNs' hardware and software components. Before implementing them in the real world, users can test various hardware and network protocol configurations in a simulated WSN environment. In order to aid users in finding and resolving problems during development, Cooja also offers tools for debugging and profiling WSNs. The physical-layer technologies it supports include IEEE 802.15.4, ZigBee, and 6LoWPAN.

While Contiki Cooja has many advantages as a network simulator, there are also some disadvantages to using this tool. Cooja is designed for simulating small to medium-sized wireless sensor networks. It may not be suitable for simulating larger networks or complex topologies. We are not too concerned about this limited scalability because our evaluation is not meant to simulate enterprise systems.

Although Cooja works at the appropriate scale that we would like and it supports the wireless protocols we need, it lacks software granularity. There are modules to simulate the physical requirements of the network, such as power demands and wireless range [1, 34]. However, we find that the process-level and packet-level granularity is sub-par when compared to Mininet.

6.2.4 Mininet

Mininet is an open-source network emulator and simulator that enables users to create a virtual network environment using a single Linux kernel [25]. Users can build intricate network topologies and test various networking configurations in an isolated environment. Lightweight virtualization techniques are used by Mininet to build

virtual switches and network nodes that run on a single host machine. Users have extensive control over the network environment because it simulates traffic using software-based switches and virtual hosts.

One of the main advantages of Mininet is its ability to create realistic network environments that closely resemble real-world networks. Mininet also provides a Python-based API that allows users to automate the creation of network topologies. Programmatically interacting with the virtual network environment is possible thanks to the API. Mininet is an open-source network simulator that is free to use and does not require any specialized hardware. It allows users to create a controlled and isolated network environment, which is important for the purposes of evaluation since we will be experimenting with DDoS attacks.

Mininet is compatible with different network technologies including OpenFlow, SDN, and Ethernet. Mininet-WiFi, an open-source fork of Mininet, focuses on wireless protocols [26]. It supports 6LoWPAN with built-in support for RPL routing. Hardware virtualization, as well as support for IoT protocols is the reason Mininet is a great contender for the simulation environment in our evaluation. The next primary reason for using Mininet in evaluation is the fine-grained control over nodes. We can view each simulated node's process list and launch different processes on each. Each node has its own network interfaces, which makes it possible to use network capturing tools for magnified measurement.

6.3 Mininet Simulation Environment

The simulation environment is implemented using Mininet-WiFi [26]. The simulations ran on a Windows 10 laptop with Intel i7 7700HQ CPU and four gigabytes of memory. We set up a wireless sensor network consisting of eleven sensors. The

sensors communicate through the 6LoWPAN data link protocol and messages are routed using the RPLD implementation of RPL. One of the sensors in the network is assigned to be the mobile security node. Another one of the sensors is assigned to be the malicious DDoS generating node. An additional node that is not part of the RPL DODAG is assigned to represent the Internet. The security controller is not part of the RPL DODAG. It communicates directly with the mobile node through a secure medium, allowing it to achieve high data transfer speeds. This is implemented in Mininet as a Wi-Fi 5 (802.11n) connection. DDoS traffic generated by the malicious node is targeted at the Internet node. All RPL nodes excluding the mobile security node and the DDoS node generate background traffic.

Background traffic is generated from a Scapy script running on every node excluding the mobile security node and the DDoS node. The data is sampled from the USC IoT traffic dataset [16]. This data was chosen because it was developed by observing 14 IoT devices over a span of ten days. The devices were interacted with by the researchers daily, resulting in more realistic traffic. The dataset does not include any DDoS traffic. All traffic generated by the background script is assumed to be legitimate. The source IP addresses of traffic originating from the Internet is randomized from 65534 available options. We use this number because it is the maximum number of IP addresses usable under a /16 CIDR subnet.

The destination IP addresses of the background traffic is randomly chosen, with most traffic routed to the Internet and some traffic routed to other nodes in the network. In addition to the sampled background traffic, each node generates RPL messages based on the RPLD instance running on each node [41, 61]. The mobile security node and DDoS node generate these RPL messages too.

Scapy is used to implement the DDoS traffic generating scripts. We implemented two types of DDoS attacks – UDP flood and SYN flood. We chose these two attacks since

they are two of the most common and fundamental attacks studied in computing literature. In addition, they are easy to measure and do not depend on complex timing like other DDoS attacks such as the pulse attack require.

The UDP flood attack is implemented by choosing a random sequence of bytes to attach to the payload of the packet. We chose the random number to vary between 0 and 1400 bytes because the maximum MTU size is 1500 bytes [5]. We leave a spare 100 bytes for the rest of the packet data. The 100 bytes should be able to give space for data link headers, IP headers, and TCP/UDP headers. The SYN flood attack is implemented without an additional payload. Each SYN packet is a TCP packet with the SYN flag set [4]. Because of this, each DDoS packet is the same length in bytes.

6.4 Case Study One: DDoS Originating From the Internet

We designed our first evaluation case study with the intent to study how effective the detection functions are. This is a direct evaluation of the attack detector threads in the security controller's decision module. In this case study, we assume that DDoS traffic is being generated somewhere in the Internet, and the packets are being routed to a destination within the IoT network.

The simulation environment is set up as depicted in Figure 6.1. The flow of legitimate traffic is represented by solid lines and DDoS traffic is represented by dashed lines.

The security controller is stationed at the edge of the IoT network. It assumes all responsibilities of the mobile security node, with the most important being reconnaissance and mitigation. The security controller scans all packets entering the network and identifies which IP addresses are sourcing the DDoS traffic. The security con-

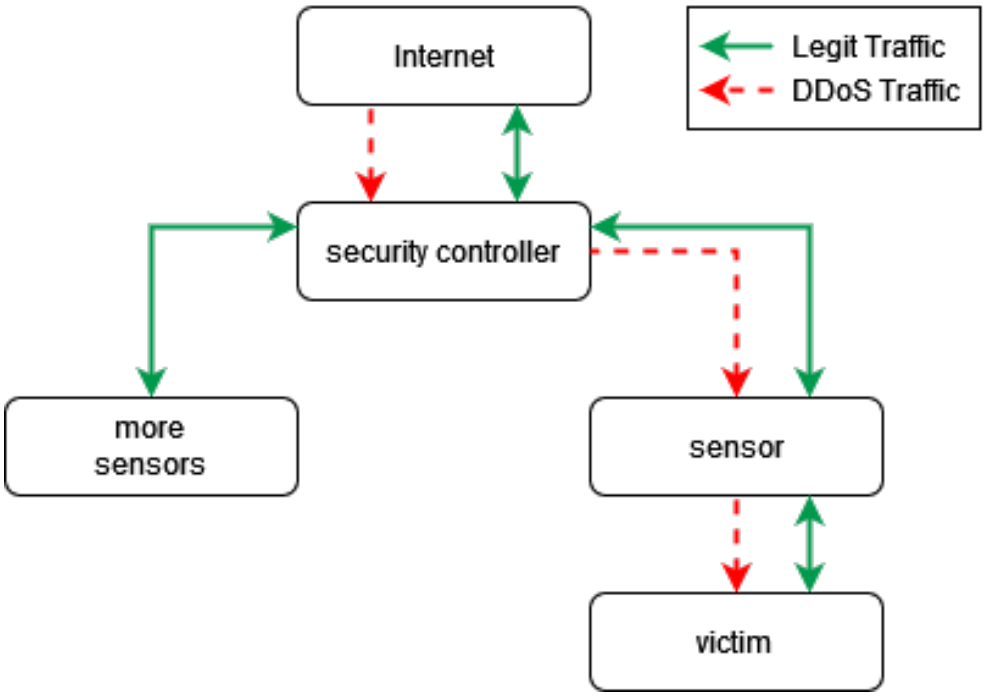

Figure 6.1: Case Study One Diagram

troller initiates an Iptables mitigation by blocking ingress traffic from the source IP addresses flagged by the decision module.

We vary the number of attacker IP addresses to measure how well the mobile firewall system performs as the attack scales. The first experiment is run with 10 attacker IP addresses. We scale exponentially up to 10000-attacker IP addresses. The sending rate in Scapy is set to be 1000 packets per second for DDoS traffic in the 10 and 100 attacker experiments. We raise the send rate to 10000 packets per second for the 1000 attacker experiment and 10000 attacker experiment.

6.5 Case Study Two: DDoS Originating From the IoT Network

We designed our second evaluation case study with the intent to study how effective the mitigation functions are. This is a direct evaluation of the enforcement of security functions by the mobile security node. In this case study, we assume that DDoS traffic

is being generated by a malicious node in the IoT network, and the packets are being routed to a destination somewhere in the Internet.

The simulation environment is set up as depicted in Figure 6.2. The flow of legitimate traffic is represented by solid lines and DDoS traffic is represented by dashed lines.

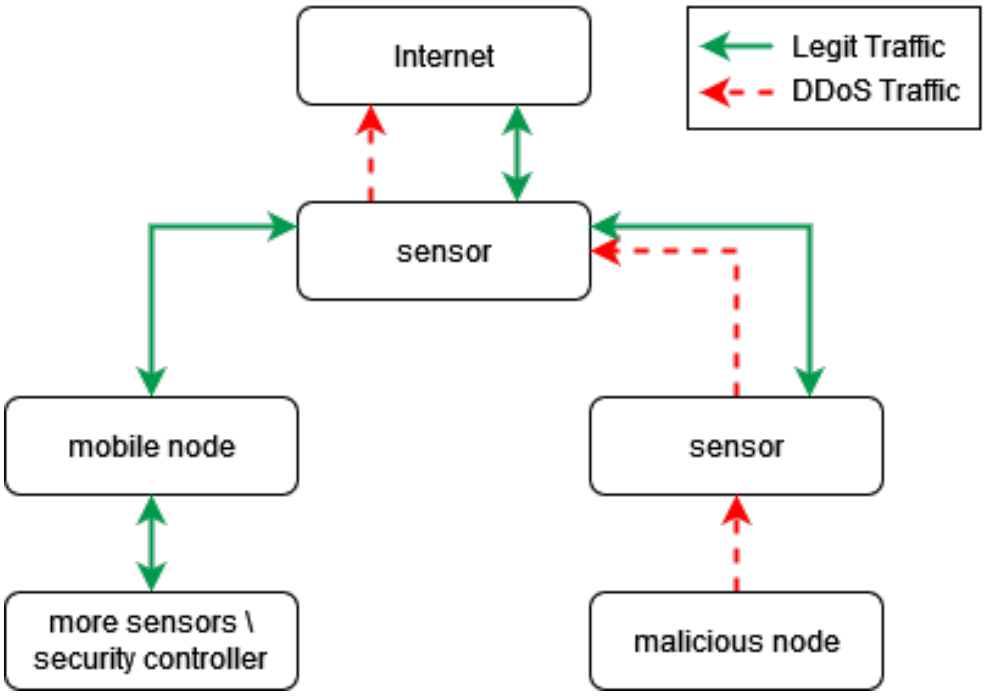

Figure 6.2: Case Study Two Diagram

One of the sensors in the network acts as a relay to the Internet. We assume that this node is not the mobile node, the security controller, or the malicious node. We assume that the security controller has already identified which node is generating DDoS traffic. Detection may be done through device profiling – each device should only be communicating with a whitelist of its appropriate servers. We make this assumption because upon experimentation, the security controller could not detect the DDoS node as malicious. This is because the size of the attack originating from the network is small. The attack consists of a single device contributing to the DDoS, resulting in one DDoS IP address that is not spoofed.

The experiment is conducted at the point when the DDoS has already been detected by the security controller. The security function has already arrived at the mobile node from the security controller. We have the mobile node wait five seconds to simulate it travelling to a physical point in the network close enough to mitigate the malicious node. The mobile node runs the security function after the five seconds are over.

6.6 Case Study Three: Node to Node Denial of Service

We designed our third evaluation case study with the intent to study how effective the system is when combining detection and mitigation in a network with no single edge. This evaluates both the attack detector threads in the security controller's decision module and the enforcement of security functions by the mobile security node. In this case study, we assume that DDoS traffic is being generated by a malicious node in the IoT network, and the packets are being routed to another node within the network. This creates a situation in which there is no single edge of the network that is monitoring traffic. This case study evaluates how the mobile firewall system is advantageous for networks that cannot rely on a traditional edge gateway. We seek to prove that mobility is an important factor when defending against DDoS in mesh networks and networks with cellular-capable devices.

The simulation environment is set up as depicted in Figure 6.3. The flow of legitimate traffic is represented by solid lines and DDoS traffic is represented by dashed lines.

