# ANTIDOSTE: detection and mitigation of network-based Denial-of-Service attacks for a location certification system

**Pedro André Ferreira Teixeira**

Thesis to obtain the Master of Science Degree in

**Telecommunications and Informatics Engineering**

Supervisor(s): Prof. Miguel Filipe Leitão Pardal

**Examination Committee**

**December 2021**

Antes de apresentar o meu trabalho gostaria de deixar aqui os meus agradecimentos, terminar a dissertação do mestrado é um passo muito importante na minha vida que me traz muito orgulho. Portanto gostaria de agradecer a todos aqueles que me permitiram chegar a esta etapa.

Ao Professor Miguel Pardal, por me ter incentivado desde o primeiro dia, por me ter acompanhado nesta jornada, por estar sempre presente, por todas as críticas, conselhos, que motivaram este meu trabalho. Ao Sam, pelo tempo que investiu em mim, pela disponibilidade, pela paciência, pelo apoio que sempre demonstrou.

Ao meu Pai e à minha Mãe que me deram tudo e que sempre acreditarem em mim, pelos seus ensinamentos que contribuíram para que eu chegasse a este momento. À minha irmã que me serviu de inspiração para ser o melhor Irmão possível para ela o que faz de mim uma pessoa melhor. Aos meus Avós, Tios, Primos, que me apoiaram em todas as circunstâncias. Aos meus amigos, por estarem sempre presentes, e pelos momentos que vivemos e que também contribuíram para o que sou hoje. Ao Roxas pela companhia, nos longos dias de investigação e escrita.

E finalmente, agradecer ao leitor por ter interesse no nosso trabalho.

Obrigado!

# Acknowledgments

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2019 (INESC-ID) and through project with reference PTDC/CCI-COM/31440/2017 (SureThing).

## Resumo

Os ataques de Negação de Serviço (DoS - do inglês *Denial of Service*) têm uma longa história e existem vários tipos, que vão desde a exploração de vulnerabilidades que interrompem dispositivos individuais até ataques com muitos clientes que sobrecarregam a capacidade dos servidores, fazendo um *grande número* de pedidos em pouco pempo. Estes ataques têm como alvo a *disponibilidade* do sistema e negam o acesso dos utilizadores legítimos a serviços eletrónicos de que necessitam para trabalhar e jogar.

Neste trabalho protegemos um *sistema de certificação de localização* contra ataques de negação de serviço distribuídos a nível de rede. A nossa solução chama-se antiDoSte– antídoto para DoS – e combina a eficiência de usar *regras feitas à medida* para detetar ataques específicos com a eficácia de usar *aprendizagem automática* dos padrões de tráfego de rede para detectar ataques anteriormente desconhecidos. Avaliamos a solução num banco de ensaio, que criámos, representando todos os nós de rede do sistema, espalhados por uma cidade. Os resultados mostram que o antiDoSte pode detectar ataques DoS, tanto conhecidos como desconhecidos, e pode mitigá-los usando redes locais virtuais (VLAN - do inglês *Virtual Local Area Network*) criadas dinamicamente utilizando mecanismos de programação da configuração de rede (SDN - do inglês *Software Defined Network*).

**Palavras-chave:** Segurança, Certificação de localização, Negação de serviço, Negação de serviço distribuído, Banco de testes

## Abstract

Denial-of-Service (DoS) attacks have a long history and there are many of them, ranging from exploits that crash individual devices to attacks that overwhelm the capacity of servers by having many clients issue a *barrage* of requests. These attacks target *availability* and deny access of rightful users to the on-line services they need to work and play.

In this work we protect a *location certification system* from network-level distributed-denial-of-service attacks. We called our solution antiDoSte – antidote for DoS – and it combines the efficiency of using *custom rules* to detect specific attacks with the effectiveness of using *machine learning* on traffic patterns to detect previously unknown attacks. We evaluated the solution on a test-bed, that we created, representing all the network nodes of the system, spread across a city. The results show that antiDoSte can detect DoS attacks, both known and unknown, and can mitigate them using virtual local area networks (VLAN), created dynamically using software-defined network (SDN) mechanisms.

**Keywords:** Security, Location Certification, Denial-of-Service, Distributed Denial-of-Service, Test-bed

# Contents

# List of Tables

# List of Figures

# List of Acronyms

**DDoS** Distributed Denial-of-Service

**DL** Deep learning

**DNS** Domain Name System

**DoS** Denial-of-Service

**HTTP** The Hypertext Transfer Protocol

**ICMP** Internet Control Message Protocol

**IT** Information technology

**LR** Linear regression

**ML** Machine learning

**OT** Operational technology

**SDN** Software-defined network

**TCP** Transmission Control Protocol

**UDP** User Datagram Protocol

**VANET** Vehicular ad hoc network

**VLAN** Virtual Local Area Network

# Chapter 1

# Introduction

Denial-of-Service (DoS) attacks can make the network resources unavailable for their intended users, temporarily or indefinitely [CDDW19]. An important sub-type of DoS attack is the Distributed Denial-of-Service (DDoS), a coordinated DoS attack that is generated by using many compromised network hosts simultaneously to act as clients and send many requests to a target server [MDML17].

In a recent example, the Russian search company, Yandex, was target by the biggest DDoS ever seen [Reu21]. Yandex said that the requests per minute reached values of 21.8 million on September 5th, 2021. Qrator Labs is a large European Internet security and attack mitigation company that works as a security consultant for Yandex. After this DDoS attack Qrator Labs stated that, at the end of June 2021, signs of a new assaulting force on the Internet have started to appear [Now21], a botnet [Ang17] of a new kind. During this joint research Qrator Labs did in conjunction with Yandex they detected a substantial attacking force, dozens of thousands of host devices, and growing. Separately, Qrator Labs saw the 30 000 host devices in through various attacks, and Yandex collected the data on the 56 000 attacking hosts, i.e. they collected the IP addresses of these devices. However, Qrator Labs presumes that the number of devices is higher, probably more than 200 000 devices, due to the rotation and absence of will, from the attacker, to show the "full force" of an attack at once. Moreover, all the attacker devices are highly capable devices, not simple devices such as IoT [MLZZ15] connected to Wi-Fi. So, we have a botnet consisting of, with the highest probability, capable devices connected through wired Ethernet connections. This statement surprised and worried many, highlighting that there is an urgent need to investigate possible ways to detect and mitigate these DoS/DDoS attacks, in order to prevent attacks with this scale.

To fight DoS attacks, first it is necessary to detect them and distinguish them from benevolent spikes due to a popularity surge or a seasonal effect on traffic. Second, it is necessary to mitigate the effects of the attack. For example, known DoS attacks can be detected with pre-defined rules that are able to detect the specific pattern of the attacks. However, rules are not effective for unknown attacks but Machine Learning (ML) models can be trained to recognize malicious/abnormal packets, likely to be part of a DoS attack. Regarding DoS attack mitigation, one possible way to improve it is to use Software Defined Networking [LLPBOD20] (SDN) architecture, with separate data and control planes. When the monitoring and analysis detect attacks, the SDN controllers can trigger a response system. The SDN controller can define "drop" rules in the SDN switch so that the packets from the malicious host do not arrive to its target.

## 1.1 Objectives

In this work we propose a prototype solution for DoS detection and mitigation, antiDoSte, i.e. antidote for DoS. We created it with the intention of protecting a specific application from DoS attacks, CROSS (loCation pROof techniqueS for consumer mobile applicationS) [MCP20]. CROSS is a distributed location certification system where users are rewarded if they complete touristic itineraries across the city using the mobile application developed for the Android operating system [MDMN12]. The CROSS application uses several techniques to verify the user presence at the locations and *prevent location spoofing* attacks; however, before this work, it lacked protections against DoS attacks, and if the system was made unavailable, users would not be able to collect their rewards and becoming unsatisfied tourists. Therefore, antiDoSte aims to provide a detection and mitigation mechanism for DoS attacks. It uses SDN and ML technologies in conjunction so that they can reinforce each other. The key contributions of this work are:

- DoS attack detection and mitigation for the CROSS location certification system;
- Test-bed for DoS attacks based on the network supporting the CROSS application.

antiDoSte was developed for and tested with the CROSS application, but its design is generic, and the solution can be adapted for other domains of applications.

In terms of security properties [ALRL04], the objective of this work is to preserve *availability*; the focus is *not* on integrity and confidentiality, as the protection of these other properties are addressed in previous work [MCP20]. Also, in this work we only focus on *flooding* attacks [EC18] that affect the data plane of the SDN architecture, since we only prepared our solution to detect these kind of attacks.

## 1.2 Dissertation Outline

The remainder of the document is structured as follows:

- Chapter 2 will start by presenting the target system that we aim to protect with this work. Next, we present reference for network infrastructure management mechanisms, then the present current status of DoS/DDoS. After that we introduce the most popular technologies available to defend against these attacks.

- Chapter 3 describes the Attacker model, as well as our solution proposal to detect and mitigate DoS/DDoS attacks.

- Chapter 4 explains the test-bed we used for the evaluation of our solution.

- Chapter 5 starts by presenting the tests description for our solution and the results of our tests.

- Finally, Chapter 6 presents our conclusions and aims for the future.

# Chapter 2

# Background & Related Work

In this Chapter we start by presenting the target system to be protected by antiDoSte. Next, we present the core network management mechanisms required for our solution and also give an overview of the current status of DoS attacks and of previous works on DoS attack prevention, detection and mitigation.

## 2.1 Target system

CROSS [MCP20] is a location certification system for Smart Tourism [GSXK15]. The users carry their personal devices to interact with existing or newly added infrastructure in emblematic city locations and record sensor data that can later be used to verify location claims. Therefore, in CROSS, the system operation starts when the tourist installs the smartphone application and signs up for an account. Before starting the trip, the application downloads the catalog of locations. During its use, the application logs visits to locations. The location sensing relies on Wi-Fi and interactive kiosks, and leverages the regular scans already performed by the mobile operating system. At the end of the trip, the logging stops, the application submits the collected information to the server, and rewards will be issued, if the conditions have been satisfied. Figure 2.1 represents a tourist that has passed through four location of interest, using the CROSS app, and redeeming his reward when the tour has ended.

The authors of CROSS use 3 strategies for the location proofs: Wi-Fi scavenging, Wi-Fi beacons and interactive kiosks. These strategies provide increasing security against *spoofing* and *Sybil* attacks [Dou02]. However, these strategies do not protect the system from DoS attacks.

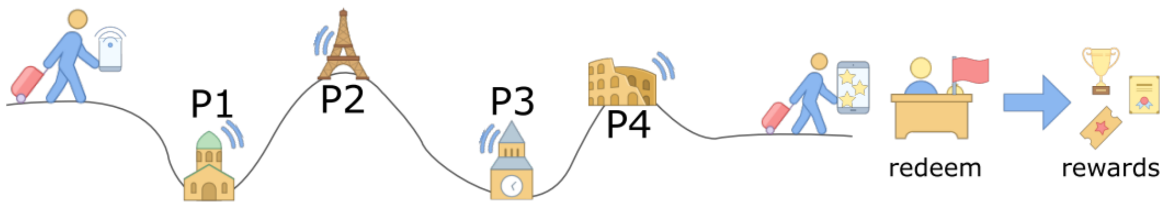


Figure 2.1: User flow throughout a tourism route [MCP20].

## 2.2 Network infrastructure management

Software-defined anything (SDx) is a technology that makes software more ”in command” of multi-piece hardware systems allowing for software control of a greater range of devices. SDx includes software-defined networking (SDN), which is the most recognized technology, and its core feature is the separation of the control plane and data plane in the network. SDN realizes flexible control of network traffic and provides a good platform for the innovation of core networks and applications. The devices in SDN are programmable, and thus the networks themselves are more dynamic, manageable, cost-effective, and adaptable.

In SDN, network intelligence is logically centralized in software-based controllers (the control plane), and network devices such as OpenFlow Switches become simple packet forwarding devices (the data plane) that can be programmed via an open interface, the OpenFlow protocol [NMN+14]. Such SDN works as follows: an OpenFlow switch has one or more flow tables. These tables are used to control packets (e.g., “forward” or “drop”) according to packet-handling rules, called the *flow rules*, and received from a centralized controller. Therefore, according to the controller policy that manages flow tables, the OpenFlow switch can act as a router, switch, or firewall, or exhibit similar functions that depended on the packet-handling rules. For example a VLAN[1] can be created through the SDN controller and the flow rules can be programmed, for that VLAN, in the SDN switch.

Therefore, with this idea of a centralized network control plane and the introduction of this new type of programmability to the network devices, which can streamline network management and enable run-time security strategies. SDN can rapidly react to network anomalies and malicious traffic, such as DoS attacks, by filtering out sources of attack and isolating parts of the network, if necessary.

Acarali et al. [ARCG20] have create a model for DoS attacks and Interoperability in the smart grid. The authors have examined the IT-OT (Information Technology - Operational Technology)

[1]A VLAN is a Virtual Local Area Network. It is a broadcast domain that is partitioned and isolated at the data link layer.

relationship and propose the OT impact chain to qualitatively capture the sequence of escalating impact events that may be caused within the OT.

The integration of the IT and OT results in interoperability requirements. Interoperability is where two or more systems "exchange information and use the information that has been exchanged" [Mom12]. For this to work, systems must operate to a common standard, defined by the data structures, protocols, and communication channels used. This relates to domain interdependency because the management of OT devices depends on accurate and timely data flows through the IT. Therefore, an IT failure can have a cascading effect on the rest of the grid.

There are two types of cascading failure in smart grids:

1. Where an overloaded line fails and trips, and its load is redistributed amongst neighboring lines which may then also become overloaded.

2. Where an IT issue disrupts the flow of sensor data and control signals, causing OT mismanagement.

If parts of the OT then fail, this causes issues in the IT (e.g. communication devices may go offline). Hence, failures 'cascade' both within and across the networks. Fig. 2.2 illustrates this concept.

The OT impact chain is designed to capture the relationship of IT-side DoS attacks and the proliferating OT-side impact they can have. It provides a means for the qualitative assessment of DoS attacks, such that the initial, secondary, and continuing impact of the attack, depending on where it lands in the IT network, can be considered. The main fields defined in the framework proposed by Acarali et al. [ARCG20] are:

- **Attackers**: Devices that transmit DoS attack packets.
- **Attack Point**: IT systems targeted in the DoS attack.
- **DoS Attack**: DoS attack itself, including:
  - **DoS Type**: Flood attack, protocol compromise, etc.
  - **Attack Rate**: Inter-arrival time for attack packets.
  - **Compromise Probability**: Likelihood of target service loss.
- **Initial IT Impact**: Initial impact (devices affected, how many) on IT systems local to DoS targets.
- **Initial OT Impact**: Initial impact of compromised IT systems on directly connected OT systems.