A malicious node sends denial of service traffic to a victim node. We assume the victim is not the security controller or the mobile node. In order for the security controller to detect that denial of service is happening withing the network, we must configure new Snort rules. We add a local Snort rule that measures the rate of traffic

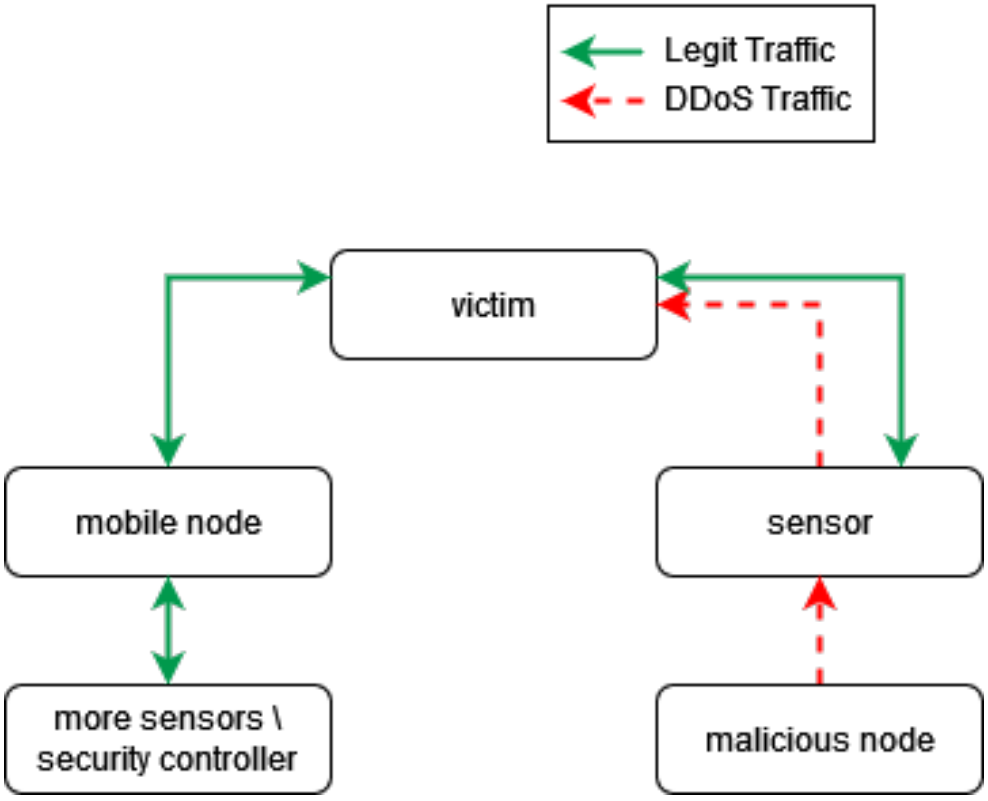

Figure 6.3: Case Study Three Diagram

between devices within the subnet. The node is flagged as malicious by Snort if the throughput exceeds 1000 packets within three seconds. The rule follows the following format:

```
alert ip SUBNET/CIDR any -> any any (msg:"Node to Node DoS";
track by_src; count 1000; seconds 3; timeout 5; sid:1000001;)
```

We propose two different types of experiments in this case study – cluster and spread. Both are tested using UDP traffic and then SYN traffic. We vary the number of attacker nodes in the IoT network as either two, five, or eight. These three amounts of attackers represent network infection rates of 20%, 50%, and 80%. The cluster class of experiments examines how the mobile firewall system performs when all attacker nodes are clustered in the same physical area. We assume that the mobile security node is within transmission radius of all malicious nodes.

The spread class of experiments emulates how a mobile node defends the network when all attackers are physically spread out. We assume that none of the attackers are within the same transmission radius. The mobile node must travel five seconds to the next attacker before being able to collect traffic measurements. The mobile node stays at the attacker for 10 seconds after mitigating its DDoS. The malicious node is removed from the network after the 10 second mitigation. The mobile node then travels to the next attacker.

An interesting behavior of the spread case is that the mobile node has a chance to mitigate the entire DDoS attack from all nodes by only defending against one attacker. This is possible when the entire RPL DODAG branch consists of malicious nodes, as illustrated in Figure 6.4. We assume the worst possible situation for case study three. We assume that the mobile node starts mitigating the DDoS at the RPL leaf nodes and works its way in to the root of the branch. The order in which the mobile node would travel in Figure 6.4 is five, four, three, two, and finally one.

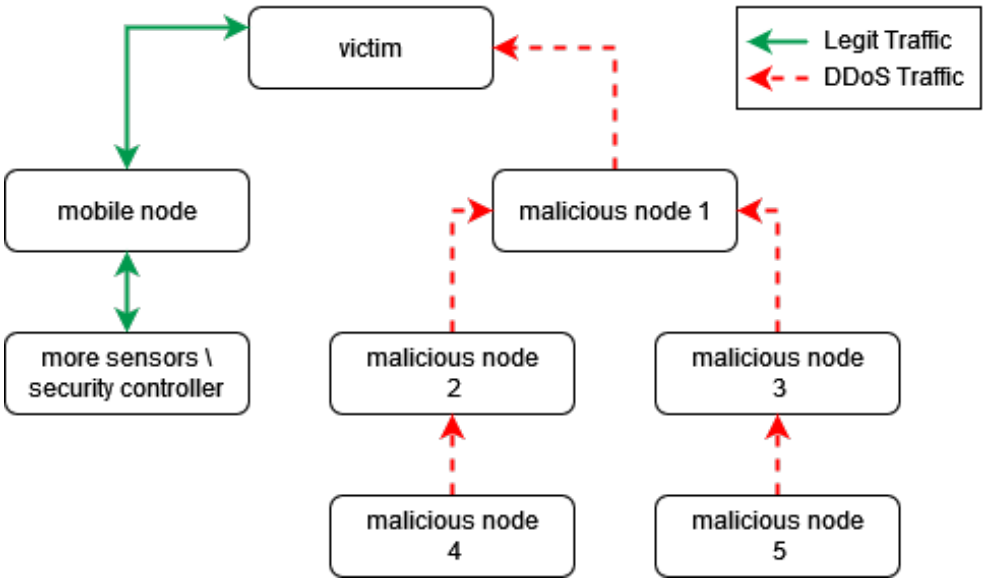

Figure 6.4: Attacker Branch Diagram

Chapter 7

RESULTS

In this chapter, we discuss the results from the evaluation. We explain the patterns present in the simulated network. Analysis of the case studies is presented. This is broken down into three parts – one section focuses on each case study. Extra attention is focused on the percentage of DDoS traffic blocked over time and the percentage of legitimate traffic blocked.

7.1 Evaluation of Case Study One: DDoS Originating From the Internet

We experimented with a varying number of attacker IP addresses. Each volume of attacker IP addresses (10, 100, 1000, 10000) is executed under both a UDP flood and a SYN flood. The traffic throughput figures depict the throughput of legitimate traffic that was allowed, the throughput of DDoS traffic that was allowed, and the throughput of DDoS traffic that was blocked. The percentage blocked figures depict the percentage of legitimate source IP addresses blocked and the percentage of malicious source IP addresses blocked over time. Figures for the 10, 100, and 10000 volumes are located in Appendix A. We discuss the 1000 attacker case in detail because it displays the most important traits of the mobile firewall system while echoing general sentiments of the other cases.

Figure 7.1 shows the throughput of legitimate traffic and UDP DDoS traffic with 1000 attacker IP addresses. The volume of UDP DDoS far exceeds the volume of legitimate traffic. We set the experiment up this way because the nature of UDP flood is volumetric. The throughput of legitimate traffic stays constant at around six

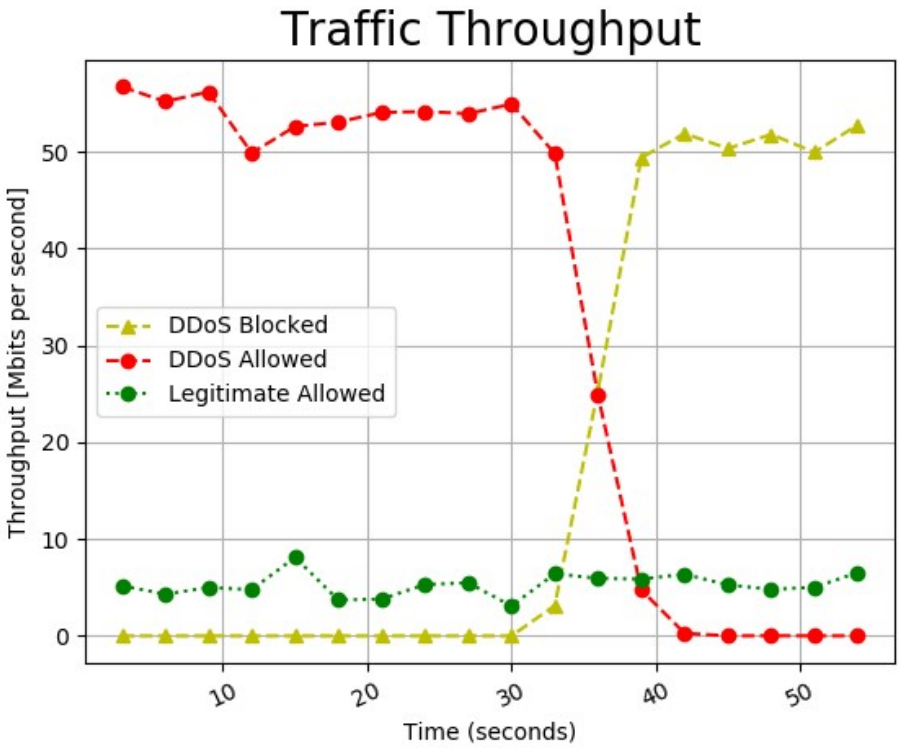

Figure 7.1: Case 1 – UDP 1000 Throughput

megabits per second across the entire time period observed. The throughput of DDoS traffic allowed into the network averages around 55 megabits per second until the 30 second mark. At this point, the security controller begins blocking IP addresses. The throughput of DDoS traffic allowed decreases sharply over a nine second period, after which all attacker IP addresses are blocked.

Figure 7.2 depicts the percentage of DDoS IP addresses blocked as well as the percentage of legitimate source IP addresses blocked. We observe that there is about a two percent increase in the number of legitimate source IP addresses blocked. This is because the attacker IP addresses are spoofed. We see this effect with a volume of 1000 attacker IP addresses because at this point, many of the attackers will overlap with the legitimate addresses. This is not the case for the volumes of 10 and 100 because there are too few attacker IP addresses spoofed to begin blocking a significant percentage of legitimate addresses.

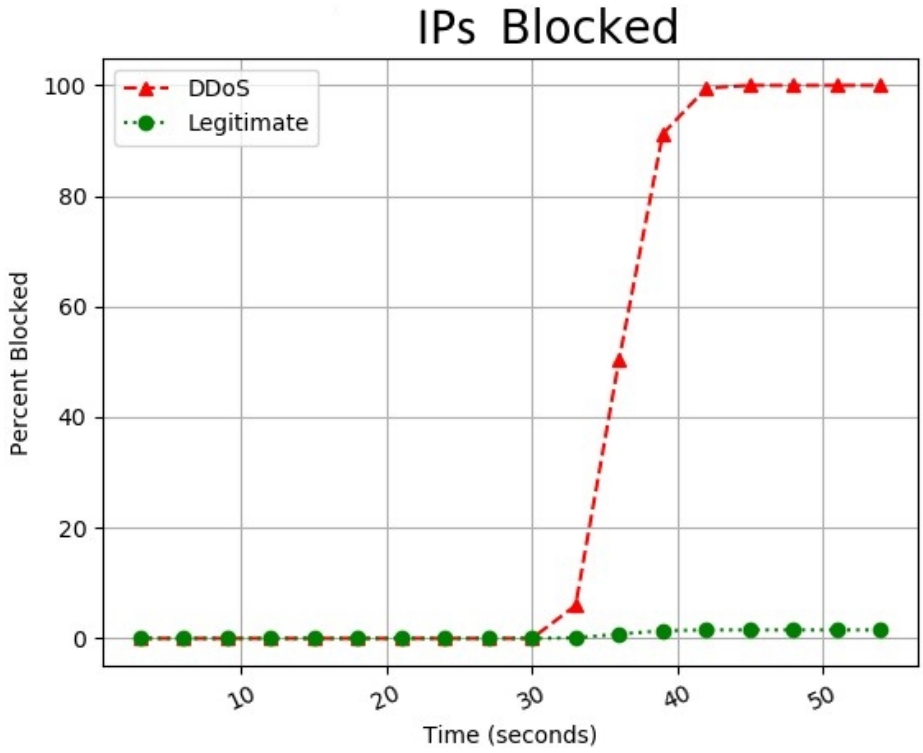

Figure 7.2: Case 1 – UDP 1000 Percentage Blocked

We observe similar trends with the SYN flood consisting of 1000 attacker IP source addresses. Figure 7.3 shows the throughput of legitimate traffic and SYN DDoS traffic with 1000 attacker IP addresses. The difference observed with the SYN flood is that the throughput of DDoS traffic is much lower than that of the UDP flood. The SYN flood throughput is in fact lower than the throughput of legitimate traffic. This is because SYN flood does not aim to exhaust hardware resources, but instead software resources. For this case only, we included the throughput of legitimate traffic blocked. Other cases do not show the legitimate traffic blocked because it is too close to zero when graphed on the same scale with DDoS traffic. Figure 7.4 depicts that the number of legitimate IP addresses blocked is significant. The effect of spoofed attacker IP addresses applies to the SYN flood as it does with the UDP flood.