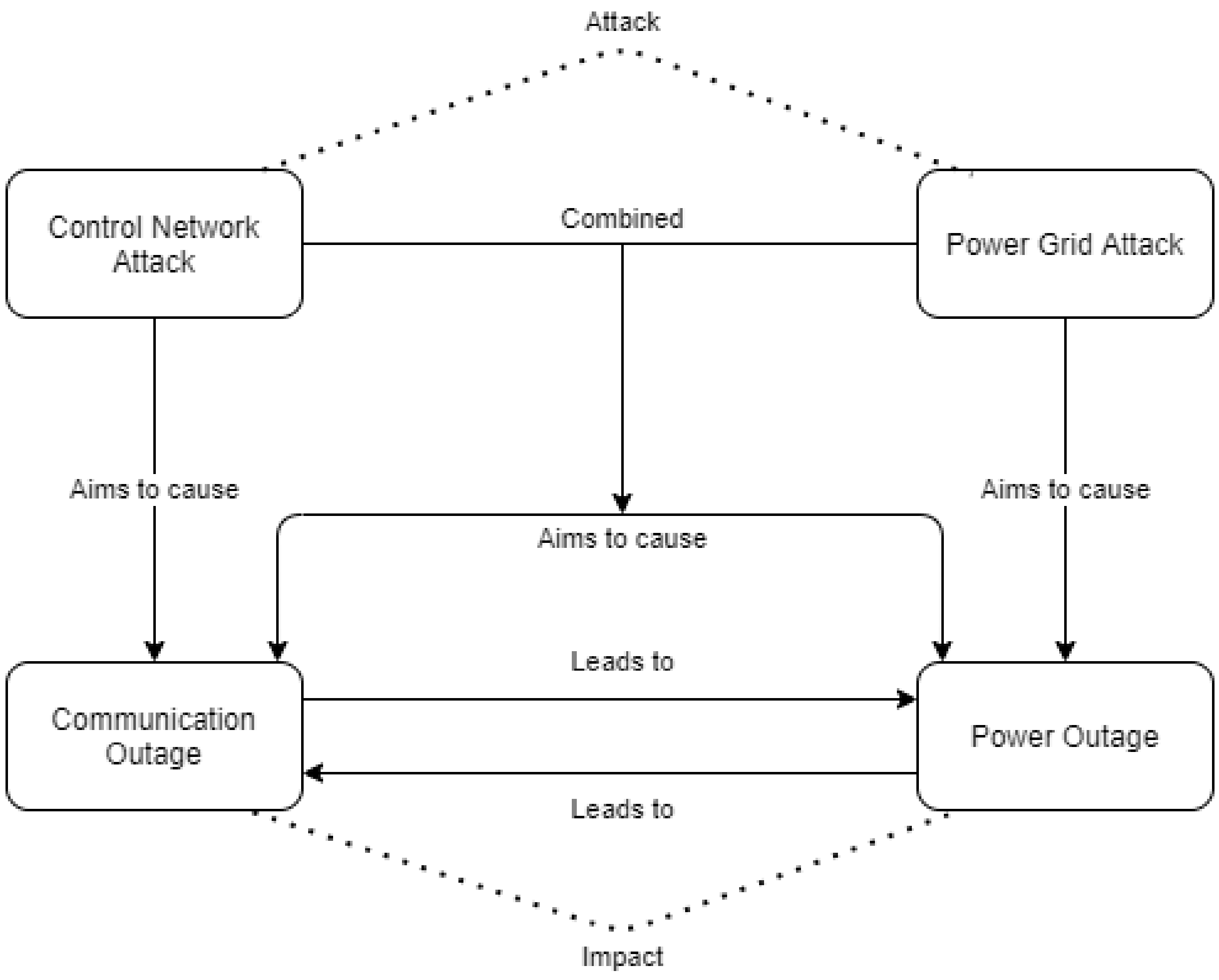


Figure 2.2: Smart grid cascading attack process, adapted from [ARCG20].

- **Consequent Impact**: Rounds of impact in the OT, caused by compromise of OT systems in previous impact rounds.
- **Intra-IT Interoperability**: Assessment of intra-IT connectivity and dependency relationships.
- **Intra-OT Interoperability**: Assessment of intra-OT connectivity and dependency relationships.
- **Inter-IT-OT Interoperability**: Assessment of IT-OT connectivity and dependency relationships.

DoS attacks may block sensor data from reaching the control center, preventing timely and accurate control signals being generated. If the correct adjustments are not made to grid devices, failures may result. DoS may also use crafted packets to manipulate OT system operations to cause device failures.

The OT impact chain enables the characterization of the relationship between the IT and the OT in several ways. Firstly, it encourages users to consider which part of the IT a cyber-attack is targeting. This directly influences the types of problems that will result in the OT. Secondly, it encourages defenders to consider what effect the loss or disruption of a particular sub-system may have on the wider grid via a qualitative assessment of the IT, OT, and IT-to-OT interdependencies. By mapping the relationships between subsystems, Acarali et al. [ARCG20] can better predict impact propagation scenarios, and by considering the worst-case scenarios (i.e. blackouts), the appropriate preventative measures can be taken.

With all this in mind, we able to change the CROSS infrastructure in order to configure it and the proceedings that we must consider when doing so. For example, we need to add controllers to the infrastructure and have the entire flow pass through them so they can detect malicious packets. For that we need to have a switch in every subnetwork working as a router and connect every device, present in the network supporting the CROSS application, to it. This switch will be connected to the controller, making it possible for the controller to set flow rules in the switch flow table, thus allowing to drop packets from malicious devices.

## 2.3 Denial-of-Service attacks

Many applications, such as autonomous vehicles, industry, etc., are dependent on real-time inputs. Network unavailability in such cases can be catastrophic. For instance, when an autonomous vehicle is on road and a DDoS attack targets the server, the vehicle will stop getting sensory inputs to steer.

According to Nikolskaya et al. [NIG+17] there are many ways to perform DoS attacks. For example:

- Transmission Control Protocol (TCP) SYN-flood attacks that exploit the features of the "triple handshake" affecting the availability, in our case, of the CROSS reward server.
- Another popular way to date is Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol-flood (HTTP-flood) attacks. The essence of this attack is to send many HTTP connections to a server. This type of attack can cause a failure of the server, that the attack is targeting, and an overflow of the bandwidth of the communication channel in the subnetwork hosting the server.
- DNS Amplification is also an active attack that have recently been used. The primary technique consists of an attacker sending a DNS name lookup request to an open DNS server with the source address spoofed to be the target's address. When the DNS server sends the DNS record response, it is sent instead to the target. Attackers will typically submit a request for as much zone information as possible to maximize the amplification Effect. Additionally, because the responses are legitimate data coming from valid servers, it is extremely difficult to prevent these types of attacks. Responding to these requests, of vulnerable DNS servers, can cause a failure of the victim machine.

A DoS attack can be launched in different network layers, as said by Salim et al. [SRP19]. For example, a DNS attack can happen in the application and transport layer, a "ping to death"[2] in the 6LoWPAN adaptation layer, and a wormhole attack in the network layer.

Salim et al. also group the DoS attacks in two types. The first is bandwidth depletion attacks. The aim of the attack is to consume all the network's bandwidths. The legitimate users suffer a DoS until the attack is detected and mitigated. The authors divide this attack in two different types: *The protocol exploit* attack aims to consume the resources of the victim by exploiting a feature as a weakness in their system. The attack is done by using transport layers such as a User Datagram Protocol (UDP) or a network layer protocol such as Internet Control Message Protocol (ICMP); *The amplification* attack aims to generate a significant response as the attacker sends small packet sizes of smaller bytes, but by amplifying them, it transmits a vast number of packets to the victim who consumes all of its bandwidth resources; Two common types of such attacks are the Network Time Protocol (NTP) and the DNS amplification attack.

The other type of DDoS attack is the resource depletion attacks. These attacks aim to deprive the user of their memory, CPU, and socket. It is possible by either sending malformed packets

[2]A type of attack that involves sending a malicious ping to a computer.

such as the “Ping of Death” attack or by exploiting the weakness in the victim’s networks, application, or transport layer protocols such as the HTTP Flood. Then again, the authors divide this attack in two different types: *Protocol Exploit* attack exploit the weaknesses in the network layer protocol resulting in the victim exhausting all their CPU and memory resources. Many protocols are exploited such as the HTTP, Session Initiation Protocol (SIP) or the TCP; *Malformed Packet* attack has as an objective to send a deformed packet to the victim which will crash their system. These include Land attack, IP packet option field, “ping of Death” and the Teardrop attack;

## 2.4 Distributed Denial-of-Service attacks

As stated before, a DDoS attack is a coordinated DoS attack [MDML17], and so, we can say that a DoS attack can be accomplished by only one device, while a DDoS attack must be performed by more than one device. One of the reasons why DDoS attacks have been more common is the increased number of IoT devices being deployed [MLZZ15]. There are several reasons for an attacker to use IoT devices to perform a DDoS attack, such as the lack of basic security protocols, no security firmware updates. and the fact that they are continuously connected.

Once the attacker has taken control of a considerable number of IoT devices, it is said that he is in possession of a botnet [Ang17]. A botnet is a number of Internet-enabled devices that a hacker temporarily controls to make simultaneous requests to a server or an array of servers for a specific service [Ang17]. Salim et al. [SRP19] identified three notorious botnets:

- **BashLite**, this botnet can launch DDoS attacks such as UDP and TCP flooding attacks and HTTP attacks with a volume capacity of 400 Gbps.
- **Mirai**, It is far more dangerous than the BashLite and was able to infect 4000 IoT device every hour. This botnet can generate DDoS attacks such as SYN and ACK, UDP flooding, HTTP trafic and DNS attacks.
- **Reaper**, this botnet is a variant of the Mirai code and is even more dangerous. Unlike the Mirai botnet which infected IoT devices using their default credentials, Reaper is known to exploit other security vulnerabilities which are present in the code of the IoT devices.

Apart from the problems that a successful DDoS attack might cause, we need to consider the costs that the owners of these devices, that have been hijacked, will have. Compared to data centers, individual household owners are much more vulnerable and sensitive to their energy bills. More importantly, considering the massive amount of smart home devices on the market, the aggregated attack impact cannot be neglected. According to Gray et al. [GAHC19], the number of IoT cameras and smart appliances in use by 2020 is around 1 billion and 5 billion, respectively. Energy-oriented DDoS attacks (E-DDoS) aim to cost maximum energy consumption on the target side through malicious traffic [TDDL20]. Tushir et al. study [TDDL20] reveals that when these devices are E-DDoS attacks for one month, the approximate increase in the electricity bills can easily reach 253.7 million dollars.

There are two possible scenarios when launching attacks on a smart home: internal and external. In the internal scenario, the attacker has access to the local network. This is possible by hacking the Wi-Fi network or gaining access to a device of the smart home. For the external case, the attacker generates attack packets from outside of the network. It is important to address the following: Firstly, different types of victim devices differ in their hardware design and available services, thereby they respond to the same attack in different ways; Secondly, during an attack, the communication protocol and port state can significantly influence a victim device's response, leading to different service disruptions or energy consumption levels; Thirdly, although it has been recognized that the impact of DDoS/E-DDoS attacks is directly related to attack rates, according to Tushir et al. there is a lack of quantitative analysis on the correlation between attack rates and the resulted service disruptions and energy consumption of victim devices; Fourthly, as IoT devices are resource-constrained in terms of processing power and memory, their behaviors may differ significantly versus the payload size of attack packets.

### 2.4.1 Prevention

Many techniques have been proposed to prevent these attacks from causing major damages in the network, such as Signature-Based Authentication [CWD$^+$17], Blockchain [DKJG17], Software-Defined Anything [YZY18], Machine Learning [DAF18] [AA19] [RS20] and Honeypots [VJ19].

When an attacker is trying to perform a DoS/DDoS attack the ideal is to prevent that attack from happening, so we do not have deal with any possible consequences of the attack.

#### Signature-Based Authentication

One way to prevent DoS attacks is by using a new signature-based authenticated key establishment scheme using the authentication model for IoT applications, as presented by Challa et al. [CWD$^+$17]. To protect the environment from strong replay attack, the authors use both random numbers as well as current timestamps. For this reason, the authors assume that all the entities involved in the environment are synchronized with their clocks. The proposed scheme consists of the following eight phases, namely:

1. System setup;
2. Sensing device registration;
3. User registration;
4. Login;
5. Authentication and key agreement;
6. Password and biometric update;
7. Smart card revocation;
8. Dynamic sensing device addition.

The authors first prove that the proposed scheme provides secure mutual authentication between a user and a sensing device with the help of the widely-accepted BAN logic [BAN89]. Furthermore, it is shown that the proposed scheme is secure against various known attacks informally. In addition, the formal security verification using the broadly-accepted AVISPA tool [ABB$^+$05] ensures that the scheme is also secure against replay and man-in-the-middle attacks. The most important demonstration that Challa et al. do are the protection against:

- Denial-of-service attacks;
- Replay attacks;
- Man-in-the-middle attacks.

Signature-based authentication stops the attacker from creating a botnet, since he needs to prove that he has the right to access the device, and thus making it difficult to perform a DDoS attack. Although this approach provides low communication cost and many other features, we need to ensure that either all devices have this mechanism of authentication, which could be difficult and energy consuming, or have a device that does the authentication for them, which again is difficult because all sub networks must have one. Another aspect that we need to take into account is the fact that all entities involved in this environment are synchronized with the same clock, which is difficult in a real-world scenario.

**Blockchain**

Another way to prevent DoS attacks is by using Blockchain, which is an immutable public record of data secured by a network of peer-to-peer participants [DKJ16]. The first and the most renowned case of Blockchain Technology (decentralized peer-to-peer database) is the Bitcoin cryptocurrency. Blockchain is the technology built on creating and exchanging consecutive information-containing blocks between network peers.

To create a block, some specific nodes, known as miners, try to solve a resource consuming cryptographic puzzle named Proof of Work (POW) It can be said that blockchain is a distributed calculation technology that inputs data in the equally distributed database. This technology can be used to prevent DDoS attacks, as proven by the work of Dorri et al. [DKJG17]. Although, adopting Blockchain in the context of less powerful devices, such as IoT, is not straightforward and entails several significant challenges such as: high resource demand for solving the POW, long latency for transaction confirmation, and low scalability that is a result of broadcasting transactions and blocks to the whole network.

Dorri et al. created a modified blockchain that eliminates the concept of POW and the need for coins. The proposed framework relies on hierarchical structure and distributed trust to maintain the Blockchain security and privacy. The authors ideas are explained in a context of a smart home, but, according to them, the framework is application agnostic and can be applied in other contexts. All transactions must be checked by the miner, which is a device that centrally processes incoming and outgoing transactions to and from the smart home. Similar to existing central security devices, the miner authenticates, authorizes, and audits transactions. So, the attacker cannot directly install malware on smart home devices since these devices are not directly accessible, making it impossible for the attacker to create a botnet. But, if the attacker, somehow, still manages to infect the devices, the attack will not succeed since all outgoing traffic has to be authorized by the miner by examining the policy header. The requests that constitute the DoS/DDoS attack traffic would not be authorized and they would be blocked from exiting the smart home.

Although this technology looks promising, we need to consider that it is a complex technology, it required a device to work as a miner, which is hard to install in all subnetworks and the mine needs to be trustworthy.

### 2.4.2 Detection

However, if it is not possible to prevent the attack from happening, we need to prepare the network environment to detect the attack.

### Software-Defined Networking

SDN provides various techniques for DoS attack detection [WWL$^+$18]. For example, to perform DoS attack detection, Yin et al. [YZY18] have proposed an algorithm that calculates the cosine similarity of the vectors of packets. A Threshold is set, in the controller, to determine a maximum number of packets a host can send with a certain cosine similarity. Thefore, the controller checks the incoming packets (`packet_in`) in each port of the boundary switches and then determines whether a DoS attack has occurred based on the value of the cosine similarity.

### Machine Learning

Various machine learning approaches have also been used recently in the detection of DDoS attacks. Alkasassbeh et al. [AANHA16] have used a database of DDoS attacks which were collected at different network layers. The work focuses on a comparative analysis where 27 features are taken into consideration and machine learning techniques such as Multilayer Perceptron, Random Forest, and Naive Bayes. Zhou et al. [ZLW$^+$18] use a correlation-based feature selection method and choose the 4 most necessary network features in their machine-learning-based DDoS detection algorithm. The authors verify the result of feature selection method by a comparative experiment and compare the detection accuracy of 3 machine learning methods, Naive Bayes, Logistic Regression and Decision Tree.

Machine learning is also a good tool to detect DoS/DDoS attacks originated by botnets since IoT traffic is often distinct from that of another Internet connected devices [DAF18]. For example, IoT devices often communicate with a small finite set of endpoints rather than a large variety of web servers. IoT devices are also more likely to have repetitive network traffic patterns, such as regular network pings with small packets at fixed time intervals for logging purposes. Building on this observation, machine learning techniques can perform data collection, feature extraction, and binary classification for IoT traffic DDoS detection.

Doshi et al. [DAF18] explored two classes of features and analyze why they are relevant to differentiating normal and attack IoT traffic. Stateless features can be derived from flow-independent characteristics of individual packets. These features are generated without splitting the incoming traffic stream by IP source. Thus, these features are the most lightweight. Stateful features capture how network traffic evolves over time. There is inherent overhead in generating these features, as the authors split the network traffic into streams by device and divide the per-device streams into time windows. The time windows serve as a simple time-series representation of the devices' evolving network behavior. After testing this features the authors demonstrated that the stateless features greatly outperformed the stateful feature. This result was expected since the differences in the cumulative distributions of normal and attack traffic were more pronounced than those of the stateless features. This result suggests that real-time anomaly detection of IoT attack traffic may be practical because the stateless features are lightweight and derived from network-flow attributes. Incorporating stateful features nonetheless improved accuracy compared to classification with the stateless features alone.

### Machine Learning in VANET

Machine learning can also be implemented in Vehicular ad hoc network (VANET). VANET is a special wireless ad hoc which is subset of Mobile Ad hoc Network (MANET) and IoT application [JNZ16]. The main objective of VANETs is to concentrate on the safety of drivers, passengers, and the vehicle itself by sharing confidential information about traffic and accident between vehicles. Based on the nature of VANET which is high mobility, large size network and frequent exchange of information between nodes, if any node exchange a malicious information, the whole network can be interrupted and compromised and thus becoming a life crucial. VANET has number of attacks under integrity like Masquerade, Black hole, and Replay attacks. Furthermore, some attacks targeting the VANET availability such as Jamming, DoS, and DDoS attacks. In VANET a large amount of data should be processed fast from each node. And to handle the huge amount of data and process them with minimum time, several machine learning techniques have been applied to cope up this issue in VANET environment.

Current research proposes different solutions to securing VANET environment. Some examples are, Multivariant Stream Analysis (MVSA) for detecting and mitigating the DDoS attack on VANET was proposed in [KMNA+18]. Another proposed scheme was introduced in [SKKS16] to secure VANET form DDoS attack. This work focused on detection and prevention DDoS by using the available resources and with least overhead on VANET. Machine learning has also been proposed to secure VANET's.

Other examples include the use of machine learning for anomaly detection on VANET that was presented in [YGL+18]. Yu et al. designed a framework to effectively detect the DDoS attack in Software Defined Vehicular Networks (SDVN) in a fast manner. Another effort using machine learning to detect a specific type of DDoS attack in VANET, namely the RF jamming attack was presented in [KA18]. This approach uses unsupervised ML [RS20], in contrast with the previous ML models presented for VANETs that used supervised models [RS20]. The proposed method was applied to three different real-life scenarios, two of them with a moving jammer present and one with interference only. In the simulation, Karagiannis et al. collected the data in different situation of jammer, the first one while the jammer was active and a second when it temporarily idle. The experimental result shows the capability of establish the crucial role of the relative speed and its variations in detection jamming efficiently.

According to Alrehan et al. [AA19] machine learning has not been applied extensively to detect and prevent DDoS attack in VANET. The authors also state that there is a need to propose a solution to detect and prevent DDoS attack in VANET and SDVN based on machine learning using a dataset from VANET environment which is a point often overlooked by studies in this field.

### Machine Learning plus Software-Defined Networking

SDN and ML can be combined to deal with DoS attacks [PPC20, NSJ16]. In particular, Ravi et al. [RS20] have demonstrated the effectiveness of using SDN in conjunction with ML by crafting a learning-driven detection mitigation mechanism (LEDEM). The authors used a semi-supervised [RS20] ML model, the semi-supervised deep extreme learning machine (SDELM) model, which is a mixture of unsupervised, where unlabeled data is used, and supervised, where labeled data is used to train the detection of DoS traffic.

### Honeypots

A honeypot is used to intentionally lure attackers with the purpose to capture the malware properties and its style of invading the security of devices by recording the whole information about it in log files [PSY+15]. Therefore, honeypots might be good to detect DDoS attacks, because once the attacker has fallen for this trap the victim will know that he is being attacked in advance, allowing the victim to know the attacker IP, and so setting drop rules for that IP. Honeypots can help to improve the machine learning model by using the information gathered from the attack, as the work of Vishwakarma [VJ19] demonstrated. There are different types of Honeypots available to be used by various applications. They can be classified depending on the level of interaction it allows with the attacker. The level of interaction depends upon the amount of data that needs to be get collected. Therefore, it is categorized into Low interaction honeypots and High interaction honeypots [ATN17]. They can also be classified on the basis of objective it wants to attain i.e. either they can be used for carrying out any research to get knowledge of possible threats and shortcomings in the system called as Research Honeypots, or they can be used for protecting the companies assets from the attacks in real time to improve the overall security called as Production Honeypots.

Zero-Day attacks are caused by different possible variants of malware infections that yet not have been identified entirely for creating a complete DDoS defense against it [MMD13]. Thus, honeypots are quite effective in dealing with Zero-Day DDoS Attacks without compromising Smart devices [WSK18].

In the proposed solution Vishwakarma et al. have used a honeypot framework to catch several malware installations attempts into the IoT device. The collected information in the form of log files can be used as input to the machine learning model we are using for training purpose. According to the authors the advantage of using honeypot over datasets to train the model is that in this way it would be able to learn by unknown variants of malware families instead of using only limited known data. Vishwakarma et al. proposed solution can be implemented by using an IoT honeypot inspired by the ThingPot [WSK18] which is an IoT virtual honeypot capable of catching various botnet binaries by emulating different IoT communication protocol along with entire IoT platform behaviors. To keep it isolated from the original IoT platform, the virtual box should be used to deploy it over every IoT device in a network. Since due to the IoT constraints, it is not possible to implement classifiers on each device, it should be implemented on the router level. Also, the amount of traffic coming to a particular IoT device is insufficient to perform any training over a machine learning model. To generate an adequate amount of IoT traffic, any IoT network simulators can be used. IoT simulators are known for generating an IoT environment for testing any IoT-based application and add storage facility using cloud if required.

As said before this way of training the ML model is good to defend against Unknown/Zero-day attacks. However, we need to consider that the attacker needs to fall for the honeypots, requiring a good simulation from the honeypots so that the attacker cannot distinguish them from a normal devices. This simulation might not be easy to perform and, even if it is, there is no guarantee that the attacker will hijack the honeypot instead of a normal device. But this idea of a self-improving solution is something that we must take into account.

### 2.4.3 Mitigation

After an attack has been detected, it is important to have a robust response mechanism that can reduce or eliminate the impact of a DDoS attack. Therefore, when it comes to the mitigation of these attacks the two most used types of strategies are, the Filtering [YZY18] [RS20] and the Rate Limitation [MXSJ17].

#### Filtering

Filtering is a technique that prevents the malicious traffic from arriving to its destination. For example, in a SDN environment, we could filter the malicious traffic by setting "drop" rules for the malicious packets as done in the work of Yin et al. [YZY18]. Therefore, when an attack is detected, a new "drop" rule is added to the flow table to drop all the packets originating from the malicious devices.

Another great filtering technique is the one proposed by Ravi et al. [RS20] since they consider the possibility of the attacker having a large botnet [Ang17], composed of many devices. Therefore, setting individual rules for the malicious devices will saturate the limited flow table space in switch and lead to overflowing issues in the data plane of the SDN [SG13]. So, Ravi et al. have proposed a mitigation strategy that prevents saturation; they create a VLAN in the switch, and then define "drop" rules for it and group all the malicious hosts in that VLAN. This is one of the strategies used in our proposed approach and will be further explained in the next Chapter.

#### Rate Limitation

We need to consider there is the possibility that the detection solution produces a large quantity of false negatives or cannot identify a precise mark between the legitimate and malicious traffic. And so, it is reasonable to apply a rate limitation instead of filtering [MXSJ17]. For example, a product is sponsored by a famous person which makes that the server, where the website of that product is allocated, receives an abnormal number of requests. In the subnetwork, where the server is, exists a detection strategy that states that this traffic is not normal. However, it does not make sense to filter this traffic since it is legitimate traffic, and so a rate limitation strategy is more appropriate for these cases.

## 2.5 Summary

During this Chapter we presented the target system for this work, CROSS, a location certification system that has the objective of rewarding tourists that have passed in emblematic city location. This system has some security, but it does only focus on confidentiality and integrity leavening it vulnerable to attacks that affect the availability such as DoS attack.

We also presented the network infrastructure management that we need to apply to the CROSS application. This management passes by using SDN so we can have a better control of the network. We presented a threat model proposed by Acarali et al. [ARCG20] that focus on DoS attacks and Interoperability in the smart grid.

After that, we exposed the current state of DoS/DDoS attack by presenting the most common attacks, such as TCP-SYN flood attack, HTTP flood and DNS amplification, the fact that they can be perform in different network layers, such as the DNS attack that can happen in

Table 2.1: Solutions capabilities.

| Solution | Prevention | Detection | Mitigation |
|---|---|---|---|
| Blockchain | X | | |
| Signature-b. authentication | X | | |
| Software Defines Anything | | X | X |
| Machine Learning | | X | |
| Honeypots | | X | |

the application and transport layers, the types of DoS/DDoS attack, bandwidth depletion and resource depletion. We mention that IoT devices are one of the reasons why DDoS are becoming more common, since these devices are deployed with lack of security protocols, and they are continuously connected. Also, we consider the costs of the hijacked devices make to the owners, creating a new type of attack an Energy-oriented DDoS.

Following that, we presented various technologies capable of dealing with DoS/DDoS attacks. Table 2.1 provides a summary of the capabilities of each approach. As we can see, these technologies have their weak and strong points. Blockchain focus on the prevention of these attack by controlling every transition made between machines, making it harder for an attacked to hijack a device. Signature-based authentication approach establishes a mechanism that proves the devices ownership, preventing an attacker from creating a botnet. SDN, Machine Learning and Honeypots are technologies capable of detecting DoS attack, SDN can do it thought algorithms, for example by calculating the cosine similarity between packets as demonstrated before, ML can do it by being trained to recognize the attack, and Honeypots require that the attacker falls for their trap. Finally, mitigation can be performed by SDN since after the attack detection the flow rules of the switch, present in the same subnetwork of the malicious device, can be updated automatically in order to “drop” or limit that device‘s packets.

# Chapter 3

# antiDoSte

antiDoSte is a prototype solution that was created to protect CROSS system from DoS/DDoS attacks. antiDoSte is a hybrid solution that combines SDN and ML techniques to provide better DoS detection and mitigation. This way, it is possible to use the strong points of some of the detection techniques to make up the weak points of the other techniques. Our solution uses SDN to provide deep packet inspection on the network controllers, so that necessary *features* from the packet are extracted to differentiate malicious from normal network traffic.

In this chapter we present the Attacker model and then the detection and mitigations strategies of antiDoSte.

## 3.1 Attacker Model

Before explaining our solution, we need to specify what an attacker can and cannot do in our test environment. For an attacker to perform a DDoS attack in our environment he needs to control many malware-infected devices, a botnet. In this model we consider that the attacker can hijack any Smart device in the network, i.e., the attacker is capable of expanding the botnet as he pleases, and Send different packets to the victim i.e. the similarity between the packets is small. This is important since it would be easy to detect and mitigate a DoS attack by simply calculate the similarity between packets and setting drop rules for those kind of packets, as we have seen in previous work.

Also, in this model we consider that the attacker is not capable of changing the MAC addresses of the devices in his botnet every time a packet is send and verify if one of his malware-infected Smart devices has been cut off the network. For the first point we consider that the attacker does not do that because it would waste time and slow down the DoS attack. The second point is because he cannot see if the lack of response from the victim is due to the

Table 3.1: Types of DoS an attacker can perform.

| Attacks | Mitigated |
|---|---|
| TCP-SYN | X |
| ICMP | X |
| UDP | |
| HTTP | |
| DNS | |

DoS attack or because the device has been cut off from the network.

Some possible attack scenario that the attacker can do:

- The attacker uses a device present in one subnetwork to perform a DoS aiming the application server, that is in a different subnetwork.
- Perform a DoS from one device to another in the same subnetwork.
- Use all the device in a sub network to perform a DDoS attack to the application server or a tourism based device present in other subnetwork.
- And it is possible to use all the devices present in the network to perform a DDoS attack to the application server.

Another important aspect about the attacker is that he only performs flooding DoS/DDoS attacks, since these types of attack are the more common types of DoS attacks and the easier to perform [SRP19]. Also, another reason for the attacker to perform only flooding DoS/DDoS attacks will be explain in chapter 3.2.2. Table 3.1 presents the types of flooding DoS attacks the hacker can do with which are mitigated instantly after the first detection and which require further analysis. This last part of the sentence will be explained in the following chapters.

## 3.2 Detection strategy

antiDoSte has two approaches that work together on detecting DoS/DDoS attacks: threshold detection and machine learning.

### 3.2.1 Threshold-based detection

The first strategy passes by setting a *Threshold*. We are able to use this method thanks to SDN, which allows us to configure traffic rules in the controllers. This approach limits the number of packets a client device can send over the network. It exploits the weakness of some attacks. For example, the TCP-SYN flood attack that uses the 3-way handshake to flood the server with

TCP-SYN packets. In a normal scenario, a client is expected to send an TCP-ACK packet after sending TCP-SYN packet, if there is no problem in the network. We can take advantage of this weakness and set a maximum threshold for the number of TCP-SYN packets send by the client without TCP-ACK packet. First, we set a predefined threshold N and then when an TCP packet is received on the controller we check if the packet contains an SYN or a ACK flag. If the packet has an SYN flag, then the counter of that client is increased and if the packet has an ACK flag, the counter of that client is decreased by one. The reason why the counter is decreased by one and not reset to zero is because of the attacker might know of this mechanism beforehand and when the counter is approaching the predefined threshold, he could just send TCP-ACK packet, resetting the counter, and continuing with the attack, making it possible for the attacker to counter this strategy. When the counter reaches the maximum threshold the mitigation strategy is applied. Figure 3.1 shows the flow diagram of this approach.

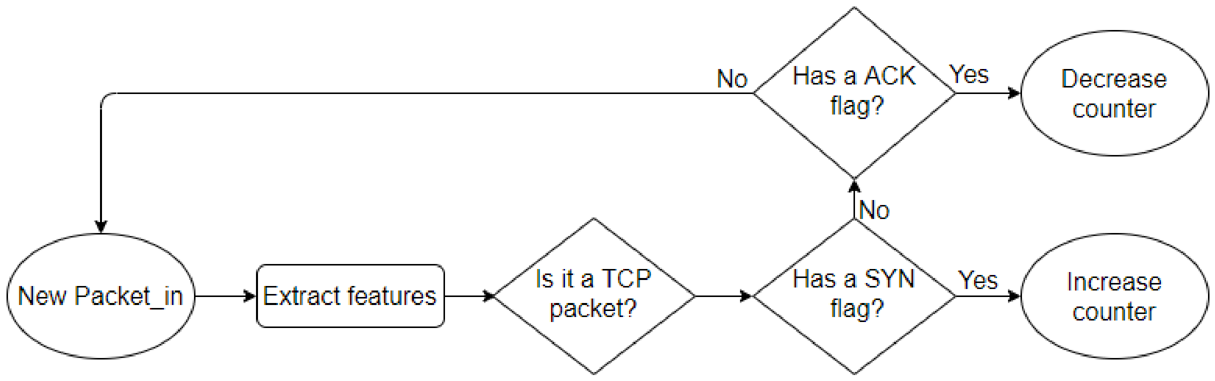


Figure 3.1: Flowchart for Threshold-based attack detection.