When experimenting with the 10000 attacker IP address attacks, we found that our hardware could not handle the volume of traffic being generated over a long period

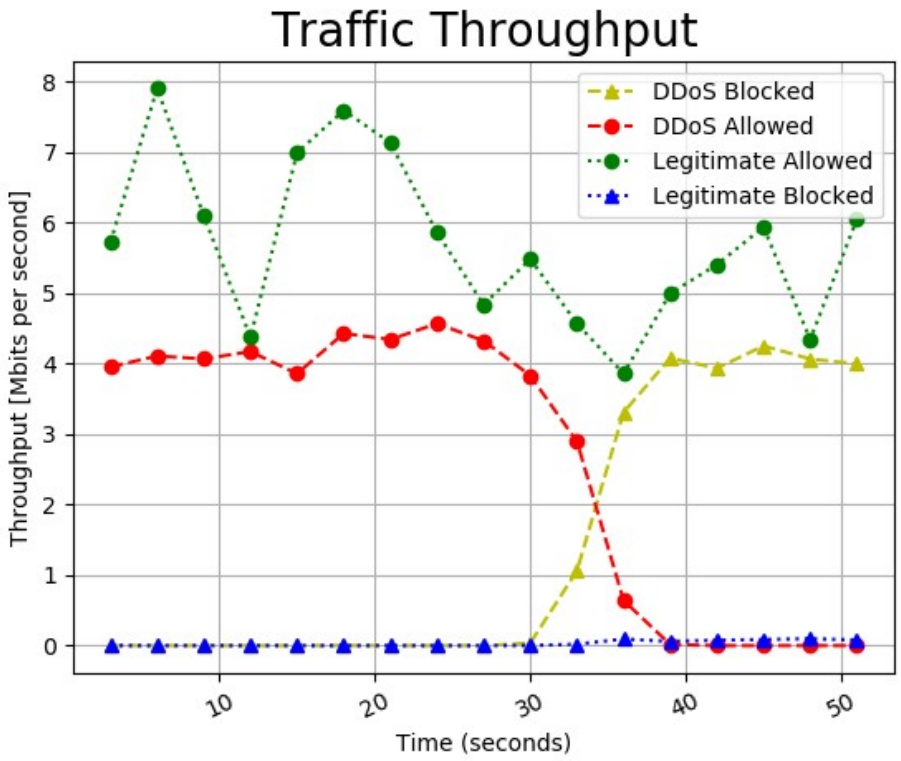

Figure 7.3: Case 1 – SYN 1000 Throughput

of time. The DDoS attack was unable to be detected within the time we measured before our hardware crashed. We are unable to make complete conclusions from the simulation because of the hardware limitations. Based on the behavior of the 1000 source volume attacks, we theorize that the DDoS detection configuration we used will successfully mitigate the attack albeit with a higher false positive rate at 10000 attackers.

The percentage of legitimate IP addresses at 1000 attack sources is about 1.5%. The number of malicious IP addresses scales by a factor of 10 while keeping the number of legitimate IP addresses constant. Therefore, we expect the percentage of legitimate IP addresses blocked to scale by a factor 10 as well, rising to around 15%. Because of this, we find that the DDoS detection configuration we used probably does not scale well for 10000 or more sources. With improved hardware to measure the traffic,

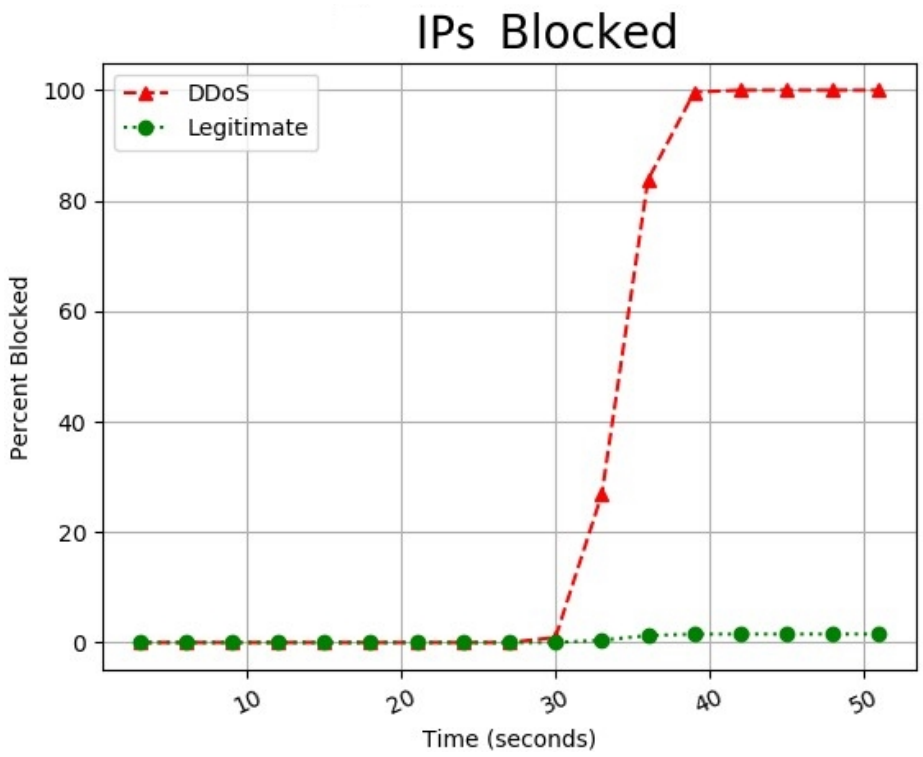

Figure 7.4: Case 1 – SYN 1000 Percentage Blocked

we believe a more scalable detection configuration would allow the mobile firewall to handle the attack while keeping false positives low.

7.2 Evaluation of Case Study Two: DDoS Originating From the IoT Network

We experimented with one malicious sensor node generating DDoS traffic. This experiment represents a malicious node contributing to a DDoS attack targeting a destination on the Internet. The experiment is executed under both a UDP flood and a SYN flood. The traffic throughput figures depict the throughput of legitimate traffic that was allowed, the throughput of DDoS traffic that was allowed, and the throughput of DDoS traffic that was blocked. The percentage blocked figures depict the percentage of legitimate source IP addresses blocked and the percentage of malicious source IP addresses blocked over time.

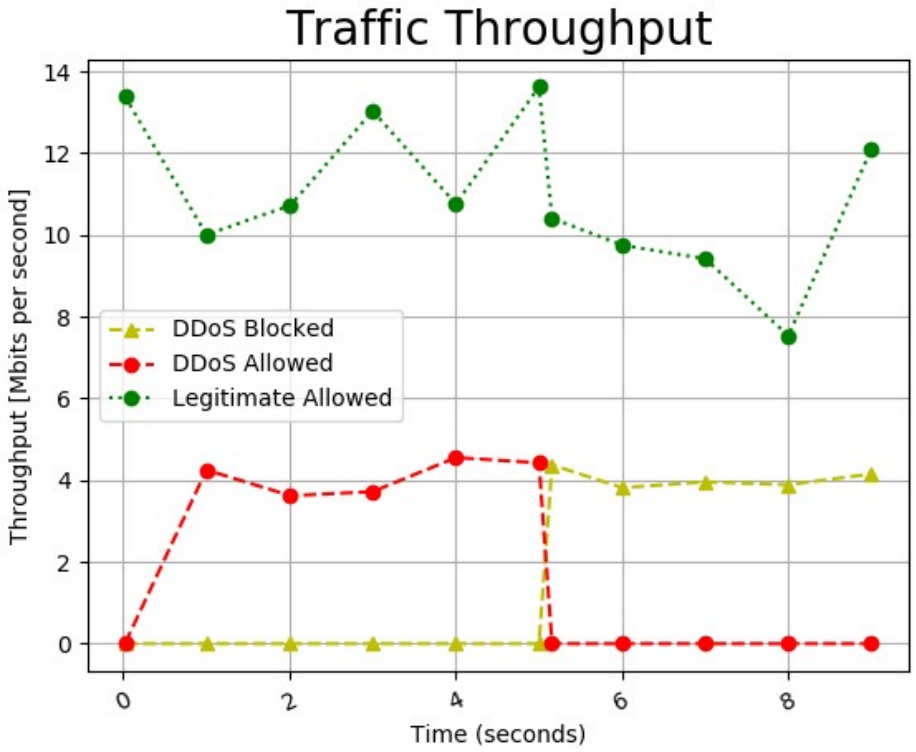

Figure 7.5: Case 2 – UDP Throughput

From Figure 7.5 and Figure 7.7 we see that the execution of the security function, the RPL black hole mitigation in this case, is fast. It takes approximately 0.14 seconds

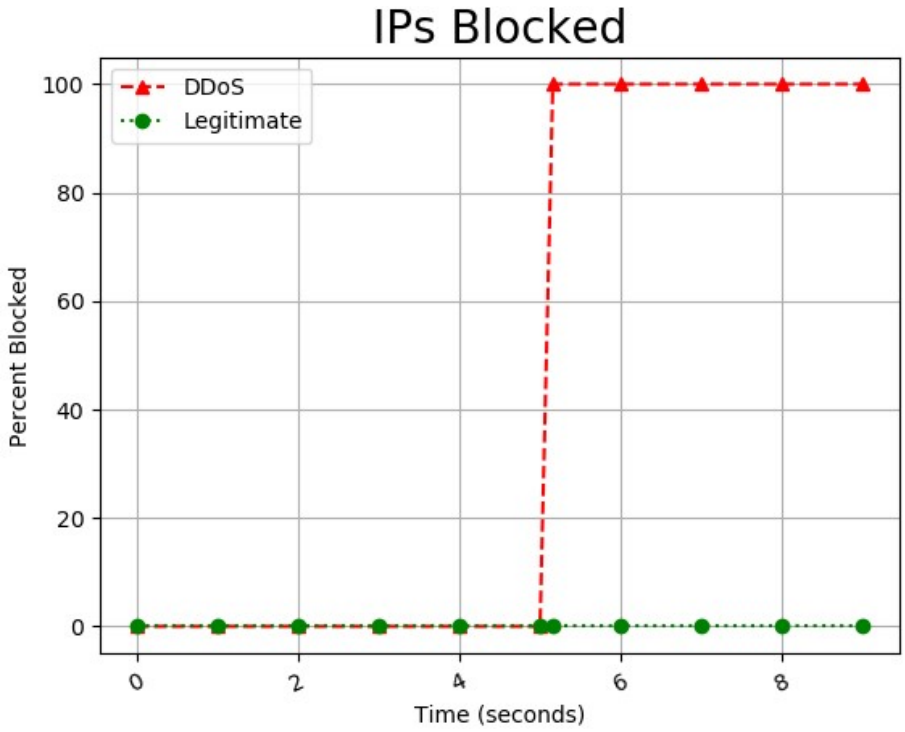

Figure 7.6: Case 2 – UDP Percentage Blocked

for the entire mitigation to start having an effect. The 0.14 seconds includes the time it takes for the mobile node to send the DIO message using the Scapy RPL script, the time for the malicious node to change its routing tables, and the time for iptables to block traffic through the mobile node.

Figures 7.6 and 7.8 echo similar findings. The percentage of DDoS IP addresses (just one address, the IP of the malicious node) surges to 100% 0.13 seconds after the five second travel time finishes. The percentage of legitimate IP addresses blocked remains zero because the malicious node is not spoofing its address.

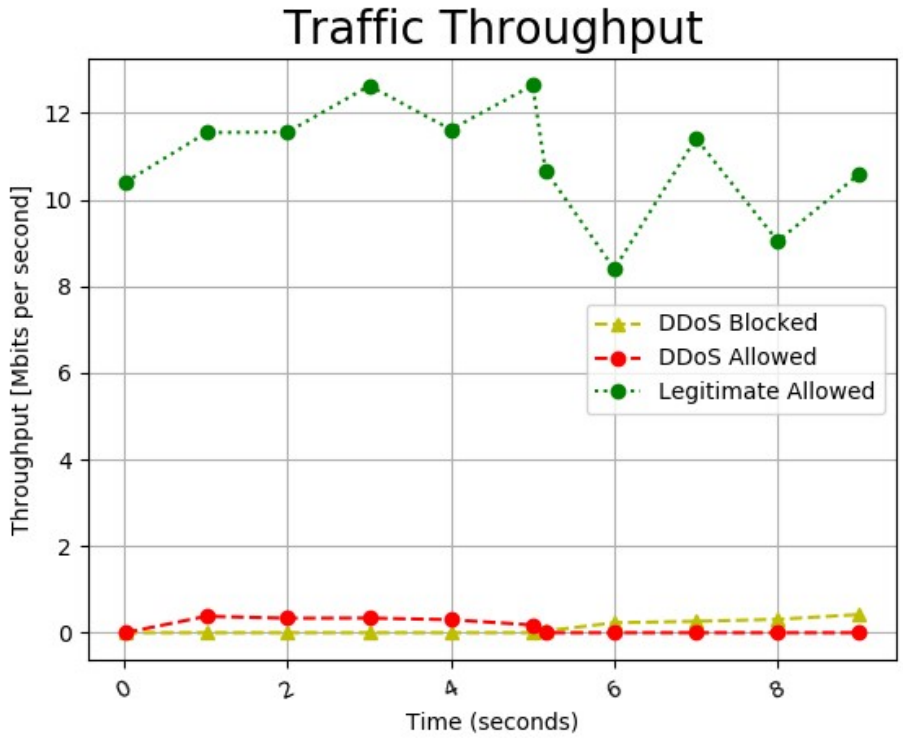

Figure 7.7: Case 2 – SYN Throughput

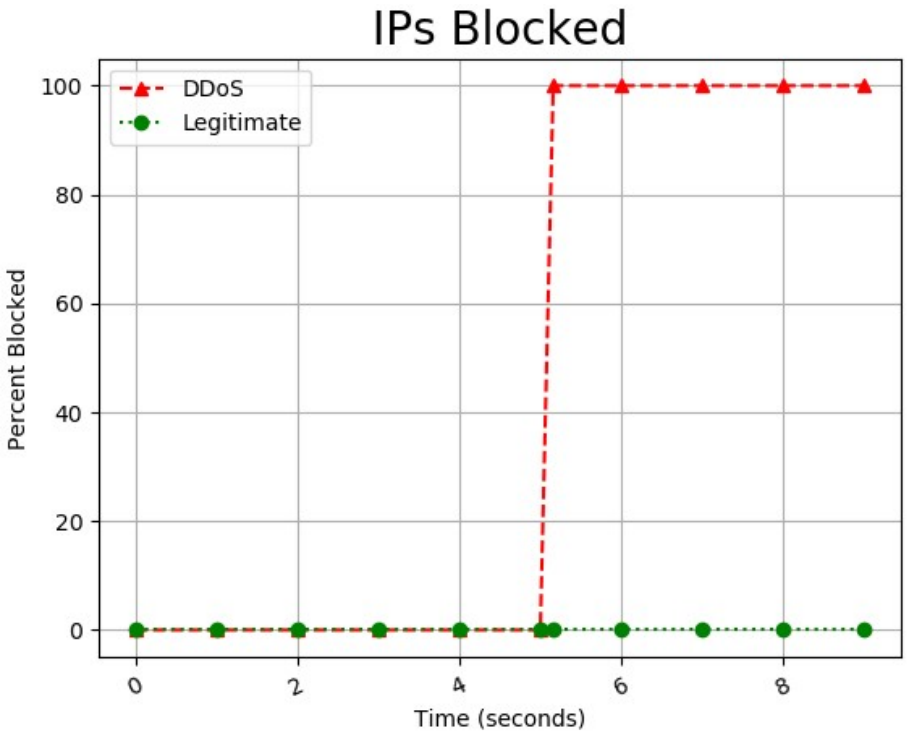

Figure 7.8: Case 2 – SYN Percentage Blocked

7.3 Evaluation of Case Study Three: Node to Node Denial of Service

We experimented with three volumes of malicious sensor nodes generating DDoS traffic. The experiments were conducted with two malicious sources, five sources, or eight sources. This experiment represents malicious nodes contributing to a DDoS attack targeting another node. UDP flood and SYN flood are the types of DDoS attacks we executed. The figures for traffic throughput show the amounts of legitimate traffic that was permitted, DDoS traffic that was permitted, and DDoS traffic that was blocked. Figures for the SYN flood experiments are located in Appendix B. We choose not to display them in this section because they parallel similar overall outcomes as the UDP experiments.