However, despite being the fastest approach to detect an attack, as chapter 5.2 will demonstrate, this technique requires a considerable knowledge of the attack, since we need to know a weak point that we can exploit to prevent the attack. Therefore, this approach is suitable for simpler attacks that are easier to find a weakness so it can be explored, such as the previously mentioned TCP-SYN flood attacks. The implementation of this approach, applied to TCP-SYN flood attacks, is shown in Algorithm 1.

**Algorithm 1:** Threshold mitigation approach

```
Result: Increase/Decrease counter from host or mitigate attack
$Threshold \leftarrow N$;
$tracker \leftarrow \{\}$;
if PacketIn.type != TCP then
    return
end
if PacketIn.flags != SYN or ACK then
    return
end
if PacketIn.flags == SYN then
    if PacketIn.src not in tracker then
        tracker[PacketIn.src] = 1 #src means packet source IP
    else
        tracker[PacketIn.src] += 1
        if tracker[PacketIn.src] == Threshold then
            Apply mitigation strategy
        end
    end
end
if PacketIn.flags == ACK then
    if PacketIn.src not in tracker then
        tracker[PacketIn.src] -= 1
    end
end
```

### 3.2.2 Machine Learning-based detection

The second detection approach is based on ML techniques. In our solution we use two different models, Deep Learning (DL), which is able to detect attacks, and Logistic Regression (LR), which is able to recognize normal behavior. The DL model allows us to predict the network status with a great certain, for prepared attacks, while the LR model allows us to detect abnormal behavior such as, unprepared attacks or new attacks, i.e. Zero-day attacks.

#### Training

To create both models we used Google colab which is a cloud service based on Jupyter notebooks [CDNN$^{+}$18]. Google Colaboratory (Colab) is a project that has the objective of disseminating machine learning [Goo18]. Colaboratory notebooks work as a Google Docs object: can be shared and users can collaborate on the same notebook. Colaboratory provides either Python 2 and 3 runtimes pre-configured with the essential machine learning and artificial intelligence libraries, such as TensorFlow, Matplotlib, and Keras. This Google service provides a GPU-accelerated runtime. The Google Colaboratory infrastructure is hosted on the Google Cloud platform and can be used to accelerate deep learning. Jupyter notebook is an open-source and browser-based tool that integrates interpreted languages, libraries, and tools for visualization [PG07]. Using this technology makes it easier to share and replicate scientific works since the experiments and results are presented in a self-contained manner [RPGB17].

In this work we use a DL model that was initially proposed by Yuan et al. [YLL17] to detect DoS/DDoS attacks. We trained the model with the data we have gathered from simulating the CROSS traffic, using 50 000 entries for the normal traffic and another 50 000 entries for the attack. The data features can be seen in Table 3.2, alongside some examples and their types. Our model is supervised since we label all the data that we capture, i.e. during the models training we also provide what the features represent (benign or attack traffic). We classify three types of fields: text, float and numerical fields. Then transform the representation for the content of text fields, turning then via Bag of Word (BoW) method to deal with a large dataset and conquer the memory scalability, we apply hashing method in the conversion of BoW [WDL+09]. After feature transformation, we get a m*n' matrix, where m indicates the number of packets and n' indicates the number of new features after transformation. In order to learn patterns in both long and short term, we use a sliding window to separate continuous packets and reshape the data into a series of time windows with window size T, which was chosen based on Yuan et al. work [YLL17]. The label y in each window illustrates the last packet. After reshaping, we have a three-dimensional matrix with shape (m-T)*T*n'. In this way, we change the features from conventional packet-based to window-based, by which we can learn network patterns from both previous (T-1) packets and current packet. This method allows us to have better results in metrics such as, accuracy, precision, and recall, when dealing with flooding attacks. However, If the attack performs a DoS/DDoS attack with a single packet, by sending a malformed packet, it is possible for the model to no detect the attack. This has to do with the number of benign packets being higher than the number of malicious packets.

After that, a Bidirectional Recurrent Neural Network (RNN) is designed in the model. Each direction includes two sequence-to-sequence RNN which help us trace the history from previous network packets. We use both Long Short-Term Memory Neural Network (LSTM), which is a specific architecture of RNN, and Gated Recurrent Unit Neural Network (GRU), which is a

Table 3.2: Features extracted.

| Field | Field Example | Field Type |
|---|---|---|
| frame number | 1 | Numerical |
| frame.len | 805 | Numerical |
| ip.protocol | tcp | Text |
| ip.ttl | 127 | Numerical |
| tcp.srcport | 2090 | Numerical |
| tcp.dstport | 443 | Numerical |
| tcp.syn | 1 | Numerical |
| tcp.ack | 0 | Numerical |
| tcp.rst | 0 | Numerical |
| time | 0 | Numerical |
| http.version | 1.1 | Float |
| http.type | GET | Text |
| http.request | synchronize with server | Text |

gated mechanism of RNN, in this work. In specific, LSTM aims to overcome vanishing gradient problem of RNN and uses a memory cell to present the previous timestamp [HS97]. GRU is a simpler variant of the standard LSTM and can be trained faster because of fewer parameters. Current modified LSTM usually includes three gates in each cell: input, forget, and output. They are calculated as follows:

$$
\begin{aligned}
it &= \sigma(Wi * [ht-1, xt] + bi),\\
C't &= tanh(WC * [ht-1, xt] + bC),\\
ft &= \sigma(Wf * [ht-1, xt] + bf),\\
Ct &= ft * Ct - 1 + it * C't,\\
ot &= \sigma(Wo * [ht-1, xt] + bo),\\
ht &= ot * tanh(Ct),
\end{aligned}
$$

Where xt is the input at time t, Wi, WC, Wf, Wb are weight matrices, bi, bC, bf , bo are biases, Ct, C´t are the new state and candidate state of memory cell, ft, ot are forget gate and output gate. We design 64 neurons in each cell and model the neuron's output f as a nonlinear activation function:

$$f(x) = tanh(x).$$

We concatenate output from layers in both directions and connect it to latter layer. Fully Connected Layers are designed followed by Recurrent Neural Network. We set the output function as Sigmoid function:

$$\frac{1}{1+e^{-x}}.$$

The deep learning model uses Sigmoid function to indicate the prediction of the last packet in the whole sequence. To capture the local information and simplify the deep neuron network, we try to stack 1-Dimensional Convolutional Neural Layers before Recurrent Neural layers with Rectified Linear Unit (RELU) as activation function [NH10]. Kernel size of Convolutional Neural Layers is 3 with stride 1. Every two RNN layers and every fully connected layer are followed by a Batch Normalization Layer to accelerate deep neuron network training [IS15].

To train and deploy the DL model we used Python libraries such as, Tensorflow [ABC+16] which is a machine learning system that operates at large scale and in heterogeneous environments. TensorFlow uses dataflow graphs to represent computation, shared state, and the operations that mutate that state. Another library used was Keras [Ker]. Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.

The DL approach is easier to train, in comparison with the Threshold-based approach, since we only need a dataset of the attack for training, while the previously presented approach requires manual intervention to create or improve rules. However, the DL approach is the slowest, as Chapter 5.2 will demonstrate. Therefore, the thresholds approach is used for more simple attack, since it requires that the attack has a weakness, as mentioned previously, and so we need more knowledge of that attack. While the DL model is used to detect more elaborated attacks that require a more complex analysis, since these attacks could take more time to detect a weakness or even might not have a weakess.

#### Approach

DL technology has been previously applied on traffic classification [Wan15]. The DL model is specialized to detect certain attacks, i.e., we provide data for a certain type of attack so that it has high accuracy when detecting that attack. In this case, we chose the *ICMP flood* attack.

The LR [KDG+02] model was specialized to recognize normal behavior of the traffic. In this case we use less entries from each attack, when comparing to the DL model, because we want the LR model to have a general knowledge about what is not normal traffic, in contrast to the DL model where we want the model to know the attack. Once the LR model detects unexpected behavior, the mitigation strategy is applied. However there is also another purpose for this approach, which is to improve the previously mentioned detection approaches, i.e., Threshold and DL. When the LR model detects abnormality and the other two approaches do not, a log/dataset is generated from the packets originated by the attack to make the attack features known to the other two approaches. Once the log is generated, it is analyzed by a network administrator to verify if it is really an attack, since it might be a peak in traffic that LR model does not recognize as normal traffic. If the traffic corresponds to an attack, we first apply the log/dataset to train the DL model, which is faster to prepare for attack detection, and so we have a more reliable way to detect an attack, than the LR model. Meanwhile, we keep studying the log to create new rules for the Thresholds, so we can detect the attacks faster than the DL approach.

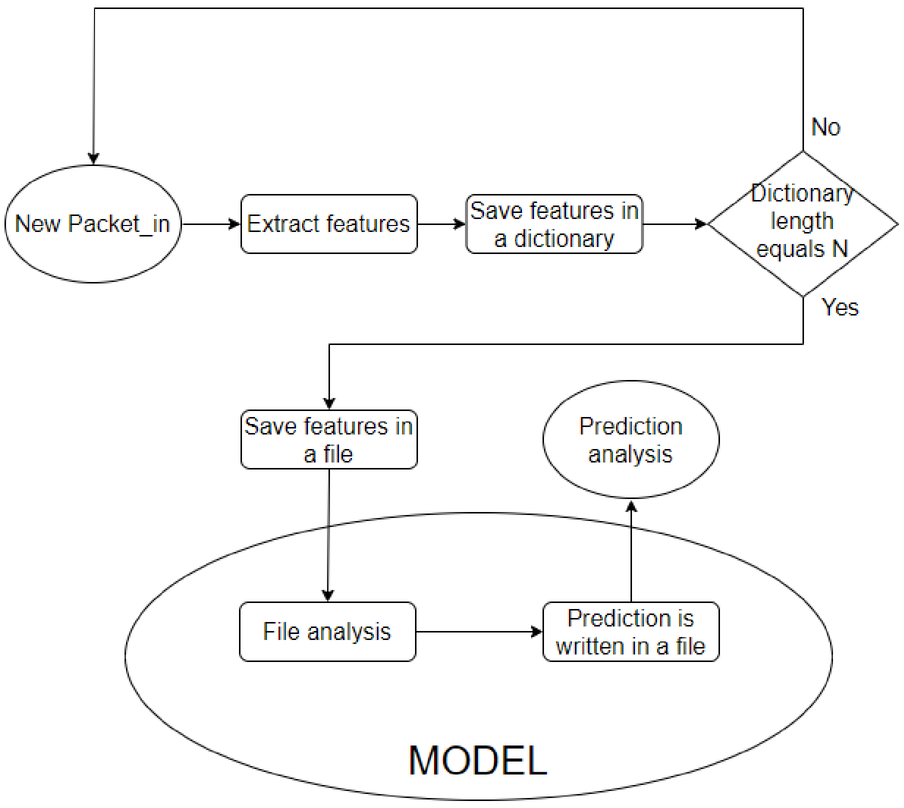


Figure 3.2: Flowchart for the DL and LR attack detection.

Therefore, for the *DL* and *LR* approaches, represented in Figure 3.2, the features are saved in a dictionary where the index is the source MAC, and the value is a list of packet features. Once the controller has received a pre-determined number of packets from that host, represented as N in the figure, these values from the dictionary are then saved in a file so the model can determine if an attack is happening. After the model has stated its prediction, the application running the model creates a new file and sends it to the controller.

And so, ANTIDOSTE takes the advantage of the two detection approaches and creates a hybrid detection to analyze the traffic simultaneously so that the detected attacks get to the mitigation process as soon as the attacks get discovered by each approach. In summary: Threshold applies rules for specific attacks, DL recognizes attacks, and LR recognizes deviations from normality.

## 3.3 Mitigation strategy

The antiDoSte mitigation strategy is based on the strategy proposed by Ravi et al. [RS20], and is used for the *Threshold* and *DL* approaches. The controller starts by creating a VLAN, if it does not exist on the SDN switch in the same sub-network as the malicious host. Then the source and destination MAC addresses are added to that VLAN with "no flow" rules, so that the packets get dropped. There is only one VLAN per sub-network which is responsible for every malicious device in that sub-network, making only one entry in the switch flow table needed to drop all the packets from all the malicious devices in one sub-network. This is done to avoid overloading the flow tables of the switches, as mentioned in Chapter 2.4.3. In Figure 3.3 it is represented the mitigation strategy after the increase of the counter for the *Threshold* approach, from Figure 3.1, and for the *DL* approach after the prediction analysis, from Figure 3.2.

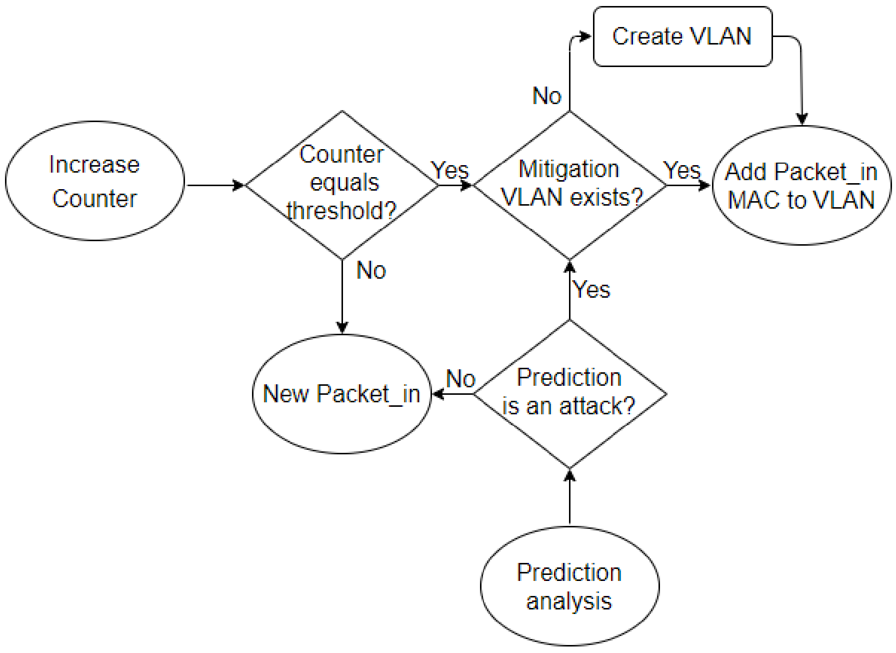


Figure 3.3: Mitigation strategy for Threshold and DL approaches.

The LR detection approach also uses the previously mentioned mitigation strategy. However, first it needs to apply a new strategy to minimize the possibilities of what was detected was not an attack. It might be just a high peak in traffic, as mentioned previously, or bad prediction from the LR model. Therefore, this new strategy consists in setting a certain number of times a host can have an unexpected behavior (e.g. three) and so once the LR model detects an abnormal behavior from that host, it is recorded that the host has committed one fault. Whenever the host has committed three faults (the specific total of faults can be adjusted), the mitigation strategy is applied and the packets from that host are recorded into the log/dataset and dropped. However, it is possible that the unusual traffic is coming from a non-malicious host that just had, for example, a high peak in traffic for a longer period of time. In those cases, a network administrator is responsible for analyzing the log/dataset, to remove the host from the VLAN and re-establish the flow rules to what they were before the mitigation. There is also the possibility that the host, after committing one or two faults, starts to behave normally. In these cases, the number of faults can be decreased after a timeout or if the model detects that the host has behaved normally a number of times in a row. For example, the host has two faults, but the model detects that he had a normal behavior two times in a row and so the number of faults is decremented by one. If this happens again then the host will not have more faults.