Figures 7.9, 7.10, and 7.11 demonstrate the effectiveness of the RPL black hole mitigation. Because the mobile node is already in range of all malicious nodes, it is able to collect their traffic at the same time. The traffic is then sent to the security controller, where the attack detector threads identify all malicious nodes at once. Security functions to mitigate the malicious nodes are sent to the mobile node, and the mobile node initiates the RPL black hole mitigation on all attackers simultaneously.

The result is that the DDoS assault is swiftly availed in approximately six seconds no matter the infection rate. Transfer of the network telemetry data from the mobile node to the security controller took about 1.456 seconds in the eight attacker case. Transfer times for the two attacker and five attacker case were slightly lower, at about 0.358 and 0.908 seconds respectively. Time for the security controller to analyze the eight attacker network telemetry data was about 0.804 seconds. Telemetry analysis times for the two attacker and five attacker case were lower, at about 0.198 and 0.526 seconds respectively.

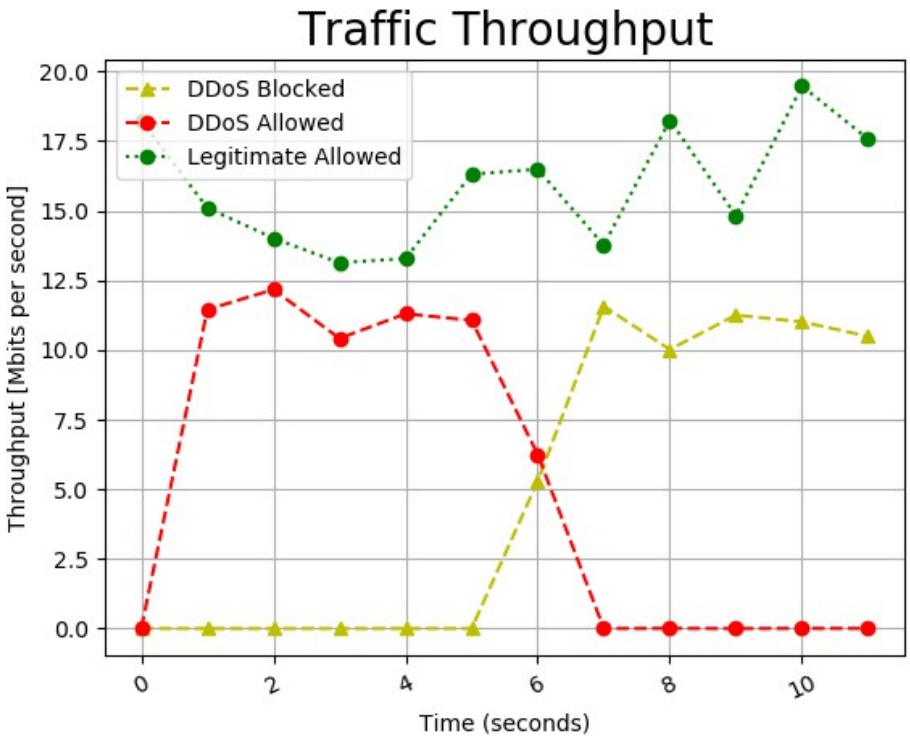

Figure 7.9: Case 3 – Cluster 2 Attacker UDP Throughput

Figures 7.12, 7.13, and 7.14 demonstrate the great benefit of mobility provided by the mobile firewall system. DDoS-allowed throughput decreases over time in the behavior of a step function as malicious nodes are detected and blocked. The waves of blocked DDoS throughput indicate the 10-second intervals in which the mobile node was actively mitigating DDoS traffic and then the attacker removed. This experiment illustrates how the mobile firewall is able to achieve defense in situations where traditional gateway solutions cannot. The graphs show that the time it takes to fully nullify the DDoS attack depends on the number of malicious sources. With just two attackers, the mobile firewall system fully mitigated the spread UDP attack in approximately 22 seconds. The defense time rose to approximately 77 seconds in the five attacker case. In the eight attacker case, the total defense time was approximately 131 seconds.

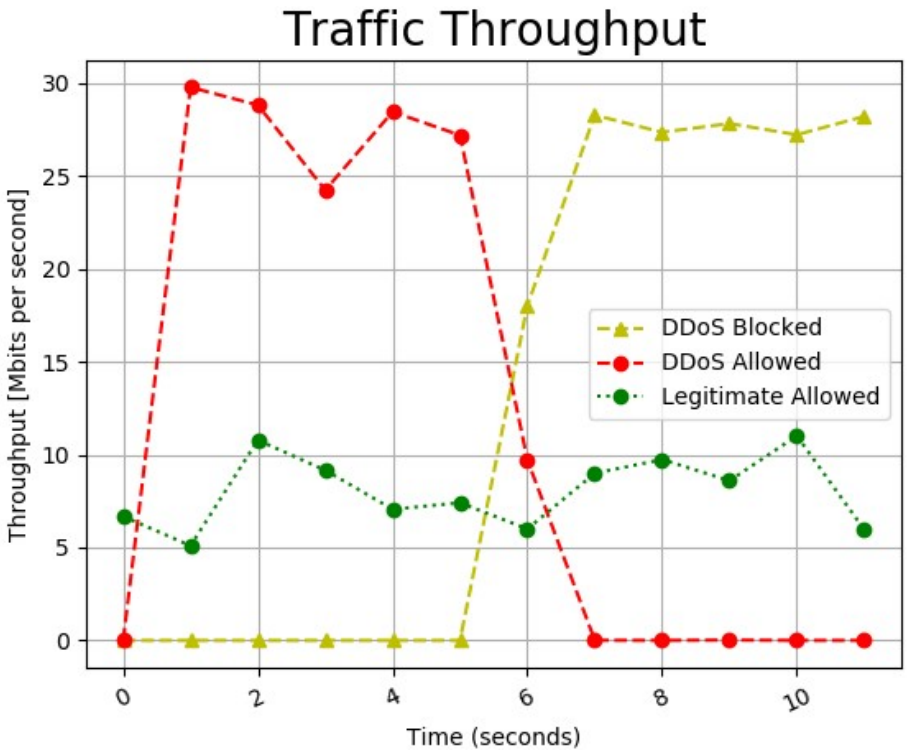

Figure 7.10: Case 3 – Cluster 5 Attacker UDP Throughput

One of the benefits of having the DDoS generating nodes in a spread out configuration is that the mobile node no longer has to collect traffic from all nodes at once. It collects data from a single node at a time, resulting in telemetry data arriving at the security controller sooner. Average data transfer times were around 0.201 seconds. Analysis times were faster too, at approximately 0.116 seconds. The ultimate result though is that the mobile security node must spend time travelling, detecting, and mitigating each attacker. The mobile node must repeat the process as many times as there are DDoS devices.