For this mitigation strategy the model needs to detect the attack a predetermined number of times (three in our example), making the detection time become more relevant than the accuracy. Being this the reason why we use a LR model instead of a DL model, since the DL model is the slowest at detecting an attack, as Chapter 5.2 will show. Figure 3.4 represents the mitigation strategy for the *LR* approach, after the prediction analysis, where N is the number of faults a device can do and M is the number of consequent well behavior detected, from a device, before reducing the number of faults.

## 3.4 Summary

In this chapter we introduced our *Attacker model* that defines the attacker capabilities. These include: hijacking any number of smart devices present in the network, do a DoS/DDoS attack with packets with little similarity, and perform various types of DoS attack in different scenarios. The attacker has also some inabilities, such as changing the MAC from the devices he controls and verify if the devices have been cut off from the network.

Hence, to prevent these attacks we created antiDoSte, which is a prototype solution created for detecting and mitigating DoS/DDoS attacks on the network supporting the CROSS application. antiDoSte is a hybrid solution that combines SDN and ML technologies. Our solution has three detection techniques to detect these attacks. The first are *Thresholds* that limit the number of specific packets a host can sent in a row. The second, the *DL model*, that was trained to recognize specific attacks. The third, the *LR model*, was trained to recognize the normal behavior of the network. The LR model also serves to improve the Thresholds and DL approaches, i.e., if both the Thresholds and DL approaches do not detect the attack and the LR does, a log/dataset is created from the packets generated from that attack, which will be used

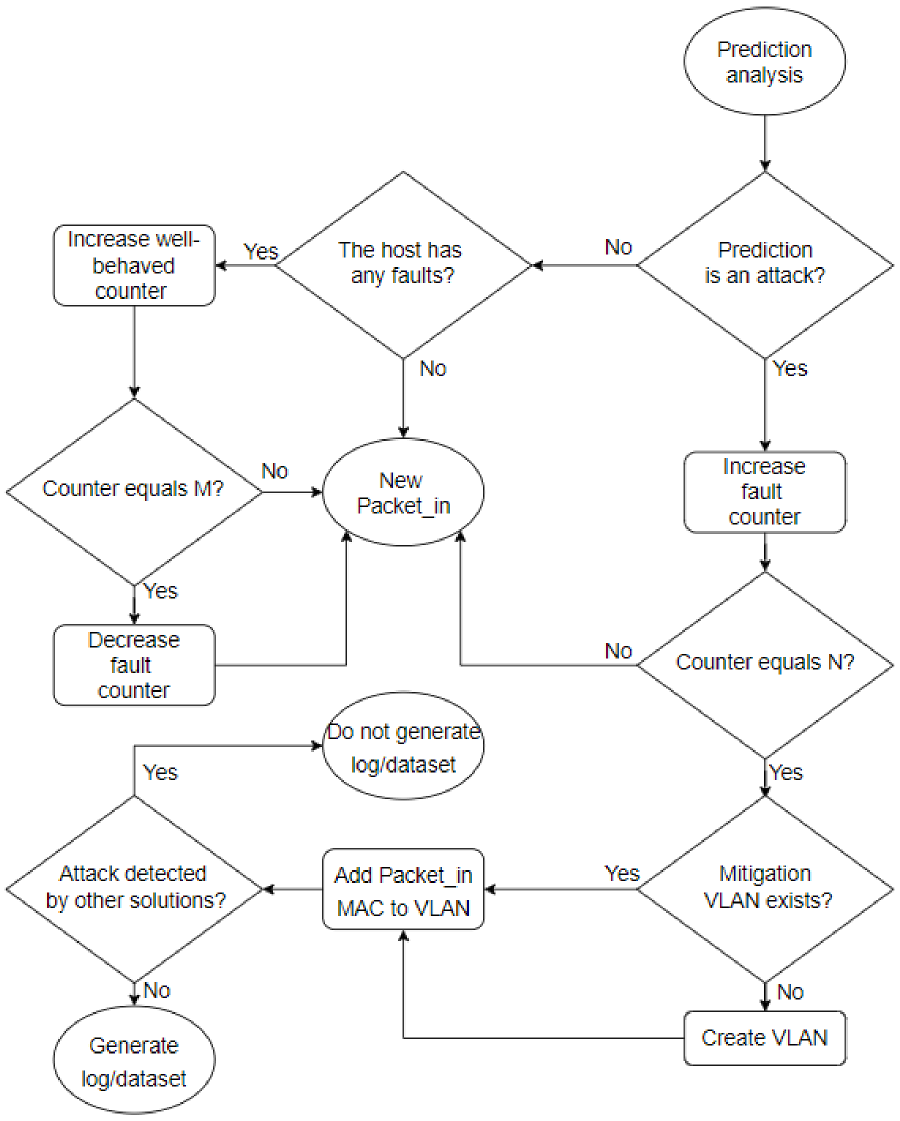


Figure 3.4: Mitigation strategy for LR approach.

later to improve the solution.

If a host is stated as malicious by the Threshold or the DL approaches, we create a VLAN in the switch flow table, if one does not exist. The VLAN is created in the switch present in the same subnetwork as the malicious host, and "drop" rules are applied to all the traffic coming from the VLAN. After that, the malicious host is added to that VLAN. For the LR approach, the host must have unexpected behavior N number of times, according to the LR model, before being mitigated by the same VLAN strategy, where N is a pre-defined integer.

Figure 3.5 represents how the three approaches work together to detect and mitigate the attack. As we can see, the strategies analyze the traffic at the same time, so the attack gets mitigated as soon as possible.

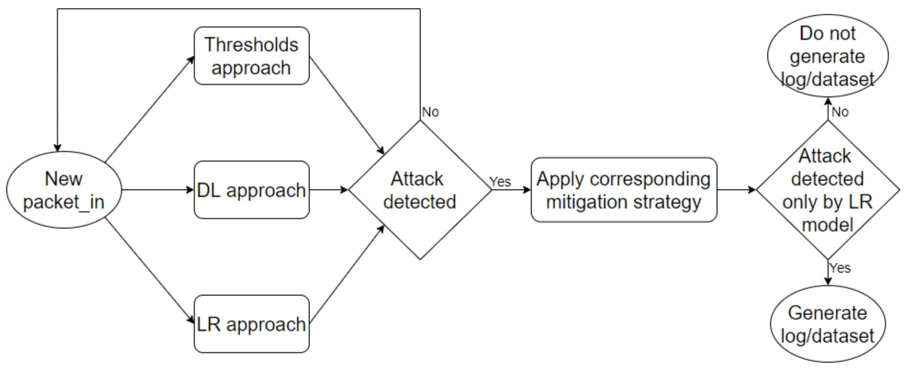


Figure 3.5: ANTIDoSTE flowchart.

# Chapter 4

# Test-bed

With the goal of testing our solution to protect CROSS, we created a test-bed based on the network supporting the CROSS application. The CROSS network has several smart devices that interact with the application server, such as Kiosks, Smart Space managers and Wi-Fi Access points [MCP20]. These devices can be hijacked by an attacker, making it possible to perform some attacks, in our case DoS/DDoS attacks. Therefore, in this chapter we explain our test-bed, starting by first describing the tool we use to emulate CROSS then we mention the hardware components and software components we emulate in our test-bed, ending with a brief overview of the network topology, the capabilities of the test-bed and some possible attack scenarios.

## 4.1 Network Emulator

To emulate the CROSS network we use Mininet [LHM10], which is an OpenFlow-based SDN emulator, giving researchers an efficient way to test their SDN frameworks and measure their performance and reliability. The Mininet is an open source emulator written in the Python programming language. It is built over the Ubuntu Linux distribution. The elements of Mininet are organized into three main components: the *host*, which sends and receives the packets, the *switch*, which stores all the required rules to forward the packets to their destinations, and a *central controller* which handles the functionality of control and management operations in the network. Mininet supports different types of virtualized hosts, switches and controllers.

## 4.2 Software and Hardware

On the top of the hardware infrastructure, we have a POX controller [KSG14] which is Python-based open-source OpenFlow/SDN. POX is used for faster development and prototyping of new network applications. The controller comes pre-installed with the Mininet virtual machine. By using them you can turn OpenFlow devices into hub, switch, load balancer, firewall devices. The POX controller allows easy way to run OpenFlow/SDN experiments. POX can be passed different parameters according to real or experimental topologies, thus allowing experiments to be run on real hardware, test-beds or in the Mininet emulator. To generate the traffic from each host we use a Python library, called Scapy. It is a powerful interactive packet manipulation program that can forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies.

This test-bed consists mainly of a server, 10 devices that can be Kiosks, Smart Space Managers (SSM) or Wi-Fi Access Point (AP), 6 OpenFlow-enabled switches and a SDN controllers. The emulation was done in a machine with a Intel Core i7-8750H CPU at 2.20GHz and 16GB of RAM.

## 4.3 Network topology

The CROSS network is represented in Figure 4.1. As we can see, there are 6 sub-networks, across the city of Lisbon, in the emulated framework, 5 of them are tourism points, Comércio, Jerónimos, Oceanário, Sé and Gulbenkian, with 2 devices, per sub-network, for proof location, and the last one, Alvalade, is for the CROSS server. Note that we only use 2 tourism devices, per sub-network, for our evaluation, but more can be added. Each sub-network has a switch which is connected to every device in the sub-network, working as a router for those devices. In the figure there is a Pox controller in each subnetwork, which is responsible for setting rules in the switch for forwarding or dropping packets, however these controllers represent the same controller, i.e., there is only one Pox controller. We decided to make the figure with many controllers so that the connections between devices is simplified. Also, take into account that, in Figure 4.1 it is internet with lower-case “i” since CROSS is a private network.

For the traffic we started by creating a simple HTTP server that responds to every request with a HTTP OK message. For the tourism-based devices, the Kiosk and the Smart Space manager have similar traffic, since both their communication, with the server, starts with a TCP 3-way handshake, followed by an HTTP GET message with a string containing the device ID and a request for synchronization, ending with the TCP connection finalization. The Access point also starts with the 3-way handshake and ends with the TCP connection finalization, but the difference here is in the HTTP GET message where the string contains the ID and the local time. This traffic pattern, for each device, happens from a x-to-x time, where x is a random generated number from 1 to 5 seconds.

The communication between the controller and model is file based, since both the controller

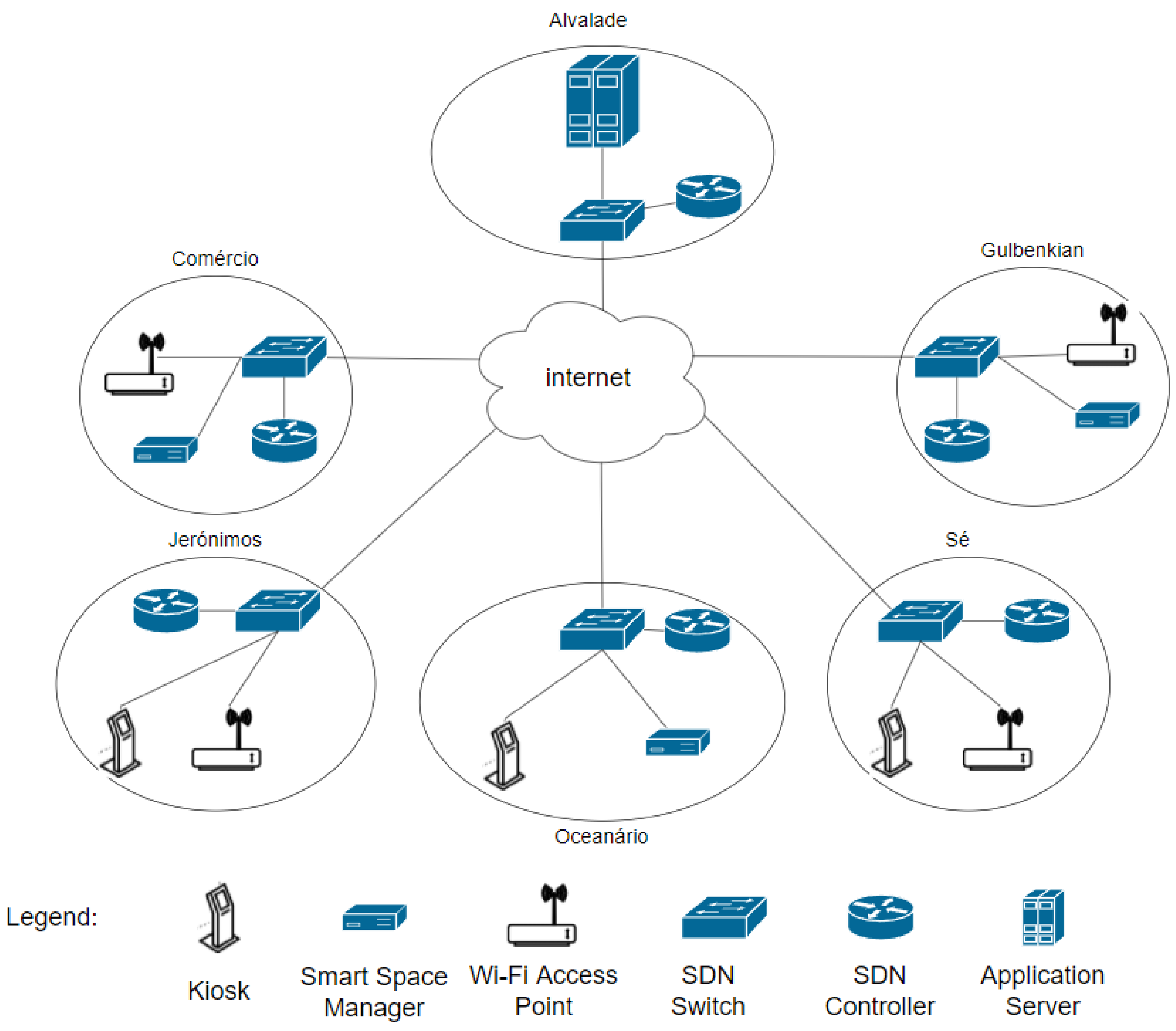


Figure 4.1: Emulated CROSS topology.

and the models will be deployed in the same device but on different applications. The encryption of the file is done with *Fernet*, which is a python library, using a symmetric key that is generated in the controller once it boots up. This is done to prevent an attacker from reading the files. The communication starts with the file produced by the controller, where the necessary features will be recorded (these features will be exposed in the next chapter). Once the application, where the model is deployed, receives the file created by the controller, the model analyses the packet features and produces a new file with the prediction about the maliciousness of those packets. This file is also encrypted and send to the controller with the same symmetric key. The controller reads the file and depending in the model's prediction, takes the required measurements, acting on stopping or not the traffic from that device.

## 4.4 Discussion

With this test-bed we are able to test our solution, antiDoSte, in many different DoS/DDoS scenarios. These scenarios range from the most simplistic where a single tourism-based device is used to perform a DoS attack, to most sophisticated ones where all the tourism-based devices from one subnetwork are used to perform a DDoS attack targeting the application server. Some examples are: a tourism-based device in the subnetwork Jerónimos is used to perform an attack targeting the application server in Alvalade or another device; one tourism-based device from each tourism point is used to perform a DDoS attack targeting the application server. Also, thanks to the presence of a controller in our test-bed our solution will have the necessary features it needs to detect an attack, since all the packet will pass through this controller, which is responsible for the features extraction of the packets.

Although we created this test-bed with the intention of testing our prototype solution, it can also be used for testing other solutions, for DoS/DDoS detection, that use SDN. According to SWAMI et al. [SDR19] there are solutions that use SDN to detect and mitigate DoS/DDoS and solutions that protect the SDN architecture from DoS/DDoS attacks. Our solution uses SDN for DoS detection and mitigation, therefore this test-bed is suited for this type of solution. For example, in the work of Yin et al. [YZY18], since the solution is deployed in the controller, it would only be needed to update the controller with the necessary functions and perform the DoS/DDoS, with the tourism-based devices, aiming the CROSS server. Another example are machine learning based mechanisms [SDR19] [RS20], where if used in this test-bed the model can be deployed in controller or alongside it in order to detect the attacks, as we do in our solution. But it is also possible to test the other type of solutions that protect the data plane. For example, the mitigation strategy, that we use, proposed by Ravi et al. could also be tested in a scenario where, the switch has N entries in the flow table and the attacker has a botnet, in the same subnetwork of the switch, with more than N devices. And so, it would be possible for the test-bed to create more tourism-based devices than the number of flow rules a switch can have to verify if this mitigation strategy would work.

# Chapter 5

# Evaluation

To evaluate our solution, we used the test-bed. The CROSS server was the target of the DoS/DDoS attacks, and it is located in the Alvalade sub-network, as represented in Figure 4.1. In table 5.1 we can see the types of DoS tested in our experiments and if they are known by the Thresholds or the DL approaches. The remainder of this Chapter will disclose the approaches we followed for the evaluation of each detection strategy and for the VLANs mitigation strategy, as well as the reason why we chose those approaches.

## 5.1 Approach

For the Thresholds detection strategy, we evaluate the time it takes to detect a DoS attack. This metric is important to evaluate how well the detection strategy behaves when an attack is happening. For the DL and LR approaches we also evaluate the time it takes to detect DoS attack, for the same reason of the Thresholds approach, and we evaluate metrics such as:

- *Accuracy*, to see how well the solution predicts the status of the network (benign or under attack) correctly;

$$Accuracy(\%) = \frac{TP + TN}{TP + TN + FP + FN} * 100 \qquad (5.1)$$

  True positive (TP); False positive (FP); True negative (TN); False negative (FN);

- *True Positive rate/Recall*, to check how well the solution can see that an attack is not

Table 5.1: Types of DoS attacks tested.

| Attack | Known | Unknown |
|---|---|---|
| TCP-SYN flood | X | |
| ICMP flood | X | |
| UDP flood | | X |

happening;

$$True\ Positive\ rate(\%) = \frac{TP}{TP + FN} * 100 \tag{5.2}$$

- *False Positive rate*, to verify how well the solution can detect when an attack is occurring;

$$False\ Positive\ rate(\%) = \frac{FP}{TN + FP} * 100 \tag{5.3}$$

- *Precision*, to examine at what precision (i.e., DoS is not marked as benign) the model predicts benign activity in the network.;

$$Precision(\%) = \frac{TP}{TP + FP} * 100 \tag{5.4}$$

- *F-score*, to help us find out the optimum threshold between Recall and Precision in the model, and so that the model can be compared with other classifiers in future work.

$$F{-}score(\%) = \frac{2 * Precision * Recall}{Precision + Recall} * 100 \tag{5.5}$$

These metrics were evaluated for the DL and ML approaches, since these approaches are based on predictions and not on rules, as with the Thresholds.

The deployment of the models was done in an application running alongside the controller. The controller is responsible for extracting the *features* from the received packets and then sending these features to the application where the model is running. The extracted features can be seen in Table 3.2, alongside some examples and their types. Both models analyze a window of traffic (multiple packets at the same time) instead of analyzing a single packet, making values such as the time between packets more meaningful, which is an important feature to detect the flooding attacks. For the DL model, these metrics were evaluated, with a dataset generated from normal traffic of the test-bed and a dataset, BUET-DDoS2020 [MH20], for a specific type of DoS attack, namely ICMP flood attack, both with 10 000 entries. For the LR model we used a dataset generated from normal traffic of the test-bed, 50 000 entries, and approximately 10 000 entries for each attack present in the test dataset from BUET-DDoS2020, namely, TCP-SYN flood, UDP flood, HTTP flood, DNS flood and ICMP flood.

## 5.2 Results

For measuring the time each approach takes to detect and mitigate the attack and the bandwidth impact from the normal, attack and mitigated traffic we evaluate each approach in three different

Table 5.2: Deep Learning and Machine Learning models evaluation metrics.

| Model | Accuracy | True Positive Rate | False Positive Rate | Precision | F1-score |
|---|---|---|---|---|---|
| DL | 99.97% | 99.95% | 0% | 100% | 99.97% |
| LR | 74.97% | 59.90% | 10.73% | 38,90% | 47.17% |

scenarios, while normal traffic is circulating:

- In the first scenario we simply do a DoS attack from a device present in a subnetwork, Jerónimos in this example, targeting the CROSS server present in the Alvalade subnetwork.
- The second scenario consists of performing a DDoS with one device from each subnetwork, except Alvalade, aiming the CROSS server.
- In the third scenario we also perform a DDoS targeting the CROSS server but with five devices from one the Jerónimos subnetwork.

We captured the traffic in the Alvalade sub-network of CROSS between the server and the SDN switch. Also, to have a point of reference of the impact that these attacks bring to the network, Figure 5.1 shows what are the normal values for the bandwidth, from the test-bed, where the axis X is in seconds and the axis Y in Kbits.

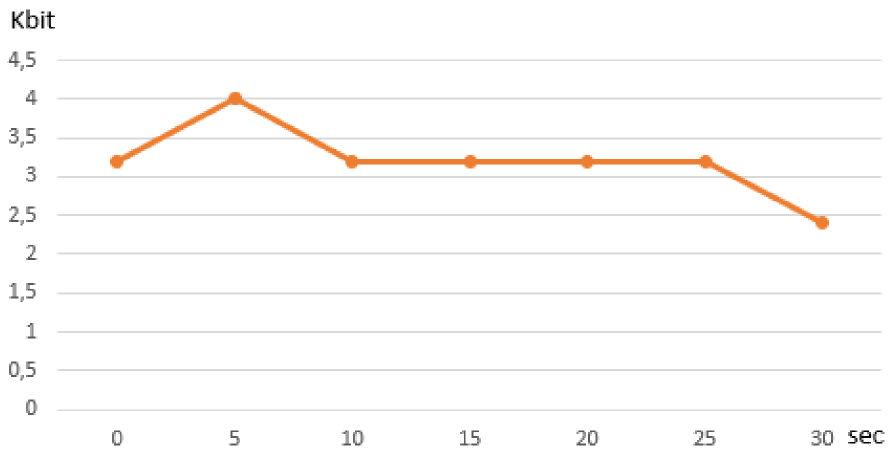
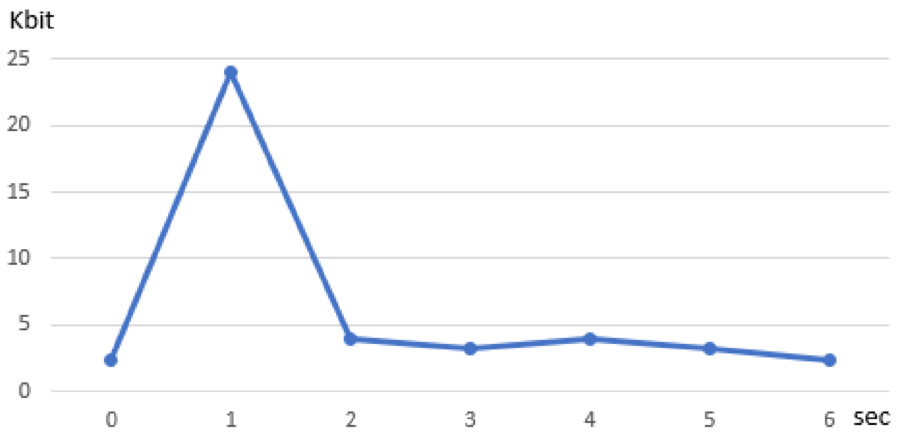


Figure 5.1: Test-bed normal bandwidth.

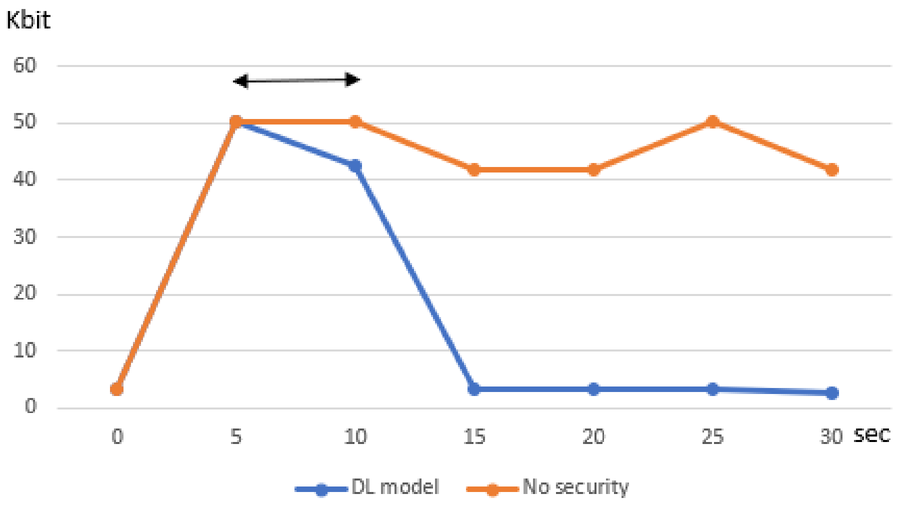
### 5.2.1 DoS scenario

For the first scenario, where there is a single device from Jerónimos performing a DoS attack targeting the CROSS server, we started by performing a TCP-SYN flood attack that will be detected by the Thresholds approach. The time of detection depends on the packet rate of the attack and the predefined Threshold. This is because the detection happens when the attacker sends pre-defined number of packets, in our experiments it is the number of TCP-SYN packets. And so, if the packet rate is low, the detection will take more time, also, the higher the threshold the more time it takes to mitigate. But of course, if the packet rate is low the impact on the victim server will be less than if the packet rate was higher, and so the attacker has little interest in low packet rates, for flooding attacks. In addition, we cannot set the threshold too low because this might cause that legitimate user will not be able to communicate with the server when the network is saturated with other requests from legitimate clients.

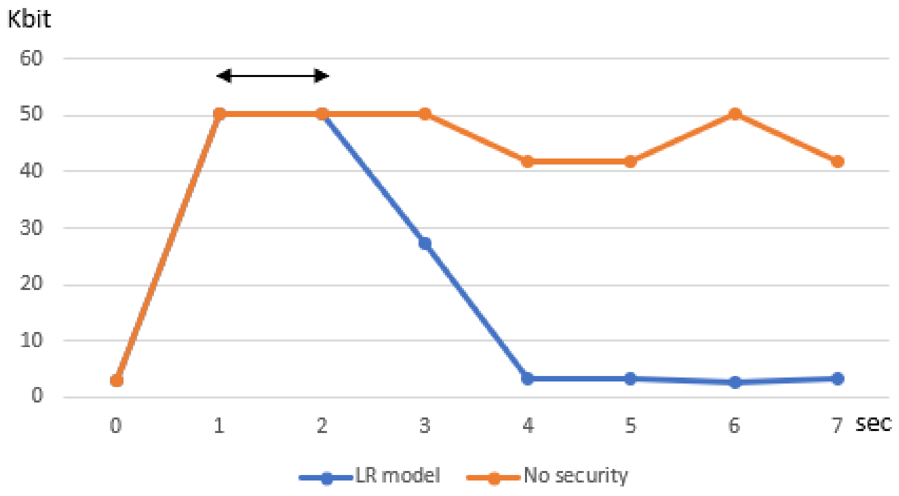
As mentioned before, the attack we use for this experiment is the TCP-SYN flood attack and it was performed with a Python library named Scapy.Therefore, for this attack scenario, represented in Figure 5.2a, we start the TCP-SYN flood attack at 5 seconds, which is represented in the graph by the spike. However, since we do the attack with high packet rate, and the Threshold is 20, it is mitigated almost instantly (after the first 20 TCP-SYN packets have passed through the controller, the mitigation strategy is applied), and so the attack does not have an impact such as what will occur in the other approaches. Note that we set the Threshold to 20 as an example of what a real-world scenario could be. There was no investigation to determine what is the optimal Threshold since the emulation of the network is done in the same machine, without packet loss. In this case, the optimal Threshold for this network was 2, which does not make sense in a real network.



(a) TCP-SYN flood with Thresholds detection.



(b) ICMP flood with DL model detection.



(c) UDP flood with LR model behavior analysis.

Figure 5.2: DoS attack aiming the server.

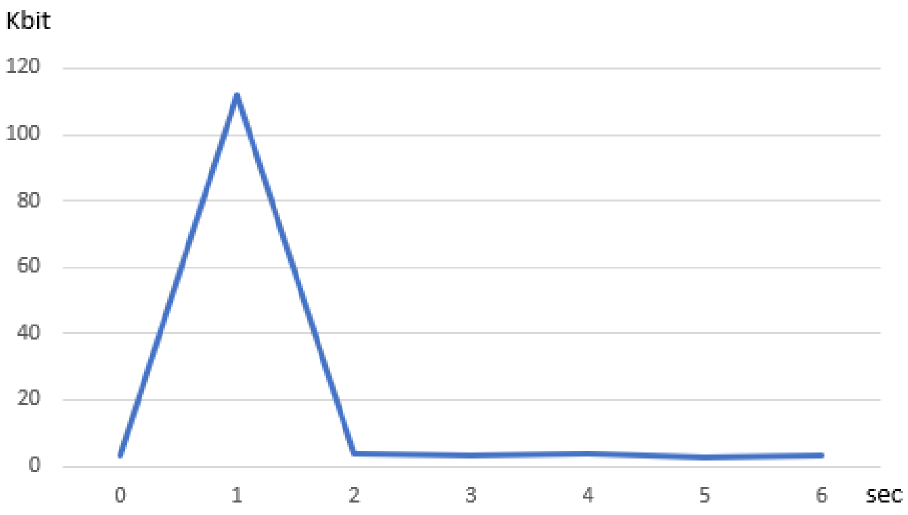
The second attack we tested in this scenario was the ICMP flood attack that will be detected by the DL model. This attack was performed with Scapy. The effect on our solution in the single DoS devices attack scenario can be seen in Figure 5.2b. We started the attack around the 5 second mark, and it is mitigated at 10 seconds (black arrow), making our solution able to detect and mitigate the attack in approximately 5 seconds. However, since there are still malicious packets in the buffer the traffic only returns to its normal state at 15 seconds.

The third attack and final attack for this scenario was a UDP flood, which our solution is not prepared to recognize. Therefore, the LR model will be responsible for the attack detection since the UDP flood will represent an abnormality in the network traffic. This attack was done with the hping3 tool [Lim21]. As stated in Chapter 3.3, we need to predefine a value, for the mitigation strategy, that is the number of times an abnormality needs to be detected so the traffic is mitigated, and so we predefined that value to be 3. As you can see from Figure 5.2c the attack started at 1 seconds and is mitigated at 2 seconds, taking about 1 second for the LR model to detect and mitigate the attack (black arrow), being faster than the DL model despite having to pass through 3 verifications. Returning the traffic to its normal state at 4 seconds, due to malicious packets still being in the switch buffer.