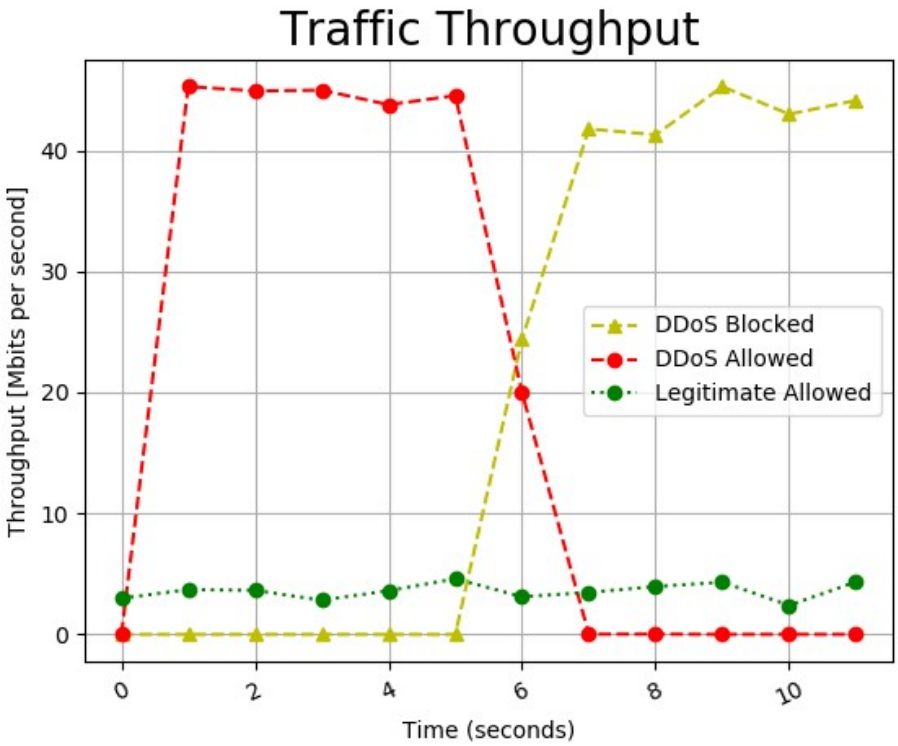

Figure 7.11: Case 3 – Cluster 8 Attacker UDP Throughput

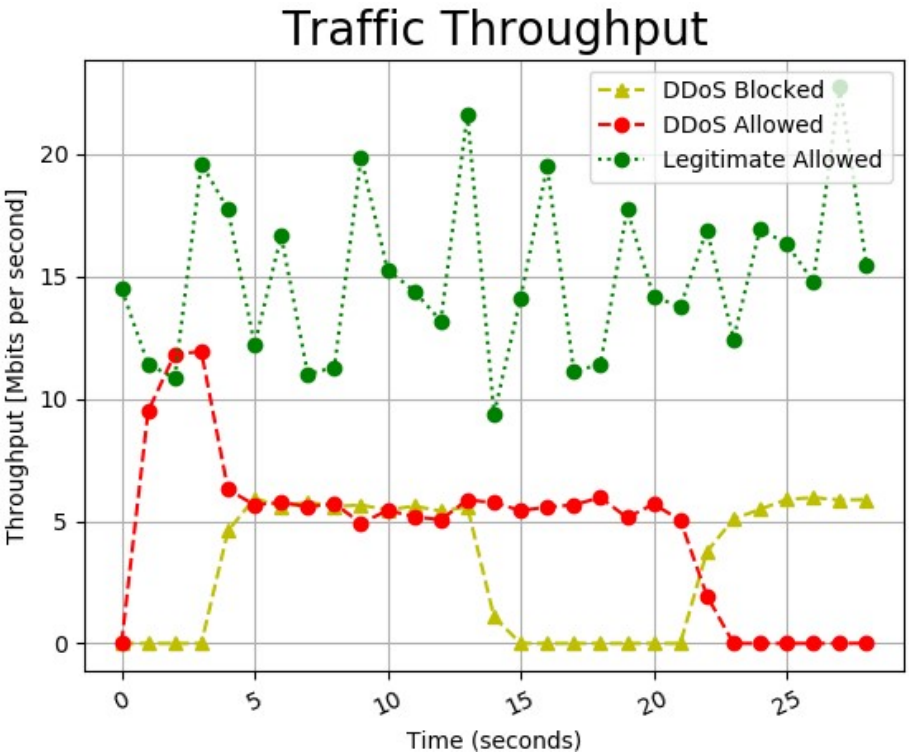

Figure 7.12: Case 3 – Spread 2 Attacker UDP Throughput

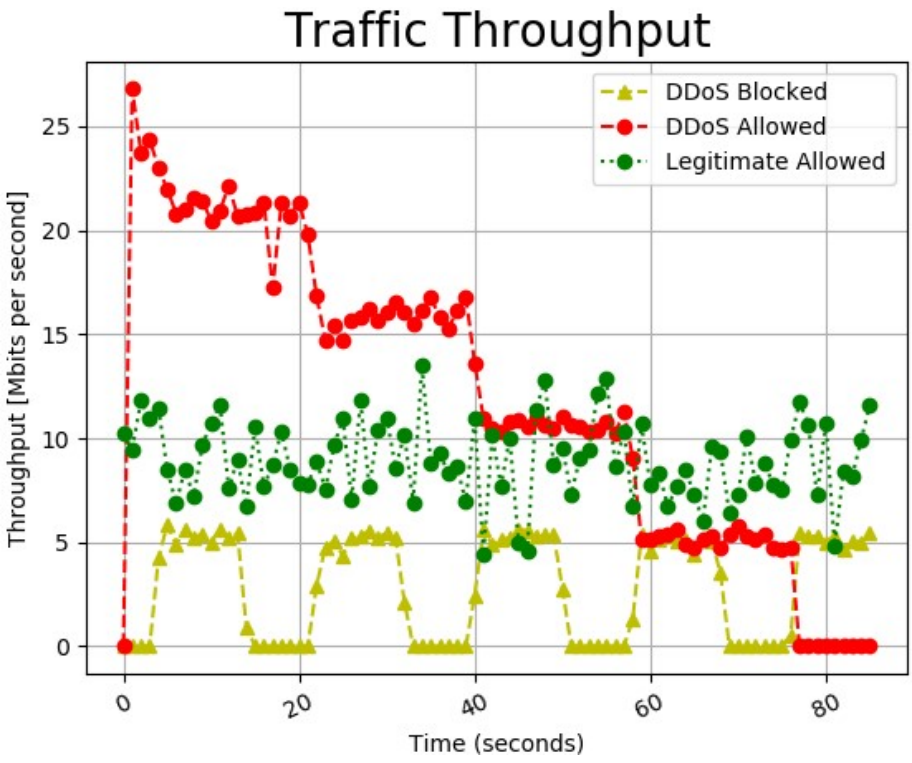

Figure 7.13: Case 3 – Spread 5 Attacker UDP Throughput

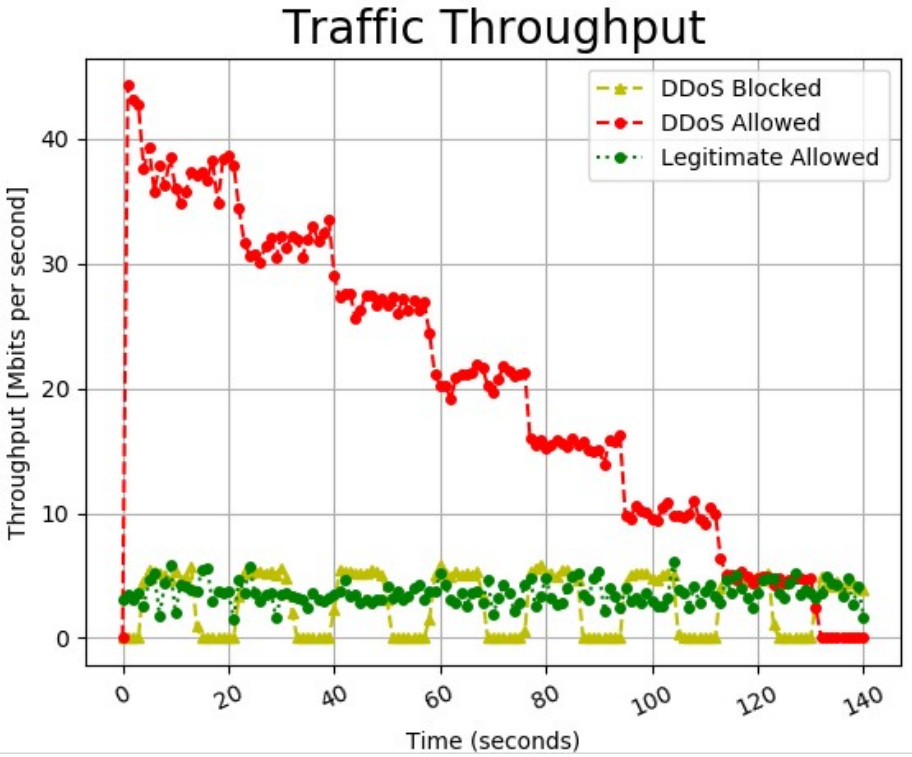

Figure 7.14: Case 3 – Spread 8 Attacker UDP Throughput

Chapter 8

FUTURE WORK & CONCLUSION

8.1 Future Work

There is room for improvement in the area of attack detectors and security functions. The mobile firewall system can be expanded upon by adding more intrusion detection systems to the attack detector threads. This may involve using tools that are not explicitly meant for DDoS detection. Attack detector threads can be added to identify more complicated DDoS attack types, such as the pulsing attack or application layer DDoS. Adding more tools could prove that the mobile firewall system is applicable to other types of malicious traffic not tested.

The mobile firewall system can be expanded upon by creating more security functions. We examine RPL black hole mitigation in this thesis, but there may be other attacks that require a different mitigation strategy. Future work should investigate the flexibility of the mobile firewall system and how well it works with other security functions.

One interesting area of research that we did not explore is the possibility of adding more than one mobile node or security controller to the network. We find the potential of adding more security nodes important because we could then further test the scalability of the mobile firewall system. More mobile security nodes could in theory block a larger volume of attackers.

Evaluating the mobile firewall system in a live IoT environment would be beneficial to better analyze the mobility of the mobile node. As with all simulations, there are

many details that cannot be perfectly replicated. Testing and evaluation could be done with a physical IoT testbed to better mimic network behaviors. Traffic may be generated directly from the devices running authentic IoT programs.

8.2 Conclusion

In this thesis, we present an implementation and evaluation of the mobile firewall system for distributed denial of service defense in Internet of Things networks. The proposed implementation achieves desirable results while addressing the challenges posed by IoT networks. Our solution is extensible for new security measures, available to constrained machines, resilient to changes, and does not rely on traditional networking architectures. It provides benefits over other current IoT-focused DDoS solutions and has potential to apply to more security related threats.

The system was implemented by integrating industry-standard tools. Three realistic use-cases targeted at IoT scenarios were evaluated in simulation. Results show that the mobile firewall system can cut 1000-attacker DDoS throughputs at the edge of the network by 100% in under 42 seconds of detection and mitigation. This mitigation applies to two observed classes of DDoS attack while keeping collateral IP bans under 1.5%. Implementation of the IoT-focused RPL black hole mitigation is a success with average deployment time on the scale of tenths of a second. When defending against attackers who are physically distant, mobility has been found to be quite advantageous; the system completely neutralizes a worst-case attack scenario in less than 131 seconds.

BIBLIOGRAPHY

[1] A. Dhondt, "rpl attacks framework," github, dec. 04, 2022. https://github.com/dhondta/rpl-attacks.

[2] Cloudflare, "Famous ddos attacks: The largest ddos attacks of all time," 2018, (access dec. 04, 2022.). https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/.

[3] Cloudflare, "http flood attack" 2023, (access dec. 04, 2022.). https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/.

[4] Cloudflare, "syn flood attack" 2018, (access dec. 04, 2022.). https://www.cloudflare.com/learning/ddos/syn-flood-ddos-attack/.

[5] Cloudflare, "what is mtu (maximum transmission unit)?" 2018, (access dec. 04, 2022.). https://www.cloudflare.com/learning/network-layer/what-is-mtu/.

[6] D. Sisodia, "On the state of internet of things security: Vulnerabilities, attacks, and recent countermeasures." [online]. available:. https://devkishen.com/pdfs/On-the-State-of-Internet-of-Things-Security-Vulnerabilities-Attacks-and-Recent-Countermeasures.pdf.

[7] Dos mitigation with snort. https://github.com/maj0rmil4d/snort-ddos-mitigation.

[8] dumpcap(1) manual page. https://www.wireshark.org/docs/man-pages/dumpcap.html.

[9] E. Ronen, a. shamir, a.-o. weingarten, and c. oflynn, "iot goes nuclear: Creating a zigbee chain reaction," 2017 ieee symposium on security and privacy (sp), may 2017, doi: 10.1109/sp.2017.14.

[10] Effective go. https://go.dev/doc/effective_go.

[11] Gns3 the software that empowers network professionals. https://www.gns3.com/.

[12] The go programming language. https://go.dev/.

[13] Go standard library. https://pkg.go.dev/std.

[14] Google cloud, "exponential growth in ddos attack volumes" 2020,. https://cloud.google.com/blog/products/identity-security/identifying-and-protecting-against-the-largest-ddos-attacks.

[15] Ids, "what is an intrusion detection system?" 2023. https://www.paloaltonetworks.com/cyberpedia/what-is-an-intrusion-detection-system-ids.

[16] Impact - iot operation traces-20200127. IoT Operation Traces-20200127 — University of Southern California-Information Sciences Institute, doi: 10.23721/109/1520166.

[17] An introduction to cooja. https://github.com/contiki-os/contiki/wiki/An-Introduction-to-Cooja.

[18] Iot, "what is the internet of things (iot)?" 2023. https://www.ibm.com/blog/what-is-the-iot/.

[19] Iptables tutorial: Ultimate guide to linux firewall. https://phoenixnap.com/kb/iptables-tutorial-linux-firewall.

[20] iptables(8) - linux man page. https://linux.die.net/man/8/iptables.

[21] J. Ren, d. j. dubois, d. choffnes, a. m. mandalari, r. kolcun, and h. haddadi, "information exposure from consumer iot devices," proceedings of the internet measurement conference on - imc '19, 2019, doi: 10.1145/3355369.3355577.

[22] L. Shi, "Two decades of ddos attacks and defenses." accessed: Dec. 06, 2022. [online]. available:. https://www.cs.uoregon.edu/Reports/AREA-202002-Shi.pdf.

[23] M. Antonakakis et al., "understanding the mirai botnet open access to the proceedings of the 26th usenix security symposium is sponsored by usenix understanding the mirai botnet," 2017. [online].

[24] Mensagens de controle do rpl. https://www.gta.ufrj.br/ensino/eel879/vf/rpl/mensagens.html.

[25] Mininet: An instant virtual network on your laptop (or other pc). https://mininet.org/.

[26] Mininet-wifi: Emulation platform for software-defined wireless networks. https://mininet-wifi.github.io/.

[27] N. Kushalnagar, "Ipv6 over low-power wireless personal area networks (6lowpans)," 2007. accessed: Dec. 06, 2022. [online]. available:. https://www.rfc-editor.org/rfc/pdfrfc/rfc4919.txt.pdf.

[28] netlink(7) — linux manual page. https://www.man7.org/linux/man-pages/man7/netlink.7.html.

[29] Omnet++ discrete event simulator. https://omnetpp.org/.

[30] Openssh. https://www.openssh.com/.

[31] Saulles, m. “internet of things statistics” 2022,. https://informationmatters.net/internet-of-things-statistics/.

[32] Snort. https://www.snort.org/.

[33] Welcome to scapy. https://scapy.net/.

[34] What can i do to implement a power trace function in a contiki cooja rpl simulation. https://networksimulationtools.com/what-can-i-do-to-implement-a-power-trace-function-in-a-contiki-cooja-rpl-simulation/.

[35] F. Ahmed and Y.-B. Ko. Mitigation of black hole attacks in routing protocol for low power and lossy networks. *Security and Communication Networks*, 9(18):5143–5154, 2016.

[36] H. Beitollahi and G. Deconinck. Analyzing well-known countermeasures against distributed denial of service attacks. *Computer Communications*, 35(11):1312–1332, 2012.

[37] D. Breitenbacher, I. Homoliak, Y. L. Aung, Y. Elovici, and N. O. Tippenhauer. Hades-iot: A practical and effective host-based anomaly detection system for iot devices (extended version). *IEEE Internet of Things Journal*, 9(12):9640–9658, 2022.

[38] P. Bull, R. Austin, E. Popov, M. Sharma, and R. Watson. Flow based security for iot devices using an sdn gateway. *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pages 157–163, 2016.

[39] X. Chen, L. Xiao, W. Feng, N. Ge, and X. Wang. Ddos defense for iot: A stackelberg game model-enabled collaborative framework. *IEEE Internet of Things Journal*, 9(12):9659–9674, 2022.

[40] M. A. Ferrag and L. Maglaras. Deepcoin: A novel deep learning and blockchain-based energy exchange framework for smart grids. *IEEE Transactions on Engineering Management*, 67:1285–1297, 2020.

[41] R. Fontes. Ramonfontes/rpld: Yet another rpl implementation for linux. https://github.com/ramonfontes/rpld.

[42] J. Frankenfield. Eavesdropping attack, Nov 2022.

[43] C. J. Fung and Y. Pillai. A privacy-aware collaborative ddos defence network. *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–5, 2020.

[44] A. Gupta, R. Christie, and P. R. Manjula. Scalability in internet of things : Features, techniques and research challenges. 2017.

[45] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. K. Sikdar. A survey on iot security: Application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019.

[46] R. F. Hayat, S. Aurangzeb, M. Aleem, G. Srivastava, and J. C.-W. Lin. Ml-ddos: A blockchain-based multilevel ddos mitigation mechanism for iot environments. *IEEE Transactions on Engineering Management*, pages 1–14, 2022.

[47] e. a. J. Nieminen. IPv6 over BLUETOOTH(R) Low Energy. RFC 7668, RFC Editor, October 2015.

[48] M. S. Kang, S. B. Lee, and V. D. Gligor. The crossfire attack. In *2013 IEEE Symposium on Security and Privacy*, pages 127–141, 2013.

[49] A. Kovačević. How to use sftp commands and options, Dec 2022.

[50] Y. Liu, K.-F. Tong, X. Qiu, Y. Liu, and X. Ding. Wireless mesh networks in iot networks. In *2017 International Workshop on Electromagnetics: Applications and Student Innovation Competition*, pages 183–185, 2017.

[51] M. Lyu, D. Sherratt, A. Sivanathan, H. H. Gharakheili, A. Radford, and V. Sivaraman. Quantifying the reflective ddos attack capability of household iot devices. In *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '17, page 46–51, New York, NY, USA, 2017. Association for Computing Machinery.

[52] S. Mergendahl, D. Sisodia, J. Li, and H. Çam. Source-end ddos defense in iot environments. *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy*, 2017.

[53] G. Newell. How to securely copy files between linux hosts using scp and sftp, Dec 2020.

[54] e. a. P. Levis. The Trickle Algorithm. RFC 6206, RFC Editor, March 2011.

[55] Pavel-Odintsov. Pavel-odintsov/fastnetmon: Fastnetmon. https://github.com/pavel-odintsov/fastnetmon.

[56] M. Roopak, G. Y. Tian, and J. Chambers. An intrusion detection system against ddos attacks in iot networks. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0562–0567, 2020.

[57] P. S. F. Sheron, K. P. Sridhar, S. Baskar, and P. M. Shakeel. A decentralized scalable security framework for end-to-end authentication of future iot communication. *Transactions on Emerging Telecommunications Technologies*, 31, 2019.

[58] F. S. D. Silva, E. Silva, E. P. Neto, M. O. O. Lemos, A. V. Neto, and F. Esposito. A taxonomy of ddos attack mitigation approaches featured by sdn technologies in iot scenarios. *Sensors (Basel, Switzerland)*, 20, 2020.

[59] D. Sisodia. *D. Sisodia, "DEFENDING AGAINST IOT-ENABLED DDOS ATTACKS AT CRITICAL VANTAGE POINTS ON THE INTERNET." [Online]. Available:*. PhD thesis, University of Oregon, 2022.

[60] D. Sisodia, J. Li, and L. Jiao. In-network filtering of distributed denial-of-service traffic with near-optimal rule selection. ASIA CCS '20, page 153–164, 2020.

[61] E. e. a. T. Winter. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, RFC Editor, March 2012.

[62] I. Tyou, H. Nagayama, T. Saeki, Y. Nagafuchi, and M. Tanikawa. Decentralized iot security gateway. *2018 3rd Cloudification of the Internet of Things (CIoT)*, pages 1–6, 2018.

[63] Q. Zhang and W. Zhang. Accurate detection of selective forwarding attack in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 15(1):1550147718824008, 2019.

[64] Zkaleem. Rfmon – radio frequency monitoring, monitor mode, Jan 2008.

APPENDICES

Appendix A

CASE STUDY ONE RESULTS

This appendix includes the figures from case study one that were not included in the Results chapter. Included here are the 10 attacker cases of both the SYN and UDP flood classes, the 100 attacker cases of both the SYN and UDP flood classes, and the 10000 attacker case which we only saved UDP data for. We did not collect SYN data for the 10000 attacker case because the results were similar to the UDP experiment. Our hardware was not able to handle it over the amount of time required to detect the attack.

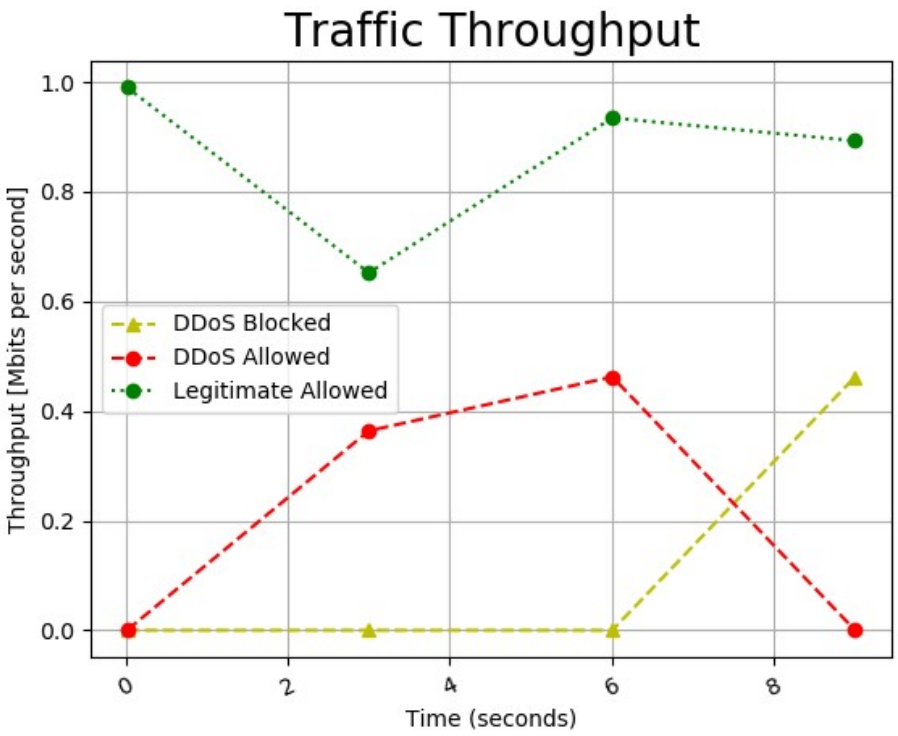

Figure A.1: Case 1 – SYN 10 Throughput

Figure A.1 shows the throughput of traffic when the DDoS consisted of 10 SYN flood attackers.

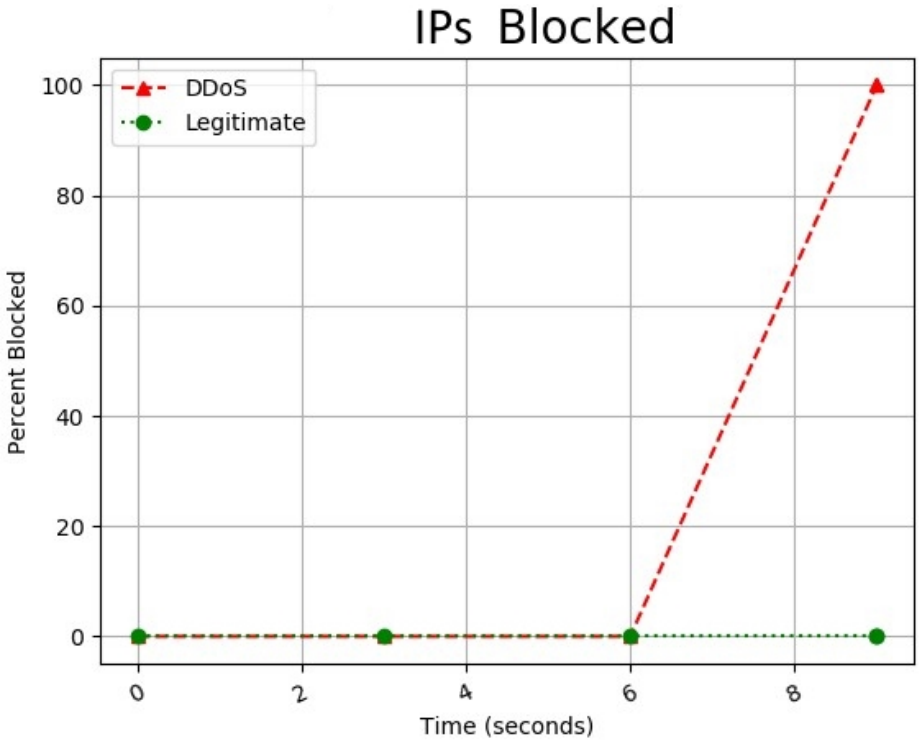

Figure A.2: Case 1 – SYN 10 Percentage Blocked

Figure A.2 shows the percentage of malicious IP addresses blocked when the DDoS consisted of 10 SYN flood attackers.

Figure A.3 shows the throughput of traffic when the DDoS consisted of 10 UDP flood attackers.

Figure A.4 shows the percentage of malicious IP addresses blocked when the DDoS consisted of 10 UDP flood attackers.

Figure A.5 shows the throughput of traffic when the DDoS consisted of 100 SYN flood attackers.

Figure A.6 shows the percentage of malicious IP addresses blocked when the DDoS consisted of 100 SYN flood attackers.

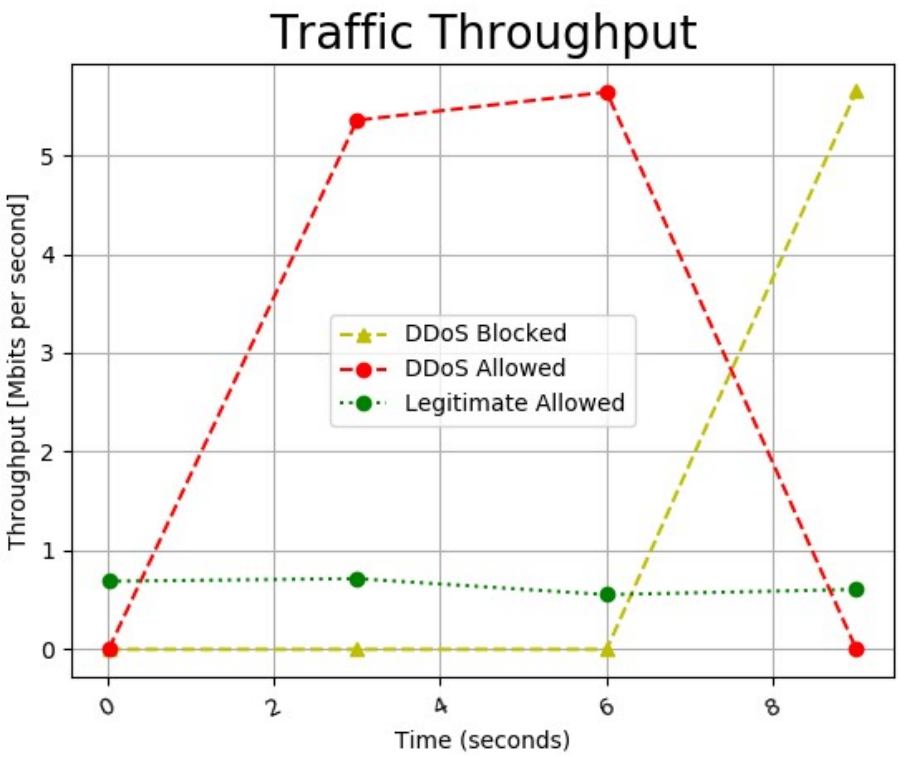

Figure A.3: Case 1 – UDP 10 Throughput

Figure A.7 shows the throughput of traffic when the DDoS consisted of 100 UDP flood attackers.

Figure A.8 shows the percentage of malicious IP addresses blocked when the DDoS consisted of 100 UDP flood attackers.

Figure A.9 shows the throughput of traffic when the DDoS consisted of 10000 UDP flood attackers.

Figure A.10 shows the percentage of malicious IP addresses blocked when the DDoS consisted of 10000 UDP flood attackers.