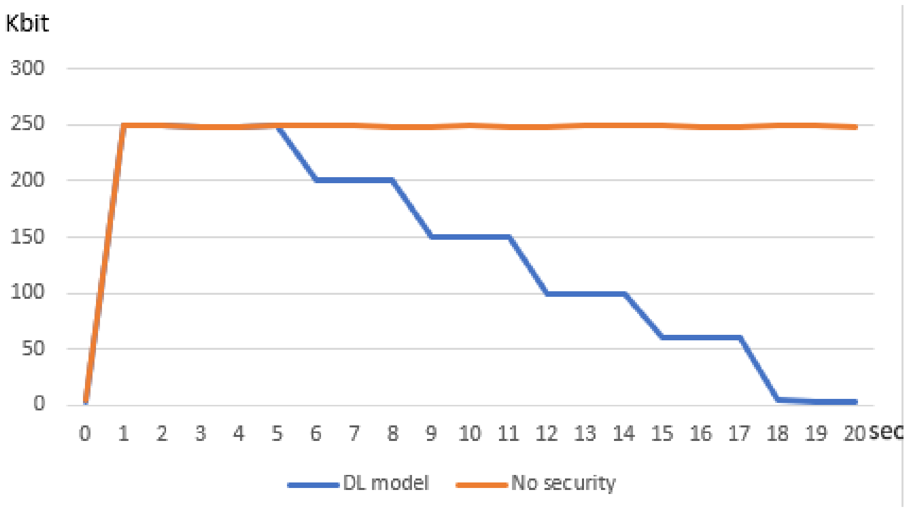
### 5.2.2 DDoS scenarios

For both DDoS scenarios we also did the previously mention attacks. Therefore, For the Thresholds approach, represented in Figures 5.3a, for the DDoS attack from different subnetworks scenario, and 5.4a, for the DDoS attack from the same subnetwork scenario, we also start the attack at 5 seconds and there is a similar result in terms of detection and mitigation velocity, however the spike reaches higher values since this time there are five devices executing a TCP-SYN flood attack at the same time.

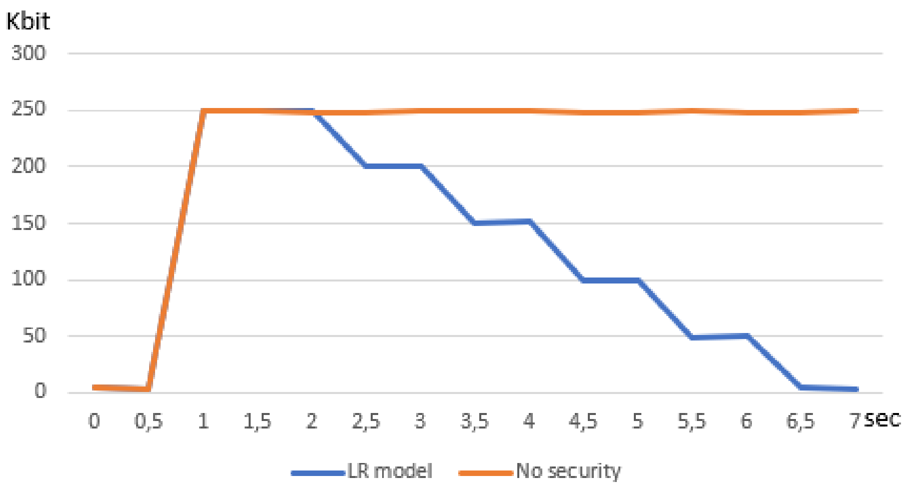
For the DL model approach, as you can see from Figures 5.3b, for the DDoS attack from different subnetworks scenario, and 5.4b, for the DDoS attack from the same subnetwork scenario, we started the DDoS at 1 second and around the 5-6 second mark. There is a decrease from the bandwidth consumed by the attack (about 50 Kbits), what this means is that one of the devices, from the DDoS attack, has been stated as malicious by the DL model and so the mitigation strategy has been applied. However, only one device, out of five, was mitigated, i.e., only one device will be mitigated at a time. This has to do with our detection approach, since we analyze the traffic one host at the time, instead of analyzing all network traffic at the same time. Therefore, the detection approach analyses the traffic from one host and decides if it is malicious, then analyzes a second host and decides if it is malicious, and so on. Consequently, all the malicious device traffic is mitigated at 17-18 seconds.



(a) TCP-SYN flood with Thresholds detection.

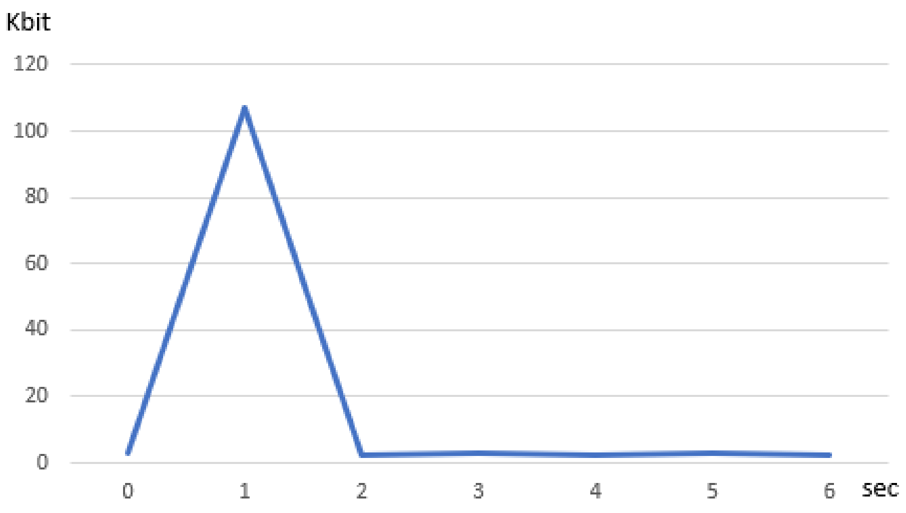


(b) ICMP flood with DL model detection.

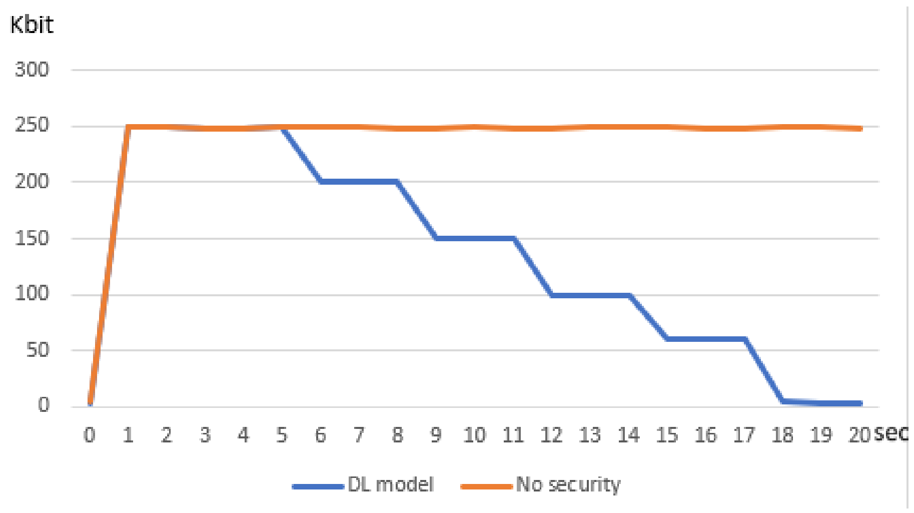


(c) UDP flood with LR model behavior analysis.

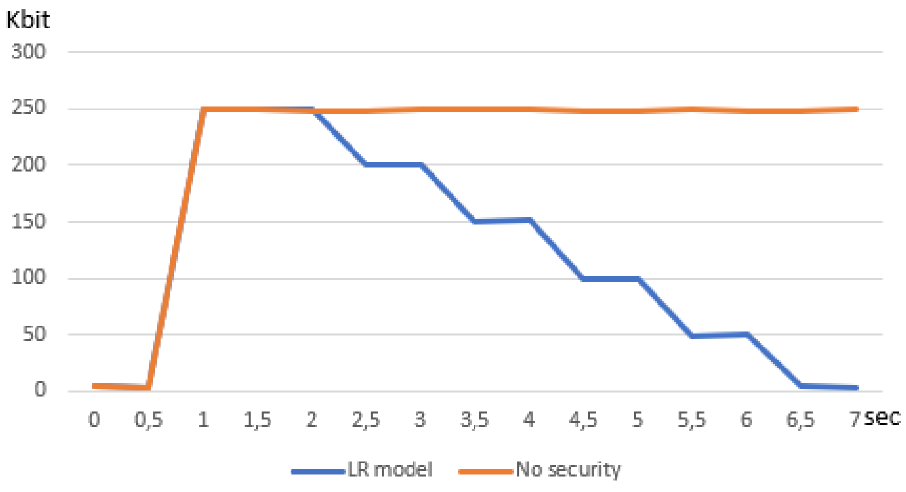
Figure 5.3: DDoS attack from different subnetworks aiming the server.



(a) TCP-SYN flood with Thresholds detection.



(b) DDoS attack from different subnetworks.



(c) DDoS attack from the same subnetwork.

Figure 5.4: DDoS attack from the same subnetwork aiming the server.

Finally, for the LR model approach, represented in Figures 5.3c, for the DDoS attack from different subnetworks scenario, and 5.4c, for the DDoS attack from the same subnetwork scenario, we start the attack at the 1 second mark and again the attacks from each device are detected and mitigated one at the time. Being the attack fully mitigated at around the 6-6.5 second mark.

### 5.2.3 Mitigation

In order to demonstrate the VLAN mitigation strategy we started by setting the limit of flow rules the switch can have to 3. Therefore, we ran the DDoS scenario where there are 5 devices in the same subnetwork performing an attack targeting the application server two times with the VLAN mitigation strategy applied and with a mitigation strategy where there is an entry per malicious host. Figure 5.5 represents the results when performing the TCP-SYN flood attack. We start the attack at 1 second, and as you can see, for the VLAN mitigation strategy, we have the same results when we tested this attack scenario previously. However, when it comes to the strategy where there is an entry per malicious host, as it is possible to see, the attack is not fully mitigated. What is happening is that only 3 attacks are being mitigated since there it is only possible to set 3 flow rules in the switch due to is previously set space. Therefore, the attack gets detected for each malicious host, but it is only putted 3 drop rules in the switch flow rules and the other 2 devices, that did not get drop rules, continue to perform the attack.

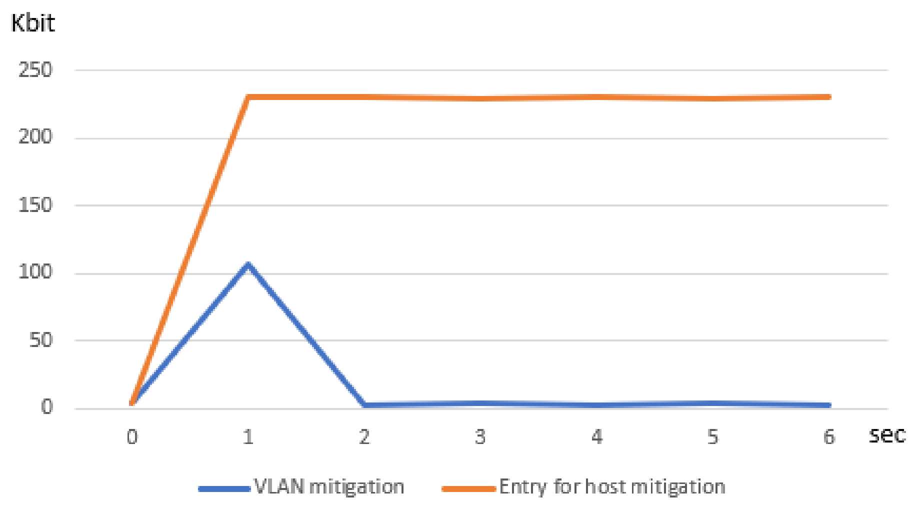
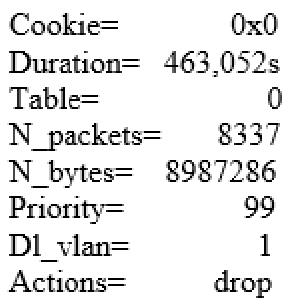


Figure 5.5: Results of having the VLAN mitigation strategy on a low capacity switch.

After the attack we analyzed the switch from Jerónimos subnetwork and verified that there was only one flow rule, represented in Figure 5.6. As it is possible to visualize, this flow rule corresponds to an entry for a VLAN, VLAN 1 in this case, and that there is an action to drop every packet from that VLAN.



```
Cookie=          0x0
Duration=   463,052s
Table=             0
N_packets=      8337
N_bytes=     8987286
Priority=         99
Dl_vlan=           1
Actions=        drop
```

Figure 5.6: Jerónimos's switch flow table after DDoS.

## 5.3 Discussion

By analyzing the results, we can conclude that the detection strategy based on SDN rules (Thresholds) is the fastest at detecting DoS attacks, but to use this approach we need to have considerable knowledge of the attack behavior so we can know which features of the attack to use to stop it. The DL approach is the slowest but is the easiest to implement since we only need data from the attack to train the model, not requiring the knowledge that the Thresholds strategy needs about the attack to detect it.

Also, in Table 5.2 we can see the performance results of the DL and LR models. As expected, the DL model is much better than the LR model, not only because of the DL models have better accuracy than LR models, but also because the DL model is specialized to detect specific attack, since we provide a large quantity of ICMP flood attack entries for its training. This allows the DL model to have a better knowledge of what an attack looks like. In contrast, the LR model was trained to detect abnormal behavior, since we train this model with a large quantity of normal traffic entries, and with small quantities of each attack datasets. Therefore, requiring much more data and knowledge of the network where the model is deployed. Being, this is the reason why we apply a different mitigation strategy for the LR approach. In addition, these results also demonstrate why we cannot deploy the behavior analysis strategy by itself, since most of the attacks could pass without being detected, and so, this strategy serves more of a support to the other two, by providing the data gathered from the attack to improve them. However, it is a great addition to the other two strategies since it can help with the development of the prototype solution, by improving the other two solutions with logs/datasets created from abnormal traffic.

Another aspect that is noticeable in the graphs from the DDoS scenarios are their similarity, i.e., the same result happens in the scenario where the DDoS is performed with one device from each subnetwork and in the scenario where the DDoS is perform with five devices from one subnetwork, this as to do with the presence of a single controller to manage the entire network.

If we had a controller per subnetwork what would happen, in the scenario where a DDoS is executed with a device from each subnetwork, is that graph would have the appearance on the DoS scenario, since each controller, in conjunction with the application running the ML model, would be able to detect and mitigate the attack at, approximately, the same time. Making this a disadvantage of using a centralized framework. Moreover, this detection and mitigation solution is not the most appropriate for a DDoS scenario where the attacker has a large number of malicious devices, since the higher the number of malicious devices the higher it takes to mitigate all the attacks.