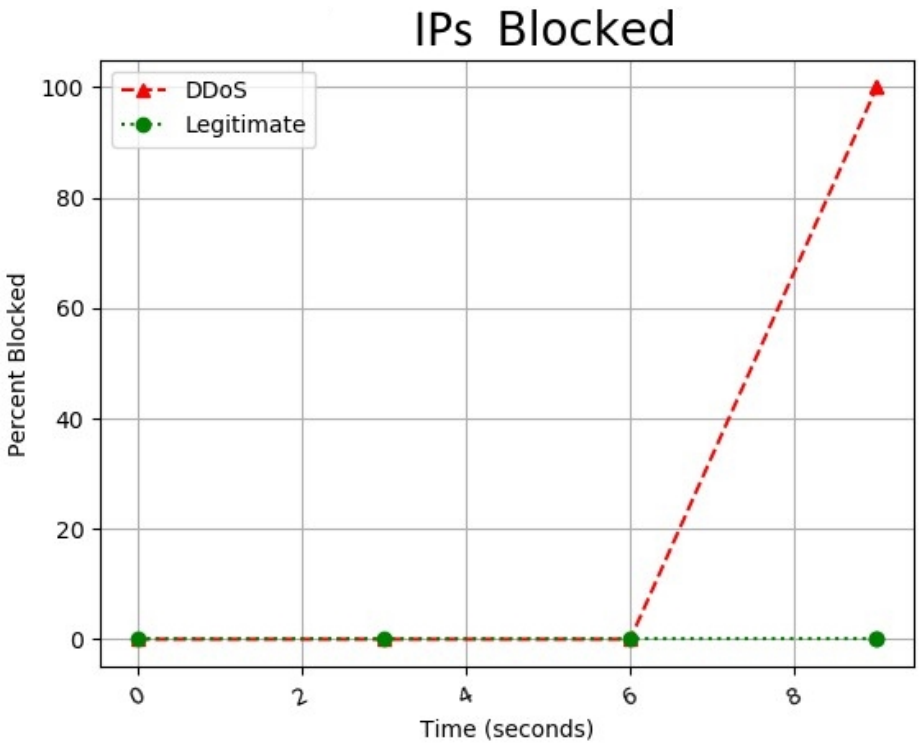

Figure A.4: Case 1 – UDP 10 Percentage Blocked

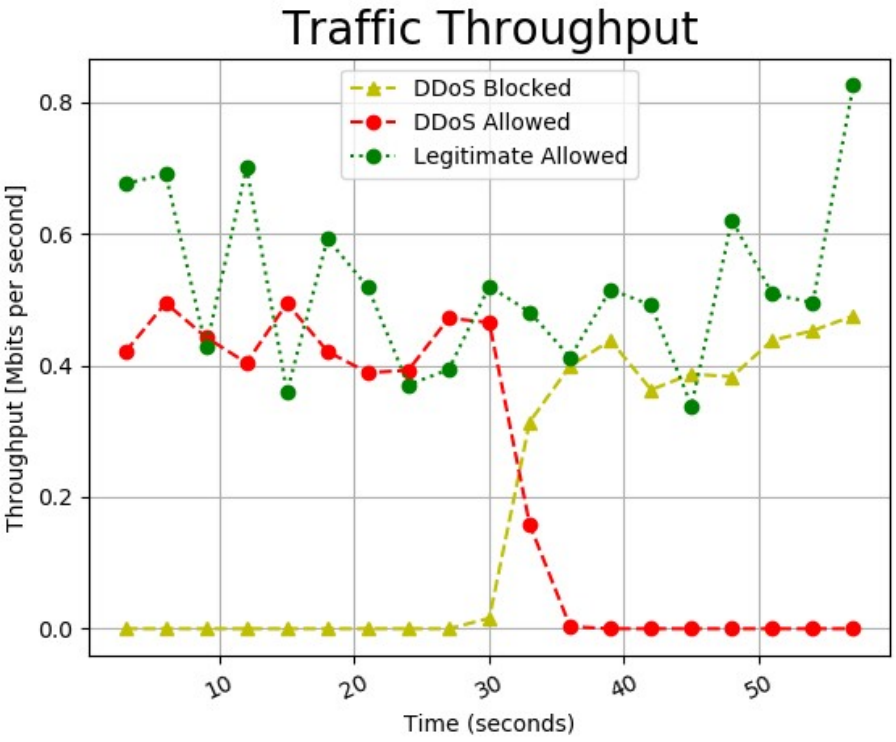

Figure A.5: Case 1 – SYN 100 Throughput

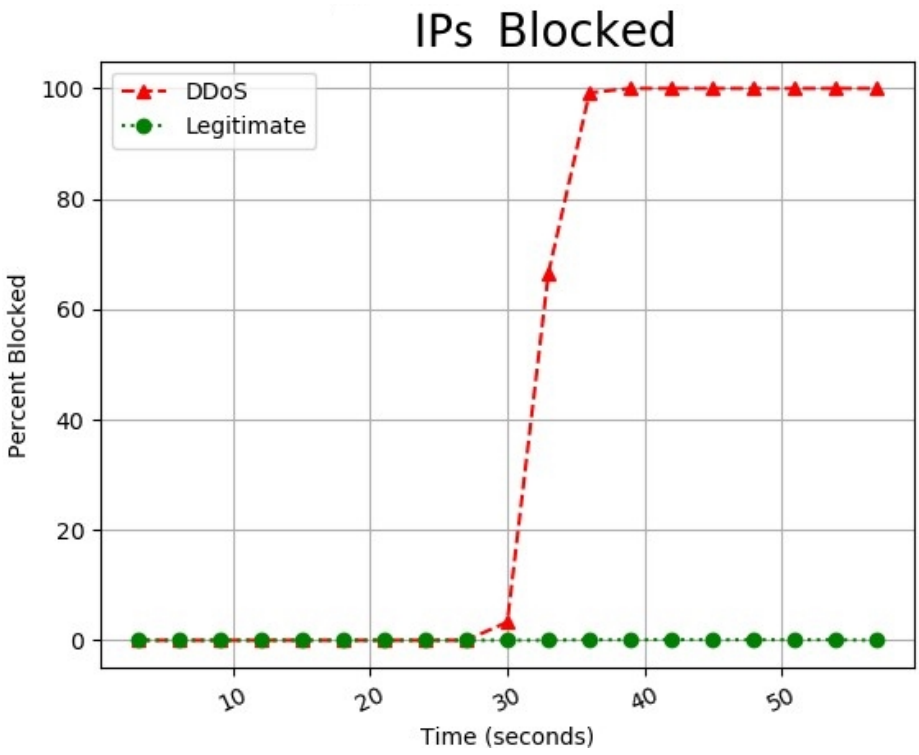

Figure A.6: Case 1 – SYN 100 Percentage Blocked

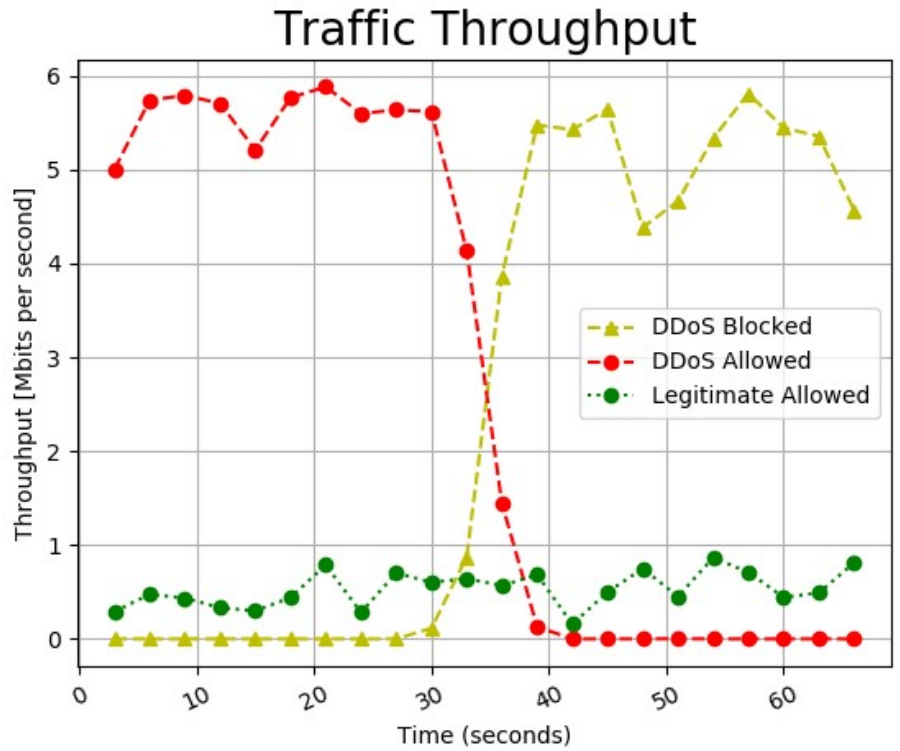

Figure A.7: Case 1 – UDP 100 Throughput

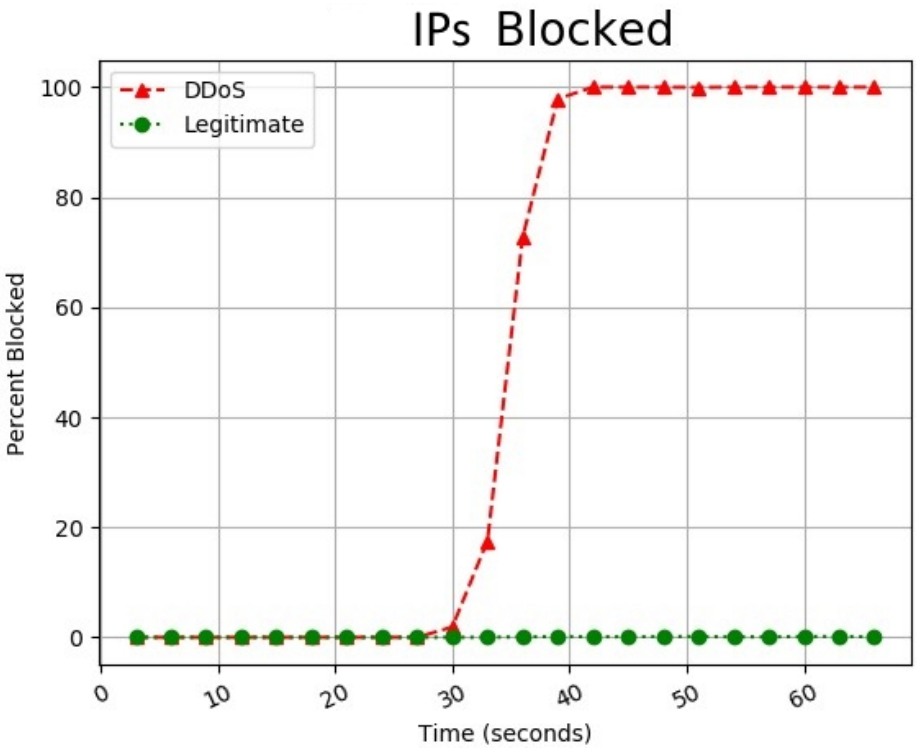

Figure A.8: Case 1 – UDP 100 Percentage Blocked

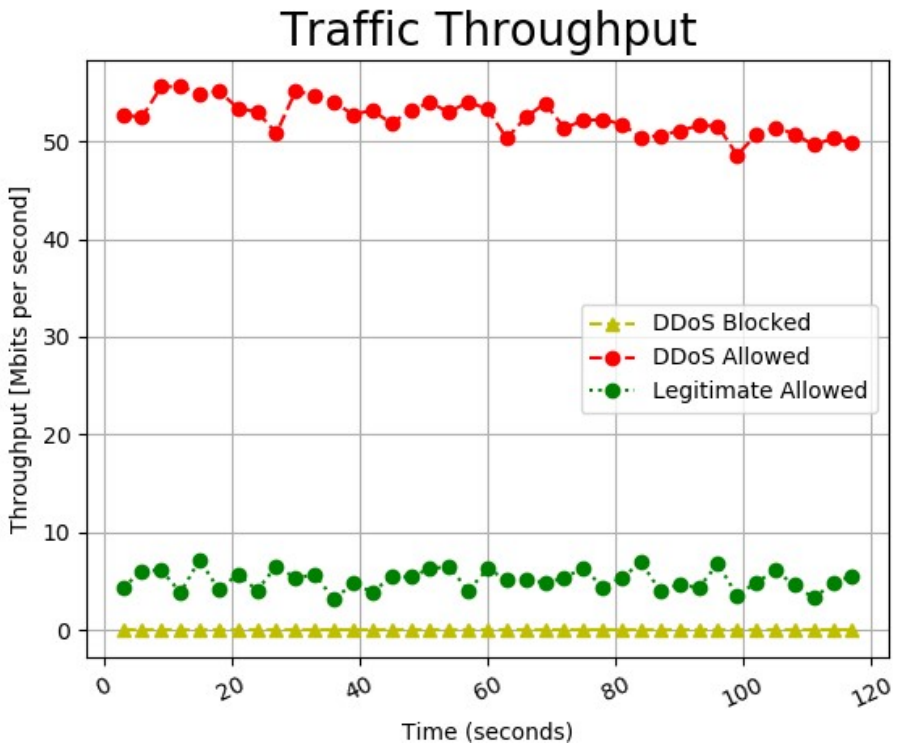

Figure A.9: Case 1 – UDP 10000 Throughput

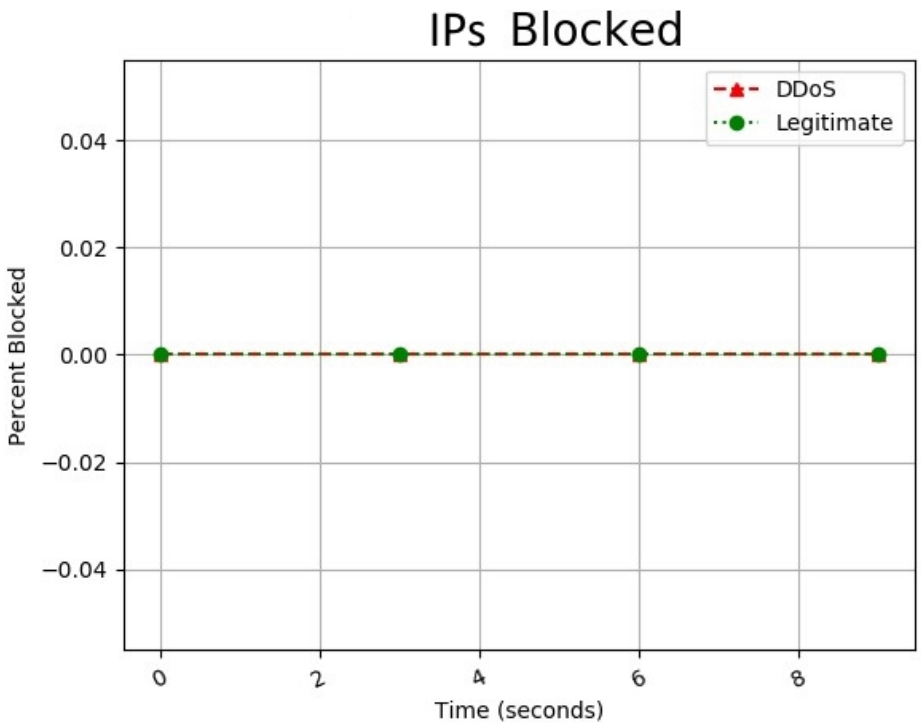

Figure A.10: Case 1 – UDP 10000 Percentage Blocked

Appendix B

CASE STUDY THREE RESULTS

This appendix includes the figures from case study three that were not included in the Results chapter. Included here are the clustered two attacker case of the SYN flood class, the clustered five attacker case of the SYN flood class, and the clustered eight attacker case of the SYN flood class. Also included are the spread two attacker case of the SYN flood class, the spread five attacker case of the SYN flood class, and the spread eight attacker case of the SYN flood class.