When comparing our solution with previous work we can see that it has strong and weak points. First let us compare the Thresholds approach with the work of Yin et al. [YZY18], since in their solution it is also used thresholds. To test their solution, Yin et al. had 50 devices scattered around multiple subnetworks. Device number 50 was responsible for performing a DoS aiming all the other 49 devices, and device number 15 was responsible for simulating normal traffic by communicating with the other 49 devices. Both devices were connected to the same switch and the bandwidth was captured between that switch and the rest of the network. The attack took 5 seconds to be detected with Yin et al. algorithm, while ours took less than a second to detect and mitigate and does not assume that the packets generated from the attack are similar, which is an assumption done by Yin et al. However, we need to consider that in their attack scenario the DoS targets 49 devices while in our scenario 5 devices attack the CROSS server. Thus, the values might change if we used a similar attack scenario.

For the ML approaches, when comparing our solution with state-of-the-art models [RS20] [MBM+18] [MMA+18] we verify that there is still improvement to be done. The DL model takes roughly 2 seconds to detect an attack which is quite a while when comparing with Meidan et al. [MBM+18] model that takes 212 milliseconds. The LR model despite being faster does not reach such values, taking approximately half a second. Also, we can compare the previously mention quantitative metrics from the DL model with the state-of-the-art models, since both models were tested to recognize attacks. As it is possible to see from Table 5.2 and 5.3 the DL model has values similar to the models from previous works, indicating that our solution can detect attacks as well as these state-of-the-art solutions. Note that the work of Meidan et al. is not in Table 5.3 since they do not present all these quantitative metrics.

Table 5.3: Related work models evaluation metrics.

| Model | Accuracy | TPR | FPR | Precision | F1-score |
|---|---|---|---|---|---|
| SDELM [RS20] | 96.28% | 97.20% | 4.85% | 95.35% | 96.20% |
| Naive Bayes [MMA+18] | 87.31% | 91.69% | 9.73% | 91.63% | 86.33% |

# Chapter 6

# Conclusion

We developed ANTIDOSTE to protect CROSS, which is a location proof system, supporting a smart tourism application, from Denial-of-Service (DoS) attacks. ANTIDOSTE uses Software Defined Networking (SDN) techniques in conjunction with Machine Learning (ML) techniques to detect DoS attacks, making it a hybrid solution that combines benefits of different technologies. By using more than one approach to detect attacks it is possible to cover the weaknesses of some detection strategies. ANTIDOSTE has a semi-automatic improvement thanks to the Logistic Regression (LR) model detection approach that is responsible for generating logs/datasets from the detected unexpected behavior of the attacks. These logs/datasets are analyzed by a network manager to verify if they represent an attack, and if so, they can be used to improve the detection, with a manual rule for the *Thresholds* approach or with automatic training using DL.

We also propose a test-bed for DoS attack detection and mitigation based on CROSS. Our test-bed uses SDN so we can do a deep inspection of the packets from each device in the network, as well as, being able to set flow rules.

For the future our priority is to investigate a solution to detect and mitigate DDoS attacks in a faster manner for the ML approaches. This improvement may consist in the following approach: instead of analyzing windows of traffic from each device we could analyze for all the devices at the same time, although we need to consider that, in case of an attack, there is the possibility that a packet from a benign host might be in that window of traffic therefore we cannot simply say that every host in that "malicious" window of traffic needs to have its traffic dropped. Another possibility is to sacrifice some accuracy by analyzing packet by packet instead of window by window therefore if a packet is determined as malicious, we would not need to worry with the mitigating benign host since the packet would be from a malicious host (if the prediction from the ML models is correct). In this scenario, what would happen, we would train the ML models to recognize the packets patterns, for example, in the case of the DL model it would need to know what the packet for a specific attack looks like, in contrast to what we have now that is what a window of attack looks like. However, we, most likely, would need datasets with more packet features for the training that the ones we used, since the models need more specification considering that they are only analyzing a packet, which may not be easy to get this type of dataset.

We demonstrated the capabilities of our prototype solution for a specific use case. We plan to test our solution in different systems to prove it can become a general solution, as we designed it to be. Moreover, we want to improve the LR mitigation strategy. The original idea was to limit the traffic bandwidth to a predefined value that would represents the maximum bandwidth a host should have during a normal behavior and prioritize the traffic from other hosts in the same sub-network. However, this was not possible due to the limitation of the OpenFlow protocol version that POX controller supports, 1.0, in which is not possible to set, dynamically, a limit to the bandwidth in the SDN switches. In contrast, version 1.2 of OpenFlow protocol allows for maximum rate setting, and this will be used in future versions. Also, we aim to improve our solution to detect more and different types of DoS/DDoS attacks and improve the used ML models so attacks can be detected better and faster.

# Bibliography

[AA19] Alia Mohammed Alrehan and Fahd Abdulsalam Alhaidari. Machine learning techniques to detect ddos attacks on vanet system: a survey. In *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, pages 1–6. IEEE, 2019.

[AANHA16] Mouhammd Alkasassbeh, Ghazi Al-Naymat, Ahmad Hassanat, and Mohammad Almseidin. Detecting distributed denial of service attacks using data mining techniques. *International Journal of Advanced Computer Science and Applications*, 7(1):436–445, 2016.

[ABB$^{+}$05] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuéllar, P Hankes Drielsma, Pierre-Cyrille Héam, Olga Kouchnarenko, Jacopo Mantovani, et al. The avispa tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification*, pages 281–285. Springer, 2005.

[ABC$^{+}$16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

[ALRL04] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan 2004.

[Ang17] Kishore Angrishi. Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets. *arXiv preprint arXiv:1702.03681*, 2017.

[ARCG20] Dilara Acarali, Muttukrishnan Rajarajan, Doron Chema, and Mark Ginzburg. Modelling dos attacks & interoperability in the smart grid. In *2020 29th Interna-*

*tional Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE, 2020.

[ATN17] M Anirudh, S Arul Thileeban, and Daniel Jeswin Nallathambi. Use of honeypots for mitigating dos attacks targeted on iot networks. In *2017 International conference on computer, communication and signal processing (ICCCSP)*, pages 1–4. IEEE, 2017.

[BAN89] Michael Burrows, Martin Abadi, and Roger Michael Needham. A logic of authentication. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 426(1871):233–271, 1989.

[CDDW19] Wei Chen, Derui Ding, Hongli Dong, and Guoliang Wei. Distributed resilient filtering for power systems subject to denial-of-service attacks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(8):1688–1697, 2019.

[CDNN$^+$18] Tiago Carneiro, Raul Victor Medeiros Da Nóbrega, Thiago Nepomuceno, Gui-Bin Bian, Victor Hugo C De Albuquerque, and Pedro Pedrosa Reboucas Filho. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, 6:61677–61685, 2018.

[CWD$^+$17] Sravani Challa, Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Alavalapati Goutham Reddy, Eun-Jun Yoon, and Kee-Young Yoo. Secure signature-based authenticated key establishment scheme for future iot applications. *IEEE Access*, 5:3028–3043, 2017.

[DAF18] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine learning ddos detection for consumer internet of things devices. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 29–35. IEEE, 2018.

[DKJ16] Ali Dorri, Salil S Kanhere, and Raja Jurdak. Blockchain in internet of things: challenges and solutions. *arXiv preprint arXiv:1608.05187*, 2016.

[DKJG17] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Blockchain for iot security and privacy: The case study of a smart home. In *2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops)*, pages 618–623. IEEE, 2017.

[Dou02] John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.

[EC18] R Groves E Chou. *Distributed Denial of Service (DDoS)*. O'Reilly, 2018.

[GAHC19] Chrispin Gray, Robert Ayre, Kerry Hinton, and Leith Campbell. 'smart'is not free: Energy consumption of consumer home automation systems. *IEEE Transactions on Consumer Electronics*, 66(1):87–95, 2019.

[Goo18] Google. Colaboratory: Frequently asked questions. 2018.

[GSXK15] Ulrike Gretzel, Marianna Sigala, Zheng Xiang, and Chulmo Koo. Smart tourism: foundations and developments. *Electronic markets*, 25(3):179–188, 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[IS15] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[JNZ16] Mohammad Reza Jabbarpour, Armin Nabaei, and Houman Zarrabi. Intelligent guardrails: an iot application for vehicle traffic congestion reduction in smart city. In *2016 ieee international conference on internet of things (ithings) and ieee green computing and communications (greencom) and ieee cyber, physical and social computing (cpscom) and ieee smart data (smartdata)*, pages 7–13. IEEE, 2016.

[KA18] Dimitrios Karagiannis and Antonios Argyriou. Jamming attack detection in a pair of rf communicating vehicles using unsupervised machine learning. *Vehicular Communications*, 13:56–63, 2018.

[KDG+02] David G Kleinbaum, K Dietz, M Gail, Mitchel Klein, and Mitchell Klein. *Logistic regression*. Springer, 2002.

[Ker] Keras. `https://keras.io/about/`.

[KMNA+18] Raenu Kolandaisamy, Rafidah Md Noor, Ismail Ahmedy, Iftikhar Ahmad, Muhammad Reza Z'aba, Muhammad Imran, and Mohammed Alnuem. A multivariant stream analysis approach to detect and mitigate ddos attacks in vehicular ad hoc networks. *Wireless Communications and Mobile Computing*, 2018, 2018.

[KSG14] Sukhveer Kaur, Japinder Singh, and Navtej Singh Ghumman. Network programmability using pox controller. In *ICCCS International Conference on Communication, Computing & Systems, IEEE*, volume 138, page 70. sn, 2014.

[LHM10] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, pages 1–6, 2010.

[Lim21] OffSec Services Limited. hping3, 2021.

[LLPBOD20] Carmelo Cascone Larry L. Peterson, Thomas Vachuska Brian O'Connor, and Bruce Davie. *Software-Defined Networks: A Systems Approach.* Systems Approach LLC, 2020.

[MBM$^+$18] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[MCP20] Gabriel A Maia, Rui L Claro, and Miguel L Pardal. Cross city: Wi-fi location proofs for smart tourism. In *International Conference on Ad-Hoc Networks and Wireless*, pages 241–253. Springer, 2020.

[MDML17] Vincenzo Matta, Mario Di Mauro, and Maurizio Longo. Ddos attacks with randomized traffic innovation: Botnet identification challenges and strategies. *IEEE Transactions on Information Forensics and Security*, 12(8):1844–1859, 2017.

[MDMN12] Zigurd R Mednieks, Laird Dornin, G Blake Meike, and Masumi Nakamura. *Programming android.* " O'Reilly Media, Inc.", 2012.

[MH20] S. Islam M. Hasan. Buet-ddos2020, 2020.

[MLZZ15] Huadong Ma, Liang Liu, Anfu Zhou, and Dong Zhao. On networking of internet of things: Explorations and challenges. *IEEE Internet of Things Journal*, 3(4):441–452, 2015.

[MMA$^+$18] Amjad Mehmood, Mithun Mukherjee, Syed Hassan Ahmed, Houbing Song, and Khalid Mahmood Malik. Nbc-maids: Naïve bayesian classification technique in multi-agent system-enriched ids for securing iot against ddos attacks. *The Journal of Supercomputing*, 74(10):5156–5170, 2018.

[MMD13] Constantin Musca, Emma Mirica, and Razvan Deaconescu. Detecting and analyzing zero-day attacks using honeypots. In *2013 19th international conference on control systems and computer science*, pages 543–548. IEEE, 2013.

[Mom12] James A Momoh. *Smart grid: fundamentals of design and analysis*, volume 63. John Wiley & Sons, 2012.

[MXSJ17] Tasnuva Mahjabin, Yang Xiao, Guang Sun, and Wangdong Jiang. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, 13(12):1550147717741463, 2017.

[NH10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[NIG$^+$17] Kseniya Yu Nikolskaya, Sergey A Ivanov, Valentin A Golodov, Aleksey V Minbaleev, and Gregory D Asyaev. Review of modern ddos-attacks, methods and means of counteraction. In *2017 International Conference" Quality Management, Transport and Information Security, Information Technologies"(IT&QM&IS)*, pages 87–89. IEEE, 2017.

[NMN$^+$14] Bruno Astuto A Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications surveys & tutorials*, 16(3):1617–1634, 2014.

[Now21] Security Now! The meris botnet, 2021.

[NSJ16] Quamar Niyaz, Weiqing Sun, and Ahmad Y Javaid. A deep learning based ddos detection system in software-defined networking (sdn). *arXiv preprint arXiv:1611.07400*, 2016.

[PG07] Fernando Pérez and Brian E Granger. Ipython: a system for interactive scientific computing. *Computing in science & engineering*, 9(3):21–29, 2007.

[PPC20] Huseyin Polat, Onur Polat, and Aydin Cetin. Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models. *Sustainability*, 12(3):1035, 2020.

[PSY$^+$15] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot: analysing the rise of iot compromises. In *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*, 2015.

[Reu21] Reuters. Russia's yandex says it repelled biggest DDoS attack in history, 2021.

[RPGB17] Bernadette M Randles, Irene V Pasquetto, Milena S Golshan, and Christine L Borgman. Using the jupyter notebook as a tool for open science: An empirical study. In *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*, pages 1–2. IEEE, 2017.

[RS20] Nagarathna Ravi and S Mercy Shalinie. Learning-driven detection and mitigation of ddos attack in iot via sdn-cloud architecture. *IEEE Internet of Things Journal*, 7(4):3559–3570, 2020.

[SDR19] Rochak Swami, Mayank Dave, and Virender Ranga. Software-defined networking-based ddos defense mechanisms. *ACM Computing Surveys (CSUR)*, 52(2):1–36, 2019.

[SG13] Seungwon Shin and Guofei Gu. Attacking software-defined networks: A first feasibility study. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 165–166, 2013.

[SKKS16] Munazza Shabbir, Muazzam A Khan, Umair Shafiq Khan, and Nazar A Saqib. Detection and prevention of distributed denial of service attacks in vanets. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 970–974. IEEE, 2016.

[SRP19] Mikail Mohammed Salim, Shailendra Rathore, and Jong Hyuk Park. Distributed denial of service attacks and its defenses in iot: a survey. *The Journal of Supercomputing*, pages 1–44, 2019.

[TDDL20] Bhagyashri Tushir, Yogesh Dalal, Behnam Dezfouli, and Yuhong Liu. A quantitative study of ddos and e-ddos attacks on wifi smart home devices. *IEEE Internet of Things Journal*, 2020.

[VJ19] Ruchi Vishwakarma and Ankit Kumar Jain. A honeypot with machine learning based detection framework for defending iot based botnet ddos attacks. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 1019–1024. IEEE, 2019.

[Wan15] Zhanyi Wang. The applications of deep learning on traffic identification. *BlackHat USA*, 24(11):1–10, 2015.

[WDL$^+$09] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of* *the 26th annual international conference on machine learning*, pages 1113–1120, 2009.

[WSK18] Meng Wang, Javier Santillan, and Fernando Kuipers. Thingpot: an interactive internet-of-things honeypot. *arXiv preprint arXiv:1807.04114*, 2018.

[WWL+18] Juan Wang, Ru Wen, Jiangqi Li, Fei Yan, Bo Zhao, and Fajiang Yu. Detecting and mitigating target link-flooding attacks using sdn. *IEEE Transactions on Dependable and Secure Computing*, 16(6):944–956, 2018.

[YGL+18] Yao Yu, Lei Guo, Ye Liu, Jian Zheng, and Yue Zong. An efficient sdn-based ddos attack detection and rapid response platform in vehicular networks. *IEEE access*, 6:44570–44579, 2018.

[YLL17] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. Deepdefense: identifying DDoS attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2017.

[YZY18] Da Yin, Lianming Zhang, and Kun Yang. A ddos attack detection and mitigation with software-defined internet of things framework. *IEEE Access*, 6:24694–24705, 2018.

[ZLW+18] Baojun Zhou, Jie Li, Jinsong Wu, Song Guo, Yu Gu, and Zhetao Li. Machine-learning-based online distributed denial-of-service attack detection using spark streaming. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.