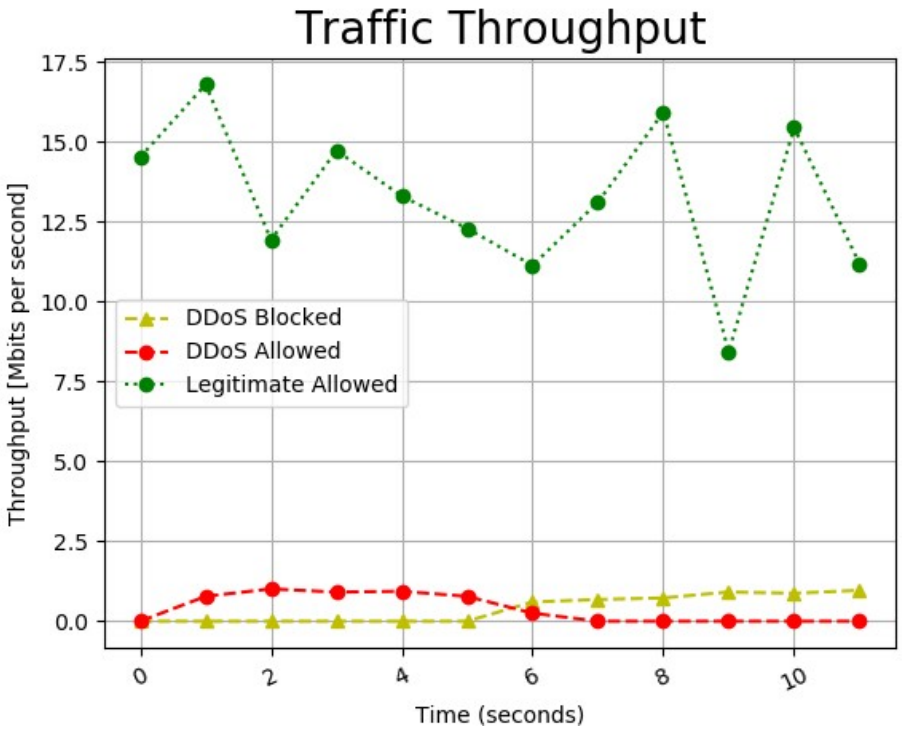

Figure B.1: Case 3 – Cluster 2 Attacker SYN Throughput

Figure B.1 shows the throughput of traffic when the DDoS consisted of two clustered SYN flood attackers.

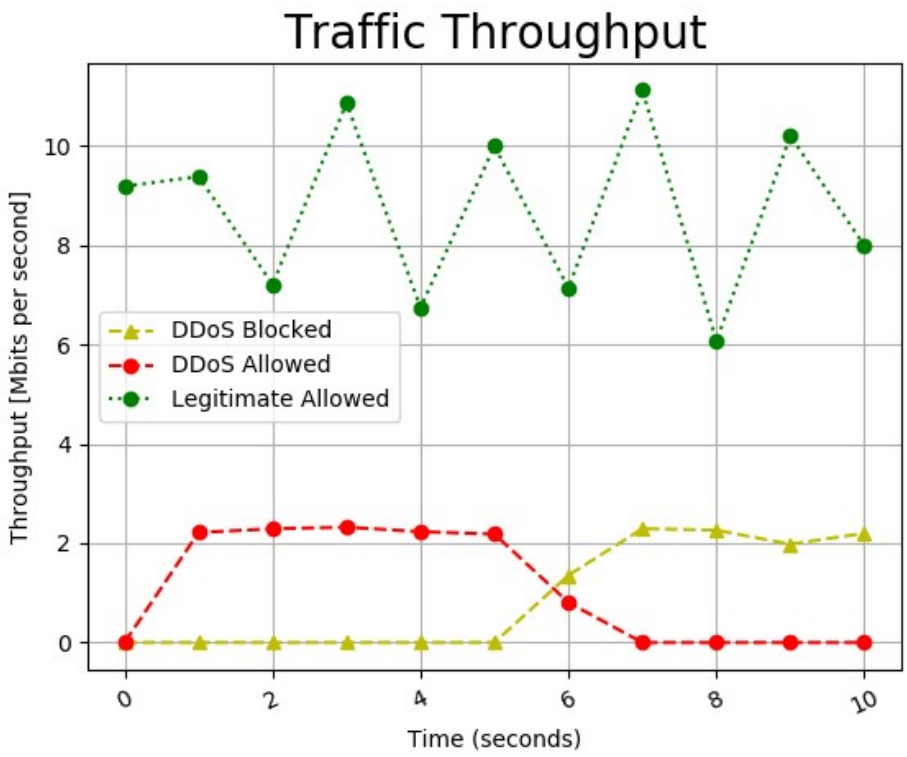

Figure B.2: Case 3 – Cluster 5 Attacker SYN Throughput

Figure B.2 shows the throughput of traffic when the DDoS consisted of five clustered SYN flood attackers.

Figure B.3 shows the throughput of traffic when the DDoS consisted of eight clustered SYN flood attackers.

Figure B.4 shows the throughput of traffic when the DDoS consisted of two spread SYN flood attackers.

Figure B.5 shows only the throughput of DDoS blocked and DDoS allowed when consisting of two spread SYN flood attackers. We include this graph because the legitimate traffic shifts the scale too much to visual the data.

Figure B.6 shows the throughput of traffic when the DDoS consisted of five spread SYN flood attackers.

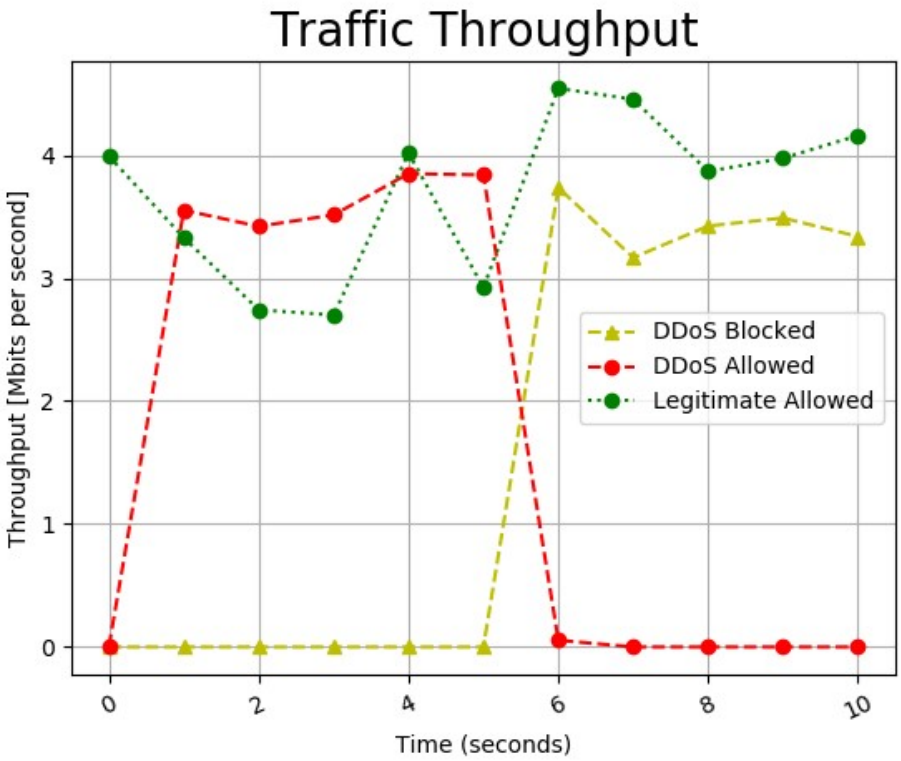

Figure B.3: Case 3 – Cluster 8 Attacker SYN Throughput

Figure B.7 shows the throughput of traffic when the DDoS consisted of eight spread SYN flood attackers.

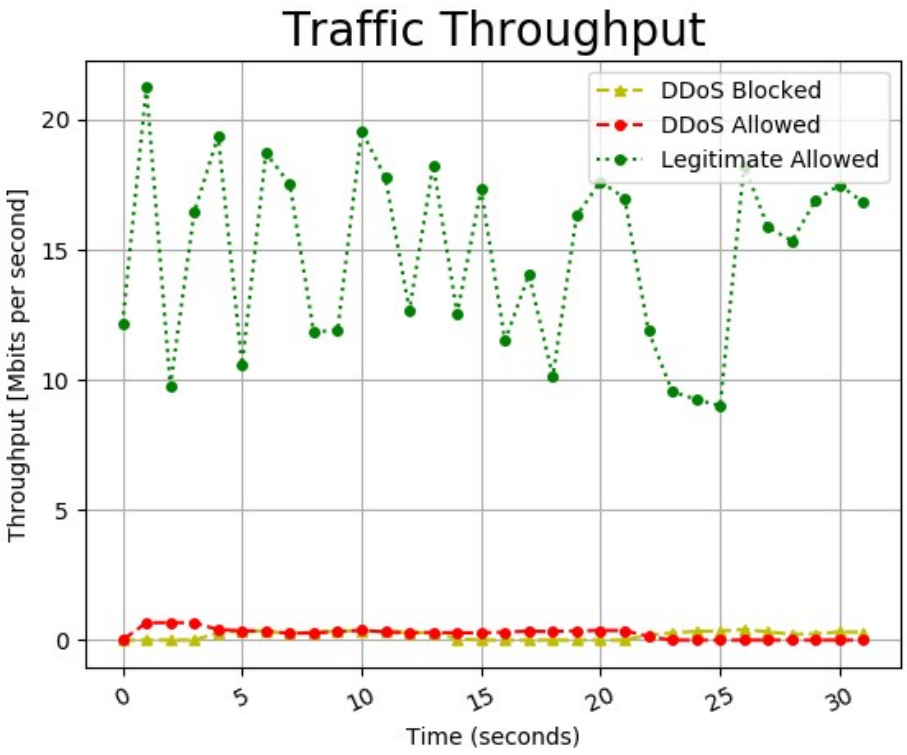

Figure B.4: Case 3 – Spread 2 Attacker SYN Throughput

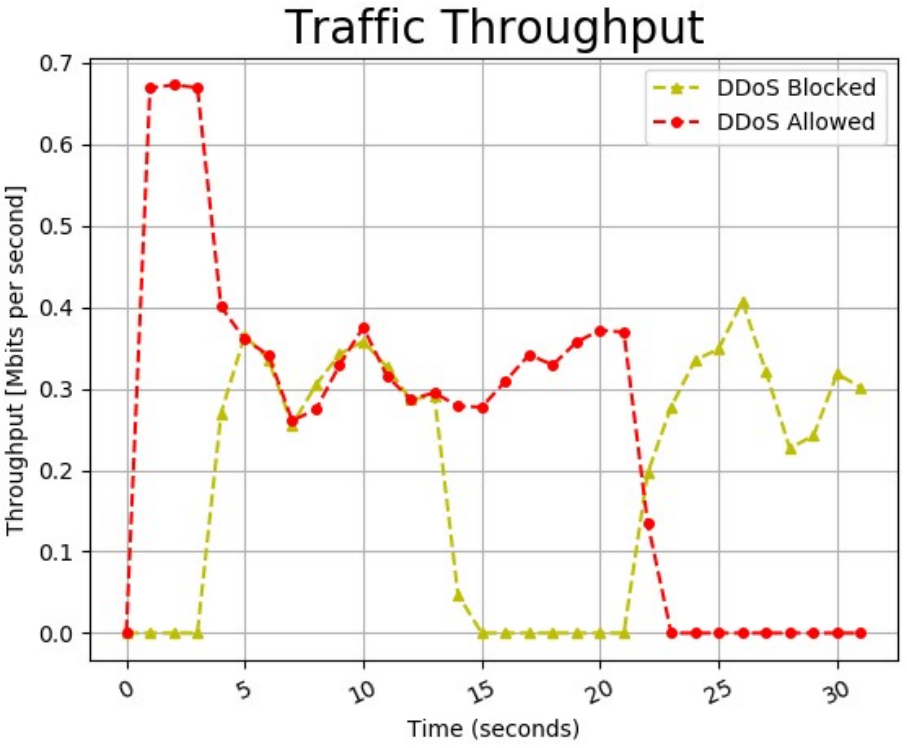

Figure B.5: Case 3 – Spread 2 Attacker SYN (Legitimate Traffic Removed)

Figure B.6: Case 3 – Spread 5 Attacker SYN Throughput

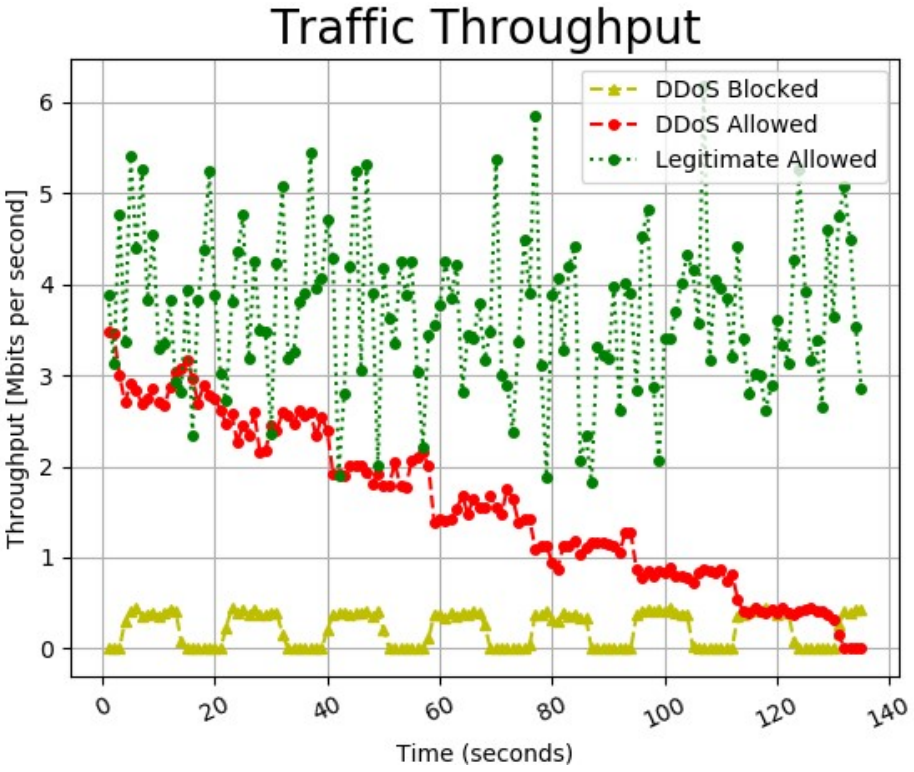

Figure B.7: Case 3 – Spread 8 Attacker SYN Throughput